Systémy IBM - iSeries

Správa systému Klastry

Verze 5, vydání 4

Systémy IBM - iSeries

Správa systému Klastry

Verze 5, vydání 4

Poznámka

Před použitím těchto informací a odpovídajícího produktu si přečtěte informace v části “Poznámky”, na stránce 149.

Sedmé vydání (únor 2006)

Toto vydání se týká verze 5, vydání 4, modifikace 0 produktu IBM i5/OS (číslo produktu 5722-SS1) a všech následujících vydání a modifikací, dokud nebude v nových vydáních uvedeno jinak. Toto vydání nefunguje na žádných modelech RISC (Reduced instruction set computer) ani na modelech CISC.

Obsah

Klastry

Klastry umožňují účinně seskupovat servery iSeries a vytvářet prostředí zajišťující téměř stoprocentní dostupnost důležitých aplikací, zařízení a dat.

Klastry také poskytují zjednodušenou správu systému a vyšší výkonovou přizpůsobivost, neboť s růstem vašeho podniku je možno do systému hladce přidávat nové komponenty.

Používáním příkladů kódu vyjadřujete svůj souhlas s podmínkami uvedenými v části Licence na kód a vyloučení záruky.

Novinky ve verzi V5R4

Toto téma uvádí, co je v tomto vydání nového.

Podpora administrativní domény klastru

Administrativní doména klastru monitoruje a synchronizuje změny na vybraných prostředcích v klastru. Administrativní doména klastru nabízí snadnou správu a synchronizaci atributů pro prostředky, které jsou sdíleny v klastru, například proměnných prostředí a uživatelských profilů. Další informace o administrativní doméně klastru naleznete v těchto tématech:

- "Administrativní doména klastru" na stránce 8
- "Plánování administrativní domény klastru" na stránce 96
- "Kontrolní seznam domény pro administraci klastrů" na stránce 96
- "Vytvoření administrativní domény klastru" na stránce 107

Podpora peer skupiny klastrových prostředků (CRG)

Všechna rozhraní CRG byla vylepšena tak, aby podporovala peer skupinu klastrových prostředků (CRG). *Peer skupina klastrových prostředků (CRG)* je nepřepínatelná skupina CRG, v níž každý uzel v doméně obnovy má stejnou úlohu při obnově prostředků souvisejících s peer skupinou CRG. Další informace naleznete v následujících tématech:

- Skupina klastrových prostředků
- "Vytvoření skupiny CRG" na stránce 102
- "Spuštění CRG" na stránce 103

Zdokonalení klastru

Toto vydání bylo zdokonaleno v oblasti operace vypínání a rovněž odstraňování problémů v rámci klastrovaného prostředí. Mezi tato zdokonalení patří:

- Systematický přístup k ukončení klastrování v klastrovém uzlu, když se ukončují všechny aktivní podsystémy nebo když se ukončuje nebo je vypínán systém. Další informace najdete v tématu "Jak systémová událost ovlivňuje klastr" na stránce 106.
- Schopnost konfigurovat novou aplikační CRG s aktivní IP adresou převzetí. Další informace naleznete v "Vytvoření aplikační CRG s aktivní IP adresou převzetí" na stránce 103.
- Schopnost odstraňovat problémy s klastry zobrazením celého klastru a jeho přidružených skupin CRG z aktivního uzlu. Další informace najdete v tématu "Shromáždění informací pro obnovu klastru" na stránce 122.
- Byly přidány nové informace o ladicích nástrojích a výsledcích, které generují. Tyto nástroje a jejich výsledky můžete použít k řešení problémů s klastrem. Další informace najdete v těchto tématech:
  - "Prověření problému prostřednictvím příkazu DMPCLUTRC (Výpis trasování klastru)" na stránce 122

– “Prověření problému s makrem CLUSTERINFO” na stránce 126

Jak zjistit, co je nového nebo co se změnilo

Informace, ve kterých byly provedeny technické změny, používají:

- Symbol » označuje začátek nových nebo změněných informací.
- Symbol « označuje konec nových nebo změněných informací.

Více informací o tom, co je nového a co se změnilo najdete v tématu Sdělení pro uživatele.

Tisk PDF

Zde naleznete informace popisující, jak lze prohlížet a tisknout tento soubor PDF.

Chcete-li si prohlížet nebo stáhnout tento dokument ve formátu PDF, vyberte téma Klastry (asi 938 KB).

Červené knihy

- Clustering and IASPs for Higher Availability (asi 6,4 MB) Tato kniha redbook uvádí přehled technologie klastrů a přepínaných disků, která je k dispozici pro servery iSeries.
- iSeries Independent ASPs: A Guide to Moving Applications to IASPs (asi 3,4 MB) Tato kniha redbook uvádí podrobné postupy pro nezávislé ASP na serverech iSeries.
- Roadmap to Availability on the iSeries 400 (asi 626 KB) Tento dokument redpaper uvádí podrobný postup pro nezávislé ASP na serverech iSeries.

Webové stránky

- High Availability and Clusters (www.ibm.com/servers/eserver/iseries/ha)
  Webové stránky IBM obsahující informace o vysoké dostupnosti a klastrech.

Jak ukládat soubory ve formátu PDF

Chcete-li uložit soubor PDF na pracovní stanici za účelem zobrazení nebo tisku:

1. V prohlížeči klepněte pravým tlačítkem myši na požadované PDF (nebo přímo na některý z výše uvedených odkazů).
2. Pokud používáte program Internet Explorer, klepněte na **Uložit cíl jako...**. Pokud používáte program Netscape Communicator, klepněte na **Save Link As**.
3. Vyhledejte adresář, kam chcete dokument ve formátu PDF uložit.
4. Klepněte na **Save** (Uložit).

Jak stáhnout produkt Adobe Reader

Program Adobe Acrobat Reader potřebujete k prohlížení a tisku souborů PDF. Jeho kopii si můžete stáhnout z webových stránek společnosti Adobe (www.adobe.com/products/acrobat/readstep.html) .

Koncepce klastrů

Pochopte důkladně, jak klastry fungují. Přečtěte si o výhodách a významu klastrů, o důležitých principech klastrů a jejich vzájemných souvislostech.

Klastr iSeries je souborem či skupinou jednoho nebo více systémů nebo logických částí, které pracují společně jako jediný systém. Systémy v klastru, nazývané jako klastrové uzly, pracují ve spolupráci, a poskytují tak jednotné

výpočetní prostředí. Klastry iSeries podporují až 128 uzlů v jednom klastru. To vám umožňuje systémy iSeries efektivně seskupit a vytvořit prostředí, které zajistí vašim kritickým aplikacím a datům dostupnost dosahující téměř 100%. Přispěje to k tomu, aby vaše kritické systémy a aplikace byly dostupné 24 hodin denně, sedm dnů v týdnu. Klastry také poskytují zjednodušenou správu systému a vyšší výkonovou přizpůsobivost, neboť s růstem vašeho podniku je možno do systému hladce přidávat nové komponenty.

Přínosy klastrů

Vyžaduje-li vaše podnikání, aby byl zajištěn provoz systémů 24 hodin denně, 7 dnů v týdnu, právě klastry se nabízejí jako řešení.

Pomocí klastrování můžete značně snížit počet a délku trvání neplánovaných výpadků, což umožní, aby systémy, data a aplikace byly nepřetržitě dostupné.

Hlavní přínosy, které mohou klastry přinést vašemu podnikání, jsou tyto:

Nepřetržitá dostupnost
Klastry zajišťují, aby vaše systémy, data a aplikace byly neustále dostupné.

Zjednodušená administrace
Skupinu systémů můžete řídit jako jeden systém nebo jednu databázi, aniž by bylo nutno se k jednotlivým systémům přihlašovat. Můžete použít administrativní domény klastrů pro snazší správu prostředků, které jsou sdíleny v klastru.

Zvýšená výkonová přizpůsobivost
Do systému lze hladce přidávat nové komponenty v souladu s tím, jak to vyžaduje růst vašeho podniku.

Související pojmy

“Přepnutí při selhání” na stránce 17
K *přepnutí při selhání* dochází, když se server v klastru automaticky přepne na jeden nebo více záložních serverů v případě selhání systému.

Související úlohy

“Přepnutí” na stránce 20
K *přepnutí* dojde, když ručně přepnete přístup k prostředku z jednoho serveru na jiný.

Jak klastr funguje

Klastrovaná infrastruktura, ketrá je poskytována jako součást systému i5/OS a je nazývána jako služby klastrových prostředků, zajišťuje odolnost vašich klíčových prostředků. Mohou to být data, aplikace, zařízení a jiné prostředky, k nimž přistupuje více klientů.

Jestliže dojde k výpadku nebo ztrátě nějakého serveru, mohou ostatní systémy, které lze definovat v klastru, přistupovat k funkcím jiného systému v rámci klastru. K dispozici jsou dva modely přístupu k takovým datům: Model primární-záložní a peer model. Podrobné informace o skupinách klastrových prostředků (CRG), které můžete vytvořit na základě těchto modelů, najdete v tématu Skupina klastrových prostředků.

Související pojmy

“Přepnutí při selhání” na stránce 17
K *přepnutí při selhání* dochází, když se server v klastru automaticky přepne na jeden nebo více záložních serverů v případě selhání systému.

“Replikace” na stránce 25
Replikace vytváří kopii něčeho v reálném čase. Představuje kopírování objektů z jednoho klastrového uzlu do jednoho nebo více dalších uzlů klastru.

“Odolná zařízení” na stránce 15
Odolná zařízení jsou fyzické prostředky reprezentované konfiguračními objekty, jako je popis zařízení, které jsou přístupné z více než jednoho klastrového uzlu.

“Odolná data” na stránce 15
Odolná data jsou data, která jsou replikována (kopírována) alespoň na jeden další klastrový uzel.

“Opětovné připojení” na stránce 20
Opětovné připojení znamená, že nezúčastněný člen se stane aktivním členem klastru.

“Porovnání logické replikace, přepínání disků a zrcadlení mezi stanovišti” na stránce 86
Toto téma uvádí přehled různých technologií pro odolnost dat, které lze použít spolu s klastry ke zdokonalení v oblasti vysoké dostupnosti.

Související úlohy

“Přepnutí” na stránce 20
K *přepnutí* dojde, když ručně přepnete přístup k prostředku z jednoho serveru na jiný.

Základy klastrů

Zde jsou vysvětleny základní pojmy a koncepce v oblasti klastrování, které musíte pochopit, než začnete s vytvářením klastrů a přizpůsobováním klastrů svým potřebám.

S klastrem souvisejí dvě základní koncepce: Klastrové uzly a skupina prostředků klastru. *Klastrový uzel* je systém iSeries nebo logická část, která je členem klastru. Když vytváříte klastr, specifikujete systémy nebo části, které chcete zahrnout do klastru jako uzly. *Skupina klastrových prostředků (CRG)* slouží jako řídicí objekt při sdružování odolných prostředků. Skupina CRG může obsahovat podmnožinu nebo všechny uzly v rámci klastru. Klastr iSeries podporuje čtyři typy skupin CRG: aplikační, datová, zařízení a peer. V těchto typech skupin CRG jsou dva společné prvky: Doména obnovy a ukončovací program.

Doména obnovy definuje úlohu každého uzlu ve skupině CRG. Když vytváříte v klastru skupinu klastrových prostředků (CRG), vytvoří se objekt CRG ve všech uzlech, které jsou specifikovány jako součásti domény obnovy. Avšak systémový obraz objektu CRG, na který lze přistupovat z kteréhokoliv aktivního uzlu v doméně obnovy CRG, je pouze jeden. To znamená, že veškeré změny provedené u CRG se provedou ve všech uzlech dané domény obnovy.

Ukončovací program je volán při událostech souvisejících s klastrem pro skupinu CRG. Jednou takovou událostí je přesouvání bodu přístupu z jednoho uzlu na jiný.

Existují dva modely pro skupiny CRG, které lze vytvořit v klastru: model primární-záložní a peer model. V modelu primární-záložní lze uzly v doméně obnovy skupiny CRG definovat následovně:

- *Primární uzel* je klastrový uzel, který je primárním přístupovým bodem pro odolný klastrový prostředek.
- *Záložní uzel* je klastrový uzel, který přebírá roli primárního přístupového bodu v případě, že dojde k selhání současného primárního uzlu nebo je provedeno manuální přepnutí.
- *Replikační uzel* je klastrový uzel, který má kopie klastrových prostředků, ale není schopen převzít roli primárního nebo záložního uzlu.

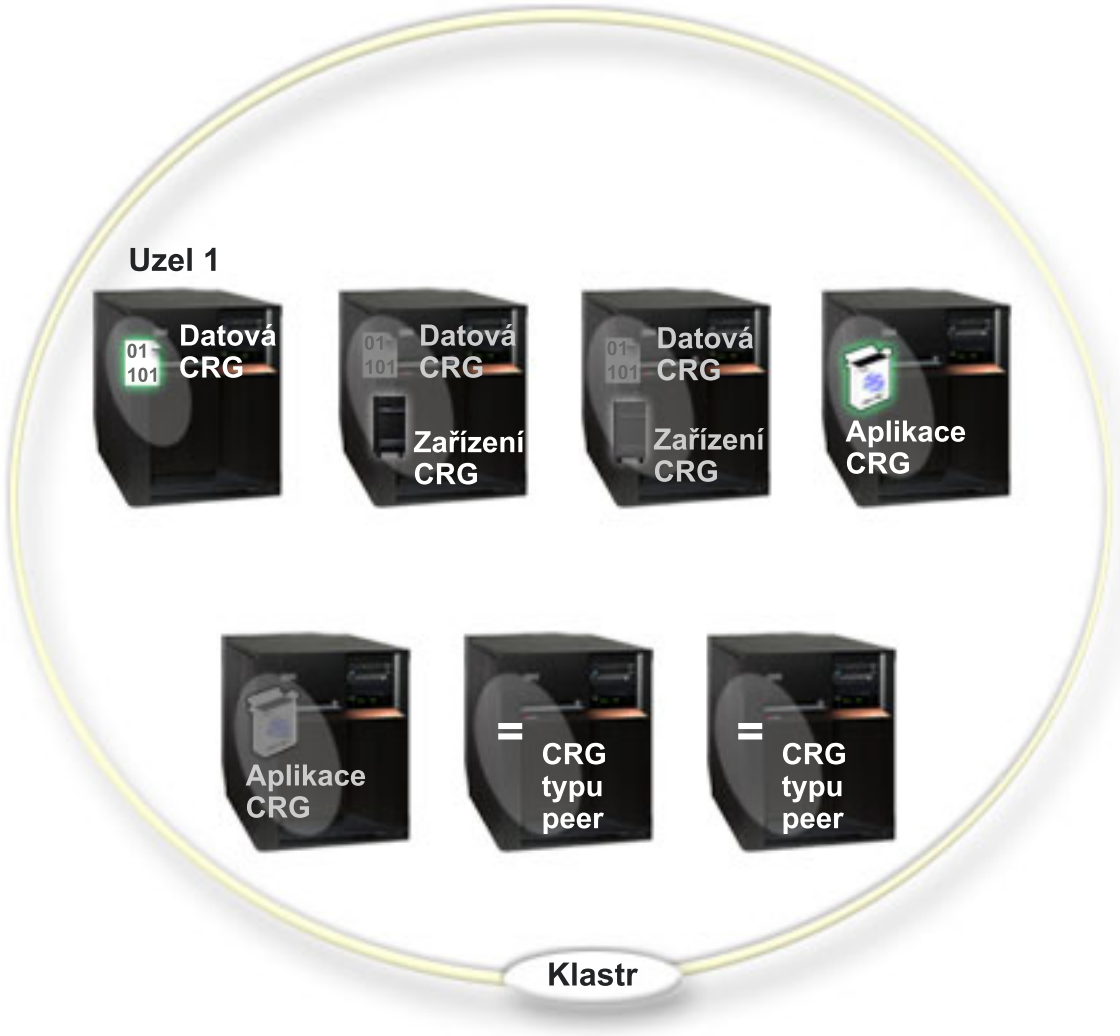
Doména obnovy peer CRG v peer modelu definuje peer vztah (tj. rovnocenný vztah) mezi uzly. Uzly v doméně obnovy peer CRG lze definovat následovně:

- *Peer uzel* je klastrový uzel, který může být aktivním bodem přístupu k prostředkům klastru.
- *Replikační uzel* je klastrový uzel, který má kopie klastrových prostředků. Uzly definované jako replikované v peer CRG představují neaktivní body přístupu k prostředkům klastru.

Uzly v doméně obnovy v peer CRG jsou rovnocenné, pokud jde o úlohu, jakou uzly hrají při obnově. Jelikož každý uzel v této peer CRG má v podstatě stejnou úlohu, koncepce přepnutí při selhání a přepnutí neplatí. Uzly mají peer vztah, a když jeden z uzlů selže, jiné peer uzly budou pokračovat v činnosti.

Můžete také vytvořit administrativní doménu klastru, která bude reprezentována pomocí peer CRG. Uzly v administrativní doméně klastrů jsou všechny peer uzly v doméně obnovy skupiny CRG. Neexistují tam žádné replikační uzly.

V níže uvedeném příkladu se objevují všechny tři typy CRG:



Datová CRG

Datová CRG se nachází v uzlu 1, uzlu 2 a uzlu 3. To znamená, že doména obnovy pro CRG dat zadala roli pro uzel 1 (primární uzel), uzel 2 (první záložní uzel) a uzel 3 (druhý záložní uzel). V tomto příkladu slouží uzel 1 aktuálně jako primární přístupový bod. Uzel 2 je definovaný v doméně obnovy jako první záložní uzel. To znamená, že uzel 2 obsahuje kopii prostředku, která se udržuje aktuální prostřednictvím logické replikace. Pokud by došlo k přepnutí při selhání nebo přepnutí, uzel 2 převezme funkci primárního přístupového bodu.

Aplikační CRG

Aplikační CRG se nachází v uzlu 4 a uzlu 5. To znamená, že doména obnovy pro aplikační CRG zadala uzel 4 a uzel 5. V tomto příkladě slouží aktuálně jako primární přístupový bod uzel 4. Pokud by došlo k přepnutí při selhání nebo přepnutí, převezme funkci primárního přístupového bodu aplikace uzel 5. Vyžaduje IP adresu převzetí.

Peer CRG

Peer CRG je přítomna v uzlu 6 a uzlu 7. To znamená, že doména obnovy pro peer CRG má specifikovaný uzel 6 a uzel 7. Uzel 6 a 7 v tomto příkladu mohou být peer nebo replikované. Pokud peer CRG reprezentuje administrativní doménu klastru, prostředky, které jsou monitorovány administrativní doménou klastru, budou mít všechny změny synchronizovány v doméně představované uzlem 6 a uzlem 7, nehledě na to, který uzel změnu vyvolal.

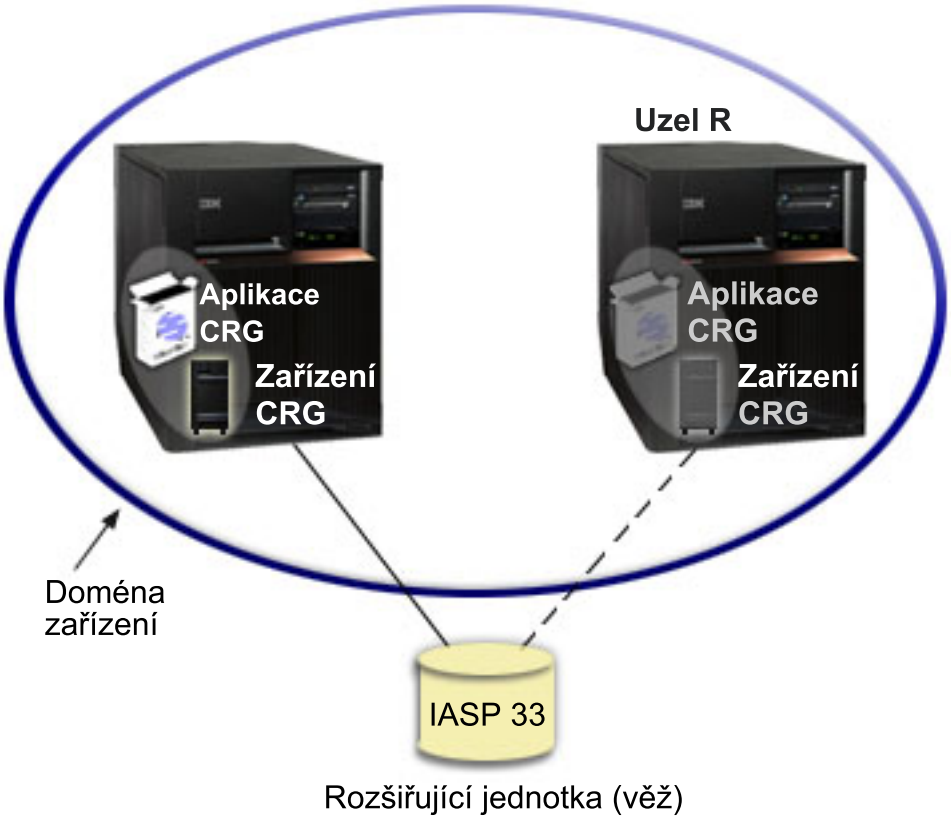
Zařízení CRG

CRG zařízení se nachází v uzlu 2 a uzlu 3. To znamená, že doména obnovy pro CRG zařízení zadala uzel 2 a

uzel 3. V tomto příkladě slouží aktuálně jako primární přístupový bod uzel 2. To znamená, že na odolné zařízení, které je součástí CRG zařízení, lze aktuálně přistupovat z uzlu 2. Pokud by došlo k přepnutí při selhání nebo přepnutí, převezme funkci přístupového bodu zařízení uzel 3.

CRG zařízení vyžaduje, aby bylo na externím zařízení, rozšiřující jednotce (věž) nebo procesoru IOP v logické části nakonfigurováno odolné zařízení zvané nezávislé ASP.

Uzly v doméně obnovy pro CRG zařízení musí být také členy stejné domény zařízení. Na níže uvedeném příkladu je znázorněno CRG zařízení, které má ve své doméně obnovy uzel L a uzel R. Oba uzly jsou zároveň členy téže domény zařízení.



Související pojmy

“Klastrový uzel” na stránce 7
Klastrový uzel je systém iSeries nebo logická část, která je členem klastru.

“Skupina klastrových prostředků” na stránce 7
Skupina klastrových prostředků (CRG) je systémový objekt operačního systému i5/OS tvořený sadou nebo skupinou prostředků klastru, které se používají pro správu událostí, k nimž dochází v klastrovaném prostředí. Skupina klastrových prostředků popisuje doménu obnovy a udává jméno ukončovacího programu skupiny klastrových prostředků, který je volán, když dojde k určitým událostem na klastru.

“Doména obnovy” na stránce 11
Doména obnovy je podmnožina klastrových uzlů, které jsou s určitým záměrem, například kvůli akci obnovy nebo synchronizaci událostí, seskupeny do skupiny klastrových prostředků (CRG).

“Ukončovací programy skupiny klastrových prostředků” na stránce 10
Ukončovací program skupiny klastrových prostředků je volán po události související s klastrem pro CRG.

Nezávislá ASP

"Domény zařízení" na stránce 16
Doména zařízení je podmnožina uzlů v klastru, které sdílejí prostředky zařízení. Konkrétněji: uzly v doméně zařízení se mohou účastnit na akci přepínání pro určitý soubor prostředků odolného zařízení.

Základní prvky klastru

Klastr iSeries je souborem či skupinou jednoho nebo více systémů nebo logických částí, které pracují společně jako jediný systém. Na základě této informace porozumíte jednotlivým prvkům a jejich vzájemnému vztahu.

Klastrový uzel

Klastrový uzel je systém iSeries nebo logická část, která je členem klastru.

Každý klastrový uzel je označen pomocí jména klastrového uzlu, které má 8 znaků a je asociováno s jednou nebo více IP adresami, jež reprezentují server iSeries. Při konfigurování klastru můžete pro uzel v klastru použít libovolné jméno. Doporučujeme však, aby se jméno uzlu shodovalo s hostitelským jménem nebo jménem systému.

Při zajištění komunikační cesty mezi službami klastru na jednotlivých uzlech klastru používá klastrová komunikace skupinu vrstev protokolu TCP/IP. Pro soubor uzlů klastru, které jsou konfigurovány jako součásti klastru, se používá název seznam členů klastru.

Skupina klastrových prostředků

Skupina klastrových prostředků (CRG) je systémový objekt operačního systému i5/OS tvořený sadou nebo skupinou prostředků klastru, které se používají pro správu událostí, k nimž dochází v klastrovaném prostředí. Skupina klastrových prostředků popisuje doménu obnovy a udává jméno ukončovacího programu skupiny klastrových prostředků, který je volán, když dojde k určitým událostem na klastru.

Klastry umožňují dvě volby pro definování vztahů mezi uzly v klastru: Model primární-záložní a peer model. Každý z těchto modelů lze používat spolu s druhým modelem nebo samostatně, což závisí na potřebách vašeho prostředí.

Model s primární zálohou

Všechny skupiny klastrových prostředků této kategorie definují v doméně obnovy uzly se specifickými úlohami: primární, záložní nebo replikovaný. Primární a záložní uzly jsou dostupné body přístupu ke klastrovým prostředkům. Avšak pouze jeden uzel bude aktivním přístupovým bodem v jeden časový okamžik. Tomuto uzlu se říká primární. Replikované uzly nejsou k dispozici jako body přístupu. To lze změnit tím, že replikovaným uzlům přiřadíte úlohu záložních uzlů. Skupiny klastrových prostředků v modelu primární-záložní se definují jako odolná data, odolné aplikace nebo odolná zařízení. Odolnost dat umožňuje udržovat více kopií dat na více než jednom uzlu a přesunovat přístupový bod na záložní uzel. Odolnost aplikace umožňuje restartovat program aplikace buď na stejném uzlu, nebo v jiném uzlu v klastru. Odolnost zařízení umožňuje, aby se zařízení přesunulo (přepnulo) na záložní uzel.

Každá CRG aplikací nebo dat je asociovaná s ukončovacím programem CRG. Pro CRG odolných zařízení je ukončovací program volitelný.

V prostředí produktu iSeries Navigator se skupiny klastrových prostředků (CRG) nazývají odlišně:

- CRG zařízení je označována jako **přepínatelné zařízení**.
- Aplikační CRG je označována jako **přepínatelná aplikace**.
- CRG dat se nazývá **skupina přepínatelných dat**.

Peer model

Všechny skupiny klastrových prostředků této kategorie definují v doméně obnovy uzly s úlohou peer nebo úlohou replikované. Peer uzly jsou k dispozici jako body přístupu ke skupině klastrových prostředků. Všechny uzly definované jako peer budou přístupovým bodem, když je skupina klastrových prostředků spuštěna. Replikované uzly nejsou k dispozici jako body přístupu. To lze změnit tím, že replikovaným uzlům přiřadíte úlohu peer uzlů. V peer CRG obsahuje každý uzel replikovaná data, která existují na každém z uzlů. Když v peer CRG selže uzel, bod selhání je sdělen jiným uzlům v klastru a tyto uzly pokračují v činnosti od bodu selhání.

Administrativní doména klastru je reprezentována peer skupinou CRG s doménou obnovy tvořenou pouze peer uzly.

Související pojmy

"Doména obnovy" na stránce 11
Doména obnovy je podmnožina klastrových uzlů, které jsou s určitým záměrem, například kvůli akci obnovy nebo synchronizaci událostí, seskupeny do skupiny klastrových prostředků (CRG).

"Ukončovací programy skupiny klastrových prostředků" na stránce 10
Ukončovací program skupiny klastrových prostředků je volán po události související s klastrem pro CRG.

Řízení zpracování skupin klastrových prostředků

Když se vyskytne selhání uzlu, dojde k přepnutí při selhání. Při sekvenčním přepínání skupin CRG při selhání se nejprve přepínají CRG zařízení, pak datové CRG a nakonec aplikační CRG. U peer CRG není žádné sekvenční pořadí, ale každý uzel je označen, jakmile dojde k selhání.

Pomocí kontroly stavu CRG si můžete ověřit, zda skupina klastrových prostředků dokončila přepnutí při selhání či přepnutí.

Dále můžete pomocí blokování aplikaci pozastavit, dokud data ke zpracování nebudou dostupná. Zatímco se zpracovávají datově odolné skupiny klastrových prostředků, můžete blokovat přístup k datům reprezentovaným datovou CRG. Přístup lze blokovat pomocí rozhraní QxdaBlockEDRS (Block EDRS Access) API a QxdaCheckEDRSBlock (Check EDRS Block Status) API. Kdyby došlo k přepnutí při selhání nebo přepnutí, můžete pomocí těchto rozhraní API zablokovat a odblokovat přístup z ukončovacího programu skupiny klastrových prostředků.

Související pojmy

"Přepnutí při selhání" na stránce 17
K *přepnutí při selhání* dochází, když se server v klastru automaticky přepne na jeden nebo více záložních serverů v případě selhání systému.

"Doména obnovy" na stránce 11
Doména obnovy je podmnožina klastrových uzlů, které jsou s určitým záměrem, například kvůli akci obnovy nebo synchronizaci událostí, seskupeny do skupiny klastrových prostředků (CRG).

"Ukončovací programy skupiny klastrových prostředků" na stránce 10
Ukončovací program skupiny klastrových prostředků je volán po události související s klastrem pro CRG.

Související úlohy

"Přepnutí" na stránce 20
K *přepnutí* dojde, když ručně přepnete přístup k prostředku z jednoho serveru na jiný.

Administrativní doména klastru

Administrativní doména klastru se používá ke správě prostředků, které je nutné udržovat konzistentní ve všech uzlech v klastrovaném prostředí.

Mohou existovat určité provozní nebo konfigurační parametry, které musejí být definovány na každém uzlu, který je přístupovým bodem pro odolná data, aplikace a zařízení. Jestliže je provedena změna jednoho z těchto parametrů na jakémkoli uzlu, který může být přístupovým bodem k něčemu, je nutné tuto změnu rozšířit na všechny ostatní uzly, které mohou být potenciálním přístupovým bodem. Administrativní doména klastru poskytuje schopnost identifikovat prostředky, které je nutné udržovat konzistentní ve všech uzlech v doméně. Tato doména pak monitoruje změny na těchto prostředcích a synchronizuje je v rámci aktivní domény. Administrativní doména klastru je představována pomocí peer CRG. Když vytvoříte administrativní doménu klastru, systém vytvoří peer CRG. Jméno administrativní domény klastru se stane jménem peer CRG. Uzly, které tvoří administrativní doménu klastru, jsou definovány doménou obnovy peer CRG. Všechny uzly jsou peer uzly. Replikační uzly v administrativní doméně klastru nejsou povoleny. Klastrový uzel může být definován pouze v jedné administrativní doméně klastru v rámci klastru. Další informace o souvisejících úlohách pro administrativní doménu klastru naleznete v následujících tématech:

1. "Plánování administrativní domény klastru" na stránce 96
2. "Kontrolní seznam domény pro administraci klastrů" na stránce 96
3. "Vytvoření administrativní domény klastru" na stránce 107

4. “Přidání záznamů o monitorovaných prostředcích” na stránce 107
5. “Spuštění CRG” na stránce 103

Jakmile je administrativní doména klastru vytvořena, normální funkce CRG se použijí ke správě administrativní domény klastru. Jestliže například chcete přidat uzel do administrativní domény, musíte přidat uzel do domény obnovy skupiny CRG s úlohou uzlu peer. Když budete chtít spustit administrativní doménu klastru, spustíte peer CRG.

Spuštěním a ukončením skupiny CRG lze řídit proces synchronizačních změn. Když se skupina CRG ukončí, změny provedené na monitorovaném prostředku na libovolném uzlu v doméně nebudou rozšířeny na zbytek domény. Jakmile bude skupina CRG spuštěna, změny provedené na jakémkoli monitorovaném prostředky, když skupina nebyla aktivní, budou přeneseny na zbytek domény. Když je skupina CRG aktivní, změny provedené na jakémkoli monitorovaném prostředku na jakémkoli aktivním uzlu budou přenášeny dynamicky, takže prostředek zůstane konzistentní v celé administrativní doméně. Další informace najdete v tématu “Monitorování administrativní domény klastru” na stránce 108.

Chcete-li přidat uzel do administrativní domény klastru, musíte přidat klastrový uzel do domény obnovy peer CRG. Když se do domény přidá uzel, na novém uzlu se vytvoří všechny spravované prostředky (za předpokladu, že neexistují) a budou se synchronizovat se zbytkem administrativní domény.

Při výmazu administrativní domény klastru se odstraní všechny prostředky definované v administrativní doméně klastru, a to z každého uzlu v doméně; avšak samotný prostředek nebude ze systému odstraněn. Další informace najdete v tématu Monitorované prostředky.

Monitorované prostředky

Monitorované prostředky jsou typy systémových prostředků, které lze spravovat pomocí administrativní domény klastru. Tyto prostředky jsou reprezentovány v administrativní doméně klastru jako *záznamy monitorovaných prostředků (MRE)*.

Prostředky, které jsou synchronizovány pomocí administrativní domény klastrů, jsou představovány záznamy monitorovaných prostředků (MRE). Jakmile je záznam MRE přidán do administrativní domény klastru, změny provedené na prostředku na libovolném uzlu v administrativní doméně klastru se rozšíří na všechny uzly v aktivní doméně. Pro správu záznamů MRE v administrativní doméně klastru můžete použít tři rozhraní Integrated Operating Environments API:

- Rozhraní QfpadAddMonitoredResourceEntry (Add Monitored Resource Entry) API
- Rozhraní QfpadRmvMonitoredResourceEntry (Remove Monitored Resource Entry) API
- Rozhraní QfpadRtvMonitoredResourceInfo (Retrieve Monitored Resource Information) API

Záznam monitorovaného prostředku lze přidat do administrativní domény klastru pro následující typy prostředků:

- Systémové hodnoty
- Uživatelské profily
- Popisy úloh
- Třída
- Popis zařízení nezávislých ASP
- Atributy sítě
- Systémové proměnné prostředí
- Atributy TCP/IP

Záznam MRE lze do administrativní domény klastru přidat pouze tehdy, když jsou veškeré uzly v doméně a jsou účastníky skupiny. Záznam MRE nelze přidat, jestliže je administrativní doména klastru rozdělena na části. Jakmile je záznam MRE přidán, změny v prostředku, který je reprezentován záznamem MRE, se rozšíří na všechny aktivní uzly v doméně, když se spustí peer CRG. Jestliže je skupina CRG ukončena, nevyřízené změny se rozšíří na aktivní doménu, jakmile se CRG znovu spustí.

K záznamu MRE je přidružen globální stav. Když má prostředek reprezentovaný záznamem MRE stejné hodnoty pro všechny atributy, které se monitorují ve všech uzlech v aktivní doméně, globální stav pro tento prostředek je konzistentní. Jestliže se administrativní doména klastru pokusí aktualizovat prostředek na jednom nebo více uzlech a aktualizace selže, globální stav prostředku bude nekonzistentní. Když je globální stav prostředku nekonzistentní, administrátor musí zjistit příčinu selhání a odstranit ji. Administrativní doména klastru se pokusí znovu synchronizovat prostředek při jeho další aktualizaci pravděpodobně tehdy, když administrátor změní prostředek ve snaze odstranit problém, který způsobil selhání aktualizace, nebo když se skupina CRG restartuje.

Když je administrativní doména klastru CRG ukončena, globální stav pro všechny záznamy MRE se nastaví na nekonzistentní. Důvodem je to, že při ukončování CRG lze změny provádět na monitorovaných prostředcích na různých uzlech, a proto se stávají nekonzistentní.

Jestliže prostředek představovaný záznamem MRE je systémový objekt, neměli byste jej mazat, přejmenovávat ani přesouvat do jiné knihovny, aniž byste nejprve odstranili záznam MRE. Když prostředek vymažete, přejmenujete nebo přesunete do jiné knihovny, globální stav pro záznam MRE bude nekonzistentní a veškeré změny provedené pak na prostředku jakékokoli uzlu nebudou přeneseny na administrativní doménu klastru.

Když je uzel přidáván do administrativní domény klastru, všechny záznamy MRE a aktivní domény budou zkopírovány do nového uzlu. Veškeré prostředky představované záznamem MRE, které neexistují na novém uzlu, budou vytvořeny, a hodnoty atributů budou nastaveny na stejné hodnoty jako v administrativní doméně klastru.

Jestliže jsou v administrativní doméně klastrů uzly, které nejsou aktivní, veškeré změny prostředků, které jsou provedeny v aktivní doméně, budou rozšířeny na neaktivní uzly, jakmile se vrátí do aktivní domény.Jestliže je administrativní doména klastrů rozdělena na části, změny budou nadále synchronizovány mezi aktivními uzly v každé části. Až budou uzly opět sloučeny, administrativní doména klastrů rozšíří všechny změny provedené v každé části tak, aby prostředky v rámci aktivní domény byly konzistentní. Jestliže bude provedeno více změn v různých částech na stejném prostředku, administrativní doména klastru zpracuje po sloučení částí každou změnu, třebaže jejich pořadí nejisté.

Související pojmy

“Administrativní doména klastru” na stránce 8

Administrativní doména klastru se používá ke správě prostředků, které je nutné udržovat konzistentní ve všech uzlech v klastrovaném prostředí.

Ukončovací programy skupiny klastrových prostředků

Ukončovací program skupiny klastrových prostředků je volán po události související s klastrem pro CRG.

U CRG odolných zařízení je ukončovací program volitelný, u jiných typů CRG je však nutný. Použije-li se ukončovací program CRG, je volán na základě události v rámci klastru, konkrétně v těchto případech:

- Uzel neočekávaně opustí klastr.
- Uzel opustí klastr v důsledku použití rozhraní QcstEndClusterNode (End Cluster Node) API nebo QcstRemoveClusterNodeEntry (Remove Cluster Node Entry) API.
- Uzel je vymazán v důsledku použití rozhraní QcstDeleteCluster (Delete Cluster) API.
- Uzel je aktivován pomocí rozhraní QcstStartClusterNode (Start Cluster Node) API.
- Je znovu navázána komunikace s rozděleným uzlem.

Ukončovací programy programují nebo dodávají obchodní partneři IBM pro middleware klastrů, nebo poskytovatelé aplikačních programů využívajících klastry.

Podrobnější informace o ukončovacích programech skupin klastrových prostředků, včetně popisu informací, které se jim předávají pro jednotlivé kódy akcí, najdete v tématu Ukončovací programy skupiny klastrových prostředků v dokumentaci k API klastrů.

Doména obnovy

Doména obnovy je podmnožina klastrových uzlů, které jsou s určitým záměrem, například kvůli akci obnovy nebo synchronizaci událostí, seskupeny do skupiny klastrových prostředků (CRG).

Doména představuje ty klastrové uzly, za nichž lze přistupovat ke klastrovým prostředkům. Tato podmnožina klastrových uzlů, která je přiřazena určité skupině klastrových prostředků, podporuje primární, sekundární (záložní), replikační nebo peer bod přístupu.

Uzel může mít v doméně obnovy čtyři typy rolí:

Primární

Klastrový uzel, který je primárním bodem přístupu k odolnému klastrovému prostředku.

- U datové CRG obsahuje primární uzel zásadní kopii prostředku.
- U aplikační CRG je primárním uzlem systém, ve kterém je aplikace v současné době spuštěna.
- U CRG zařízení je primárním uzlem aktuální vlastník prostředku zařízení.

  Poznámka: Uzly v doméně obnovy CRG zařízení vyžadují při použití geografického zrcadlení jméno serveru a IP adresy datových portů. Podrobné informace najdete v tématu Jméno serveru a IP adresy datových portů.
- Pro peer CRG není primární uzel podporován.

Jestliže primární uzel pro CRG selže nebo bude iniciováno manuální přepnutí, primární bod přístupu pro tento CRG se přesune do prvního záložního uzlu.

Záložní

Klastrový uzel, který převezme roli primárního přístupového bodu v případě, že současný primární uzel selže nebo je vyvoláno ruční přepnutí.

- U CRG dat obsahuje tento klastrový uzel kopii prostředku, který je aktualizován tak, aby odpovídal replikačnímu uzlu.
- Pro peer CRG není záložní uzel podporován.

Replikační

Klastrový uzel, který obsahuje kopie klastrových prostředků, ale nemůže převzít roli primárního nebo záložního uzlu. Přepnutí při selhání nebo přepnutí na replikační uzel není dovoleno. Pokud byste přesto chtěli, aby se replikační uzel stal primárním, musíte nejprve změnit roli replikačního uzlu na záložní.

- Uzly definované jako replikované pro peer CRG představují neaktivní body přístupu k prostředkům klastru.

Peer Klastrový uzel, který není přiřazen a může být aktivním bodem přístupu k prostředkům klastru. Když se spouští CRG, všechny uzly definované jako peer budou aktivním bodem přístupu.

- Pro peer CRG je bod přístupu zcela ovládán spravující aplikací a ne systémem. Úloha peer je podporována pouze pomocí peer CRG.

Model s primární zálohou

Každý z uzlů, které jsou v modelu s primární zálohou, má v doméně obnovy roli vztahující se k aktuálnímu provoznímu prostředí klastru. Tato role se nazývá *aktuální role* uzlu v doméně obnovy. Aktuální role klastru se mění v souvislosti s tím, jak klastr prochází provozními změnami, například ukončování uzlů, spouštění uzlů nebo selhávání uzlů. Každý uzel v doméně obnovy má také roli vztahující se k preferovanému nebo ideálnímu prostředí klastru. Tato role se nazývá *preferovaná role* uzlu v doméně obnovy. Preferovaná role je nastavena staticky na počátku při vytvoření skupiny klastrových prostředků. Při změnách prostředí klastru se tato role nemění. Preferovaná role se změní pouze při přidání uzlů do domény obnovy, odebrání uzlů z domény obnovy nebo při odebrání uzlu z klastru. Preferované role lze také změnit ručně.

Principiálně lze pohlížet na doménu obnovy v modelu s primární zálohou takto:

Tabulka 1. Úlohy uzlů pro CRG s primární zálohou

Uzel	Aktuální role	Preferovaná role
A	1. záložní	Primární
B	2. záložní	1. záložní
C	Primární	2. záložní
D	Replikační	Replikační

Uzly A, B, C a D v tomto příkladu tvoří skupinu CRG, která je uspořádána v modelu primární-záložní. Uzel C slouží jako aktuální primární uzel. Protože má uzel C preferovanou roli druhého záložního uzlu, musí být jeho aktuální role primárního uzlu výsledkem dvou akcí přepnutí při selhání nebo přepnutí. Při první akci přepnutí při selhání nebo přepnutí přešla primární role z uzlu A na uzel B, protože uzel B je definován jako první záložní uzel. Při druhém přepnutí při selhání nebo přepnutí se primárním uzlem stal uzel C, protože je definován jako druhý záložní uzel. Aktuální a upřednostňovaná úloha uzlu D je replikovat. Replikační uzel se nemůže stát bodem přístupu při přepnutí při selhání ani při přepnutí, pokud je jeho úloha manuálně změněna na primární nebo záložní.

Poznámka: Roli každého uzlu v doméně obnovy lze změnit také ručně. Příklad ukazuje, jak se mění role v doméně obnovy, jestliže nastanou akce přepnutí nebo přepnutí při selhání a v doméně obnovy nejsou provedeny žádné ruční změny určení rolí.

Peer model

V peer modelech může uzel ve skupinách CRG mít jednu ze dvou úloh: peer nebo replikační.

Tabulka 2. Úlohy uzlů pro peer skupiny CRG

Uzel	Aktuální role	Preferovaná role
A	Peer	Peer
B	Peer	Peer
C	Peer	Peer
D	Replikační	Replikační

Uzly A, B a C jsou definovány v doméně obnovy jako peer uzly. Když dojde k selhání v uzlu A, je to sděleno všem uzlům v doméně obnovy nehledě na jejich aktuální úlohu. Tyto uzly pokračují v činnosti od okamžiku, kdy uzel A selhal. Uzel D obsahuje data, ale nemůže v činnosti pokračovat, protože je definován jako replikační.

Libovolný počet uzlů lze definovat jako peer nebo replikační. Peer uzly nejsou uspořádány a mohou se stát aktivními body přístupu k prostředkům klastru. Replikační uzly nejsou uspořádány a nemohou se stát aktivními body přístupu k prostředkům klastru, pokud uživatel pomocí rozhraní QcstChangeClusterResourceGroup (Change Cluster Resource Group) API nezmění jejich úlohu z replikační na peer.

Související úlohy

“Změna domény obnovy pro skupinu klastrových prostředků” na stránce 103
U skupiny klastrových prostředků je možné měnit role uzlů v doméně obnovy, do domény obnovy lze přidávat uzly a odebírat je z ní. U skupiny klastrových prostředků můžete také změnit jméno serveru (stanoviště) a IP adresy datových portů uzlů v doméně obnovy.

“Provedení přepnutí” na stránce 104
Provedení ručního přepnutí způsobí, že se aktuální primární uzel přepne na záložní uzel, který je definován v doméně obnovy skupiny klastrových prostředků.

Klastrová verze

Klastrová verze představuje úroveň funkce dostupné v klastru.

Určení verze je technika, která umožní klastru obsahovat systémy s více úrovněmi vydání a provádět úplné řízení pomocí určení úrovně komunikačního protokolu, který se má použít. Pokud používáte klastr, který bude obsahovat systémy s různými úrovněmi vydání, přečtěte si téma Klastry s více vydáními.

Ve skutečnosti existují dvě klastrové verze:

Potenciální klastrová verze
Představuje nejpokročilejší úroveň klastrové funkce dostupné pro daný uzel. Je to verze, ve které je uzel schopen komunikovat s ostatními klastrovými uzly.

Aktuální klastrová verze
Představuje verzi aktuálně používanou pro všechny klastrové operace. Je to verze komunikací mezi uzly v klastru.

Potenciální klastrová verze je přidávána ke každému vydání operačního systému, který má důležité nové funkční vybavení pro klastrování, které není v dřívějších klastrových verzích k dispozici. Pokud je aktuální klastrová verze nižší než potenciální klastrová verze, nemůže být funkce použita, protože některé uzly nebudou schopny rozeznat nebo zpracovat požadavek. Aby bylo možné využít tuto novou funkci, musí být každý systém v klastru ve stejné potenciální klastrové verzi a aktuální klastrová verze musí být rovněž nastavena na tuto úroveň.

Když se uzel pokouší připojit ke klastru, je jeho potenciální klastrová verze porovnána s aktuální klastrovou verzí. Není-li hodnota potenciální klastrové verze stejná jako aktuální verze (N) nebo není-li stejná jako úroveň další verze (N+1), není uzlu dovoleno, aby se připojil ke klastru. Všimněte si, že aktuální klastrová verze je zpočátku nastavena prvním uzlem definovaným v klastru pomocí hodnoty zadané v API nebo příkazu pro vytvoření klastru.

Když například chcete, aby uzly V5R3 existovaly s uzly V5R4, můžete provést jednu z následujících možností:

- Vytvořte klastr na systému V5R3 a přidejte uzel V5R4.
- Vytvořte klastr na systému V5R4 a umožněte, aby mohly být ke klastru přidány předchozí uzly. Potom přidejte server V5R3 ke klastru.

V klastrech s více vydáními budou klastrové protokoly vždy spouštěny na nejnižší úrovní vydání uzlu, aktuální klastrové verzi. Toto je definováno při prvním vytvoření klastru. N může být nastaveno buď na potenciální verzi uzlu běžící v uzlu, ze kterého vzešel požadavek na vytvoření klastru, nebo na jednu klastrovou verzi předcházející potenciální verzi uzlu původce. Uzly v klastru se mohou lišit nanejvýš o jednu úroveň klastrové verze.

Jakmile byly všechny systémy v klastru aktualizovány na následující vydání, může být aktualizována klastrová verze, aby byly k dispozici nové funkce. To je možné provést pomocí úpravy klastrové verze.

Upozornění: Jestliže nová verze operačního systému není stejná jako aktuální verze klastru nebo je o jednu verzi vyšší, klastrový uzel při restartu selže. Chcete-li provést obnovu z této situace, musíte uzel vymazat a znovu jej vytvořit se správnou verzí.

Upozornění: Když používáte v klastru přepínatelná ASP, existují určitá omezení při provádění přepnutí mezi vydáními. Musíte přepnout ASP s předchozím vydáním na systém, na němž je aktuální vydání operačního systému i5/OS, a zpřístupnit ji. Jakmile bude ASP zpřístupněna na systému používajícím aktuální vydání operačního systému i5/OS, její vnitřní obsah se změní a nebude možné jej znovu zpřístupnit pro systém s předchozím vydáním.

Další informace o klastrových verzích, včetně informací o omezeních a také o tom, jak klastrové verze odpovídají vydáním operačního systémui5/OS, najdete v dokumentaci k rozrahním API pro klastry.

Související pojmy

"Klastry s více vydáními" na stránce 84
Vytváříte-li klastr, který bude zahrnovat uzly s více klastrovými verzemi, je třeba při vytváření takového klastru provést určité kroky.

"Běžné problémy s klastry" na stránce 132
Zde jsou popsány některé z nejčastějších problémů, ke kterým dochází v klastru, a způsoby, jak se těchto problémů vyvarovat a jak provést nápravu chyb.

Související úlohy

"Vytvoření klastru" na stránce 97
Chcete-li vytvořit a nakonfigurovat klastr, musíte do klastru zahrnout alespoň jeden uzel a musíte mít přístup nejméně k jednomu z uzlů, které budou v klastru.

"Úprava verze klastru" na stránce 101
Verze klastru definuje úroveň, na které spolu všechny klastrové uzly aktivně komunikují.

Odolné prostředky

Odolné prostředky jsou systémové prostředky (data, zařízení, aplikace), které jsou díky klastrování systémů vysoce dostupné.

Jestliže klastrový uzel, který je primárním přístupovým bodem k určité sadě odolných prostředků klastru, utrpí výpadek, stane se přístupovým bodem jiný klastrový uzel, který je pro danou sadu prostředků definován jako záložní.

Odolné mohou být tyto typy systémových prostředků:

1. Data replikovaná mezi uzly.
2. Aplikace používající IP adresu, která může být přepínána z jednoho uzlu na jiný.
3. Hardwarová zařízení, která mohou být přepínána z jednoho uzlu na jiný.
4. Peer prostředky, které jsou podporovány v administrativní doméně klastru.

Definice vztahu mezi uzly asociovanými se sadou odolných prostředků se nachází v objektu *CRG (skupina klastrových prostředků)*. Skupiny klastrových prostředků jsou replikovány a koordinovány mezi klastrovými uzly pomocí služeb klastrových prostředků.

Související pojmy

"Skupina klastrových prostředků" na stránce 7
Skupina klastrových prostředků (CRG) je systémový objekt operačního systému i5/OS tvořený sadou nebo skupinou prostředků klastru, které se používají pro správu událostí, k nimž dochází v klastrovaném prostředí. Skupina klastrových prostředků popisuje doménu obnovy a udává jméno ukončovacího programu skupiny klastrových prostředků, který je volán, když dojde k určitým událostem na klastru.

"Administrativní doména klastru" na stránce 8
Administrativní doména klastru se používá ke správě prostředků, které je nutné udržovat konzistentní ve všech uzlech v klastrovaném prostředí.

Odolné aplikace:

Odolná aplikace je aplikace, která může opětovně spuštěna v jiném klastrovém uzlu, aniž by bylo nutné rekonfigurovat klienty.

Informace o charakteristikách odolných aplikací najdete v tématu Co dělá aplikační programy odolnými.

Odolná aplikace musí být schopna rozpoznat dočasnou ztrátu spojení protokolem Internetu (IP) mezi klientem a serverem. Klientská aplikace musí zaregistrovat, že spojení IP není dočasně k dispozici, a místo aby skončila nebo vyvolala přepnutí při selhání, musí se znovu pokusit o přístup. Podobně když provádíte přepnutí, serverová aplikace musí zaregistrovat, že spojení IP není nadále k dispozici. Serverové aplikaci může být případně vrácen chybový stav. Jakmile je tento chybový stav přijat, nejlepší je, když serverová aplikace tento stav rozpozná a normálně skončí.

Převzetí IP adresy je funkce vysoké dostupnosti, která se používá k ochraně klientů před výpadky aplikačního serveru. **IP adresa převzetí aplikace** je pohyblivá adresa, která má být asociována s aplikací. Princip spočívá v použití jmen alias IP adresy k definování pohyblivé IP adresy, která je asociována s více aplikačními servery či hostitelskými systémy. Jestliže jeden aplikační server v klastru selže, jiný klastrový uzel převezme povinnosti aplikačního serveru, aniž by bylo třeba rekonfigurovat klienty.

Podpora převzetí IP adresy zavádí také koncepci aplikačních skupin klastrových prostředků (CRG). Aplikační CRG jsou skupiny klastrových prostředků, které obsahují prostředek IP adresy převzetí aplikace a doménu obnovy. Doména

obnovy obsahuje seznam aplikačních serverů v klastru, které podporují konkrétní aplikaci. Pokud některý prostředek selže, služby klastrových prostředků vyvolají u skupiny, do které daný prostředek patří, přepnutí při selhání.

Související pojmy

"Jak udělat aplikační programy odolnými" na stránce 29
Zde se dozvíte, jak zajistit, aby aplikační programy byly odolné.

"Doména obnovy" na stránce 11
Doména obnovy je podmnožina klastrových uzlů, které jsou s určitým záměrem, například kvůli akci obnovy nebo synchronizaci událostí, seskupeny do skupiny klastrových prostředků (CRG).

Související úlohy

"Klastrové aplikace" na stránce 28
Jedním z klíčových prvků v klastrovaném prostředí je odolnost aplikací. Jestliže chcete psát a používat ve svém klastru velmi dostupné aplikace, měli byste zajistit, aby tyto aplikace měly specifické vlastnosti pro dostupnost.

Odolná data:

Odolná data jsou data, která jsou replikována (kopírována) alespoň na jeden další klastrový uzel.

Každý uzel v doméně obnovy obsahuje kopii odolných dat, která je udržována pomocí určitého replikačního mechanismu. Uzly, které jsou v doméně obnovy definovány jako záložní, mohou převzít roli primárního bodu přístupu k odolným datům. Uzly, které jsou definovány jako replikační, obsahují rovněž kopii dat, nemohou však převzít roli primárního uzlu. Data zkopírovaná na replikační uzel se obvykle používají k ulehčení práce primárního uzlu, například k zálohování nebo k provádění dotazů pouze pro čtení.

Související pojmy

"Replikace" na stránce 25
Replikace vytváří kopii něčeho v reálném čase. Představuje kopírování objektů z jednoho klastrového uzlu do jednoho nebo více dalších uzlů klastru.

Odolná zařízení:

Odolná zařízení jsou fyzické prostředky reprezentované konfiguračními objekty, jako je popis zařízení, které jsou přístupné z více než jednoho klastrového uzlu.

Dojde-li k výpadku, přepne se bod přístupu k prostředku na první záložní uzel v doméně obnovy skupiny klastrových prostředků. Nezávislá ASP (nezávislé společné diskové oblasti) jsou odolná zařízení, která mohou přecházet do stavu offline nebo online, aniž by to mělo vliv na zbytek systémové paměti. Kromě toho můžete použít geografické zrcadlení, což je podfunkce zrcadlení XSM (cross-site mirroring), které je součástí i5/OS volba 41, High Available Switchable Resources. Geografické zrcadlení je funkcí, která udržuje dvě totožné kopie nezávislého ASP na dvou pracovištích, aby byla zajištěna vysoká dostupnost a náprava po nehodě. Kopie vlastněná primárním uzlem je výrobní kopie a kopie vlastněná záložním uzlem na jiném stanovišti je zrcadlená kopie. Uživatelské operace a aplikace přistupují k nezávislému ASP na primárním uzlu, což je uzel, který vlastní výrobní kopii.

Skupina CRG (skupina klastrových prostředků) odolných zařízení může obsahovat seznam přepínatelných zařízení. Každé zařízení v seznamu určuje přepínatelné nezávislé ASP. Dojde-li k výpadku, je celá kolekce zařízení přepnuta na záložní uzel. Zařízení lze rovněž volitelně logicky zapínat jako součást procesu přepnutí nebo přepnutí při selhání. Pro fyzickou konfiguraci vztahující se k seznamu přepínatelných zařízení platí určitá omezení. Další informace o tom, jak nastavit odpovídající konfiguraci pro nezávislé ASP definované jako odolné, najdete v tématu Nezávislá ASP.

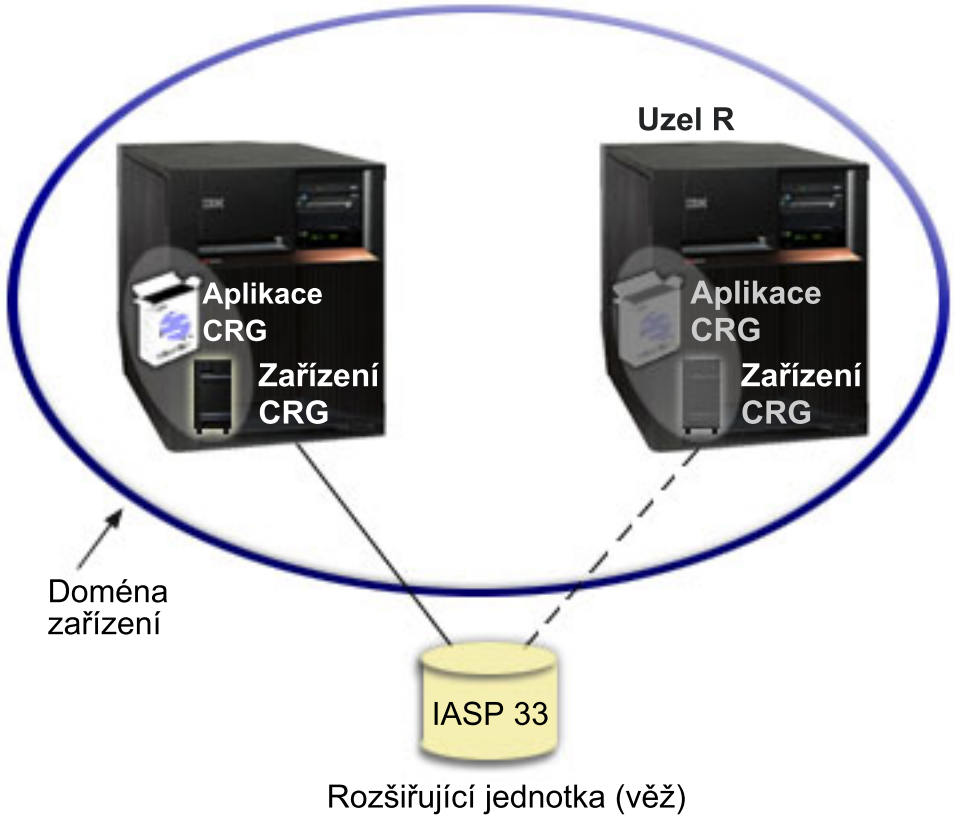
Skupina CRG odolných zařízení se velmi podobá ostatním typům CRG. Jeden z rozdílů, tj. seznam přepínatelných zařízení, byl zmíněn výše. Další odlišností je, že pro CRG zařízení je volitelný ukončovací program. Je-li zapotřebí zpracování specifické z hlediska prostředí nebo dat, lze u CRG použít ukončovací program. Další informace o tomto typu CRG najdete v tématu Rozhraní QcstCreateClusterResourceGroup (Create Cluster Resource Group) API .

Domény zařízení

Doména zařízení je podmnožina uzlů v klastru, které sdílejí prostředky zařízení. Konkrétněji: uzly v doméně zařízení se mohou účastnit na akci přepínání pro určitý soubor prostředků odolného zařízení.

Domény zařízení se identifikují a řídí prostřednictvím sady rozhraní, která vám umožňují přidat uzel do domény zařízení nebo odebrat uzel z domény zařízení.

Domény zařízení se používají k řízení některých globálních informací nutných pro přepnutí odolného zařízení z jednoho uzlu na jiný uzel. Tyto informace potřebují všechny uzly v doméně zařízení, aby při přepnutí zařízení nevznikl žádný konflikt. Například u souboru přepínatelných nezávislých ASP je nutné, aby v rámci celé domény zařízení byly jedinečné identifikace nezávislých ASP, přiřazení diskových jednotek a přiřazení virtuálních adres.



Klastrový uzel může patřit pouze do jedné domény zařízení. Předtím, než lze uzel přidat do domény obnovy pro CRG zařízení, musí být uzel definovaný jako člen domény zařízení. Všechny uzly, které budou v doméně obnovy pro CRG zařízení, musí být v téže doméně zařízení.

Chcete-li vytvářet a spravovat domény zařízení, musí být v systému nainstalována volba 41 (i5/OS - HA Switchable Resources) spolu s platným licenčním klíčem.

Související pojmy

"Příklad: Klastr s přepínanými disky používající nezávislá ASP" na stránce 118
Klastr používající technologii přepínaných disků poskytuje alternativu k replikaci dat. V klastru s přepínanými disky jsou data ve skutečnosti obsažena v nezávislých ASP.

Související úlohy

"Přidání uzlu do domény zařízení" na stránce 105
Doména zařízení je podmnožina uzlů klastru, které sdílejí prostředky zařízení.

"Odebrání uzlu z domény zařízení" na stránce 106
Doména zařízení je podmnožina uzlů klastru, které sdílejí prostředky zařízení.

Volba 41 (HA Switchable Resources):

Chcete-li vytvářet a spravovat domény zařízení, musí být v systému nainstalována volba 41 (i5/OS - HA Switchable Resources) spolu s platným licenčním klíčem.

Tuto část musíte mít nainstalovánu, když chcete provést cokoli z následujícího v klastrovaném prostředí:

- Používat rozhraní iSeries Navigator pro správu klastru.
- Přepínat nezávislá ASP mezi systémy.
- Zrcadlení mezi stanovišti u geograficky vzdálených systémů.

Související úlohy

"Přidání uzlu do domény zařízení" na stránce 105
Doména zařízení je podmnožina uzlů klastru, které sdílejí prostředky zařízení.

"Odebrání uzlu z domény zařízení" na stránce 106
Doména zařízení je podmnožina uzlů klastru, které sdílejí prostředky zařízení.

Události klastru

V rámci klastru může docházet k několika typům událostí, akcí a služeb.

Přepnutí při selhání

K *přepnutí při selhání* dochází, když se server v klastru automaticky přepne na jeden nebo více záložních serverů v případě selhání systému.

Na rozdíl od toho přepnutí nastává tehdy, když manuálně přepnete přístup z jednoho serveru na jiný. Poté, co je přepnutí a přepnutí při selhání zahájeno, fungují již shodně. Jediný rozdíl spočívá v tom, jak jsou tyto události vyvolány.

Když dojde k přepnutí při selhání, přepne se přístup z klastrového uzlu, který aktuálně fungoval jako primární uzel v doméně obnovy pro skupinu klastrových prostředků, na uzel označený jako první záložní uzel. Informace o tom, jak se určuje pořadí přepnutí při selhání, najdete v tématu doména obnovy.

Je-li do akce přepnutí zapojeno více skupin klastrových prostředků (CRG), systém nejprve zpracovává CRG zařízení (přepínatelná zařízení), pak datové CRG (skupiny přepínatelných dat) a nakonec aplikační CRG (přepínatelné softwarové produkty).

Fronta zpráv přepnutí při selhání dostává zprávy týkající se průběhu přepnutí při selhání. Můžete ji využít při kontrole zpracování přepnutí při selhání pro skupinu klastrových prostředků.

Související pojmy

"Doména obnovy" na stránce 11
Doména obnovy je podmnožina klastrových uzlů, které jsou s určitým záměrem, například kvůli akci obnovy nebo synchronizaci událostí, seskupeny do skupiny klastrových prostředků (CRG).

"Skupina klastrových prostředků" na stránce 7
Skupina klastrových prostředků (CRG) je systémový objekt operačního systému i5/OS tvořený sadou nebo skupinou prostředků klastru, které se používají pro správu událostí, k nimž dochází v klastrovaném prostředí. Skupina klastrových prostředků popisuje doménu obnovy a udává jméno ukončovacího programu skupiny klastrových prostředků, který je volán, když dojde k určitým událostem na klastru.

"Fronta zpráv přepnutí při selhání" na stránce 113
Fronta zpráv přepnutí při selhání dostává zprávy týkající se průběhu přepnutí při selhání.

"Hardwarové požadavky pro klastry" na stránce 80
Pro použití klastrování je kompatibilní jakýkoli model iSeries, který je schopen spouštět operační systém i5/OS verze V4R4M0 nebo novější.

Související úlohy

“Přepnutí” na stránce 20
K *přepnutí* dojde, když ručně přepnete přístup k prostředku z jednoho serveru na jiný.

Příklad: Selhání:

K přepnutí při selhání obvykle dojde následkem selhání uzlu, ale existují také další důvody, které mohou způsobit přepnutí při selhání.

Je možné, že problém ovlivní pouze jednu skupinu klastrových prostředků, která může způsobit přepnutí při selhání pro tuto skupinu klastrových prostředků (CRG), ale nikoli pro žádnou jinou skupinu CRG.

Níže uvedená tabulka ukazuje různé typy selhání a jejích kategorií:

Selhání	Obecná kategorie
Porucha hardwaru CEC (například CPU).	2
Porucha komunikačního adaptéru, linky, směrovače nebo ENDTCPIFC ovlivňující všechny adresy rozhraní IP definované pro uzel.	4
Výpadek proudu pro CEC.	1
Porucha softwarového zařízení operačního systému	2
Je vydán příkaz ENDTCP(*IMMED nebo *CNTRLD s časovým omezením)	1
Je vydán příkaz ENDSBS QSYSWRK(*IMMED nebo *CNTRLD).	1
Je vydán příkaz ENDSBS(*ALL, *IMMED nebo *CNTRLD).	1
Je vydán příkaz ENDSYS (*IMMED nebo *CNTRLD).	1
Je vydán příkaz PWRDWNSYS(*IMMED nebo *CNTRLD).	1
Během aktivity Služeb klastrových prostředků v systému je stlačeno tlačítko IPL.	1
Je vydán příkaz Cancel Job (*IMMED nebo *CNTRLD s časovým limitem) z úlohy QCSTCTL	1
Je vydán příkaz Cancel Job (*IMMED nebo *CNTRLD s časovým limitem) z úlohy QCSTCRGM	1
Je vydán příkaz Cancel Job (*IMMED nebo *CNTRLD s časovým limitem) z úlohy skupiny klastrových prostředků.	3
Je voláno rozhraní End Cluster Node API.	1
Je voláno rozhraní Remove Cluster Node API.	1
Úloha skupiny klastrových prostředků má softwarovou chybu, která způsobí její abnormální ukončení.	3
Zadání funkce 8 nebo 3 z ovládacího panelu za účelem vypnutí systému.	2
Zadání funkce 7 za účelem odloženého vypnutí části.	1
Selhání aplikačního programu pro aplikační skupinu klastrových prostředků.	3

Selhání	Obecná kategorie
Obecná kategorie: 1. Všechny Služby klastrových prostředků (CRS) v uzlu selžou, což je detekováno jako selhání uzlu. Uzel může být ve skutečnosti funkční nebo mohl také selhat (mohlo například dojít k selhání systémy kvůli výpadku proudu). Pokud selžou všechny Služby klastrových prostředků, projdou prostředky spravované CRS procesem přepnutí při selhání. 2. V uzlu selžou všechny CRS, ale toto selhání je detekováno jako rozdělení klastru. Uzel může nebo nemusí být funkční. 3. V jedné skupině klastrových prostředků dojde k selhání. Tyto okolnosti jsou vždy detekovány jako selhání. 4. Dojde k selhání, ale uzel a Služby klastrových prostředků jsou stále funkční a toto selhání je detekováno jako rozdělení klastru.	

Když dojde k selhání, závisejí akce provedené Službami klastrových prostředků pro danou skupinu klastrových prostředků na typu selhání a stavu skupiny klastrových prostředků. V každém případě je však volán ukončovací program. Přepnutí při selhání bude možná muset pracovat se seznamem uzlů, u kterých došlo k selhání. Když je volán ukončovací program, potřebuje tento program určit, zda se musí zabývat pouze selháním jednoho uzlu nebo se seznamem uzlů, u kterých došlo k selhání.

Je-li skupina klastrových prostředků *neaktivní*, stav členství uzlu, u kterého došlo k selhání, v doméně obnovy skupiny klastrových prostředků se změní na stav *Inactive* nebo *Partition*. Úlohy uzlů se však nezmění a nedojde k novému uspořádání záložních uzlů. Záložní uzly jsou znovu uspořádány v neaktivní skupině klastrových prostředků v případě, kdy je volán příkaz STRCRG (Spuštění skupiny klastrových prostředků) nebo rozhraníQcstStartClusterResourceGroup) (Start Cluster Resource Group) API.Pokud však není primární uzel aktivní, API QcstStartClusterResourceGroup (Spuštění skupiny klastrových prostředků) selže. Chcete-li označit aktivní uzel jako primární uzel, musíte vydat příkaz CHGCRG (Změna skupiny klastrových prostředků) nebo rozhraní QcstChangeClusterResourceGroup (Change Cluster Resource Group) API a potom znovu zavolat rozhraní Start Cluster Resource Group API.

Je-li skupina klastrových prostředků *aktivní* a uzel, u kterého došlo k selhání, *není* primárním uzlem, přepnutí při selhání aktualizuje stav člena domény obnovy v doméně obnovy skupiny klastrových prostředků, u kterého došlo k selhání. Jestliže je uzel, u kterého došlo k selhání, záložním uzlem, dojde k novému uspořádání seznamu záložních uzlů tak, aby záložní uzly byly na začátku seznamu.

Je-li skupina klastrových prostředků *aktivní* a člen domény obnovy je primárním uzlem, provedou se v závislosti na typu selhání níže uvedené akce:

Selhání kategorie 1
Dojde k přepnutí při selhání. Primární uzel je v takové skupině klastrových prostředků označen jako *neaktivní* a je zařazen jako poslední záložní uzel. Uzel, který byl prvním záložním uzlem, se stane novým primárním uzlem. Všechny skupiny klastrových prostředků pro zařízení přepnou při selhání jako první. Potom přepnou při selhání všechny datové skupiny klastrových prostředků. Nakonec přepnou všechny aplikační skupiny klastrových prostředků. Detekuje-li přepnutí při selhání pro některou skupinu CRG, že není aktivní žádný záložní uzel, nastaví se stav skupiny CRG na *indoubt*.

Selhání kategorie 2
Dojde k přepnutí při selhání, ale primární uzel se nezmění. Všechny uzly v části klastru, které nemají primární uzel jako člena této části, ukončí aktivní skupinu klastrových prostředků. Stav uzlů v doméně obnovy v skupině klastrových prostředků je nastaven na stav *partition* pro každý uzel, který se nachází v primární části. Pokud skutečně došlo k selhání uzlu, ale toto selhání je detekováno pouze jako problém s rozdělením a uzel, který selhal, byl primárním uzlem, ztratíte všechna data a aplikační služby na tomto uzlu a nedojde k automatickému přepnutí při selhání. Musíte buď uzel deklarovat jako selhaný, nebo zavést zálohu uzlu a opět spustit klastrování v tomto uzlu. Další informace najdete v tématu Změna stavu uzlů z ″Partitioned″ na ″Failed″.

Selhání kategorie 3
Pokud je ovlivněna pouze jedna skupina klastrových prostředků, dojde k přepnutí při selhání jen pro jednotlivé případy, protože skupiny klastrových prostředků jsou na sobě nezávislé. Může se stát, že je současně ovlivněno několik skupin klastrových prostředků, protože někdo rušil několik úloh klastrových prostředků.

Typ selhání je však zpracováván ve skupině CRG v rámci jedné skupiny CRG a nedochází ke koordinovanému přepnutí při selhání mezi skupinami. Primární uzel je označen jako *neaktivní* v každé skupině klastrových prostředků a je zařazen jako poslední záložní uzel. Uzel, který byl prvním záložním uzlem, se stane novým primárním uzlem. Neexistuje-li žádný aktivní záložní uzel, je stav skupiny klastrových prostředků nastaven na *indoubt*.

Selhání kategorie 4

Tato kategorie je podobná kategorii 2. Avšak zatímco všechny uzly a Služby klastrových prostředků v uzlech jsou stále funkční, ne všechny uzly mohou mezi sebou komunikovat. Klastr je rozdělen na části, ale primární uzel nebo uzly stále poskytují služby. Kvůli rozdělení na části však může docházet k různým problémům. Když je například primární uzel v jedné části a všechny záložní uzly nebo replikační uzly v jiné části, nedochází již k replikaci dat a nemáte žádnou ochranu v případě, že selže primární uzel. V části, která obsahuje primární uzel, aktualizuje proces přepnutí při selhání stav uzlů v doméně obnovy skupiny klastrových prostředků na stav *partition* pro všechny uzly v druhé části. V části, která neobsahuje primární uzel, je stav uzlů v doméně obnovy skupiny klastrových prostředků nastaven na stav *partition* pro všechny uzly v druhé části.

Související pojmy

“Chyby logických částí” na stránce 134
Některé podmínky klastru lze snadno opravit. Pokud došlo k rozdělení klastru, můžete se naučit, jak provést nápravu chyb. Toto téma také popisuje, jak se vyvarovat rozdělení klastru, a udává příklad způsobu opětovného sloučení jednotlivých částí.

Přepnutí

K *přepnutí* dojde, když ručně přepnete přístup k prostředku z jednoho serveru na jiný.

Ruční přepnutí obvykle vyvoláte, pokud chcete provádět údržbu systému, například aplikovat PTF, instalovat nové vydání nebo přejít na vyšší verzi systému. Něco jiného je přepnutí při selhání, které nastává automaticky při výpadku primárního uzlu.

Když dojde k přepnutí, přístup je přepnut z klastrového uzlu, který momentálně funguje jako primární uzel domény obnovy skupiny klastrových prostředků (CRG), na klastrový uzel určený jako první záložní uzel. Informace o tom, jak se určuje pořadí přepnutí při selhání, najdete v tématu doména obnovy.

Provádíte-li administrativní přepnutí více CRG, měli byste při určování pořadí vzít v úvahu vztahy mezi jednotlivými CRG. Pokud například máte CRG aplikační závisející na datech asociovaných s CRG zařízení, postupujte při přepínání takto:

1. Ukončete aplikaci v starém primárním uzlu (aby přestalo docházet ke změnám dat).
2. Přepněte CRG zařízení na nový primární uzel.
3. Přepněte aplikační CRG na nový primární uzel.
4. Spusťte znovu aplikaci v novém primárním uzlu.

Související pojmy

“Přepnutí při selhání” na stránce 17
K *přepnutí při selhání* dochází, když se server v klastru automaticky přepne na jeden nebo více záložních serverů v případě selhání systému.

“Doména obnovy” na stránce 11
Doména obnovy je podmnožina klastrových uzlů, které jsou s určitým záměrem, například kvůli akci obnovy nebo synchronizaci událostí, seskupeny do skupiny klastrových prostředků (CRG).

Související úlohy

“Provedení přepnutí” na stránce 104
Provedení ručního přepnutí způsobí, že se aktuální primární uzel přepne na záložní uzel, který je definován v doméně obnovy skupiny klastrových prostředků.

Opětovné připojení

Opětovné připojení znamená, že nezúčastněný člen se stane aktivním členem klastru.

Když je například restartováno klastrování uzlu, který byl neaktivní, klastrový uzel se opětovně připojí ke klastru. Služby klastrových prostředků v uzlu můžete spustit z uzlu, který je již v klastru aktivní. Začínáte-li s klastrem verze 3, může se uzel spustit sám. Uzel bude schopen znovu se připojit k aktuálně aktivnímu klastru za předpokladu, že v klastru nalezne aktivní uzel. Další informace najdete v tématu Spuštění klastrového uzlu.

Předpokládejme, že klastr je tvořen uzly A, B a C. Uzel A selže. Aktivní klastr je nyní tvořen uzly B a C. Až bude porouchaný uzel znovu provozuschopný, může být znovu připojen ke klastru - spuštěním tohoto uzlu z libovolného klastrového uzlu (včetně uzlu samotného). Operace opětovného připojení probíhá na základě skupin klastrových prostředků, což znamená, že každá skupina klastrových prostředků (CRG) se připojuje ke klastru nezávisle.

Primární funkce opětovného připojení zajišťuje, aby byl objekt CRG replikován na všechny aktivní uzly domény obnovy. Opětovně připojovaný uzel i všechny aktivní klastrové uzly musí obsahovat identickou kopii objektu CRG. Kromě toho musí obsahovat identickou kopii určitých interních dat.

Jestliže jsou v administrativní doméně klastrů uzly, které nejsou aktivní, veškeré změny prostředků, které jsou provedeny v aktivní doméně, budou rozšířeny na neaktivní uzly, jakmile se vrátí do aktivní domény.

Jestliže nějaký uzel selže, může pokračující volání služeb klastrových prostředků na zbývajících uzlech klastru změnit data v objektu CRG. K modifikaci musí dojít kvůli volání API nebo následnému selhání uzlu. U jednoduchých klastrů je opětovně připojovaný uzel aktualizován kopií CRG z některého uzlu, který je momentálně aktivní v klastru. Tak tomu však nemusí být vždy.

Související úlohy

"Spuštění klastrového uzlu" na stránce 100
Spuštěním klastrového uzlu spustíte Služby klastrových prostředků v uzlu v klastru. Počínaje klastry verze 3 se může uzel spustit sám a opětovně se připojit ke klastru - za předpokladu, že najde v klastru aktivní uzel.

"Změna stavu uzlů na ″Failed″" na stránce 136
Někdy je hlášen stav rozdělení i v případě, kdy ve skutečnosti došlo k výpadku uzlu. Může k tomu dojít tehdy, když Služby klastrových prostředků ztratí komunikaci s jedním nebo více uzly a nejsou schopny detekovat, zda jsou uzly stále funkční. Pokud dojde k této situaci, existuje jednoduchý způsob, jak označit, že uzel selhal.

Příklad: Opětovné připojení:

Toto téma popisuje akce, které se mohou vyskytnout, když se uzel vrátí do klastru.

Následující tabulka popisuje akce prováděné při opětovném připojování uzlu ke klastru. Navíc se stav opětovně připojovaných uzlů v poli stavu členství v doméně obnovy CRG změní z hodnoty *neaktivní* na *aktivní*. Ve všech uzlech v doméně obnovy CRG je zavolán ukončovací program a je mu předán kód akce Rejoin (opětovné připojení).

Tabulka 3. Operace opětovného připojení

Operace opětovného připojení			
Opětovně připojovaný uzel		Klastrové uzly	
Obsahuje kopii CRG	Neobsahuje kopii CRG	Obsahují kopii CRG	Neobsahují kopii CRG
(1)	(2)	(3)	(4)

Podle výše uvedené tabulky jsou možné tyto situace:

1. 1 a 3
2. 1 a 4
3. 2 a 3
4. 2 a 4

Obsahuje-li klastrový uzel kopii CRG, obecně pro opětovné připojení platí, že objekt CRG je zkopírován z aktivního klastrového uzlu do opětovně připojovaného uzlu.

Situace 1 opětovného připojení

Kopie objektu CRG z klastrového uzlu je odeslána do připojovaného uzlu. Důsledek je:

- Objekt CRG je v připojovaném objektu aktualizován daty poslanými z klastru.
- Objekt CRG může být vymazán z připojovaného uzlu. K tomu může dojít, pokud byl připojovaný uzel odstraněn z domény obnovy CRG v době, kdy byl připojovaný uzel mimo klastr.

Situace 2 opětovného připojení

Kopie objektu CRG z připojovaného uzlu je odeslána do všech klastrových uzlů. Důsledek je:

- Pokud žádný z klastrových uzlů není v doméně obnovy CRG, nedojde k žádné změně.
- Objekt CRG může být vytvořen v jednom nebo více klastrových uzlech. Může k tomu dojít při tomto scénáři:
  - Klastr je tvořen uzly A, B, C a D.
  - Všechny čtyři uzly jsou v doméně obnovy CRG.
  - V době, kdy je uzel A mimo klastr, je objekt CRG modifikován tak, aby byl uzel B odstraněn z domény obnovy.
  - Uzly C a D selžou.
  - Klastr je tvořen pouze uzlem B, který neobsahuje kopii CRG.
  - Uzel A se opětovně připojí ke klastru.
  - Uzel A má CRG (ačkoli je nyní na nižší úrovni) a uzel B nemá. Objekt CRG je vytvořen v uzlu B. Když se uzly C a D opětovně připojí ke klastru, kopie CRG v klastru aktualizuje uzly C a D a předchozí změna, při které byl uzel B odstraněn z domény obnovy, je ztracena.

Situace 3 opětovného připojení

Kopie objektu CRG z klastrového uzlu je odeslána do připojovaného uzlu. Důsledek je:

- Pokud připojovaný uzel není v doméně obnovy CRG, nedojde k žádné změně.
- Objekt CRG může být vytvořen v připojovaném uzlu. K tomu může dojít, jestliže byl objekt CRG vymazán z připojovaného uzlu v době, kdy služby klastrových prostředků nebyly v uzlu aktivní.

Situace 4 opětovného připojení

Určité interní informace z některého klastrového uzlu mohou být použity k aktualizaci informací v připojovaném uzlu, ale nestane se nic, čeho byste si mohli všimnout.

Sloučení

Operace *sloučení* je podobná operaci opětovné připojení, až na to, že k ní dochází tehdy, když uzly, které jsou rozdělené, začínají opět komunikovat.

Rozdělení může být skutečné rozdělení v tom, že služby klastrových prostředků jsou stále aktivní ve všech uzlech. Některé uzly však nemohou komunikovat s jinými uzly kvůli poruše komunikační linky. Nebo může problém spočívat v tom, že uzel ve skutečnosti selhal, ale tento stav nebyl zaznamenán jako selhání.

V prvním případě se části klastru sloučí zpátky dohromady automaticky, jakmile je problém komunikace vyřešen. K tomu dojde, když se obě části klastru pravidelně pokoušejí komunikovat s rozdělenými uzly a nakonec mezi sebou znovu navážou kontakt. V druhém případě je nutno v uzlu, kde došlo k selhání, restartovat služby klastrových prostředků, a to spuštěním uzlu z jiného uzlu v klastru.

Související pojmy

"Opětovné připojení" na stránce 20
Opětovné připojení znamená, že nezúčastněný člen se stane aktivním členem klastru.

"Chyby logických částí" na stránce 134
Některé podmínky klastru lze snadno opravit. Pokud došlo k rozdělení klastru, můžete se naučit, jak provést nápravu chyb. Toto téma také popisuje, jak se vyvarovat rozdělení klastru, a udává příklad způsobu opětovného sloučení jednotlivých částí.

Související úlohy

| “Spuštění klastrového uzlu” na stránce 100
| Spuštěním klastrového uzlu spustíte Služby klastrových prostředků v uzlu v klastru. Počínaje klastry verze 3 se
| může uzel spustit sám a opětovně se připojit ke klastru - za předpokladu, že najde v klastru aktivní uzel.

| “Změna stavu uzlů na ″Failed″” na stránce 136
| Někdy je hlášen stav rozdělení i v případě, kdy ve skutečnosti došlo k výpadku uzlu. Může k tomu dojít tehdy, když
| Služby klastrových prostředků ztratí komunikaci s jedním nebo více uzly a nejsou schopny detekovat, zda jsou uzly
| stále funkční. Pokud dojde k této situaci, existuje jednoduchý způsob, jak označit, že uzel selhal.

| **Příklad: Sloučení:**

| Operace sloučení mohou probíhat v několika různých situacích.

| Operace sloučení může nastat v některé z těchto konfigurací:

| *Tabulka 4. Sloučení primární a sekundární části*

sloučení	
primární část	sekundární část

| *Tabulka 5. Sloučení sekundární a sekundární části*

sloučení	
sekundární část	sekundární část

| Primární a sekundární části jsou ve skupinách klastrových prostředků (CRG) jedinečné. Primární část CRG v modelu
| primární-záložní je definována jako část, která obsahuje uzel označený jako primární přístupový bod. Sekundární část
| je definována jako část, která neobsahuje uzel označený jako primární přístupový bod.

| Jestliže uzly domény obnovy jsou v případě peer CRG plně obsaženy v jedné části, bude tato část primární. Jestliže
| uzly domény obnovy přesahují přes jednu logickou část, nebude primární žádná část. Obě části budou sekundární části.

| Když je v případě administrativní domény klastru doména rozdělena na dvě nebo více logických částí, každá část bude
| nadále pracovat jako jedna skupina. Změny prostředků budou nadále synchronizovány v každé části. Když se části
| znovu spojí zpět, systém bude synchronizovat změny z každé části. Konečný výsledek je to, že monitorované
| prostředky budou konzistentní v celé doméně. Nemůžete přidat ani odstranit záznamy MRE, když je administrativní
| doména klastru rozdělena na části.

| *Tabulka 6. Sloučení primární a sekundární části*

operace sloučení			
primární část		sekundární část	
obsahuje kopii CRG	NEOBSAHUJE kopii CRG	obsahuje kopii CRG	NEOBSAHUJE kopii CRG
(1)	(2)	(3)	(4)

| Při sloučení primární a sekundární části podle výše uvedené tabulky jsou možné tyto situace:
| 1. 1 a 3
| 2. 1 a 4
| 3. 2 a 3 (Nemůže nastat, protože majoritní část má aktivní primární uzel a musí mít kopii CRG.)
| 4. 2 a 4 (Nemůže nastat, protože majoritní část má aktivní primární uzel a musí mít kopii CRG.)

Situace sloučení primární a sekundární části

Kopie objektu CRG je odeslána do všech uzlů v sekundární části. V uzlech sekundární části může dojít k těmto akcím:

- Nedojde k žádné akci, protože sekundární uzel není v doméně obnovy CRG.
- Kopie CRG sekundárního uzlu je aktualizována údaji z primární části.
- Objekt CRG je vymazán ze sekundárního uzlu, protože sekundární uzel není nadále v doméně obnovy CRG.
- Objekt CRG je vytvořen v sekundárním uzlu, protože tento objekt neexistuje. Uzel je však v doméně obnovy kopie CRG, která je poslána z primární části.

Tabulka 7. Scénář sloučení sekundární a sekundární části

operace sloučení			
sekundární část		sekundární část	
obsahuje kopii CRG	NEOBSAHUJE kopii CRG	obsahuje kopii CRG	NEOBSAHUJE kopii CRG
(1)	(2)	(3)	(4)

Při sloučení dvou sekundárních logických částí podle výše uvedené tabulky jsou možné tyto situace:

1. 1 a 3
2. 1 a 4
3. 2 a 3
4. 2 a 4

Situace 1 sloučení dvou sekundárních logických částí

V modelu CRG primární-sekundární se jako uzel, který odešle kopii objektu CRG do všech uzlů druhé části, vybírá uzel s nejnovější změnou CRG. Pokud je vybráno více uzlů, protože se zdá, že v nich došlo k nejnovější změně, bude při výběru uzlu použito pořadí domény obnovy.

Při spojování dvou sekundárních logických částí pro peer CRG se peer skupiny CRG s aktivním stavem zkopírují do jiných uzlů v druhé části. Jestliže obě části mají stejný stav peer skupin CRG, bude část, která obsahuje první uzel uvedený v doméně obnovy skupiny klastrových prostředků, zkopírována do uzlů na druhé části.

V uzlech v přijímající části mohou proběhnout následující akce, a to v CRG primární-záložní i v peer CRG:

- Nedojde k žádné akci, protože uzel není v doméně obnovy CRG.
- Objekt CRG je v uzlu vytvořen, protože uzel je v doméně obnovy kopie objektu CRG, který přijímá.
- Objekt CRG je z uzlu vymazán, protože uzel není v doméně obnovy kopie objektu CRG, který přijímá.

Situace 2 a 3 sloučení dvou sekundárních logických částí

Jako uzel, který odešle data objektu do všech uzlů druhé části, je vybrán uzel z části, která obsahuje kopii objektu CRG. Objekt CRG může být vytvořen v uzlech přijímající části, pokud je uzel v doméně obnovy CRG.

Situace 4 sloučení dvou sekundárních logických částí

K zajištění konzistence v celém klastru dojde k výměně interních dat.

Primární část může být následně rozdělena na primární a sekundární část. Jestliže selže primární uzel, služba CRS (Služba klastrových prostředků) to detekuje jako selhání uzlu. Primární část se stane sekundární částí. Pokud byste ukončili primární uzel používající rozhraní End Cluster Node API, bude to mít stejné důsledky. Sekundární část se může stát primární částí, jestliže se operací opětovného připojení nebo sloučení stane aktivní primární uzel části.

U operace sloučení je ve všech uzlech v doméně obnovy CRG zavolán ukončovací program, bez ohledu na to, ve které části uzel je. Je použit stejný kód akce jako u opětovného připojení. V důsledku sloučení se nezmění žádné role, ale stav členství uzlů v doméně obnovy CRG se změní z *rozdělení* na *aktivní*. Až se všechny části sloučí, je stav rozdělení zrušen a lze používat všechna rozhraní API CRG.

Replikace

Replikace vytváří kopii něčeho v reálném čase. Představuje kopírování objektů z jednoho klastrového uzlu do jednoho nebo více dalších uzlů klastru.

Replikace vytváří a uchovává identické objekty v systémech. Provedete-li v klastrovém uzlu změnu objektu, je tato změna replikována do dalších uzlů v klastru.

Související pojmy

"Odolná data" na stránce 15
Odolná data jsou data, která jsou replikována (kopírována) alespoň na jeden další klastrový uzel.

"Plánování pro logickou replikaci" na stránce 87
Pomocí logické replikace je udržováno více kopií dat. Data jsou replikována nebo kopírována z primárního uzlu v klastru do záložních uzlů označených v doméně obnovy. Když dojde v primárním uzlu k výpadku, data zůstanou k dispozici, neboť označený záložní uzel převezme roli primárního bodu přístupu.

Monitorování pulsu (heartbeat)

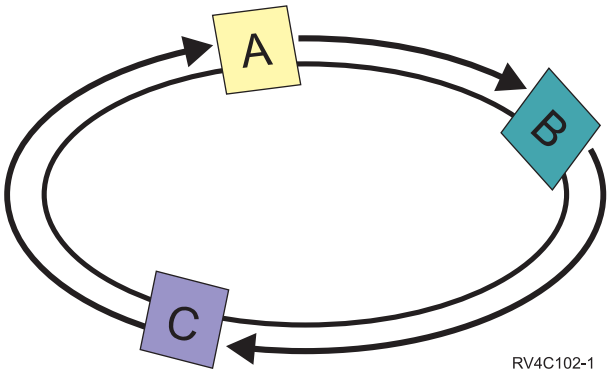
Monitorování pulsu je jednou z funkcí služeb klastrových prostředků, která zajišťuje, že každý uzel je aktivní: zasílá signál z jednotlivých uzlů klastru na všechny další uzly v klastru a tím ověřuje, že jsou stále aktivní.

Když selže monitorování pulsu u nějakého uzlu, služby prostředků klastru provedou příslušnou akci.

Podívejte se následující příklady, abyste lépe pochopili, jak monitorování pulsu funguje:

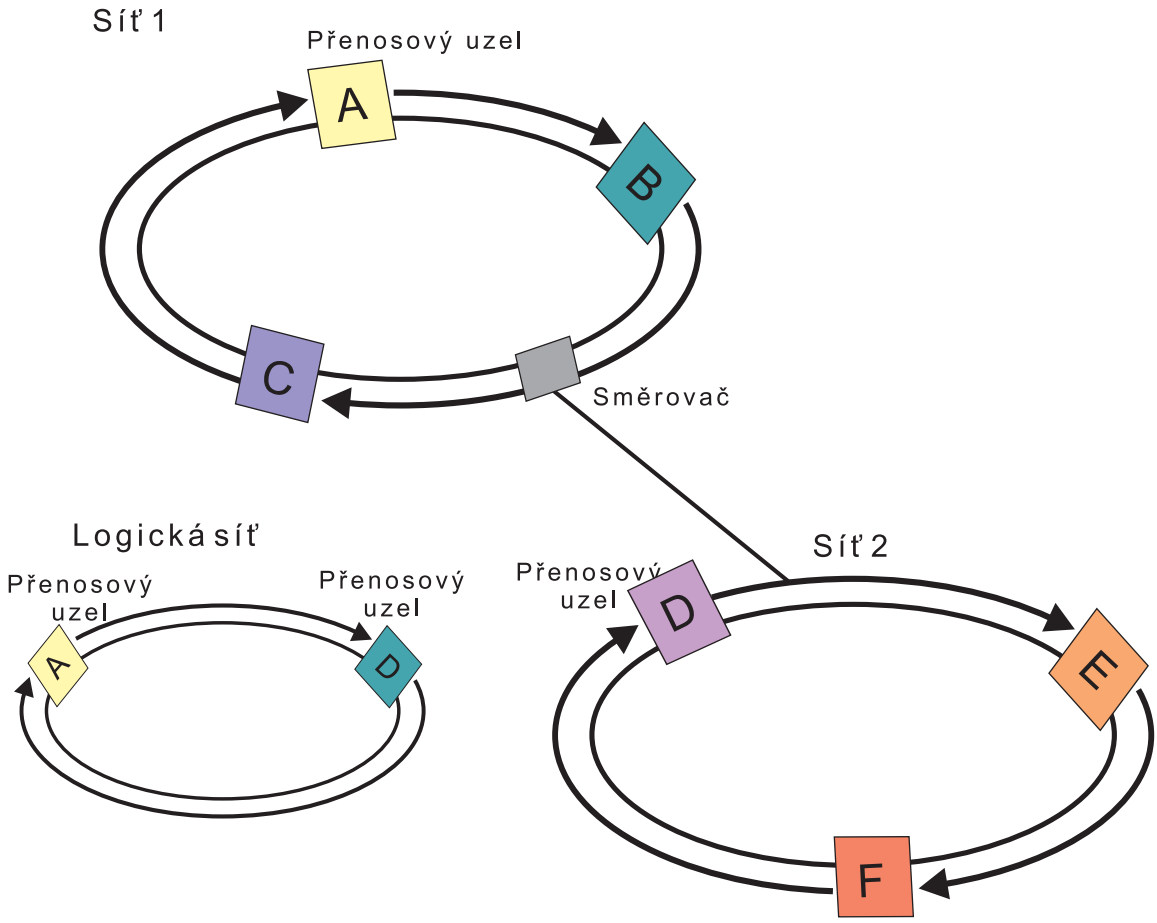
Příklad 1

Síť 1



Při předvoleném (nebo normálním) nastavení se pulsová zpráva odesílá každé tři vteřiny z každého uzlu v klastru na jeho souseda v protisměru. Například jestliže nakonfigurujete uzel A, uzel B a uzel C v síti 1, uzel A odešle zprávu na uzel B, uzel B odešle zprávu na uzel C a uzel C odešle zprávu na uzel A. Uzel A očekává potvrzení příjmu pulsu od uzlu B a rovněž příchozí puls od uzlu C. Ve skutečnosti tedy pulsový okruh probíhá oběma směry. Pokud uzel A neobdrží puls od uzlu C, bude uzel A a uzel B pokračovat v odesílání pulsu každé 3 vteřiny. Pokud uzel C promešká čtyři po sobě jdoucí pulsy, bude signalizováno selhání pulsu.

Příklad 2



Nyní přidejme do předchozího příkladu další síť, abychom si ukázali použití směrovačů a přenosových uzlů. Nakonfigurujete uzel D, uzel E a uzel F v síti 2. Síť 2 je připojena k síti 1 prostřednictvím směrovače. Směrovačem může být další server iSeries, nebo samostatný směrovač, který směruje komunikaci na další směrovač někde jinde. Každé lokální síti je přiřazen přenosový uzel. Tento přenosový uzel je přiřazen k uzlu, který má nejnižší ID uzlu v dané síti. V síti 1 je jako přenosový uzel přiřazen uzel A, v síti 2 je jako přenosový uzel přiřazen uzel D. Tím je vytvořena logická síť obsahující uzel A a uzel D. Pomocí směrovačů a přenosových uzlů se mohou uzly v těchto dvou sítích navzájem monitorovat a signalizovat selhání kteréhokoliv z nich.

Související pojmy

“Správa klastrů” na stránce 98
Toto téma obsahuje informace zaměřené na některé z operací zahrnujících správu klastrů.

“Výkon klastru” na stránce 109
Změny prováděné v klastru mohou mít vliv na režii potřebnou ke správě klastru.

Související úlohy

“Monitorování stavu klastru” na stránce 109
Služby klastrových prostředků provádějí základní monitorování klastru a jeho komponent pomocí funkce spolehlivých zpráv a monitorování pulsu (heartbeat). Je-li to nutné, podniknou odpovídající akce.

Funkce spolehlivých zpráv

Funkce spolehlivých zpráv služeb klastrových prostředků sleduje každý klastrový uzel a zajišťuje, aby všechny uzly měly konzistentní informace o stavu klastrových prostředků.

Spolehlivé posílání zpráv využívá hodnoty opakování a časového limitu, které jsou pro klastrování jedinečné. Tyto hodnoty jsou předem nastaveny tak, aby vyhovovaly většině prostředí. Lze je však změnit pomocí rozhraní pro změnu

nastavení služeb klastrových prostředků. Hodnoty opakování a časového limitu zpráv určují, kolikrát bude zpráva poslána do uzlu, než bude signalizováno selhání nebo stav rozdělení (na části). Při použití předvolených hodnot opakování a časového limitu je u lokální sítě (LAN) celková doba opakování před signalizací selhání nebo stavu rozdělení přibližně 45 sekund. U vzdálené sítě je pro určení existence stavu selhání nebo rozdělení dovolena delší doba. V případě vzdálené sítě můžete počítat přibližně se 4 minutami a 15 sekundami.

Související pojmy

"Změna nastavení služeb klastrových prostředků"
Předvolené hodnoty ovlivňující časový limit a opakování zpráv jsou nastaveny tak, aby byly vhodné pro většinu typických instalací. Je však možné změnit tyto hodnoty tak, aby to více odpovídalo vašemu komunikačnímu prostředí.

"Správa klastrů" na stránce 98
Toto téma obsahuje informace zaměřené na některé z operací zahrnujících správu klastrů.

Související úlohy

"Monitorování stavu klastru" na stránce 109
Služby klastrových prostředků provádějí základní monitorování klastru a jeho komponent pomocí funkce spolehlivých zpráv a monitorování pulsu (heartbeat). Je-li to nutné, podniknou odpovídající akce.

Změna nastavení služeb klastrových prostředků

Předvolené hodnoty ovlivňující časový limit a opakování zpráv jsou nastaveny tak, aby byly vhodné pro většinu typických instalací. Je však možné změnit tyto hodnoty tak, aby to více odpovídalo vašemu komunikačnímu prostředí.

Hodnoty je možné upravit těmito způsoby:

- Nastavit všeobecnou úroveň výkonu, která vyhovuje danému prostředí.
- Nastavit hodnoty optimalizačních parametrů zpráv přesnějším způsobem.

Při první metodě se přenos zpráv nastaví na jednu ze tří úrovní komunikace. Normální úroveň je předvolená a je podrobně popsána v tématu Monitorování pulsu (heartbeat).

Druhou metodu je obvykle vhodné použít pouze za pomoci odborníka.

Podrobné informace o obou metodách obsahuje rozhraní QcstChgClusterResourceServices (Change Cluster Resource Services) API.

Související pojmy

"Monitorování pulsu (heartbeat)" na stránce 25
Monitorování pulsu je jednou z funkcí služeb klastrových prostředků, která zajišťuje, že každý uzel je aktivní: zasílá signál z jednotlivých uzlů klastru na všechny další uzly v klastru a tím ověřuje, že jsou stále aktivní.

Část klastru

Část klastru je podmnožina aktivních uzlů klastru, která vzniká rozdělením klastru v důsledku selhání komunikace. Členové části klastru mezi sebou udržují navzájem spojení.

Rozdělení klastru a vznik části klastru nastává v klastru vždy, když dojde ke ztrátě komunikace mezi jedním nebo více uzly v klastru a nelze potvrdit selhání ztracených uzlů. Když je zaznamenán stav rozdělení klastru, služby klastrových prostředků omezí typy akcí, které můžete v uzlech v části klastru provádět. K omezení funkcí během stavu rozdělení klastru dochází proto, aby služby klastrových prostředků byly schopny provést sloučení částí klastru, jakmile je problém, který rozdělení způsobil, odstraněn.

Některé CRG operace jsou omezeny, když je klastr rozdělený na části. Další informace o tom, které operace jsou omezeny pro každý typ klastru, najdete v tématu Rozhraní CRG API.

Jestliže je administrativní doména klastrů rozdělena na části, změny budou nadále synchronizovány mezi aktivními uzly v každé části. Až budou uzly opět sloučeny, administrativní doména klastrů rozšíří všechny změny provedené v každé části tak, aby prostředky v rámci aktivní domény byly konzistentní.

Související pojmy

| “Zabránění rozdělení klastru” na stránce 83
| Typickému rozdělení klastru souvisejícího se sítí se lze nejlépe vyhnout tím, že nakonfigurujete redundantní komunikační cesty mezi všemi uzly v klastru.
| “Chyby logických částí” na stránce 134
| Některé podmínky klastru lze snadno opravit. Pokud došlo k rozdělení klastru, můžete se naučit, jak provést nápravu chyb. Toto téma také popisuje, jak se vyvarovat rozdělení klastru, a udává příklad způsobu opětovného sloučení jednotlivých částí.
| “Hardwarové požadavky pro klastry” na stránce 80
| Pro použití klastrování je kompatibilní jakýkoli model iSeries, který je schopen spouštět operační systém i5/OS verze V4R4M0 nebo novější.

Klastrové aplikace

| Jedním z klíčových prvků v klastrovaném prostředí je odolnost aplikací. Jestliže chcete psát a používat ve svém klastru velmi dostupné aplikace, měli byste zajistit, aby tyto aplikace měly specifické vlastnosti pro dostupnost.

Využijete-li výhod odolných aplikací ve vašem klastru, bude moci být aplikace restartována v jiném klastrovém uzlu, aniž by bylo nutno rekonfigurovat klienty. Navíc budou po přepnutí nebo přepnutí při selhání dostupná data přiřazená k aplikaci. To znamená, že když se aplikace a její data přepnou z primárního uzlu na záložní uzel, u uživatelů aplikace se projeví pouze minimální přerušení nebo vůbec žádné. Koncový uživatel nepotřebuje vědět, že se aplikace a data přesunula na záložní systém.

| Chcete-li dosáhnout ve vašem klastru odolnosti aplikací, je nutno používat aplikace, které splňují určité požadavky na dostupnost. K tomu, aby bylo možné aplikaci přepínat, tzn. aby byla pro uživatele aplikace v klastru vždy dostupná, musí mít jisté charakteristiky. Další informace o těchto vlastnostech aplikaci najdete na webových stránkách High Availability and Clusters . Protože existují tyto požadavky, máte následující možnosti pro používání přepínatelné aplikace ve svém klastru:

1. **Koupit softwarovou aplikaci umožňující použití v klastru**

   Softwarové produkty, které umožňují použití v klastru, splňují jisté požadavky na vysokou dostupnost.
2. **Napište nebo změňte svou vlastní aplikaci tak, aby byla velmi dostupná**

   Nezávislí dodavatelé softwaru a programátoři aplikací mohou upravit aplikace tak, aby se umožnilo jejich přepínání v prostředí klastrů iSeries.

Jakmile jde o odolnou aplikaci, je nutno ji řídit v rámci klastru.

Související pojmy

“Odolné aplikace” na stránce 14
Odolná aplikace je aplikace, která může opětovně spuštěna v jiném klastrovém uzlu, aniž by bylo nutné rekonfigurovat klienty.

Architektura i5/OS pro aplikace umožňující použití v klastru

Aplikace, která je vysoce dostupná, znamená zvýšení hodnoty pro koncové uživatele, neboť tito oceňují, pokud je zachována dostupnost aplikace v případě výpadku systému, plánovaného či neplánovaného.

Systém i5/OS poskytuje architekturu pro odolnost aplikací, která podporuje různou úroveň vysoce dostupných aplikací. V horní části tohoto spektra se nacházejí aplikace vybavené integrovanými funkcemi, které poskytují vlastnosti vysoké dostupnosti a automatizaci vysoce dostupného prostředí, řízenými obslužnými programy pro řízení klastru.

Tyto aplikace mají následující vlastnosti:

- Aplikace je schopna se přepnout na záložní klastrový uzel, když se primární uzel stane nedostupným.
- Aplikace definuje odolné prostředí v tématu Resilient Definition and Status Data Area, aby se umožnila automatická konfigurace a aktivace aplikace pomocí aplikace pro řízení klastru.
- Aplikace realizuje odolnost aplikace prostřednictvím ukončovacího programu CRG, který řídí události související s klastrem, přičemž využívá schopností služeb klastrových prostředků systému i5/OS.
- Aplikace poskytuje funkci pro restart aplikace, která přesune uživatele na obrazovku s menu aplikace nebo dále.

Aplikace, které vykazují vyšší úroveň dostupnosti a možnosti restartu, mají tyto vlastnosti:

- Aplikace poskytuje zdokonalenou odolnost aplikace prostřednictvím robustnějšího zpracování událostí klastru (kódů akce) ukončovacím programem aplikační CRG.
- Aplikace poskytuje vyšší úroveň podpory pro restart. U aplikací typu serveru-centric bude uživatel přesunut na okraj transakce pomocí vázaného zpracování nebo funkcí kontrolního bodu. U aplikací typu client-centric uživatel zaznamená hladké přepnutí s minimálním přerušením služby.

Související pojmy

iSeries - vysoká dostupnost a klastry

Naprogramování vysoce dostupné klastrové aplikace

Vysoce dostupná aplikace je taková aplikace, které je odolná vůči výpadku systému v klastrovém prostředí.

Existuje několik úrovní dostupnosti aplikace:

1. Když dojde k chybě aplikace, aplikace se sama restartuje na stejném uzlu a opraví všechny potenciální příčiny chyby (jako např. poškozená řídicí data). Vám se aplikace bude jevit tak, jako kdyby se spustila poprvé.
2. Aplikace provádí určitou formu zpracování restartu od kontrolního bodu. Vám se aplikace bude jevit tak, jako by se nacházela blízko místa selhání.
3. Dojde-li k výpadku systému, aplikace se restartuje na záložním serveru. Vám se aplikace bude jevit tak, jako kdyby se spustila poprvé.
4. Dojde-li k výpadku systému, aplikace se restartuje na záložním serveru a provádí na serverech určitou formu zpracování restartu od kontrolního bodu. Vám se aplikace bude jevit tak, jako by se nacházela blízko místa selhání.
5. Dojde-li k selhání systému, následuje koordinované přepnutí při selhání jak aplikace tak jejích přiřazených dat na jiný uzel nebo uzly v klastru. Vám se aplikace bude jevit tak, jako kdyby se spustila poprvé.
6. Dojde-li k selhání systému, následuje koordinované přepnutí při selhání jak aplikace tak jejích přiřazených dat na jiný uzel nebo uzly v klastru. Aplikace provádí na serverech určitou formu zpracování restartu od kontrolního bodu. Vám se aplikace bude jevit tak, jako by se nacházela blízko místa selhání.

Poznámka: V případech 1 až 4 jste zodpovědni za obnovu dat.

Jak udělat aplikační programy odolnými:

Zde se dozvíte, jak zajistit, aby aplikační programy byly odolné.

Odolná aplikace by měla mít tyto vlastnosti:

- Aplikaci lze restartovat na daném uzlu nebo v jiném uzlu.
- Klient na aplikaci přistupuje prostřednictvím IP adresy.
- Aplikace nemá stav nebo jsou informace o stavu známé.
- Data přiřazená k aplikaci jsou po přepnutí dostupná.

Tři základní prvky, které způsobují, že je aplikace odolná vůči výpadku systému v prostředí klastru, jsou:

Aplikace samotná

Nakolik je aplikace tolerantní vůči chybám nebo výpadkům systému a jak transparentně se aplikace může sama restartovat?

Toto může aplikace zvládnout pomocí využití klastrových schopností.

Přiřazená data

Když dojde k výpadku, ovlivní to dostupnost libovolných přiřazených dat?

Toto může zvládnout některý replikační produkt obchodních partnerů IBM pro middleware klastrů, který využívá výhod klastrových schopností. Anebo lze data ukládat do přepínatelných nezávislých ASP.

Řídicí schopnosti a administrace

Jak je snadné definovat prostředí, které podporuje dostupnost dat a aplikace?

Toto může zvládnout produkt pro řízení klastrů od nějaké třetí strany, který používá klastrová API a také kombinuje odolné aplikace s odolnými daty.

Restart vysoce dostupných klastrových aplikací:

Při restartu aplikace potřebuje aplikace znát svůj stav v okamžiku přepnutí při selhání nebo přepnutí.

Stavové informace jsou specifické podle aplikace: to, které informace jsou nutné, tudíž určuje aplikace. Bez jakýchkoliv informací o stavu lze aplikaci na PC restartovat. Budete však muset znovu vytvořit vaši pozici v rámci aplikace.

Pro uložení informací o stavu aplikace v záložním systému je k dispozici několik metod. Každá aplikace si musí určit, které metoda bude pro ni fungovat nejlépe.

- Aplikace může přenést všechny informace o stavu na žádající klientský systém. Když dojde k přepnutí při selhání nebo přepnutí, aplikace použije uložený stavu na klientovi k tomu, aby znovu vytvořila stav na novém serveru. Toho lze dosáhnout pomocí některého rozhraní API (Distribute information API nebo Clustered Hash Table API).
- Aplikace může replikovat informace o stavu (jako např. informace o úlohách a další řídicí struktury asociované s aplikací) v reálném čase. Při každé změně ve strukturách aplikace zasílá změnu záložnímu systému.
- Aplikace může ukládat relevantní informace o stavu, které jsou asociované s touto aplikací, do datové části ukončovacího programu skupiny klastrových prostředků pro tuto aplikaci. Tato metoda předpokládá, že se vyžaduje pouze malý objem informací o stavu. Můžete to provést pomocí rozhraní QcstChangeClusterResourceGroup (Change Cluster Resource Group) API.
- Aplikace může ukládat informace o stavu do datového objektu, který se replikuje do záložního systému spolu s daty aplikace.
- Aplikace může ukládat informace o stavu do datového objektu obsaženého v přepínatelném nezávislém ASP, které také obsahuje data aplikace.
- Aplikace může ukládat informace o stavu klienta.
- Informace o stavu se neukládají a vy musíte provést obnovu.

Poznámka: Množství informací, které se musí ukládat, se zmenšuje, jestliže aplikace využívá některou z forem zpracování restartu od kontrolního bodu. Informace o stavu se ukládají pouze v předem stanovených kontrolních bodech aplikace. Při restartu se pak dostanete na poslední známý kontrolní bod, což je podobné tomu, jak funguje vázané zpracování databáze.

Volání ukončovacího programu skupiny klastrových prostředků:

K volání ukončovacího programu skupiny klastrových prostředků (CRG) dochází během různých fází prostředí klastru.

Tento program vytváří potřebnou odolnost prostředí pro prostředky v klastru. U CRG odolných zařízení je ukončovací program volitelný, u jiných typů CRG je však nutný. Použije-li se ukončovací program CRG, je volán na základě události v rámci klastru, konkrétně v těchto případech:

- Uzel neočekávaně opustí klastr.
- Uzel opustí klastr v důsledku použití rozhraní QcstEndClusterNode (End Cluster Node) API nebo QcstRemoveClusterNodeEntry (Remove Cluster Node Entry) API.
- Uzel je vymazán v důsledku použití rozhraní QcstDeleteCluster (Delete Cluster) API.
- Uzel je aktivován pomocí rozhraní QcstStartClusterNode (Start Cluster Node) API.
- Je znovu navázána komunikace s rozděleným uzlem.

Tento ukončovací program:

- Spouští se v označené aktivační skupině nebo aktivační skupině volajícího (*CALLER).
- Ignoruje parametr restartu, jestliže má ukončovací program nezpracovanou výjimku nebo je zrušený.
- Poskytuje obslužný program zrušení.

Když je spuštěno API skupiny klastrových prostředků, je ukončovací program volán ze samostatné úlohy uživatelským profilem zadaným v rozhraní QcstCreateClusterResourceGroup (Create Cluster Resource Group) API. API vytvoří samostatnou úlohu automaticky, když je volán ukončovací program. Jestliže ukončovací program pro CRG dat je neúspěšný nebo skončí abnormálně, je volán ukončovací program skupiny klastrových prostředků ve všech aktivních uzlech v doméně obnovy kódem akce Undo (Storno). Tento kód akce umožňuje, aby se ukončily všechny nedokončené aktivity a obnovil se původní stav skupiny klastrových prostředků.

Jestliže ukončovací program pro aplikační CRG je neúspěšný nebo skončí abnormálně, služby klastrových prostředků se pokusí aplikaci restartovat, pokud je stav CRG aktivní. Ukončovací program skupiny klastrových prostředků je volán kódem akce Restart. Jestliže aplikaci nelze během zadaného maximálním počtu pokusů restartovat, je ukončovací program skupiny klastrových prostředků volán kódem akce Failover (Selhání). Počítání restartů se obnovuje, pouze když je ukončovací program volán kódem akce Start, což může být důsledkem spuštění CRG, přepnutí při selhání nebo přepnutí.

Když je skupina klastrových prostředků (CRG) spuštěna, neměl by ukončovací program aplikační CRG volaný na primárním uzlu vracet řízení službám klastrových prostředků, dokud se aplikace sama neukončí nebo se nevyskytne chyba. Je-li aplikační CRG aktivní, pak když musí služby klastrových prostředků oznámit ukončovací program aplikační CRG nebo jinou událost, spustí se další instance ukončovacího programu v jiné úloze. Očekává se vrácení kteréhokoliv jiného kódu akce než Start nebo Restart.

Když je volán ukončovací program skupiny klastrových prostředků, předává se sada parametrů, které identifikují událost klastru, která se zpracovává, současný stav klastrových prostředků a očekávaný stav klastrových prostředků.

Podrobnější informace o ukončovacích programech skupin klastrových prostředků, včetně popisu informací, které se ukončovacímu programu předávají pro jednotlivé kódy akcí, najdete v tématu Ukončovací programy skupiny klastrových prostředků v dokumentaci k API klastrů.V knihovně QUSRTOOL se nachází vzor zdrojového kódu, který lze použít jako základ pro napsání ukončovacího programu. Vyhledejte si člen TCSTAPPEXT v souboru QATTSYSC.

Pokyny ohledně CRG aplikací

Skupina klastrových prostředků (CRG) aplikace řídí odolnost aplikace.

Správa IP adres převzetí aplikační CRG:

| Můžete spravovat IP adresy převzetí aplikační CRG pomocí služeb klastrových prostředků. Můžete je také spravovat
| ručně.

Existují dva způsoby, jak spravovat IP adresu převzetí aplikace asociovanou s aplikační CRG. Nejjednodušší (předvolený) způsob je ponechat správu IP adresy převzetí službám klastrových prostředků. Při této metodě vytvoří služby klastrových prostředků IP adresu převzetí ve všech uzlech domény obnovy, a to včetně uzlů, které byly k doméně obnovy přidány dodatečně. Je-li vybrána tato metoda, nemůže být IP adresa převzetí v současné době definována v žádném uzlu domény obnovy.

Alternativní způsob je, že budete spravovat IP adresy převzetí sami. Při této metodě nepodnikají služby klastrových prostředků žádné kroky pro konfiguraci IP adresy převzetí; za konfiguraci je odpovědný uživatel. Před spuštěním skupiny klastrových prostředků musíte přidat IP adresu převzetí ve všech uzlech domény obnovy (kromě replikačních uzlů). U každého uzlu přidávaného do domény obnovy aktivní skupiny klastrových prostředků (CRG) musí být IP adresa převzetí nakonfigurována před přidáním uzlu.

Několik podsítí

IP adresa převzetí aplikace může fungovat v několika podsítích, ačkoliv předvoleno je, že všechny uzly domény obnovy jsou ve stejné podsíti. Postup konfigurace IP adresy převzetí aplikace při umístění uzlů domény obnovy ve více podsítích najdete v tématu Povolení přepínání aplikací mezi podsítěmi.

Související pojmy

"Příklad: Akce při přepnutí při selhání aplikační CRG" na stránce 33
Podívejte se, jak funguje ukázkový scénář přepnutí při selhání.

"Vytvoření aplikační CRG s aktivní IP adresou převzetí" na stránce 103
Když vytváříte aplikační CRG, můžete uvést, že chcete umožnit aktivní IP adresu převzetí. To je povoleno pouze tehdy, jestliže uživatel konfiguruje IP adresu převzetí.

Povolení přepínání aplikací mezi podsítěmi:

Klastrování obecně vyžaduje, aby všechny klastrové uzly v doméně obnovy aplikační skupiny klastrových prostředků byly umístěny ve stejné lokální síti (LAN) - aby používaly stejné adresování podsítě.

Jako síťový protokol umožňující přepínání nakonfigurované IP adresy převzetí aplikace z jednoho uzlu domény obnovy na jiný se používá protokol ARP (Address Resolution Protocol). Také je však možné rozšířit doménu obnovy tak, aby zahrnovala klastrové uzly umístěné v jiných lokálních sítích oddělených komerčními směrovači.

Tohoto rozšíření lze dosáhnout použitím podpory virtuálních IP adres a využitím protokolu RIP (Routing Information Protocol) na klastrových uzlech a komerčních směrovačích v síti. Další informace najdete v tématu "Povolení přepínání aplikací".

Povolení přepínání aplikací:

Služby prostředků klastru podporují při konfiguraci aplikační CRG uživatelsky konfigurovatelnou IP adresu převzetí.

Chcete-li umožnit přepínání aplikací, je potřeba provést tyto manuální konfigurační kroky. **Uvedenou sadu pokynů je nutno provést ve všech uzlech v doméně obnovy a zopakovat na dalších uzlech v klastru, které se stanou uzly domény obnovy pro danou aplikační CRG.**

1. Vyberte IP adresu převzetí, kterou bude používat aplikační CRG.
   - Chcete-li se vyvarovat zmatků, neměla by se tato adresa krýt s žádnou jinou existující adresou používanou klastrovými uzly nebo směrovači. Zvolíte-li, například, adresu 19.19.19.19, ověřte si, že adresy 19.0.0.0 (19.19.0.0 nebo ...) nejsou přenosové cesty, které již směrovací tabulka systému zná.
   - Přidejte rozhraní převzetí (například 19.19.19.19); při jeho tvorbě nastavte parametry: popis linky *VIRTUALIP, maska podsítě 255.255.255.255 (předepsaná cesta k hostiteli), maximální přenosová jednotka 1500 (kterékoliv hodnota v rozmezí 576-16388) a automatické spuštění na hodnotu *NO. Tato adresa převzetí (například 19.19.19.19) musí existovat jako adresa *VIRTUALIP předtím, než bude v dalším kroku označena jako přiřazené lokální rozhraní. Nemusí však být aktivní.
2. Přiřaďte plánovanou IP adresu převzetí k jedné nebo více IP adresám označeným pro použití při komunikaci klastru, když vytváříte klastr nebo přidáváte do klastru uzel.
   - To znamená, že adresa převzetí, např. 19.19.19.19, se stane přiřazeným lokálním rozhraním na IP adrese pro klastrový uzel na sběrnici Ethernet, který se bude lokálně používat pro činnost klastru. Toto je nutno provést pro každou adresu klastru na každém klastrovém uzlu.

     Poznámka: Budete-li tuto změnu provádět v příkazu CFGTCP, je nutno adresy klastru ukončit.
3. Vytvořte klastr a vytvořte libovolné CRG. U aplikační CRG do pole 'konfigurace IP adresy převzetí' zadejte QcstUserCfgsTakeoverIpAddr. Nespouštějte žádnou aplikační CRG.
4. V příkazu CFGTCP použijte postupně volby Konfigurace TCP/IP aplikací (volba 20), Konfigurace RouteD (volba 2), Změna atributů RouteD (volba 1) a zjistěte, zda je parametr Dodat nastaven na *YES. Pokud není, nastavte jej na *YES a spusťte nebo restartujte ROUTED (RIP nebo RIP-2) na každém klastrovém uzlu.
   - Příkaz NETSTAT volba 3 zobrazí ROUTED pomocí lokálního portu, je-li aktuálně spuštěný. ROUTED musí spouštět a šířit přenosové cesty (Dodat = *YES) na každém klastrovém uzlu v doméně obnovy CRG.
5. Zajistěte, aby všechny komerční směrovače v síti propojující sítě LAN domény obnovy přijímaly a šířily předepsanou cestu k hostiteli pro RIP.
   - Toto není nutně předvolené nastavení směrovačů. Jazyk se bude lišit podle výrobce směrovače, ale u rozhraní RIP očekávejte zasílání předepsaných cest k hostiteli a přijímání dynamických hostitelů.
   - To se týká rovněž jak rozhraní směrovačů směrujících na servery iSeries, tak rozhraní mezi směrovači.

Poznámka: V tomto scénáři nepoužívejte jako směrovač server iSeries. Použijte nějaký komerční směrovač (IBM nebo jiný), který je určen pro účely směrování. Směrování serverů iSeries nelze nakonfigurovat, aby zvládlo tuto funkci.

6. Nyní můžete manuálně aktivovat adresu převzetí na jednom z uzlů klastru, ponechejte RIP až 5 minut na šíření přenosových cest a otestujte spojení na adresu převzetí ze všech uzlů v doméně obnovy CRG a z vybraných klientů v sítích LAN, kteří budou tuto adresu používat.
   - Po tomto ověřovacím testu zajistěte, aby se adresa přepnutí znovu ukončila.
   - Klastr spustí adresu na zadaném primárním uzlu, když se spustí CRG.
7. Spusťte aplikační CRG.
   - Klastr nyní spustí adresu převzetí na zadaném preferovaném uzlu a RIP bude šířit přenosové cesty v celé doméně obnovy. Aktualizace přenosových cest v doméně může RIP zabrat až 5 minut. Funkce RIP je nezávislá na funkci spuštění CRG.

Důležité poznámky:

- Jestliže se výše uvedená procedura neprovede ve všech uzlech klastru v doméně obnovy aplikační CRG, klastr se během procesu přepnutí zastaví.
- Přestože se při selhání nepřepínáme na replikační uzly, je vhodné provést tuto proceduru i na replikačních uzlech pro případ, že by někdy později byly změněny na záložní uzly.
- Jestliže chcete používat více virtuálních IP adres, pak bude každá z nich vyžadovat samostatnou aplikační CRG a samostatnou IP adresu, se kterou bude asociována. Touto adresou může být další logická IP adresa na stejném fyzickém adaptéru nebo to může být úplně jiný fyzický adaptér. Také je potřeba dát pozor na to, aby se předešlo dvojznačnostem ve směrovacích tabulkách. Nejlépe toho docílíte takto:
  - Přidejte *DFTROUTE do směrovací tabulky pro každou virtuální IP adresu.
  - Můžete to provést pomocí příkazu CFGTCP (volba 2).
  - Nastavte všechny parametry, včetně následného směrovacího uzlu, stejně pro dosažení zvoleného směrovače, ale jako preferované rozhraní vázaného zpracování byste měli nastavit IP adresu lokálního systému asociovanou s virtuální IP adresou, která bude reprezentována touto přenosovou cestou.

Příklad: Akce při přepnutí při selhání aplikační CRG:

Podívejte se, jak funguje ukázkový scénář přepnutí při selhání.

Pokud dojde k selhání skupiny klastrových prostředků (CRG) aplikace v důsledku překročení limitu nebo zrušení úlohy, následují tyto akce:

- Ve všech uzlech v doméně obnovy pro CRG je pomocí kódu akce Failover (Selhání) volán ukončovací program skupiny klastrových prostředků. To znamená, že se služby klastrových prostředků připravují na přepnutí přístupového bodu aplikace na první záložní uzel.
- Služby klastrových prostředků ukončí převzaté IP spojení na primárním uzlu. Další informace o IP adrese převzetí najdete v tématu Správa IP adres CRG aplikací.
- Služby klastrových prostředků spustí IP adresu převzetí na prvním záložním (novém primárním) uzlu.
- Služby klastrových prostředků spustí pomocí kódu akce Start (Spuštění) úlohu, která volá ukončovací program skupiny klastrových prostředků pouze na novém primárním uzlu. Tato akce restartuje aplikaci.

Výše uvedený příklad ukazuje, jak funguje jeden ze scénářů přepnutí při selhání. Jiné scénáře přepnutí při selhání mohou fungovat odlišně.

Příklad: Ukončovací program aplikace:

Tento příklad kódu obsahuje vzorový ukončovací program skupiny klastrových prostředků.

Tento příklad kódu najdete v knihovně QUSRTOOL.

Používáním příkladů kódu vyjadřujete svůj souhlas s podmínkami uvedenými v části Licence na kód a vyloučení záruky.

```
/*********************************************************************/
/*                                                                   */
/* Library:  QUSRTOOL                                                */
/* File:     QATTSYSC                                                */
/* Member:   TCSTAPPEXT                                              */
/* Type:     ILE C                                                   */
/*                                                                   */
/* Description:                                                      */
/* This is an example application CRG exit program which gets called for   */
/* various cluster events or cluster APIs. The bulk of the logic must     */
/* still be added because that logic is really dependent upon the unique   */
/* things that need to be done for a particular application.             */
/*                                                                   */
/* The intent of this example to to provide a shell which contains the     */
/* basics for building a CRG exit program. Comments throughout the example*/
/* highlight the kinds of issues that need to be addressed by the real     */
/* exit program implementation.                                         */
/*                                                                   */
/* Every action code that applies to an application CRG is handled in this */
/* example.                                                          */
/*                                                                   */
/* The tcstdtaara.h include is also shipped in the QUSRTOOL library.  See  */
/* the TCSTDTAARA member in the QATTSYSC file.                       */
/*                                                                   */
/* Change log:                                                       */
/* Flag Reason   Ver    Date   User Id   Description                 */
/* ____ _______ _____ ______ _______ ____________________________*/
/* ...  D98332   v5r1m0 000509 ROCH     Initial creation.            */
/* $A1  P9950070 v5r2m0 010710 ROCH     Dataarea fixes               */
/* $A2  D99055   v5r2m0 010913 ROCH     Added CancelFailover action code   */
/* $A3  D98854   v5r2m0 010913 ROCH     Added VerificationPhase action code*/
/* $A4  P9A10488 v5r3m0 020524 ROCH     Added example code to wait for data*/
/*                                      CRGs on switchover action code     */
/*                                                                   */
/*********************************************************************/


/*-------------------------------------------------------------------*/
/*                                                                   */
/* Header files                                                      */
/*                                                                   */
/*-------------------------------------------------------------------*/
#include             /* Useful when debugging                  */
#include            /* offsetof macro                          */
#include            /* system function                         */
#include            /* String functions                        */
#include            /* Exception handling constants/structures  */
#include              /* Various cluster constants              */
#include          /* Structure of CRG information            */
#include "qusrtool/qattsysc/tcstdtaara" /* QCSTHAAPPI/QCSTHAAPPO data areas*/
#include          /* API to Retrieve contents of a data area  */
#include             /* API error code type definition         */
#include             /* mitime builtin                         */
#include          /* waittime builtin                        */


/*-------------------------------------------------------------------*/
/*                                                                   */
/* Constants                                                         */
/*                                                                   */
/*-------------------------------------------------------------------*/
#define UnknownRole -999
```

```
#define DependCrgDataArea "QCSTHAAPPO"
#define ApplCrgDataArea "QCSTHAAPPI"
#define Nulls 0x0000000000000000000


/*--------------------------------------------------------------------------*/
/*                                                                          */
/* The following constants are used in the checkDependCrgDataArea()         */
/* function.  The first defines how long to sleep before checking the data */
/* area.  The second defines that maximum time to wait for the data area    */
/* to become ready before failing to start the application when the Start   */
/* CRG function is being run.  The third defines the maximum wait time for */
/* the Initiate Switchover or failover functions.                           */
/*                                                                          */
/*--------------------------------------------------------------------------*/
#define WaitSecondsIncrement 30
#define MaxStartCrgWaitSeconds 0
#define MaxWaitSeconds 900

/*--------------------------------------------------------------------------*/
/*                                                                          */
/* As this exit program is updated to handle new action codes, change the   */
/* define below to the value of the highest numbered action code that is    */
/* handled.                                                                 */
/*                                                                          */
/*--------------------------------------------------------------------------*/
#define MaxAc 21

/*--------------------------------------------------------------------------*/
/*                                                                          */
/* If the exit program data in the CRG has a particular structure to it,    */
/* include the header file for that structure definition and change the     */
/* define below to use that structure name rather than char.                */
/*                                                                          */
/*--------------------------------------------------------------------------*/
#define EpData char

/*--------------------------------------------------------------------------*/
/*                                                                          */
/* Change the following define to the library the application resides in    */
/* and thus where the QCSTHAAPPO and QCSTHAAPPI data areas will be found.   */
/*                                                                          */
/*--------------------------------------------------------------------------*/
#define ApplLib "QGPL"


/*--------------------------------------------------------------------------*/
/*                                                                          */
/* Prototypes for internal functions.                                       */
/*                                                                          */
/*--------------------------------------------------------------------------*/
static int getMyRole(Qcst_EXTP0100_t *, int, int);
#pragma argopt(getMyRole)
static int doAction(int, int, int, Qcst_EXTP0100_t *, EpData *);
#pragma argopt(doAction)
static int createCrg(int, int, Qcst_EXTP0100_t *, EpData *);
static int startCrg(int, int, Qcst_EXTP0100_t *, EpData *);
static int restartCrg(int, int, Qcst_EXTP0100_t *, EpData *);
static int endCrg(int, int, Qcst_EXTP0100_t *, EpData *);
static int verifyPhase(int, int, Qcst_EXTP0100_t *, EpData *);
static int deleteCrg(int, int, Qcst_EXTP0100_t *, EpData *);
static int memberIsJoining(int, int, Qcst_EXTP0100_t *, EpData *);
static int memberIsLeaving(int, int, Qcst_EXTP0100_t *, EpData *);
static int switchPrimary(int, int, Qcst_EXTP0100_t *, EpData *);
static int addNode(int, int, Qcst_EXTP0100_t *, EpData *);
static int rmvNode(int, int, Qcst_EXTP0100_t *, EpData *);
static int chgCrg(int, int, Qcst_EXTP0100_t *, EpData *);
```

```
static int deleteCrgWithCmd(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoPriorAction(int, int, Qcst_EXTP0100_t *, EpData *);
static int endNode(int, int, Qcst_EXTP0100_t *, EpData *);
static int chgNodeStatus(int, int, Qcst_EXTP0100_t *, EpData *);
static int cancelFailover(int, int, Qcst_EXTP0100_t *, EpData *);
static int newActionCode(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoCreateCrg(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoStartCrg(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoEndCrg(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoMemberIsJoining(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoMemberIsLeaving(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoSwitchPrimary(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoAddNode(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoRmvNode(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoChgCrg(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoCancelFailover(int, int, Qcst_EXTP0100_t *, EpData *);
static void bldDataAreaName(char *, char *, char *);
#pragma argopt(bldDataAreaName)
static int checkDependCrgDataArea(unsigned int);
#pragma argopt(checkDependCrgDataArea)
static void setApplCrgDataArea(char);
#pragma argopt(setApplCrgDataArea)
static void cancelHandler(_CNL_Hndlr_Parms_T *);
static void unexpectedExceptionHandler(_INTRPT_Hndlr_Parms_T *);
static void endApplication(unsigned int, int, int, Qcst_EXTP0100_t *, EpData *);
#pragma argopt(endApplication)

/*--------------------------------------------------------------------------*/
/*                                                                          */
/* Some debug routines                                                      */
/*                                                                          */
/*--------------------------------------------------------------------------*/
static void printParms(int, int, int, Qcst_EXTP0100_t *, EpData *);
static void printActionCode(unsigned int);
static void printCrgStatus(int);
static void printRcvyDomain(char *,
                            unsigned int,
                            Qcst_Rcvy_Domain_Array1_t *);
static void printStr(char *, char *, unsigned int);


/*--------------------------------------------------------------------------*/
/*                                                                          */
/* Type definitions                                                         */
/*                                                                          */
/*--------------------------------------------------------------------------*/

/*--------------------------------------------------------------------------*/
/*                                                                          */
/* This structure defines data that will be passed to the exception and     */
/* cancel handlers.  Extend it with information unique to your application.*/
/*                                                                          */
/*--------------------------------------------------------------------------*/
typedef struct {
  int *retCode;             /* Pointer to return code                     */
  EpData *epData;           /* Exit program data from the CRG             */
  Qcst_EXTP0100_t *crgData; /* CRG data                                   */
  unsigned int actionCode;  /* The action code                            */
  int role;                 /* This node's recovery domain role           */
  int priorRole;            /* This node's prior recovery domainrole      */
} volatile HandlerDataT;


/*--------------------------------------------------------------------------*/
/*                                                                          */
/* Function pointer array for handling action codes.  When the exit program*/
/* is updated to handle new action codes, add the new function names to     */
```

```
/* this function pointer array.                                          */
/*                                                                       */
/*-----------------------------------------------------------------------*/
static int (*fcn[MaxAc+1]) (int role,
                            int priorRole,
                            Qcst_EXTP0100_t *crgData,
                            EpData *epData) = {
  newActionCode,       /* 0 - currently reserved */
  createCrg,       /* 1  */
  startCrg,        /* 2  */
  restartCrg,      /* 3  */
  endCrg,          /* 4  */
  verifyPhase,     /* 5 - currently reserved */
  newActionCode,       /* 6 - currently reserved */
  deleteCrg,       /* 7  */
  memberIsJoining, /* 8  */
  memberIsLeaving, /* 9  */
  switchPrimary,   /* 10 */
  addNode,         /* 11 */
  rmvNode,         /* 12 */
  chgCrg,          /* 13 */
  deleteCrgWithCmd,/* 14 */
  undoPriorAction, /* 15 */
  endNode,         /* 16 */
  newActionCode,       /* 17 - applies only to a device CRG */
  newActionCode,       /* 18 - applies only to a device CRG */
  newActionCode,       /* 19 - applies only to a device CRG */
  chgNodeStatus,   /* 20 */
  cancelFailover   /* 21 */
};


/*-----------------------------------------------------------------------*/
/*                                                                       */
/* Function pointer array for handling prior action codes when called with */
/* the Undo action code.  When the exit program is updated to handle       */
/* Undo for new action codes, add the new function names to this function  */
/* pointer array.                                                        */
/*                                                                       */
/*-----------------------------------------------------------------------*/
static int (*undoFcn[MaxAc+1]) (int role,
                                int priorRole,
                                Qcst_EXTP0100_t *crgData,
                                EpData *epData) = {
  newActionCode,       /* 0 - currently reserved */
  undoCreateCrg,       /* 1  */
  undoStartCrg,        /* 2  */
  newActionCode,       /* 3  */
  undoEndCrg,          /* 4  */
  newActionCode,       /* 5 - no undo for this action code */
  newActionCode,       /* 6 - currently reserved */
  newActionCode,       /* 7  */
  undoMemberIsJoining, /* 8  */
  undoMemberIsLeaving, /* 9  */
  undoSwitchPrimary,   /* 10 */
  undoAddNode,         /* 11 */
  undoRmvNode,         /* 12 */
  undoChgCrg,          /* 13 */
  newActionCode,       /* 14 */
  newActionCode,       /* 15 */
  newActionCode,       /* 16 */
  newActionCode,       /* 17 - applies only to a device CRG */
  newActionCode,       /* 18 - applies only to a device CRG */
  newActionCode,       /* 19 - applies only to a device CRG */
  newActionCode,       /* 20 */
  undoCancelFailover   /* 21 */
};
```

```
/*********************************************************************/
/*                                                                   */
/* This is the entry point for the exit program.                     */
/*                                                                   */
/*********************************************************************/
void main(int argc, char *argv[]) {

  HandlerDataT hdlData;


/*-------------------------------------------------------------------*/
  /*                                                                 */
  /* Take each of the arguments passed in the argv array and castit to */
  /* the correct data type.                                          */
  /*                                                                 */

/*-------------------------------------------------------------------*/
  int *retCode    = (int *)argv[1];
  unsigned int *actionCode = (unsigned int *)argv[2];
  EpData *epData           = (EpData *)argv[3];
  Qcst_EXTP0100_t *crgData = (Qcst_EXTP0100_t *)argv[4];
  char *formatName         = (char *)argv[5];


/*-------------------------------------------------------------------*/
  /*                                                                 */
  /* Ensure the format of the data being passed is what we areexpecting. */
  /* If not, a change has been made and this exit program needs tobe */
  /* updated to accomodate the change.  Add appropriate errorlogging for */
  /* your application design.                                        */
  /*                                                                 */

/*-------------------------------------------------------------------*/
  if (0 != memcmp(formatName, "EXTP0100", 8))
    abort();


/*-------------------------------------------------------------------*/
  /*                                                                 */
  /* Set up the data that will be passed to the exception andcancel  */
  /* handlers.                                                       */
  /*                                                                 */

/*-------------------------------------------------------------------*/
  hdlData.retCode    = retCode;
  hdlData.epData     = epData;
  hdlData.crgData    = crgData;
  hdlData.actionCode = *actionCode;
  hdlData.role       = UnknownRole;
  hdlData.priorRole  = UnknownRole;
  _VBDY();  /* force changed variables to home storage location      */


/*-------------------------------------------------------------------*/
  /*                                                                 */
  /* Enable an exception handler for any and all exceptions.         */
  /*                                                                 */

/*-------------------------------------------------------------------*/
#pragma exception_handler(unexpectedExceptionHandler, hdlData, \
                          _C1_ALL, _C2_ALL, _CTLA_INVOKE )


/*-------------------------------------------------------------------*/
```

```
  /*                                                                  */
  /* Enable a cancel handler to recover if this job is canceled.      */
  /*                                                                  */

/*------------------------------------------------------------------*/
#pragma cancel_handler(cancelHandler, hdlData)


/*------------------------------------------------------------------*/
  /*                                                                  */
  /* Extract the role and prior role of the node this exit program is */
  /* running on.  If the cluster API or event changes the recovery domain */
  /* (node role or membership status), the new recovery domain's offset is */
  /* passed in Offset_Rcvy_Domain_Array and the offset of the recovery */
  /* domain as it looked prior to the API or cluster event is passed in */
  /* Offset_Prior_Rcvy_Domain_Array.  If the recovery domain isn't changed,*/
  /* only Offset_Rcvy_Domain_Array can be used to address the recovery */
  /* domain.                                                          */
  /*                                                                  */

/*------------------------------------------------------------------*/
  hdlData.role = getMyRole(crgData,
                           crgData->Offset_Rcvy_Domain_Array,
                           crgData->Number_Nodes_Rcvy_Domain);
  if (crgData->Offset_Prior_Rcvy_Domain_Array)
    hdlData.priorRole =
                getMyRole(crgData,

crgData->Offset_Prior_Rcvy_Domain_Array,

crgData->Number_Nodes_Prior_Rcvy_Domain);
  else
    hdlData.priorRole = hdlData.role;
  _VBDY();  /* force changed variables to home storage location     */


/*------------------------------------------------------------------*/
  /*                                                                  */
  /* Enable the following to print out debug information.             */
  /*                                                                  */

/*------------------------------------------------------------------*/
  /*
  printParms(*actionCode, hdlData.role, hdlData.priorRole, crgData,
epData);
  */


/*------------------------------------------------------------------*/
  /*                                                                  */
  /* Do the correct thing based upon the action code.  The return code */
  /* is set to the function result of doAction().                     */
  /*                                                                  */

/*------------------------------------------------------------------*/
  *retCode = doAction(*actionCode,
                      hdlData.role,
                      hdlData.priorRole,
                      crgData,
                      epData);


/*------------------------------------------------------------------*/
  /*                                                                  */
  /* The exit program job will end when control returns to the operating */
  /* system at this point.                                            */
  /*                                                                  */
```

```
/*--------------------------------------------------------------------*/
  return;

#pragma disable_handler  /* unexpectedExceptionHandler                 */
#pragma disable_handler  /* cancelHandler                              */
}  /* end main()                                                       */


/*********************************************************************/
/*                                                                   */
/* Get the role of this particular node from one of the views of the */
/*     recovery domain.                                              */
/*                                                                   */
/* APIs and cluster events which pass the updated and prior recovery domain*/
/* to the exit program are:                                          */
/* QcstAddNodeToRcvyDomain                                           */
/* QcstChangeClusterNodeEntry                                        */
/* QcstChangeClusterResourceGroup                                    */
/* QcstEndClusterNode (ending node does not get the prior domain)    */
/* QcstInitiateSwitchOver                                            */
/* QcstRemoveClusterNodeEntry (removed node does not get the prior domain) */
/* QcstRemoveNodeFromRcvyDomain                                      */
/* QcstStartClusterResourceGroup (only if inactive backup nodes are  */
/*                                reordered)                         */
/* a failure causing failover                                        */
/* a node rejoining the cluster                                      */
/* cluster partitions merging                                        */
/*                                                                   */
/* All other APIs pass only the updated recovery domain.             */
/*                                                                   */
/*********************************************************************/
static int getMyRole(Qcst_EXTP0100_t *crgData, int offset, int count) {

  Qcst_Rcvy_Domain_Array1_t *nodeData;
  unsigned int iter = 0;


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Under some circumstances, the operating system may not be able to     */
  /* determine the ID of this node and passes *NONE.  An example of such a */
  /* circumstance is when cluster resource services is not active on a     */
  /* node and the DLTCRG CL command is used.                          */
  /*                                                                  */

/*--------------------------------------------------------------------*/
  if (0 == memcmp(crgData->This_Nodes_ID, QcstNone,
sizeof(Qcst_Node_Id_t)))
    return UnknownRole;


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Compute a pointer to the first element of the recovery domain array.  */
  /*                                                                  */

/*--------------------------------------------------------------------*/
  nodeData = (Qcst_Rcvy_Domain_Array1_t *)((char *)crgData + offset);


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Find my node in the recovery domain array.  I will not be in the      */
  /* prior recovery domain if I am being added by the Add Node to Recovery */
  /* Domain API.                                                      */
  /*                                                                  */
```

```
/*--------------------------------------------------------------------*/
  while (  0 != memcmp(crgData->This_Nodes_ID,
                       nodeData->Node_ID,
                       sizeof(Qcst_Node_Id_t))
         &&
           iter &lt; count
        ) {
    nodeData++;
    iter++;
  }

  if (iter &lt; count)
    return nodeData->Node_Role;
  else
    return UnknownRole;
}  /* end getMyRole()                                                 */


/**********************************************************************/
/*                                                                    */
/* Call the correct function based upon the cluster action code.  The */
/* doAction() function was split out from main() in order to clarify the */
/* example.  See the function prologues for each called function for  */
/* information about a particular cluster action.                     */
/*                                                                    */
/* Each action code is split out into a separate function only to help */
/* clarify this example.  For a particular exit program, some action codes */
/* may perform the same function in which case multiple action codes could */
/* be handled by the same function.                                   */
/*                                                                    */
/**********************************************************************/
static int doAction(int actionCode,
                    int role,
                    int priorRole,
                    Qcst_EXTP0100_t *crgData,
                    EpData *epData) {


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* For action codes this exit program knows about, call a function to */
  /* do the work for that action code.                                */
  /*                                                                  */

/*--------------------------------------------------------------------*/

  if (actionCode &lt;= MaxAc )
    return (*fcn[actionCode]) (role, priorRole, crgData, epData);
  else

/*--------------------------------------------------------------------*/
    /*                                                                */
    /* IBM has defined a new action code in a new operating system release */
    /* and this exit program has not yet been updated to handle it.  Take a*/
    /* default action for now.                                        */
    /*                                                                */

/*--------------------------------------------------------------------*/
    return newActionCode(role, priorRole, crgData, epData);
}  /* end doAction()                                                  */


/**********************************************************************/
/*                                                                    */
/* Action code = QcstCrgAcInitialize                                  */
/*                                                                    */
```

```
/* The QcstCreateClusterResourceGroup API was called.  A new cluster      */
/* resource group object is being created.                                */
/*                                                                        */
/* Things to consider:                                                    */
/*   - Check that the application program and all associated objects are on*/
/*     the primary and backup nodes.  If the objects are not there,       */
/*     consider sending error/warning messages or return a failure return  */
/* code.                                                                  */
/*   - Check that required data or device CRGs are on all nodes in the    */
/*     recovery domain.                                                   */
/*   - Perform any necessary setup that is required to run the       */
/*     the application on the primary or backup nodes.                    */
/*   - If this CRG is enabled to use the QcstDistributeInformation API,   */
/*     the user queue needed by that API could be created at this time.   */
/*                                                                        */
/**************************************************************************/
static int createCrg(int role,
                     int doesNotApply,
                     Qcst_EXTP0100_t *crgData,
                     EpData *epData) {

  return QcstSuccessful;
}  /* end createCrg()                                                     */


/**************************************************************************/
/*                                                                        */
/* Action code = QcstCrgAcStart                                           */
/*                                                                        */
/* The QcstStartClusterResourceGroup API was called.  A cluster resource  */
/* group is being started.                                                */
/* The QcstInitiateSwitchOver API was called and this is the second action */
/* code being passed to the exit program.                                 */
/* The fail over event occurred and this is the second action code being  */
/* passed to the exit program.                                            */
/*                                                                        */
/* A maximum wait time is used when checking to see if all dependent CRGs  */
/* are active.  This is a short time if the CRG is being started because of*/
/* the QcstStartClusterResourceGroup API.  It is a longer time if it is   */
/* because of a failover or switchover.  When failover or switchover are  */
/* being done, it make take a while for data or device CRGs to become     */
/* ready so the wait time is long.  If the Start CRG API is being used, the*/
/* dependent CRGs should already be started or some error occurred, the   */
/* CRGs were started out of order, etc. and there is no need for a long   */
/* wait.                                                                  */
/*                                                                        */
/* Things to consider:                                                    */
/*   - If this node's role is primary, the application should be started.  */
/*     This exit program should either call the application so that it runs*/
/*     in this same job or it should monitor any job started by this      */
/*     exit program so the exit program knows when the application job    */
/*     ends.  By far, the simplest approach is run the application in this */
/*     job by calling it.                                                 */
/*     Cluster Resource Services is not expecting this exit program to    */
/*     return until the application finishes running.                     */
/*   - If necessary, start any associated subsystems, server jobs, etc.   */
/*   - Ensure that required data CRGs have a status of active on all nodes */
/*     in the recovery domain.                                            */
/*                                                                        */
/**************************************************************************/
static int startCrg(int role,
                    int doesNotApply,
                    Qcst_EXTP0100_t *crgData,
                    EpData *epData) {

  unsigned int maxWaitTime;
```

```
  /* Start the application if this node is the primary                    */
  if (role == QcstPrimaryNodeRole) {

/*--------------------------------------------------------------------*/
    /*                                                                  */
    /* Determine if all CRGs that this application CRG is dependent upon*/
    /* are ready.  If the check fails, return from the Start action code.*/
    /* Cluster Resource Services will change the state of the CRG to    */
    /* Inactive.                                                        */
    /*                                                                  */

/*--------------------------------------------------------------------*/
    if (crgData->Cluster_Resource_Group_Status == QcstCrgStartCrgPending)
      maxWaitTime = MaxStartCrgWaitSeconds;
    else
      maxWaitTime = MaxWaitSeconds;
    if (QcstSuccessful != checkDependCrgDataArea(maxWaitTime))
      return QcstSuccessful;



/*--------------------------------------------------------------------*/
    /*                                                                  */
    /* Just before starting the application, update the data area to    */
    /* indicate the application is running.                             */
    /*                                                                  */

/*--------------------------------------------------------------------*/
    setApplCrgDataArea(Appl_Running);


/*--------------------------------------------------------------------*/
    /*                                                                  */
    /* Add logic to call application here.  It is expected that control */
    /* will not return until something causes the application to end: a */
    /* normal return from the exit program, the job is canceled, or an  */
    /* unhandled exception occurs.  See the cancelHandler() function for */
    /* some common ways this job could be canceled.                     */
    /*                                                                  */

/*--------------------------------------------------------------------*/




/*--------------------------------------------------------------------*/
    /*                                                                  */
    /* After the application has ended normally, update the data area to */
    /* indicate the application is no longer running.                   */
    /*                                                                  */

/*--------------------------------------------------------------------*/
    setApplCrgDataArea(Appl_Ended);
  }
  else

/*--------------------------------------------------------------------*/
    /*                                                                  */
    /* On backup or replicate nodes, mark the status of the application in */
    /* the data area as not running.                                    */
    /*                                                                  */

/*--------------------------------------------------------------------*/
    setApplCrgDataArea(Appl_Ended);


  return QcstSuccessful;
```

```
}  /* end startCrg()                                                      */


/*********************************************************************/
/*                                                                   */
/* Action code = QcstCrgAcRestart                                    */
/*                                                                   */
/* The previous call of the exit program failed and set the return   */
/* code to QcstFailWithRestart or it failed due to an exception and the    */
/* exception was allowed to percolate up the call stack.  In either  */
/* case, the maximum number of times for restarting the exit program has   */
/* not been reached yet.                                             */
/*                                                                   */
/* This action code is passed only to application CRG exit programs which  */
/* had been called with the Start action code.                       */
/*                                                                   */
/*********************************************************************/
static int restartCrg(int role,
                      int doesNotApply,
                      Qcst_EXTP0100_t *crgData,
                      EpData *epData) {


/*-------------------------------------------------------------------*/
  /*                                                                 */
  /* Perform any unique logic that may be necessary when restarting the    */
  /* application after a failure and then call the startCrg() function to  */
  /* do the start functions.                                         */
  /*                                                                 */

/*-------------------------------------------------------------------*/


  return startCrg(role, doesNotApply, crgData, epData);
}  /* end restartCrg()                                                */


/*********************************************************************/
/*                                                                   */
/* Action code = QcstCrgAcEnd                                        */
/*                                                                   */
/* The end action code is used for one of the following reasons:     */
/*   - The QcstEndClusterResourceGroup API was called.               */
/*   - The cluster has become partitioned and this node is in the secondary*/
/*     partition.  The End action code is used regardless of whether the   */
/*     CRG was active or inactive.  Action code dependent data of    */
/*     QcstPartitionFailure will also be passed.                     */
/*   - The application ended.   Action code dependent data of        */
/*     QcstResourceEnd will also be passed.  All nodes in the recovery     */
/*     domain will see the same action code (including the primary).       */
/*   - The CRG job has been canceled.  The exit program on this node will */
/*     be called with the End action code.  QcstMemberFailure will be      */
/*     passed as action code dependent data.                         */
/*                                                                   */
/*                                                                   */
/*                                                                   */
/* Things to consider:                                               */
/*   - If the CRG is active, the job running the application is canceled  */
/*     and the IP takeover address is ended AFTER the exit program is      */
/*     called.                                                       */
/*   - If subsystems or server jobs were started as a result of the        */
/*     QcstCrgAcStart action code, end them here or consolidate all logic  */
/*     to end the application in the cancelHandler() since it will be      */
/*     invoked for all Cluster Resource Services APIs which must end the   */
/*     application on the current primary.                           */
/*                                                                   */
/*********************************************************************/
```

```
static int endCrg(int role,
                  int priorRole,
                  Qcst_EXTP0100_t *crgData,
                  EpData *epData) {


/*--------------------------------------------------------------------------*/
  /*                                                                        */
  /* End the application if it is running on this node.               */
  /*                                                                        */

/*--------------------------------------------------------------------------*/
  endApplication(QcstCrgAcRemoveNode, role, priorRole, crgData,
epData);

  return QcstSuccessful;
}  /* end endCrg()                                                         */


/****************************************************************************/
/*                                                                          */
/* Action code = QcstCrgAcVerificationPhase                                 */
/*                                                                          */
/* The verification phase action code is used to allow the exit program to */
/* do some verification before proceeding with the requested function      */
/* identified by the action code depended data. If the exit program        */
/* determines that the requested function cannot proceed it should return  */
/* QcstFailWithOutRestart.                                                  */
/*                                                                          */
/*                                                                          */
/* NOTE: The exit program will NOT be called with Undo action code.        */
/*                                                                          */
/****************************************************************************/
static int verifyPhase(int role,
                       int doesNotApply,
                       Qcst_EXTP0100_t *crgData,
                       EpData *epData) {


/*--------------------------------------------------------------------------*/
  /*                                                                        */
  /* Do verification                                                        */
  /*                                                                        */

/*--------------------------------------------------------------------------*/
  if (crgData->Action_Code_Dependent_Data == QcstDltCrg) {
         /* do verification  */
        /* if ( fail ) */
          /* return QcstFailWithOutRestart */
  }

  return QcstSuccessful;
}  /* end verifyPhase()                                                    */


/****************************************************************************/
/*                                                                          */
/* Action code = QcstCrgAcDelete                                            */
/*                                                                          */
/* The QcstDeleteClusterResourceGroup or QcstDeleteCluster API was called. */
/* A cluster resource group is being deleted while Cluster Resource        */
/* Services is active.                                                      */
/* If the QcstDeleteCluster API was used, action code dependent data of    */
/* QcstDltCluster is passed.                                                */
/* If the QcstDeleteCluster API was used and the CRG is active, the exit   */
/* program job which is still active for the Start action code is canceled*/
/* after the Delete action code is processed.                               */
```

```
/*                                                                    */
/* Things to consider:                                                */
/*   - Delete application programs and objects from nodes where they are   */
/*     no longer needed such as backup nodes.  Care needs to be exercised  */
/*     when deleting application objects just because a CRG is being       */
/*     deleted since a particular scenario may want to leave the           */
/*     application objects on all nodes.                                   */
/*                                                                    */
/**********************************************************************/
static int deleteCrg(int role,
                     int doesNotApply,
                     Qcst_EXTP0100_t *crgData,
                     EpData *epData) {

  return QcstSuccessful;
}  /* end deleteCrg()                                                 */


/**********************************************************************/
/*                                                                    */
/* Action code = QcstCrgAcReJoin                                      */
/*                                                                    */
/* One of three things is occurring-                                  */
/*  1. The problem which caused the cluster to become partitioned has been */
/*     corrected and the 2 partitions are merging back together to become  */
/*     a single cluster.  Action code dependent data of QcstMerge will be  */
/*     passed.                                                        */
/*  2. A node which either previously failed or which was ended has had    */
/*     cluster resource services started again and the node is joining the */
/*     cluster.  Action code dependent data of QcstJoin will be passed.    */
/*  3. The CRG job on a particular node which may have been canceled or   */
/*     ended has been restarted.  Action code dependent data of QcstJoin   */
/*     will be passed.                                                */
/*                                                                    */
/* Things to consider:                                                */
/*   - If the application replicates application state information to other*/
/*     nodes when the application is running, this state information will  */
/*     need to be resynchronized with the joining nodes if the CRG is      */
/* active.                                                            */
/*   - Check for missing application objects on the joining nodes.         */
/*   - Ensure the required data CRGs are on the joining nodes.            */
/*   - If the application CRG is active, ensure the required data CRGs are */
/* active.                                                            */
/*                                                                    */
/**********************************************************************/
static int memberIsJoining(int role,
                           int priorRole,
                           Qcst_EXTP0100_t *crgData,
                           EpData *epData) {


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Ensure the data area status on this node starts out indicating    */
  /* the application is not running if this node is not the primary.     */
  /*                                                                  */

/*--------------------------------------------------------------------*/
  if (role != QcstPrimaryNodeRole) {
    setApplCrgDataArea(Appl_Ended);
  }


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* If a single node is rejoining the cluster, you may do a certain set of*/
  /* actions.  Whereas if the nodes in a cluster which became partitioned  */
```

```
  /* are merging back together, you may have a different set of actions.   */
  /*                                                                       */

/*-----------------------------------------------------------------------*/
  if (crgData->Action_Code_Dependent_Data == QcstJoin) {
    /* Do actions for a node joining.                                      */
  }
  else {
    /* Do actions for partitions merging.                                  */
  }

  return QcstSuccessful;
}  /* end memberIsJoining()                                              */


/*************************************************************************/
/*                                                                       */
/* Action code = QcstCrgAcFailover                                       */
/*                                                                       */
/* Cluster resource services on a particular node(s) has failed or ended */
/* for this cluster resource group.  The Failover action code is passed  */
/* regardless of whether the CRG is active or inactive.  Failover can    */
/* happen for a number of reasons:                                       */
/*                                                                       */
/*   - an operator canceled the CRG job on a node.  Action code dependent */
/*     data of QcstMemberFailure will be passed.                         */
/*   - cluster resource services was ended on the node (for example, the */
/*     QSYSWRK subsystem was ended with CRS still active).  Action code  */
/*     dependent data of QcstNodeFailure will be passed.                 */
/*   - the application for an application CRG has failed on the primary  */
/*     node and could not be restarted there.  The CRG is Active.        */
/*     Action code dependent data of QcstApplFailure will be passed.     */
/*   - the node failed (such as a power failure).  Action code dependent */
/*     data of QcstNodeFailure will be passed.                           */
/*   - The cluster has become partitioned due to some communication failure*/
/*     such as a communication line or LAN failure.  The Failover action */
/*     code is passed to recovery domain nodes in the majority partition.  */
/*     Nodes in the minority partition see the End action code.  Action  */
/*     code dependent data of QcstPartitionFailure will be passed.       */
/*   - A node in the CRG's recovery domain is being ended with the       */
/*     QcstEndClusterNode API.  The node being ended will see the End Node */
/*     action code.  All other nodes in the recovery domain will see the */
/*     Failover action code.  Action code dependent data of QcstEndNode  */
/*     will be passed for the Failover action code.                      */
/*   - An active recovery domain node for an active CRG is being removed */
/*     from the cluster with the QcstRemoveClusterNodeEntry API.   Action  */
/*     code dependent data of QcstRemoveNode will be passed.  If an      */
/*     inactive node is removed for an active CRG, or if the CRG is      */
/*     inactive, an action code of Remove Node is passed.                */
/*                                                                       */
/* The exit program is called regardless of whether or not the CRG is    */
/* active.  The exit program may have nothing to do if the CRG is not    */
/* active.                                                               */
/*                                                                       */
/* If the CRG is active and the leaving member was the primary node,     */
/* perform the functions necessary for failover to a new primary.        */
/*                                                                       */
/* The Action_Code_Dependent_Data field can be used to determine if:     */
/*  - the failure was due to a problem that caused the cluster to become */
/*     partitioned (all CRGs which had the partitioned nodes in the      */
/*     recovery domain are affected)                                     */
/*  - a node failed or had cluster resource services ended on the node (all*/
/*    CRGs which had the failed/ended node in the recovery  domain are   */
/*    affected)                                                          */
/*  - only a single CRG was affected (for example a single CRG job was   */
/*    canceled on a node or a single application failed)                */
/*                                                                       */
```

```
/*                                                                    */
/* Things to consider:                                                */
/*   - Prepare the new primary node so the application can be started.*/
/*   - The application should NOT be started at this time.  The exit  */
/*     program will be called again with the QcstCrgAcStart action code if */
/*     the CRG was active when the failure occurred.                  */
/*   - If the application CRG is active, ensure the required data CRGs are */
/* active.                                                            */
/*                                                                    */
/**********************************************************************/
static int memberIsLeaving(int role,
                           int priorRole,
                           Qcst_EXTP0100_t *crgData,
                           EpData *epData) {


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* If the CRG is active, perform failover.  Otherwise, nothing to do.*/
  /*                                                                  */

/*--------------------------------------------------------------------*/
  if (crgData->Original_Cluster_Res_Grp_Stat == QcstCrgActive) {


/*--------------------------------------------------------------------*/
    /*                                                                */
    /* The CRG is active.  Determine if my role has changed and I am now */
    /* the new primary.                                               */
    /*                                                                */

/*--------------------------------------------------------------------*/

    if (priorRole != role && role == QcstPrimaryNodeRole) {


/*--------------------------------------------------------------------*/
      /*                                                              */
      /* I was not the primary but am now.  Do failover actions but don't */
      /* start the application at this time because this exit program will */
      /* be called again with the Start action code.                  */
      /*                                                              */

/*--------------------------------------------------------------------*/


/*--------------------------------------------------------------------*/
      /*                                                              */
      /* Ensure the data area status on this node starts out indicating */
      /* the application is not running.                                */
      /*                                                              */

/*--------------------------------------------------------------------*/
      setApplCrgDataArea(Appl_Ended);


/*--------------------------------------------------------------------*/
      /*                                                              */
      /* If the application has no actions to do on the Start action code */
      /* and will become active as soon as the takeover IP address is */
      /* activated, then this code should be uncommented.  This code will */
      /* determine if all CRGs that this application CRG is dependent upon */
      /* are ready.  If this check fails, return failure from the action */
      /* code.                                                        */
      /*                                                              */

/*--------------------------------------------------------------------*/
```

```
/*      if (QcstSuccessful != checkDependCrgDataArea(MaxWaitSeconds))      */
/*         return QcstFailWithOutRestart;                                  */

    }
  }

  return QcstSuccessful;
}  /* end memberIsLeaving()                                                */


/*************************************************************************/
/*                                                                       */
/* Action code = QcstCrgAcSwitchover                                     */
/*                                                                       */
/* The QcstInitiateSwitchOver API was called.  The first backup node in  */
/* the cluster resource group's recovery domain is taking over as the    */
/* primary node and the current primary node is being made the last backup.*/
/*                                                                       */
/* Things to consider:                                                   */
/*   - Prepare the new primary node so the application can be started.   */
/*   - The application should NOT be started at this time.  The exit     */
/*     program will be called again with the QcstCrgAcStart action code. */
/*   - The job running the application is canceled and the IP takeover  */
/*     address is ended prior to the exit program being called on the    */
/*     current primary.                                                  */
/*   - Ensure required data or device CRGs have switched over and are    */
/* active.                                                               */
/*                                                                       */
/*************************************************************************/
static int switchPrimary(int role,
                         int priorRole,
                         Qcst_EXTP0100_t *crgData,
                         EpData *epData) {


/*-----------------------------------------------------------------------*/
  /*                                                                     */
  /* See if I am the old primary.                                        */
  /*                                                                     */

/*-----------------------------------------------------------------------*/
  if (priorRole == QcstPrimaryNodeRole) {

/*-----------------------------------------------------------------------*/
    /*                                                                   */
    /* Do what ever needs to be done to cleanup the old primary before the */
    /* switch.  Remember that that job which was running the exit program  */
    /* which started the application was canceled already.               */
    /*                                                                   */
    /* One example may be to clean up any processes holding locks on the   */
    /* database.  This may have been done by the application cancel        */
    /* handler if one was invoked.                                       */

/*-----------------------------------------------------------------------*/
  }


/*-----------------------------------------------------------------------*/
  /*                                                                     */
  /* I'm not the old primary.  See if I'm the new primary.               */
  /*                                                                     */

/*-----------------------------------------------------------------------*/
  else if (role == QcstPrimaryNodeRole) {

/*-----------------------------------------------------------------------*/
    /*                                                                   */
```

```
    /* Do what ever needs to be done on the new primary before the         */
    /* application is started with the QcstCrgAcStart action code.         */
    /*                                                                     */

/*---------------------------------------------------------------------*/


/*---------------------------------------------------------------------*/
    /*                                                                     */
    /* Ensure the data area status on this nodes starts out indicating     */
    /* the application is not running.                                     */
    /*                                                                     */

/*---------------------------------------------------------------------*/
    setApplCrgDataArea(Appl_Ended);


/*---------------------------------------------------------------------*/
    /*                                                                     */
    /* If the application has no actions to do on the Start action code    */
    /* and will become active as soon as the takeover IP address is        */
    /* activated, then this code should be uncommented.  This code will    */
    /* determine if all CRGs that this application CRG is dependent upon   */
    /* are ready.  If this check fails, return failure from the action     */
    /* code.                                                               */
    /*                                                                     */

/*---------------------------------------------------------------------*/
/*      if (QcstSuccessful != checkDependCrgDataArea(MaxWaitSeconds))      */
/*         return QcstFailWithOutRestart;                                  */

  }
  else {

/*---------------------------------------------------------------------*/
    /*                                                                     */
    /* This node is one of the other backup nodes or it is a replicate     */
    /* node.  If there is anything those nodes must do, do it here.  If */
    /* not, remove this else block.                                        */
    /*                                                                     */

/*---------------------------------------------------------------------*/


/*---------------------------------------------------------------------*/
    /*                                                                     */
    /* Ensure the data area status on this nodes starts out indicating     */
    /* the application is not running.                                     */
    /*                                                                     */

/*---------------------------------------------------------------------*/
    setApplCrgDataArea(Appl_Ended);
  }

  return QcstSuccessful;
}  /* end switchPrimary()                                              */


/***********************************************************************/
/*                                                                     */
/* Action code = QcstCrgAcAddNode                                      */
/*                                                                     */
/* The QcstAddNodeToRcvyDomain API was called.  A new node is being added  */
/* to the recovery domain of a cluster resource group.                 */
/*                                                                     */
/* Things to consider:                                                 */
/*   - A new node is being added to the recovery domain.  See the      */
```

```
/*     considerations in the createCrg() function.                        */
/*   - If this CRG is enabled to use the QcstDistributeInformation API,    */
/*     the user queue needed by that API could be created at this time.    */
/*                                                                         */
/*************************************************************************/
static int addNode(int role,
                   int priorRole,
                   Qcst_EXTP0100_t *crgData,
                   EpData *epData) {


/*-----------------------------------------------------------------------*/

  /*                                                                     */
  /* Determine if I am the node being added.                             */
  /*                                                                     */

/*-----------------------------------------------------------------------*/

  if (0 == memcmp(&crgData->This_Nodes_ID,
                  &crgData->Changing_Node_ID,
                  sizeof(Qcst_Node_Id_t)))
  {

/*-----------------------------------------------------------------------*/

    /*                                                                   */
    /* Set the status of the data area on this new node.                 */
    /*                                                                   */

/*-----------------------------------------------------------------------*/
    setApplCrgDataArea(Appl_Ended);


/*-----------------------------------------------------------------------*/

    /*                                                                   */
    /* Create the queue needed by the Distribute Information API.        */
    /*                                                                   */

/*-----------------------------------------------------------------------*/

    if (0 == memcmp(&crgData->DI_Queue_Name,
                    Nulls,
                    sizeof(crgData->DI_Queue_Name)))
    {
    }
  }

  return QcstSuccessful;
}  /* end addNode()                                                      */


/*************************************************************************/
/*                                                                       */
/* Action code = QcstCrgAcRemoveNode                                     */
/*                                                                       */
/* The QcstRemoveNodeFromRcvyDomain or the QcstRemoveClusterNodeEntry    */
/* API was called.  A node is being removed from the recovery domain of  */
/* a cluster resource group or it is being removed entirely from the     */
/* cluster.                                                              */
/*                                                                       */
/* This action code is seen by:                                          */
/*  For the QcstRemoveClusterNodeEntry API:                              */
/*    - If the removed node is active and the CRG is Inactive, all nodes in*/
/*      the recovery domain including the node being removed see this    */
/*      action code.  The nodes NOT being removed see action code dependent*/
```

```
/*      data of QcstNodeFailure.                                          */
/*    - If the removed node is active and the CRG is Active, the node being*/
/*      removed sees the Remove Node action code.  All other nodes in the  */
/*      recovery domain see an action code of Failover and action code     */
/*      dependent data of QcstNodeFailure.                                 */
/*    - If the node being removed is not active in the cluster, all nodes  */
/*      in the recovery domain will see this action code.                  */
/*  For the QcstRemoveNodeFromRcvyDomain API:                              */
/*    - All nodes see the Remove Node action code regardless of whether or */
/*      not the CRG is Active.  Action code dependent data of              */
/*      QcstRmvRcvyDmnNode will also be passed.                            */
/*                                                                         */
/* Things to consider:                                                     */
/*   - You may want to cleanup the removed node by deleting objects no     */
/*     longer needed there.                                                */
/*   - The job running the application is canceled and the IP takeover    */
/*     address is ended after the exit program is called if this is the    */
/*     primary node and the CRG is active.                                 */
/*   - If subsystems or server jobs were started as a result of the        */
/*     QcstCrgAcStart action code, end them here or consolidate all logic  */
/*     to end the application in the cancelHandler() since it will be      */
/*     invoked for all Cluster Resource Services APIs which must end the   */
/*     application on the current primary.                                 */
/*                                                                         */
/*************************************************************************/
static int rmvNode(int role,
                   int priorRole,
                   Qcst_EXTP0100_t *crgData,
                   EpData *epData) {


/*-----------------------------------------------------------------------*/

  /*                                                                     */
  /* Determine if I am the node being removed.                           */
  /*                                                                     */

/*-----------------------------------------------------------------------*/

  if (0 == memcmp(&crgData->This_Nodes_ID,
                  &crgData->Changing_Node_ID,
                  sizeof(Qcst_Node_Id_t)))
  {

/*---------------------------------------------------------------------*/
      /*                                                                */
      /* End the application if it is running on this node.             */
      /*                                                                */

/*---------------------------------------------------------------------*/
      endApplication(QcstCrgAcRemoveNode, role, priorRole, crgData,
epData);

  }
  return QcstSuccessful;
}  /* end rmvNode                                                      */


/*************************************************************************/
/*                                                                       */
/* Action code = QcstCrgAcChange                                         */
/*                                                                       */
/* The QcstChangeClusterResourceGroup API was called.  Some attribute    */
/* or information stored in the cluster resource group object is being   */
/* changed.  Note that not all changes to the CRG object cause the exit  */
/* program to be called.  As of V5R1M0, only these changes will cause the*/
/* exit program to be called-                                            */
```

```
/*   - the current recovery domain is being changed                       */
/*   - the preferred recovery domain is being changed                     */
/*                                                                        */
/* If any of the above changes are being made but additionally the exit   */
/* program is being changed to *NONE, the exit program is not called.     */
/*                                                                        */
/* Things to consider:                                                    */
/*   - None unless changing the recovery domain affects information or    */
/*     processes for this cluster resource group.  Note that the primary  */
/*     node cannot be changed with the QcstChangeClusterResourceGroup API */
/*     if the CRG is active.                                              */
/*                                                                        */
/**************************************************************************/
static int chgCrg(int role,
                  int priorRole,
                  Qcst_EXTP0100_t *crgData,
                  EpData *epData) {

  return QcstSuccessful;
}  /* end chgCrg()                                                        */


/**************************************************************************/
/*                                                                        */
/* Action code = QcstCrgAcDeleteCommand                                   */
/*                                                                        */
/* The Delete Cluster Resource Group (DLTCRG) CL command has been called  */
/* to delete a cluster resource group object, the QcstDeleteCluster API   */
/* has been called, or the QcstRemoveClusterNodeEntry API has been called.*/
/* In each case, cluster resource services is not active on the cluster   */
/* node where the command or API was called.  Thus, this function is not  */
/* distributed cluster wide but occurs only on the node where the CL      */
/* command or API was called.                                             */
/*                                                                        */
/* If the QcstDeleteCluster API was used, action code dependent data of   */
/* QcstDltCluster is passed.                                              */
/*                                                                        */
/* See the considerations in the deleteCrg() function                     */
/*                                                                        */
/**************************************************************************/
static int deleteCrgWithCmd(int role,
                            int doesNotApply,
                            Qcst_EXTP0100_t *crgData,
                            EpData *epData) {

  return QcstSuccessful;
}  /* end deleteCrgWithCmd()                                              */


/**************************************************************************/
/*                                                                        */
/* Action code = QcstCrgEndNode                                           */
/*                                                                        */
/* The QcstEndClusterNode API was called or a CRG job was canceled.      */
/*                                                                        */
/* The QcstCrgEndNode action code is passed to the exit program only on the*/
/* node being ended or where the CRG job was canceled.  On the node where */
/* a Cluster Resource Services job is canceled, action code dependent data*/
/* of QcstMemberFailure will be passed.                                   */
/* When Cluster Resource Services ends on this node or the CRG job ends, it*/
/* will cause all other nodes in the cluster to go through failover       */
/* processing.  The action code passed to all other nodes will be         */
/* QcstCrgAcFailover.   Those nodes will see action code dependent data of */
/* QcstMemberFailure if a CRG job is canceled or QcstNodeFailure if the   */
/* node is ended.                                                         */
/*                                                                        */
/* Things to consider:                                                    */
```

```
/*   - The job running the application is canceled and the IP takeover    */
/*     address is ended after the exit program is called if this is the    */
/*     primary node and the CRG is active.                                 */
/*   - If subsystems or server jobs were started as a result of the        */
/*     QcstCrgAcStart action code, end them here.                          */
/*                                                                         */
/*************************************************************************/
static int endNode(int role,
                   int priorRole,
                   Qcst_EXTP0100_t *crgData,
                   EpData *epData) {


/*-----------------------------------------------------------------------*/
  /*                                                                     */
  /* End the application if it is running on this node.               */
  /*                                                                     */

/*-----------------------------------------------------------------------*/
  endApplication(QcstCrgEndNode, role, priorRole, crgData, epData);

  return QcstSuccessful;
}  /* end endNode()                                                      */


/*************************************************************************/
/*                                                                       */
/* Action code = QcstCrgAcChgNodeStatus                                  */
/*                                                                       */
/* The QcstChangeClusterNodeEntry API was called.  The status of a node  */
/* is being changed to failed.  This API is used to inform cluster resource*/
/* services that the node did not partition but really failed.           */
/*                                                                       */
/* Things to consider:                                                   */
/*   - The exit program was called previously with an action code of     */
/*     QcstCrgAcEnd if the CRG was active or an action code of           */
/*     QcstCrgAcFailover if the CRG was inactive because cluster resource  */
/*     services thought the cluster had become partitioned.  The user is   */
/*     now telling cluster resource services that the node really failed   */
/*     instead of partitioned.  The exit program has something to do only  */
/*     if it performed some action previously that needs to be changed now */
/*     that node failure can be confirmed.                               */
/*                                                                       */
/*************************************************************************/
static int chgNodeStatus(int role,
                         int priorRole,
                         Qcst_EXTP0100_t *crgData,
                         EpData *epData) {

  return QcstSuccessful;
}  /* end chgNodeStatus()                                                */


/*************************************************************************/
/*                                                                       */
/* Action code = QcstCrgAcCancelFailover                                 */
/*                                                                       */
/* Cluster resource services on the primary node has failed or ended     */
/* for this cluster resource group.  A message was sent to the failover  */
/* message queue specified for the CRG, and the result of that message   */
/* was to cancel the failover.  This will change the status of the CRG to  */
/* inactive and leave the primary node as primary.                       */
/*                                                                       */
/* Things to consider:                                                   */
/*   - The primary node is no longer participating in cluster activities.  */
/*     The problem which caused the primary node to fail should be fixed   */
/*     so that the CRG may be started again.                             */
```

```
/*                                                                    */
/**********************************************************************/
static int cancelFailover(int role,
                          int priorRole,
                          Qcst_EXTP0100_t *crgData,
                          EpData *epData) {

  return QcstSuccessful;
}  /* end cancelFailover()                                            */


/**********************************************************************/
/*                                                                    */
/* Action code = exit program does not know it yet                    */
/*                                                                    */
/* A new action code has been passed to this exit program.  This can occur */
/* after a new i5/OS release has been installed and some new cluster API  */
/* was called or some new cluster event occurred.  The logic in this exit  */
/* program has not yet been updated to understand the new action code.     */
/*                                                                    */
/* Two different strategies could be used for the new action code.  The    */
/* correct strategy is dependent upon the kinds of things this particular  */
/* exit program does for the application.                             */
/*                                                                    */
/* One strategy is to not do anything and return a successful return code. */
/* This allows the new cluster API or event to run to completion.  It      */
/* allows the function to be performed even though this exit program       */
/* did not understand the new action code.  The risk, though, is that the  */
/* exit program should have done something and it did not.  At a minimum,  */
/* you may want to log some kind of error message about what happened so   */
/* that programming can investigate and get the exit program updated.      */
/*                                                                    */
/* The opposite strategy is to return an error return code such as         */
/* QcstFailWithRestart.  Of course doing this means that the new cluster   */
/* API or event cannot be used until the exit program is updated for the   */
/* new action code.  Again, logging some kind of error message for         */
/* programming to investigate would be worthwhile.                    */
/*                                                                    */
/* Only the designer of the exit program can really decide which is the    */
/* better course of action.                                           */
/*                                                                    */
/**********************************************************************/
static int newActionCode(int role,
                         int doesNotApply,
                         Qcst_EXTP0100_t *crgData,
                         EpData *epData) {


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Add logic to log an error somewhere - operator message queue, job     */
  /* log, application specific error log, etc. so that the exit program    */
  /* gets updated to properly handle the new action code.               */
  /*                                                                  */
  /* Note that if this is left coded as it is, this is the "don't do       */
  /* anything" strategy described in the prologue above.                */
  /*                                                                  */

/*--------------------------------------------------------------------*/

  return QcstSuccessful;
}  /* end newActionCode()                                             */


/**********************************************************************/
/*                                                                    */
/* Action code = QcstCrgAcUndo                                        */
```

```
/*                                                                    */
/* Note: The exit program is never called with an undo action code for*/
/* any of these prior action codes:                                   */
/*   QcstCrgAcChgNodeStatus                                           */
/*   QcstCrgAcDelete                                                  */
/*   QcstCrgAcDeleteCommand                                           */
/*   QcstCrgEndNode                                                   */
/*   QstCrgAcRemoveNode (If the node being removed is active in the   */
/*                       cluster and the API is Remove Cluster Node.  */
/*                       The Remove Node From Recovery Domain will call*/
/*                       with Undo and the Remove Cluster Node API will*/
/*                       call with Undo if the node being removed is  */
/*                       inactive.                                    */
/*   QcstCrgAcRestart                                                 */
/*   QcstCrgAcUndo                                                    */
/*                                                                    */
/* APIs that call an exit program do things in 3 steps.               */
/*   1. Logic which must be done prior to calling the exit program.   */
/*   2. Call the exit program.                                        */
/*   3. Logic which must be done after calling the exit program.      */
/*                                                                    */
/* Any errors that occur during steps 2 or 3 result in the exit program*/
/* being called again with the undo action code.  This gives the exit */
/* program an opportunity to back out any work performed when it was first */
/* called by the API.  The API will also be backing out any work it   */
/* performed trying to return the state of the cluster and cluster objects */
/* to what it was before the API was called.                          */
/*                                                                    */
/* It is suggested that the following return codes be returned for the*/
/* specified action code as that return code will result in the most  */
/* appropriate action being taken.                                    */
/*                                                                    */
/*   QcstCrgAcInitialize: QcstSuccessful; The CRG is not created.     */
/*   QcstCrgAcStart:      QcstSuccessful; The CRG is not started.     */
/*   QcstCrgAcEnd:        QcstFailWithOutRestart; The CRG is set to Indoubt*/
/*                                        The cause of the failure needs to*/
/*                                        investigated.               */
/*   QcstCrgAcReJoin:     QcstFailWithOutRestart; The CRG is set to Indoubt*/
/*                                        The cause of the failure needs to*/
/*                                        investigated.               */
/*   QcstCrgAcFailover:   QcstFailWithOutRestart; The CRG is set to Indoubt*/
/*                                        The cause of the failure needs to*/
/*                                        investigated.               */
/*   QcstCrgAcSwitchover: QcstFailWithOutRestart; The CRG is set to Indoubt*/
/*                                        The cause of the failure needs to*/
/*                                        investigated.               */
/*   QcstCrgAcAddNode:    QcstSuccessful; The node is not added.      */
/*   QcstCrgAcRemoveNode: QcstFailWithOutRestart; The CRG is set to Indoubt*/
/*                                        The cause of the failure needs to*/
/*                                        investigated.               */
/*   QcstCrgAcChange:     QcstSuccessful; The recovery domain is not  */
/*                                        changed.                    */
/*                                                                    */
/**********************************************************************/
static int undoPriorAction(int role,
                           int priorRole,
                           Qcst_EXTP0100_t *crgData,
                           EpData *epData) {


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* The prior action code defines what the exit program was doing when */
  /* it failed, was canceled, or returned a non successful return code. */
  /*                                                                  */

/*--------------------------------------------------------------------*/
```

```
  if (crgData->Prior_Action_Code &lt;= MaxAc )
    return (*undoFcn[crgData-&lt;Prior_Action_Code])
                                      (role, priorRole, crgData,
epData);
  else

/*--------------------------------------------------------------------*/
    /*                                                                */
    /* IBM has defined a new action code in a new operating system release */
    /* and this exit program has not yet been updated to handle it.  Take a*/
    /* default action for now.                                        */
    /*                                                                */

/*--------------------------------------------------------------------*/
    return newActionCode(role, priorRole, crgData, epData);
}  /* end undoPriorAction()                                           */


/**********************************************************************/
/*                                                                    */
/* Action code = QcstCrgAcUndo                                        */
/*                                                                    */
/* Prior action code = QcstCrgAcInitialize                            */
/*                                                                    */
/* Things to consider:                                                */
/*   The CRG will not be created.  Objects that might have been created    */
/*   on nodes in the recovery domain should be deleted since a subsequent  */
/*   create could fail if those objects already exist.                */
/*                                                                    */
/**********************************************************************/
static int undoCreateCrg(int role,
                         int doesNotApply,
                         Qcst_EXTP0100_t *crgData,
                         EpData *epData) {

  return QcstSuccessful;
}  /* end undoCreateCrg()                                           */


/**********************************************************************/
/*                                                                    */
/* Action code = QcstCrgAcUndo                                        */
/*                                                                    */
/* Prior action code = QcstCrgAcStart                                 */
/*                                                                    */
/* Things to consider:                                                */
/*   Cluster Resource Services failed when it was finishing the Start CRG  */
/*   API after it had already called the exit program with the Start       */
/*   Action code.                                                     */
/*                                                                    */
/*   On the primary node, the exit program job which is running the        */
/*   application will be canceled.  The exit program will then be called  */
/*   with the Undo action code.                                       */
/*                                                                    */
/*   All other nodes in the recovery domain will be called with the Undo   */
/*   action code.                                                     */
/*                                                                    */
/**********************************************************************/
static int undoStartCrg(int role,
                        int doesNotApply,
                        Qcst_EXTP0100_t *crgData,
                        EpData *epData) {

  return QcstSuccessful;
}  /* end undoStartCrg()                                            */
```

```
/*****************************************************************************/
/*                                                                           */
/* Action code = QcstCrgAcUndo                                               */
/*                                                                           */
/* Prior action code = QcstCrgAcEnd                                          */
/*                                                                           */
/* Things to consider:                                                       */
/*   The CRG will not be ended.  If the exit program did anything to bring */
/*   down the application it can either restart the application or it can  */
/*   decide to not restart the application.  If the application is not     */
/*   restarted, the return code should be set to QcstFailWithOutRestart so */
/*   the status of the CRG is set to Indoubt.                                */
/*                                                                           */
/*****************************************************************************/
static int undoEndCrg(int role,
                      int doesNotApply,
                      Qcst_EXTP0100_t *crgData,
                      EpData *epData) {

  return QcstFailWithOutRestart;
}  /* end undoEndCrg()                                                       */


/*****************************************************************************/
/*                                                                           */
/* Action code = QcstCrgAcUndo                                               */
/*                                                                           */
/* Prior action code = QcstCrgAcReJoin                                       */
/*                                                                           */
/* Things to consider:                                                       */
/*   An error occurred which won't allow the member to join this CRG       */
/*   group.  Anything done for the Join action code needs to be looked at  */
/*   to see if something must be undone if this member is not an active    */
/*   member of the CRG group.                                                */
/*                                                                           */
/*****************************************************************************/
static int undoMemberIsJoining(int role,
                               int doesNotApply,
                               Qcst_EXTP0100_t *crgData,
                               EpData *epData) {

  return QcstFailWithOutRestart;
}  /* end undoMemberIsJoining()                                              */


/*****************************************************************************/
/*                                                                           */
/* Action code = QcstCrgAcUndo                                               */
/*                                                                           */
/* Prior action code = QcstCrgAcFailover                                     */
/*                                                                           */
/* Things to consider:                                                       */
/*   This does not mean that the node failure or failing member is being   */
/*   undone.  That failure is irreversible.  What it does mean is that the */
/*   exit program returned an error from the Failover action code or       */
/*   Cluster Resource Services ran into a problem after it called the exit */
/*   program.  If the CRG was active when Failover was attempted, it is    */
/*   not at this point.  End the resilient resource and expect a human to  */
/*   look into the failure.  After the failure is corrected, the CRG will  */
/*   must be started with the Start CRG API.                          */
/*                                                                           */
/*                                                                           */
/*****************************************************************************/
static int undoMemberIsLeaving(int role,
                               int doesNotApply,
                               Qcst_EXTP0100_t *crgData,
                               EpData *epData) {
```

```
  return QcstFailWithOutRestart;
}  /* end undoMemberIsLeaving()                                              */


/*************************************************************************/
/*                                                                       */
/* Action code = QcstCrgAcUndo                                           */
/*                                                                       */
/* Prior action code = QcstCrgAcSwitchover                               */
/*                                                                       */
/* Things to consider:                                                   */
/*   Some error occurred after the point of access was moved from the    */
/*   original primary and before it could be brought up on the new primary.*/
/*   The IP address was ended on the original primary before moving the  */
/*   point of access but is started on the original primary again.  Cluster*/
/*   Resource Services will now attempt to move the point of access back */
/*   to the original primary.  The application exit program and IP takeover*/
/*   address will be started on the original primary.                    */
/*                                                                       */
/*                                                                       */
/*************************************************************************/
static int undoSwitchPrimary(int role,
                             int doesNotApply,
                             Qcst_EXTP0100_t *crgData,
                             EpData *epData) {

  return QcstFailWithOutRestart;
}  /* end undoSwitchPrimary()                                            */


/*************************************************************************/
/*                                                                       */
/* Action code = QcstCrgAcUndo                                           */
/*                                                                       */
/* Prior action code = QcstCrgAcAddNode                                  */
/*                                                                       */
/* Things to consider:                                                   */
/*   If objects were created on the new node, they should be removed so  */
/*   that a subsequent Add Node to aRecovery Domain does not fail if it   */
/*   attempts to create objects again.                                   */
/*                                                                       */
/*                                                                       */
/*************************************************************************/
static int undoAddNode(int role,
                       int doesNotApply,
                       Qcst_EXTP0100_t *crgData,
                       EpData *epData) {

  return QcstSuccessful;
}  /* end undoAddNode()                                                  */


/*************************************************************************/
/*                                                                       */
/* Action code = QcstCrgAcUndo                                           */
/*                                                                       */
/* Prior action code = QcstCrgAcRemoveNode                               */
/*                                                                       */
/* Things to consider:                                                   */
/*   The node is still in the recovery domain.  If objects were removed  */
/*   from the node, they should be added back.                           */
/*                                                                       */
/*************************************************************************/
static int undoRmvNode(int role,
                       int doesNotApply,
                       Qcst_EXTP0100_t *crgData,
```

```
                      EpData *epData) {

  return QcstFailWithOutRestart;
}  /* end undoRmvNode()                                                    */


/*************************************************************************/
/*                                                                       */
/* Action code = QcstCrgAcUndo                                           */
/*                                                                       */
/* Prior action code = QcstCrgAcChange                                   */
/*                                                                       */
/* Things to consider:                                                   */
/*   Changes to the CRG will be backed out so that the CRG and its       */
/*   recovery domain look just like it did prior to the attempted change.*/
/*   Any changes the exit program made should also be backed out.        */
/*                                                                       */
/*************************************************************************/
static int undoChgCrg(int role,
                      int doesNotApply,
                      Qcst_EXTP0100_t *crgData,
                      EpData *epData) {

  return QcstSuccessful;
}  /* end undoChgCrg()                                                   */


/*************************************************************************/
/*                                                                       */
/* Action code = QcstCrgAcUndo                                           */
/*                                                                       */
/* Prior action code = QcstCrgAcCancelFailover                           */
/*                                                                       */
/* Things to consider:                                                   */
/*   This does not mean that the node failure or failing member is being */
/*   undone.  That failure is irreversible.  What it does mean is that   */
/*   Cluster Resource Services ran into a problem after it called the exit */
/*   program.  The CRG will be InDoubt regardless of what is returned from */
/*   this exit program call.  Someone will need to manually look into the */
/*   the failure.  After the failure is corrected, the CRG will must be */
/*   started with the Start CRG API.                                     */
/*                                                                       */
/*                                                                       */
/*************************************************************************/
static int undoCancelFailover(int role,
                              int doesNotApply,
                              Qcst_EXTP0100_t *crgData,
                              EpData *epData) {

  return QcstSuccessful;
}  /* end undoCancelFailover()                                           */


/*************************************************************************/
/*                                                                       */
/* A simple routine to take a null terminated object name and a null     */
/* terminated library name and build a 20 character non-null terminated  */
/* qualified name.                                                       */
/*                                                                       */
/*************************************************************************/
static void bldDataAreaName(char *objName, char* libName, char *qualName) {

  memset(qualName, 0x40, 20);
  memcpy(qualName, objName, strlen(objName));
  qualName += 10;
  memcpy(qualName, libName, strlen(libName));
  return;
```

```
}  /* end bldDataAreaName                                                  */


/*****************************************************************************/
/*                                                                           */
/* The data area is checked to see if all the CRGs that this application   */
/* is dependent upon are ready.  If they are not ready, a wait for a       */
/* certain amount of time is performed and the data area is checked again. */
/* This check, wait loop continues until all dependent CRGs become ready or*/
/* until the maximum wait time has been reached.                           */
/* The length of the wait can be changed to some other value if a          */
/* particular situation would be better with shorter or longer wait times. */
/*                                                                           */
/*                                                                           */
/*****************************************************************************/
static int checkDependCrgDataArea(unsigned int maxWaitTime) {

  Qus_EC_t errCode = { sizeof(Qus_EC_t), 0 };
  char dataAreaName[20];
  struct {
    Qwc_Rdtaa_Data_Returned_t stuff;
    char ready;
  } data;


/*---------------------------------------------------------------------------*/
  /*                                                                         */
  /* This is an accumulation of the time waited for the dependent CRGs to  */
  /* become ready.                                                           */
  /*                                                                         */

/*---------------------------------------------------------------------------*/
  unsigned int timeWaited = 0;


/*---------------------------------------------------------------------------*/
  /*                                                                         */
  /* Build definition of the amount of time to wait.                       */
  /*                                                                         */

/*---------------------------------------------------------------------------*/
  _MI_Time   timeToWait;
  int hours     = 0;
  int minutes   = 0;
  int seconds   = WaitSecondsIncrement;
  int hundreths = 0;
  short int options = _WAIT_NORMAL;
  mitime( &timeToWait, hours, minutes, seconds, hundreths );


/*---------------------------------------------------------------------------*/
  /*                                                                         */
  /* Build the qualified name of the data area.                            */
  /*                                                                         */

/*---------------------------------------------------------------------------*/
  bldDataAreaName(DependCrgDataArea, ApplLib, dataAreaName);


/*---------------------------------------------------------------------------*/
  /*                                                                         */
  /* Get the data from the data area that indicates whether or not the     */
  /* CRGs are all ready.  This data area is updated by the High            */
  /* Availability Business Partners when it is ok for the application to   */
  /* proceed.                                                                */
  /*                                                                         */
```

```
/*--------------------------------------------------------------------*/
  QWCRDTAA(&data,
           sizeof(data),
           dataAreaName,
           offsetof(Qcst_HAAPPO_t,Data_Status)+1,  /* API wants a 1 origin */
           sizeof(data.ready),
           &errCode);


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* If the dependent CRGs are not ready, wait for a bit and check again.  */
  /*                                                                  */

/*--------------------------------------------------------------------*/
  while (data.ready != Data_Available) {


/*--------------------------------------------------------------------*/
    /*                                                                */
    /* If the dependent CRGs have not become ready by the time we have     */
    /* waited our maximum wait time, return an error.  Consider logging    */
    /* some message to describe why the application did not start so that  */
    /* the problem can be looked into.                                */
    /*                                                                */

/*--------------------------------------------------------------------*/
    if (timeWaited >= maxWaitTime)
      return QcstFailWithOutRestart;


/*--------------------------------------------------------------------*/
    /*                                                                */
    /* Wait to allow the data CRGs to become ready.                   */
    /*                                                                */

/*--------------------------------------------------------------------*/
    waittime(&timeToWait, options);
    timeWaited += WaitSecondsIncrement;


/*--------------------------------------------------------------------*/
    /*                                                                */
    /* Get information from the data area again to see if the data CRGs are*/
    /* ready.                                                         */
    /*                                                                */

/*--------------------------------------------------------------------*/
    QWCRDTAA(&data,
             sizeof(data),
             dataAreaName,
             offsetof(Qcst_HAAPPO_t,Data_Status)+1,  /* API wants a 1 origin */
             sizeof(data.ready),
             &errCode);
  }

  return QcstSuccessful;
}  /* end checkDependCrgDataArea                                      */


/*********************************************************************/
/*                                                                   */
/* The application CRG data area is updated to indicate that the     */
/* application is running or to indicate it is not running.  This data area*/
/* information is used by the High Availability Business Partners to  */
/* coordinate the switchover activities between CRGs that have dependencies*/
/* on each other.                                                    */
```

```
/*                                                                    */
/**********************************************************************/
static void setApplCrgDataArea(char status) {

  char cmd[54];
  char cmdEnd[3] = {0x00, ')', 0x00};


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Set up the CL command string with the data area library name, the data*/
  /* area name, and the character to put into the data area.  Then run the */
  /* CL command.                                                      */
  /*                                                                  */

/*--------------------------------------------------------------------*/
  memcpy(cmd, "CHGDTAARA DTAARA(", strlen("CHGDTAARA DTAARA(")+1);
  strcat(cmd, ApplLib);
  strcat(cmd, "/");
  strcat(cmd, ApplCrgDataArea);
  strcat(cmd, " (425 1)) VALUE(");                                 /* @A1C */
  cmdEnd[0] = status;
  strcat(cmd, cmdEnd);

  system(cmd);

  return;
}  /* end setApplCrgDataArea                                        */


/**********************************************************************/
/*                                                                    */
/* This function is called any time the exit program receives an exception */
/* not specifically monitored for by some other exception handler.  Add    */
/* appropriate logic to perform cleanup functions that may be required.    */
/* A failure return code is then set and control returns to the operating  */
/* system.  The job this exit program is running in will then end.         */
/*                                                                    */
/* When this function gets called, myData->role may still contain the      */
/* UnknownRole value if an exception occurred before this node's role      */
/* value was set.  To be completely correct, the role should be tested     */
/* for UnknownRole before making any decisions based upon the value of     */
/* role.                                                              */
/*                                                                    */
/**********************************************************************/
static void unexpectedExceptionHandler(_INTRPT_Hndlr_Parms_T *exData) {


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Get a pointer to the structure containing data we passed to the       */
  /* exception handler.                                               */
  /*                                                                  */

/*--------------------------------------------------------------------*/
  HandlerDataT *myData = (HandlerDataT *)exData->Com_Area;


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Perform as much cleanup function as necessary.  Some global state     */
  /* information may must be kept so the exception handler knows what   */
  /* steps were completed before the failure occurred and thus knows what  */
  /* cleanup steps must be performed.  This state information could be     */
  /* kept in the HandlerDataT structure or it could be kept in some other  */
  /* location that this function can address.                         */
  /*                                                                  */
```

```
/*--------------------------------------------------------------------*/


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* If this is the primary node and the application was started, end it.  */
  /* The application is ended because the exit program will be called again*/
  /* with the Restart action code and we want the restartCrg() function to */
  /* always work the same way.  In addition, ending the application may    */
  /* clear up the condition that caused the exception that got us here.    */
  /* If possible, warn users and have them stop using the application so   */
  /* things are done in an orderly manner.                                */
  /*                                                                  */

/*--------------------------------------------------------------------*/
  endApplication(myData->actionCode,
                 myData->role,
                 myData->priorRole,
                 myData->crgData,
                 myData->epData);


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Set the exit program return code.                                */
  /*                                                                  */

/*--------------------------------------------------------------------*/
  *myData->retCode = QcstFailWithRestart;


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Let the exception percolate up the call stack.                 */
  /*                                                                  */

/*--------------------------------------------------------------------*/
  return;
}  /* end unexpectedExceptionHandler                                 */


/**********************************************************************/
/*                                                                    */
/* This function is called any time the job this exit program is running in*/
/* is canceled.  The job could be canceled due to any of the following   */
/* (the list is not intended to be all inclusive)-                    */
/*   - an API cancels an active application CRG.  The End CRG, Initiate    */
/*     Switchover, End Cluster Node, Remove Cluster Node or Delete Cluster */
/*     API cancels the job which was submitted when the exit program was   */
/*     called with a Start action code.                               */
/*   - operator cancels the job from some operating system display such as */
/*     Work with Active Jobs                                          */
/*   - the subsystem this job is running in is ended                  */
/*   - all subsystems are ended                                       */
/*   - the system is powered down                                     */
/*   - an operating system machine check occurred                     */
/*                                                                    */
/* When this function gets called, myData->role may still contain the     */
/* UnknownRole value if cancelling occurred before this node's role       */
/* value was set.  To be completely correct, the role should be tested    */
/* for UnknownRole before making any decisions based upon the value of    */
/* role.                                                              */
/*                                                                    */
/**********************************************************************/
static void cancelHandler(_CNL_Hndlr_Parms_T *cnlData) {
```

```
/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Get a pointer to the structure containing data we passed to the  */
  /* cancel handler.                                                  */
  /*                                                                  */

/*--------------------------------------------------------------------*/
  HandlerDataT *myData = (HandlerDataT *)cnlData->Com_Area;


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Perform as much cleanup function as necessary.  Some global state*/
  /* information may must be kept so the cancel handler knows what    */
  /* steps were completed before the job was canceled and thus knows if*/
  /* the function had really completed successfully or was only partially*/
  /* complete and thus needs some cleanup to be done.  This state     */
  /* information could be kept in the HandlerDataT structure or it could*/
  /* be kept in some other location that this function can address.   */
  /*                                                                  */

/*--------------------------------------------------------------------*/


/*--------------------------------------------------------------------*/

  /*                                                                  */
  /* This job is being canceled.  If I was running the application as a*/
  /* result of the Start or Restart action codes, end the application now.*/
  /* This job is being canceled because a Switch Over or some other   */
  /* Cluster Resource Services API was used which affects the primary node*/
  /* or someone did a cancel job with a CL command, from a system display,*/
  /* etc.                                                             */


/*--------------------------------------------------------------------*/
  endApplication(myData->actionCode,
                 myData->role,
                 myData->priorRole,
                 myData->crgData,
                 myData->epData);


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Set the exit program return code.                                */
  /*                                                                  */

/*--------------------------------------------------------------------*/
  *myData->retCode = QcstSuccessful;


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Return to the operating system for final ending of the job.      */
  /*                                                                  */

/*--------------------------------------------------------------------*/
  return;
}  /* end cancelHandler                                             */


/**********************************************************************/
/*                                                                    */
/* A common routine used to end the application by various action code*/
/* functions, the exception handler, and the cancel handler.          */
```

```
/*                                                                    */
/**********************************************************************/
static void endApplication(unsigned int actionCode,
                           int role,
                           int priorRole,
                           Qcst_EXTP0100_t *crgData,
                           EpData *epData) {

  if (  role == QcstPrimaryNodeRole
      &&
        crgData->Original_Cluster_Res_Grp_Stat == QcstCrgActive) {

/*--------------------------------------------------------------------*/
    /*                                                                */
    /* Add logic to end the application here.  You may need to add logic   */
    /* to determine if the application is still running because this       */
    /* function could be called once for an action code and again from     */
    /* the cancel handler (End CRG is an example).                        */
    /*                                                                */

/*--------------------------------------------------------------------*/




/*--------------------------------------------------------------------*/
    /*                                                                */
    /* After the application has ended, update the data area to indicate   */
    /* the application is no longer running.                              */
    /*                                                                */

/*--------------------------------------------------------------------*/
    setApplCrgDataArea(Appl_Ended);
  }

  return;
}  /* end endApplication                                              */


/**********************************************************************/
/*                                                                    */
/* Print out the data passed to this program.                         */
/*                                                                    */
/**********************************************************************/
static void printParms(int actionCode,
                       int role,
                       int priorRole,
                       Qcst_EXTP0100_t *crgData,
                       EpData *epData) {

  unsigned int i;
  char *str;

  /* Print the action code.                                           */
  printf("%s", "Action_Code = ");
  printActionCode(actionCode);

  /* Print the action code dependent data.                            */
  printf("%s", "    Action_Code_Dependent_Data = ");
  switch (crgData->Action_Code_Dependent_Data) {
    case QcstNoDependentData:  str = "QcstNoDependentData";
                               break;
    case QcstMerge:            str = "QcstMerge";
                               break;
    case QcstJoin:             str = "QcstJoin";
                               break;
    case QcstPartitionFailure: str = "QcstPartitionFailure";
```

```
                            break;
  case QcstNodeFailure:     str = "QcstNodeFailure";
                            break;
  case QcstMemberFailure:   str = "QcstMemberFailure";
                            break;
  case QcstEndNode:         str = "QcstEndNode";
                            break;
  case QcstRemoveNode:      str = "QcstRemoveNode";
                            break;
  case QcstApplFailure:     str = "QcstApplFailure";
                            break;
  case QcstResourceEnd:     str = "QcstResourceEnd";
                            break;
  case QcstDltCluster:      str = "QcstDltCluster";
                            break;
  case QcstRmvRcvyDmnNode:  str = "QcstRmvRcvyDmnNode";
                            break;
  case QcstDltCrg:          str = "QcstDltCrg";
                            break;
  default: str = "unknown action code dependent data";
}
printf("%s \n", str);


/* Print the prior action code.                                       */
printf("%s", "  Prior_Action_Code = ");
if (crgData->Prior_Action_Code)
  printActionCode(crgData->Prior_Action_Code);
printf("\n");

/* Print the cluster name.                                            */
printStr("  Cluster_Name = ",
         crgData->Cluster_Name, sizeof(Qcst_Cluster_Name_t));

/* Print the CRG name.                                                */
printStr("  Cluster_Resource_Group_Name = ",
         crgData->Cluster_Resource_Group_Name, sizeof(Qcst_Crg_Name_t));

/* Print the CRG type.                                                */
printf("%s \n", "  Cluster_Resource_Group_Type = QcstCrgApplResiliency");

/* Print the CRG status.                                                */
printf("%s", "  Cluster_Resource_Group_Status = ");
printCrgStatus(crgData->Cluster_Resource_Group_Status);

/* Print the CRG original status.                                     */
printf("%s", "  Original_Cluster_Res_Grp_Stat = ");
printCrgStatus(crgData->Original_Cluster_Res_Grp_Stat);

/* Print the Distribute Information queue name.                       */
printStr("  DI_Queue_Name = ",
         crgData->DI_Queue_Name, sizeof(crgData->DI_Queue_Name));
printStr("  DI_Queue_Library_Name = ",
         crgData->DI_Queue_Library_Name,
         sizeof(crgData->DI_Queue_Library_Name));

/* Print the CRG attributes.                                          */
printf("%s", "  Cluster_Resource_Group_Attr = ");
if (crgData->Cluster_Resource_Group_Attr & QcstTcpConfigByUsr)
  printf("%s", "User Configures IP Takeover Address");
printf("\n");

/* Print the ID of this node.                                         */
printStr("  This_Nodes_ID = ",
         crgData->This_Nodes_ID, sizeof(Qcst_Node_Id_t));

/* Print the role of this node.                                       */
```

```
printf("%s %d \n", "  this node's role = ", role);

/* Print the prior role of this node.                                 */
printf("%s %d \n", "  this node's prior role = ", priorRole);

/* Print which recovery domain this role comes from.                  */
printf("%s", "  Node_Role_Type = ");
if (crgData->Node_Role_Type == QcstCurrentRcvyDmn)
  printf("%s \n", "QcstCurrentRcvyDmn");
else
  printf("%s \n", "QcstPreferredRcvyDmn");

/* Print the ID of the changing node (if any).                        */
printStr("  Changing_Node_ID = ",
         crgData->Changing_Node_ID, sizeof(Qcst_Node_Id_t));

/* Print the role of the changing node (if any).                      */
printf("%s", "  Changing_Node_Role = ");
if (crgData->Changing_Node_Role == -3)
  printf("%s \n", "*LIST");
else if (crgData->Changing_Node_Role == -2)
 printf("%s \n", "does not apply");
else
  printf("%d \n", crgData->Changing_Node_Role);

/* Print the takeover IP address.                                     */
printStr("  Takeover_IP_Address = ",
         crgData->Takeover_IP_Address, sizeof(Qcst_TakeOver_IP_Address_t));

/* Print the job name.                                                */
printStr("  Job_Name = ", crgData->Job_Name, 10);

/* Print the CRG changes.                                             */
printf("%s \n", "  Cluster_Resource_Group_Changes = ");
if (crgData->Cluster_Resource_Group_Changes & QcstRcvyDomainChange)
  printf("      %s \n", "Recovery domain changed");
if (crgData->Cluster_Resource_Group_Changes & QcstTakeOverIpAddrChange)
  printf("      %s \n", "Takeover IP address changed");

/* Print the failover wait time.                                      */
printf("%s", "Failover_Wait_Time = ");
if (crgData->Failover_Wait_Time == QcstFailoverWaitForever)
   printf("%d %s \n", crgData->Failover_Wait_Time, "Wait forever");
else if (crgData->Failover_Wait_Time == QcstFailoverNoWait)
   printf("%d %s \n", crgData->Failover_Wait_Time, "No wait");
else
   printf("%d %s \n", crgData->Failover_Wait_Time, "minutes");

/* Print the failover default action.                                 */
printf("%s", "Failover_Default_Action = ");
if (crgData->Failover_Default_Action == QcstFailoverProceed)
   printf("%d %s \n", crgData->Failover_Default_Action, "Proceed");
else
   printf("%d %s \n", crgData->Failover_Default_Action, "Cancel");

/* Print the failover message queue name.                             */
printStr("  Failover_Msg_Queue = ",
         crgData->Failover_Msg_Queue, sizeof(crgData->Failover_Msg_Queue));
printStr("  Failover_Msg_Queue_Lib = ",
         crgData->Failover_Msg_Queue_Lib,
         sizeof(crgData->Failover_Msg_Queue_Lib));

/* Print the cluster version.                                         */
printf("%s %d \n",
       "  Cluster_Version = ", crgData->Cluster_Version);

/* Print the cluster version mod level                                */
```

```
  printf("%s %d \n",
         "  Cluster_Version_Mod_Level = ",
         crgData->Cluster_Version_Mod_Level);

  /* Print the requesting user profile.                                 */
  printStr("  Req_User_Profile = ",
           crgData->Req_User_Profile, sizeof(crgData->Req_User_Profile));

  /* Print the length of the data in the structure.                     */
  printf("%s %d \n",
         "  Length_Info_Returned = ", crgData->Length_Info_Returned);

  /* Print the offset to the recovery domain array.                     */
  printf("%s %d \n",
         "  Offset_Rcvy_Domain_Array = ", crgData->Offset_Rcvy_Domain_Array);

  /* Print the number of nodes in the recovery domain array.            */
  printf("%s %d \n",
         "  Number_Nodes_Rcvy_Domain = ",
crgData->Number_Nodes_Rcvy_Domain);

  /* Print the current/new recovery domain.                             */
  printRcvyDomain("   The recovery domain:",
                  crgData->Number_Nodes_Rcvy_Domain,
                  (Qcst_Rcvy_Domain_Array1_t *)
                  ((char *)crgData + crgData->Offset_Rcvy_Domain_Array));

  /* Print the offset to the prior recovery domain array.               */
  printf("%s %d \n",
         "  Offset_Prior_Rcvy_Domain_Array = ",
         crgData->Offset_Prior_Rcvy_Domain_Array);

  /* Print the number of nodes in the prior recovery domain array.      */
  printf("%s %d \n",
         "  Number_Nodes_Prior_Rcvy_Domain = ",
         crgData->Number_Nodes_Prior_Rcvy_Domain);

  /* Print the prior recovery domain if one was passed.                 */
  if (crgData->Offset_Prior_Rcvy_Domain_Array) {
    printRcvyDomain("   The prior recovery domain:",
                    crgData->Number_Nodes_Prior_Rcvy_Domain,
                    (Qcst_Rcvy_Domain_Array1_t *)
                ((char *)crgData + crgData->Offset_Prior_Rcvy_Domain_Array));
  }

  return;
}  /* end printParms                                                  */


/*********************************************************************/
/*                                                                   */
/* Print a string for the action code.                               */
/*                                                                   */
/*********************************************************************/
static void printActionCode(unsigned int ac) {

  char *code;
  switch (ac) {
    case QcstCrgAcInitialize: code = "QcstCrgAcInitialize";
                              break;
    case QcstCrgAcStart:      code = "QcstCrgAcStart";
                              break;
    case QcstCrgAcRestart:    code = "QcstCrgAcRestart";
                              break;
    case QcstCrgAcEnd:        code = "QcstCrgAcEnd";
                              break;
    case QcstCrgAcDelete:     code = "QcstCrgAcDelete";
```

```
                            break;
    case QcstCrgAcReJoin:     code = "QcstCrgAcReJoin";
                            break;
    case QcstCrgAcFailover:   code = "QcstCrgAcFailover";
                            break;
    case QcstCrgAcSwitchover: code = "QcstCrgAcSwitchover";
                            break;
    case QcstCrgAcAddNode:    code = "QcstCrgAcAddNode";
                            break;
    case QcstCrgAcRemoveNode: code = "QcstCrgAcRemoveNode";
                            break;
    case QcstCrgAcChange:     code = "QcstCrgAcChange";
                            break;
    case QcstCrgAcDeleteCommand: code = "QcstCrgAcDeleteCommand";
                            break;
    case QcstCrgAcUndo:       code = "QcstCrgAcUndo";
                            break;
    case QcstCrgEndNode:      code = "QcstCrgEndNode";
                            break;
    case QcstCrgAcAddDevEnt:  code = "QcstCrgAcAddDevEnt";
                            break;
    case QcstCrgAcRmvDevEnt:  code = "QcstCrgAcRmvDevEnt";
                            break;
    case QcstCrgAcChgDevEnt:  code = "QcstCrgAcChgDevEnt";
                            break;
    case QcstCrgAcChgNodeStatus: code = "QcstCrgAcChgNodeStatus";
                            break;
    case QcstCrgAcCancelFailover: code = "QcstCrgAcCancelFailover";
                            break;
    case QcstCrgAcVerificationPhase: code = "QcstCrgAcVerificationPhase";
                            break;
    default:                  code = "unknown action code";
                            break;
  }
  printf("%s", code);

  return;
}  /* end printActionCode                                                  */


/*****************************************************************************/
/*                                                                           */
/* Print the CRG status.                                                     */
/*                                                                           */
/*****************************************************************************/
static void printCrgStatus(int status) {

  char * str;
  switch (status) {
    case QcstCrgActive:               str = "QcstCrgActive";
                                      break;
    case QcstCrgInactive:             str= "QcstCrgInactive";
                                      break;
    case QcstCrgIndoubt:              str = "QcstCrgIndoubt";
                                      break;
    case QcstCrgRestored:             str = "QcstCrgRestored";
                                      break;
    case QcstCrgAddnodePending:       str = "QcstCrgAddnodePending";
                                      break;
    case QcstCrgDeletePending:        str = "QcstCrgDeletePending";
                                      break;
    case QcstCrgChangePending:        str = "QcstCrgChangePending";
                                      break;
    case QcstCrgEndCrgPending:        str = "QcstCrgEndCrgPending";
                                      break;
    case QcstCrgInitializePending:    str = "QcstCrgInitializePending";
                                      break;
```

```
    case QcstCrgRemovenodePending:    str = "QcstCrgRemovenodePending";
                                      break;
    case QcstCrgStartCrgPending:      str = "QcstCrgStartCrgPending";
                                      break;
    case QcstCrgSwitchOverPending:    str = "QcstCrgSwitchOverPending";
                                      break;
    case QcstCrgDeleteCmdPending:     str = "QcstCrgDeleteCmdPending";
                                      break;
    case QcstCrgAddDevEntPending:     str = "QcstCrgAddDevEntPending";
                                      break;
    case QcstCrgRmvDevEntPending:     str = "QcstCrgRmvDevEntPending";
                                      break;
    case QcstCrgChgDevEntPending:     str = "QcstCrgChgDevEntPending";
                                      break;
    case QcstCrgChgNodeStatusPending: str = "QcstCrgChgNodeStatusPending";
                                      break;
    default: str = "unknown CRG status";
  }
  printf("%s \n", str);

  return;
}  /* end printCrgStatus                                                    */


/*****************************************************************************/
/*                                                                           */
/* Print the recovery domain.                                                */
/*                                                                           */
/*****************************************************************************/
static void printRcvyDomain(char *str,
                            unsigned int count,
                            Qcst_Rcvy_Domain_Array1_t *rd) {

  unsigned int i;
  printf("\n %s \n", str);
  for (i=1; i&lt;=count; i++) {
    printStr("    Node_ID = ", rd->Node_ID, sizeof(Qcst_Node_Id_t));
    printf("%s %d \n", "      Node_Role = ", rd->Node_Role);
    printf("%s", "      Membership_Status = ");
    switch (rd->Membership_Status) {
      case 0: str = "Active";
              break;
      case 1: str = "Inactive";
              break;
      case 2: str = "Partition";
              break;
      default: str = "unknown node status";
    }
    printf("%s \n", str);
    rd++;
  }
  return;
}  /* end printRcvyDomain                                                   */

/*****************************************************************************/
/*                                                                           */
/* Concatenate a null terminated string and a non null terminated string   */
/* and print it.                                                             */
/*                                                                           */
/*****************************************************************************/
static void printStr(char *s1, char *s2, unsigned int len) {

  char buffer[132];
  memset(buffer, 0x00, sizeof(buffer));
  memcpy(buffer, s1, strlen(s1));
  strncat(buffer, s2, len);
```

```
  printf("%s \n", buffer);
  return;
}  /* end printStr                                                  */
```

Plánování pro klastry

Zjistěte, co potřebujete před vytvořením klastrů na serverech iSeries. Zjistěte předpoklady pro vytvoření klastrů a seznamte se s pokyny k návrhu klastru. Přečtěte si také rady k nastavení sítě a některé pokyny týkající se výkonnosti klastrů.

Toto téma popisuje požadavky, které budete muset splnit před použitím klastrování. Následující témata uvádějí obecné pojmy, koncepce, požadavky a pokyny pro návrh klastrového řešení.

Řešení pro konfiguraci a správu klastrů

Služby klastrových prostředků poskytují základní klastrovou infrastrukturu. Existuje několik způsobů, které umožňují využít schopností klastrování pomocí Služeb klastrových prostředků.

Služby klastrových prostředků systému i5/OS na serveru iSeries poskytují základní infrastrukturu, která umožňuje použití klastru. Služby klastrových prostředků poskytují sadu integrovaných služeb, které udržují klastrovou topologii, provádějí pulsaci a umožňují vytváření a administraci klastrové konfigurace a skupin klastrových prostředků. Služby klastrových prostředků poskytují také spolehlivé funkce zpráv, které sledují každý uzel v klastru a zajišťují, že všechny uzly mají shodné informace o stavu klastrových prostředků.

Zatímco Služby klastrových prostředků poskytují základní infrastrukturu klastru, existuje několik způsobů, které umožňují využít výhod těchto klastrovacích schopností. Každá z nich má své zvláštní výhody a schopnosti.

Důležité: Použijte výhradně jedno z těchto řešení. Pokusíte-li se při vytváření a správě klastru použít více než jedno řešení, může dojít ke konfliktům, problémům nebo nepředvídatelným událostem. Informace, které najdete v aplikaci iSeries Information Center, se týkají procedur produktu iSeries Navigator a rovněž příkazů a rozhraní API pro služby klastrových prostředků. Použijete-li řešení obchodního partnera IBM pro klastrové aplikační programové prostředky, přečtěte si dokumentaci dodávanou s produktem. Najdete v ní procedurální informace týkající se prováděných úloh.

Správa klastrů pomocí produktu iSeries Navigator

IBM nabízí rozhraní pro správu klastru, které je dostupné pomocí produktu iSeries Navigator a přístupné pomocí Volby 41 (i5/OS - HA Switchable Resources).

Toto rozhraní umožňuje vytvořit a spravovat klastr, který k zajištění dostupnosti dat používá přepínatelné nezávislé ASP. Také vám umožňuje vytvářet a spravovat klastry, skupiny CRG, administrativní domény klastru a prostředky.

Důležité: Rozhraní produktu iSeries Navigator pro správu klastru neobsahuje všechny schopnosti poskytované službami klastrových prostředků. Produkt iSeries Navigator poskytuje mnoho funkcí potřebných ke konfiguraci a správě klastru, ale určité schopnosti jsou dostupné pouze pomocí klastrových příkazů a rozhraní API nebo případně pomocí klastrových aplikací typu middleware dodávaných obchodními partnery IBM (v závislosti na konkrétní aplikaci). Klastrová architektura iSeries například podporuje maximálně 128 uzlů v klastru, avšak rozhraní produktu iSeries Navigator podporuje maximálně pouze čtyři uzly v klastru. Při správě klastru pomocí produktu iSeries Navigator můžete použít průvodce, který vás povede jednotlivými kroky vytvoření jednoduchého klastru tvořeného čtyřmi uzly. Potřebujete-li klastr obsahující více uzlů, použijte raději klastrové příkazy a rozhraní API od IBM nebo produkty typu klastrový middleware dodané obchodními partnery IBM.

K provedení jiných úloh souvisejících s klastrem můžete použít také produkt iSeries Navigator. Toto jsou některé z těchto úloh:

- Přidání uzlu do existujícího klastru.
- Přidání přepínatelných zařízení do klastru.

- Přidání přepínatelných aplikací do klastru.
- Přidání skupiny přepínatelných dat do klastru.
- Změna role uzlů v doméně obnovy.
- Změna popisu klastru.
- Vymazání klastru.
- Spuštění klastrování.
- Zastavení klastrování.
- Zobrazení zpráv o činnosti klastru.
- Vytvoření administrativní domény klastru
- Přidání záznamu monitorovaných prostředků

Kompletní seznam úloh souvisejících s klastrem, které jsou k dispozici v produktu iSeries Navigator, naleznete v online nápovědě ke klastrům.

Poznámka: Rozhraní produktu iSeries Navigator pro správu klastru nepodporuje replikaci logických objektů. Pro replikaci uvažte použití klastrových produktů nabízených obchodními partnery IBM dodávajícími produkty k zajištění vysoké dostupnosti.

Související pojmy

iSeries Navigator

"Klastrové příkazy a rozhraní API"
Služby klastrových prostředků i5/OS poskytují sadu CL příkazů, API a systémových prostředků, které mohou být použity poskytovateli aplikací iSeries nebo jejich zákazníky k zvýšení dostupnosti svých aplikací.

"Obchodní partneři IBM pro klastrové aplikační programové prostředky a dostupné klastrovací produkty" na stránce 79
Můžete si zakoupit produkt od obchodního partnera IBM pro klastrové aplikační programové prostředky, který nabízí replikační funkce, jež jsou nedílnou součástí klastrování a jež zjednodušují vytváření a správu klastrů.

"Běžné problémy s klastry" na stránce 132
Zde jsou popsány některé z nejčastějších problémů, ke kterým dochází v klastru, a způsoby, jak se těchto problémů vyvarovat a jak provést nápravu chyb.

Související odkazy

"Časté otázky ke správě klastru pomocí produktu iSeries Navigator" na stránce 141
Otázky a odpovědi týkající se grafického uživatelského rozhraní produktu iSeries Navigator pro vytváření a správu klastrů.

Klastrové příkazy a rozhraní API

Služby klastrových prostředků i5/OS poskytují sadu CL příkazů, API a systémových prostředků, které mohou být použity poskytovateli aplikací iSeries nebo jejich zákazníky k zvýšení dostupnosti svých aplikací.

Pomocí klastrových příkazů jazyka CL a programovacích rozhraní API můžete psát vlastní uživatelské aplikace pro konfiguraci a správu klastru. Tyto příkazy a rozhraní API využívají technologii poskytovanou službami klastrových prostředků, které jsou dodávány jako součást systému i5/OS.

QUSRTOOL

Služby klastrových prostředků poskytují také sadu ukázkových příkazů v knihovně QUSRTOOL, které provádějí tytéž činnosti jako rozhraní API, jež nepodporují příkazové rozhraní. Příkazy z knihovny QUSRTOOL mohou být užitečné v určitých prostředích. Mohou například sloužit ke změně monitorování pulsu nebo k rozesílání informací po klastru. Další informace o těchto ukázkových příkazech najdete v členu TCSTINFO v souboru QUSRTOOL/QATTINFO. Knihovna QUSRTOOL obsahuje také ukázkový ukončovací program aplikační CRG. Ukázkový zdrojový kód lze použít jako základ při psaní ukončovacího programu. Ukázkový zdroj - TCSTDTAEXT v souboru QATTSYSC - obsahuje zdrojový tvar programu pro vytvoření datových oblastí QCSTHAAPPI a QCSTHAAPP0 a dále soubor QACSTOSDS (specifikátor objektů).

Související úlohy

"Přidání uzlu do klastru" na stránce 99
Pomocí aplikace iSeries Navigator nebo příkazů můžete přidat uzel do klastru.

Popisy CL příkazů a rozhraní API:

Existuje mnoho rozhraní API a CL příkazů, které můžete použít pro konfiguraci, aktivaci a správu klastrů, klastrových uzlů a skupin klastrových prostředků.

Následující tabulky obsahují jméno a stručný popis dostupných příkazů a rozhraní API pro řízení klastru a skupin klastrových prostředků (CRG). Klastrové CL příkazy jsou k dispozici pouze v systému OS/400 verze V5R2M0 nebo novější.

První tabulka obsahuje příkazy a rozhraní API pro konfiguraci, aktivaci a správu **klastru a uzlů** v klastru. Druhá tabulka obsahuje příkazy a rozhraní API pro konfiguraci, aktivaci a správu **skupin klastrových prostředků (CRG)** v klastru. Třetí tabulka obsahuje příkazy pro konfiguraci a správu **administrativní domény klastru**. Čtvrtá tabulka obsahuje popisy jednotlivých rozhraní Integrated Operating System API, které lze použít k přidání a odstranění záznamů o monitorovaném prostředku z administrativní domény klastru.

Tabulka 8. Popisy CL příkazů a rozhraní API pro řízení klastru

Popis	CL příkaz pro řízení klastru	Jméno rozhraní API pro řízení klastru
Přidání záznamu klastrového uzlu Přidá uzel do seznamu členů existujícího klastru. Přiřadí také adresy IP rozhraní, které budou používány pro komunikaci klastru.	ADDCLUNODE	QcstAddClusterNodeEntry (Přidání záznamu klastrového uzlu)
Přidání záznamu domény zařízení Přidá uzel do seznamu členů domény zařízení, tak aby se mohl zúčastnit akcí obnovy odolných zařízení. Přidání prvního uzlu do domény zařízení způsobí vytvoření domény zařízení.	ADDDEVDMNE	QcstAddDeviceDomainEntry (Přidání záznamu domény zařízení)
Nastavení verze klastru, změna verze klastru Upraví aktuální verzi klastru na další úroveň, například proto, aby bylo možné v klastru používat novou funkci.	CHGCLUVER	QcstAdjustClusterVersion (Adjust Cluster Version)
Změna záznamu klastrového uzlu Změní pole v záznamu klastrového uzlu. Například lze změnit adresy IP rozhraní používané pro komunikaci klastru.	CHGCLUNODE	QcstChangeClusterNodeEntry (Change Cluster Node Entry)
Změna služeb klastrových prostředků, změna konfigurace klastru Upraví parametry optimalizace výkonu a konfigurace klastru tak, aby vyhovovaly komunikačnímu prostředí sítě používanému pro komunikaci klastru.	CHGCLUCFG	QcstChgClusterResourceServices (Změna služeb klastrových prostředků)
Vytvoření klastru Vytvoří nový klastr tvořený jedním nebo více uzly.	CRTCLU	QcstCreateCluster (Vytvoření klastru)

Tabulka 8. Popisy CL příkazů a rozhraní API pro řízení klastru (pokračování)

Popis	CL příkaz pro řízení klastru	Jméno rozhraní API pro řízení klastru
Vymazání klastru Odstraní existující klastr. Služby klastrových prostředků jsou ve všech aktivních klastrových uzlech ukončeny a jsou odebrány z klastru.	DLTCLU	QcstDeleteCluster (Vymazání klastru)
Ukončení klastrového uzlu Ukončí služby klastrových prostředků v jednom nebo ve všech uzlech seznamu členů existujícího klastru. Uzel se stane pro klastr nedostupným, dokud nebude restartován pomocí rozhraní Spuštění klastrového uzlu.	ENDCLUNOD	QcstEndClusterNode (Ukončení klastrového uzlu)
Výpis informací o klastru, zobrazení informací o klastru Vyvolá informace o klastru. Například může být vrácen úplný seznam členů klastru.	DSPCLUINF	QcstListClusterInfo (Výpis informací o klastru)
Výpis informací o doménách zařízení, zobrazení informací o klastru Vypíše informace o doménách zařízení. Například může být vrácen seznam momentálně definovaných domén zařízení.	DSPCLUINF	QcstListDeviceDomainInfo (Výpis informací o doménách zařízení)
Odstranění záznamu klastrového uzlu Odebere uzel ze seznamu členů klastru. Uzel je odstraněn ze všech domén obnovy, v uzlu jsou ukončeny klastrové operace a jsou z něj vymazány všechny objekty služeb klastrových prostředků.	RMVCLUNODE	QcstRemoveClusterNodeEntry (Odstranění záznamu klastrového uzlu)
Odstranění záznamu domény zařízení Odebere uzel ze seznamu členů domény zařízení. Pokud to byl poslední uzel v doméně zařízení, je zároveň z klastru vymazána doména zařízení.	RMVDEVDMNE	QcstRemoveDeviceDomainEntry (Odstranění záznamu domény zařízení)
Načtení informací o klastru, zobrazení informací o klastru Vyvolá informace o klastru. Vrátí například jméno klastru a aktuální verzi klastru.	DSPCLUINF	QcstRetrieveClusterInfo (Vyvolání informací o klastru)
Načtení informací o službách klastrových prostředků, zobrazení informací o klastru Vyvolá informace o parametrech konfigurace a optimalizace výkonu služeb klastrových prostředků.	DSPCLUINF	QcstRetrieveCRSInfo (Vyvolání informací o službách klastrových prostředků)
Spuštění klastrového uzlu Spustí služby klastrových prostředků v uzlu, který je součástí klastru, ale není momentálně aktivní.	STRCLUNOD	QcstStartClusterNode (Spuštění klastrového uzlu)

Tabulka 8. Popisy CL příkazů a rozhraní API pro řízení klastru (pokračování)

Popis	CL příkaz pro řízení klastru	Jméno rozhraní API pro řízení klastru
Práce s klastrem Zobrazuje a pracuje s klastrovými uzly a objekty.	WRKCLU	Žádný

Tabulka 9. Popisy CL příkazů a rozhraní API pro skupinu klastrových prostředků

Popis	CL příkaz pro řízení skupin klastrových prostředků	Rozhraní API pro řízení skupin klastrových prostředků
Přidání záznamu zařízení do skupiny klastrových prostředků Přidá nový záznam zařízení do skupiny klastrových prostředků. Zařízení se stane členem skupiny přepínatelných zařízení.	ADDCRGDEVE	QcstAddClusterResourceGroupDevice (Přidání záznamu zařízení do skupiny klastrových prostředků)
Přidání uzlu do domény obnovy, přidání záznamu uzlu do skupiny klastrových prostředků Přidá nový uzel do domény obnovy existující skupiny klastrových prostředků.	ADDCRGNODE	QcstAddNodeToRcvyDomain (Přidání uzlu do domény obnovy)
Změna skupiny klastrových prostředků Změní atributy skupiny klastrových prostředků. Může například modifikovat IP adresu převzetí pro .	CHGCRG	QcstChangeClusterResourceGroup (Změna skupiny klastrových prostředků)
Změna záznamu zařízení ve skupině klastrových prostředků Změní záznam zařízení ve skupině klastrových prostředků. Může například modifikovat volbu přepnutí konfiguračního objektu do režimu online pro případ přepnutí nebo přepnutí při selhání.	CHGCRGDEVE	QcstChangeClusterResourceGroupDev (Změna záznamu zařízení ve skupině klastrových prostředků)
Vytvoření skupiny klastrových prostředků Vytvoří objekt skupiny klastrových prostředků. Objekt skupiny klastrových prostředků identifikuje doménu obnovy, což je sada uzlů klastru, která plní svou úlohu při obnově.	CRTCRG	QcstCreateClusterResourceGroup (Create Cluster Resource Group)
Výmaz skupiny klastrových prostředků Vymaže skupinu klastrových prostředků (CRG) pouze na lokálním uzlu. Výmaz skupiny prostředků lokálního klastru vyžaduje, aby služby klastrových prostředků nebyly aktivní.	DLTCRG	Žádný
Výmaz skupiny klastrových prostředků, výmaz CRG klastru Vymaže skupinu klastrových prostředků z klastru. Objekt CRG bude vymazán ze všech aktivních uzlů v doméně obnovy.	DLTCRGCLU	QcstDeleteClusterResourceGroup (Vymazání skupiny klastrových prostředků)

Tabulka 9. Popisy CL příkazů a rozhraní API pro skupinu klastrových prostředků (pokračování)

Popis	CL příkaz pro řízení skupin klastrových prostředků	Rozhraní API pro řízení skupin klastrových prostředků
Distribuce informací Doručí informace z uzlu v doméně obnovy skupiny klastrových prostředků (CRG) do ostatních uzlů v doméně obnovy dané skupiny klastrových prostředků.	Žádný	QcstDistributeInformation (Distribuce informací)
Ukončení skupiny klastrových prostředků Deaktivuje odolnost zadané skupiny klastrových prostředků. Po úspěšném provedení tohoto API je stav skupiny klastrových prostředků nastaven na neaktivní.	ENDCRG	QcstEndClusterResourceGroup (Ukončení skupiny klastrových prostředků)
Iniciování přepnutí, změna primárního uzlu skupiny klastrových prostředků Způsobí provedení administrativního přepnutí pro skupinu klastrových prostředků. Doména obnovy bude změněna takto: Aktuální primární uzel se stane posledním záložním uzlem a aktuální první záložní uzel se stane novým primárním uzlem.	CHGCRGPRI	QcstInitiateSwitchover (Inicializace přepnutí)
Výpis informací o skupině klastrových prostředků, zobrazení informací o skupině klastrových prostředků Vygeneruje seznam skupin klastrových prostředků a určitých informací o těchto skupinách v klastru.	DSPCRGINF	QcstListClusterResourceGroups (Výpis skupin klastrových prostředků)
Výpis informací o skupině klastrových prostředků, zobrazení informací o skupině klastrových prostředků Vrátí obsah objektu skupiny klastrových prostředků (CRG). Například bude vrácena doména obnovy spolu s aktuálními rolemi uzlů.	DSPCRGINF	QcstListClusterResourceGroupInf (Výpis informací o skupině klastrových prostředků)
Odstranění záznamu zařízení ze skupiny klastrových prostředků Odebere záznam zařízení ze skupiny klastrových prostředků. Zařízení přestane být přepínatelným prostředkem.	RMVCRGDEVE	QcstRemoveClusterResourceGroupDev (Odstranění záznamu zařízení ze skupiny klastrových prostředků)
Odstranění uzlu z domény obnovy, odstranění záznamu uzlu skupiny klastrových prostředků Odebere uzel z domény obnovy existující skupiny klastrových prostředků. Uzel se nebude nadále účastnit akce obnovy dané skupiny prostředků.	RMVCRGNODE	QcstRemoveNodeFromRcvyDomain (Odstranění uzlu z domény obnovy)

Tabulka 9. Popisy CL příkazů a rozhraní API pro skupinu klastrových prostředků (pokračování)

Popis	CL příkaz pro řízení skupin klastrových prostředků	Rozhraní API pro řízení skupin klastrových prostředků
Spuštění skupiny klastrových prostředků Aktivuje odolnost pro zadanou skupinu klastrových prostředků. Skupina klastrových prostředků se stane v klastru aktivní.	STRCRG	QcstStartClusterResourceGroup (Spuštění skupiny klastrových prostředků)

Poznámka: Služby klastrových prostředků poskytují také sadu ukázkových příkazů v knihovně QUSRTOOL, které provádějí tytéž činnosti jako CL příkazy a rozhraní API popsané výše. Příkazy z knihovny QUSRTOOL mohou být užitečné v určitých prostředích. Mohou například sloužit k snadnému vytvoření klastru pro testování aplikací podporujících klastry. Další informace o těchto ukázkových příkazech najdete v členu TCSTINFO v souboru QUSRTOOL/QATTINFO.

Tabulka 10. Popisy CL příkazů pro administrativní doménu

Popis	CL příkaz pro administrativní doménu	Rozhraní API pro administrativní doménu
Vytvoření administrativní domény Vytváří peer CRG, která představuje administrativní doménu klastru. Jakmile je administrativní doména klastru vytvořena, lze do domény přidat záznamy monitorovaných prostředků (MRE), aby bylo možné synchronizovat změny prostředků. **Poznámka:** Jakmile je vytvořena administrativní doména klastru, můžete pro její správu použít příkazy CRG, které uvádí Tabulka 9 na stránce 76.	CRTADMDMN	Žádný
Výmaz administrativní domény Vymaže skupinu CRG představující administrativní doménu klastru. Po dokončení se všechny záznamy MRE odstraní z domény a změny prostředků, které byly monitorovány, se již nebudou šířit.	DLTADMDMN	Žádný

Tabulka 11. Popisy rozhraní Integrated Operating System API. Kromě těchto CL příkazů pro administrativní doménu existuje také několik aplikačních programovatených rozhraní Integrated Operating System (API), která umožňují přidávat a odstraňovat záznamy prostředků.

Popis	CL příkazy[1]	Integrated Operating System API
Add Monitored Resource Entry Přidá záznam monitorovaného prostředků pro prostředek systému a jeho atributy.	Žádný	QfpadAddMonitoredResourceEntry (Add Monitored Resource Entry)
Remove Monitored Resource Entry Odstraní záznam monitorovaného prostředku (MRE) z adresáře monitorovaných prostředků.	Žádný	QfpadRmvMonitoredResourceEntry (Remove Monitored Resource Entry)
Retrieve Monitored Resource Information Vrací informace o monitorovaných prostředcích.	Žádný	QfpadRtvMonitoredResourceInfo (Retrieve Monitored Resource Information)

Poznámka:

1. Pro tuto funkci neexistuje žádný ekvivalentní CL příkaz. Zdroj pro nepodporovaný příkaz a program pro zpracování volání (CPP) byl poskytnut v knihovně QUSRTOOL. Chcete-li se dozvědět více o tomto zdroji příkazu a CPP, prohlédněte si člen QFPADINFO v souboru QATTINFO.

Související odkazy

Klastrová rozhraní API

Obchodní partneři IBM pro klastrové aplikační programové prostředky a dostupné klastrovací produkty

Můžete si zakoupit produkt od obchodního partnera IBM pro klastrové aplikační programové prostředky, který nabízí replikační funkce, jež jsou nedílnou součástí klastrování a jež zjednodušují vytváření a správu klastrů.

Obchodní partneři IBM pro klastrové aplikační programové prostředky poskytují softwarová řešení pro funkce vyhrazení replikace a správy klastrů. Chcete-li koupit produkt nabízející replikační funkce, které jsou nedílnou součástí klastrování a které zjednodušují vytváření a správu klastrů, kontaktujte obchodního zástupce nebo obchodního partnera IBM. Poskytnou vám úplný seznam produktů umožňujících klastrování nabízených obchodními partnery IBM pro klastrové aplikační programové prostředky.

Produkt pro správu klastrů od obchodního partnera IBM pro klastrové aplikační programové prostředky:

- Nabízí uživatelské rozhraní pro definování a údržbu klastrové konfigurace.
- Poskytuje uživatelské rozhraní pro definování a správu zařízení, dat a aplikačních skupin klastrových prostředků.
- Pomocí použití klastrových API udržuje informace o tom, jaké skupiny klastrových prostředků jsou definovány v klastru a jaké jsou vyžadovány vztahy.
- Vytváří zařízení, data a aplikační skupiny klastrových prostředků.

Produkt pro replikaci od obchodního partnera IBM pro klastrové aplikační programové prostředky:

- Vytváří struktury řízení aplikačních programových prostředků určujících data a objekty, které mají být odolné.
- Vytváří skupinu klastrových prostředků pro kritická data a přiřazuje tento objekt k jeho kontrolním strukturám.
- Poskytuje ukončovací program pro datovou skupinu klastrových prostředků.

Související úlohy

"Přidání uzlu do klastru" na stránce 99
Pomocí aplikace iSeries Navigator nebo příkazů můžete přidat uzel do klastru.

Požadavky na klastry

Zde jsou popsány požadavky na hardware, software a komunikaci při používání klastrů.

Požadavky na používání klastrů se liší podle toho, které schopnosti klastru si zvolíte, že chcete používat. Můžete se například rozhodnout používat jednoduchý, dvouuzlový klastr a využít možnosti logické replikace. Nebo si můžete zvolit klastr, který bude využívat přepínané disky a přepínatelná nezávislá ASP.

Související pojmy

"Příklady: Klastrové konfigurace" na stránce 115
Tyto příklady typických implementací klastrů vám pomohou porozumět, kdy a za jakých podmínek může být použití klastrů výhodné.

Hardwarové požadavky pro klastry

Pro použití klastrování je kompatibilní jakýkoli model iSeries, který je schopen spouštět operační systém i5/OS verze V4R4M0 nebo novější.

Kromě toho byste měli zajistit ochranu před výpadkem proudu pomocí externího zdroje nepřerušitelného napájení (UPS) nebo podobného zařízení. Jinak by měl náhlý výpadek proudu v klastrovém uzlu za následek stav rozdělení klastru (namísto přepnutí při selhání).

Klastrování využívá schopností výběrového vysílání protokolu IP (Internet Protocol). Výběrové vysílání dobře nemapuje na všechny typy fyzických médií. Další informace o omezení výběrového vysílání, která se mohou týkat vašeho hardwaru, naleznete v tématu TCP/IP Setup.

Pokud plánujete, že budete v klastru používat nezávislá ASP, prostudujte si informace o nezávislých ASP uvedené v tématu Hardwarové požadavky. Můžete také chránit disky pomocí zrcadlení nebo pomocí RAID. Použití těchto řešení v primárním systému zabrání přepnutí při selhání v případě, že selže chráněný disk. Je také dobré mít tato řešení v záložním systému pro případ, že by došlo k přepnutí při selhání. Další podrobnosti najdete v tématu Ochrana disku.

Poznámka: Podrobné informace o jiných požadavcích, které jsou zapotřebí před konfigurací klastrů, najdete v tématu "Kontrolní seznam pro konfiguraci klastru" na stránce 90.

Související pojmy

Zdroj nepřerušitelného napájení

"Část klastru" na stránce 27
Část klastru je podmnožina aktivních uzlů klastru, která vzniká rozdělením klastru v důsledku selhání komunikace. Členové části klastru mezi sebou udržují navzájem spojení.

"Přepnutí při selhání" na stránce 17
K *přepnutí při selhání* dochází, když se server v klastru automaticky přepne na jeden nebo více záložních serverů v případě selhání systému.

Softwarové a licenční požadavky pro klastry

K tomu, abyste mohli používat klastrování, musíte mít správný software a licence.

1. OS/400(R) V4R4M0 nebo novější konfigurovaný pro TCP/IP (TCP/IP Connectivity Utilities).
2. Řešení pro klastrovou konfiguraci a správu softwaru. Může jím být:
   - Správa klastrů pomocí produktu iSeries Navigator
   - Řešení od obchodního partnera IBM pro klastrové aplikační programové prostředky
   - Vlastní aplikační program pro správu klastrů napsaný pomocí příkazů a rozhraní API pro služby klastrových prostředků
3. Viz "Kontrolní seznam pro konfiguraci klastru" na stránce 90

Důležité: Plánujete-li implementaci ASP, abyste mohli využít výhod přepínatelných zařízení, existují ještě další požadavky. Podrobné informace najdete v tématu Plánování pro nezávislá ASP.

Související pojmy

“Řešení pro konfiguraci a správu klastrů” na stránce 72
Služby klastrových prostředků poskytují základní klastrovou infrastrukturu. Existuje několik způsobů, které umožňují využít schopností klastrování pomocí Služeb klastrových prostředků.

“Klastrová verze” na stránce 12
Klastrová verze představuje úroveň funkce dostupné v klastru.

Komunikační požadavky pro klastry

Ve svém klastrovém prostředí můžete použít libovolný typ komunikačních médií, pokud podporují protokol IP (Internet Protocol).

Služby klastrových prostředků používají ke komunikaci mezi uzly pouze protokoly TCP/IP. Jsou podporovány sítě LAN (Local area networks), WAN (Wide area networks), OptiConnect SAN (System area networks) nebo jakákoli kombinace těchto zařízení pro připojení. Vaše volba by měla být založena na těchto činitelích:

- Objemu transakcí
- Požadavků na dobu odezvy
- Vzdálenosti mezi uzly
- Výše nákladů

Můžete zvážit stejné okolnosti také při určování médií, která budou použita pro spojení primárních a záložních míst prostředků. Když plánujete klastr, doporučujeme, abyste určili jeden nebo více záložních uzlů na vzdálených místech kvůli přežití po ztrátě serveru.

Chcete-li zabránit problémům s výkonem, které by mohly být způsobeny nedostatečnou kapacitou, musíte ohodnotit komunikační média, která používáte pro manipulaci s objemem informací posílaným z jednoho uzlu do druhého. Můžete zvolit, kterým fyzickým médiím dáváte přednost: Token-ring, Ethernet, ATM (asynchronous transfer mode), SPD OptiConnect, HSL (High-Speed Link) OptiConnect nebo Virtual OptiConnect (vysokorychlostní vnitřní spojení mezi logickými částmi).

HSL OptiConnect je technologie poskytovaná softwarem OptiConnect for i5/OS (i5/OS Option 23 - i5/OS OptiConnect). Můžete ji použít pro vytváření vysoce dostupných řešení. HSL OptiConnect je síť SAN, která poskytuje vysokorychlostní, dvoubodovou připojitelnost mezi klastrovými uzly pomocí technologie HSL (High Speed Link) Loop. HSL OptiConnect vyžaduje standardní kabely HSL a nepotřebuje žádný další hardware.

Pro přepínatelný hardware, nazývaný také jako odolné zařízení CRG, potřebujete ve vašem prostředí přepínatelné nezávislé ASP. V prostředí s logickými částmi se jedná o kolekci diskových jednotek na sběrnici, která je sdílena logickými částmi, nebo o kolekci diskových jednotek, které jsou připojeny k procesoru I/O přiřazenému ke společné oblasti I/O. V prostředí s více systémy jde o jednu nebo více přepínatelných rozšiřujících jednotek (věží) řádně konfigurovaných na HSL Loop a obsahujících také systémy v doméně obnovy. Tato přepínatelná věž může být použita také v prostředí LPAR. Více informací o plánování přepínatelného hardwaru a nezávislých ASP najdete v tématu Plánování pro nezávislá ASP.

Poznámka: Používáte-li adaptéry 2810 LAN používající **pouze** TCP/IP a nepoužíváte-li SNA nebo IPX, můžete zvýšit výkon adaptéru na serveru V4R5M0 tím, že pomocí příkazu WRKLIND (Práce s popisy linky) pro popis vaší specifické linky zadáte Enable only for TCP(*YES). Enable only for TCP(*YES) je nastaveno automaticky ve verzi V5R1M0 a vyšších verzích.

Související informace

OptiConnect for i5/OS

Návrh klastru

Určete své potřeby pro stanovení toho, jak navrhnout klastr.

Jelikož existuje řada způsobů, jak používat klastrování v závislosti na tom, čeho chcete dosáhnout, je důležité strávit určitý čas určením vašich potřeb, abyste mohli navrhnout váš klastr.

Návrh sítě pro klastry

Před konfigurací sítí pro klastrování musíte provést pečlivé plánování a určitou předklastrovou konfiguraci zahrnující TCP/IP.

Je důležité, abyste si před konfigurací klastru přečetli níže uvedená témata. Řeknou vám o:

- Nastavení IP adres
- Nastavení atributů konfigurace TCP/IP
- Zabránění rozdělení klastru

Informace o nastavení nezávislých komunikačních cest a o tom, zda potřebujete mít vyhrazenou síť pro klastrování, najdete v tématu Vyhrazení sítě pro klastry.

Nastavení IP adres:

Protože Služby klastrových prostředků používají ke komunikaci s jinými klastrovými uzly *pouze* protokol IP, musejí být všechny klastrové uzly *dosažitelné pomocí IP*.

To znamená, že musíte mít konfigurovaná rozhraní IP, abyste mohli spojit uzly v klastru. Tyto IP adresy musejí být nastaveny buď správcem sítě manuálně ve směrovacích tabulkách TCP/IP na každém uzlu, nebo mohou být generovány přenosovými protokoly na směrovačích v síti. Směrovací tabulka TCP/IP je mapa, kterou klastrování používá k vyhledání každého uzlu. Proto musí mít každý uzel svou *jedinečnou* IP adresu. Každý uzel může mít přiřazeny až dvě IP adresy. Tyto adresy nesmějí být za žádných okolností změněny jinými aplikacemi síťové komunikace. Při přiřazování každé adresy se ujistěte, že jste vzali v úvahu druh komunikační linky, který adresa používá. Dáváte-li přednost použití určitého typu komunikačního média, ujistěte se, že jste konfigurovali první IP adresu pro použití vámi preferovaného média. Spolehlivá funkce zpráv a monitorování pulsu (heartbeat) pokládají první IP adresu za preferenční. Všechny IP adresy v uzlu musejí být schopny dosáhnout jiné IP adresy v klastru. Jedna adresa může dosáhnout jiné adresy, jestliže můžete v obou směrech používat ping a trasování zprávy UDP.

Poznámka: Musíte si být jisti, že pro klastrování je aktivní adresa zkratovací smyčky (127.0.0.1). Tato adresa, která se používá pro posílání zpráv zpět do lokálního uzlu, je obvykle nastavena předvoleně jako aktivní. Když však bude omylem ukončena, nebude fungovat klastrové posílání zpráv, dokud tuto adresu znovu nespustíte.

Související pojmy

“Funkce spolehlivých zpráv” na stránce 26
Funkce spolehlivých zpráv služeb klastrových prostředků sleduje každý klastrový uzel a zajišťuje, aby všechny uzly měly konzistentní informace o stavu klastrových prostředků.

“Monitorování pulsu (heartbeat)” na stránce 25
Monitorování pulsu je jednou z funkcí služeb klastrových prostředků, která zajišťuje, že každý uzel je aktivní: zasílá signál z jednotlivých uzlů klastru na všechny další uzly v klastru a tím ověřuje, že jsou stále aktivní.

Nastavení atributů konfigurace TCP/IP:

Pro umožnění Služby klastrových prostředků jsou v síti vyžadována určitá nastavení konfigurace TCP/IP.

Před přidáním uzlu do klastru musíte nastavit tyto atributy:

- Jestliže plánujete používat server iSeries jako směrovač pro komunikaci s jinými sítěmi a nespouštíte na serveru žádné další přenosové protokoly, nastavte postoupení datagramu pomocí příkazu CHGTCPA (Změna atributů TCP/IP).
- Nastavte server INETD na START. Informace o spuštění serveru INETD najdete v tématu Server INETD.
- Pomocí příkazu CHGTCPA (Změna atributů TCP/IP) nastavte Protokol uživatelských datagramů (UDP) CHECKSUM na *YES.
- Pokud používáte ke spojení sítí Token-ring můstky, nastavte doručování MCAS na *YES.

- Používáte-li ke komunikaci mezi klastrovými uzly OptiConnect for i5/OS, spusťte podsystém QSOC pomocí STRSBS(QSOC/QSOC).

Rady: Komunikace klastru:

Při vytváření komunikačních cest vezměte v úvahu tyto rady.

- Používejte komunikační linky s šířkou pásma, která je dostatečná pro zpracování neklastrové aktivity i pro funkci monitorování pulsu (heartbeat) u klastrů, a pokračujte v monitorování zvýšené aktivity.
- Chcete-li dosáhnout vysoké spolehlivosti, nekonfigurujte jedinou komunikační cestu spojující jednotlivé uzly.
- Nepřetěžujte linku, která je zodpovědná za komunikaci s uzlem.
- Eliminujte co možná nejvíce jediných bodů selhání například tím, že do téhož adaptéru, procesoru IOP nebo věže přivedete dvě komunikační linky.
- Pokud vašimi komunikačními linkami prochází mimořádně velký objem dat, uvažujte o umístění replikace dat a monitorování pulsu (heartbeat) do samostatných sítí.
- Používáte-li výběrové vysílání protokolem IP, seznamte se s publikací TCP/IP Configuration and Reference, kde naleznete omezení výběrového vysílání, která mohou platit pro různé typy fyzických médií.
- Výběrové vysílání protokolem uživatelských datagramů (UDP) se v klastrové komunikační infrastruktuře přednostně používá k posílání správních informací klastru mezi klastrovými uzly. Jestliže fyzické médium podporuje schopnosti výběrového vysílání, komunikace klastru využívá výběrové vysílání UDP k rozesílání zpráv pro správu klastru do všech lokálních klastrových uzlů, které podporují tutéž adresu podsítě. Zprávy posílané do uzlů ve vzdálených sítích jsou vždy posílány pomocí dvoubodových metod UDP. Komunikace klastru se nespoléhá na schopnost směrování zpráv výběrového vysílání.
- Provoz při výběrovém vysílání, který podporuje posílání zpráv pro správu klastru, má přirozenou tendenci ke kolísání. V závislosti na počtu uzlů dané lokální sítě (LAN) (která podporuje společnou adresu podsítě) a na složitosti struktury správy klastru zvolené administrátorem klastru může rychlost přenosu paketů výběrového vysílání vztahujícího se ke klastru snadno překročit 40 paketů za sekundu. Kolísání tohoto druhu může mít negativní vliv na starší vybavení sítě. Příkladem jsou problémy zahlcení u zařízení LAN, která plní úlohu agentů SNMP (Simple Network Management Protocol), protože u nich je nutné vyhodnotit každý paket výběrového vysílání UDP. Některá starší síťová zařízení nemají pro tento typ provozu dostatečnou šířku pásma. Je třeba, abyste vy nebo správce sítě zkontrolovali kapacitu sítí zpracovávajících provoz výběrovým vysíláním UDP a ujistili se, že klastrování nebude mít negativní dopad na rychlost sítí.

  Související pojmy

  “Plánování pro logickou replikaci” na stránce 87
  Pomocí logické replikace je udržováno více kopií dat. Data jsou replikována nebo kopírována z primárního uzlu v klastru do záložních uzlů označených v doméně obnovy. Když dojde v primárním uzlu k výpadku, data zůstanou k dispozici, neboť označený záložní uzel převezme roli primárního bodu přístupu.

  “Funkce spolehlivých zpráv” na stránce 26
  Funkce spolehlivých zpráv služeb klastrových prostředků sleduje každý klastrový uzel a zajišťuje, aby všechny uzly měly konzistentní informace o stavu klastrových prostředků.

  Související informace

  Nastavení TCP/IP

Zabránění rozdělení klastru:

Typickému rozdělení klastru souvisejícího se sítí se lze nejlépe vyhnout tím, že nakonfigurujete redundantní komunikační cesty mezi všemi uzly v klastru.

Nezávislá komunikační cesta znamená, že máte mezi dvěma uzly v klastru konfigurovány dvě linky. Pokud by došlo k selhání na první komunikační cestě, druhá komunikační cesta může převzít její funkci a udržet komunikaci mezi uzly v chodu, a tak minimalizovat podmínky, které mohou vést k rozdělení klastru v jednom nebo více uzlech klastru. Při konfiguraci těchto cest je důležité určit, zda obě komunikační linky vedou do stejného adaptéru v systému. Pokud tomu tak je, představují tyto linky stále riziko v případě, že tento adaptér selže.

Je však třeba poznamenat, že rozdělení klastru není možné se vždy vyhnout. Jestliže váš systém zaznamenává ztrátu výkonu nebo když dochází k selhání hardwaru, klastr je možná rozdělený.

Související pojmy

"Část klastru" na stránce 27
Část klastru je podmnožina aktivních uzlů klastru, která vzniká rozdělením klastru v důsledku selhání komunikace. Členové části klastru mezi sebou udržují navzájem spojení.

"Rady: Komunikace klastru" na stránce 83
Při vytváření komunikačních cest vezměte v úvahu tyto rady.

"Chyby logických částí" na stránce 134
Některé podmínky klastru lze snadno opravit. Pokud došlo k rozdělení klastru, můžete se naučit, jak provést nápravu chyb. Toto téma také popisuje, jak se vyvarovat rozdělení klastru, a udává příklad způsobu opětovného sloučení jednotlivých částí.

Vyhrazení sítě pro klastry:

Během běžných operací bude základní komunikace klastrování minimální. Velmi však doporučujeme, abyste měli konfigurovány nezávislé komunikační cesty pro každý uzel v klastru.

Tím že konfigurujete dvě linky, můžete vyhradit jednu linku pro provoz klastrování a druhá linka může obsluhovat běžný provoz a být zároveň záložní linkou pro případ, že vyhrazená linka pro klastrování selže.

Související pojmy

"Zabránění rozdělení klastru" na stránce 83
Typickému rozdělení klastru souvisejícího se sítí se lze nejlépe vyhnout tím, že nakonfigurujete redundantní komunikační cesty mezi všemi uzly v klastru.

Klastry s více vydáními

Vytváříte-li klastr, který bude zahrnovat uzly s více klastrovými verzemi, je třeba při vytváření takového klastru provést určité kroky.

Aktuální klastrová verze bude standardně nastavena na potenciální klastrovou verzi prvního uzlu přidaného do klastru. Takový postup je vhodný v případě, že tento uzel má nejnižší úroveň verze, která má být v klastru. Když je však uzel na novější úrovni verze, nebudete následně moci přidávat uzly s nižší úrovní verze. Alternativou je použit při vytváření klastru hodnoty cílové klastrové verze a nastavení aktuální klastrové verze o jednu verzi níže, než je potenciální verze klastru prvního uzlu přidaného do klastru.

Vezměme si příklad, kdy má být vytvořen dvouuzlový klastr. Uzly pro tento klastr jsou:

Identifikátor uzlu	Vydání	Potenciální klastrová verze
Uzel A	V5R3	4
Uzel B	V5R4	5

Má-li být klastr vytvořen z uzlu B, je třeba označit, že se bude jednat o klastr se smíšeným vydáním. Cílová klastrová verze musí být nastavena tak, aby označovala, že klastrové uzly budou komunikovat na verzi o jednu nižší, než je potenciální verze uzlu s požadavkem.

Související pojmy

"Klastrová verze" na stránce 12
Klastrová verze představuje úroveň funkce dostupné v klastru.

Určení serverů, které mají být zahrnuty do klastru

K tomu, abyste mohli označit servery, které chcete zahrnout do klastru, musíte určit, které servery jsou schopny poskytnout přiměřenou zálohu pro data a aplikace, jež potřebujte pro provoz vaší činnosti.

Musíte určit:

- Které servery obsahují kritická data a kritické aplikace?
- Které servery budou tvořit zálohu pro tyto systémy?

Odpověďmi na tyto otázky jsou servery, které budete chtít zahrnout do klastru.

Porovnání modelu s primární zálohou a peer modelu

CRG s primární zálohou i peer CRG zajišťují odolnost pro prostředky v klastru; je však důležité chápat jejich rozdíly a použití.

Klastry podporují dva modely pro definování CRG ve vašem prostředí. Jsou definovány úlohy pro model s primární zálohou a pro peer model. V modelu s primární zálohou musejí definovat pořadí. Uzly, které jsou definovány jako záložní uzly, zajišťují přístup k prostředkům, pokud dojde k selhání nějakého uzlu. V peer modelu mají všechny uzly rovnocennou úlohu a mohou zajistit přístup k prostředku; není tam však žádná uspořádanost.

Model s primární zálohou

U modelu s primární zálohou musejí uživatelé definovat uzel jako primární, záložní nebo replikovaný. Tyto úlohy se definují a spravují v rámci domény obnovování. Jestliže byl nějaký uzel definován jako primární přístupový bod pro prostředek, pak ostatní uzly slouží jako záložní pro případ, že by primární uzel selhal.

Peer model

CRS s peer modelem eliminují potřebu definovat uspořádanou doménu obnovy. Pro peer model lze definovat uzly buď jako peer nebo jako replikované. Když jsou uzly definovány jako peer, pak všechny uzly v doméně obnovy jsou stejné a mohou poskytnout přístupový bod k prostředku.

Určení aplikací, které mají být zahrnuty do klastru

Ne všechny aplikace vám nabídnou výhody klastrování.

Aplikace musí být odolná, aby mohla využít schopností přepnutí a přepnutí při selhání, které nabízí klastrování. Odolnost aplikace umožňuje, aby byla aplikace restartována na záložním uzlu, aniž by bylo nutné konfigurovat klienty používají aplikaci. Proto musí aplikace splňovat určité požadavky, aby mohla plně využít schopností nabízených klastrováním. Další informace o odolných aplikacích najdete v tématu Klastrové aplikace.

Plánování pro datovou odolnost

Datové odolnosti je dosaženo tehdy, když jsou data kdykoli k dispozici uživateli nebo aplikaci. Datové odolnosti můžete dosáhnout pomocí logické replikace nebo pomocí přepínatelných nezávislých ASP.

Určení, která data by měla být odolná:

Přečtěte si, které typy dat byste měli zvážit při plánování datové odolnosti.

Určení, která data musíte učinit odolnými, je podobné určení toho, který druh dat potřebujete zálohovat a uložit, když pro systémy připravujete strategii zálohování a obnovy. Potřebujete určit, která data ve vašem prostředí jsou kritická pro zachování vaší činnosti v chodu.

Pokud například obchodujete na Webu, vašimi kritickými daty mohou být:

- Dnešní objednávky
- Zásoby
- Záznamy o zákaznících

Obecně se dá říci, že informace, které neměníte často nebo které nepotřebujete používat každý den, nebudou pravděpodobně vyžadovat, aby byly odolné.

Související pojmy

Plánování strategie zálohování a obnovy

Porovnání logické replikace, přepínání disků a zrcadlení mezi stanovišti:

Toto téma uvádí přehled různých technologií pro odolnost dat, které lze použít spolu s klastry ke zdokonalení v oblasti vysoké dostupnosti.

Odolnost dat umožňuje, aby data zůstala přístupná aplikacím a uživatelům i tehdy, když selže systém, který byl původně hostitelem dat. Volba správné sady technologií pro odolnost dat může být v kontextu vaší celkové strategie na rozvoj podniku složitá a obtížná. Je důležité, abyste porozuměli různým řešením pro odolnost dat, která lze použít ke zvýšení dostupnosti v prostředích s více systémy. Můžete si buď zvolit jedno řešení nebo použít kombinaci těchto technologií, která bude splňovat vaše potřeby.

Další podrobnosti o těchto řešeních naleznete na stránkách Data Resilience Solutions for IBM i5/OS High Availability Clusters. Téma nazvané ″Data Resilience Solutions for IBM i5/OS High Availability Clusters″ obsahuje podrobné porovnání atributů každé z těchto technologií.

Logická replikace

Logická replikace je proces kopírování objektů z jednoho klastrového uzlu do jednoho nebo více dalších uzlů klastru. Tento proces slouží k udržení shodnosti objektů ve všech systémech.

Replikace prostředků umožňuje kopírování objektů (například aplikací a jejich dat) z jednoho klastrového uzlu do jednoho nebo více dalších uzlů klastru. Tento proces udržuje objekty na všech serverech v doméně obnovy replikovaných prostředků totožné. Změníte-li nějaký objekt v jednom klastrovém uzlu, je tato změna replikována do dalších uzlů klastru. Dojde-li k přepnutí při selhání nebo k přepnutí, může záložní uzel bez přerušení převzít roli primárního uzlu. Server či servery, které slouží jako záložní, jsou definovány v doméně obnovy. Jestliže dojde k výpadku na serveru definovaném jako primární uzel domény obnovy a je vyvoláno přepnutí nebo přepnutí při selhání, stane se uzel určený v doméně obnovy jako záložní primárním přístupovým bodem k prostředku.

Replikace vyžaduje vlastnoručně napsanou aplikaci nebo softwarovou aplikaci vytvořenou obchodním partnerem vyvíjejícím middleware pro klastry. Podrobné informace najdete v tématu Plánování logické replikace.

Přepínatelné disky

Přepínatelné disky umožňují, aby byly prostředky, jako jsou data a aplikace, umístěné v rozšiřující jednotce nebo v procesoru IOP na sdílené sběrnici nebo ve společné oblasti I/O logické části, přepínány mezi primárním a záložním klastrovým uzlem. Umožňuje to, aby byla sada diskových jednotek přístupná z druhého serveru (serveru definovaného v doméně obnovy skupiny klastrových prostředků jako záložní uzel) v případě, že na serveru momentálně používajícím tyto diskové jednotky dojde k výpadku a nastane přepnutí při selhání nebo přepnutí.

Chcete-li využívat přepínání prostředků v klastru, musíte používat nezávislá ASP. Další informace najdete v tématu Plánování nezávislých ASP.

Zrcadlení mezi stanovišti

Zrcadlení mezi stanovišti umožňuje spolu s funkcí geografického zrcadlení zrcadlit data mezi disky na stanovištích, která mohou být ve velké zeměpisné vzdálenosti. Tato technologie může sloužit k rozšíření funkcí skupiny klastrových prostředků (CRG) za hranice fyzického propojení komponent. Geografické zrcadlení umožňuje replikovat změny provedené v provozní kopii nezávislého ASP do zrcadlové kopie této části. Při zápisu dat do provozní kopie nezávislého ASP zrcadlí operační systém tato data do druhé kopie této části přes jiný systém. Tento proces udržuje více identických kopií dat.

Dojde-li k přepnutí při selhání nebo k přepnutí, může záložní uzel pomocí CRG zařízení bez přerušení převzít roli primárního uzlu. Server či servery, které slouží jako záložní, jsou definovány v doméně obnovy. Záložní uzly mohou být ve stejném nebo jiném fyzickém umístění jako primární uzel. Jestliže dojde k výpadku na serveru definovaném jako primární uzel domény obnovy a je vyvoláno přepnutí nebo přepnutí při selhání, stane se uzel určený v doméně obnovy jako záložní primárním přístupovým bodem k prostředku a stane se vlastníkem provozní kopie nezávislého ASP.

Můžete tak získat ochranu před jediným bodem selhání asociovaným s přepínatelnými prostředky.

Tabulka 12. Porovnání technologií pro odolnost dat, které lze použít s klastry. Zde se dozvíte o charakteristických vlastnotech jednotlivých technologií pro odolnost dat, takže budete moci určit nejlepší řešení pro váš klastr.

Faktor	Replikace	Přepínatelné disky	Zrcadlení mezi stanovišti
Přizpůsobivost	Desítky serverů	2 servery	4 servery
Jediný bod selhání	Žádný	Diskový podsystém	Žádný
Náklady	• Je vyžadována dodatečná kapacita disků. • Replikační software podnikatelského partnera. • Duplicitní disk	• Přepínatelná I/O rozšiřující věž nebo procesor IOP • Volba 41	• Další disk pro zrcadlovou kopii nezávislého ASP. • Volitelně přepínatelná I/O rozšiřující jednotka • Volba 41
Výkon	Režie replikace	Malý vliv	Režie geografického zrcadlení
Provoz v reálném čase	Žurnálované objekty	Objekty obsažené v nezávislém ASP	Objekty obsažené v nezávislém ASP
Geografický rozptyl	Může být omezen požadavky na výkon	Omezena je vzdálenost připojení, protože servery a rozšiřující jednotky musí být připojeny ke smyčce HSL OptiConnect (maximálně 250 metrů)	Může být omezen požadavky na výkon (Systém nevyvolává žádná omezení. Doba odezvy a propustnost vybraných komunikačních linek však mohou určovat jistá praktická omezení.)
Ochrana pro obnovu po zhroucení	Ano	Ne	Ano
Souběžné zálohování	Ano	Ne	Ne
Nastavení	• Prostředí replikace. • Určení, co má být replikováno.	• Prostředí nezávislých ASP. • Naplnění nezávislého ASP	• Prostředí nezávislých ASP (včetně nastavení geografického zrcadlení) • Naplnění nezávislého ASP

Související pojmy

"Plánování pro logickou replikaci"
Pomocí logické replikace je udržováno více kopií dat. Data jsou replikována nebo kopírována z primárního uzlu v klastru do záložních uzlů označených v doméně obnovy. Když dojde v primárním uzlu k výpadku, data zůstanou k dispozici, neboť označený záložní uzel převezme roli primárního bodu přístupu.

"Plánování pro přepínatelná nezávislá ASP a zrcadlení mezi stanovišti (XSM)" na stránce 88
Jedna kopie dat je udržována na přepínatelném hardwaru, kterým je buď rozšiřující jednotka (věž) nebo procesor IOP v prostředí logických částí.

Plánování pro logickou replikaci:

Pomocí logické replikace je udržováno více kopií dat. Data jsou replikována nebo kopírována z primárního uzlu v klastru do záložních uzlů označených v doméně obnovy. Když dojde v primárním uzlu k výpadku, data zůstanou k dispozici, neboť označený záložní uzel převezme roli primárního bodu přístupu.

Replikace vytváří kopii něčeho v reálném čase. Jedná se o proces kopírování objektů z jednoho uzlu v klastru do jednoho nebo více jiných uzlů v klastru. Replikace vytváří a uchovává identické objekty v systémech. Provedete-li v klastrovém uzlu změnu objektu, je tato změna replikována do dalších uzlů v klastru.

Musíte se rozhodnout pro určitou softwarovou technologii, kterou použijete pro logickou replikaci. Pro logickou replikaci v klastru jsou k dispozici následující řešení:

- **Produkty obchodních partnerů IBM**

Software pro replikaci dat od uznávaných obchodních partnerů IBM vám umožní replikovat objekty přes více uzlů. Další informace najdete v tématu “Obchodní partneři IBM pro klastrové aplikační programové prostředky a dostupné klastrovací produkty” na stránce 79.

- **Uživatelem napsané replikační aplikace**

  Správce žurnálu IBM poskytuje prostředky, pomocí kterých můžete zaznamenat aktivity objektů v systému. Můžete využít správy žurnálu a napsat aplikaci, která vám umožní provádět logickou replikaci. Podrobné informace o fungování správy žurnálu najdete v tématu Správa žurnálu iSeries.

  Související pojmy

  Správa žurnálů

Určení, které systémy by měly používat logickou replikaci:

Když určujete, které systémy by měly používat logickou replikaci, musíte zvážit několik důležitých věcí.

Jsou to tyto:

- Kapacita výkonu
- Disková kapacita
- Kritická data
- Zamezení katastrofy

Jestliže váš systém přepne při selhání, potřebujete vědět, jaká data a aplikace spouštíte v primárním a záložním systému. Chcete umístit kritická data do systému, který má větší schopnost vypořádat se s pracovní zátěží v případě, že systém přepne při selhání. Nechcete spotřebovat místo na disku. Vyčerpá-li váš primární systém místo a přepne při selhání, je vysoce pravděpodobné, že váš záložní systém rovněž přepne při selhání kvůli nedostatku prostoru na disku. Pokud chcete zajistit, že vaše datové centrum nebude úplně zničeno v případě přírodní katastrofy, jako je povodeň, tornádo nebo hurikán, měli byste umístit replikovaný systém do vzdáleného místa.

Plánování pro přepínatelná nezávislá ASP a zrcadlení mezi stanovišti (XSM):

Jedna kopie dat je udržována na přepínatelném hardwaru, kterým je buď rozšiřující jednotka (věž) nebo procesor IOP v prostředí logických částí.

Když dojde v primárním uzlu k výpadku, přístup k datům na přepínatelném hardwaru se přepne na označený záložní uzel. Kromě toho mohou být nezávislá ASP použita v prostředí XSM (cross-site mirroring). To umožní, aby byla zrcadlená kopie nezávislého ASP udržována v systému, který je (volitelně) geograficky vzdálený od serveru původce kvůli účelům dostupnosti či ochrany.

Chcete-li využít výhody přepínatelných prostředků uložených na přepínatelných nezávislých ASP nebo geografické zrcadlení (XSM), je vyžadováno pečlivé plánování.

Související pojmy

Plánování nezávislých ASP

Zabezpečení klastru

Zvažte některé otázky zabezpečení, které je třeba vzít v úvahu při plánování použití klastrování v systému.

Povolení přidání uzlu do klastru

Chcete-li přidat uzel do klastru, musíte nejdříve nastavit hodnotu atributu sítě ALWADDCLU (Povolit přidání do klastru).

Na libovolném serveru, který chcete nastavit jako klastrový uzel, použijte příkaz CHGNETA (Změna atributů sítě). Příkaz CHGNETA (Změna atributů sítě) mění síťové atributy systému. Síťový atribut ALWADDCLU určuje, zda uzel dovolí, aby ho jiný systém přidal jako uzel do klastru.

Poznámka: Ke změně atributu sítě ALWADDCLU musíte mít oprávnění *IOSYSCFG.

Můžete vybrat některou z těchto hodnot:

***SAME**
Hodnota se nezmění. Systém je dodáván s hodnotou *NONE.

***NONE**
Žádný jiný systém nesmí přidat tento systém jako uzel do klastru.

***ANY** Jakýkoliv jiný systém může přidat tento systém jako uzel do klastru.

***RQSAUT**
Jakýkoliv jiný systém může přidat tento systém jako uzel do klastru, pokud byl požadavek na přidání uzlu do klastru autentizován.

Atribut sítě ALWADDCLU se kontroluje, aby se zjistilo, zda se přidávaný uzel může stát součástí klastru a zda je nutné ověřit požadavek na přidání uzlu do klastru pomocí digitálních certifikátů X.509. **Digitální certifikát** je forma osobní identifikace, kterou lze elektronicky ověřovat. Pokud je požadováno ověřování, musí žádající uzel a přidávaný uzel mít v systémech nainstalovány tyto programy:

- i5/OS Volba 34 (Digital Certificate Manager)
- Cryptographic Access Provider

Pokud je vybrána hodnota *RQSAUT, musí být řádně nastaven seznam důvěryhodných vydavatelů certifikátů i5/OS pro serverovou aplikaci pro zabezpečení klastru. Identifikátor serverové aplikace je QIBM_QCST_CLUSTER_SECURITY. Minimálně musíte přidat vydavatele certifikátů pro uzly, kterým povolíte připojení ke klastru.

Související pojmy

Správa digitálních certifikátů

“Běžné problémy s klastry” na stránce 132
Zde jsou popsány některé z nejčastějších problémů, ke kterým dochází v klastru, a způsoby, jak se těchto problémů vyvarovat a jak provést nápravu chyb.

Související odkazy

Příkaz CHGNETA (Změna atributů sítě)

Distribuce celoklastrových informací

Zde se dozvíte o bezpečnostních aspektech použití a správy celoklastrových informací.

Rozhraní QcstDistributeInformation (Distribute Information) API lze používat k rozesílání zpráv z jednoho uzlu skupiny klastrových prostředků do dalších uzlů dané domény obnovy. To může být výhodné při zpracování ukončovacího programu. Je však třeba poznamenat, že tyto informace nejsou šifrovány. Nepoužíváte-li zabezpečenou síť, neměl by tento mechanismus sloužit k posílání informací podléhajících zabezpečení.

Netrvalá data lze sdílet a replikovat mezi klastrovými uzly pomocí rozhraní API klastrované transformační tabulky. Data jsou uložena v netrvalé paměti. To znamená, že data jsou zachována pouze po dobu, kdy je klastrový uzel součástí klastrované transformační tabulky. Tato rozhraní API lze používat pouze z klastrového uzlu, který je definován v doméně klastrované transformační tabulky. Tento klastrový uzel musí být v klastru aktivní.

Podobně ani další informace distribuované pomocí posílání klastrových zpráv nejsou zabezpečeny. Zahrnuto je i posílání klastrových zpráv nízké úrovně. V zásadě platí, že když jsou provedeny změny dat ukončovacího programu, zpráva obsahující tato data není šifrována.

Související odkazy

Rozhraní QcstDistributeInformation (Distribuce informací) API

Rozhraní ClusteredHashTable (Clustered Hash Table) API

Správa uživatelských profilů ve všech uzlech

Pro udržování uživatelských profilů ve všech uzlech v klastru můžete použít dva mechanismy.

Jeden mechanismus je vytvořit administrativní doménu klastru ke sledování sdílených prostředků ve všech uzlech v klastru. Administrativní doména klastru dokáže kromě uživatelských profilů monitorovat několik typů prostředků, čímž zajišťuje snadnou správu prostředků, které se sdílejí mezi uzly. Podrobné informace o těchto prostředcích najdete v tématu Monitorované prostředky. Když se aktualizují uživatelské profily, změny se automaticky přenesou na jiné uzly, jestliže je administrativní doména klastru aktivní. Jestliže administrativní doména klastru není aktivní, změny se rozšíří, jakmile bude administrativní doména klastru aktivována.

Poznámka: Jestliže plánujete sdílení uživatelských profilů, které používají synchronizaci hesel v rámci klastru, musíte nastavit systémovou hodnotu Retain Server Security (QRETSVRSEC) na 1.

V druhém mechanismu mohou administrátoři použít také Centrální správu v produktu iSeries Navigator k provádění funkcí na více systémech a skupinách systémů. Tato podpora zahrnuje některé běžné úkoly správy uživatelů, které operátoři potřebují provádět u více systémů v klastru. Centrální správa umožňuje provádět činnosti s uživatelskými profily pro skupiny systémů. Administrátor může určit, že v cílových systémech má být při vytvoření uživatelského profilu spuštěn příkaz pro následné přenesení zabezpečovacích informací.

Související pojmy

“Struktura úloh a uživatelské fronty” na stránce 112
Když spravujete klastr, musíte znát strukturu a uživatelské fronty.

“Administrativní doména klastru” na stránce 8
Administrativní doména klastru se používá ke správě prostředků, které je nutné udržovat konzistentní ve všech uzlech v klastrovaném prostředí.

Pokyny pro použití klastrů s firewally

Jestliže používáte klastrování v síti, která používá firewally (ochranné bariéry), měli byste si být vědomi určitých omezení a požadavků.

Jestliže používáte klastrování s firewallem, musíte umožnit, aby každý uzel odesílal odchozí zprávy a přijímal příchozí zprávy od jiných uzlů klastru. Ve firewallu musí existovat otvor pro každou adresu klastru na každém uzlu, aby bylo možné komunikovat s každou klastrovou adresou na každém jiném uzlu. IP pakety cestující po síti mohou pocházet z různých typů přenosů. Klastrování používá funkci ping, což je typ ICMP, a používá také UDP a TCP. Když uživatel konfiguruje firewall, může filtrovat přenosy podle typu. K tomu, aby klastrování mohou fungovat, musí firewall umožnit přenosy ICPM UDP a TCP. Odchozí přenosy mohou být odesílány na libovolný port a příchozí přenosy jsou přijímány na portech 5550 a 5551.

Kontrolní seznam pro konfiguraci klastru

Vyplňte kontrolní seznam konfigurace klastru, abyste zajistili, že vaše prostředí bude řádně připraveno, než začnete konfigurovat svůj klastr.

Tabulka 13. Kontrolní seznam konfigurace TCP/IP pro klastry

Požadavky TCP/IP	
__	Pomocí příkazu STRTCP (Spuštění TCP/IP) spusťte TCP/IP v každém uzlu, který zamýšlíte zahrnout do klastru.
__	Nakonfigurujte adresu smyčkového testu TCP (127.0.0.1) a ověřte, že ukazuje stav *Aktivní*. Ověřte to pomocí příkazu WRKTCPSTS (Práce se stavem sítě TCP/IP) na každém uzlu v klastru.
__	Ověřte pomocí příkazu WRKTCPSTS (Práce se stavem sítě TCP/IP) na předmětném uzlu, že IP adresy použité ke klastrování na daný uzel ukazují stav *Aktivní*.

Tabulka 13. Kontrolní seznam konfigurace TCP/IP pro klastry (pokračování)

Požadavky TCP/IP	
__	Ověřte, že INETD je aktivní ve všech uzlech klastru, a to pomocí příkazu STRTCPSVR *INETD nebo pomocí aplikace iSeries Navigator tak, že provedete následující kroky: 1. V prostředí produktu iSeries Navigator rozbalte **Síť**. 2. Rozbalte **Servery**. 3. Rozbalte **TCP/IP**. 4. Klepněte pravým tlačítkem myši na **INETD** a vyberte **Spustit**. Můžete to ověřit tak, že zkontrolujete existenci úlohy QTOGINTD (uživatel QTCP) v seznamu Aktivní úlohy na předmětném uzlu.
__	Ověřte, že uživatelský profil pro INETD, který je uveden v /QIBM/ProdData/OS400/INETD/inetd.config, nemá vyšší než minimální oprávnění. Příkaz STRCLUNOD (Spuštění klastrového uzlu) selže, pokud tento uživatelský profil má více než minimální oprávnění. Je předvoleno, že uživatel QUSER je uveden jako uživatelský profil pro INETD.
__	Ověřte, že každá IP adresa v klastru může směrovat a odesílat UDP datagramy do každé jiné IP adresy v klastru. Použijte příkaz PING a uveďte lokální IP adresu a příkaz TRACEROUTE a uveďte UDP zprávy.
__	Ověřte, že porty 5550 a 5551 nejsou používány jinými aplikacemi. Tyto porty jsou vyhrazeny pro klastrování IBM. Použití portů lze zobrazit pomocí příkazu WRKTCPSTS (Práce se stavem sítě TCP/IP). Po spuštění INETD bude port 5550 otevřen a ve stavu 'Listen' (naslouchání).

Plánujete-li použití přepínatelných zařízení v klastru, musí být splněny tyto požadavky:

Tabulka 14. Kontrolní seznam konfigurace odolných zařízení pro klastry

Požadavky na odolná zařízení	
__	Ověřte, že ve všech klastrových uzlech, které budou v doméně zařízení, je nainstalována Volba 41 (HA Switchable Resources) a že má platné licenční klíče. Pamatujte, že tuto volbu vyžaduje jakékoli použití rozhraní produktu iSeries Navigator pro správu klastru.
__	Aby byly přístupné funkce správy disků v prostředí produktu iSeries Navigator, nakonfigurujte server servisních nástrojů (STS) s přístupem DST a uživatelskými profily. Podrobné informace najdete v tématu Nastavení komunikace.
__	Pokud mezi logickými částmi systému přepínáte odolná zařízení a pokud ke správě logických částí používáte jiný nástroj než konzoli HMC, aktivujte na těchto logických částech Virtuální OptiConnect. Provádí se to při přihlášení DST (Dedicated service tools). Podrobné informace najdete v tématu Virtuální OptiConnect. Pokud ke správě logických částí používáte konzoli HMC, změňte vlastnosti profilu části na kartě OptiConnect tak, abyste povolili používání Virtuálního OptiConnectu u každé části v přepínatelné konfiguraci. Změny se projeví, když aktivujete profil části.
__	Pokud je mezi dvěma systémy přepínána věž ve smyčce HSL OptiConnect a jeden ze systémů má logické části, povolte pro tyto logické části HSL OptiConnect. Pokud ke správě logických částí používáte jiný nástroj než konzoli HMC, proveďte tuto činnost při přihlášení DST (Dedicated service tools). Pokud ke správě logických částí používáte konzoli HMC, změňte vlastnosti profilu logické části na kartě OptiConnect tak, abyste povolili používání OptiConnectu u každé logické části v přepínatelné konfiguraci. Změny se projeví, když aktivujete profil logické části.
__	Pokud mezi logickými částmi přepínáte odolná zařízení a pokud ke správě logických částí používáte jiný nástroj než konzoli HMC, musíte nakonfigurovat sběrnici tak, aby byla sdílena logickými částmi, nebo musíte nakonfigurovat společnou oblast I/O. Sběrnice musí být nakonfigurována na jedné logické části jako ″own bus shared (vlastník sdílené sběrnice)″, a na ostatních částech, které se budou účastnit přepínání zařízení, musí být nakonfigurována jako ″use bus shared (uživatel sdílené sběrnice)″. Pokud ke správě logických částí používáte konzoli HMC, musíte nakonfigurovat I/O Pool, který bude obsahovat procesor I/O, adaptér I/O a všechny připojené prostředky, čímž vytvoříte nezávislé ASP, které již bude možné přepínat mezi logickými částmi. Každá část musí mít přístup do společné oblasti I/O. Další podrobnosti najdete v tématu Jak učinit hardware přepínatelným.Podrobné informace o požadavcích na fyzické plánování pro přepínatelná zařízení najdete v tématu Požadavky na fyzické plánování.

Tabulka 14. Kontrolní seznam konfigurace odolných zařízení pro klastry (pokračování)

Požadavky na odolná zařízení	
__	Chcete-li přepínat věž ve smyčce HSL mezi dvěma různými systémy, musí být věž nakonfigurována jako přepínatelná. Podrobné informace najdete v tématu Jak učinit hardware přepínatelným.
__	Přidáte-li věž do existující smyčky HSL, spusťte znovu servery v této stejné smyčce.
__	Maximální přenosová jednotka (MTU) pro komunikační cesty musí být větší než optimalizační parametr komunikace klastru nazývaný velikost fragmentu zpráv. MTU pro určitou IP adresu klastru lze ověřit v předmětném uzlu pomocí příkazu WRKTCPSTS (Práce se stavem sítě TCP/IP).MTU je nutné ověřit také na každém kroku podél celé komunikační cesty. Pravděpodobně bude snazší zmenšit parametr velikost fragmentu zpráv po vytvoření klastru než zvětšovat MTU pro komunikační cestu. Další informace o velikosti fragmentu zpráv najdete v tématu Optimalizační parametry komunikace klastru.Můžete použít API QcstRetrieveCRSInfo (Načtení informací o Službách klastrových prostředků) a prohlédnout si aktuální nastavení parametrů ladění. Pomocí API QcstChgClusterResourceServices (Změna Služeb klastrových prostředků) můžete nastavení změnit.

Tabulka 15. Kontrolní seznam konfigurace zabezpečení pro klastry

Požadavky na zabezpečení	
__	Při pokusu o spuštění vzdáleného uzlu musí být na cílovém uzlu správně nastaven atribut sítě ALWADDCLU (Povolit přidání do klastru). V závislosti na prostředí by měl být nastaven na hodnotu *ANY nebo *RQSAUT. Pokud je nastaven na *RQSAUT, musí být nainstalována Volba 34 systému i5/OS (Digital Certificate Manager) a produkt Cryptographic Access Provider (AC2 nebo AC3). Podrobné informace o nastavení atributu sítě ALWADDCLU najdete v tématu Povolení přidání uzlu do klastru.
__	Umožněte stav uživatelského profilu pro INETD uvedený v souboru /QIBM/ProdData/OS400/INETD/inetd.config. Nesmí mít zvláštní oprávnění *SECADM nebo *ALLOBJ. Je předvoleno, že uživatel QUSER je uveden jako uživatelský profil pro INETD.
__	Ověřte, že uživatelský profil vyvolávající rozhraní API pro služby klastrových prostředků existuje ve všech klastrových uzlech a má oprávnění *IOSYSCFG.
__	Ověřte, že uživatelský profil, který bude spouštět ukončovací program skupiny klastrových prostředků (CRG) existuje ve všech uzlech domény obnovy.

Tabulka 16. Kontrolní seznam konfigurace úloh pro klastry

Pokyny k úlohám	
__	Úlohy mohou být zadávány pomocí rozhraní API pro služby klastrových prostředků za účelem zpracování požadavků. Úlohy mohou být spuštěny buď pod uživatelským profilem, který spustí ukončovací program určený při vytvoření skupiny klastrových prostředků, nebo pod uživatelským profilem, který požadoval dané rozhraní API (pouze při logickém zapínání zařízení ve skupinách klastrových prostředků odolných zařízení). Uživatel musí zajistit, aby podsystém, který obsluhuje frontu úloh asociovanou s uživatelským profilem, byl nakonfigurován jako *NOMAX pro počet úloh, které může spustit z dané fronty úloh.
__	Úlohy budou předávány do fronty úloh určené v popisu úlohy, který je získán z uživatelského profilu definovaného pro skupinu klastrových prostředků (CRG). Předvolený popis úlohy způsobí, že úlohy budou posílány do fronty úloh QBATCH. Protože se tato fronta úloh používá pro mnoho uživatelských úloh, nemusí být úloha ukončovacího programu spuštěna včas. Uživatelé by měli uvážit jedinečný popis úlohy s jedinečnou uživatelskou frontou.
__	Když jsou úlohy ukončovacích programů spuštěny, použijí údaje o směrování z popisu úlohy k tomu, aby zvolily, které hlavní ASP a prováděcí atributy budou používat. Předvolené hodnoty způsobí, že úlohy budou spuštěny ve společné oblasti s dalšími dávkovými úlohami s prioritou spuštění 50. Žádná z těchto hodnot nemůže vést k požadované výkonnosti úloh ukončovacích programů. Podsystém, který aktivuje úlohy ukončovacích programů (tentýž podsystém, který používá jedinečnou frontu úloh), by měl přiřadit úlohy ukončovacích programů k ASP, které není používáno dalšími úlohami aktivovanými tímto podsystémem nebo jinými podsystémy. Kromě toho by měla být úlohám ukončovacích programů přiřazena priorita spuštění 15, aby byly spuštěny dříve než téměř všechny ostatní uživatelské úlohy.
__	Systémová hodnota QMLTTHDACN musí být nastavena na 1 nebo 2.

Pro konfiguraci a správu klastru existuje několik softwarových řešení. Jedno z těchto řešení je správa klastru pomocí produktu iSeries Navigator. Rozhodnete-li se používat iSeries Navigator, musí být splněny tyto požadavky:

Tabulka 17. Kontrolní seznam konfigurace iSeries Navigator pro klastry

Pokyny pro správu klastru iSeries Navigator	
__	Ve všech klastrových uzlech, které budou v doméně zařízení, musí být nainstalována Volba 41 (i5/OS - HA Switchable Resources) a musí mít platné licenční klíče.
__	Ověřte, že všechny hostitelské servery jsou spuštěny pomocí příkazu STRHOSTSVR (Spuštění hostitelského serveru): STRHOSTSVR SERVER(*ALL)
__	Ověřte, že server Centrální správy je spuštěn pomocí příkazu STRTCPSVR (Spuštění serveru TCP/IP): STRTCPSVR SERVER(*MGTC)

Související pojmy

"Správa klastrů pomocí produktu iSeries Navigator" na stránce 72
IBM nabízí rozhraní pro správu klastru, které je dostupné pomocí produktu iSeries Navigator a přístupné pomocí Volby 41 (i5/OS - HA Switchable Resources).

"Server INETD"
Aby bylo možné přidat nebo spustit uzel a zpracovávat slučování částí, musí být spuštěn server INEGD (Internet daemon).

"Laditelné parametry klastrové komunikace" na stránce 94
Rozhraní QcstChgClusterResourceServices (Change Cluster Resource) API umožňuje ladit některé služby klastrové topologie, výkon klastrové komunikace a parametry konfigurace tak, aby lépe vyhovovaly mnoha jedinečným aplikacím a prostředím výstavby sítí, ve kterých dochází ke klastrování. Toto API je k dispozici pro každý klastr, který je spušten v klastrové verzi 2 nebo vyšší.

Související odkazy

"Kontrolní seznam domény pro administraci klastrů" na stránce 96
Tato část pojednává o všech požadavcích, které musejí být splněny před vytvořením domény pro administraci klastrů.

Server INETD

Aby bylo možné přidat nebo spustit uzel a zpracovávat slučování částí, musí být spuštěn server INEGD (Internet daemon).

Server INETD by měl být v klastru vždy spuštěn.

Pomocí produktu iSeries Navigator

K tomu potřebujete mít nainstalovánu Volbu 41 (HA Switchable Resources) s licencí.

Při spouštění serveru INETD postupujte takto:

1. V prostředí produktu iSeries Navigator rozbalte **Síť**.
2. Rozbalte **Servery**.
3. Rozbalte **TCP/IP**.
4. Klepněte pravým tlačítkem myši na **INETD** a vyberte **Spustit**.

Pomocí CL příkazů a API

Server INETD lze spustit také příkazem STRTCPSVR (Spuštění serveru TCP/IP) a zadáním parametru *INETD. Bude-li server INETD spuštěn, bude v seznamu Aktivní úlohy na předmětném uzlu existovat úloha QTOGINTD (uživatel QTCP).

Související pojmy

"Běžné problémy s klastry" na stránce 132
Zde jsou popsány některé z nejčastějších problémů, ke kterým dochází v klastru, a způsoby, jak se těchto problémů vyvarovat a jak provést nápravu chyb.

Související odkazy

Příkaz STRTCPSVR (Spuštění serveru TCP/IP)

Laditelné parametry klastrové komunikace

Rozhraní QcstChgClusterResourceServices (Change Cluster Resource) API umožňuje ladit některé služby klastrové topologie, výkon klastrové komunikace a parametry konfigurace tak, aby lépe vyhovovaly mnoha jedinečným aplikacím a prostředím výstavby sítí, ve kterých dochází ke klastrování. Toto API je k dispozici pro každý klastr, který je spušten v klastrové verzi 2 nebo vyšší.

Příkaz Příkaz CHGCLUCFG (Změna ladění klastrové konfigurace) poskytuje základní úroveň ladění, zatímco rozhraní QcstChgClusterResourceServices API poskytuje jak základní, tak pokročilé úrovně ladění.

Rozhraní QcstChgClusterResourceServices API a příkaz CHGCLUCFG mohou být použity pro ladění klastrového výkonu a konfigurace. API a příkaz poskytují základní úroveň podpory ladění, kdy bude klastr upraven na předdefinovanou sadu hodnot určených pro dlouhé, krátké a běžné hodnoty prodlevy a časového limitu posílání zpráv. Požadujete-li pokročilou úroveň ladění, obvykle s podporou zástupce IBM, mohou být jednotlivé parametry naladěny pomocí API na jiný než předdefinovaný rozsah hodnot. Nesprávné změny jednotlivých parametrů mohou snadno vést ke snížení klastrového výkonu.

Kdy a jak ladit klastrové parametry?

Příkaz CHGCLUCFG a rozhraní QcstChgClusterResourceServices API poskytují rychlý způsob nastavení výkonu klastru a konfiguračních parametrů, aniž byste museli rozumět podrobnostem. Tato základní úroveň ladění ovlivňuje zejména citlivost pulsu a hodnoty časových prodlev klastrových zpráv. Platné hodnoty pro základní úroveň podpory ladění jsou:

1 (Vysoké hodnoty prodlev/Méně častý puls)

2 (Předvolené hodnoty)
Pro výkon klastrové komunikace a konfigurační parametry jsou použity předvolené hodnoty. Toto nastavení může být použito pro návrat všech parametrů na původní předvolené hodnoty.

3 (Nízké hodnoty prodlev / Častější puls)
Úpravy se týkají klastrové komunikace kvůli snížení intervalu pulsu a snížení hodnot časových limitů různých zpráv. Při častějších pulsech a kratších časových limitech bude klastr rychleji odpovídat (bude více citlivý) na chyby v komunikaci.

Výsledné příklady časů odpovědí jsou zobrazeny v níže uvedené tabulce pro selhání pulsu vedoucího k rozdělení uzlu:

	1 (Méně citlivý)			**2 (Předvolba)**			**3 (Více citlivý)**		
	Problém s detekcí pulsu	Analýza	Celkem	Problém s detekcí pulsu	Analýza	Celkem	Problém s detekcí pulsu	Analýza	Celkem
Jedna podsíť	00:24	01:02	01:26	00:12	00:30	00:42	00:04	00:14	00:18
Několik podsítí	00:24	08:30	08:54	00:12	04:14	04:26	00:04	02:02	02:06

Poznámka: Časy jsou ve formátu minuty:vteřiny.

V závislosti na typické zátěži sítě a použitých specifických fyzických médiích může klastrový administrátor upravit úrovně citlivosti pulsu a časové limity pro zasílání zpráv. Například při vysokorychlostním, vysoce spolehlivém přenosu, jako je OptiConnect se všemi systémy v klastru na společné sběrnici OptiConnect, možná budete chtít vytvořit více citlivé prostředí, abyste zajistili rychlou detekci vedoucí k rychlejšímu přepnutí při selhání. Je vybrána volba 3.

Pokud pracujete na silně zatížené sběrnici typu 10Mbs Ethernet a předvolené nastavení vedlo k občasným rozdělením právě kvůli maximálnímu zatížení sítě, mohli byste vybrat volbu 1 a snížit citlivost klastrování při maximálním zatížení.

API QcstChgClusterResourceServices (Změna Služby klastrových prostředků) umožňuje také ladit specifické individuální parametry, kdy požadavky síťového prostředí představují jedinečnou situaci. Vezměme opět jako příklad klastr se všemi uzly společnými na sběrnici OptiConnect. Výkon klastrových zpráv může být výrazně zvýšen nastavením parametru Message Fragment Size (Velikost fragmentu zpráv) na maximum 32.500 bajtů, aby lépe vyhovoval velikosti OptiConnect MTU (maximální přenosová jednotka) než předvolených 1,464 bajtů. Tím se sníží nároky na fragmentaci a opětovné shromažďování velkých zpráv. Výsledný efekt závisí samozřejmě na klastrových aplikacích a použití klastrového posílání zpráv následkem těchto aplikací. Ostatní parametry jsou definovány v dokumentaci API a mohou být použity pro ladění buď výkonu klastrového posílání zpráv, nebo pro úpravu citlivosti klastru při dělení.

Související pojmy

"Ladění klastrového výkonu" na stránce 110
Jelikož ve vašem komunikačním prostředí existují potenciálně značné rozdíly, máte možnost upravit proměnné, které ovlivňují klastrovou komunikaci tak, aby lépe vyhovovaly vašemu prostředí.

Kontrolní seznam pro dekonfiguraci klastru

Jestliže chcete vymazat klastr nebo CRG, musíte systematicky odstranit různé komponenty klastru, abyste zajistili kompletní dekonfiguraci klastru.

Tabulka 18. Kontrolní seznam dekonfigurace oblastí ASP pro klastry

Nezávislá ASP	
__	Jestliže plánujete odstranit podmnožinu skupiny ASP nebo odstranit poslední nezávislou ASP v přepínatelných zařízeních, musíte nejprve ukončit CRG. Použijte příkaz ENDCRG (Ukončení skupiny klastrových prostředků).
__	Jestliže chcete vymazat ASP, které je součástí klastru, velmi doporučujeme, abyste nejprve odstranili konfigurační objekt ASP z přepínatelného zařízení známého rovněž jako skupina zařízení klastrových prostředků (CRG). Chcete-li vymazat konfigurační objekt ASP z přepínatelného zařízení, postupujte takto: Chcete-li vymazat ASP z přepínatelného zařízení, postupujte takto: 1. V produktu iSeries Navigator rozbalte **Centrální správu → Klastry**. 2. Rozbalte *jméno klastru obsahující přepínatelné zařízení* → **Přepínatelná zařízení**. 3. Klepněte na jméno přepínatelného zařízení. 4. V pravém podokně produktu iSeries Navigator klepněte pravým tlačítkem myši na ASP a vyberte **Odstranit**. K odstranění konfiguračního objektu ASP ze skupiny CRG můžete také použít příkaz RMVCRGDEVE (Odstranění záznamu zařízení CRG).
__	Jakmile odstraníte konfigurační objekt ASP z přepínatelného zařízení klastru, můžete vymazat oblast ASP.
__	Vymažte popis zařízení pro nezávislé ASP tím, že provedete tyto úlohy: 1. Na rozhraní příkazového řádku napište příkaz WRKDEVD DEVD(*ASP) a stiskněte klávesu Enter. 2. Stránkujte dolů, až se zobrazí popis zařízení pro ASP, které chcete vymazat. 3. Vyberte volbu 4 (Vymazat) vedle jména zařízení a stiskněte klávesu Enter.

Tabulka 19. Kontrolní seznam dekonfigurace skupiny klastrových prostředků pro klastry

Požadavek na skupinu klastrových prostředků	
__	Vymažte skupinu klastrových prostředků tím, že provedete jeden z následujících úkolů: 1. Jestliže v uzlu není aktivní klastrování, napište příkaz DLTCRG CRG(CRGNAME) na rozhraní příkazového řádku. CRGNAME je jméno skupiny CRG, kterou chcete vymazat. Stiskněte klávesu Enter. 2. Jestliže v uzlu je aktivní klastrování, napište příkaz DLTCRGCLU CLUSTER(CLUSTERNAME) CRG(CRGNAME) na rozhraní příkazového řádku. CLUSTERNAME je jméno klastru. CRGNAME je jméno skupiny CRG, kterou chcete vymazat. Stiskněte klávesu Enter.

Plánování administrativní domény klastru

Administrativní doména klastru vyžaduje plánování pro správu prostředků, které jsou sdíleny mezi uzly v rámci administrativní domény klastru.

Když vytvoříte administrativní doménu klastru, skupina peer CRG se vytvoří automaticky, aby reprezentovala tuto doménu. Administrativní doménu klastru můžete spravovat pomocí rozhraní API, CL příkazů a produktem iSeries Navigator.

Administrátor klastru může vytvořit doménu pro administraci klastru a pak přidat monitorované prostředky, které jsou sdíleny mezi uzly. Klastr i5/OS poskytuje seznam systémových prostředků, které lze sdílet mezi uzly v rámci administrativní domény klastru - jsou představované *záznamy monitorovaných prostředků (MRE)*. Kompletní seznam systémových prostředků, které lze monitorovat, najdete v tématu Monitorované prostředky.

Když navrhujete administrativní doménu klastru, měli byste si odpovědět na tyto otázky:

Jaké prostředky budou sdíleny?
Budete muset zjistit, jaké systémové prostředky bude nutné sdílet. Můžete si vybrat atributy pro každý z těchto prostředků, abyste mohli přizpůsobit, co bude sdíleno mezi uzly. Aplikace, které jsou spouštěny na více uzlech, budou možná potřebovat specifické proměnné prostředí, aby běžely správně. Také data, která se rozkládají na více uzlech, budou možná potřebovat určité uživatelské profily, aby byla přístupná. Měli byste znát provozní požadavky svých aplikací a dat před tím, než určíte, jaké prostředky budou sdíleny.

Jaké uzly budou zahrnuty do administrativní domény klastru?
Měli byste určit, jaké uzly v klastru musejí být spravovány administrativní doménou klastru. Uzly nesmějí být ve více administrativních doménách klastru. Dejme tomu, že máte čtyři uzly v klastru (uzel A, uzel B, uzel C a uzel D). Uzly A a B mohou být v jedné administrativní doméně klastru a uzly C a D mohou být jiné. Avšak uzel B a uzel C nesmějí být v nějaké další administrativní doméně klastru.

Jaká bude konvence pojmenování pro administrativní domény klastru?
Podle složitosti a velikosti vašeho klastrovaného prostředí je možné, že budete chtít vytvořit standardní konvenci pojmenování pro peer skupiny CRG a administrativní domény klastru. Jelikož peer CRG se vytváří k tomu, aby reprezentovala administativní doménu klastru, budete chtít rozpoznat peer CRG od těch, tkeré monitorují prostředky ve vašem klastru. Například skupiny peer CRG, které představují administrativní domény klastru, lze pojmenovat *ADMDMN1*, *ADMDMN2* atd., kdežto jiné peer CRG lze pojmenovat *PEER1*. Můžete také použít rozhraní QcstListClusterResourceGroupIn (List Cluster Resource Group Information) API k určení toho, zda peer CRG se používá jako administrativní doména klastru.

Kontrolní seznam domény pro administraci klastrů

Tato část pojednává o všech požadavcích, které musejí být splněny před vytvořením domény pro administraci klastrů.

Tabulka 20. Kontrolní seznam domény pro administraci klastrů

Požadavky na domény pro administraci klastrů	
__	Ověřte, že klastr byl konfigurován. Viz Kontrolní seznam konfigurace klastrů.
__	Jestliže plánujete monitorování uživatelských profilů, které používají synchronizaci hesel v rámci klastru, musíte nastavit systémovou hodnotu Retain Server Security (QRETSVRSEC) na 1.
__	Chcete-li přidat prostředky do domény pro administraci klastrů, všechny uzly v doméně administrace klastrů musejí být aktivní, musejí se účastnit skupiny a nesmějí být rozděleny na části.

Konfigurace klastrů

Pochopte postup vytváření klastru.

Společnost IBM a obchodní partneři společnosti IBM vyvíjející klastrový middleware se spojili, aby pro správu klastrů poskytli nejvyspělejší funkce služeb klastrových prostředků spolu s grafickým uživatelským rozhraním (GUI). i5/OS služby klastrovaných prostředků systému poskytují sadu integrovaných služeb, které udržují klastrovou topologii, provádějí pulzaci a umožňují vytváření a administraci klastrové konfigurace a skupin klastrových prostředků. Služby klastrových prostředků poskytují také funkce spolehlivých zpráv, které sledují každý klastrový uzel a zajišťují, aby všechny uzly měly konzistentní informace o stavu klastrových prostředků. Služby klastrových prostředků dále poskytují sadu CL příkazů, rozhraní API a systémových prostředků, které mohou dodavatelé aplikací iSeries nebo zákazníci použít ke zlepšení dostupnosti aplikací. K funkcím služeb klastrových prostředků lze přistupovat také pomocí řešení využívajících grafické uživatelské rozhraní - pomocí správy klastru v prostředí produktu iSeries Navigator nebo pomocí produktů typu klastrový middleware dodaných obchodními partnery IBM.

Začínáme

Při konfiguraci klastru postupujte podle těchto kroků:

1. **Vyberte softwarové řešení.**

   Udělejte si úplný obraz o možnostech konfigurace a správy klastrů - seznamte se s informacemi uvedenými v tématu “Řešení pro konfiguraci a správu klastrů” na stránce 72.
2. **Splňte požadavky na hardware, software a komunikace.**

   Téma Plánování klastrů uvádí požadavky spojené s klastry.
3. **Nastavte pro klastry prostředí sítě a serverů.**

   Použijte “Kontrolní seznam pro konfiguraci klastru” na stránce 90 k ověření, zda jste připraveni ke konfiguraci klastrů ve vašem prostředí.
4. **Nakonfigurujte klastr.**

   Související pojmy

   “Kam zavolat, pokud potřebujete podporu” na stránce 147
   Toto téma si přečtěte v případě, že potřebujete kontaktovat společnost IBM kvůli otázkám ohledně klastrů.

Vytvoření klastru

Chcete-li vytvořit a nakonfigurovat klastr, musíte do klastru zahrnout alespoň jeden uzel a musíte mít přístup nejméně k jednomu z uzlů, které budou v klastru.

| Určíte-li pouze jeden uzel, musí to být server, ke kterému momentálně přistupujete. Úplný seznam požadavků pro
| vytváření klastrů najdete v tématu “Kontrolní seznam pro konfiguraci klastru” na stránce 90.

Budete-li v klastru používat přepínatelná zařízení, jsou na tento klastr kladeny další požadavky v porovnání s klastrem, který nepoužívá přepínatelná zařízení. Při nastavování prostředí zahrnujícího přepínatelná zařízení je nutné vyvarovat se konfliktů v rámci klastru. Podrobné instrukce ke konfiguraci klastru používajícího přepínatelná zařízení najdete v tématu Vytvoření přepínatelného nezávislého ASP.

Pomocí produktu iSeries Navigator

K tomu potřebujete mít nainstalovánu Volbu 41 (HA Switchable Resources) s licencí.

Při správě klastru pomocí produktu iSeries Navigator můžete použít průvodce, který vás povede jednotlivými kroky vytvoření a spuštění jednoduchého klastru tvořeného jedním nebo více klastrovými uzly. Po vytvoření klastru obsahujícího jeden nebo dva uzly můžete přidat další uzly. Klastr vytvořený a spravovaný v prostředí produktu iSeries Navigator může obsahovat maximálně čtyři uzly. Tento průvodce vás povede kroky určení serverů zahrnutých do klastru a vytvoření skupin klastrových prostředků. Při vytváření jednoduchého klastru musí být server, na kterém klastr vytváříte, jedním z uzlů.

Chcete-li vytvořit jednoduchý klastr pomocí průvodce vytvořením nového klastru v prostředí produktu iSeries Navigator, postupujte takto:

1. V produktu iSeries Navigator rozbalte **Centrální správu**.

2. Klepněte pravým tlačítkem myši na **Klastry** a vyberte **Nový klastr**.
3. Při vytvoření nového klastru postupujte podle pokynů průvodce.

Po vytvoření nového klastru proveďte tyto akce:

1. Přidejte všechny uzly, které chcete zahrnout do klastru. Klastr vytvořený a spravovaný v prostředí produktu iSeries Navigator může obsahovat maximálně čtyři uzly.
2. Přidejte požadované uzly do domén zařízení (pro použití v přepínatelných skupinách hardwaru a nezávislých ASP).
3. Vytvořte a spusťte přepínatelné prostředky (přepínatelné zařízení, přepínatelná aplikace a přepínatelná data).

Online nápověda v produktu iSeries Navigator obsahuje podrobné postupy provádění těchto úkolů.

Pomocí CL příkazů a API

K vytvoření klastru můžete použít také CL příkazy nebo rozhraní API:

1. **Vytvořte klastr.**
   Příkaz CRTCLU (Vytvoření klastru)
   Rozhraní QcstCreateCluster (Create Cluster) API
2. **Přidejte uzly do klastru z aktivního klastrového uzlu.**
   Příkaz ADDCLUNODE (Přidání záznamu klastrového uzlu)
   Rozhraní QcstAddClusterNodeEntry (Add Cluster Node Entry) API
3. **Spusťte klastrový uzel.**
   Příkaz STRCLUNOD Spuštění klastrového uzlu)
   Rozhraní QcstStartClusterNode (Start Cluster Node) API
4. **Definujte domény zařízení.** Plánujete-li používat přepínatelná zařízení, musíte zahrnout požadované uzly do domény zařízení.
   Příkaz ADDDEVDMNE (Přidání záznamu domény zařízení)
   Rozhraní QcstAddDeviceDomainEntry (Add Device Domain Entry) API
5. **Vytvořte skupiny klastrových prostředků (CRG).**
   Příkaz CRTCRG (Vytvoření skupiny klastrových prostředků)
   Rozhraní QcstCreateClusterResourceGroup) (Create Cluster Resource Group) API
6. **Spusťte skupiny klastrových prostředků (CRG).**
   Příkaz STRCRG (Spuštění skupiny klastrových prostředků)
   Rozhraní QcstStartClusterResourceGroup (Start Cluster Resource Group) API

Správa klastrů

Toto téma obsahuje informace zaměřené na některé z operací zahrnujících správu klastrů.

Pokud jste si dosud nerozmysleli typ rozhraní, který budete chtít používat ke správě klastrů, přečtěte si nejprve informace v tématu Řešení pro správu klastrů.

Ke změnám, které je možné provádět s klastrem poté, co byl nakonfigurován, patří tyto změny:

Operace s klastry

- Přidání uzlu do klastru
- Odebrání uzlů z klastru.
- Spuštění klastrového uzlu
- Ukončení klastrového uzlu
- Nastavení verze klastru na poslední úroveň
- Vymazání klastru
- Změna klastrového uzlu

Úlohy skupiny klastrových prostředků

- Vytvoření nových skupin klastrových prostředků.
- Vymazání stávajících skupin klastrových prostředků.
- Spuštění skupiny klastrových prostředků.
- Přidání uzlu do skupiny klastrových prostředků
- Odstranění uzlu ze skupiny klastrových prostředků
- Ukončení skupiny klastrových prostředků.
- Změna domény obnovy pro skupinu klastrových prostředků
- Provedení přepnutí
- Přidání uzlu do domény zařízení
- Odebrání uzlu z domény zařízení

Toto téma vám pomůže také při ukládání konfigurací klastru. Měli byste si přečíst, jak jsou strukturovány úlohy služeb klastrových prostředků a jak klastrová rozhraní API používají uživatelské fronty. Seznamte se se správným způsobem ukončování klastrových úloh a monitorování stavu klastru. Můžete se také dozvědět, jak funkce spolehlivých zpráv a monitorování pulsu (heartbeat) zajišťují aktuálnost informací o stavu klastru.

Úlohy administrativní domény klastru

- Vytvoření administrativní domény klastru
- Přidání monitorovaných prostředků
- Vymazání administrativní domény klastru

Související pojmy

“Funkce spolehlivých zpráv” na stránce 26
Funkce spolehlivých zpráv služeb klastrových prostředků sleduje každý klastrový uzel a zajišťuje, aby všechny uzly měly konzistentní informace o stavu klastrových prostředků.

“Monitorování pulsu (heartbeat)” na stránce 25
Monitorování pulsu je jednou z funkcí služeb klastrových prostředků, která zajišťuje, že každý uzel je aktivní: zasílá signál z jednotlivých uzlů klastru na všechny další uzly v klastru a tím ověřuje, že jsou stále aktivní.

Přidání uzlu do klastru

Pomocí aplikace iSeries Navigator nebo příkazů můžete přidat uzel do klastru.

Pomocí aplikace iSeries Navigator

Tento způsob vyžaduje, aby byla nainstalována a licencována Volba 41 (HA Switchable Resources).

Jednoduchý klastr podporovaný produktem iSeries Navigator může být tvořen maximálně čtyřmi uzly. Pokud v klastru již existují čtyři uzly, je volba **Přidat uzel...** znepřístupněna. Potřebujete-li vytvořit klastr obsahující více než čtyři uzly, použijte klastrové příkazy a rozhraní API nebo produkt typu klastrový middleware dodaný obchodním partnerem IBM. Tak můžete vytvořit klastr obsahující až 128 uzlů.

Chcete-li přidat uzel ke stávajícímu klastru, postupujte takto:

1. V prostředí produktu iSeries Navigator rozbalte Centrální správu.
2. Rozbalte **Klastry**.
3. Rozbalte klastr, ke kterému chcete přidat poznámku.
4. Klepněte pravým tlačítkem na **Uzly** a vyberte **Přidat uzel...**

Použití klastrových příkazů a rozhraní API

K přidání uzlu do klastru můžete použít také tyto metody:

- Příkaz ADDCLUNODE (Přidání záznamu klastrového uzlu)
- Rozhraní QcstAddClusterNodeEntry (Add Cluster Node Entry) API

Související pojmy

"Klastrové příkazy a rozhraní API" na stránce 73
Služby klastrových prostředků i5/OS poskytují sadu CL příkazů, API a systémových prostředků, které mohou být použity poskytovateli aplikací iSeries nebo jejich zákazníky k zvýšení dostupnosti svých aplikací.

"Obchodní partneři IBM pro klastrové aplikační programové prostředky a dostupné klastrovací produkty" na stránce 79
Můžete si zakoupit produkt od obchodního partnera IBM pro klastrové aplikační programové prostředky, který nabízí replikační funkce, jež jsou nedílnou součástí klastrování a jež zjednodušují vytváření a správu klastrů.

Spuštění klastrového uzlu

Spuštěním klastrového uzlu spustíte Služby klastrových prostředků v uzlu v klastru. Počínaje klastry verze 3 se může uzel spustit sám a opětovně se připojit ke klastru - za předpokladu, že najde v klastru aktivní uzel.

Pomocí aplikace iSeries Navigator

Tento způsob vyžaduje, aby byla nainstalována a licencována Volba 41 (HA Switchable Resources).

Když jsou na zadaném uzlu úspěšně spuštěny Služby klastrových prostředků, stav uzlu bude nastaven na *Started*.

Chcete-li spustit klastrování v uzlu, proveďte následující kroky:

1. V produktu iSeries Navigator rozbalte **Centrální správu**.
2. Rozbalte **Klastry**.
3. Rozbalte klastr, který obsahuje uzel, na kterém chcete spustit klastrování.
4. Klepněte na **Uzly**.
5. Klepněte pravým tlačítkem myši na na uzel, na kterém chcete spustit klastr, a vyberte **Klastr → Spustit**.

Klastr

Použití CL příkazů a rozhraní API

Ke spuštění uzlu můžete použít také CL příkazy a API. Když jsou na zadaném uzlu úspěšně spuštěny Služby klastrových prostředků, stav uzlu bude nastaven na *Active*.

- Příkaz STRCLUNOD (Spuštění klastrového uzlu)
- Rozhraní QcstStartClusterNode (Start Cluster Node) API

Související úlohy

"Ukončení klastrových úloh" na stránce 111
Nikdy nezkoušejte ukončit úlohu klastru přímo.

"Obnova po selhání klastrové úlohy" na stránce 138
Selhání úlohy Služeb klastrových prostředků obvykle označuje nějaký další problém.

Ukončení klastrového uzlu

Zastavením (ukončením) uzlu zastavíte služby klastrových prostředků v daném uzlu.

Pomocí aplikace iSeries Navigator

Tento způsob vyžaduje, aby byla nainstalována a licencována Volba 41 (HA Switchable Resources).

Po úspěšném zastavení služeb klastrových prostředků v zadaném uzlu bude stav uzlu nastaven na *Zastavený*.

Chcete-li ukončit klastrování uzlu, postupujte takto:

1. V produktu iSeries Navigator rozbalte **Centrální správu**.
2. Rozbalte **Klastry**.
3. Rozbalte klastr, který obsahuje uzel, na kterém chcete zastavit klastrování.
4. Klepněte na **Uzly**.
5. Klepněte pravým tlačítkem myši na uzel, na kterém chcete ukončit klastry, a vyberte **Klastr → Zastavit**.

Použití CL příkazů a rozhraní API

K ukončení uzlu můžete použít také CL příkazy nebo rozhraní API. Po úspěšném ukončení služeb klastrových prostředků v zadaném uzlu bude stav uzlu nastaven na *Neaktivní*.

- Příkaz ENDCLUNOD (Ukončení klastrového uzlu)
- Rozhraní QcstEndClusterNode (End Cluster Node) API

Související úlohy

"Ukončení klastrových úloh" na stránce 111
Nikdy nezkoušejte ukončit úlohu klastru přímo.

"Obnova po selhání klastrové úlohy" na stránce 138
Selhání úlohy Služeb klastrových prostředků obvykle označuje nějaký další problém.

Úprava verze klastru

Verze klastru definuje úroveň, na které spolu všechny klastrové uzly aktivně komunikují.

Správa verzí klastru je metoda, která umožňuje, aby klastr obsahoval systémy na různých úrovních vydání, které plně spolupracují díky určení používané úrovně komunikačního protokolu.

Chcete-li změnit verzi klastru, musí být všechny klastrové uzly na stejné potenciální verzi. Verzi klastru je pak možné změnit, tak aby odpovídala potenciální verzi. To umožní používat novou funkci. Verzi lze zvyšovat pouze po jedné. Snížit ji nelze jinak, než vymazáním klastru a jeho opětovným vytvořením na nižší verzi. Aktuální verze klastru je na začátku nastavena prvním uzlem definovaným v klastru. Následující uzly přidávané do klastru musí mít verzi stejnou jako aktuální verze klastru nebo verzi následující úrovně; jinak nemohou být do klastru přidány.

Pomocí aplikace iSeries Navigator

Tento způsob vyžaduje, aby byla nainstalována a licencována Volba 41 (HA Switchable Resources).

Chcete-li upravit verzi klastru, postupujte takto:

1. V produktu iSeries Navigator rozbalte **Centrální správu**.
2. Rozbalte **Klastry**.
3. Klepněte pravým tlačítkem myši na klastr a vyberte volbu **Vlastnosti**.
4. Změňte verzi klastru na požadovanou hodnotu.

Použití klastrových příkazů a rozhraní API

K úpravě verze klastru můžete použít také tyto metody:

- Příkaz CHGCLUVE (Změna verze klastru)
- Rozhraní QcstAdjustClusterVersion (Adjust Cluster Version) API

Související pojmy

"Klastrová verze" na stránce 12
Klastrová verze představuje úroveň funkce dostupné v klastru.

"Běžné problémy s klastry" na stránce 132
Zde jsou popsány některé z nejčastějších problémů, ke kterým dochází v klastru, a způsoby, jak se těchto problémů vyvarovat a jak provést nápravu chyb.

Související úlohy

"Vymazání klastru"
Vymažete-li klastr, služby klastrových prostředků budou ve všech aktivních klastrových uzlech ukončeny a budou odebrány z klastru.

Vymazání klastru

Vymažete-li klastr, služby klastrových prostředků budou ve všech aktivních klastrových uzlech ukončeny a budou odebrány z klastru.

Důležité: Pokud v klastru máte nezávislá ASP, měli byste před vymazáním klastru nejprve pomocí příkazu RMVDEVDMNE (Odstranění záznamu domény zařízení) odebrat všechny uzly z domény zařízení.

Pomocí aplikace iSeries Navigator

Tento způsob vyžaduje, aby byla nainstalována a licencována Volba 41 (HA Switchable Resources).

Chcete-li vymazat klastr, postupujte takto:

1. V produktu iSeries Navigator rozbalte **Centrální správu**.
2. Rozbalte **Klastry**.
3. Klepněte pravým tlačítkem myši na klastr, který chcete vymazat, a vyberte volbu **Vymazat**.

Použití CL příkazů a rozhraní API

K vymazání klastru můžete použít také CL příkazy nebo rozhraní API.

- Příkaz DLTCLU (Vymazání klastru)
- Rozhraní QcstDeleteCluster (Delete Cluster) API

Související úlohy

"Úprava verze klastru" na stránce 101
Verze klastru definuje úroveň, na které spolu všechny klastrové uzly aktivně komunikují.

Vytvoření skupiny CRG

Můžete vytvořit několik typů skupin CRG: aplikační, datovou, zařízení a peer CRG.

Chcete-li vytvořit skupinu CRG v klastru, proveďte následující kroky:

1. V produktu Series Navigator rozbalte **Centrální správu → Klastry**.
2. Rozbalte klastr tam, kam chcete přidat skupinu CRG.
   a. Jestliže chcete vytvořit CRG zařízení, klepněte pravým tlačítkem myši na **Přepínatelný hardware** a vyberte **Nová skupina**. Poznámka: Volba **Nová skupina** je k dispozici pouze tehdy, jestliže všechny uzly v doméně obnovy jsou spuštěny. Podrobné informace najdete v tématu Spuštění klastrového uzlu.
   b. Jestliže chcete vytvořit aplikační CRG, klepněte pravým tlačítkem myši na **Přepínatelný software** a vyberte **Přidat produkt**.
   c. Jestliže chcete vytvořit datovou CRG, klepněte pravým tlačítkem myši na **Přepínatelná data** a vyberte **Nová skupina**.
   d. Jestliže chcete vytvořit peer CRG, klepněte pravým tlačítkem myši na **Peer prostředky** a vyberte **Nová peer CRG**.

Použití CL příkazů a rozhraní API

K vytvoření skupiny CRG můžete použít následující příkazy a rozhraní API:

- Příkaz CRTCRG (Vytvoření skupiny klastrových prostředků)
- QcstCreateClusterResourceGroup (Create Cluster Resource Group) API

Vytvoření aplikační CRG s aktivní IP adresou převzetí

Když vytváříte aplikační CRG, můžete uvést, že chcete umožnit aktivní IP adresu převzetí. To je povoleno pouze tehdy, jestliže uživatel konfiguruje IP adresu převzetí.

Dříve jste mohli vytvořit aplikační CRG s aktivní IP adresou převzetí pouze tehdy, pokud ji konfiguroval uživatel. Ale aplikační CRG nebylo možné spustit, jestliže IP adresa převzetí již byla aktivní. Když nyní vytváříte aplikační CRG, můžete uvést, že chcete umožnit aktivní IP adresu převzetí. Když spustíte aplikační CRG, která umožňuje aktivní IP adresu převzetí, CRG bude moci se spustit.

Chcete-li umožnit aktivní IP adresu převzetí, když vytváříte aplikační CRG, proveďte následující kroky:

1. Na rozhraní příkazového řádku napište:

   ```
   CRTCRG CLUSTER(MYCLUSTER) CRG(MYCRG) CRGTYPE(*APP) EXITPGM(QDEVELOP/EXITPGM)
   USRPRF(USER) RCYDMN((NODE1 *PRIMARY)(NODE2 *BACKUP)) TKVINTNETA('10.1.2.1') CFGINTNETA(*USR *YES)
   ```

   Parametr **TKVINTNETA** označuje IP adresu převzetí, která se má použít, a parametr **CFGINTNETA** označuje, že uživatel bude konfigurovat IP adresu převzetí a že tato adresa může být aktivní v době spouštění CRG.

Po vytvoření aplikační CRG k umožnění aktivní IP adresy převzetí můžete spustit CRG.

Spuštění CRG

Můžete spouštět několik typů skupin CRG: aplikační, datová, zařízení a peer CRG.

Chcete-li spouštět CRG, proveďte následující kroky:

1. V produktu Series Navigator rozbalte **Centrální správu → Klastry**.
2. Rozbalte klastr, v němž chcete spouštět skupinu CRG.
   a. Jestliže chcete spustit skupinu CRG zařízení, klepněte na **Přepínatelný hardware**, klepněte pravým tlačítkem myši skupinu přepínatelného hardwaru, kterou chcete spouštět, a vyberte **Spustit**.
   b. Jestliže chcete spustit aplikační skupinu CRG, klepněte na **Přepínatelný software**, klepněte pravým tlačítkem myši na přepínatelný softwarový produkt, který chcete spouštět, a vyberte **Spustit**.
   c. Jestliže chcete spouštět datovou skupinu CRG, klepněte na **Přepínatelná data**, klepněte pravým tlačítkem myši na přepínatelnou datovou skupinu, kterou chcete spustit, a vyberte **Spustit**.
   d. Jestliže chcete spustit skupinu peer CRG, klepněte na **Peer prostředky** a uveďte všechny skupiny peer CRG, klepněte pravým tlačítkem myši na peer CRG, kterou chcete spouštět, a vyberte **Spustit**.

Použití CL příkazů a rozhraní API

Ke spouštění skupiny CRG můžete použít následující příkazy a rozhraní API:

- Příkaz STRCRG (Spuštění skupiny klastrových prostředků)
- Rozhraní QcstStartClusterResourceGroupStart (Cluster Resource Group) API

Změna domény obnovy pro skupinu klastrových prostředků

U skupiny klastrových prostředků je možné měnit role uzlů v doméně obnovy, do domény obnovy lze přidávat uzly a odebírat je z ní. U skupiny klastrových prostředků můžete také změnit jméno serveru (stanoviště) a IP adresy datových portů uzlů v doméně obnovy.

Pomocí aplikace iSeries Navigator

Tento způsob vyžaduje, aby byla nainstalována a licencována Volba 41 (HA Switchable Resources).

Chcete-li změnit roli uzlů v doméně obnovy pro skupinu klastrových prostředků (přepínatelný hardware, přepínatelný software nebo přepínatelná data), přidat uzly do domény obnovy nebo je z ní odebrat, postupujte takto:

1. V produktu iSeries Navigator rozbalte **Centrální správu**.
2. Rozbalte **Klastry**.
3. Rozbalte klastr obsahující přepínatelný hardware, přepínatelný software nebo přepínatelná data, u nichž chcete změnit doménu obnovy.
4. Rozbalte přepínatelný hardware, přepínatelný software nebo přepínatelná data.
5. Klepněte pravým tlačítkem myši na přepínatelný hardware, přepínatelný software nebo přepínatelná data a vyberte volbu **Vlastnosti**.
6. Vyberte stránku **Doména obnovy**.

Chcete-li získat pokyny, jak měnit role nebo přidávat či odebírat uzly, klepněte na tlačítko Nápověda.

Použití CL příkazů a rozhraní API

Chcete-li změnit roli uzlů v doméně obnovy nebo přidat či odebrat uzly, použijte následující CL příkazy nebo rozhraní API:

- Příkaz ADDCRGNODE (Přidání záznamu uzlu do skupiny klastrových prostředků)
- Rozhraní QcstAddNodeToRcvyDomain (Add a Node to Recovery Domain) API
- Příkaz CHGCRG (Změna skupiny klastrových prostředků)
- QcstChangeClusterResourceGroup (Change Cluster Resource Group) API
- Příkaz RMVCRGNODE (Odstranění záznamu uzlu ze skupiny klastrových prostředků)
- Rozhraní QcstRemoveNodeFromRcvyDomain (Remove Node from Recovery Domain) API

Související pojmy

“Doména obnovy” na stránce 11

Doména obnovy je podmnožina klastrových uzlů, které jsou s určitým záměrem, například kvůli akci obnovy nebo synchronizaci událostí, seskupeny do skupiny klastrových prostředků (CRG).

Provedení přepnutí

Provedení ručního přepnutí způsobí, že se aktuální primární uzel přepne na záložní uzel, který je definován v doméně obnovy skupiny klastrových prostředků.

Když k tomu dojde, aktuální role uzlů v doméně obnovy skupiny klastrových prostředků se změní takto:

- Aktuálnímu primárnímu uzlu je přidělena role posledního aktivního záložního uzlu.
- Aktuálnímu prvnímu záložnímu uzlu je přidělena role primárního uzlu.
- Následující záložní uzly jsou posunuty o jeden nahoru v pořadí záložních uzlů.

Přepnutí je povoleno pouze u skupin klastrových prostředků (CRG) v modelu primární-sekundární, které jsou ve stavu AKTIVNÍ.

Poznámka: Provádíte-li přepnutí přepínatelného zařízení (známé také jako CRG zařízení), měli byste z výkonových důvodů provést synchronizaci jména uživatelského profilu, UID a GID.

Pomocí aplikace iSeries Navigator

Tento způsob vyžaduje, aby byla nainstalována a licencována Volba 41 (HA Switchable Resources).

Aby bylo možné přepnout daný prostředek (skupinu přepínatelného hardwaru, přepínatelnou aplikaci nebo skupinu přepínatelných dat) z primárního uzlu na záložní uzel domény obnovy, musí být prostředek ve stavu **Spuštěný**.

Chcete-li provést přepnutí prostředku, postupujte takto:

1. V produktu iSeries Navigator rozbalte **Centrální správu**.
2. Rozbalte **Klastry**.
3. Rozbalte klastr obsahující požadovaný prostředek.
4. Klepněte na **Přepínatelný hardware**, **Přepínatelný software** nebo **Přepínatelná data**.
5. Klepněte pravým tlačítkem myši na požadovaný prostředek a vyberte **Přepnout**.

Použití klastrových rozhraní API

K provedení přepnutí můžete použít také tyto nástroje:

- Příkaz CHGCRGPRI (Změna primárního uzlu skupiny klastrových prostředků)
- Rozhraní QcstInitiateSwitchOver (Initiate Switchover) API

Související pojmy

"Doména obnovy" na stránce 11
Doména obnovy je podmnožina klastrových uzlů, které jsou s určitým záměrem, například kvůli akci obnovy nebo synchronizaci událostí, seskupeny do skupiny klastrových prostředků (CRG).

Související úlohy

"Přepnutí" na stránce 20
K *přepnutí* dojde, když ručně přepnete přístup k prostředku z jednoho serveru na jiný.

Synchronizace jména uživatelského profilu, UID a GID

Přidání uzlu do domény zařízení

Doména zařízení je podmnožina uzlů klastru, které sdílejí prostředky zařízení.

Aby bylo možné uzel přidat do domény obnovy skupiny klastrových prostředků (CRG) zařízení, musí být uzel nejprve definován jako člen domény zařízení. Všechny uzly, které budou v doméně obnovy CRG zařízení, musí být ve stejné doméně zařízení. Klastrový uzel může patřit nejvýše do jedné domény zařízení.

Chcete-li vytvářet a spravovat domény zařízení, musí být ve všech klastrových uzlech, které budou v doméně zařízení, nainstalována Volba 41 (HA Switchable Resources) a musí mít platné licenční klíče.

Pomocí aplikace iSeries Navigator

Chcete-li přidat uzel do domény zařízení v prostředí produktu iSeries Navigator, postupujte takto:

1. V produktu iSeries Navigator rozbalte **Centrální správu**.
2. Rozbalte **Klastry**.
3. Rozbalte klastr, který obsahuje uzel, jenž chcete přidat do domény zařízení.
4. Klepněte na **Uzly**.
5. Klepněte pravým tlačítkem myši na na uzel, který chcete přidat do domény zařízení, a vyberte volbu **Vlastnosti**.
6. Na stránce **Klastrování** zadejte do pole **Doména zařízení** jméno domény zařízení, do které chcete uzel přidat.

Použití CL příkazů a rozhraní API

K přidání uzlu do domény zařízení můžete použít také tyto metody:

- Příkaz ADDDEVDMNE (Přidání záznamu domény zařízení)
- Rozhraní QcstAddDeviceDomainEntry (Add Device Domain Entry) API

Související pojmy

"Domény zařízení" na stránce 16
Doména zařízení je podmnožina uzlů v klastru, které sdílejí prostředky zařízení. Konkrétněji: uzly v doméně zařízení se mohou účastnit na akci přepínání pro určitý soubor prostředků odolného zařízení.

Související úlohy

“Odebrání uzlu z domény zařízení”
Doména zařízení je podmnožina uzlů klastru, které sdílejí prostředky zařízení.

Odebrání uzlu z domény zařízení

Doména zařízení je podmnožina uzlů klastru, které sdílejí prostředky zařízení.

Důležité:

Při odebírání uzlu z domény zařízení buďte opatrní. Odeberete-li určitý uzel z domény zařízení a tento uzel byl aktuálním primárním bodem přístupu k nějakým nezávislým ASP, ″zůstanou″ tato nezávislá ASP v uzlu, který jste odebrali. To znamená, že tato nezávislá ASP přestanou být přístupná ze zbývajících uzlů domény zařízení.

Jakmile je uzel odebrán z domény zařízení a do této domény zařízení ještě patří jeden nebo více stávajících klastrových uzlů, nelze tento uzel přidat zpět do téže domény zařízení. Chcete-li přidat uzel zpět do domény zařízení, musíte tedy postupovat takto:

1. Vymažte nezávislá ASP, která jsou momentálně ve vlastnictví uzlu přidávaného do domény zařízení.
2. Proveďte v uzlu restart systému (IPL).
3. Přidejte uzel do domény zařízení. Viz část Přidání uzlu do domény zařízení.
4. Vytvořte znovu nezávislá ASP, která jste vymazali v 1. kroku.

Pomocí aplikace iSeries Navigator

Tento způsob vyžaduje, aby byla nainstalována a licencována Volba 41 (HA Switchable Resources).

Chcete-li odebrat uzel z domény zařízení v prostředí produktu iSeries Navigator, postupujte takto:

1. V produktu iSeries Navigator rozbalte **Centrální správu**.
2. Rozbalte **Klastry**.
3. Rozbalte klastr, který obsahuje uzel, jenž chcete odebrat z domény zařízení.
4. Klepněte na **Uzly**.
5. Klepněte pravým tlačítkem myši na uzel, který chcete odebrat z domény zařízení, a vyberte volbu **Vlastnosti**.
6. Na stránce Klastrování odeberte záznam (položku) v poli **Doména zařízení**.

Použití CL příkazů a rozhraní API

K odebrání uzlu z domény zařízení můžete použít také tyto metody:

- Příkaz RMVDEVDMNE (Odstranění záznamu domény zařízení)
- Rozhraní QcstRemoveDeviceDomainEntry (Remove Device Domain Entry) API

Související pojmy

“Domény zařízení” na stránce 16
Doména zařízení je podmnožina uzlů v klastru, které sdílejí prostředky zařízení. Konkrétněji: uzly v doméně zařízení se mohou účastnit na akci přepínání pro určitý soubor prostředků odolného zařízení.

Související úlohy

“Přidání uzlu do domény zařízení” na stránce 105
Doména zařízení je podmnožina uzlů klastru, které sdílejí prostředky zařízení.

Přidání diskové jednotky nebo diskové oblasti

Jak systémová událost ovlivňuje klastr

Určité příkazy, které ukončují funkce systému, například PWRDWNSYS (Vypnutí systému), ENDSYS (Ukončení systému) a ENDSBS (Ukončení podsystému) mohou náhle ukončit systém, což způsobí rozdělení klastru.

Ve verzi V5R4 byla provedena určitá zdokonalení, pokud jde o příkazy PWRDWNSYS, ENDSYS, a ENDSBS. Jestliže je v uzlu aktivní klastrování, když se provádějí tyto příkazy, zobrazí se rozhraní QcstEndClusterNode (End Cluster Node) API.

Jestliže chcete, aby se tyto příkazy dokončily, měli byste použít OPTION(*CNTRLD) a uvést odpovídající prodlevu v parametru DELAY. Jinak je možné, že rozhraní End Cluster Node API se nedokončí před tím, než se řízení vrátí do funkce ukončení systému.

Poznámka: Když uživatel uvede **OPTION(*IMMED)**, rozhraní QcstEndClusterNode (End Cluster Node) API bude mít přibližně 30 sekund na dokončení, než bude systém ukončen. To může vést k přepnutí při selhání namísto k ukončení klastrového uzlu.

Vytvoření administrativní domény klastru

Administrativní doménu klastru lze vytvořit v aplikaci iSeries Navigator nebo příkazem CRTADMDMN (Vytvoření administrativní domény klastru).

Chcete-li vytvořit a spravovat administrativní doménu klastru, musí uživatel mít oprávnění k CRG, která se vytváří, příkazům CRG a uživatelský profil QCLUSTER.

Pomocí aplikace iSeries Navigator

Chcete-li vytvořit administrativní doménu klastru, postupujte podle těchto kroků:

1. V produktu iSeries Navigator rozbalte **Centrální správu → Klastry**.
2. Rozbalte klastr, do kterého chcete přidat administrativní doménu klastru.
3. Klepněte pravým tlačítkem myši na **Peer prostředky** a vyberte **Nová administrativní doména**.

Použití CL příkazů a rozhraní API

K vytvoření administrativní domény klastru můžete použít následující příkazy a rozhraní API:

- Příkaz CRTADMDMN (Vytvoření administrativní domény klastru)
- Neexistuje žádné rozhraní API, které vytváří administrativní doménu klastru.

Související pojmy

"Klastrové příkazy a rozhraní API" na stránce 73
Služby klastrových prostředků i5/OS poskytují sadu CL příkazů, API a systémových prostředků, které mohou být použity poskytovateli aplikací iSeries nebo jejich zákazníky k zvýšení dostupnosti svých aplikací.

Přidání záznamů o monitorovaných prostředcích

Můžete přidat záznam o monitorovaném záznamu do administrativní domény klastrů, která představuje prostředek sdílený mezi uzly.

Záznam o monitorovaném prostředku můžete přidat těmito kroky:

1. V produktu iSeries Navigator rozbalte **Centrální správu → Klastry**.
2. Rozbalte klastr, do kterého chcete přidat záznam o monitorovaném prostředku.
3. Rozbalte **Prostředky peer**, čímž se zobrazí seznam všech prostředků peer v klastru.
4. Rozbalte administrativní doménu klastrů, do které chcete přidat záznam o monitorovaném prostředku.
5. Klepněte pravým tlačítkem myši na typ prostředku a vyberte **Přidat záznam o monitorovaném prostředku**.
6. Vyberte atributy, které se mají monitorovat pro záznam o monitorovaném prostředku, a klepněte na **OK**.

Poznámka: Jestliže přidáváte uživatelské profily, které používají synchronizaci hesel jako záznamy o monitorovaném prostředku, pak musí být systémová hodnota Retain Server Security (QRETSVRSEC) nastavena na 1.

Použití CL příkazů a rozhraní API

K přidání monitorovaných prostředků můžete použít následující příkazy a rozhraní API:

- Pro tuto funkci neexistuje žádný ekvivalentní CL příkaz. Zdroj pro nepodporovaný příkaz a program pro zpracování volání (CPP) byl poskytnut v knihovně QUSRTOOL. Chcete-li se dozvědět více o tomto zdroji příkazu a CPP, prohlédněte si člen QFPADINFO v souboru QATTINFO.
- Rozhraní QfpadAddMonitoredResourceEntry (Add Monitored Resource Entry) API

Monitorování administrativní domény klastru

Jakmile je administrativní doména klastru vytvořena a jsou přidány odpovídající záznamy monitorovaných prostředků, administrátor klastru by měl monitorovat činnost v administrativní doméně, aby zajistil další konzistentnost monitorovaných prostředků.

Jestliže je globální stav monitorovaného prostředku nekonzistentní, administrátor by měl provést potřebné kroky k určení toho, proč je prostředek nekonzistentní, problém opravit a provést novou synchronizaci prostředku.

Jestliže je prostředek nekonzistentní, protože aktualizace selhala na jednom nebo více uzlech, uchovávají se informace pro záznam MRE, které vám pomohou zjistit příčinu selhání. V uzlu, na kterém došlo k selhání, se zaprotokoluje zpráva se záznamem MRE, který je příčinou selhání aktualizace. Na jiných uzlech se zaprotokoluje zpráva, která bude uvádět, že došlo k selhání, spolu se seznamem uzlů, na kterých k selhání došlo.

Jakmile bude zjištěna příčina nekonzistence, prostředek lze znovu synchronizovat, buď jako výsledek operace aktualizace v uzlu, kde došlo k selhání, nebo ukončením a restartováním administrativní domény.

Globální stav pro monitorovaný prostředek je vždy nastaven na nekonzistentní, jestliže dojde k výmazu prostředku, jeho přejmenování nebo přesunutí na jiný uzel v doméně. Pokud je to tento případ, záznam MRE byste měli odstranit, protože prostředek již nebude synchronizován v administrativní doméně klastru.

Pomocí aplikace iSeries Navigator

Chcete-li monitorovat administrativní doménu klastru, postupujte podle těchto kroků:

1. V produktu iSeries Navigator rozbalte **Centrální správu → Klastry**.
2. Rozbalte klastr, ke kterému je administrativní doména klastru přidružena.
3. Rozbalte položku **Peer prostředky** a klepněte pravým tlačítkem myši na **Nová administrativní doména** a vyberte **Vlastnosti**. Možné hodnoty pro stav prostředku v aktivní administrativní doméně klastru jsou:

   Consistent
   Hodnoty pro všechny atributy prostředku, které systém monitoruje, jsou stejné ve všech aktivních uzlech v administrativní doméně klastru.

   Inconsistent
   Hodnoty pro všechny atributy prostředku, které systém monitoruje, nejsou stejné ve všech aktivních uzlech v administrativní doméně klastru, nebo administrativní doména klastru není aktivní.

   Pending
   Hodnoty monitorovaných atributů jsou v procesu synchronizace v administrativní doméně klastru.

   Added Záznam monitorovaného prostředku byl přidán do adresáře monitorovaného prostředku v administrativní doméně klastru, ale nebyl ještě synchronizován.

Použití CL příkazů a rozhraní API

K monitorování administrativní domény klastru můžete použít následující příkazy a rozhraní API:

- Pro tuto funkci neexistuje žádný ekvivalentní CL příkaz. Zdroj pro nepodporovaný příkaz a program pro zpracování volání (CPP) byl poskytnut v knihovně QUSRTOOL. Chcete-li se dozvědět více o tomto zdroji příkazu a CPP, prohlédněte si člen QFPADINFO v souboru QATTINFO.
- Rozhraní QfpadRtvMonitoredResourceInfo (Retrieve Monitored Resource Information) API

Monitorování stavu klastru

Služby klastrových prostředků provádějí základní monitorování klastru a jeho komponent pomocí funkce spolehlivých zpráv a monitorování pulsu (heartbeat). Je-li to nutné, podniknou odpovídající akce.

Stav klastru a jeho komponent můžete také monitorovat ručně.

Pomocí aplikace iSeries Navigator

Tento způsob vyžaduje, aby byla nainstalována a licencována Volba 41 (HA Switchable Resources).

Chcete-li monitorovat stav klastru v prostředí produktu iSeries Navigator, postupujte takto:

1. V prostředí produktu iSeries Navigator rozbalte Centrální správu.
2. Rozbalte **Klastry**.
3. Vyhledejte ve složkách v prostředí produktu iSeries Navigator požadovaný klastr. V seznamu v prostředí produktu iSeries Navigator si ve sloupci Stav prohlédněte stav klastru, jeho uzlů a prostředků. Popis možných hodnot ve sloupci Stav najdete v online nápovědě. Také můžete klepnout pravým tlačítkem myši na komponenty klastru a vybráním volby **Vlastnosti** zobrazit informace o klastru.

Použití CL příkazů a rozhraní API

K monitorování stavu klastru můžete použít také tyto příkazy a rozhraní API:

Informace o klastru

Tyto příkazy a rozhraní API načtou informace o klastru - například o uzlech klastru nebo o tom, které IP adresy adaptérů jsou použity v každém uzlu, a o stavu každého klastrového uzlu.

- Příkaz DSPCLUINF (Zobrazení informací o klastru)
- Rozhraní QcstListClusterInfo (List Cluster Information) API
- Rozhraní QcstListDeviceDomainInfo (List Device Domain Info) API
- Rozhraní QcstRetrieveCRSInfo (Retrieve Cluster Resource Services Information) API
- Rozhraní QcstRetrieveClusterInfo (Retrieve Cluster Information) API

Informace o skupinách klastrových prostředků

Tyto příkazy a rozhraní API vygenerují seznam skupin klastrových prostředků a informací o těchto skupinách v klastru, například jméno primárního uzlu každé skupiny klastrových prostředků (CRG) v klastru.

- Příkaz DSPCRGINF (Zobrazení informací o skupině klastrových prostředků)
- Rozhraní QcstListClusterResourceGroups (List Cluster Resource Groups) API
- Rozhraní QcstListClusterResourceGroupInf (List Cluster Resource Group Information) API

Související pojmy

“Funkce spolehlivých zpráv” na stránce 26
Funkce spolehlivých zpráv služeb klastrových prostředků sleduje každý klastrový uzel a zajišťuje, aby všechny uzly měly konzistentní informace o stavu klastrových prostředků.

“Monitorování pulsu (heartbeat)” na stránce 25
Monitorování pulsu je jednou z funkcí služeb klastrových prostředků, která zajišťuje, že každý uzel je aktivní: zasílá signál z jednotlivých uzlů klastru na všechny další uzly v klastru a tím ověřuje, že jsou stále aktivní.

Výkon klastru

Změny prováděné v klastru mohou mít vliv na režii potřebnou ke správě klastru.

Jediné prostředky, které klastrování vyžaduje, jsou ty, které jsou nezbytné k provádění monitorování pulsu (heartbeat), ke správě skupin klastrových prostředků a klastrových uzlů a ke zpracování zpráv posílaných mezi skupinami klastrových prostředků a klastrových uzlů. Jakmile je prostředí klastrování v provozu, může režie vzrůst pouze při provádění změn klastru.

Během normálního provozu by aktivita klastrování měla mít na klastrované systémy minimální vliv.

Související pojmy

“Monitorování pulsu (heartbeat)” na stránce 25
Monitorování pulsu je jednou z funkcí služeb klastrových prostředků, která zajišťuje, že každý uzel je aktivní: zasílá signál z jednotlivých uzlů klastru na všechny další uzly v klastru a tím ověřuje, že jsou stále aktivní.

“Běžné problémy s klastry” na stránce 132
Zde jsou popsány některé z nejčastějších problémů, ke kterým dochází v klastru, a způsoby, jak se těchto problémů vyvarovat a jak provést nápravu chyb.

Vyrovnávání zatížení sítě u klastrů

Můžete vyvažovat zatížení sítě pomocí rozdělení práce mezi komunikační linky, které používáte k spojení uzlů v klastru.

Čím více můžete vyvážit práci tak, abyste udrželi nízké využití zdroje, tím hladčeji poběží váš systém.

Zatížení CPU v záložních uzlech:

Můžete využívat záložní systémy co nejvíce, ale musíte si uvědomit, že na záložní uzel může být přeneseno další pracovní zatížení, když dojde k přepnutí při selhání.

Je velmi důležité mít na vědomí, co je a co není důležité pro vaši práci. Máte-li vysoce kritické aplikace, které přepínají při selhání, musíte zajistit, aby zatížení CPU v záložních uzlech nebylo tak vysoké, že by kritické aplikace nemohly běžet.

Ladění klastrového výkonu

Jelikož ve vašem komunikačním prostředí existují potenciálně značné rozdíly, máte možnost upravit proměnné, které ovlivňují klastrovou komunikaci tak, aby lépe vyhovovaly vašemu prostředí.

Předvolené hodnoty by obvykle měly být přijatelné pro většinu běžných prostředí. Pokud vaše konkrétní prostředí není dobře přizpůsobeno pro tyto předvolby, můžete ladit klastrovou komunikaci tak, aby lépe vyhovovala vašemu prostředí. K dispozici máte dvě úrovně ladění.

Ladění na základní úrovni

Ladění na základní úrovni umožňuje nastavit parametry ladění na předdefinovanou sadu hodnot určených pro dlouhé, krátké a běžné hodnoty prodlev a časového limitu pro posílání zpráv. Když vyberete běžnou úroveň, jsou předvolené hodnoty použity pro výkon klastrové komunikace a konfigurační parametry. Pomocí nižší úrovně zvýší klastrování interval pulsu a hodnoty časových limitů pro posílání různých zpráv. Při méně pulsech a delších časových limitech bude klastr méně citlivý na chyby v komunikaci. Pomocí vysoké úrovně sníží klastrování interval pulsu a hodnoty časových limitů pro posílání různých zpráv. Při častějších pulsech a kratších časových limitech bude klastr více citlivý na chyby v komunikaci.

Ladění na pokročilé úrovni

K dispozici je také ladění na pokročilé úrovni, takže jednotlivé parametry mohou být naladěny jinak než na předdefinovaný rozsah hodnot. To umožňuje více strukturované ladění, aby bylo vyhověno všem speciálním podmínkám ve vašem komunikačním prostředí. Požadujete-li pokročilou úroveň ladění, doporučujeme, abyste získali podporu od zástupce IBM nebo podobnou podporu. Nesprávné nastavení jednotlivých parametrů by mohlo mít snadno za následek snížení výkonu.

Související pojmy

“Laditelné parametry klastrové komunikace” na stránce 94
Rozhraní QcstChgClusterResourceServices (Change Cluster Resource) API umožňuje ladit některé služby klastrové topologie, výkon klastrové komunikace a parametry konfigurace tak, aby lépe vyhovovaly mnoha jedinečným aplikacím a prostředím výstavby sítí, ve kterých dochází ke klastrování. Toto API je k dispozici pro každý klastr, který je spušten v klastrové verzi 2 nebo vyšší.

Související odkazy

Rozhraní Change Cluster Resource Services (QcstChgClusterResourceServices) API

Ukončení klastrových úloh

Nikdy nezkoušejte ukončit úlohu klastru přímo.

Pokud potřebujete ukončit jakoukoliv úlohu spuštěnou v klastrovaném prostředí, postupujte takto:

1. Ukončete klastrový uzel.
2. Odstraňte problém.
3. Spusťte klastrový uzel.

Související úlohy

"Ukončení klastrového uzlu" na stránce 100
Zastavením (ukončením) uzlu zastavíte služby klastrových prostředků v daném uzlu.

"Spuštění klastrového uzlu" na stránce 100
Spuštěním klastrového uzlu spustíte Služby klastrových prostředků v uzlu v klastru. Počínaje klastry verze 3 se může uzel spustit sám a opětovně se připojit ke klastru - za předpokladu, že najde v klastru aktivní uzel.

Monitorování a řízení prostředků (RMC)

Monitorování a řízení prostředků (RMC) je obecný rámec pro správu, monitorování a manipulaci s prostředky, jako jsou například fyzické nebo logické systémové objekty.

RMC se používá jako komunikační mechanismus pro oznamování servisních událostí konzoli HMC (Hardware Management Console). Jestliže RMC není aktivní, pak servisní události nebudou oznámeny konzoli HMC. Následující seznam popisuje služby, které jsou přidruženy k RMC:

Démon CAS

Účel: Působí jako autentizační server pro RMC.

Jméno úlohy: QCSTCTCASD

Démon RMC

Účel: Monitoruje prostředky tím, že komunikuje s manažerem prostředků.

Jméno úlohy: QCSTCTRMCD

Démon SRC

Účel: Monitoruje stav jiných úloh RMC; restartuje úlohu, pokukd tato určitá úloha neočekávaně skončí.

Jméno úlohy: QCSTSRCD

Správci prostředků (RM)

Správce prostředků (RM) je úloha, která spravuje a zajišťuje rozhraní mezi TMC a skutečnými fyzickými nebo logickými objekty. Třebaže RMC zajišťuje základní výtahy, například třídy prostředků, prostředky a atributy pro reprezentaci fyzických nebo logických objektů, sám o sobě nereprezentuje žádné skutečné objetky. Manažer RM mapuje skutečné objekty do výtahů RMC. Následující seznam popisuje různé správce prostředků podporované pro RMC:

Auditační protokol RM

Účel: Zajišťuje schopnost zaznamenávání informací o činnosti systému.

Jméno úlohy: QYUSALRMD

CSMAgent RM

Účel: Zajišťuje třídy pro reprezentaci serveru správy, což je konzole HMC.

Jméno úlohy: QYUSCMCRMD

Hostitel RM

Účel: Zajišťuje třídy prostředků, které představují jednotlivé počítače.

Jméno úlohy: QCSTCTHRMD

Servisní RM

Účel: Spravuje informace o problému a připravuje je pro dodání do konzole HMC.

Jméno úlohy: QSVRMSERMD

Spuštění nebo ukončení RMC

Všechny úlohy RMC včetně úloh RM jsou v podsystému QSYSWRK a spouští se automaticky, když se spouští podsystém. Protokol TCP/IP musí být aktivní, aby spuštění mohlo proběhnout. Démon RMC vyžaduje, aby protokol TCP/IP byl aktivní. Jestliže se protokol TCP/IP stane neaktivní, démon RMC bude ukončen. Démon RMC se automaticky restartuje démonem SRC, jakmile se protokol TCP/IP stane znovu aktivním. Za normálních okolností se od uživatele nevyžadují žádné kroky. Jestliže je nutné RMC spustit ručně, spusťte následující příkaz:

```
SBMJOB CMD(CALL PGM(QSYS/QCSTCTSRCD)) JOBD(QSYS/QCSTSRCD) PRTDEV(*JOBD) OUTQ(*JOBD)
USER(*JOBD) PRTTXT(*JOBD) RTGDTA(RUNPTY50)
```

Jestliže je nutné RMC ukončit ručně, použijte k ukončení úlohy QCSTSRCD příkaz ENDJOB. Tento příkaz by měl ukončit všechny úlohy RMC. Jestliže se všechny úlohy RMC neukončí, pak ručně ukončete každou z úloh uvedených výše.

Struktura úloh a uživatelské fronty

Když spravujete klastr, musíte znát strukturu a uživatelské fronty.

Struktura úloh služeb klastrových prostředků

Služby klastrových prostředků jsou tvořeny sadou vícevláknových úloh. Pokud je na serveru aktivní klastrování, jsou v podsystému QSYSWRK pod uživatelským profilem QSYS spuštěny následující úlohy. Úlohy jsou spuštěny pomocí popisu úlohy QDFTSVR, ale s úrovní protokolování nastavenou tak, aby byl generován protokol úlohy.

- Řízení klastru je tvořeno jednou úlohou, která se jmenuje QCSTCTL.
- Správce skupin klastrových prostředků je tvořen jednou úlohou, která se jmenuje QCSTCRGM.
- Skupiny klastrových prostředků jsou tvořeny jednou úlohou pro každý objekt skupiny klastrových prostředků (CRG). Jméno úlohy je stejné jako jméno skupiny klastrových prostředků. To zahrnuje administrativní doménu klastru.
- Pokud je jedna nebo více položek seznamu zařízení ve skupině CRG odolných zařízení nastavena tak, aby byly v případě přepnutí nebo přepnutí při selhání převedeny do režimu online, budou spuštěny (umístěny do fronty úloh) další úlohy, které budou provádět funkci logického zapnutí.

Úlohy QCSTCTL a QCSTCRGM jsou úlohy nezbytné pro činnost klastru. Tyto úlohy musí být spuštěny, aby mohl být uzel v klastru aktivní.

Při použití většiny rozhraní API skupin klastrových prostředků je spuštěna (umístěna do fronty úloh) samostatná úloha, která použije uživatelský profil, jenž byl zadán v okamžiku vytvoření skupiny klastrových prostředků. V této úloze je zavolán ukončovací program definovaný ve skupině klastrových prostředků. Úlohy jsou standardně umísťovány do fronty úloh QBATCH. Tato fronta úloh obecně slouží pro dávkové provozní úlohy. Ukončovací programy zde proto budou zpožděny nebo nebudou provedeny vůbec. Chcete-li, aby příkazy rozhraní API byly prováděny efektivně, vytvořte samostatný uživatelský profil, popis úlohy a frontu úloh, které budou používány skupinami klastrových prostředků. Určete nový uživatelský profil pro všechny skupiny klastrových prostředků, které vytvoříte. Ve všech uzlech domény obnovy definované pro skupinu klastrových prostředků je zpracováván stejný program.

Příkazem CHGCLURCY (Změna obnovy klastru) můžete restartovat úlohu skupiny klastrových prostředků, která skončila, aniž byste ukončili a restartovali klastrování v uzlu.

Klastrová rozhraní API a uživatelské fronty

Funkce prováděné rozhraním API, které mají informační parametr Results, probíhají asynchronně a své výsledky posílají po ukončení zpracování do uživatelské fronty. Tato uživatelská fronta musí být vytvořena dříve, než rozhraní API zavoláte. Uživatelskou frontu lze vytvořit pomocí rozhraní QUSCRTUQ (Create User Queue) API.Fronta musí být vytvořena jako klíčovaná fronta. Klíč uživatelské fronty je popsán ve formátu záznamu (položky) uživatelské fronty. Jméno uživatelské fronty je předáno do rozhraní API. Dokumentace k API klastru obsahuje příklady toho, jak používat uživatelské fronty v rozhraních API u klastrů.

Použijete-li rozhraní QcstDistributeInformation (Distribute Information), informace posílané mezi uzly jsou ukládány do uživatelské fronty, která byla zadána v okamžiku vytvoření CRG. Tato fronta musí být vytvořena uživatelem ve všech aktivních uzlech domény obnovy před tím, než bude rozhraní Distribute Information API použito. Podrobné informace o tom, kdy musí fronty pro distribuci informací existovat, najdete v rozhraní QcstCreateClusterResourceGroup (Create Cluster Resource Group) API .

Fronta zpráv přepnutí při selhání dostává zprávy týkající se průběhu přepnutí při selhání.

Související pojmy

“Správa uživatelských profilů ve všech uzlech” na stránce 90
Pro udržování uživatelských profilů ve všech uzlech v klastru můžete použít dva mechanismy.

“Určení, zda existuje problém s klastrem” na stránce 121
Zde začněte při diagnostikování problémů s klastry.

Související úlohy

“Obnova po selhání klastrové úlohy” na stránce 138
Selhání úlohy Služeb klastrových prostředků obvykle označuje nějaký další problém.

Fronta zpráv přepnutí při selhání

Fronta zpráv přepnutí při selhání dostává zprávy týkající se průběhu přepnutí při selhání.

Pomocí této fronty zpráv může být administrátor uvědoměn předtím, než dojde k přepnutí při selhání. To dává administrátorovi možnost zrušit přepnutí při selhání, je-li v daném okamžiku žádoucí přepnutí zabránit.

Fronta zpráv přepnutí při selhání se definuje při tvorbě skupiny klastrových prostředků pomocí rozhraní QcstCreateCluster (Create Cluster) API. Lze ji také upravovat pomocí CL příkazu a API pro změnu skupin klastrových prostředků. Frontu zpráv přepnutí při selhání nelze použít v rozhraní produktu iSeries Navigator pro správu klastru.

Další informace o frontách zpráv přepnutí při selhání najdete v dokumentaci k rozhraní Cluster Resource Group API. Přečtěte si následující popis obsahující podrobné informace o používání fronty zpráv přepnutí při selhání.

CL příkazy

- Příkaz CRTCRG (Vytvoření skupiny klastrových prostředků)
- Příkaz CHGCRG (Změna skupiny klastrových prostředků)

Rozhraní API

- Rozhraní QcstCreateClusterResourceGroup (Create Cluster Resource Group) API
- Rozhraní QcstChangeClusterResourceGroup (Change Cluster Resource Group) API

Správa uživatelských profilů ve všech uzlech

Pro udržování uživatelských profilů ve všech uzlech v klastru můžete použít dva mechanismy.

Jeden mechanismus je vytvořit administrativní doménu klastru ke sledování sdílených prostředků ve všech uzlech v klastru. Administrativní doména klastru dokáže kromě uživatelských profilů monitorovat několik typů prostředků, čímž zajišťuje snadnou správu prostředků, které se sdílejí mezi uzly. Podrobné informace o těchto prostředcích najdete v

tématu Monitorované prostředky. Když se aktualizují uživatelské profily, změny se automaticky přenesou na jiné uzly, jestliže je administrativní doména klastru aktivní. Jestliže administrativní doména klastru není aktivní, změny se rozšíří, jakmile bude administrativní doména klastru aktivována.

Poznámka: Jestliže plánujete sdílení uživatelských profilů, které používají synchronizaci hesel v rámci klastru, musíte nastavit systémovou hodnotu Retain Server Security (QRETSVRSEC) na 1.

V druhém mechanismu mohou administrátoři použít také Centrální správu v produktu iSeries Navigator k provádění funkcí na více systémech a skupinách systémů. Tato podpora zahrnuje některé běžné úkoly správy uživatelů, které operátoři potřebují provádět u více systémů v klastru. Centrální správa umožňuje provádět činnosti s uživatelskými profily pro skupiny systémů. Administrátor může určit, že v cílových systémech má být při vytvoření uživatelského profilu spuštěn příkaz pro následné přenesení zabezpečovacích informací.

Zálohování a obnova klastrů

Pokud ve svých systémech používáte klastrování, je i tehdy důležité, abyste k ochraně dat vytvořili strategii zálohování a obnovy.

Plánujete-li jako strategii zálohování používat klastrování, tj. chcete mít jeden systém v provozu, zatímco druhý bude během zálohování mimo provoz, měli byste mít v klastru minimálně tři systémy. Budete-li mít v klastru tři systémy, bude vždy jeden systém k dispozici, na který bude možné přepnout v případě poruchy.

Ukládání a obnova skupin klastrových prostředků

Skupinu klastrových prostředků je možné ukládat bez ohledu na to, zda je klastr aktivní nebo neaktivní. Pro obnovu skupiny klastrových prostředků platí tato omezení:

- Pokud je klastr v provozu a skupina klastrových prostředků je pro tento klastr známa, nelze tuto skupinu obnovit.
- Není-li uzel nakonfigurován pro klastr, nelze skupinu klastrových prostředků obnovit.

Skupinu klastrových prostředků je možné obnovit, jestliže je klastr aktivní, skupina klastrových prostředků není tomuto klastru známa, uzel je v doméně obnovy této skupiny klastrových prostředků a jméno klastru odpovídá jménu klastru ve skupině klastrových prostředků. Skupinu klastrových prostředků lze obnovit, pokud je klastr nakonfigurován, avšak není v daném uzlu aktivní, a pokud daný uzel je v doméně obnovy skupiny klastrových prostředků.

Příprava na zhroucení systému

Zhroutí-li se systém, bude nutné klastr rekonfigurovat. Chcete-li se připravit na takovýto scénář, doporučujeme, abyste uložili informace o konfiguraci klastru a pořídili si tištěnou kopii těchto informací.

1. Po provedení změn konfigurace klastru použijte příkaz SAVCFG (Uložení konfigurace) nebo příkaz SAVSYS (Uložení systému), a to tak, aby obnovené interní informace klastru byly aktuální a konzistentní s ostatními klastrovými uzly. Podrobné informace o provádění příkazů SAVCFG a SAVSYS najdete v tématu Ukládání konfiguračních informací.
2. Kopii informací o konfiguraci klastru si můžete vytisknout po každé změně konfigurace. Příkaz DSPCLUINF (Zobrazení informací o klastru můžete použít k tisku informací o konfiguraci klastru. Uchovávejte kopii záložních pásek pro případ zhroucení systému, pokud byste v takovém případě chtěli rekonfigurovat celý klastr.

Související pojmy

“Obnova klastru ze záložních pásek” na stránce 140
Během běžných operací byste nikdy neměli muset provádět obnovu ze záložní pásky.

Uložení informací o konfiguraci

“Obnova klastru po úplné ztrátě systému” na stránce 140
Tyto informace použijte společně s příslušným kontrolním seznamem v publikaci Zálohování a obnova při obnově celého systému po jeho úplné ztrátě následkem neočekávaného výpadku serveru.

“Uložení konfigurací klastrů” na stránce 115
Chcete-li uložit objekty skupiny klastrových prostředků, můžete použít příkazy.

“Obnova klastru po zhroucení” na stránce 140
V případě katastrofy, při které dojde ke ztrátě všech uzlů, budete muset znovu konfigurovat klastr.

Související úlohy

Plánování strategie zálohování a obnovy

Tisk systémových informací

Uložení konfigurací klastrů

Chcete-li uložit objekty skupiny klastrových prostředků, můžete použít příkazy.

Příkazem SAVSYS (Uložení systému) můžete uložit svůj celý systém, nejen konfigurovaný klastr. Příkazem SAVCFG (Uložení konfigurace) můžete uložit konfigurovaný systém.

SAVOBJ(QUSRSYS/*ALL) OBJTYPE (*CRG)

Poznámka: Objekty skupiny klastrových prostředků mohou být uloženy pouze pro aktuální vydání.

Související úlohy

“Zálohování a obnova klastrů” na stránce 114
Pokud ve svých systémech používáte klastrování, je i tehdy důležité, abyste k ochraně dat vytvořili strategii zálohování a obnovy.

Související odkazy

Příkaz SAVSYS (Uložení systému)

Příkaz SAVCFG (Uložení konfigurace)

Příklady: Klastrové konfigurace

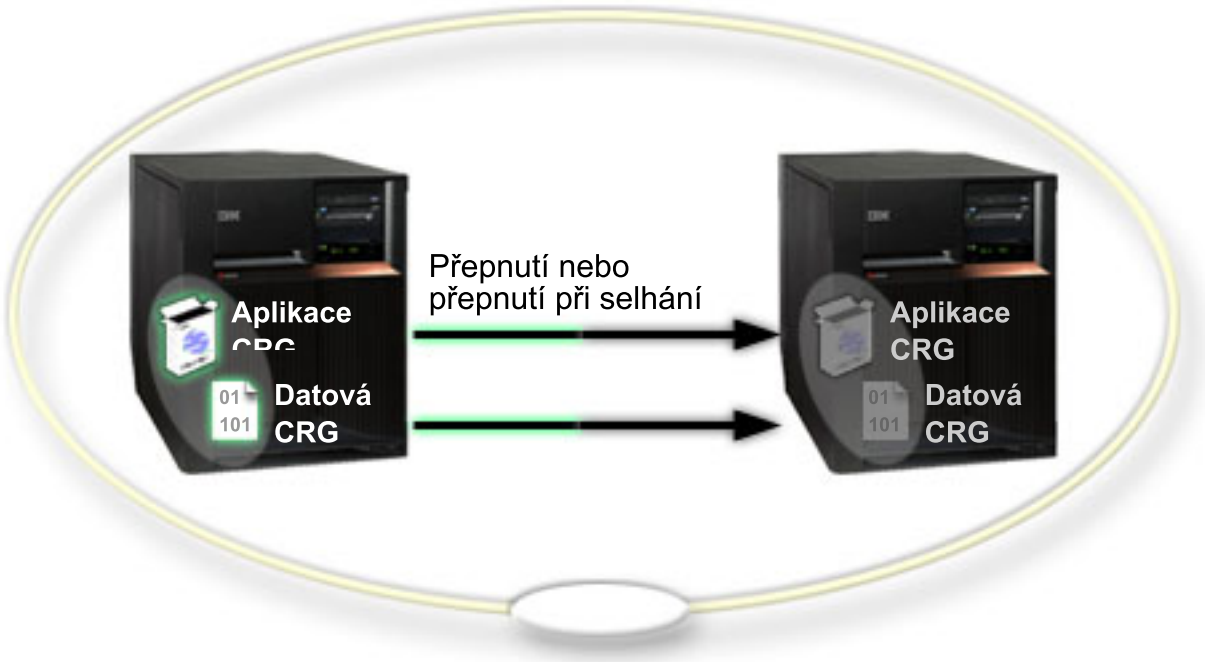
Tyto příklady typických implementací klastrů vám pomohou porozumět, kdy a za jakých podmínek může být použití klastrů výhodné.

Příklad: Jednoduchý dvouuzlový klastr

Tento příklad konfigurace popisuje základní klastr, který obsahuje dva uzly.

Tento příklad konfigurace nabízí následující:

- Obousměrnou replikaci a přepnutí při selhání
- Dvouvrstvé prostředí
- Aplikace a data se přesouvají společně
- Pro offline zpracování dat se používá zálohování
- Souvislá operace na peer CRG



V tomto příkladu funguje uzel L jako primární uzel pro dvě skupiny klastrových prostředků - aplikační skupinu CRG a datovou skupinu CRG. Obsahuje také peer CRG, která zajišťuje souvislé činnosti pro kterýkoli z těchto uzlů. Pro aplikační skupinu CRG v uzlu L budou pravidelně spouštěny dva ukončovací programy. Důvodem, proč mohou být oba ukončovací programy spouštěny ve stejnou dobu, je situace, kdy voláte rozhraní Start CRG API. Ukončovací program se spustí a nepřetržitě běží, zatímco aplikační skupina CRG je aktivní. Pokud byste zavolali rozhraní End CRG API pro aplikační skupinu CRG, spustil by se jiný ukončovací program. Uzel R je prvním a jediným záložním uzlem označeným v doméně obnovy každé skupiny klastrových prostředků. Data, která jsou asociovaná s datovou skupinou CRG a související informace o aplikacích, které jsou asociované s aplikační skupinou CRG, jsou replikovány z uzlu L do uzlu R. Jestliže uzel L selže nebo musí být z administrativních důvodů vypnut, inicializuje se přepnutí při selhání nebo přepnutí a uzel R se stane primárním uzlem pro aplikační i datové skupiny CRG. Uzel R převezme IP adresu definovanou pro aplikační skupinu CRG.

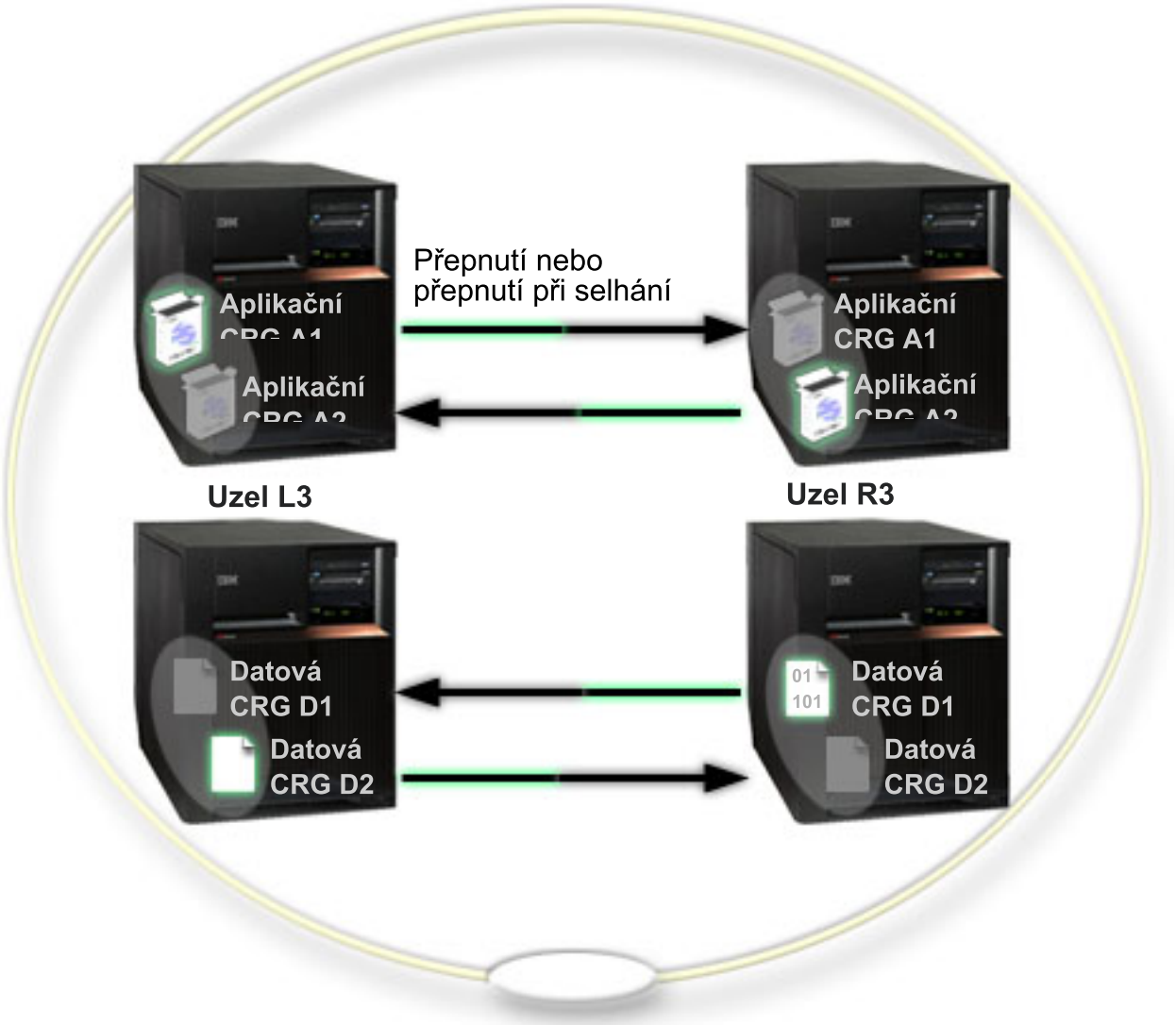
Poznámka: Zatímco je uzel L vypnutý, dostupnost systému je nechráněná, protože neexistuje žádná záloha pro případ, že selže také uzel R. Po obnově uzlu L a jeho opětovném připojení ke klastru se z něj stane záloha pro obě skupiny klastrových prostředků. V tomto okamžiku bude replikace probíhat z uzlu R do uzlu L. Pokud chcete, aby uzel L opět převzal roli primárního uzlu, je třeba provést administrativní přepnutí.

Příklad: Čtyřuzlový klastr

Tento příklad použijte k vytvoření složitějšího klastru, který obsahuje čtyři uzly.

Tento příklad konfigurace nabízí následující:

- Obousměrnou replikaci a přepnutí při selhání
- Třívrstvé prostředí
- Aplikace a data se přesouvají nezávisle
- Pro běžnou produkci různého zatížení se používá zálohování



Příklad čtyřuzlového klastru nabízí dodatečnou pružnost, která je možná u klastru iSeries. Existují dvě aplikační skupiny klastrových prostředků (A1 a A2) a dvě datové skupiny klastrových prostředků (D1 a D2). Data asociovaná se skupinou D1 jsou kritická data pro aplikaci asociovanou se skupinou A1. Data asociovaná se skupinou D2 jsou kritická data pro aplikaci asociovanou se skupinou A2. Jelikož se jedná o třívrstvé prostředí, aplikace existují ve druhé vrstvě (uzel L2 a uzel R2) a data jsou oddělena do třetí vrstvy (uzel L3 a uzel R3).

Skupina klastrových prostředků (CRG)	Primární	Záložní
Aplikační CRG A1	L2	R2
Aplikační CRG A2	R2	L2
Datová CRG D1	R3	L3
Datová CRG D2	L3	R3

To umožňuje schopnost přebírání jak na aplikační, tak na datové úrovni. Všechny čtyři uzly jsou použity pro běžnou produkci. Jsou zároveň také použity pro zálohování ostatních systémů v klastru. Obě aplikace a jejich asociovaná data by měly být v klastru vždy dostupné. Výpadek kteréhokoli uzlu neporuší dostupnost. Kromě toho neporuší dostupnost ani současný výpadek uzlu na aplikační úrovni a uzlu na datové úrovni.

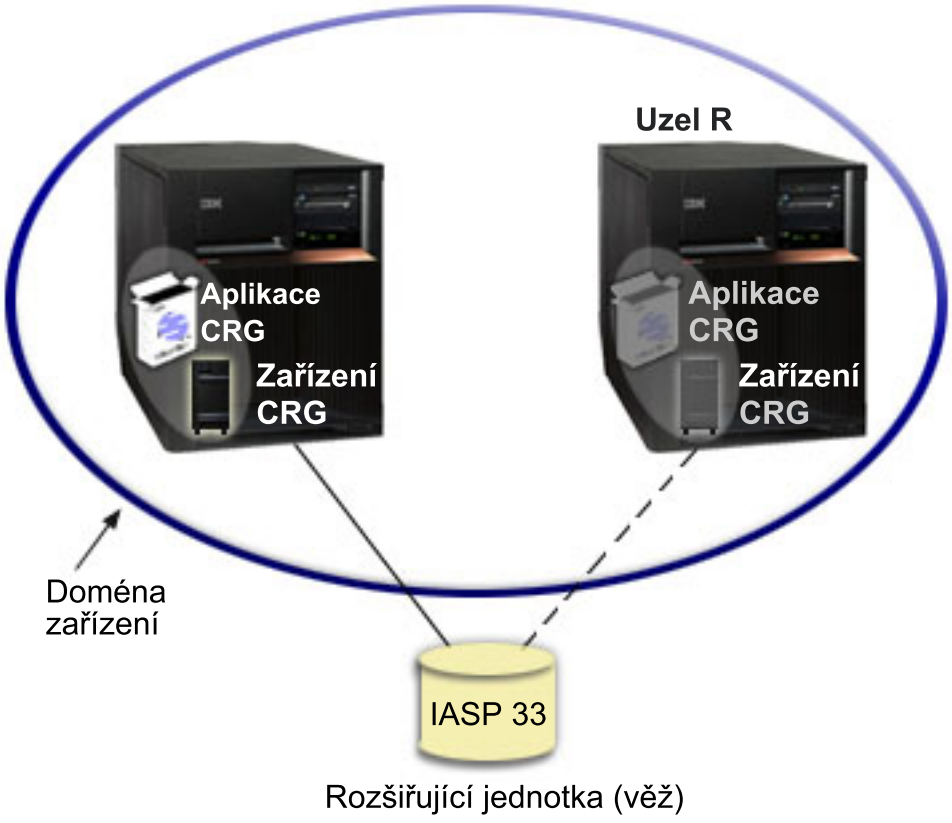
Poznámka: V obou případech není u klastru zajištěno, že budou všechny klastrové prostředky během výpadku uzlu replikovány. Tento problém můžete vyřešit tak, že budete mít více než jednu zálohu pro každý kritický klastrový prostředek.

Příklad: Klastr s přepínanými disky používající nezávislá ASP

Klastr používající technologii přepínaných disků poskytuje alternativu k replikaci dat. V klastru s přepínanými disky jsou data ve skutečnosti obsažena v nezávislých ASP.

Tento příklad konfigurace nabízí následující:

- Jedno přepínatelné nezávislé ASP s nečinným serverem ve stavu standby. Nezávislé ASP je umístěno v rámci kolekce přepínatelných diskových jednotek.
- Dvouvrstvé prostředí
- Aplikace a data se přesouvají společně.
- Záloha používaná pro různé pracovní zatížení neasociované s těmito aplikačními daty.
- Žádná replikace dat - v klastru existuje pouze jedna kopie dat.



V tomto příkladu patří uzel L a uzel R do stejné domény zařízení. Uzel L aktuálně funguje jako primární uzel pro dvě skupiny klastrových prostředků - aplikační skupinu CRG a skupinu zařízení CRG. Uzel R je první (a jedinou) zálohou pro obě skupiny klastrových prostředků. Data asociovaná se skupinou zařízení CRG jsou obsažena v přepínatelném zdroji, jako je například externí rozšiřující jednotka (věž). Související informace o aplikacích asociované s aplikační skupinou CRG jsou buď uloženy ve stejné věži, nebo jsou replikovány z uzlu L do uzlu R. Selže-li uzel L nebo nastane-li nutnost vypnout uzel L z administrativních důvodů, stane se uzel R primárním uzlem pro obě skupiny klastrových prostředků. Uzel R převezme IP adresu definovanou pro aplikační skupinu CRG. Uzel R také převezme vlastnictví přepínatelného zdroje definovaného pro skupinu zařízení CRG.

Poznámka: Zatímco je uzel L vypnutý, dostupnost systému je nechráněná, protože neexistuje žádná záloha pro případ, že selže také uzel R. Po obnově uzlu L a jeho opětovném připojení ke klastru se z něj stane záloha pro obě skupiny klastrových prostředků. Pokud chcete, aby uzel L opět převzal roli primárního uzlu, je třeba provést administrativní přepnutí.

Související pojmy

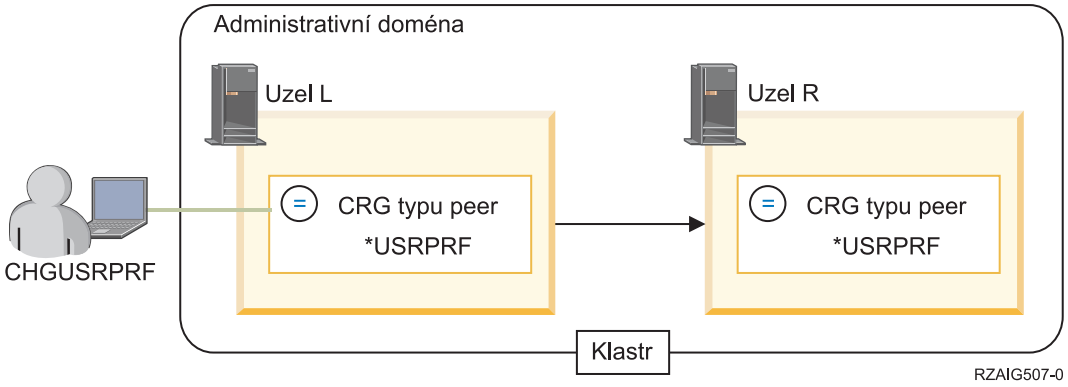
Konfigurace nezávislých ASP

Příklad: Administrativní doména klastru pro správu peer prostředků

Uvádí příklad konfigurace administrativní domény klastru používané k monitorování prostředků v klastru.

Tento příklad konfigurace nabízí následující:

- Klastr se dvěma uzly
- Administrativní doména klastru se dvěma uzly ve svém seznamu uzlů
- Záznam monitorovaného prostředku (MRE) pro uživatelský profil, který má být synchronizován v doméně.



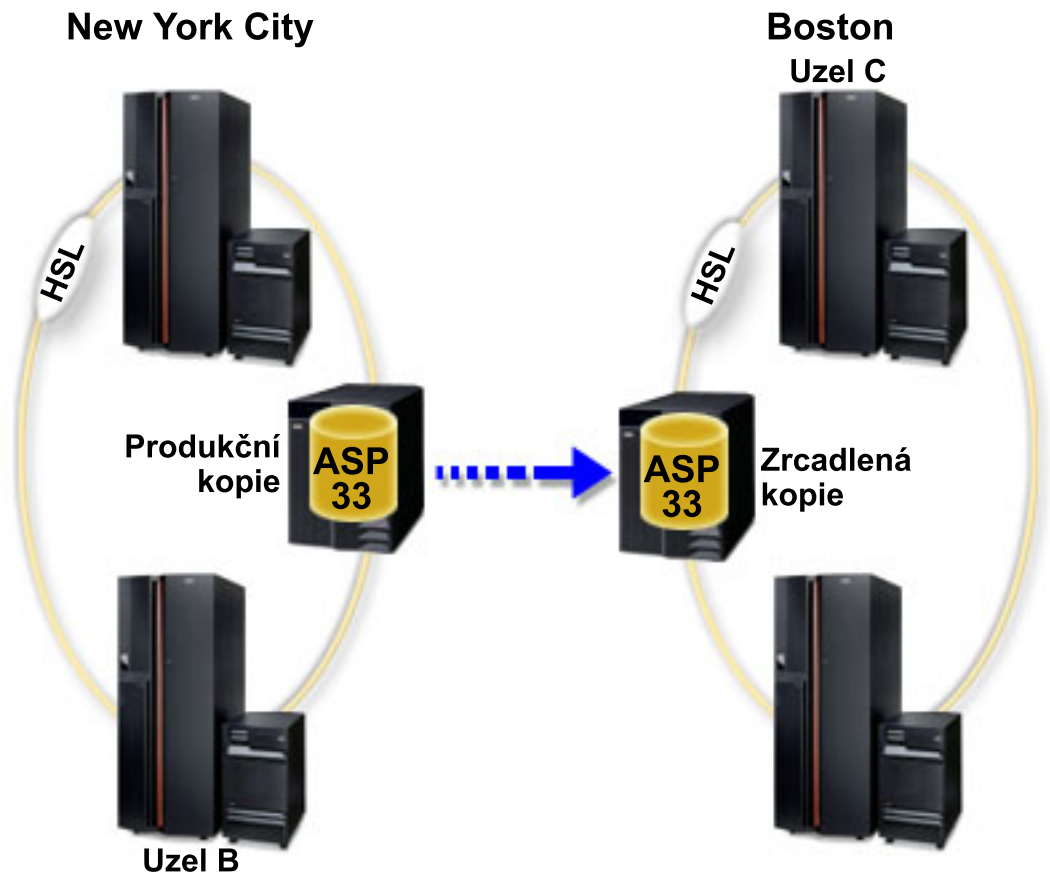
Administrátor v tomto příkladu chce zajistit, aby uživatelský profil zůstal v celém klastru konzistentní, takže vytvořil administrativní doménu klastru ke sledování a synchronizaci změn na uživatelském profilu. Administrativní doména klastru je reprezentována peer CRG, která obsahuje uzel L a uzel R. Záznam monitorovaného prostředku se přidá do administrativní domény klastru pro uživatelský profil. V tomto příkladu byly zadány všechny atributy uživatelského profilu, když byl přidán záznam MRE. Když se tedy jakýkoli atribut uživatelského profilu změní v uzlu L nebo R, změna se automaticky rozšíří na aktivní uzly v doméně, jakmile se spustí skupina CRG.

Následující kroky popisují, co administrátor provedl pro nastavení tohoto příkladu:

1. Vytvoření klastru s uzly L a R
2. Vytvoření administrativní domény klastru v uzlech L a R
3. Přidání záznamu MRE, který představuje uživatelský profil
4. Spuštění peer CRG, která představuje administrativní doménu klastru
5. Změna uživatelského profilu v uzlu L nebo v uzlu R. Uživatelský profil v jiném uzlu se změní automaticky v administrativní doméně klastru. Globální stav monitorovaného prostředku bude konzistentní, když změna bude úspěšná.

Příklad: Nezávislá ASP s geografickým zrcadlením

Následující příklad ukazuje jeden způsob, jak lze konfigurovat geografické zrcadlení. Uzel A a Uzel B jsou umístěny v městě New York. Uzel C a uzel D jsou umístěny v Bostonu. Všechny čtyři uzly jsou konfigurovány ve stejné doméně obnovy. Výrobní kopii lze přepínat mezi uzly A a B. Zrcadlenou kopii lze přepínat mezi uzly C a D. Jelikož všechny uzly jsou ve stejné doméně obnovy, zdrojový systém v New Yorku si může také vyměnit úlohy s cílových systémem v Bostonu, takže Boston může hostit výrobní kopii.



Tato firma definovala následující úlohy pro uzly v doméně obnovy:

Uzel	Úloha
Uzel A	Primární
Uzel B	1. záložní
Uzel C	2. záložní
Uzel D	Záloha 3

V případě přírodní katastrofy v New Yorku se uzel C v Bostonu stane primárním uzlem tím, že svou zrcadlenou kopii povýší na výrobní kopii. Uzel C se stane zdrojovým systémem pro geografické zrcadlení, ačkoli geografické zrcadlení bude pozastaveno, protože už nebude žádný cílový uzel kvůli přírodní katastrofě v New Yorku. Až se stanoviště v New Yorku zotaví, uzel A se stane záložním uzlem a jeho předchozí výrobní kopie se stane zrcadlenou kopií.

Odstraňování problémů s klastry

Vyhledejte řešení týkající se nápravy chyb u problémů, které jsou specifické pro klastry.

Občas se může stát, že klastr nebude pracovat správně. Toto téma obsahuje informace o problémech, se kterými se můžete u klastrů setkat.

Určení, zda existuje problém s klastrem

Zde začněte při diagnostikování problémů s klastry.

Občas se může zdát, že klastr nepracuje správně. Domníváte-li se, že se vyskytl problém, můžete použít níže uvedenou nápovědu a určit, zda problém existuje a jaké je povahy.

- **Určení toho, zda je v systému aktivní klastrování.**

  Chcete-li určit, zda jsou aktivní Služby klastrových prostředků, vyhledejte v podsystému QSYSWRK dvě úlohy - QCSTCTL a QCSTCRGM. Pokud jsou tyto úlohy aktivní, potom jsou aktivní také Služby klastrových prostředků. Můžete použít funkci Work Management v produktu iSeries Navigator a prohlédnout si úlohy v podsystému nebo můžete použít CL příkaz WRKACTJOB (Práce s aktivními úlohami). Chcete-li si prohlédnout stavové informace pro klastr, můžete také použít příkaz DSPCLUINF (Zobrazení informací o klastru).

  – Mohou být aktivní také další úlohy pro Služby klastrových prostředků. Struktura úlohy služeb klastrových prostředků poskytuje informace o tom, jak jsou úlohy služeb klastrových prostředků formátovány.

- **Vyhledejte zprávy označující problém.**

  – V QSYSOPR vyhledejte zprávy, které čekají na odpověď.

  – V QSYSOPR vyhledejte chybové zprávy, které označují problém s klastrem. Obecně budou tyto zprávy v rozsahu CPFBB00 až CPFBBFF.

  – Zobrazte protokol historie (CL příkaz DSPLOG) pro zprávy, které označují problém s klastrem. Obecně budou tyto zprávy v rozsahu CPFBB00 až CPFBBFF.

- **V protokolech úloh pro klastrové úlohy vyhledejte závažné chyby.**

  Tyto úlohy jsou zpočátku nastaveny s úrovní protokolování (4 0 *SECLVL), takže si můžete prohlédnout potřebné chybové zprávy. Měli byste zajistit, aby tyto úlohy a úlohy ukončovacích programů měly nastaveny vhodnou prokolovací úroveň. Není-li klastrování aktivní, můžete přesto vyhledávat soubor pro souběžný tisk pro klastrové úlohy a úlohy ukončovacích programů.

- **Pokud máte podezření, že došlo k zastavení, prohlédněte si zásobníky volání klastrových úloh.**

  Určete, zda není některý program v určitém druhu DEQW (dequeue wait - čekání mimo frontu). Pokud tomu tak je, zkontrolujte zásobník volání každé cesty zpracování a podívejte se, zda některý z nich neobsahuje getSpecialMsg.

- **Zkontrolujte klastrové záznamy v protokolech kódu VLIC (vertical licensed internal code).**

  Tyto záznamy v protokolu mají hlavní kód 4800.

- **Použijte příkaz NETSTAT a určete, zda se ve vašem komunikačním prostředí nevyskytují některé abnormality.**

  Příkaz NETSTAT vrátí informace o stavu předepsaných cest k síti TCP/IP, rozhraní, spojení TCP a UDP portech v systému.

  – Použijte příkaz Netstat Option 1 (Práce se stavem rozhraní TCP/IP) a ujistěte se, že IP adresy vybrané pro klastrování vykazují stav 'Active'. Ujistěte se také, že adresa zkratovací smyčky (127.0.0.1) je rovněž aktivní.

  – Použijte příkaz Netstat Option 3 (Práce se stavem připojení TCP/IP) a zobrazte čísla portů (F14). Lokální port 5550 by měl být ve stavu 'Listen'. Tento port musí být otevřen pomocí příkazu STRTCPSVR *INETD doloženým existencí úlohy QTOGINTD (User QTCP) v seznamu aktivních úloh. Je-li v uzlu spuštěno klastrování, lokální port 5551 musí být otevřený a ve stavu '*UDP'. Jestliže klastrování není spuštěno, port 5551 nesmí být otevřený, protože by zabránil úspěšnému spuštění klastrování na daném uzlu.

- Použijte ping. Jestliže chcete spustit klastrový uzel a nelze použít ping, vrátí se interní chyba klastrování (CPFBB46).

- **Použijte makro CLUSTERINFO a zobrazte přehled Služby klastrových prostředků pro aktuálně používané uzly v klastru, uzly v různých skupinách klastrových prostředků a IP adresy klastrů.**

  Zde nalezené nesrovnalosti vám mohou pomoci přesně určit problémové oblasti v případě, že klastr nefunguje tak, jak jste očekávali. Podrobné informace o používání a interpreatci výsledků makra CLUSTERINFO najdete v tématu "Prověření problému s makrem CLUSTERINFO" na stránce 126.

Související pojmy

“Struktura úloh a uživatelské fronty” na stránce 112
Když spravujete klastr, musíte znát strukturu a uživatelské fronty.

Související úlohy

Prohlížení úloh v podsystému

Související odkazy

Příkaz WRKACTJOB (Práce s aktivními úlohami)

Příkaz DSPCLUINF (Zobrazení informací o klastru)

Shromáždění informací pro obnovu klastru

Ke shromáždění informací pro ucelený obraz vašeho klastru můžete použít příkaz WRKCLU (Práce s klastrem) použít ke shrommáždění informací, které dávají kompletní obrázek o vašem klastru. Tyto informace lze použít při řešení problémů.

Příkaz WRKCLU (Práce s klastrem) se používá pro zobrazení a pro práci s uzly a objekty klastru. Když spustíte tento příkaz, zobrazí se obrazovka Práce s klastrem. Kromě zobrazování uzlů v klastru a informací o klastru můžete tento příkaz použít k zobrazení informací o klastru a ke shromáždění dat o klastru.

Chcete-li shromáždit informace o obnově po chybě, proveďte tyto kroky:

1. Na znakově orientované rozhraní napište WRKCLU OPTION(OPTION). Můžete uvést následující volby, abyste uvedli, s jakými informacemi o stavu klastru chcete pracovat.

   ***SELECT**
   Zobrazí nabídku Práce s klastrem.

   ***NODE**
   Zobrazí panel Informace o klastru, což je seznam uzlů v klastru.

   ***CFG** Zobrazí kompletní parametry konfigurace pro klastr. Zajistí také schopnost získat podrobné informace o skupině prostředků klastru.

   ***CRG** Zobrazí seznam skupin klastrových prostředků v klastru.

   ***SERVICE**
   Shromáždí související informace o trasování a ladění pro všechny úlohy služby prostředků klastru v klastru. Tyto informace se zapíší do souboru se členem pro každou úlohu služby prostředku klastru. Tuto volbu použijte pouze tehdy, když to nařídí poskytovatel služeb. Zobrazí se panel náznaku pro příkaz DMPCLUTRC (Výpis trasování klastru).

Prověření problému prostřednictvím příkazu DMPCLUTRC (Výpis trasování klastru)

Příkaz DMPCLUTRC vám pomůže určit a vyřešit problémy s klastrem.

Příkaz DMPCLUTRC (Výpis trasování klastru) vám pomůže určit, zda byla úloha klastru dokončena, nebo jaká úloha se momentálně zpracovává. Příkaz vypisuje trasu související s klastrem a ladicí informace do souboru. Informace se vypisují místně na jednom nebo více uzlech klastru. Příkaz lze použít pro výpis jedné nebo všech úloh služby klastrových prostředků (CRS). Každá úloha CRS, která je vypsaná, má v souboru jeden souborový člen. Jméno souborového členu je jméno úlohy CRS. Aby mohl příkaz vytvořit výstup, klastrování musí být aktivní. Výstup budou mít pouze uzly, které mají aktivní úlohu CRS. Informace, které se vypisují, pocházejí z trasování uživatele a jiné informace se přebírají z klastrových objektů. Množství informací, které se vypisují, je určováno úrovní výpisu. Jednotlivé úrovně výpisu jsou základní informace, chybové informace, informativní informace a slovní informace. Úroveň výpisu určuje, kolik informací se zasílá do souboru. Ve většině případů vás servisní zástupce IBM bude informovat, jakou úroveň máte zadat, a to podle vašich potřeb - avšak úroveň LEVEL(*ERROR) stačí pro většinu situací při odstraňování problémů. Jestliže máte otázky ohledně toho, jaká úroveň je vhodná pro vaši situaci, obraťte se na servisního zástupce IBM.

Vysvětlení výsledků trasování

Můžete analyzovat výsledky trasování, abyste pochopili, co klastrování působí, například to, která úloha klastru způsobuje, že protokol čeká. Výstup pocházející z trasování uživatele bude obsahovat oddělovací řádek, což je vlastně řada rovnítek (=). Počet, kolikrát byl příkaz DMPCLUTRC vydán, bude mít vliv na to, kolik oddělovacích řádek bude v souboru zobrazeno. Ve stejném souboru může být příkaz DMPCLUTRC volán vícekrát. Nejaktuálnější informace obsahuje poslední sada výpisu zásobníků. V některých případech má úloha CRG dvě skupiny. Každá skupina má v souboru oddělenou sekci výpisu.

V následujícím příkladu výsledků trasování výpisu klastru jsou dva uzly (SYSTEM1 a SYSTEM2) v klastru pojmenovaném MYCLUSTER. Obsahuje jednu skupinu CRG nazvanou MYCRG. Oba uzly jsou v doméně obnovy MYCRG. Uživatel vydal CL příkaz STRCRG a procesu dlouho trvá, než vrátí výsledky. Z různých pracovních stanic uživatel zadal DMPCLUTRC CLUSTER(MYCLUSTER) CRG(*ALL) LEVEL(*ERROR) FILE(MYFILE) na rozhraní příkazového řádku.

V tomto příkladu je výstup z příkazu DMPCLUTRC umístěn v souboru nazvaném MYFILE v členu MYCRG. Pro snazší vysvětlování obsahu členu MYCRG byl obsah rozdělen do sekcí. V rámci těchto sekcí jsou čísla zvýrazněna v závorkách, abyste poznali informace, kteér jsou popisovány. Tyto podrobnosti vám mohou pomoci při odstraňování problémy s klastrem.

Poznámka: Svislé výpustky znamenají, že část trasování byla odstraněna a není zobrazena ve výstupu.

Sekce 1 výsledků příkazu DMPCLUTRC

```
User Trace Dump for job 073586/QSYS/MYCRG. Size: 300K, Wrapped 0 times.

--- 08/22/2005 16:43:32 ---
(1a) 00000006:658536 Main thread handle 2
(1b) 00000008:748016 Work thread 1 handle 13
(1b) 00000007:754576 Work thread 2 handle 11
--- 08/22/2005 16:46:04 ---
 00000008:269608 CSTDAMBR 1115: WaitForMsg 4 1005 CPFBB3C
--- 08/22/2005 16:48:17 ---
       00000006:925112
(1c)       DMPCLUTRC  Node SYSTEM1  Group MYCRG
    ================================================
```

První sekce obsahuje čísla a ovladače vláken úlohy klastru. Úlohy klastru mohou mít dvě nebo více vláken. V tomto příkladu je jedno hlavní vlákno (1a), do kterého přichází veškerá práce, a dvě pracovní vlákna (1b). Tato sekce také obsahuje informace o tom, z jakého systému toto trasování pochází, a jaké úlohy klastru se týká (1c).

Sekce 2 výsledků příkazu DMPCLUTRC

```
00000006:925168 Stack Dump For Target Thread: Handle 2 (0x00000002)
       00000006:925192 Stack:
 (2aa) Main Thread Stack MYCRG
       00000006:925256 Stack:  Library    / Program     Module      Stmt    Procedure
       00000006:933432 Stack:  QSYS       / QCSTCRGJOB  CSTCRGJOB   0     : _CXX_PEP__Fv
       00000006:933488 Stack:  QSYS       / QCSTCRGJOB  CSTCRGJOB   46    : main
       00000006:933536 Stack:  QSYS       / QCSTCRGJOB  CSTCRGJOB   65    : completeStartup__FP8CstDAMbr
       00000006:933584 Stack:  QSYS       / QCSTCRGJOB  CSTCRGJOB   26    : mainQueueProcessLoop__FP8Cs
                                  DAMbr
       00000006:933616 Stack:  QSYS       / QCSTCMN     CSTDAMBR    57    : processQueueMsgs__8CstDAMbrF
                                  Q2_8CstDAMbr13CstQueueIndex
       00000006:933664 Stack:  QSYS       / QCSTCMN     CSTDAMBR    53    : processMsg__8CstDAMbrFP6CstM
                                  sg
       00000006:933712 Stack:  QSYS       / QCSTCMN     CSTDAMBR    17    : callFnPtr__8CstDAMbrFPQ2_8Cs
                                  tDAMbr19MsgFunctionPtrEntryP6
       00000006:933744 Stack:  QSYS       / QCSTCRGJOB  CSTCRGJOB   94    : crgDump__FP6CstMsgP8CstDAMbr
       00000006:933792 Stack:  QSYS       / QCSTCMN     CSTACK      95    : CstAckQueryMsg
       00000006:933832 Stack:  QSYS       / QP0ZCPA     QP0ZUDBG    3     : Qp0zDumpTargetStack
00000006:933864 Stack:  QSYS       / QP0ZSCPA    QP0ZSDBG    12    : Qp0zSUDumpTargetStack
       00000006:934016 Stack:  Exception         In Stack Dump Code
```

```
|         00000006:934040 Stack:  thread is likely terminated or no longer running the same code as the
|  captured stack
|         00000006:934080
|   (2a)  Work Thread Index 1  Group MYCRG      Last or current values
|   (2e)  00000006:934112   Request handle 8E3E1002 EE3218A1 824F0004 AC000456
|   (2c)  00000006:934136   SPI name QcstStartClusterResourceGroup
|         00000006:934160 (2g)   POF 10, Completed ack round 1 (2i)
|         00000006:934176  (2o) In waitForJobEnd QDFTJOBD  MYCLUSTER     073590
|         00000006:934216         Node         Ack Status      POF     (2bb) Nack Msg Id
|         00000006:934240 (2n)    SYSTEM1   (2cc) Ready
|         00000006:934272     SYSTEM2   Ack    10 (2k)
|         00000006:934296   Messages
|         00000006:934320 Stack Dump For Target Thread: Handle 13 (0x0000000d)
|         00000006:934344 Stack:  Work Thread 1 Stack MYCRG
|         00000006:934792 Stack:  Library    / Program     Module      Stmt    Procedure
|         00000006:934840 Stack:  QSYS       / QCSTCRGJOB  CSTCRGJOB   9     : workThreadRoutine__FPv
|         00000006:934888 Stack:  QSYS       / QCSTCRGJOB  CSTCRGJOB   28    : workQueueProcessLoop__FP8Cst
|                                    DAMbr
|         00000006:941688 Stack:  QSYS       / QCSTCMN     CSTDAMBR    57    : processQueueMsgs__8CstDAMbrF
|                                    Q2_8CstDAMbr13CstQueueIndex
|         00000006:941696 Stack:  QSYS       / QCSTCMN     CSTDAMBR    33    : processMsg__8CstDAMbrFP6CstM
|                                    sg
|         00000006:941712 Stack:  QSYS       / QCSTCMN     CSTDAMBR    17    : callFnPtr__8CstDAMbrFPQ2_8Cs
|                                    tDAMbr19MsgFunctionPtrEntryP6
|         00000006:941728 Stack:  QSYS       / QCSTCMN     CSTACK      3     : CstStripOffHeaderMsgPart
|         00000006:941736 Stack:  QSYS       / QCSTCMN     CSTDAMBR    53    : processMsg__8CstDAMbrFP6CstM
|                                    sg
|         00000006:941752 Stack:  QSYS       / QCSTCMN     CSTDAMBR    17    : callFnPtr__8CstDAMbrFPQ2_8Cs
|                                    tDAMbr19MsgFunctionPtrEntryP6
|         00000006:970888 Stack:  QSYS       / QCSTCRGS2   CSTCRGSS    39    : startCrg
|         00000006:970912 Stack:  QSYS       / QCSTCRGS2   CSTCRGSS    344   : doMessageProcessing__FP6CstM
|                                    sgP8CstDAMbr
|         00000006:970928 Stack:  QSYS       / QCSTCRGS2   CSTCRGSS    57    : doExitPgmPhase__FP6CstMsgP8C
|                                    stDAMbr
|         00000006:981984 Stack:  QSYS       / QCSTCMN     CSTDAMBR    52    : waitForJobEnd__8CstDAMbrFPA2
|                                    6_ci
|         00000006:982000 Stack:  QSYS       / QCSTCMN     CSTDAMBR    73    : waitForSpecialMsg__8CstDAMbr
|                                    FP17CstSpecialMsgListPA8_ciT3
|         00000006:982016 Stack:  QSYS       / QC2UTIL1    QC2MI3      1     : (2dd) deq
|         00000006:982136 Stack:  Exception         In Stack Dump Code
|         00000006:982136 Stack:  thread is likely terminated or no longer running the same code as the
|  captured stack
|         00000006:982160
|  (2b)Work Thread Index 2  Group MYCRG      Last or current values
|  (2f)00000006:982176   Request handle D9C3C8C3 E2E3F5F2 0003 0000
|  (2cc)00000006:982176   SPI name
|         00000006:982184 (2h)  POF 0, (2d)Completed ack (2j)round 0
|         00000006:982184   In getNextWorkMsg
|         00000006:982208   Node         Ack Status      POF     Nack Msg Id
|  (2l) 00000006:982208     SYSTEM1   Ready
|  (2l) 00000006:982232     SYSTEM2   Ready
|         00000006:982248   Messages
|         00000006:982256 Stack Dump For Target Thread: Handle 11 (0x0000000b)
|         00000006:982256 Stack:  Work Thread 2 Stack MYCRG
|         00000006:982344 Stack:  Library    / Program     Module      Stmt    Procedure
|         00000006:982360 Stack:  QSYS       / QCSTCRGJOB  CSTCRGJOB   9     : workThreadRoutine__FPv
|         00000006:982376 Stack:  QSYS       / QCSTCRGJOB  CSTCRGJOB   28    : workQueueProcessLoop__FP8Cst
|                                    DAMbr
|         00000006:982392 Stack:  QSYS       / QCSTCMN     CSTDAMBR    51    : processQueueMsgs__8CstDAMbrF
|                                    Q2_8CstDAMbr13CstQueueIndex
|  (2m) 00000006:982400 Stack:  QSYS   / QCSTCMN     CSTDAMBR    105   : getNextWorkMsg__8CstDAMbrFv
|         00000006:982416 Stack:  QSYS       / QC2UTIL1    QC2MI3      1     : deq
|         00000006:982480 Stack:  Exception        In Stack Dump Code
|         00000006:982480 Stack:  thread is likely terminated or no longer running the same code as the
|  captured stack
```

Druhá sekce obsahuje zásobníky volání pro každé vlákno, které je součástí úlohy klastru. Největší část hlavního vlákna uvádí příkaz DMPCLUTRC, který práce skončil (2aa). Pracovní vlákna (2a a 2b) obsahují trasovací informace, které pomohou určit, co se děje s úlohou klastru. Tato sekce obsahuje podrobnosti o zásobníku volání, například jméno SPI (2c), dokončené potvrzení (ACK)(2d), ovladač požadavku pro přidružená rozhraní API (2e) nebo ovladač posledního dokončeného požadavku (2f), aktuální bod selhání (POF) (2g a 2h), aktuální kolo potvrzení (ACK) (2i a 2j) a uzly, které byly potvrzeny (ACK) (2k a 2l).

Aktuální *bod selhání (POF)* je interní hodnota, která představuje, kde v kódu je aktuální protokol, a nemusí nutně znamenat, že došlo k selhání. Výraz *Ack* znamená, že uzel úspěšně dokončil tuto část protokolu a čeká na ostatní uzly, aby byly potvrzeny (ACK) nebo nepotvrzeny (Nack). Výraz *Nack* znamená, že uzel nemůže úspěšně dokončit tuto část protokolu a čeká na odpovědi od ostatních uzlů. ID zprávy pro Nack je uvedeno v dalším sloupci (2bb). Toto je stejná zpráva, která byla odeslána původci RIQ. Jestliže uzel selže během protokolu, jeho stav ukáže selhání (Fail) a - v závislosti na protokolu nebo uzlu - může ale nemusí být považován za Nack. Ack stav Inactive znamená, že uzel se nepodílel na protokolu. Hodnota Ready znamená, že uzel ještě neodpověděl. Když je nějaké vlákno v getNextWorkMsg (2m), znamená to, že vlákno čeká na provedení práce.

Přečtěte si jména procedur. Začněte zdola a postupujte nahoru až po zásobník volání. Tento příklad souboru obsahuje deq (2dd) s waitForSpecialMsg, waitForJobEnd a doExitPgmPhase. To ukazuje, že protokol čeká, až ukončovací program skončí, než bude moci pokračovat ve zpracování. Z položky Ack Status (2k) můžeme zjistit, na který uzel protokol čeká. V tomto příkladu čekáme na uzel SYSTEM1 (2n). Kvalifikované jméno úlohy (2e) uvádí úlohu, na kterou systém čeká. Jakmile zjistíte jméno úlohy, můžete pracovat s úlohou na odstraňování příčiny prodlevy. K možným příčinám patří, že úloha stále čeká na frontu úloh, že úloha je spuštěna, ale její zpracování trvá nějaký čas, nebo že úloha čeká na objekt, který je uzamčen.

V tomto příkladu protokol čeká na dokončení ukončovacího programu. Snazší způsob, jak zjistit, zda protokol čeká na ukončovací program nebo na dokončení úlohy logického zapnutí, je podívat se na první sekci, zda je v ní waitForJobEnd (2o). Jméno úlohy, na kterou se čeká, je na stejném řádku. Pak si nemusíte prohlížet zásobníky.

Sekce 3 výsledků příkazu DMPCLUTRC

```
5722SS1 V5R4M0  060210    AS/400 DUMP    073586/QSYS/MYCRG       08/22/05 16:48:18        PAGE    1
DMPSYSOBJ PARAMETERS
(3a)OBJ- MYCRGAIX                        CONTEXT- QTEMP
TYPE- *ALL SUBTYPE-*ALL
OBJECT TYPE-           INDEX                                        *CRGM
NAME-        MYCRGAIX                        TYPE-          0E   SUBTYPE-          A5
LIBRARY-     QTEMP     006B8A19B00C9E807000  TYPE-          04   SUBTYPE-          C1
CREATION-    08/22/05  16:43:32              SIZE-          0000007000
ATTRIBUTES-          0000                    ADDRESS-       C7FE286F04 000000
.
.
.
.POINTERS-
  NONE
OIR DATA-
  NONE
END OF DUMP
                                  * * * * *  E N D  O F  L I S T I N G  * * * * *
```

Třetí uvedená sekce je vnitřní objekt, který obsahuje informace o úloze klastru. V tomto příkladu je to vnitřní index nazvaný MYCRGAIX (3a). Zde uvedené informace se čtou mnohem snadněji než vyše uvedená sekce 2.

Sekce 4 výsledků příkazu DMPCLUTRC

```
5722SS1 V5R4M0  060210   AS/400 DUMP     073586/QSYS/USER     08/22/05 16:48:18        PAGE    1
DMPSYSOBJ PARAMETERS
(4a)OBJ- MYCRGTQ                         CONTEXT- QTEMP
TYPE- 0A   SUBTYPE-EF
OBJECT TYPE-           QUEUE                                        *QTQ
NAME-        MYCRGTQ                         TYPE-          0A   SUBTYPE-          EF
LIBRARY-     QTEMP     006B8A19B00C9E807000  TYPE-          04   SUBTYPE-          C1
CREATION-    08/22/05  16:43:32              SIZE-          000002C000
```

```
ATTRIBUTES-          0000                    ADDRESS-       CC6765CAA2 000000
QUEUE ATTRIBUTES-
.
.
.
.POINTERS-
  000010   SPP 1A EF MYCRG     QSYS      073586                                                  00002160  0000
  000020   SPP 1A EF MYCRG     QSYS      073586                                                  00001540  8000
  000030   SPP 1A EF MYCRG     QSYS      073586                                                  000016E0  0000
  000040   SPP 1A EF MYCRG     QSYS      073586                                                  00001690  0000
  000050   SPP 1A EF MYCRG     QSYS      073586                                                  000016A0  0000
  000070   SPP 1A EF MYCRG     QSYS      073586                                                  00002160  0000
OIR DATA-
  NONE
END OF DUMP
                                    * * * * *  E N D  O F  L I S T I N G  * * * * *
```

Čtvrté uvedené sekci se říká trasovací fronta (4a). V této situaci se jmenuje MYCRGTQ. Obsahuje informace o tom, který klastr spustil tuto úlohu a jak každá úloha reagovala na požadavek.

Poznámka: Zde není každá zpráva popsána plně.

Sekce 5 výsledků příkazu DMPCLUTRC

```
5722SS1 V5R4M0  060210  AS/400 DUMP  073586/QSYS/MYCRG     08/22/05 16:48:18        PAGE    1
DMPSYSOBJ PARAMETERS
(5a) OBJ- MYCRG                          CONTEXT- QUSRSYS
TYPE- 19   SUBTYPE-2C
OBJECT TYPE-           SPACE                                        *CRG
NAME-        MYCRG                          TYPE-          19   SUBTYPE-          2C
LIBRARY-     QUSRSYS                        TYPE-          04   SUBTYPE-          01
CREATION-    08/17/05  07:16:40             SIZE-          0000002000
OWNER-       MYCLUSTER                          TYPE-          08   SUBTYPE-          01
ATTRIBUTES-          0800                    ADDRESS-       1EC687A1F3 000000
SPACE ATTRIBUTES-
.
.
.
END OF DUMP
                                    * * * * *  E N D  O F  L I S T I N G  * * * * *
```

Pátá část obsahuje informace o objektu CRG (5a).

Prověření problému s makrem CLUSTERINFO

Makro CLUSTERINFO zobrazuje informace, které obsahují služby klastrových prostředků o uzlech, skupinách CRG a aktivních IP adresách.

Makro CLUSTERINFO vytváří snímek informací o aktuálním klastru. Příkaz prochází klastrovanými objekty a vytváří popis klastru na lokálním uzlu. Makro CLUSTERINFO poskytuje ″letový záznam″ pro různé objekty klastru a může pomoci určit zdroj problému v klastru. Chcete-li přistoupit k makru CLUSTERINFO, proveďte tyto kroky:

1. Do znakově orientovaného rozhraní zadejte STRSST.
2. Přihlaste se s uživatelským profilem Service Tools.
3. Na obrazovce Start Service Tool vyberte volbu 1 (Start a service tool).
4. Vyberte volbu 4 (Display/Alter/Dump).
5. Vyberte volbu 2 (Dump to Printer).
6. Vyberte volbu 2 (Licensed Internal Code (LIC) data).
7. Vyberte volbu 14 (Advanced Analysis).
8. Před volbu makra CLUSTERINFO napište 1. Stiskněte klávesu Enter.

Jakmile se makro CLUSTERINFO zobrazí, zobrazte pomocí volby -H nápovědu ke všem volbám dostupným pro toto makro. Následující diagram použití popisuje všechny volby, které jsou k dispozici pro makro CLUSTERINFO:

Tabulka 21. Volby pro makro CLUSTERINFO

Volba	Popis
-H	Zobrazí obrazovku nápovědy pro volby.
-A	Zobrazí veškeré informace.
-FR	Zobrazí záznamy ″letového záznamu″.
-HB	Zobrazí informace o pulsu (heartbeat).
-PERF	Zobrazí čítače výkonu.
-Q	Zobrazí stav fronty odchozích zpráv.
-G	Zruší potlačení zobrazování všech skupin CC vysílání.
-T	Zobrazí ladicí parametry.
-M	Zobrazí matice pro všechny DA.
-DP	Zobrazí informace o datových portech.

Jak rozumět výsledkům makra CLUSTERINFO

V tomto příkladu byla zvolena volba -A, takže se zobrazují všechna pole. Hlavními nástroji pro ladění jsou ″letové záznamy″. Všimněte si, že tyto letové záznamy se vymažou, když je klastrový uzel ukončen nebo vymazán. Pro analýzu problému musí být makro CLUSTERINFO spuštěno před ukončením nebo výmazem klastru. V některých případech mohou být ″letové záznamy″ zapsány do protokolu vlog, když je klastr vymazán nebo ukončen. ″Letové záznamníky″ ukládají různé události, které mají vliv na strukturu a výkon klastru. Podrobné vysvětlení dat v letovém záznamníku je nad rozsah této publikace.

| **Poznámka:** Svislé výpustky znamenají, že část výsledků byla odstraněna a není zobrazena ve výstupu.

Sekce 1 výsledků makra CLUSTERINFO

```
DISPLAY/ALTER/DUMP    CLUSTERINFO -NEW2     08/23/05  13:36:37   PAGE      1
Running macro: CLUSTERINFO               -A
Use -H for command information
Cluster Name   : MYCLUSTER
Local Node Name: SYSTEM1
CC/CTS Version : 5
Macro Timestamp: 08/23/05 13:36:37.079
```

Sekce 1 obsahuje generické informace o klastru, například jméno klastru, verzi klastru a časové označení, kdy byla zpráva vygenerována. Jméno klastru v tomto příkladu je MYCLUSTER a jméno lokálního uzlu je SYSTEM1.

Sekce 2 výsledků makra CLUSTERINFO

```
Cluster Object Addresses
CstcClusterServices Address: DBF08681C9161580
Cluster Address            : FC5B04B0D4001000
Cluster Task Address       : B00010000E932000
Cluster Task Q Address     : DBF08681C9169A00
Clue Group Services Address: CDAB6D0339001000
CC Services Address        : FC5B04B0D4008000
```

Sekce 2 uvádí ukazatele na umístění klíčových objektů klastru.

Sekce 3 výsledků makra CLUSTERINFO

```
Message Statistics
Number of non-fragmented messages: 250
Number of fragmented messages    : 1
Number of fragments              : 7
Number of acks                   : 148
```

Sekce 3 obsahuje statistiky pro rychlé posílání zpráv pro klastr, například počet fragmentů a počet potvrzení (acks).

Sekce 4 výsledků makra CLUSTERINFO

```
Node Map
Node ID : SYSTEM1
 GenesisSubnetId : 9.5.251.0
 CCNode *   : FC5B04B0D4007000
 CCSrvNode *: FC5B04B0D404F000
 Adapter 1  : 9.5.251.46 Primary
  Status    : 0x01 Reachable
  Line Type : 0x09 Ethernet
Node ID : SYSTEM2
 GenesisSubnetId : 9.5.251.0
 CCNode *   : FC5B04B0D4060000
 CCSrvNode *: FC5B04B0D4061000
 Adapter 1  : 9.5.251.47 Primary
  Status    : 0x01 Reachable
  Line Type : 0x09 Ethernet
```

Sekce 4 uvádí všechny momentálně aktivní klastrové uzly na mapě uzlů. V tomto příkladu jsou dva aktivní uzly, SYSTEM1 a SYSTEM2.

Sekce 5 výsledků makra CLUSTERINFO

```
Subnet Map
Subnet ID: 127.0.0.0
 CCSubnet *            : FC5B04B0D4006000
 CCSrvSubnet*          : FC5B04B0D400C000
 Retries               : 0
 Msg Timeouts          : 0
 Bad Msg Counter       : 0
 Failed Default Address: 0
Subnet ID: 226.5.5.5
 CCSubnet *            : FC5B04B0D405B100
 CCSrvSubnet*          : FC5B04B0D405C000
 Retries               : 0
 Msg Timeouts          : 0
 Bad Msg Counter       : 0
 Failed Default Address: 0
```

Sekce 5 obsahuje seznam všech objektů podsítě, které jsou v klastru.

Sekce 6 výsledků makra CLUSTERINFO

```
Group Map
Group ID: 0x0000000000000001
 Name  : CTS
 CCGroup *   : FC5B04B0D405FF00
 CCSrvGroup *: FC5B04B0D4064B00
 Member Nodes
  SYSTEM1
  SYSTEM2
Group ID: 0x0000000000000002
 Name  : CTS
 CCGroup *   : FC5B04B0D4055100
 CCSrvGroup *: FC5B04B0D4055200
 Member Nodes
  SYSTEM1
```

```
  SYSTEM2
.
.
.
```

Sekce 6 uvádí seznam všech aktuálních skupin klastrů. Každé skupině se říká skupina distribuované činnosti. Tyto skupiny se používají pro komunikaci mezi skupinami na každém aktivním uzlu v klastru. Největší část činnosti se skupinami souvisí s licencovaným interním kódem (LIC). Lze je identifiovat podle jména skupiny CTS a BADA. Rovněž se zobrazí skupina pro CCTL (úloha QCSTCTL v operačním systému), CRGM (úloha QCSTCRGM v operačním systému) a úloha každé skupiny klastrových prostředků (CRG). Skupiny pro úlohy CRG nebudou mít jméno skupiny. Každá skupina má členské uzly. Členské uzly jsou uzly, které komunikují spolu v této skupině.

Sekce 7 výsledků makra CLUSTERINFO

```
Partition Map
Partition Map is empty
```

Sekce 7 obsahuje seznam všech uzlů v seznamu logických částí SLIC.

Poznámka: Toto není stejná koncepce jako uzly XPF rozdělené na logické části.

Sekce 8 výsledků makra CLUSTERINFO

```
CTS Client List
CTS Client List is empty
```

Sekce 8 obsahuje seznam všech registrovaných klientů klastru, například datových portů.

Sekce 9 výsledků makra CLUSTERINFO

```
Flight Recorder        : CSTCSVFR
Flight Recorder Address: DBF08681C9161620
```

Sekce 9 obsahuje letový záznamník služeb klastru (CSTCSVFR), který zůstává v systému až do IPL.

Sekce 10 výsledků makra CLUSTERINFO

```
Message Statistics
Number of non-fragmented messages: 250
Number of fragmented messages    : 1
Number of fragments              : 7
Number of acks                   : 148
Time Stamp: 08/18/05 14:00:15.329
Trace Point: 0x0010 CstcClusterServicesTracePtCreatedFlightRecorder
C3D9C5C1E3C5C6D9          <CREATEFR>
Time Stamp: 08/22/05 16:43:28.912
Trace Point: 0x0020 CstcClusterServicesTracePtCreatedClusterObject
D4D6D9C5E8404040 4040C5F8D3770500  <MYCLUSTER   E8L...>
1000               <..        >
Time Stamp: 08/23/05 13:33:40.935
Trace Point: 0x0030 CstcClusterServicesTracePtDeletedClusterObject
D4D6D9C5E8404040 404040E2E3       <MYCLUSTER      ST  >
Time Stamp: 08/23/05 13:33:41.204
Trace Point: 0x0030 CstcClusterServicesTracePtDeletedClusterObject
C3D4D7E3                          <CMPT            >
Time Stamp: 08/23/05 13:33:55.122
Trace Point: 0x0020 CstcClusterServicesTracePtCreatedClusterObject
D4D6D9C5E8404040 4040FC5B04B0D400 <MYCLUSTER   ....M.>
1000              <..         >
```

Sekce 10 obsahuje letový záznamník pro CSTCCCFR. Tento klastrový letový záznamník zůstává na systému, dokud nebude klastrování na tomto uzlu ukončeno.

Sekce 11 výsledků makra CLUSTERINFO

```
Flight Recorder        : CSTCCLFR
Flight Recorder Address: FC5B04B0D4001E80
--------------------------------------------------------------------------------
Time Stamp: 08/23/05 13:33:54.944
Trace Point: 0x1010 CstcClusterTracePtCreatedSubnetObject
7F000000FC5B04B0 D4006000         <........M.-.    >
Time Stamp: 08/23/05 13:33:55.062
Trace Point: 0x1000 CstcClusterTracePtCreatedNodeObject
C3E2E3D9D9C3C8C3 E2E3F5F2FC5B04B0 <CSTRSYSTEM1....>
D4007000                          <M...            >
Time Stamp: 08/23/05 13:33:55.122
Trace Point: 0x1020 CstcClusterTracePtCreatedMCGroupObject
0000000000000001 00000000D9C3C8C3 <............RCHC>
E2E3F5F2                          <ST52            >
.
.
.
```

Sekce 11 obsahuje klastrový komunikační letový záznamník (CSTECLFR). Tento klastrový letový záznamník zůstává na systému, dokud nebude klastrování na tomto uzlu ukončeno.

Sekce 12 výsledků makra CLUSTERINFO

```
Flight Recorder        : CSTCCCFR
Flight Recorder Address: FC5B04B0D4006380
--------------------------------------------------------------------------------
Time Stamp: 08/23/05 13:33:55.080
Trace Point: 0x3000 CstcCCScamTracePtScamOpen
FC5B04B0D400E480 0000000000000000 <....M.U.........>
Time Stamp: 08/23/05 13:33:55.097
Trace Point: 0x3010 CstcCCScamTracePtScamBind
FC5B04B0D400E480 0000000000000000 <....M.U.........>
Time Stamp: 08/23/05 13:33:55.100
Trace Point: 0x3000 CstcCCScamTracePtScamOpen
FC5B04B0D400E480 0000000000000000 <....M.U.........>
D6E4E3                            <OUT             >
Time Stamp: 08/23/05 13:33:55.100
Trace Point: 0x3010 CstcCCScamTracePtScamBind
FC5B04B0D400E480 0000000000000000 <....M.U.........>
.
.
.
```

Sekce 12 obsahuje letový záznamník klíčů (CSTCCCFR), který zůstává na systému, dokud nebude klastrování na tomto uzlu ukončeno.

Sekce 13 výsledků makra CLUSTERINFO

```
Time Stamp: 08/23/05 13:33:55.201
C3A2A385C7E27A7A C3A2A385C7E24082 <CsteGS::CsteGS b>
85878995A2                        <egins           >
Time Stamp: 08/23/05 13:33:55.201
C3A2A385C4C14083 9695A2A399A483A3 <CsteDA construct>
85847A40C2C1C4C1 404040404040     <ed: BADA        >
Time Stamp: 08/23/05 13:33:55.201
C3A2A385C7E27A7A C3A2A385C7E24081 <CsteGS::CsteGS a>
8484408281848140 A39640C4C16D9389 <dd bada to DA_li>
A2A3                              <st              >
.
.
.
```

Sekce 13 zobrazuje obsah odeslaných front a aktivní fronty zpráv. Jestliže tato sekce není prázdná po delší dobu, znamená to, že v klastru je problém.

Sekce 14 výsledků makra CLUSTERINFO

```
Flight Recorder        : CSTECLF2
Flight Recorder Address: CDAB6D0339001300
--------------------------------------------------------------------------------
Time Stamp: 08/23/05 13:33:55.201
C3A2A385C4C17A7A C3A2A385C4C16B40 <CsteDA::CsteDA, >
83998581A385D4C3 C79996A49740C9C4 <createMCGroup ID>
407E40F0A7F1F5                    < = 0x15         >
Time Stamp: 08/23/05 13:33:55.209
C3A2A385E2C3D985 977A7A84859389A5 <CsteSCRep::deliv>
85994094A287E3A8 97857EF0A7F140A2 <er msgType=0x1 s>
A482E3A897857EF0 A7F240C4C17EC2C1 <ubType=0x2 DA=BA>
C4C1404040404040                  <DA              >
Time Stamp: 08/23/05 13:33:55.209
C3A2A385C4C17A7A A58985A66B409985 <CsteDA::view, re>
9496A585D4C3C799 96A497D485948285 <moveMCGroupMembe>
99A240C9C4407E40 F0A7F1F540969384 <rs ID = 0x15 old>
6D959684856D9389 A2A340A289A98540 <_node_list size >
7E40F0A7F0                        <= 0x0           >
.
.
.
```

Sekce 14 obsahuje informace o letovém záznamníku.

Sekce 15 výsledků makra CLUSTERINFO

```
Message Queues
Send Queues:
Send Queue: 00 Messages: 00 MessageQueue*: FC5B04B0D400BF80
Send Queue: 01 Messages: 00 MessageQueue*: FC5B04B0D400DF80
Send Queue: 02 Messages: 00 MessageQueue*: FC5B04B0D400E600
Send Queue: 03 Messages: 00 MessageQueue*: FC5B04B0D400E680
Send Queue: 04 Messages: 00 MessageQueue*: FC5B04B0D400E700
Send Queue: 05 Messages: 00 MessageQueue*: FC5B04B0D400E780
Send Queue: 06 Messages: 00 MessageQueue*: FC5B04B0D400E800
Send Queue: 07 Messages: 00 MessageQueue*: FC5B04B0D400E880
Send Queue: 08 Messages: 00 MessageQueue*: FC5B04B0D400E900
Send Queue: 09 Messages: 00 MessageQueue*: FC5B04B0D400E980
Send Queue: 10 Messages: 00 MessageQueue*: FC5B04B0D400EA00
Send Queue: 11 Messages: 00 MessageQueue*: FC5B04B0D400EA80
Send Queue: 12 Messages: 00 MessageQueue*: FC5B04B0D400EB00
Send Queue: 13 Messages: 00 MessageQueue*: FC5B04B0D400EB80
Send Queue: 14 Messages: 00 MessageQueue*: FC5B04B0D400EC00
Send Queue: 15 Messages: 00 MessageQueue*: FC5B04B0D400EC80
Send Queue: 16 Messages: 00 MessageQueue*: FC5B04B0D400ED00
Send Queue: 17 Messages: 00 MessageQueue*: FC5B04B0D400ED80
Send Queue: 18 Messages: 00 MessageQueue*: FC5B04B0D400EE00
Send Queue: 19 Messages: 00 MessageQueue*: FC5B04B0D400EE80
Active Message Queues:
Active Message Queue: 00 Messages: 00 MessageQueue*: FC5B04B0D4008570
Active Message Queue: 01 Messages: 00 MessageQueue*: FC5B04B0D4008640
Active Message Queue: 02 Messages: 00 MessageQueue*: FC5B04B0D4008710
Active Message Queue: 03 Messages: 00 MessageQueue*: FC5B04B0D40087E0
Active Message Queue: 04 Messages: 00 MessageQueue*: FC5B04B0D40088B0
Active Message Queue: 05 Messages: 00 MessageQueue*: FC5B04B0D4008980
Active Message Queue: 06 Messages: 00 MessageQueue*: FC5B04B0D4008A50
Active Message Queue: 07 Messages: 00 MessageQueue*: FC5B04B0D4008B20
Active Message Queue: 08 Messages: 00 MessageQueue*: FC5B04B0D4008BF0
Active Message Queue: 09 Messages: 00 MessageQueue*: FC5B04B0D4008CC0
Active Message Queue: 10 Messages: 00 MessageQueue*: FC5B04B0D4008D90
Active Message Queue: 11 Messages: 00 MessageQueue*: FC5B04B0D4008E60
Active Message Queue: 12 Messages: 00 MessageQueue*: FC5B04B0D4008F30
Active Message Queue: 13 Messages: 00 MessageQueue*: FC5B04B0D4009000
Active Message Queue: 14 Messages: 00 MessageQueue*: FC5B04B0D40090D0
Active Message Queue: 15 Messages: 00 MessageQueue*: FC5B04B0D40091A0
```

```
Active Message Queue: 16 Messages: 00 MessageQueue*: FC5B04B0D4009270
Active Message Queue: 17 Messages: 00 MessageQueue*: FC5B04B0D4009340
Active Message Queue: 18 Messages: 00 MessageQueue*: FC5B04B0D4009410
Active Message Queue: 19 Messages: 00 MessageQueue*: FC5B04B0D40094E0
------------------------------------------------------------------------------
Tuning Values
cstcRcvSendTimerRatio        : 2 Default: 2
cstcMcastRelayTimerRatio     : 8 Default: 8
cstcMcastRelayHBTimerRatio   : 4 Default: 4
cstcHeartbeatBaseTimer       : 12288000000 Default: 12288000000
cstcHeartbeatBasePrecision   : 4096000000 Default: 4096000000
cstcRetryPrecision           : 4096000000 Default: 4096000000
cstcRetryTimerVal            : 4096000000 Default: 4096000000
cstcCDATBaseTimer            : 491520000000 Default: 491520000000
cstcCDATBasePrecision        : 40960000000 Default: 40960000000
cstcRecoveryBaseTimer        : 3686400000000 Default: 3686400000000
cstcRecoveryBasePrecision    : 491520000000 Default: 491520000000
cstcMaxRetryTime             : 32768000000 Default: 32768000000
cstcCCFragmentSize           : 1464 Default: 1464
cstcCCSendQOverflow          : 1024 Default: 1024
cstcBadMsgCtrThreshold       : 3 Default: 3
cstcUnreachableHBAckThreshold: 1 Default: 1
cstcReachableHBAckThreshold  : 3 Default: 3
cstcUnreachableHBThreshold   : 4 Default: 4
cstcReachableHBThreshold     : 4 Default: 4
cstcMaxHBThreshold           : 16 Default: 16
cstcDisableMsgTimer          : 0 Default: 0
cstcRepeatAckThreshold       : 10 Default: 10
     DISPLAY/ALTER/DUMP    CLUSTERINFO -NEW2   08/23/05  13:36:37   PAGE     87
cstcDelayedAckTimer          : 409600000 Default: 409600000
cstcDelayedAckPrecision      : 40960000 Default: 40960000
cstcCCSendWindow             : 2 Default: 2
cstcCCEnableMcast            : 1 Default: 1
cstcCCPerfClass              : 2 Default: 2
------------------------------------------------------------------------------
****** END OF DUMP ******
```

Sekce 15 obsahuje ladicí hodnoty. Ladicí hodnoty jsou ekvivalentní k výkonu klastru a k informacím o konfiguraci, které jsou popsány v tématu Rozhraní QcstRetrieveCRSInfo (Retrieve Cluster Resource Services Information) API. Sekce 14 obsahuje aktuální hodnotu a předvolbu pro tato pole.

Běžné problémy s klastry

Zde jsou popsány některé z nejčastějších problémů, ke kterým dochází v klastru, a způsoby, jak se těchto problémů vyvarovat a jak provést nápravu chyb.

Níže uvedeným obvyklým problémům lze snadno zabránit nebo jsou snadno opravitelné.

Nemůžete spustit nebo restartovat klastrový uzel.

K této situace obvykle dochází kvůli problému s komunikačním prostředím. Chcete-li této situaci zabránit, ujistěte se, že máte správně nastaveny atributy sítě, včetně adres zkratovací smyčky, nastavení serveru INETD, atributu ALWADDCLU a IP adres pro klastrové komunikace.

- Pokoušíte-li se spustit vzdálený uzel, musí být správně nastaven atribut sítě ALWADDCLU. Měl by být nastaven buď na *ANY, nebo *RQSAUT v závislosti na vašem prostředí.
- Vybraná IP adresa použitá pro lokální klastrování a na vzdáleném uzlu musí být ve stavu *Active*.
- Lokální adresa zkratovací smyčky (127.0.0.1) a na cílovém uzlu musí být také aktivní.
- Lokální a vzdálené uzly musejí být schopny provádět testování spojení pomocí IP adres, které mají být použity pro klastrování, aby se zajistilo, že směrování po síti bude aktivní.
- Server INETD musí být aktivní na cílovém uzlu. Když je server INETD aktivní, port 5550 na cílovém uzlu by měl být ve stavu *Listen*. Informace o spuštění serveru INETD najdete v tématu Server INETD.

- Před pokusem o spuštění uzlu nesmí být na spouštěném uzlu otevřen port 5551, protože by zabránil úspěšnému spuštění klastrování na daném uzlu.

Skončíte s několika rozpojenými jednouzlovými klastry.

K tomu může dojít v případě, kdy nemůže spouštěný uzel komunikovat s ostatními uzly v klastru. Zkontrolujte komunikační cesty.

Odezva z ukončovacích programů je pomalá.

Obvyklou příčinou této situace je nesprávné nastavení pro popis úlohy používané ukončovacím programem. Parametr MAXACT může být nastaven příliš nízko, takže například v určitém okamžiku může být aktivní pouze jedna instance ukončovacího programu. Doporučujeme, aby byl tento parametr nastaven na *NOMAX.

Celkový výkon se zdát být příliš pomalý.

Pro tento příznak existuje několik obvyklých příčin.

- Nejpravděpodobnější příčinou je silný komunikační provoz na sdílených komunikačních linkách.
- Další možou příčinou je neshoda mezi komunikačním prostředím a parametry ladění klastrových zpráv. Můžete použít rozhraní QcstRetrieveCRSInfo (Retrieve Cluster Resource Services Information) API a prohlédnout si aktuální nastavení parametrů ladění. Pomocí rozhraní QcstChgClusterResourceServices (Change Cluster Resource Services) API můžete nastavení změnit. Výkon klastru může být snížen při předvoleném nastavení parametrů ladění klastru v případě, že používáte starý hardwarový adaptér. Typy hardwarového adaptéru zahrnuté do definice *old* zahrnují 2617, 2618, 2619, 2626 a 2665. V takovém případě je požadováno nastavení parametru ladění *Performance class* na *Normal*.
- Další obvyklou příčinou této situace jsou problémy se skupinami výběrového vysílání IP. Jsou-li adresy primárního klastru (první adresa zadaná pro daný uzel při vytváření klastru nebo přidávání uzlu) pro několik uzlů umístěny v běžných sítích LAN, klastr použije schopnosti výběrového vysílání IP. Pomocí příkazu NETSTAT zajistěte, aby adresy primárního klastru zobrazovaly hostitelskou skupinu výběrového vysílání 226.5.5.5. Tu je možné si prohlédnout pomocí volby *Display multicast group* pro danou adresu. Pokud skupina výběrového vysílání neexistuje, ověřte pomocí API QcstRetrieveCRSInfo (Načtení informací o Službách klastrových prostředků), zda je pro parametr ladění klastru *Enable multicast* stále nastaveno předvolené nastavení TRUE.
- Jsou-li všechny klastrové uzly v lokální síti LAN nebo mají-li schopnosti směrování, které mohou prostřednictvím předepsaných cest k síti zpracovávat pakety MTU (Maximum Transmission Unit) vetší než 1.464 bajtů, mohou být přenosy velkých klastrových zpráv (větší než 1.536 kB) značně urychleny zvětšením hodnoty parametru ladění klastru *Message fragment size* tak, aby lépe vyhovala přenosové cestě MTU.

Nemůžete použít žádnou funkci nového vydání.

Pokusíte-li se použít funkci nového vydání a zobrazí se chybová zpráva CPFBB70, je vaše aktuální klastrová verze stále nastavena na předchozí úroveň vydání. Musíte aktualizovat všechny klastrové uzly na úroveň nového vydání a potom použít rozhraní pro úpravu klastrové verze k nastavení aktuální klastrové verze na novou úroveň. Další informace najdete v tématu Úprava klastrové verze klastru.

Nemůžete přidat uzel k doméně zařízení nebo nemáte přístup k rozhraní správy klastrů pomocí produktu iSeries Navigator.

Chcete-li mít přístup k rozraní pro správu klastru pomocí produktu iSeries Navigator nebo chcete-li používat přepínatelná zařízení, musíte mít v systému instalovanou volbu 41 i5/OS, HA Switchable Resources. Musíte mít také platný licenční klíč pro tuto volbu.

Aplikujete klastrové PTF a zdá se, že toto PTF nefunguje.

Po aplikaci PTF byste neměli opomenout provést následující úkoly:

1. Ukončení klastru

2. Odhlášení a pak přihlášení

   Dokud nezničíte aktivační skupinu, je v ní stále aktivní starý program. Všechny klastrové kódy (dokonce i klastrové API) jsou spouštěny v předvolené aktivační skupině.

3. Spuštění klastru

   Většina klastrových oprav PTF vyžaduje, aby se opravy PTF aktivovala tím, že klastrování v uzlu ukončíte a restartujete.

V protokolu úlohy ukončovacího programu se objeví CEE0200.

Při této chybové zprávě je ″from module″ QLEPM a ″from procedure″ Q_LE_leBdyPeilog. Všechny programy, které vyvolává ukončovací program, musejí být spouštěny buď v *CALLER nebo v pojmenované aktivační skupině. Chcete-li opravit tento stav, musíte změnit ukončovací program nebo program hlásící chybu.

V protokolu úlohy služby klastrových prostředků se objeví CPD000D následované CPF0001.

Když obdržíte tuto chybovou zprávu, ujistěte se, že systémová hodnota QMLTTHDACN je nastavena na 1 nebo 2.

Zdá se, že se klastr zastavil.

Ujistěte se, že se neprovedly ukončovací programy skupiny klastrových prostředků. Chcete-li zkontrolovat ukončovací program, použijte příkaz WRKACTJOB (Práce s aktivními úlohami) a potom si prohlédněte sloupec Funkce, zda v něm naleznete PGM-QCSTCRGEXT.

Související pojmy

“Povolení přidání uzlu do klastru” na stránce 88
Chcete-li přidat uzel do klastru, musíte nejdříve nastavit hodnotu atributu sítě ALWADDCLU (Povolit přidání do klastru).

“Výkon klastru” na stránce 109
Změny prováděné v klastru mohou mít vliv na režii potřebnou ke správě klastru.

“Klastrová verze” na stránce 12
Klastrová verze představuje úroveň funkce dostupné v klastru.

“Správa klastrů pomocí produktu iSeries Navigator” na stránce 72
IBM nabízí rozhraní pro správu klastru, které je dostupné pomocí produktu iSeries Navigator a přístupné pomocí Volby 41 (i5/OS - HA Switchable Resources).

Související úlohy

“Úprava verze klastru” na stránce 101
Verze klastru definuje úroveň, na které spolu všechny klastrové uzly aktivně komunikují.

Chyby logických částí

Některé podmínky klastru lze snadno opravit. Pokud došlo k rozdělení klastru, můžete se naučit, jak provést nápravu chyb. Toto téma také popisuje, jak se vyvarovat rozdělení klastru, a udává příklad způsobu opětovného sloučení jednotlivých částí.

K rozdělení klastru dojde vždy, když se ztratí komunikace mezi jedním nebo více uzly v klastru a není možné potvrdit selhání ztracených uzlů. Tato situace by neměla být zaměňována s rozdělením v prostředí logických částí (LPAR).

Obdržíte-li v protokolu historie (QHST) nebo protokolu úlohy QCSTCTL chybovou zprávu CPFBB20, došlo k rozdělení klastru a vy musíte vědět, jak provést operaci obnovy. Níže uvedené příklady ukazují rozdělení klastru, které zahrnuje klastr tvořený čtyřmi uzly: A, B, C a D. Příklad ukazuje ztrátu komunikace mezi klastrovými uzly, což má za následek rozdělení klastru na dvě části. Před rozdělením klastru existovaly čtyři skupiny klastrových prostředků, které mohly být libovolného druhu. Označili jsme je CRG A, CRG B, CRG C a CRG D. Příklad ukazuje doménu obnovy každé skupiny klastrových prostředků.

Tabulka 22. Příklad domény obnovy při logickém rozdělení klastru

Uzel A	Uzel B	x	Uzel C	Uzel D
CRG A (záložní 1)	CRG A (primární)			
	CRG B (primární)		CRG B (záložní 1)	
	CRG C (primární)		CRG C (záložní 1)	CRG C (záložní 2)
CRG D (záložní 2)	CRG D (primární)		CRG D (záložní 1)	
Část 1			**Část 2**	

K rozdělení klastru může dojít v případě, že MTU (maximum transmission unit) v libovolném bodu komunikační cesty je nižší než parametr ladění klastrové komunikace, Message Fragment Size. MTU pro určitou IP adresu klastru lze ověřit v předmětném uzlu pomocí příkazu WRKTCPSTS (Práce se stavem sítě TCP/IP). MTU musí být také změněna na každé kroku podél celé komunikační cesty. Pokud je MTU nižší než velikost fragmentu zpráv, zvyšte MTU cesty nebo snižte velikost fragmentu zpráv. Můžete použít API QcstRetrieveCRSInfo (Načtení informací o Službách klastrových prostředků) a prohlédnout si aktuální nastavení parametrů ladění. Pomocí API QcstChgClusterResourceServices (Změna Služeb klastrových prostředků) můžete nastavení změnit.

Jakmile odstraníte příčinu stavu rozdělení klastru, klastr detekuje znovu obnovenou komunikační linku a vydá zprávu CPFBB21 buď v protokolu historie (QHST), nebo v protokolu úlohy QCSTCTL. Tím informuje operátora, že by klastr po rozdělení obnoven. Vezměte na vědomí, že jakmile byl odstraněn stav rozdělení klastru, můžete trvat několik minut, než se klastr opět sloučí.

Související pojmy

"Část klastru" na stránce 27
Část klastru je podmnožina aktivních uzlů klastru, která vzniká rozdělením klastru v důsledku selhání komunikace. Členové části klastru mezi sebou udržují navzájem spojení.

"Zabránění rozdělení klastru" na stránce 83
Typickému rozdělení klastru souvisejícího se sítí se lze nejlépe vyhnout tím, že nakonfigurujete redundantní komunikační cesty mezi všemi uzly v klastru.

"Sloučení" na stránce 22
Operace *sloučení* je podobná operaci opětovné připojení, až na to, že k ní dochází tehdy, když uzly, které jsou rozdělené, začínají opět komunikovat.

"Příklad: Selhání" na stránce 18
K přepnutí při selhání obvykle dojde následkem selhání uzlu, ale existují také další důvody, které mohou způsobit přepnutí při selhání.

Určení primárních a sekundárních částí klastru

K tomu, abyste mohli určit typy akcí skupiny klastrových prostředků, které můžete provádět v rámci části klastru, potřebujete vědět, zda je část primární nebo sekundární částí klastru. Když je detekována nějaká část, je označena jako primární nebo sekundární pro skupinu klastrových prostředků definovanou v klastru.

U modelu primární-záložní obsahuje primární část uzel, který má aktuální úlohu uzlu primární. Všechny ostatní části jsou sekundární. Primární oblasi nemůže být stejná pro všechny skupiny klastrových prostředků.

Peer model má následující pravidla pro logické části:

- Jestliže uzly domény obnovy jsou plně obsaženy v jedné části, bude tato část primární.
- Jestliže uzly domény obnovy přesahují přes jednu část, nebude primární žádná část. Obě části budou sekundárními částmi.
- Jestliže skupina klastrových prostředků je aktivní a v dané části nejsou žádné peer uzly, skupina prostředků klastru bude v této části ukončena.
- Provozní změny jsou přípustné v sekundární části, pokud jsou dodržena omezení pro provozní změny.
- V sekundární části nejsou přípustné žádné změny konfigurace.

Omezení pro každé API skupiny klastrových prostředků jsou:

Tabulka 23. Omezení části rozhraní API pro řízení skupiny klastrových prostředků

Rozhraní API pro řízení skupin klastrových prostředků	Umožněno v primární části	Umožněno v sekundární části
Přidání uzlu do domény obnovy	X	
Přidání záznamu zařízení CRG		
Změna skupiny klastrových prostředků	X	
Změna záznamu CRG zařízení	X	X
Vytvoření skupiny klastrových prostředků		
Vymazání skupiny klastrových prostředků	X	X
Distribuce informací	X	X
Ukončení skupiny klastrových prostředků[1]	X	
Inicializace přepnutí	X	
Výpis skupin klastrových prostředků	X	X
Výpis informací o skupině klastrových prostředků	X	X
Odstranění uzlu z domény obnovy	X	
Odstranění záznamu CRG zařízení	X	
Spuštění skupiny klastrových prostředků[1]	X	
Poznámka: 1. Je umožněno ve všech částech pro peer CRG, ale má vliv pouze na části používající rozhraní API.		

Použijete-li tato omezení, můžete synchronizovat skupiny klastrových prostředků v případě, že klastr již není rozdělený. Když se po stavu rozdělení uzly znovu připojí ke klastru, verze skupiny klastrových prostředků v primární části se zkopíruje do uzlů ze sekundární části.

Když spojujete dvě sekundární části pro peer model, část, která má skupinu klastrových prostředků se stavem Active, bude prohlášena jako vítěz (winner). Jestliže obě části mají stejný stav pro skupinu klastrových prostředků, pak část, která obsahuje první uzel uvedený ve skupině klastrových prostředků, bude prohlášena jako vítěz. Verze skupiny klastrových prostředků ve vítězné části budou zkopírovány do uzlů v druhé části.

Když je detekována část, v žádné z logických částí nelze spustit rozhraní Add Cluster Node Entry, Adjust Cluster Version ani Create Cluster API. Rozhraní Add Device Domain Entry API může být spuštěno jen v případě, že není rozdělen žádný z uzlů v doméně zařízení. Všechna ostatní rozhraní API pro řízení klastru mohou být spouštěna v libovolné části. Operace provedená rozhraní API však bude mít účinek pouze v té části, která toto rozhraní API spouští.

Změna stavu uzlů na ″Failed″

Někdy je hlášen stav rozdělení i v případě, kdy ve skutečnosti došlo k výpadku uzlu. Může k tomu dojít tehdy, když Služby klastrových prostředků ztratí komunikaci s jedním nebo více uzly a nejsou schopny detekovat, zda jsou uzly stále funkční. Pokud dojde k této situaci, existuje jednoduchý způsob, jak označit, že uzel selhal.

Upozornění: Když sdělíte Službám klastrových prostředků, že uzel selhal, zjednodušíte tím obnovu ze stavu rozdělení. Neměla by však být prováděna změna stavu uzlu na ″failed″, kdy je uzel ve skutečnosti stále aktivní a opravdu došlo k rozdělení na části. Pokud byste tak učinili, mohlo by dojít k rozdělení uzlu na více částí, aby uzel mohl převzít primární úlohu pro skupinu klastrových prostředků. Když dojde k situaci, kdy se dva uzly domnívají, že jsou primárním uzlem, může dojít k rozdělení nebo poškození dat, jako jsou soubory a databáze, protože více uzlů by nezávisle na sobě provádělo změny ve svých kopiích souborů. V případě, že by byl uzel v každé části označen jako primární, není navíc možné sloučit obě části zpět.

Když změníte stav uzlu na ″Failed″, může dojít k novému uspořádání úloh uzlů v doméně obnovy pro každou skupinu klastrových prostředků v části. Uzel se stavem nastaveným na ″Failed″ bude zařazen jako poslední záložní uzel. Pokud

selhalo více uzlů a je třeba změnit jejich stavy, ovlivní pořadí změny stavu uzlů konečné pořadí záložních uzlů v doméně obnovy. Pokud byl uzel, který selhal, primárním uzlem pro skupinu CRG, bude první aktivní záložní uzel zařazen jako nový primární uzel.

Související pojmy

"Sloučení" na stránce 22
Operace *sloučení* je podobná operaci opětovné připojení, až na to, že k ní dochází tehdy, když uzly, které jsou rozdělené, začínají opět komunikovat.

"Opětovné připojení" na stránce 20
Opětovné připojení znamená, že nezúčastněný člen se stane aktivním členem klastru.

Související úlohy

"Rady: Části klastru"
Tyto rady uplatněte pro části v klastru.

Související odkazy

Příkaz CHGCLUNODE

Rozhraní Change Cluster Node Entry (QcstChangeClusterNodeEntry) API

Příkaz STRCLUNOD

Rozhraní Start Cluster Node (QcstStartClusterNode) API

Pomocí produktu iSeries Navigator:

K tomu potřebujete mít nainstalovánu Volbu 41 (HA Switchable Resources) s licencí.

Když Služby klastrových prostředků ztratily komunikaci s uzlem a nemohou detekovat, zda je uzel stále funkční, bude mít klastrový uzel v zásobníku uzlů v produktu iSeries Navigator stav **Not communicating**. Možná budete chtít změnit stav uzlu z **Not communicating** na **Failed**. Potom budete moci uzel restartovat.

Chcete-li změnit stav uzlu z **Not communicating** na **Failed**, postupujte takto:

1. V produktu iSeries Navigator rozbalte **Centrální správu**.
2. Rozbalte **Klastry**.
3. Rozbalte klastr, který obsahuje uzel, u kterého chcete změnit stav.
4. Klepněte na **Uzly**.
5. Klepněte pravým tlačítek myši na uzel, u kterého byste chtěli změnit stav, a vyberte **Klastr → Změna stavu**.

a vyberte stav ClusterChange

Chcete-li restartovat uzel, postupujte takto:

1. Klepněte pravým tlačítkem myši na uzel a vyberte **Klastr → Spustit**.

Pomocí CL příkazů a API:

Chcete-li změnit stav uzlu z **Not communicating** na **Failed**, postupujte takto:

1. Chcete-li změnit stav uzlu z ″Partitioned″ na ″Failed″, použijte příkaz CHGCLUNODE nebo rozhraní QcstChangeClusterNodeEntry (Change Cluster Node Entry) API. To byste měli provést pro všechny uzly, které opravdu selhaly.
2. Použijte příkaz STRCLUNOD nebo rozhraní QcstStartClusterNode (Start Cluster Node) API ke spuštění klastrového uzlu, čímž umožníte uzlu, aby se znovu spojil s klastrem.

Rady: Části klastru

Tyto rady uplatněte pro části v klastru.

1. Pravidla pro operace omezení v rámci části jsou stanovena proto, aby bylo umožněno slučování částí. Bez těchto omezení by obnova klastru vyžadovala značné úsilí.

2. Pokud byly zničeny uzly v primární části, může být nutné zvláštní zpracování v sekundární části. Nejběžnějším scénářem, který způsobuje tuto situaci, je ztráta uzlu, který vytvořil primární části. Použijme tento příklad popsaný v tématu Obnova po chybách rozdělení a předpokládejme, že byla zničena část 1. V tomto případě musí být primární uzel pro skupiny klastrových prostředků B, C a D umístěn v části 2. Nejjednodušší procedurou obnovy je použít příkaz pro změnu položky klastrovacího uzlu a nastavit uzly A a B do stavu ″failed″. Další informace o této proceduře najdete v tématu Změna stavu uzlů z ″Partitioned″ na ″Failed″.Obnovu lze provést také manuálně. Chcete-li provést manuální obnovu, proveďte tyto operace:
   a. Odstraňte uzly A a B z klastru v části 2. Část 2 je nyní klastr.
   b. Vytvořte libovolná logická replikační prostředí, která jsou potřebná v novém klastru. Tj. API/CL příkaz pro spuštění skupiny klastrových prostředků, atd.

   Jelikož z definice klastru v části 2 byly odstraněny uzly, pokus o sloučení části 1 a 2 selže. Chcete-li opravit nesrovnalosti v definici klastru, spusťte rozhraní QcstDeleteCluster (Delete Cluster) API na každém uzlu v části 1. Potom přidejte uzly z části 1 do klastru a znovu vytvořte všechny definice skupin klastrových prostředků, domény obnovy a logické replikace. To vyžaduje značnou práci, při které můžete udělat chyby. Je velmi důležité, abyste tuto proceduru prováděli pouze při výpadku serveru.
3. Zpracování operace spuštění uzlu závisí na stavu spouštěného uzlu:

   Uzel buď selhal, nebo ho ukončila operace pro ukončení uzlu:
   a. Na přidávaném uzlu se spustí Služby klastrových prostředků.
   b. Definice klastru se zkopíruje z aktivního uzlu v klastru do spouštěného uzlu.
   c. Každá skupina klastrových prostředků, která má v doméně obnovy spouštěný uzel, je zkopírována z aktivního uzlu v klastru do spouštěného uzlu. Ze spouštěného uzlu nejsou do aktivního uzlu v klastru kopírovány žádné skupiny klastrových prostředků.

   Uzel je rozděleným uzlem:
   a. Definice klastru aktivního uzlu je porovnána s definicí klastru spouštěného uzlu. Pokud jsou definice stejné, bude spouštění pokračovat jako operace slučování. Pokud definice nejsou stejné, slučování se zastaví a bude třeba zásahu uživatele.
   b. Jestliže slučování pokračuje, stav spouštěného uzlu se nastaví na aktivní.
   c. Každá skupina klastrových prostředků, která má v doméně obnovy spouštěný uzel, je zkopírována z primární části skupiny klastrových prostředků do sekundární části skupiny klastrových prostředků. Skupiny klastrových prostředků mohou být kopírovány ze spouštěného uzlu do uzlů, které jsou již v klastru aktivní.

Související úlohy

“Změna stavu uzlů na ″Failed″” na stránce 136
Někdy je hlášen stav rozdělení i v případě, kdy ve skutečnosti došlo k výpadku uzlu. Může k tomu dojít tehdy, když Služby klastrových prostředků ztratí komunikaci s jedním nebo více uzly a nejsou schopny detekovat, zda jsou uzly stále funkční. Pokud dojde k této situaci, existuje jednoduchý způsob, jak označit, že uzel selhal.

Související odkazy

Rozhraní Delete Cluster (QcstDeleteCluster) API

Obnova klastru

Přečtěte si, jak provést obnovu (a nápravu chyb) po jiných selhání klastru, které se mohou vyskytnout.

Obnova po selhání klastrové úlohy

Selhání úlohy Služeb klastrových prostředků obvykle označuje nějaký další problém.

Měli byste vyhledat protokol úlohy asociovaný s úlohou, která selhala, a vyhledat zprávy, které popisují, proč úloha selhala. Opravte všechny chybové stavy.

| Příkazem CHGCLURCY (Změna obnovy klastru) můžete restartovat úlohu skupiny klastrových prostředků, která
| skončila, aniž byste ukončili a restartovali klastrování v uzlu.

1. CHGCLURCY CLUSTER(EXAMPLE)CRG(CRG1)NODE(NODE1)ACTION(*STRCRGJOB) Tento příkaz způsobí, že úloha skupiny klastrových prostředků, CRG1, v uzlu NODE1 bude zadána. Chcete-li spustit úlohu skupiny klastrových prostředků v uzlu NODE1, musí být klastrování aktivní v uzlu NODE1.
2. Restartujte klastrování v uzlu.

Používáte-li produkt pro správu klastrů od některého obchodního partnera IBM, přečtěte si dokumentaci k produktu.

Související pojmy

"Struktura úloh a uživatelské fronty" na stránce 112
Když spravujete klastr, musíte znát strukturu a uživatelské fronty.

Související úlohy

"Ukončení klastrového uzlu" na stránce 100
Zastavením (ukončením) uzlu zastavíte služby klastrových prostředků v daném uzlu.

"Spuštění klastrového uzlu" na stránce 100
Spuštěním klastrového uzlu spustíte Služby klastrových prostředků v uzlu v klastru. Počínaje klastry verze 3 se může uzel spustit sám a opětovně se připojit ke klastru - za předpokladu, že najde v klastru aktivní uzel.

Obnova poškozeného objektu klastru

I když je nepravděpodobné, že se někdy setkáte s poškozeným objektem, teoreticky může dojít k poškození objektu Služeb klastrových prostředků.

Pokud je systém aktivním uzlem, pokusí se provést obnovy z jiného aktivního uzlu v klastru. Systém provede následující kroky obnovy:

Pro poškozený vnitřní objekt

1. Poškozený uzel je ukončen.
2. Existuje-li v klastru alespoň jeden další aktivní uzel, poškozený uzel se automaticky restartuje a připojí se ke klastru. Proces opětovného připojení vyřeší situaci poškození.

Pro poškozenou skupinu klastrových prostředků

1. Uzel, který má poškozenou skupinu CRG, nebude moci provést žádné právě zpracovávané operace, které jsou asociované s touto skupinou CRG. Systém se potom pokusí automaticky obnovit skupinu CRG z jiného aktivního uzlu.
2. Pokud v doméně obnovy existuje alespoň jeden aktivní člen, bude obnova skupiny CRG fungovat. V opačném případě se úloha skupiny CRG ukončí.

Nemůže-li systém identifikovat žádný další aktivní uzel nebo na něj dosáhnout, budete muset provést následující kroky obnovy.

Pro poškozený vnitřní objekt

Obdržíte interní chybu klastrování (CPFBB46, CPFBB47 nebo CPFBB48).

1. Ukončete klastrování pro uzel, který obsahuje chybu.
2. Restartujte klastrování pro uzel, který obsahuje chybu. Proveďte to z jiného aktivního uzlu v klastru.
3. Pokud kroky 1 a 2 nevyřeší problém, odstraňte poškozený uzel z klastru.
4. Přidejte systém zpět do klastru a domény obnovy pro příslušné skupiny klastrových prostředků.

Pro poškozenou skupinu klastrových prostředků

Obdržíte chybu se zprávou, že došlo k poškození objektu (CPF9804).

1. Ukončete klastrování v uzlu, který obsahuje poškozenou skupinu klastrových prostředků.
2. Vymažte skupinu CRG (pomocí příkazu DLTCRG).
3. Není-li v klastru žádný další aktivní uzel který obsahuje objekt skupiny CRG, proveďte obnovu z média.

4. Spusťte klastrování v uzlu, který obsahuje poškozenou skupinu klastrových prostředků. To je možné provést z libovolného aktivního uzlu.
5. Když spustíte klastrování, systém znovu synchronizuje všechny skupiny klastrových prostředků. Pokud žádný další uzel v klastru neobsahuje skupinu CRG, budete možná muset skupinu CRG znovu vytvořit.

Obnova klastru po úplné ztrátě systému

Tyto informace použijte společně s příslušným kontrolním seznamem v publikaci Zálohování a obnova při obnově celého systému po jeho úplné ztrátě následkem neočekávaného výpadku serveru.

Scénář 1: Obnova do stejného systému

1. Chcete-li zabránit neshodám v informacích domény zařízení mezi interním kódem LIC a operačním systémem i5/OS, doporučujeme, abyste instalovali interní kód LIC pomocí volby 3 (Install Licensed Internal Code and Recover Configuration).

   Poznámka: K tomu, aby byla operace Install Licensed Internal Code and Recover Configuration úspěšná, musíte mít stejné diskové jednotky - s výjimkou diskové jednotky zaváděcího zdroje, pokud tato jednotka selhala. Musíte také provádět obnovu stejného vydání.
2. Jakmile nainstalujete interní kód LIC, postupujte podle procedury *Jak obnovit konfiguraci disku* popsané v kapitole 5 publikace *Zálohování a obnova*. Kroky této procedury vám pomohou vyhnout se konfiguraci ASP.
3. Když obnovíte informace o systému a jste připraveni zahájit klastrování v uzlu, který jste právě obnovili, musíte spustit klastrování z aktivního uzlu. Tím přenesete nejnovější informace o konfiguraci do obnoveného uzlu.

Scénář 2: Obnova do jiného systému

Jakmile jste obnovili informace o konfiguraci a zkontrolovali protokol úlohy, abyste se ujistili, že byly obnoveny všechny objekty, musíte provést následující kroky, abyste získali správnou konfiguraci domény klastrového zařízení.

1. Z uzlu, který jste právě obnovili, vymažte klastr.
2. Z aktivního uzlu proveďte tyto kroky:
   a. Odstraňte obnovený uzel z klastru.
   b. Přidejte obnovený uzel zpět do klastru.
   c. Přidejte obnovený uzel do domény zařízení.
   d. Vytvořte skupinu klastrových prostředků nebo přidejte uzel do domény obnovy.

Související úlohy

“Zálohování a obnova klastrů” na stránce 114
Pokud ve svých systémech používáte klastrování, je i tehdy důležité, abyste k ochraně dat vytvořili strategii zálohování a obnovy.

Související informace

Zálohování a obnova

Obnova klastru po zhroucení

V případě katastrofy, při které dojde ke ztrátě všech uzlů, budete muset znovu konfigurovat klastr.

Chcete-li se připravit na takovýto scénář, doporučujeme, abyste uložili informace o konfiguraci klastru a pořídili si tištěnou kopii těchto informací.

Související úlohy

“Zálohování a obnova klastrů” na stránce 114
Pokud ve svých systémech používáte klastrování, je i tehdy důležité, abyste k ochraně dat vytvořili strategii zálohování a obnovy.

Obnova klastru ze záložních pásek

Během běžných operací byste nikdy neměli muset provádět obnovu ze záložní pásky.

To je nutné pouze tehdy, když dojde k havárii a všechny uzly ve vašem klastru byly ztraceny. Pokud by došlo ke katastrofě, provedli byste obnovu pomocí běžných procedur obnovy, které jste stanovili po vytvoření své strategie zálohování a obnovy.

Související úlohy

"Zálohování a obnova klastrů" na stránce 114
Pokud ve svých systémech používáte klastrování, je i tehdy důležité, abyste k ochraně dat vytvořili strategii zálohování a obnovy.

Související informace

Zálohování a obnova

Časté otázky ke správě klastru pomocí produktu iSeries Navigator

Otázky a odpovědi týkající se grafického uživatelského rozhraní produktu iSeries Navigator pro vytváření a správu klastrů.

Grafické uživatelské rozhraní IBM pro vytváření a správu klastrů je dostupné pomocí produktu iSeries Navigator a přístupné pomocí Volby 41 (HA Switchable Resources). Podrobné informace o tomto rozhraní najdete v tématu Správa klastru pomocí produktu iSeries Navigator.

Zde je uveden seznam otázek a odpovědí ke správě klastru pomocí produktu iSeries Navigator.

Všeobecné

1. Existuje kontrolní seznam předpokladů pro vytvoření klastru?

Správa klastrů pomocí produktu iSeries Navigator

1. Kde je v rozhraní produktu iSeries Navigator umístěna funkce Klastry?
2. Jak postupovat při vytváření klastru?
3. V jakém vztahu je složka Klastry a systémová skupina Centrální správa?
4. V některých systémech iSeries v síti máme již klastry definovány. Jak je můžeme přidat, abychom je mohli zobrazit a spravovat pomocí produktu iSeries Navigator?
5. Žádný z uzlů v našem klastru není ve spuštěném stavu. Který uzel máme spustit jako první?
6. Proč záleží na tom, který uzel je spuštěn jako první?
7. Co znamená sloupec ″Aktuální primární uzel″ ve složkách Přepínatelná zařízení a přepínatelné aplikace?
8. Jak najdeme skupinu klastrových prostředků (CRG) zařízení v prostředí produktu iSeries Navigator?
9. Jak najdeme skupinu klastrových prostředků (CRG) aplikací v prostředí produktu iSeries Navigator?
10. Jak najdeme skupinu klastrových prostředků (CRG) dat v prostředí produktu iSeries Navigator?
11. Chtěli bychom zobrazit stav přepínatelného zařízení (přepínatelné CRG), aniž bychom se museli vracet zpět do složky Přepínatelný hardware. Je to možné?

Komunikace

1. Jakou IP adresu používá funkce Klastry v produktu iSeries Navigator ke komunikaci s klastrovými uzly? Používá IP adresu jména uzlu?

Zabezpečení

1. Proč je většina kontextových menu ve složce Klastry v produktu iSeries Navigator znepřístupněna nebo se nezobrazí?
2. Používá funkce Klastry v produktu iSeries Navigator hodnoty Administrace aplikací?
3. Proč funkce Klastry v produktu iSeries Navigator zobrazuje okno pro přihlášení k uzlům klastru?

Odstraňování problémů

1. Proč není v Centrální správě zobrazena složka Klastry?
2. Klastr již máme, ale nezobrazuje se ve složce Klastry. Proč?
3. Proč se ve složce Klastry nezobrazuje nejnovější stav?
4. Proč se neuskutečnilo přepnutí při selhání u skupiny přepínatelného hardwaru nebo u přepínatelné aplikace?
5. Obdrželi jsme zprávu o poškození objektu. Co máme dělat?
6. K přecházení mezi IP adresami používáme v průvodcích tlačítko ″Procházet″. Proč se v okně Procházet neobjeví všechny TCP/IP adresy, které očekáváme?
7. Proč je většina kontextových menu ve složce Klastry v produktu iSeries Navigator znepřístupněna nebo se nezobrazí?
8. Při práci v průvodci vytvořením nového klastru se zobrazil panel s titulkem: ″Nový klastr - Nebyl nalezen žádný přepínatelný software″. Je to špatně?
9. Jeden z uzlů je ve stavu bez komunikace. Jak to lze opravit?

Všeobecné

Existuje kontrolní seznam předpokladů pro vytvoření klastru?

Ano. Pomocí kontrolního seznamu pro konfiguraci klastru ověřte, zda jste připraveni ke konfiguraci klastrů ve vašem prostředí.

Zpět k otázkám

Správa klastru iSeries Navigator: Kde je v rozhraní produktu iSeries Navigator umístěna funkce Klastry?

Rozhraní produktu iSeries Navigator pro správu klastru je k dispozici jako součást softwarového balíku IBM iSeries Access. Funkce Klastry je umístěna v prostředí produktu iSeries Navigator ve složce Centrální správa. Podrobné informace najdete v tématu Správa klastru pomocí produktu iSeries Navigator.

Zpět k otázkám

Jak postupovat při vytváření klastru?

Chcete-li vytvořit jednoduchý klastr pomocí průvodce vytvořením nového klastru v prostředí produktu iSeries Navigator, postupujte takto:

1. V produktu iSeries Navigator rozbalte **Centrální správu**.
2. Klepněte pravým tlačítkem myši na **Klastry** a vyberte **Nový klastr**.
3. Při vytvoření nového klastru postupujte podle pokynů průvodce.

Úplné podrobnosti o vytváření a konfiguraci klastru najdete v tématu Konfigurace klastru.

Zpět k otázkám

V jakém vztahu je složka Klastry a systémová skupina Centrální správa?

Použijete-li k vytvoření klastru produkt iSeries Navigator, je na serveru Centrální správy vytvořena také systémová skupina. Tato systémová skupina je pojmenována stejně, jako se jmenuje klastr, a koncové systémy v této systémové skupině jsou klastrovými uzly. Tato systémová skupina má rovněž svůj vlastní zvláštní typ, takže produkt iSeries Navigator pozná, že je to zvláštní systémová skupina, která představuje klastr.

Důležité: Systém Centrální správy obsahuje systémové skupiny. Rozhodnete-li se změnit svůj aktuální systém Centrální správy v produktu iSeries Navigator, nový systém centrální správy nebude mít zvláštní klastrové systémové skupiny, a proto se tyto klastry nezobrazí ve složce Klastry.

Zpět k otázkám

V některých systémech iSeries v síti máme již klastry definovány.Jak je můžeme přidat, abychom je mohli zobrazit a spravovat pomocí produktu iSeries Navigator?

Chcete-li přidat existující klastr, aby se zobrazil v prostředí produktu iSeries Navigator, postupujte takto:

1. V produktu iSeries Navigator rozbalte **Centrální správu**.
2. Klepněte pravým tlačítkem myši na **Klastry** a vyberte volbu **Přidat stávající klastr.**
3. V okně **Přidat stávající klastr** zadejte jeden ze serverů v klastru.
4. Klepněte na tlačítko OK.

Zpět k otázkám

Žádný z uzlů v našem klastru není ve spuštěném stavu. Který uzel máme spustit jako první?

Měli byste spustit uzel, který byl ve spuštěném stavu jako poslední. Řekněme například, že máte v klastru dva uzly: A a B. Ani jeden z nich není momentálně spuštěn. Uzlem, který byl ve spuštěném stavu jako poslední, je uzel B. Jako první byste měli spustit uzel B, protože bude mít nejnovější informace o klastru.

Zpět k otázkám

Proč záleží na tom, který uzel je spuštěn jako první?

Záleží na tom proto, neboť uzel, který byl ve spuštěném stavu jako poslední, obsahuje nejnovější informace o klastru. Je to důležité, protože kdybyste spustili jiný uzel, který byl mimo provoz delší dobu, mohl by tento uzel obsahovat zastaralé informace o klastru. Nebezpečí spočívá v tom, že zastaralé informace se při spuštění tohoto uzlu mohou rozšířit do ostatních uzlů. Řekněme, že máte klastr obsahující dva uzly: A a B. Pokud byl uzel B aktivní ve spuštěném stavu jako poslední, bude obsahovat nejnovější informace o klastru. Rozhodnete-li se spustit jako první uzel A, může obsahovat určité zastaralé informace, ale přesto bude spuštěn. Když potom spustíte uzel B, připojí se k momentálně aktivnímu klastrovému uzlu (připojí se k uzlu A). Zastaralé informace z uzlu A se přenesou do uzlu B a výsledkem bude, že oba uzly budou obsahovat staré informace o klastru. Proto je důležité spustit uzel B jako první. Zastaralé informace o klastru mohou mít vliv na konfiguraci skupin přepínatelného zařízení. Objeví-li se určité problémy při spouštění skupin přepínatelného zařízení, neboť diskové jednotky jsou hlášeny na záložním uzlu, zatímco skupina přepínatelného hardwaru ukazuje jiný aktuální uzel, budete muset změnit roli uzlů v doméně obnovy, tj. učinit z uzlu, který je vlastníkem diskových jednotek, primární uzel.

Zpět k otázkám

Co znamená sloupec ″Aktuální primární uzel″ ve složkách Přepínatelný hardware, Přepínatelný software a Přepínatelná data?

Sloupec ″Aktuální primární uzel″ indikuje, že daný uzel momentálně slouží jako primární uzel skupiny přepínatelného zařízení nebo přepínatelného softwarového produktu. V terminologii klastrového rozhraní API to znamená, že je to uzel, jehož aktuální role v doméně obnovy je primární.

Zpět k otázkám

Jak najdeme skupinu klastrových prostředků (CRG) zařízení v prostředí produktu iSeries Navigator?

CRG (skupiny klastrových prostředků) zařízení se nazývají Skupiny přepínatelného hardwaru a lze je nalézt ve složce **Přepínatelný hardware** ve složce Klastry.

Zpět k otázkám

Jak najdeme skupinu klastrových prostředků (CRG) aplikací v prostředí produktu iSeries Navigator?

CRG (skupiny klastrových prostředků) aplikací se nazývají Přepínatelné softwarové produkty a lze je nalézt ve složce **Přepínatelný software** ve složce Klastry.

Zpět k otázkám

Jak najdeme skupinu klastrových prostředků (CRG) dat v prostředí produktu iSeries Navigator?

CRG (skupiny klastrových prostředků) dat se nazývají Skupiny přepínatelných dat a lze je nalézt ve složce **Přepínatelná data** ve složce Klastry.

Zpět k otázkám

Chtěli bychom zobrazit stav skupiny přepínatelného hardwaru (CRG zařízení), aniž bychom se museli vracet zpět do složky Přepínatelný hardware. Je to možné?

Abyste vždy, když chcete vidět stav, nemuseli přecházet do složky Přepínatelný hardware, můžete otevřít nové okno se zobrazením Přepínatelného hardwaru, a to tak, že klepnete pravým tlačítkem myši na složku **Přepínatelný hardware** a vyberete volbu **Otevřít**. V samostatném okně se ukážou skupiny přepínatelného hardwaru (CRG zařízení) a příslušné stavové informace. Totéž platí pro složky **Přepínatelný software** a **Přepínatelná data**.

Zpět k otázkám

Komunikace: Jakou IP adresu používá funkce Klastry v produktu iSeries Navigator ke komunikaci s klastrovými uzly? Používá IP adresu jména uzlu?

V hlavní složce ″Klastry″ je sloupec ″Server″, ve kterém se zobrazují informace o nakonfigurovaných klastrech. Jméno serveru je také na panelu vlastností každého klastru. Server uvedený ve sloupci Server představuje klastrový uzel, který je rozhraním produktu iSeries Navigator používán ke komunikaci s klastrem. Týká se to pouze způsobu, jakým produkt iSeries Navigator komunikuje s objektem klastru na serveru, nikoli způsobu, jakým klastrové uzly komunikují mezi sebou. Server používaný správou klastru v prostředí produktu iSeries Navigator nemá nic společného s aktuálním serverem Centrální správy.

Pokud uzel používaný produktem iSeries Navigator ke komunikaci s klastrem selže, můžete změnit komunikační prostředek na jiný klastrový uzel, který pak bude provádět klastrové akce.

Chcete-li změnit server, který bude rozhraním produktu iSeries Navigator používán ke komunikaci s klastrem, postupujte takto:

1. V produktu iSeries Navigator rozbalte **Centrální správu**.
2. Rozbalte **Klastry**.
3. Klepněte pravým tlačítkem myši na klastr a vyberte volbu **Změna serveru**.

Zpět k otázkám

Zabezpečení: Proč je většina kontextových menu ve složce Klastry v produktu iSeries Navigator znepřístupněna nebo se nezobrazí?

Některé operace jsou dostupné pouze v závislosti na stavu aktuální konfigurace klastru. Nemůžete například zastavit uzel, který je již zastaven, nebo přidat uzel do klastru, který již má nakonfigurován maximální počet uzlů, tj. čtyři. Vysvětlení, proč jsou u konkrétních úkolů některé položky znepřístupněny nebo nejsou k dispozici, najdete v online nápovědě.

Některé operace nejsou dostupné, jestliže nemáte dostatečná oprávnění. Budete-li používat produkt iSeries Navigator a budete mít oprávnění třídy uživatele *SECOFR, budete mít přístup ke všem klastrovým operacím i k administraci.

Produkt iSeries Navigator použije oprávnění typu Administrace aplikací z aktuálního systému Centrální správy k tomu, aby určil, zda máte oprávnění typu Administrace aplikací pro různé operace správy klastru pomocí produktu iSeries Navigator.

Podrobné informace o práci s funkcí Administrace aplikací najdete v tématu Administrace aplikací.

Zpět k otázkám

Používá funkce Klastry v produktu iSeries Navigator hodnoty Administrace aplikací?

Ano. Správa klastru v produktu iSeries Navigator použije hodnoty oprávnění typu Administrace aplikací z aktuálního systému Centrální správy k tomu, aby určila, zda máte oprávnění typu Administrace aplikací pro různé klastrové operace.

Produkt iSeries Navigator používá pro přístup dva typy nastavení oprávnění: **Operace s klastrem** a **Administrace klastrů**.

Oprávnění **Operace s klastrem** umožňuje:
- Prohlížet stav klastru.
- Spouštět a zastavovat uzly.
- Spouštět a zastavovat přepínatelný hardware a software.
- Ručně přepínat přepínatelný hardware a software.

Oprávnění **Administrace klastrů** umožňuje:
- Vytvářet a odstraňovat klastry.
- Přidávat a odebírat uzly.
- Přidávat a odstraňovat přepínatelný hardware, přepínatelný software a ASP.
- Měnit vlastnosti přepínatelného hardwaru a softwaru.

Zpět k otázkám

Proč funkce Klastry v produktu iSeries Navigator zobrazuje okno pro přihlášení k uzlům klastru?

Produkt iSeries Navigator se v některých případech pokouší komunikovat se všemi klastrovými uzly. Záleží to na stavu klastru. Potřebuje-li iSeries Navigator komunikovat s určitým uzlem, prohledá nejprve stávající vyrovnávací paměť přihlášení v produktu iSeries Navigator, kde se snaží najít existující otevřené spojení. Pokud takové spojení nenajde, vyzve uživatele, aby se přihlásil. Zrušíte-li přihlašovací okno, iSeries Navigator se pokusí umožnit uživateli provádění klastrových operací. Jestliže iSeries Navigator nemůže komunikovat s uzly, nebudou některé operace pravděpodobně možné.

Zpět k otázkám

Odstraňování problémů: Proč není v Centrální správě zobrazena složka Klastry?

Možná jste v počítači neprovedli úplnou instalaci produktu iSeries Access for Windows. Pravděpodobně jste provedli základní instalaci nebo jste vybrali některé uživatelské volby. Podrobné informace k instalaci najdete v tématu iSeries Access.

Zpět k otázkám

Klastr již máme, ale nezobrazuje se ve složce Klastry. Proč?

Stručně řečeno: Není zobrazen, protože v systému Centrální správy neexistuje systémová skupina reprezentující klastr. Tato systémová skupina je vytvořena správou klastru v produktu iSeries Navigator buď při vytvoření klastru, nebo při

přidání klastru do složky Klastry pomocí akce ″Přidat stávající klastr″. Chcete-li vidět systémové skupiny, rozbalte v Centrální správě složku **Systémové skupiny**. Klastrové systémové skupiny se zobrazují jako systémové skupiny ″třetích stran″, ale neplatí, že všechny systémové skupiny ″třetích stran″ jsou klastrové.

Zpět k otázkám

Proč se ve složce Klastry nezobrazuje nejnovější stav?

Produkt iSeries Navigator zobrazí informace o nakonfigurovaných klastrech jako ″snímek″ momentálního stavu v okamžiku, kdy získal od klastrových uzlů nejnovější informace o klastru. Tyto informace se pak zobrazí v okně produktu iSeries Navigator. Nejsou prováděny automatické pravidelné aktualizace informací. Nejnovější ″snímek″ informací lze získat ruční aktualizací. Použijte menu **Zobrazení** v produktu iSeries **Obnovit**. Další možnost je, že nastavíte iSeries Navigator tak, aby prováděl automatické aktualizace.

Zpět k otázkám

Proč se neuskutečnilo přepnutí při selhání u skupiny přepínatelného zařízení, přepínatelné aplikace nebo přepínatelné datové skupiny?

Nejpravděpodobnější je, že v klastru nebylo spuštěno přepínání prostředků (skupina klastrových prostředků). Jinými slovy, předtím, než mělo dojít k automatickému přepnutí při selhání, nebyl přepínatelný prostředek ve spuštěném stavu. Pokud má probíhat přepínání při selhání, musí být přepínatelné prostředky spuštěny.

Zpět k otázkám

Obdrželi jsme zprávu o poškození objektu. Co máme dělat?

Pravděpodobně jste obdrželi zprávu tohoto typu: CPF811C Uživatelská fronta QUGCLUSRQ v QCLUMGT je poškozená.

1. První možností je objekt vymazat a obnovit. To je možné, pokud jste předtím objekt uložili.

2. Druhou možností je poškozený objekt vymazat. Je-li například poškozen objekt QUGCLUSRQ v knihovně QCLUMGT, je nejprve třeba tento objekt vymazat. Potom přidejte existující klastr pomocí produktu iSeries Navigator. Při přidávání klastru grafické uživatelské rozhraní kontroluje, zda objekty existují, a pokud ne, znovu je vytvoří. V části Jak můžu přidat existující klastr, abych jej mohl zobrazit a spravovat pomocí produktu iSeries Navigator? naleznete podrobnosti o přidání existujícího klastru.

Zpět k otázkám

K přecházení mezi IP adresami používáme v průvodcích tlačítko ″Procházet″. Proč se v okně Procházet neobjeví všechny TCP/IP adresy, které očekáváme?

Seznam obsahuje pouze možné IP adresy (kandidáty). Nejste omezeni pouze na použití adres ze seznamu možných adres zobrazeného v okně. Můžete zadat libovolnou adresu klastrového rozhraní. Pokud se však iSeries Navigator nebude moci připojit pomocí IP adresy, kterou jste zadali jako primární IP adresu, dojde později k chybě. Produkt iSeries Navigator používá primární IP adresu pro připojení k uzlům klastru.

Zpět k otázkám

Při práci v průvodci vytvořením nového klastru se zobrazil panel s titulkem: ″Nový klastr - Nebyl nalezen žádný přepínatelný software″. Je to špatně?

Ne, není to špatně a není to chyba. Význam přesně odpovídá textu: Rozhraní produktu iSeries Navigator nenalezlo žádný přepínatelný software, který by mohl být automaticky nainstalován pomocí průvodce. Produkt iSeries Navigator

vyžaduje, aby jakýkoliv automaticky instalovatelný přepínatelný software byl v souladu s architekturou i5/OS pro aplikace podporující klastry. Produkt iSeries Navigator přitom podporuje pouze podmnožinu této architektury, nikoli celou architekturu.

Zpět k otázkám

Jeden z uzlů je ve stavu bez komunikace. Jak to lze opravit?

Pokud se ztratí kontakt s jedním nebo více klastrovými uzly a selhání ztracených uzlů není potvrzeno, dojde k rozdělení klastru. Další informace najdete v tématu Chyby rozdělení.

Stav rozdělení je někdy ohlášen, když skutečně došlo k výpadku uzlu. Může k tomu dojít tehdy, když Služby klastrových prostředků ztratí komunikaci s jedním nebo více uzly a nejsou schopny detekovat, zda jsou uzly stále funkční. Pokud tento stav vznikne, existuje jednoduchý mechanismus, jak signalizovat, že uzel selhal. Podrobné informace najdete v tématu Změna rozdělených uzlů na selhané.

Zpět k otázkám

Kam zavolat, pokud potřebujete podporu

Toto téma si přečtěte v případě, že potřebujete kontaktovat společnost IBM kvůli otázkám ohledně klastrů.

Jestliže potřebujete pomoc při rozhodování, zda mohou klastry pomoci při vašem podnikání, nebo jestliže se dostanete do problémů po implementaci klastrů, můžete kontaktovat následující služby:

- Pro dodatečnou technickou marketingovou pomoc nebo v případě, že si chcete najmout poradenské služby IBM, zašlete e-mail do střediska Continuous Availability Center ve středisku iSeries Technology Center na adresu rchclst@us.ibm.com.
- Při jiných problémech kontaktujte buď obchodního partnera IBM, od kterého jste koupili klastrovací softwarový balík, nebo zavolejte na číslo 1-800-IBM-4YOU (1-800-426-4968).

Související úlohy

“Konfigurace klastrů” na stránce 96
Pochopte postup vytváření klastru.

Související informace pro klastry

Zde naleznete související informace pro klastry.

Redbooks

- Data Resilience Solutions for IBM i5/OS High Availability Clusters
- Clustering and IASPs for Higher Availability
- High Availability on the AS/400 System: A System Manager’s Guide
- IBM eServer iSeries Independent ASPs: A Guide to Moving Applications to IASPs
- The System Administrator’s Companion to AS/400 Availability and Recovery

Webové stránky

- High Availability and Clusters (www.ibm.com/servers/eserver/iseries/ha)
  Stanice IBM pro vysokou dostupnost a klastry

Jak ukládat soubory ve formátu PDF

Chcete-li uložit soubor PDF na pracovní stanici za účelem zobrazení nebo tisku:

1. V prohlížeči klepněte pravým tlačítkem myši na požadované PDF (nebo přímo na některý z výše uvedených odkazů).
2. Pokud používáte program Internet Explorer, klepněte na **Uložit cíl jako...**. Pokud používáte program Netscape Communicator, klepněte na **Save Link As**.
3. Vyhledejte adresář, kam chcete dokument ve formátu PDF uložit.
4. Klepněte na **Save** (Uložit).

Jak stáhnout produkt Adobe Reader

Program Adobe Acrobat Reader potřebujete k prohlížení a tisku souborů PDF. Jeho kopii si můžete stáhnout z webových stránek společnosti Adobe (www.adobe.com/products/acrobat/readstep.html)  .

Licence na kód a prohlášení o vyloučení záruky

IBM vám uděluje nevýhradní copyright licenci na užívání všech uvedených příkladů programového kódu, z nichž můžete generovat obdobné funkce přizpůsobené vašim specifickým potřebám.

KROMĚ JAKÝCHKOLIV ZÁKONNÝCH ZÁRUK, KTERÉ NEMOHOU BÝT VYLOUČENY, IBM, JEJÍ PROGRAMOVÍ VÝVOJÁŘI A DODAVATELÉ NEPOSKYTUJÍ ZÁRUKY ANI PODMÍNKY, VYJÁDŘENÉ NEBO ODVOZENÉ VČETNĚ, BEZ OMEZENÍ, ODVOZENÝCH ZÁRUK PRODEJNOSTI NEBO VHODNOSTI PRO URČITÝ ÚČEL A ZÁRUK NEPORUŠENÍ PRÁV TŘETÍCH STRAN V SOUVISLOSTI S PROGRAMEM NEBO TECHNICKOU PODPOROU, POKUD EXISTUJE.

ZA ŽÁDNÝCH OKOLNOSTÍ NEJSOU IBM, JEJÍ PROGRAMOVÍ VÝVOJÁŘI NEBO DODAVATELÉ ODPOVĚDNI ZA ŽÁDNOU Z NÍŽE UVEDENÝCH SITUACÍ, ANI V PŘÍPADĚ, ŽE BYLI O MOŽNOSTI JEJICH VZNIKU PŘEDEM INFORMOVÁNI:

1. ZTRÁTA NEBO POŠKOZENÍ DAT;
2. PŘÍMÉ, ZVLÁŠTNÍ, NAHODILÉ NEBO NEPŘÍMÉ ŠKODY, NEBO JAKÉKOLIV JINÉ EKONOMICKÉ NÁSLEDKY ŠKOD.
3. UŠLÝ ZISK, ZTRÁTA OBCHODNÍCH TRANSAKCÍ, VÝNOSŮ, DOBRÉHO JMÉNA NEBO PŘEDPOKLÁDANÝCH ÚSPOR.

VZHLEDEM K TOMU, ŽE NĚKTERÉ PRÁVNÍ SYSTÉMY NEDOVOLUJÍ VYLOUČENÍ NEBO OMEZENÍ PŘÍMÝCH, NAHODILÝCH NEBO NÁSLEDNÝCH ŠKOD, NEMUSÍ SE NA VÁS NĚKTERÁ NEBO VŠECHNA VÝŠE UVEDENÁ OMEZENÍ NEBO VYLOUČENÍ VZTAHOVAT.

Dodatek. Poznámky

Tyto informace platí pro produkty a služby nabízené v USA.

IBM nemusí v ostatních zemích nabídnout produkty, služby a funkce popsané v tomto dokumentu. Informace o produktech a službách, které jsou momentálně dostupné ve vašem regionu, můžete získat od místního zástupce IBM. Žádný odkaz na produkt, program nebo službu IBM neznamená a ani z něj nelze vyvozovat, že smí být použit pouze uvedený produkt, program či služba společnosti IBM. Použít lze jakýkoli funkčně ekvivalentní produkt, program či službu neporušující práva IBM k duševnímu vlastnictví. Za vyhodnocení a ověření činnosti libovolného produktu, programu či služby jiného výrobce než IBM však odpovídá uživatel.

IBM může mít patenty nebo podané žádosti o patent, které zahrnují předmět tohoto dokumentu. Získání tohoto dokumentu uživateli neposkytuje licenci na tyto patenty. Písemné dotazy ohledně licencí můžete zaslat na adresu:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Pokud máte zájem o licenci v zemi s dvoubajtovou znakovou sadou (DBCS), kontaktujte zastoupení společnosti IBM ve vaší zemi, nebo písemně zastoupení společnosti IBM na adrese:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

Následující odstavec se netýká Velké Británie nebo kterékoliv jiné země, kde taková opatření odporují místním zákonům: SPOLEČNOST INTERNATIONAL BUSINESS MACHINES CORPORATION TUTO PUBLIKACI POSKYTUJE TAKOVOU, "JAKÁ JE", BEZ JAKÝCHKOLIV ZÁRUK, VYJÁDŘENÝCH NEBO ODVOZENÝCH VČETNĚ, BEZ OMEZENÍ, ODVOZENÝCH ZÁRUK NEPORUŠENÍ PRÁV TŘETÍCH STRAN, ZÁRUKY PRODEJNOSTI NEBO VHODNOSTI PRO URČITÝ ÚČEL. Právní řády některých zemí nepřipouštějí vyloučení vyjádřených nebo odvozených záruk v určitých transakcích a proto se na vás výše uvedené omezení nemusí vztahovat.

Tato publikace může obsahovat technické nepřesnosti nebo typografické chyby. Informace zde uvedené jsou pravidelně aktualizovány a v nových vydáních této publikace již budou tyto změny zahrnuty. IBM má právo kdykoliv bez upozornění zdokonalovat nebo měnit produkty a programy popsané v této publikaci.

Jakékoliv odkazy v této publikaci na webové stránky jiných společností než IBM jsou poskytovány pouze pro pohodlí uživatele a nemohou být žádným způsobem vykládány jako doporučení těchto webových stránek ze strany IBM. Materiály obsažené na takovýchto webových stránkách nejsou součástí materiálů k tomuto produktu IBM a tyto webové stránky mohou být používány pouze na vlastní nebezpečí.

IBM může použít nebo distribuovat jakékoliv informace, které jí sdělíte, libovolným způsobem, který společnost považuje za odpovídající, bez vzniku jakýchkoliv závazků vůči vám.

Držitelé licence na tento program, kteří si přejí mít přístup i k informacím o programu za účelem (i) výměny informací mezi nezávisle vytvořenými programy a jinými programy (včetně tohoto) a (ii) vzájemného použití sdílených informací, mohou kontaktovat:

IBM Corporation
Software Interoperability Coordinator, Department YBWA
3605 Highway 52 N

Rochester, MN 55901
U.S.A.

Informace tohoto typu mohou být dostupné za určitých podmínek. V některých případech připadá v úvahu zaplacení poplatku

IBM poskytuje licencovaný program popsaný v těchto informacích a veškeré dostupné licencované materiály na základě podmínek uvedených ve smlouvě IBM Customer Agreement, v Mezinárodní licenční smlouvě IBM na strojový kód nebo v jiné ekvivalentní smlouvě.

Veškerá data týkající se výkonu, která jsou uvedena v tomto dokumentu, byla získána v řízeném prostředí. Výsledky získané v jiných provozních prostředích se proto mohou významně lišit. Některá měření mohla být prováděna v systémech na úrovni vývoje a nelze tedy zaručit, že tato měření budou ve všeobecně dostupných systémech stejná. Některá měření mohla být navíc odhadnuta pomocí extrapolace. Skutečné výsledky se mohou lišit. Uživatelé tohoto dokumentu by si měli ověřit použitelnost dat pro svoje specifické prostředí.

Informace týkající se produktů jiných firem než IBM, byly získány od dodavatelů těchto produktů, z jejich publikovaných sdělení, nebo z jiných veřejně dostupných zdrojů. IBM nezkoumala tyto produkty a nemůže tudíž potvrdit spolehlivost, kompatibilitu a další konstatování, vztahující se k těmto produktům. Dotazy, které se týkají vlastností produktů od jiných dodavatelů, musí být adresovány příslušným dodavatelům.

Veškerá prohlášení týkající se budoucích trendů nebo strategií IBM podléhají změnám bez předchozího upozornění a představují pouze cíle a záměry.

Všechny uváděné ceny IBM jsou maloobchodní ceny navržené společností IBM, jsou nyní platné a mohou se bez upozornění změnit. Ceny prodejců se mohou lišit.

Informace zde uvedené slouží pouze pro účely plánování. Informace v tomto dokumentu mohou být změněny, než se produkty popsané v tomto dokumentu stanou obecně dostupnými.

Tyto publikace obsahují příklady údajů a sestav, používaných v každodenních obchodních činnostech. Abyste si udělali co neúplnější představu, obsahují příklady názvy konkrétních podniků, firemních značek a produktů. Všechny tyto názvy jsou fiktivní a jakákoliv podobnost se jmény a adresami, používanými ve skutečných obchodních podnicích, je čistě náhodná.

LICENČNÍ INFORMACE:

Tyto informace obsahují vzorové aplikační programy ve zdrojovém jazyce ilustrující programovací techniky na různých operačních platformách. Tyto vzorové programy můžete bez závazků vůči IBM jakýmkoliv způsobem kopírovat, měnit a distribuovat za účelem vývoje, používání, odbytu či distribuce aplikačních programů odpovídajících rozhraní API pro operační platformu, pro kterou byly ukázkové programy napsány. Tyto vzorové programy nebyly důkladně testovány za všech podmínek. Proto IBM nemůže zaručit ani naznačit spolehlivost, provozuschopnost ani funkčnost těchto programů.

Každá kopie nebo část těchto vzorových programů nebo práce z nich odvozené musí zahrnovat následující copyrightovou výhradu:

© (jméno vaší společnosti) (rok). Části tohoto kódu jsou odvozeny ze vzorových programů společnosti IBM Corporation. © Copyright IBM Corp. _zadejte rok nebo roky_. Všechna práva vyhrazena.

Jestliže si prohlížíte tyto informace ve formě softcopy, nemusí se zobrazit fotografie a barevné ilustrace.

Informace o programovacím rozhraní

Tato publikace Klastry je určena pro programovací rozhraní, které umožňuje zákazníkům psát programy za účelem získání služeb operačního systému IBM i5/OS.

Ochranné známky

Níže uvedené termíny jsou ochranné známky společnosti International Business Machines Corporation ve Spojených státech a případně v dalších jiných zemích.

| 400
| i5/OS
| IBM
| iSeries
| OS/400
| Redbooks

| Intel, Intel Inside (loga), MMX a Pentium jsou ochranné známky společnosti Intel Corporation ve Spojených státech
| a případně v dalších jiných zemích.

Microsoft, Windows, Windows NT a logo Windows jsou registrované ochranné známky společnosti Microsoft Corporation ve Spojených státech a případně v dalších jiných zemích.

Java a všechny ochranné známky obsahující slovo Java jsou ochranné známky společnosti Sun Microsystems, Inc. ve Spojených státech a případně v dalších jiných zemích.

| Linux je ochranná známka, jejímž majitelem je Linus Torvalds, ve Spojených státech a případně v dalších jiných
| zemích.

UNIX je registrovaná ochranná známka skupiny The Open Group ve Spojených státech a jiných zemích.

Další jména společností, produktů nebo služeb mohou být ochrannými známkami jiných společností.

Ustanovení a podmínky

Oprávnění k užívání těchto publikací je uděleno na základě následujících ustanovení a podmínek.

Osobní použití: Pokud zachováte všechny výhrady týkající se vlastnických práv, můžete tyto publikace kopírovat pro své osobní nekomerční použití. Tyto publikace ani jakékoliv jejich části nesmíte bez výslovného souhlasu IBM distribuovat, prezentovat ani z nich vytvářet odvozená díla.

Komerční použití: Pokud zachováte všechny výhrady týkající se vlastnických práv, můžete tyto publikace kopírovat, distribuovat a prezentovat výhradně uvnitř svého podniku. Bez výslovného souhlasu IBM nesmíte z těchto publikací vytvářet odvozená díla ani je (nebo jejich části) nesmíte kopírovat, distribuovat či prezentovat mimo rámec svého podniku.

Kromě oprávnění, která jsou zde výslovně udělena, se na publikace nebo jakékoliv informace, data, software a další duševní vlastnictví obsažené v těchto publikacích nevztahují žádná další vyjádřená ani odvozená oprávnění, povolení či práva.

IBM si vyhrazuje právo odvolat oprávnění zde udělená, kdykoli usoudí, že používání publikací poškozuje jeho zájmy nebo že výše uvedené pokyny nejsou řádně dodržovány.

Tyto informace můžete stahovat, exportovat či reexportovat pouze při dodržení všech příslušných zákonů a nařízení včetně veškerých vývozních zákonů a nařízení USA.

IBM NEPOSKYTUJE ŽÁDNOU ZÁRUKU, POKUD JDE O OBSAH TĚCHTO PUBLIKACÍ. TYTO PUBLIKACE JSOU POSKYTOVÁNY NA BÁZI ″JAK JSOU″ (AS-IS), BEZ JAKÝCHKOLIV ZÁRUK, VYJÁDŘENÝCH NEBO ODVOZENÝCH VČETNĚ, BEZ OMEZENÍ, ODVOZENÝCH ZÁRUK PRODEJNOSTI, NEPORUŠENÍ PRÁV TŘETÍCH STRAN NEBO ZÁRUKY VHODNOSTI PRO URČITÝ ÚČEL.

Vytištěno v Dánsku společností IBM Danmark A/S.