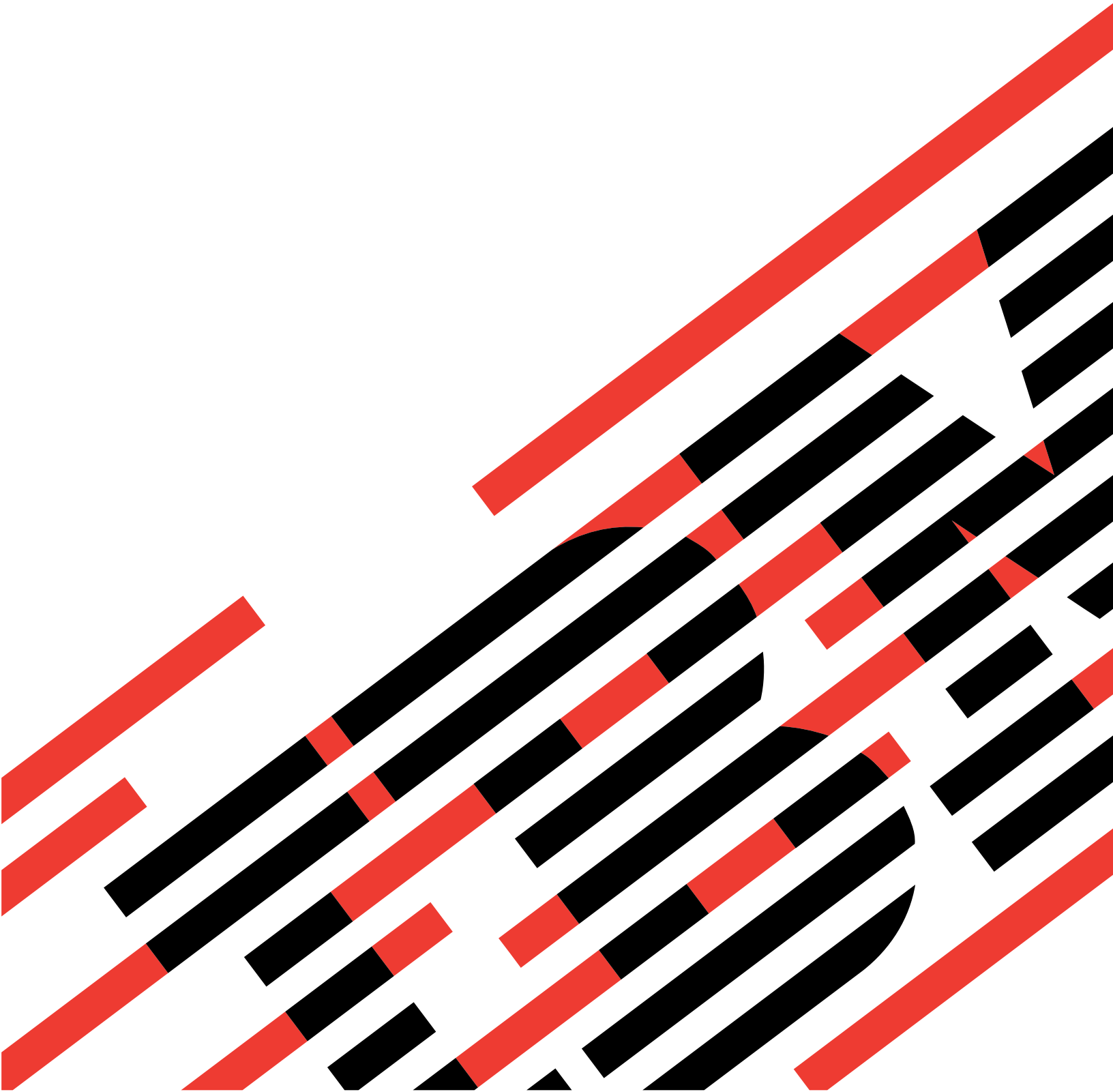
IBM

eserver

iSeries

パフォーマンス



IBM

@server

iSeries

パフォーマンス

目次

パフォーマンス

システムのパフォーマンスを管理するためにどれだけの投資をするでしょうか。業務変更の必要性は、時には予想よりも早い場合があります。業務変更に対して有効な対応をするためには、システムも変更しなければなりません。システムの管理は、一見するとほんのありきたりの、時間ばかりかかる仕事のようにも見えるかもしれません。しかし、投資は、システムがより効率的に稼働して、それが業務に反映されることにより、速やかに効果を上げることになります。それが効率的なのは、変更が計画に基づき管理されているためです。

iSeries サーバーのパフォーマンスの管理は、 iSeries 実行管理機能の完全な理解を必要とする複合タスクです。システム・パフォーマンスに影響のあるすべての異なるプロセスを理解することは、未経験のユーザーにとって挑戦となります。パフォーマンス上の問題を解決するには、それぞれ独自の要件およびサポートされる機能を使用して、大規模なツールを効率よく使用する必要があります。パフォーマンス・データを収集および分析した後でさえ、その情報を使用して行う事柄を知ると、気が遠くなるかもしれません。

このトピックでは、パフォーマンス管理と関連したタスクおよびツールを通じて説明します。

V5R2 の新機能
このトピックでは、このリリースでの新規または変更点について説明します。

トピックの印刷
この情報の印刷版が必要な場合には、 PDF を印刷するためにここに進みます。

パフォーマンスの計画
iSeries サーバーのパフォーマンス目標を設定すると、測定可能なパフォーマンス・ベンチマークによりパフォーマンス・データを比較することができます。このトピックでは、それらのベンチマークの設定方法および設定後のベンチマークの使用方法について説明します。

パフォーマンスを管理するための環境のセットアップ
iSeries サーバーには、システム・パフォーマンスを管理するための強力なアプリケーションが含まれています。ただし、それらは、独自の業務環境の特定のニーズを満たすために正しく構成されていなければなりません。パフォーマンス・データを定期的に収集、モニター、および分析するようにアプリケーションを構成する方法について学習します。

iSeries パフォーマンスの管理
パフォーマンス管理は、現在の機能を測定し、傾向を認識し、適当な調整を行うことによってご使用のコンピューター・システムの稼働率を最適化し、エンド・ユーザーを満足させ、応答時間またはジョブ・スループットなどの管理要件を満たすために必要です。パフォーマンス管理は、業務の効率を保ち、通常の業務活動の延期を避けるために必要です。したがって、パフォーマンスの管理は、日常の操作の一部です。

パフォーマンス管理用のアプリケーション
iSeries システムにおけるパフォーマンスの管理では、さまざまな特殊アプリケーションを使用する必要があります。これらのアプリケーションのそれぞれは、システム・パフォーマンスに特定の洞察を提供します。このトピックでは、いくつかのアプリケーションおよびそれぞれのアプリケーションの使用方法について説明します。

シナリオ: パフォーマンス
パフォーマンス管理について学習する最善の方法の 1 つは、ご使用の業務環境でこれらのアプリケーションまたはツールを使用できる方法を示す例をご覧になることです。これらの例について調べてください。

関連情報
IBM(R) の関連情報には、技術、ノウハウ、および「方法」に関する情報が含まれています。

注: このトピックには、コード例が含まれています。法律上の重要な情報に関しては、『コードの特記事項情報』をお読みください。

V5R2 の新機能

ここでは、このリリースでの新機能を挙げます。

- **iSeries ナビゲーターのモニター**
  サポートされる新規モニターおよび既存のモニターへの変更について調べてください。
- **収集サービス**
  パフォーマンス・データベース・ファイルへの変更および新機能についてお読みください。
- **Performance Tools**
  報告書への変更についてお読みください。
- **Performance Management/400**
  PM/400 がデータを自動収集する方法についてお読みください。
- **Performance Explorer**
  Performance Explorer データベース・ファイルへの変更および新機能についてお読みください。
- **Capacity Upgrade on Demand (CUoD)**
  iSeries の Capacity Upgrade on Demand のサポートについてお読みください。
- **e-business パフォーマンスの管理**
  e-business 環境における iSeries のパフォーマンスの管理に有効なタスク、考慮事項、および資源についてお読みください。
- **主記憶域の類縁性**
  プロセスとスレッドでメモリーおよびプロセッサーの資源の類縁性を上げる方法についてお読みください。

新機能: モニター

ここでは、このリリースで新しくなった部分を紹介します。

- ファイル・モニター: 複数のプラットフォーム環境を管理する機能。 iSeries ナビゲーターで、コマンドを実行し、ファイルをモニターし、複数のプラットフォームのエンドポイント・システムを管理することができます。
- B2B 活動モニター: 指定された統合システムに関する詳細情報とともにトランザクション・データを表示し、 OS/400 コマンドを自動的に実行してトランザクション・カウント全体またはトランザクションの所要時間を制御できる活動モニター。
- システム・モニター・メトリックには、Point-to-Point Protocol (PPP) と関連のある情報が含まれています。いくつかのメトリックは、CPU 稼働率プロパティーに追加されました。これらの新規メトリックは、ファイル・システム、ストリーム・ファイル、ジャーナル操作、およびカウンター・セットに関連があります。

新機能: 収集サービス

ここでは、このリリースで新しくなった部分を紹介します。

新規リリースのインストール

新規リリースにアップグレードするときにパフォーマンス・データを処理する方法に関していくつかの選択項目があります。パフォーマンス・データの変換 (CVTPFRDTA) コマンドを使用して収集ライブラリー内のデータの変換を継続することができます。新規自動データ収集サポートを使用することができます。パフォーマンス・データベース・ファイルを変換しない場合や、収集サービスが開始され、データベース・ファイルを自動的に作成するためのオプションを指定した場合、前のリリース・ファイルのパフォーマンス・ライブラリーが作成されます。パフォーマンス・データベース・ファイルがそのライブラリーに移動されます。このアクションにより、新規ファイルを作成でき、前のリリース・レベルから既存のデータ・ファイルを保存します。最後に、パフォーマンス・データベース・ファイルを変換せず、収集サービスを実行しない場合、ファイルの削除 (DLTF) コマンド (DLTF library/QAPM*) を使用して QAPMxxxx ファイルを削除することができます。

ユーザー定義カテゴリー

V5R2 より前は、すべてのパフォーマンス収集カテゴリーが事前定義され、すべてのデータ収集プログラムがシステムに付属していました。 V5R2 以降では、独自のパフォーマンス収集カテゴリーを定義することができます。ユーザー定義パフォーマンス収集カテゴリーを使用すると、以下のことを行うことができます。

- コレクターに新規パフォーマンス収集カテゴリーを追加します。 コレクター・データ・カテゴリーの登録 (QypsRegCollectorDataCategory) は、マネージメント・セントラルの収集サービス機能の 1 つ以上のコレクター定義にユーザー定義データ・カテゴリーを追加します。
- データを収集するためにデータ収集プログラムに新規カテゴリーを提供します。
- 管理収集オブジェクト内のカテゴリーのために収集されるデータを保管します。
- システム定義カテゴリーのデータ収集と同じ方法でカテゴリーのためのデータ収集をスケジュールし、実行します。
- 管理収集オブジェクト API (Management Collection Object APIs) を使用して、管理収集オブジェクト内の新規カテゴリーのために収集されるデータにアクセスします。

ドミノおよび HTTP サーバー (Apache 駆動) は、この新機能を使用して、それらのパフォーマンス・データを収集サービスに統合します。

ユーザー定義トランザクション

V5R2 以降では、収集サービスは、 Performance Explorer トランザクション境界 API (performance explorer transaction boundary APIs) を使用して、アプリケーション・ユーザーに独自のトランザクションを定義する機能を提供します。収集サービス・データ・カテゴリーは USRTNS で、ユーザー定義トランザクションと関連したパフォーマンス・データベース・ファイルは QAPMUSRTNS です。これらの API には、以下のものが含まれます。

- トランザクションの開始
- トランザクションの終了

パフォーマンス・データベース・ファイル

すべてのデータベース・ファイル (QAPMAPPN は例外) は対話式です。これは、列内のテキストをアルファベット順にソートし、表内の特定の語を検索できることを意味しています。

以下の表では、新規および変更データベース・ファイルを示しています。

データベース・ファイル	説明
QAPMJOBWT	ジョブ、タスク、およびスレッド待ち条件に関するデータを含む新規ファイル。
QAPMJOBWTD	ファイル QAPMJOBWT 内にあるカウンター・セットの説明を含む新規ファイル。
QAPMHTTPB	IBM HTTP Server (Apache 駆動) に関する基本データを含む新規ファイル。
QAPMHTTPD	IBM HTTP Server (Apache 駆動) に関する詳細データを含む新規ファイル。
QAPMDOMINO	ドミノ (iSeries 版) に関するデータを含む新規ファイル。
QAPMPPP	Point-to-Point Protocol (PPP) に関するデータを含む新規ファイル。
QAPMUSRTNS	ユーザー定義トランザクションに関するデータを含む新規ファイル。
QAPMSYSTEM	ファイル・システム・カウンターおよびジャーナル・カウンターをサポートする新規フィールド。
QAPMSYS	単一システムの異なる区画上のシステム・クロック間の相違を判別する方法を提供する新規フィールド。
QAPMECL	サポートされないプロトコルから破棄されたフレームの数を報告する新規フィールド。
QAPMETH	サポートされないプロトコルから破棄されたフレームの数を報告する新規フィールド。
QAPMCONF	新規レコード・キー。
QAPMIOPD	入出力アダプター・データをサポートする新規フィールド。
QAPMMIOP	処理時間をサポートする新規フィールド。
QAPMJOBMI	ファイル・システム・カウンターおよびジャーナル・カウンターをサポートする新規フィールド。
QAPMDISK	独立 ASP 関連の変更をサポートする新規および変更フィールド。
QAPMJOBOS	ファイル・システム・カウンターをサポートする新規フィールド。

新機能: Performance Tools ライセンス・プログラム

ここでは、このリリースで新しくなった部分を紹介します。

- BEST/1 は、このリリースで有効な Performance Tools からは除去されています。キャパシティー要件のサイズ変更を行うための代替手段が含まれていますが、 IBM Workload Estimator (PM/400 に統合) および BMC の PATROL for iSeries (AS/400) - Predict 製品 (分散システムの場合) に限定されていません。以下のコマンドまたはメニュー・オプションのサポートが、 V5R2 で有効な Performance Tools ライセンス・プログラムからは除去されています。
  - BEST/1 の開始 (Start BEST/1) (Start BEST/1)
  - BEST/1 モデルの分析 (Analyze BEST/1 Model) (ANZBESTMDL)
  - BEST/1 モデルの作成 (Create BEST/1 Model) (CRTBESTMDL)

- MDLSYS ファイルの変換 (Convert MDLSYS File) (CVTMDLSYSF)
- BEST/1 モデルの削除 (Delete BEST/1 Model) (DLTBESTMDL)
- BEST/1 ファイルの印刷 (Print BEST/1 File) (PRTBESTF)
- 指定されたパフォーマンス・データ・メンバーのバージョンの検査 (Check Version of Specified Performance Data Member) (QCYCHKV)
- MDLSYS から BEST/1 への System/36 ファイルの変換 (Convert System/36 Files from MDLSYS to BEST/1) (QCYCVTBD)
- 指定されたパフォーマンス・データ・メンバーの初日および最終日の検索 (Find First and Last Date of Specified Performance Data Member) (QCYFLDT)
- 前の層からのメッセージの受信および呼び出し層への再送信 (Receive Messages from Previous Layer and Resend to Invoking Layer) (QCYRSNDM)
- MDLSYS の開始 (Start MDLSYS) (MDLSYS)
- キャパシティー・プランニング/モデル化 (Capacity planning/modeling) は、PERFORM メニューから除去されています。

- パフォーマンス・データの表示 (DSPPFRDTA) コマンドと関連のある機能は、 Performance Tools プラグインとして iSeries ナビゲーターから使用可能です。インターフェースは、DSPPFRDTA コマンドとほぼ同等の機能を提供します。これにより、Performance Tools ライセンス・プログラム報告書のサブセットを生成し、表示することができます。
- 構成要素報告書の印刷 (PRTCPTRPT) コマンドは、いくつかの機能強化を反映するために更新され、その中には以下のものが含まれます。
  - 新規選択カテゴリーを使用すると、印刷するジョブ・タイプを選択できます。この選択は、いくつかのモデルにおける対話式作業を分析するために役立ちます。
  - 新規選択カテゴリーを使用すると、報告書内に含まれるジョブ優先順位を選択することができます。この選択は、システム・オーバーヘッドが高いときに、高優先順位の作業の合計を入手し、優先順位 00 の作業すべてを検出するために役立ちます。
  - 新規列は、書き込みキャッシュ・オーバーランのパーセンテージを報告します (% 書き込みキャッシュ・オーバーラン)。
  - 「データベースのジャーナル処理の要約 (Database Journaling Summary)」のセクションには、ジャーナル・カウンターおよび操作に関連した情報を示す新規サブセクションが含まれています。新規カウンターのデータは、QAPMJOBMI ファイルに保管されています。
- システム報告書の印刷 (PRTSYSRPT) コマンドは、いくつかの機能強化を反映するために更新され、その中には以下のものが含まれています。
  - 「通信の要約 (Communications Summary)」には、Point-to-Point Protocol (PPP) 情報が含まれています。 PPP プロトコルについては、「資源報告書 (Resource Report)」の「通信の詳細 (Communications Detail)」セクションにも示されています。パフォーマンス・データの表示 (DSPPFRDTA) コマンドは、「通信回線の詳細の表示 (Display Communications Line Detail)」画面からの Point-to-Point Protocol 情報も示します。
  - HTTP サーバー・ジョブによって処理されたトランザクションに関する情報を示す新規セクションが追加されました。この情報は、報告書の下部にある要約および平均行を使用してそれぞれの間隔ごとに表示されます。
  - 非対話式サーバー・ジョブの詳細を示すために、新規カテゴリーが「ワークロード (Workload)」、「資源稼働率 (Resource Utilization)」、および「資源稼働率の拡張 (Resource Utilization Expansion)」セクション内に含まれました。

- 「システム活動の処理 (Work with System Activity)」画面は、現在の処理キャパシティーを示します。この情報は、特に、区画のキャパシティーが変更されますが、仮想プロセッサーの数が変更されない場合に、二重プロセッサー環境内で役立ちます。「システム活動の処理 (Work with System Activity)」画面は、待ち状態にあるジョブまたはタスクによって費やされた時間の合計パーセントも示します。オプション 6 (待ち状態の詳細) を指定して、待ち状態のカテゴリーのリストを表示することができます。待ち時間のアカウンティング・データに関連のあるいくつかのフィールドが QAITMON ファイルに追加されます。このフィールドは WRKSYSACT コマンドによって作成されます。
- 「トランザクション・レポート (Transaction Report)」の「オブジェクトによる占有/ロック対立の要約 (Summary of Seize/Lock Conflicts by Object)」セクションは、最大 500 オブジェクトを表示することができます。これは、トレース・データが 500 以上の占有/ロック対立を持つことができることを意味しています。したがって、このセクションでは、500 個の最も重要なオブジェクトが示されます。
- パフォーマンス・トレースの開始 (STRPFRTRC) コマンドは、より大きいトレース・データ表をサポートします。
- この報告書内のいくつかのフィールドのサイズは、より大きい値を示すように増分されました。この変更は、以下の報告書に影響を与えます。

報告書	セクション	フィールド
システム	資源稼働率	ディスク入出力 (秒単位)
システム	資源稼働率の拡張	物理ディスク入出力、ローカル・データベース入出力、ディスク入出力 (同期および非同期)、 DIO/Sec (同期および非同期)
構成要素	構成要素間隔活動	ディスク入出力 (秒単位)
ジョブ間隔	非対話型ジョブの要約	入出力の数 (秒単位)
ジョブ間隔	非対話型ジョブの詳細	Nbr 入出力 (秒単位)

新機能: Performance Management/400

ここでは、このリリースで新しくなった部分を紹介します。

前のリリースからアップグレードしたり、前のリリースで PM/400 パフォーマンス・データ収集をオフにしたりしない限り、 PM/400 は、自動的にパフォーマンス・データを収集します。 IBM への送信許可を与えるまで、データは送信されません。パフォーマンス・データの自動収集の利点は、パフォーマンス・データが必要なときにデータがあり、 PM/400 を活動化するとすぐに PM/400 報告書を受信できることです。

PM/400 報告書の最新情報については、 PM/400 Web サイト  に進んでください。

パフォーマンス・データの変換

新しいリリースをインストールした後は、パフォーマンス・データベース・ファイルは下位レベルであるため、パフォーマンス・データの収集を継続できるようにアクションを実行する必要があります。Performance Management/400、収集サービス、または Performance Tools ライセンス・プログラムを使用し、新しいリリースをインストールする前にデータを収集した場合、前のリリースからのパフォーマンス・データの処理方法に関して、以下の選択項目があります。

- データの変換
  パフォーマンス・データの変換 (CVTPFRDTA) コマンドを使用して、収集ライブラリー内のデータを変換します。
- 自動データ収集サポートの使用
  パフォーマンス・データベース・ファイルを変換しない場合や、収集サービスが開始され、データベース・ファイルを自動的に作成するためのオプションを指定した場合、前のリリース・ファイルのパフォーマンス・ライブラリーが作成されます。パフォーマンス・データベース・ファイルがそのライブラリーに移動されます。このアクションにより、新規ファイルを作成でき、前のリリース・レベルから既存のデータ・ファイルを保存します。以下の項目に気付く必要があります。
  - 作成されるライブラリー名は QPFRD*vrmnn* です。ここで、*vrm* は現行バージョン、リリース、およびモディフィケーションで、 *nn* は 01 で始まる固有の順序番号です。たとえば、QPFRD52001 です。
  - ライブラリーは、*EXCLUDE 共通権限を使用して作成されます。ライブラリーは QSYS ユーザー・プロファイルによって所有され、元のライブラリーの所有者は *ALL 権限を与えられています。
  - すべての QAPM*xxxx* ファイルは移動されます。
  - 前のリリースからのデータを保持したくない場合には、ライブラリーの削除 (DLTLIB) コマンド (DLTLIB qpfrdvrmnn) を使用して QPFRDvrmnn ライブラリーを削除することができます。
- QAPM*xxxx* ファイルの削除
  パフォーマンス・データベース・ファイルを変換せず、収集サービスを実行しない場合、ファイルの削除 (DLTF) コマンドを使用して QAPM*xxxx* ファイルを削除することができます (DLTF *library*/QAPM*)。

新機能: Performance Explorer

ここでは、このリリースで新しくなった部分を紹介します。

- Performance Explorer コマンドを使用するには、ユーザーが *SERVICE 権限を持っている必要があります。
- 新規 CL コマンドを使用すると、取り込むデータ量を削減するためのフィルターを作成することができます。
  - PEX フィルターの追加 (ADDPEXFTR)
  - PEX フィルターの処理 (WRKPEXFTR)
  - PEX フィルターの除去 (RMVPEXFTR)
  - STRPEX コマンドは、フィルターをサポートするために拡張されました。
- Performance Explorer は、管理収集オブジェクト (*MGTCOL) を使用して、収集されたデータをそのデータベース・ファイルに変換します。 PEX データの作成 (CRTPEXDTA) コマンドは、変換を実行します。
- 以下のイベントが PEX 定義の追加 (ADDPEXDFN) コマンドに追加されました。
  - アプリケーション・イベント (APPEVT)
  - ポータブル・アプリケーション・ソリューション環境 (PASE) イベント (PASEEVT)
  - ジャーナル・イベント (JRNEVT)
  - iSeries ネットサーバー、ファイル・サーバー、ネットワーク・ファイル・システム・サーバー、およびクライアント・イベント (FILSVREVT)
  - 同期イベント (SYNCEVT)
  - エキスパート・キャッシュ・イベント (EXPCCHEVT)

- ADDPEXDFN コマンド上のオペレーティング・システム・イベント (OSEVT) パラメーター *HOSTSVRCNN に新規値が追加されました。この値は、*DBSVRCNN 値と同じです。 *HOSTSVRCNN は、推奨値です。
- Performance Explorer 定義を処理できるようにする新規コマンドが提供されます。 PEX 定義の処理 (WRKPEXDFN) コマンド。
- Performance Explorer セッションの終了時に、不完全な新規収集状態を表示できます。この状態は、収集が予期せずに終了したことを意味しています。
- このリリースの新規データベース・ファイルには、以下のものが含まれます。

新規ファイル	説明
QAYPEPROCI	このファイルには、収集データ内のすべての命令アドレスと関連のあるプロシージャー情報が含まれています。このファイルには、以前に QAYPEMII、QAYPELICI、QAYPENMI、および QAYPENLIC ファイル内にあったデータが含まれています。
QAYPECFGI	このファイルには、収集定義情報が含まれています。このファイルには、以前に QAYPECOFG、QAYPECICFG、QAYPESTCFG、QAYPETRCFG、および QAYPEHWCFG ファイル内にあったデータが含まれています。
QAYPEFTRI	このファイルには、収集フィルター定義情報が含まれています。
QAYPELTASK	このファイルには、定義情報のタスク名およびタスク番号が含まれています。
QAYPERINF	このファイルには、各種解決情報 (例: それぞれの IP アドレスの文字ストリング・バージョン) が含まれています。
QAYPETSKSW	このファイルには、タスク切り替えイベントに特定の情報が含まれています。このファイルには、以前に QAYPEBASE ファイル内にあったデータが含まれています。

- このリリースの削除済みデータベース・ファイルには、以下のものが含まれます。

削除済みファイル	データの移動先
QAYPECICFG	QAYPECFGI
QAYPECOCFG	QAYPECFGI
QAYPEHWCFG	QAYPEPROCI
QAYPELICI	QAYPECFGI
QAYPELNAMT	QAYPELTASK
QAYPELNUMT	QAYPELTASK
QAYPEMII	QAYPEPROCI
QAYPENLIC	QAYPEPROCI
QAYPENMIC	QAYPEPROCI
QAYPEPERD	周期的なモードがサポートされていないため、表は必要ありません。

削除済みファイル	データの移動先
QAYPEPSUM	プロファイルの要約データは、QAYPEPPANE および QAYPETASKI データから計算できます。
QAYPEPWDW	このファイルには、使用されなかったプロファイルの要約データが含まれていました。
QAYPESTCFG	QAYPECFGI
QAYPES36	システム/36 イベントはサポートされていません。
QAYPETRCFG	QAYPECFGII
QAYPETRCPT	QAYPETIDX
QAYPEUNKWN	QAYPEUSRDF

- このリリースのデータベース・ファイルへの変更には、以下のものが含まれています。
  - 文字データ・フィールドでは、デフォルトの CCSID 65535 を使用します。
  - 8 バイトのアドレス、命令、およびセグメントは、それぞれのバイトを 2 つの 16 進文字に変換するのではなく、 8 バイトの 16 進フィールド内に保管され、16 バイトの文字フィールドまたは 16 バイトの 16 進フィールドのいずれかに保管されます。
  - パック 10 進フィールド内に保管されている真の数値データは、以下の整数フィールドに変更されました。
    - 5 桁のパック 10 進フィールドは、9B 整数フィールドに変更されます。
    - 10 桁のパック 10 進フィールドは、18B 整数フィールドに変更されます。
    - 20 桁のパック 10 進フィールドは、値が 8 バイトの符号付き整数より大きくない限り、18B 整数フィールドに変更されます。

トピックの印刷

パフォーマンス・トピックの PDF 版をダウンロードし、表示するには、 パフォーマンス (約 1235 KB 134 ページ) を選択します。この PDF には、パフォーマンス・データ・ファイル情報はありません。

パフォーマンス・データ・ファイル情報の PDF 版をダウンロードし、表示するには、パフォーマンス・データ・ファイル (約 946 KB 142 ページ) を選択します。

関連するトピックを表示またはダウンロードすることもできます。

- また、マネージメント・セントラル (約 656 KB、62 ページ) には、マネージメント・セントラルが、以下のようなサーバー管理タスクを合理化するのに役立つさまざまな情報も含まれています。
  - ユーザーおよびグループの管理
  - データのパッケージおよび送信
  - コマンドの実行
  - マネージメント・セントラル・スケジューラーまたは拡張ジョブ・スケジューラーを使用したタスクまたはジョブのスケジュール
- 実行管理では、以下の作業管理の概念を説明しています。
  - ジョブの存続期間
  - 日次作業管理
  - システムの構造
  - 作業の実行方法

以下の PDF のいずれかを表示または印刷することもできます。

- 資料:
  - Performance Tools for iSeries (約 400 ページ)
  - iSeries Performance Capabilities Reference
    本書の最新版を含む Web サイトにリンクしてください。この解説書には、パフォーマンス・ベンチマークに役立つサーバー・パフォーマンス、キャパシティー・プランニング、およびサーバー・パフォーマンスの計画に関する高水準の技術情報を提供しています。
  - この解説書には、パフォーマンス・ベンチマークに役立つサーバー・パフォーマンス、キャパシティー・プランニング、およびサーバー・パフォーマンスの計画に関する高水準の技術情報を提供しています。
- Redbooks:
  - AS/400 Performance Management (約 504 ページ)
  - AS/400 HTTP Server Performance and Capacity Planning (約 205 ページ)
  - Java and WebSphere Performance on IBM eserver iSeries Servers (約 235 ページ)
  - Managing OS/400 V5R1 with Operations Navigator (約 550 ページ)
  - DB2 UDB/WebSphere Performance Tuning Guide (約 360 ページ)
  - Lotus Domino for AS/400: Performance, Tuning, and Capacity Planning (約 550 ページ)
  - AS/400 Performance Explorer Tips and Techniques (約 550 ページ)

表示用または印刷用の PDF ファイルを Netscape Navigator からワークステーションに保存するには、次のようにします。

1. ブラウザーで PDF を開く (上記のリンクをクリックする)。
2. ブラウザーのメニューから**「ファイル」**をクリックする。
3. **「名前を付けて保存」**をクリックする。(IE の場合は、フロッピー・ディスクのアイコン (名前を付けて保存) をクリックする。)
4. PDF を保存したいディレクトリーに進む。
5. **「保存」**をクリックする。

PDF ファイルを表示したり印刷したりするには、Adobe Acrobat Reader が必要です。これは、Adobe Web サイト (www.adobe.com/prodindex/acrobat/readstep.html) からダウンロードできます。

パフォーマンスの計画

システムのパフォーマンスの計画には、パフォーマンス目標の設定、それらの目標に基づくベンチマークの作成、およびシステムの展開の計画が必要です。このセクションでは、ご使用のシステムのパフォーマンスの計画に必要なステップについて解説します。

ご使用のシステムのパフォーマンスを計画するとき、ご使用のシステムが提示している業務要件を完全に理解し、それらの業務ニーズをパフォーマンス目標に変換できる必要があります。業務ニーズが変化するにつれて、パフォーマンス要求も変化しなければならないことに注意してください。

開始する最善の方法は、業務のピーク時に、ご使用のコンピューター・システムに必要な 1 時間ごとおよび 1 日ごとの最大対話式トランザクション・スループットを見積もることです。その後、ローカルおよびリモート・ワークステーションにアクセス可能な平均応答時間を決定することができます。業務要件を満たすために時間内に完了するようにするため、通常のバッチ処理にかかる時間、およびそのスケジュール方法について考慮する必要があります。

次に、統計の基本セットを確立することができ、その後、以下のものを含むパフォーマンス目標計画を文書化する必要があります。

- 1 時間ごとのピーク・トランザクション
- 1 日ごとのピーク・トランザクション
- ローカル・ワークステーションの許容平均応答時間
- ピーク対話式トランザクション
- 実行時および予期される所要時間を指定した、主なスケジュール済みバッチ・ジョブのリスト
- その他の必要なスケジュールされていないバッチ・ジョブのリスト

パフォーマンスを計画するには、以下のタスクを完了してください。

システム・ベンチマークの設定
適切なシステム・ベンチマークを設定すると、正しく調整されたシステムに対してパフォーマンス・データを提供します。システム変更前およびシステム変更後の両方のパフォーマンス・ベンチマークは、トラブルシューティングおよび計画の両方のために重要な情報を提供します。

システムの拡張時および拡張方法の決定
業務ニーズが変化するにつれて、ご使用のシステムも変更しなければなりません。変更の準備を行うには、現行システムをモデル化し、システム、構成、またはワークロードが変更された場合に何が起こるかを知りたいと思われることでしょう。

Capacity Upgrade on Demand (CUoD)
業務ニーズが変化してさらに資源が必要になれば、それに応じて Capacity Upgrade on Demand を使用して、現在ご使用の iSeries サーバーにプロセッサーを追加することができます。プロセッサーの注文および活動化を行いますが、プロセッサーは試行として (一時的) または永続的に活動化することができます。

パフォーマンス管理ストラテジーの選択
異なる業務ニーズには、異なるパフォーマンス管理ストラテジーが必要です。以下に、3 つの基本業務モデルおよび提案されているパフォーマンス管理ストラテジーを示します。

システム・ベンチマークの設定

新しい対話式アプリケーションの追加、またはシステムのアップグレードを実行するなど、システム構成に大きな変更を加える前には、システム・ベンチマークを設定するべきです。正確なベンチマーク情報を保守することは、トラブルシューティングに必要不可欠な情報です。最低限、ベンチマークは収集サービスからの現行の収集オブジェクトを含んでいるべきです。ユーザーの環境によっては、 パフォーマンス・エクスプローラーを使用して、さらに詳細な情報を保守する必要があります。

ベンチマークの設定は、以下のものを必要とします。

- 正しい iSeries の構成が使用可能になっていること。
- アプリケーションおよびデータが典型的かつ有効なものであること。
- すべてのプログラムおよびソフトウェアの正しいバージョンが使用可能であること。
- テストを実行するために必要な数のユーザー、およびワークステーションが使用可能であること。
- 各ユーザー用のトランザクションが定義されていること。

ワークステーション上のユーザーをシミュレートすることのできる特殊な装置を使用しないと、対話式のワークロードに対する意味のあるベンチマークを実行することはほとんど不可能です。もちろん、バッチ・ベンチマークを実行することは、対話式アプリケーションのパフォーマンスをテストするタスクほど複雑ではありません。このタイプのテストにおいては、前述のポイントの最初の 3 つが有効です。しかし、実際のカスタマーの環境にあるような並行バッチおよび対話式作業のシステム・ベンチマークの設定では、やはり適切な数のユーザーおよびワークステーションが必要です。

システムの拡張時および拡張方法の決定

業務ニーズが発展するのに従って、システム・ニーズも発展します。将来のシステム・ニーズおよび成長を計画するには、システム、構成、またはワークロードが変更された場合に何が起きるかを判断する必要があります。このプロセスはトレンド分析と呼ばれ、毎月行うべきです。システムをリソース・キャパシティーの指針に近づけるにつれ、このデータをさらに頻繁に収集したいと思うかもしれません。

トレンド分析は、対話式およびバッチ環境で別々に行うべきです。会社でいくつかの大規模なアプリケーションが使用されているのであれば、そのアプリケーションのトレンド分析を行いたいと思うでしょう。追跡することが重要だと思われるその他の環境は、月末に処理することになります。トレンド分析データを一環して収集するのは重要なことです。システムのワークロードのピークが午前 10 時から午後 2 時の間で、この時間のトレンド分析データを収集した場合は、このデータを別の時間に取得したデータと比較してはなりません。

キャパシティー・プランニングおよびパフォーマンス分析というジョブを正しく実行するには、パフォーマンス・データの収集、分析、保守、および保存を行う必要があります。 IBM は、キャパシティー・プランニング、リソース見積もり、およびサイジングに役立ついくつかのツールを提供しています。

Performance Management/400 (PM/400)
PM/400 は、データの収集、データの分析、およびデータの保存を完全に自動化し、理解しやすい、要約されたパフォーマンスおよびキャパシティー情報を提供します。 PM/400 は、かぎとなるパフォーマンス標識の分析を続けることにより、システム・リソースを計画し、管理するのに役立ちます。この機能は、OS/400 ライセンス・プログラムに付属しています。機能を活動化し、データが収集され、IBM の送信されていることを定期的にチェックする以外は、何も行う必要はありません。すべてのコレクション・サイトはネットワーク保護されており、 PM/400e サービスは、所有権のないパフォーマンス・データだけを IBM に送信します。転送時間は、ユーザーによって完全に制御されています。

Workload Estimator
Workload Estimator は、特定のタイプのワークロードの見積もりに基づいて、システム・ニーズのサイジングを行うのに役立ちます。 Web ベース・アプリケーションを通して、既存のシステムの使用状況、パフォーマンスおよび PM/400 によって報告される成長に適合した、必要とされる iSeries システムへのアップグレードのサイジングが行えます。また追加のオプションとして、ドミノ、Java、および WebSphere のような特定のアプリケーションを追加するためのキャパシティーや、複数の AS/400 または iSeries の従来型の OS/400 ワークロードを 1 つのシステムに統合するためのキャパシティーも、計画に含めることができます。このようなキャパシティーを含めておくと、使用してい

る独自のシステムから得られた既存の使用状況データに基づいて、将来のシステム要件を計画することが可能です。このアプリケーションは、プロセッサー・オンデマンド環境、また論理区画を使用した環境はサポートしていません。

PATROL for iSeries (AS/400) - Predict
この製品は、高可用性および最適なパフォーマンスのために必要な日常の管理タスクの多くを自動化することにより、 iSeries のパフォーマンスを管理しやすくします。さらに、iSeries 環境の成長を計画するのに役立つ、詳細なキャパシティー・プランニング情報を提供します。

パフォーマンス戦略の作成およびインプリメントの詳細については、『パフォーマンス管理ストラテジーの選択』を参照してください。

Capacity Upgrade on Demand (CUoD)

iSeries 用 Capacity Upgrade on Demand (CUoD) は、選択したサーバー・モデルの 1 つ以上の中央プロセッサーを、サーバーを再始動したり業務を中断する必要なしに、動的に活動化することができるフィーチャーです。ご使用のサーバーにすでにインストールされている待機プロセッサーを購入して、試行または永続ベースで活動化することができます。

CUoD を使用して追加プロセッサーを活動化することができるため、ニーズの増大に応じた新しい処理能力分のみの対価を支払えばよいことになります。プロセッサーのキャパシティーを増強するときに、現行の運用を中断する必要はありません。

このトピックは、 CUoD の動作と、それを利用する際に必要なことを理解するためのものです。

Capacity Upgrade on Demand (CUoD) の概念
CUoD を概念的に理解するには、ここから始めてください。

Capacity Upgrade on Demand (CUoD) の準備
計画および準備の重要な要件を考察します。

Capacity Upgrade on Demand (CUoD) の注文
待機プロセッサーを活動化できるようになる活動化コードを入手するには、活動化フィーチャーを注文する必要があります。

待機プロセッサーの活動化
ここには、ご使用の iSeries サーバーの待機プロセッサーを 1 つ以上活動化する際の情報が記載されています。

シナリオ: Capacity Upgrade on Demand (CUoD)
ここには、管理者が追加キャパシティーの計画、注文、および活動化を行う際に必要なステップの例が記載されています。

Capacity Upgrade on Demand (CUoD) の概念

14 日間無料の試行で、またはプロセッサーの永続的活動化を購入すると、 Capacity Upgrade on Demand (CUoD) を使用して、選択した iSeries モデル上で追加プロセッサーを活動化することができます。この機能により、新しいワークロードに応じてキャパシティーを迅速にしかも経済的に追加することができ、予期していなかったパフォーマンスの要求にサーバーを適応させることができるため、 iSeries サーバーの価値が大幅に高まります。

選択した iSeries モデルは、複数の**スタートアップ・プロセッサー**が付属して出荷されます。スタートアップ・プロセッサーは、 iSeries サーバーの出荷時にすでに活動化されているプロセッサーのことです。**待機**

プロセッサーは、サーバーに付属していますが、ユーザーが活動化しないと使用できないプロセッサーのことです。待機プロセッサーは 14 日間の試行で一時的に活動化することができます。あるいは活動化フィーチャーを購入して、指定される活動化コードを入力することにより永続的に活動化することができます。次の表は、各モデルごとに使用可能なスタートアップ・プロセッサーと待機プロセッサーの数を示したものです。

モデル	プロセッサー・フィーチャー	スタートアップ・プロセッサー	待機プロセッサー	インストール済みプロセッサー
825	2473	3	3	6
830	2349	4	4	8
830	2351*	1	7	8
840	2352	8	4	12
840	2353	12	6	18
840	2354	18	6	24
840	2416	8	4	12
840	2417	12	6	18
840	2419	18	6	24
870	2486	8	8	16
890	2487	16	8	24
890	2488	24	8	32
890	2497	16	8	24
890	2498	24	8	32

* 限定使用

CUoD を利用するにあたって必要な概念として、次のものがあります。

Capacity Upgrade on Demand (CUoD) の試行期間
14 日間の無料の試行期間を利用して、追加プロセッサー・キャパシティーの利点を評価します。

Capacity Upgrade on Demand (CUoD) のソフトウェア・ライセンスの考慮事項
CUoD を活動化するとソフトウェア層にどのように影響するかが記載されています。

プロセッサー・オン・デマンド活動化コード
待機プロセッサーの一部またはすべてを永続的に活動化することにした場合は、プロセッサー・オン・デマンド活動化フィーチャーを 1 つ以上注文して購入する必要があります。これによってユーザーに活動化コードが提供され、これを使用して待機プロセッサーを活動化することができます。

Electronic Service Agent の要件
CUoD 待機プロセッサーを活動化するために必要な重要プロダクト・データを迅速にそして正確に送信するには、 Electronic Service Agent を使用します。

Capacity Upgrade on Demand (CUoD) の試行期間: Capacity Upgrade on Demand の14 日間の試行期間を利用して、待機プロセッサーの使用の評価を無料で行うことができます。活動化後に試行期間として有効なのは、サーバーの最初の開始後の 1 回限りで、かつ活動化コードが入力されるたびに 1 回限りです。

試行期間は、開始後、パワーオンの日数として 14 日間有効です。試行期間の時間が刻時されるのは、サーバーの電源がオンになっている間のみです。試行期間中は、スタートアップおよび待機の両方のすべてのプロセッサーが活動化されます。試行期間を停止して再開することはできません。

このフィーチャーを実際に使用してみる場合は、『試行期間としての待機プロセッサーの活動化』を参照してください。

Capacity Upgrade on Demand (CUoD) のソフトウェア・ライセンスの考慮事項: 多くのビジネス・パートナーが、サーバーで使用可能なプロセッサー・フィーチャー・コードのシステム値を使用して、ソフトウェアのソフトウェア・ライセンス料金を設定しています。このシステム値は待機プロセッサーがあるサーバーでも使用できますが、プロセッサー・フィーチャー・コード (プロセッサー機能コードのシステム値) は、活動化される待機プロセッサーの数にかかわらず同じです。

ソフトウェア・ライセンス料金をサーバー上のプロセッサーの数を基にしているソフトウェア・プロバイダーは、従来から「穏やかな (″soft″)」承諾のアプローチをとっています。プロセッサーの活動化ごとに、プロセッサーの数に基づいてライセンス交付されるソフトウェアがあるサーバーに常駐するソフトウェアに関連した、必要なソフトウェア・ライセンス料金を通知して支払うのは、お客様の責任で行われています。

プロセッサーを活動化してもソフトウェア層は変わりません。

注: プロセッサー・オン・デマンド活動化フィーチャーの注文が、いずれかの IBM コンフィギュレーター経由で行われた場合、お客様のサーバーに現在インストールされている、プロセッサー単位でライセンス交付されるソフトウェア・プロダクトの IBM ソフトウェア・ライセンスの追加料金が請求されます。

プロセッサー・オン・デマンド活動化コード: 待機プロセッサーの一部またはすべてを永続的に活動化することにした場合は、プロセッサー・オン・デマンド (POD) 活動化フィーチャーを 1 つ以上注文して購入する必要があります。

注文が行われると、注文記録は、ご使用のサーバーから収集される重要プロダクト・データ (VPD) と組み合わされます。この情報は、ご使用のサーバーに固有の POD 活動化コードを生成するのに使用されます。これは、ユーザーが待機プロセッサーを活動化するために使用するキャパシティー・ライセンス・キーと考えることができます。

この活動化コードは、すぐに利用できるように、IBM Web サイトに掲載されます。これは、注文にサーバーから収集された必要な VPD が伴っていれば、通常は 1 労働日 (24 時間) 以内に行われます。生成された活動化コードには、次の iSeries Capacity Upgrade on Demand Web サイトで、ご使用のシステム・タイプとシリアル番号を使用してアクセスすることができます。

http://www.ibm.com/servers/eserver/iseries/hardware/ondemand 

POD 活動化フィーチャーの注文方法と POD 活動化コードの入手方法についての説明は、『Capacity Upgrade on Demand (CUoD) の注文』を参照してください。

Electronic Service Agent の要件: Capacity Upgrade on Demand の購入を注文すると、 IBM はユーザーの注文情報とサーバーの重要プロダクト・データ (VPD) を組み合わせて、ご使用のサーバーの待機プロセッサーのアンロックと活動化に必要な活動化コードを作成します。

VPD 情報は Electronic Service Agent を使用して電子的に IBM に送信されます。この Electronic Service Agent は、 IBM の包括的な技術サービスで IBM iSeries 専用のサポート・イニシアチブである、エクストリーム・サポートの一部です。 Electronic Service Agent は無料のライセンス・プログラム (5798-RZG) で、ご使用のサーバーに置かれ、イベントをモニターし、サーバーのインベントリー情報をお客様が定義できるタイム・テーブルに基づいて定期的に IBM に送信することを目的としています。

Electronic Service Agent のインストールも含めた詳細な説明は、 Electronic Service Agent for iSeriesUser's Guide を参照してください。

Capacity Upgrade on Demand (CUoD) の準備

Capacity Upgrade on Demand を使用すると、サーバーの運用を中断することなくプロセッサー・キャパシティーを追加することができます。ただし、そのようにシームレスに新規キャパシティーを組み込むためには、注文する前にご使用のサーバーを準備する必要があります。

Capacity Upgrade on Demand (CUoD) によるキャパシティー・プランニング
このトピックには、 CUoD を使用可能なサーバー・モデルのキャパシティー・プランニングを行う際の考慮事項と情報資源が記載されています。

プロセッサーの活動化時の決定
このトピックでは、資源の稼働率の傾向をモニターできるツールを説明し、どのようなときに追加キャパシティーが必要になるかを示しています。また待機プロセッサーがあるサーバーの CPU 稼働率がツールによってどのように報告されるかを説明しています。

Capacity Upgrade on Demand (CUoD) の環境のセットアップ
追加キャパシティーが必要なときに、新しいプロセッサー・キャパシティーを組み込むためにご使用のサーバーの準備を行い、追加キャパシティーを注文できる状態にします。

Capacity Upgrade on Demand (CUoD) によるキャパシティー・プランニング: 待機プロセッサーがあるサーバーのキャパシティー・プランニングで使用する手順と情報資源は、他の iSeries モデルのサイジングの際に使用するものと基本的には同じです。サーバーの必要なキャパシティーを決定する際に使用できるツール、情報資源、およびオファリングの包括的なセットは、待機プロセッサーがあるサーバーをサポートできるように更新されています。

キャパシティー・プランニングに役立つ情報資源として、次を参照してください。

サーバーの拡張時および拡張方法の決定
このトピックでは、キャパシティー・プランニングと資源稼働率の傾向の識別に使用できるいくつかのツールの概要を説明しています。

iSeries Benchmark Center
この IBM Web サイトには、アプリケーション環境のベンチマークを行う際に役立つ情報があります。

iSeries Solutions Center — Capacity Planning Services
この IBM コンサルティング・サービスを、お客様の業務の増大する要求に合うサーバー・ソリューションを計画する際にお役立てください。

注： Workload Estimator は CUoD をサポートしていますが、アクティブなプロセッサーの推奨数の見積もりのみを戻します。待機プロセッサーによって提供される追加キャパシティーは、推奨数には表されません。

プロセッサーの活動化時の決定: Capacity Upgrade on Demand を使用すると、ワークロードのために資源を追加する必要があるときに、プロセッサーをシステムに追加することができます。ご使用のシステムの CPU 稼働率と CPU 稼働率の傾向をモニターして、追加プロセッサーいつ活動化してどれだけのプロセッサーが必要になるかを決定する必要があります。

CPU 稼働率情報の報告に使用できるパフォーマンス・ツールは多数あります。特に、 PM/400 を使用すると資源の稼働率の傾向を識別することができ、また iSeries ナビゲーターのモニターを使用すると、資源の使われ方のより詳細な情報と、稼働率が事前定義クリティカル・レベルに達したときのアラートを得ることができます。

待機プロセッサーがあるサーバーの CPU 稼働率の測定

CPU 稼働率を報告するシステム機能が、使用可能なすべてのプロセッサーの平均使用量を計算するときは、待機プロセッサーを CPU キャパシティーの総量に含めません。 CPU 稼働率のパーセンテージを報告するさまざまなシステム機能では、待機プロセッサーはアクティブとして見なされません。使用された CPU キャパシティーのパーセンテージ (iSeries ナビゲーターでは CPU 稼働率 CPU パーセント) は、経過時間内にプロセッサーがアクティブだった時間の量に基づいて計算されたメトリックです。これは通常、パーセンテージで報告され、 100% は経過時間全体を通してプロセッサーがビジーだったことを示します。複数のプロセッサーがある場合は、すべてのプロセッサーの平均使用量になるように CPU 時間が調整されて、常に稼働率は使用可能な全キャパシティーのパーセンテージとして報告されます。

対話型の能力は、購入した対話型のフィーチャーで決まります。この機能は待機プロセッサーの数には影響を受けません。また、待機プロセッサーがアクティブになってもこの機能は変わりません。対話式の能力のパーセンテージとして報告される対話式稼働率は、 Capacity Upgrade on Demand テクノロジーには影響されません。 iSeries ナビゲーターでは、このメトリックを CPU 稼働率 (対話型のフィーチャー) と呼びます。

対話型 CPU 稼働率も、全システム CPU のパーセンテージとして報告されます。マネージメント・セントラルでは、このメトリックを CPU 稼働率 (対話型ジョブ) と呼びます。このメトリックは、 CUoD を使用するサーバーに対しても上記のシステム CPU 稼働率の場合と同じように使用されます。

Capacity Upgrade on Demand (CUoD) の環境のセットアップ: プロセッサー・オン・デマンド (POD) 活動化コードを注文する前に、追加キャパシティーの注文および組み込みのための環境を準備する必要があります。

注文の準備

POD 活動化コードをご購入の際は、ご使用のサーバーの重要プロダクト・データ (VPD) も提供していただく必要があります。この方法としては、ファクシミリを使用して手動で行う方法と、 Electronic Service Agent を使用して電子的に行う方法があります。 Electronic Service Agent を使用して VPD データを送信すると、手動による方法の場合の遅延がなくなり、 POD 活動化コードは注文受信後 24 時間以内に掲示されます。ご注文の送信と処理に要する時間を考慮に入れて、追加プロセッサーを試行ベースで使用してパフォーマンスの障害が起こらないようにする必要があります。

VPD データを電子的に送信する場合は、『Electronic Service Agent のセットアップ』を参照してください。

追加キャパシティーの準備

活動化されるプロセッサーをご使用のサーバーが完全に使用できるようにするには、次の準備を行う必要があります。

- I/O (入出力) の調整を行う。
- メモリーのアップグレードを行う。
- 論理区画 (LPAR) を準備する。

LPAR を使用するサーバーでは、待機プロセッサーは常に基本区画に関連付けられます。すべてのプロセッサーを 1 区画に割り当てる必要があります。

Electronic Service Agent のセットアップ: Electronic Service Agent を使用して VPD データを送信することができます。 Electronic Service Agent をセットアップするには、次のステップで行います。

1. Electronic Service Agent をインストールします。

   セットアップとインストールの説明については、 Electronic Service Agent for iSeries User's Guide を参照してください。
2. TCP/IP がセットアップされていて開始済みであることを確認します。

Electronic Service Agent の詳細については、『Electronic Service Agent の要件』を参照してください。

Capacity Upgrade on Demand (CUoD) の注文

新規サーバー用、モデルのアップグレード用、またはインストール済みサーバー用にプロセッサー・オン・デマンド (POD) 活動化フィーチャーを注文することができます。新規サーバーまたはモデル・アップグレードの場合、 POD 活動化フィーチャーを 1 つ以上注文に含めることができます。この場合、POD 活動化コードは、そのサーバーがお客様に配送される前に入力されます。

インストール済みサーバーの場合、待機プロセッサーの一部またはすべてを永続的に活動化することにした場合は、 POD 活動化フィーチャーを 1 つ以上注文し、その結果として得られる POD 活動化コードを使用して待機プロセッサーを活動化する必要があります。

注:

1. ご注文を処理してそれに関する POD 活動化コードを掲示するまで、数日間要する場合があります。追加キャパシティーの永続的活動化のご注文の対応が完了するまでの間、 14 日間の試行期間として待機プロセッサーを活動化して、ワークロード要件を満たすことができます。
2. ご注文にその他のフィーチャーが含まれていなければ、 POD 活動化フィーチャーのご注文の処理はさらに早くなります。

POD 活動化フィーチャーを 1 つ以上注文するには、次のステップで行います。

1. 活動化する待機プロセッサーの数を決定します。
   これについては、『プロセッサーの活動化時の決定』トピックを参照してください。
2. IBM ビジネス・パートナーまたは IBM 営業担当員に連絡して、 POD 活動化フィーチャーを 1 つ以上ご注文ください。米国内のお客様は、次の IBM Web サイトで注文することもできます。

   http://www.ibm.com
3. ご使用のサーバーの重要プロダクト・データを IBM に送信します。
   ご注文が処理される前に、ご使用のサーバーから収集された重要プロダクト・データ (VPD) とご注文が組み合わされます。この情報を基に、ご使用のサーバーに固有の POD 活動化コードが生成されます。 POD 活動化コードはお客様宛てにメールで送られますが、すぐに利用できるように iSeries Capacity Upgrade on Demand Web サイトにも掲示されます。
4. ご使用のサーバーの POD 活動化コードを入力して、 待機プロセッサーを永続的に活動化します。

IBM への重要プロダクト・データの送信: プロセッサー・オン・デマンド (POD) 活動化フィーチャーを 1 つ以上注文するときは、ご使用の iSeries サーバーの重要プロダクト・データ (VPD) を IBM に提供する必要があります。 VPD を収集して IBM に送るには、 Electronic Service Agent(TM) を使用するか、またはデータを印刷してファックスで送ります。

Electronic Service Agent を使用して VPD を収集して送信するには、次のステップで行います。

1. Electronic Service Agent がセットアップされていることを確認します。
2. Electronic Service Agent ウィザードを開始します。
3. 必要な VPD 情報を収集して送信するには、**「ハードウェア (Hardware)」**を選択します。

VPD を印刷してファックスで送るには、次のステップで行います。

1. システム・コンソールから、コマンド行で STRSST と入力して、システム保守ツール (SST) を開始します。 SST にサインオンします。

   注: システム保守ツールを使用するには、システム・キャパシティー管理者特権がある有効な保守ツール・ユーザー ID が必要です。
2. オプション 6 (システム・キャパシティーの処理 (Work with System Capacity)) を選択して Enter を押します。
3. オプション 1 (システム・キャパシティー情報の表示 (Display System Capacity Information)) を選択して Enter を押します。「システムの永続キャパシティーの表示 (Display permanent system capacity)」画面が表示されます。
4. F6 を押して情報を印刷します。
5. 次の情報を含むファクシミリ文書を作成します。
   - ファクシミリ送信先情報:
     - **宛先:** Capacity on Demand Administrator (507-253-7019)
     - **FAX 番号:** 507-253-4553
     - **場所:** Rochester, Minnesota
   - ファクシミリ発信元情報:
     - **お客様の名前:**
     - **お客様の連絡窓口:**
     - **お客様の住所:**
     - **お客様の電話番号:**
     - **お客様の FAX 番号:**
6. VPD を次の FAX 番号にファックスで送ります。
   507-253-4553

待機プロセッサーの活動化

ご使用のサーバーにすでにインストールされている 1 つ以上の待機プロセッサーを、試行または永続ベースで活動化することができます。

試行期間としての待機プロセッサーの活動化
試行期間の 14 日間、ご使用のサーバーにインストールされているすべての待機プロセッサーを活動化することができます。これによりユーザーは、追加キャパシティーを購入するかどうかを決定する前にそれが実際の環境でどのように役立つかを評価することができ、また、追加キャパシティーの永続的活動化のご注文の対応が完了するまでの間のワークロードのピーク要件を満たすことができます。

待機プロセッサーの永続的活動化
指定した数の待機プロセッサーを永続的に活動化することができます。

試行期間としての待機プロセッサーの活動化: 試行期間としてご使用のサーバーにインストールされているすべての待機プロセッサーを活動化するには、以下のステップを実行します。

1. システム・コンソールから、コマンド行で STRSST と入力して、システム保守ツール (SST) を開始します。 SST にサインオンします。

   注: システム保守ツールを使用するには、システム・キャパシティー管理者特権がある有効な保守ツール・ユーザー ID が必要です。
2. オプション 6 (システム・キャパシティーの処理 (Work with system capacity)) を選択して Enter を押します。「システム・キャパシティーの処理 (Work with System Capacity)」画面が表示されます。
3. オプション 3 (システム一時キャパシティーの処理 (Work with temporary system capacity)) を選択して、 Enter を押します。「システム一時キャパシティーの活動化の開始の確認 (Confirm start temporary system capacity activation)」画面が表示されます。
   注: システム一時キャパシティーの活動化が使用中、またはすでに使用されていた場合は、このオプションは表示されません。
4. Enter を押してシステム・キャパシティーの活動化を確認します。
5. 「システム・キャパシティーの処理 (Work with System Capacity)」画面を終了します。
6. サーバーが区画に分割されている場合は、新規キャパシティーを使用する前に、論理区画に対して使用可能なプロセッサーを新規に割り当てなければなりません。

   活動化されたプロセッサーを論理区画に割り当てる方法については、『処理能力の動的な移動』を参照してください。
7. サーバーが区画に分割されていない場合は、次のステップを行います。
   a. オプション 5 (システム区画の処理 (Work with system partitions)) を選択して、 Enter を押します。
   b. オプション 3 (区画構成の処理 (Work with partition configuration)) を選択して Enter を押します。
   c. PRIMARY の隣に 2 (区画処理資源の変更 (Change partition processing resources)) を入力して、Enter を押します。
   d. プロセッサーの新規数の値を入力します。これは活動状態のプロセッサーの総数を表すものです。

      注: 画面の下部に構成エラーが表示された場合は、デフォルトの基本区画が変更されている (論理区分化が活動状態で使用されている) ことを示しており、この場合は論理区画計画を参照して、活動化されたプロセッサーをサーバーに適切に割り当てる必要があります。
   e. Enter を押して変更を確認します。
8. SST を終了します。

これで新規キャパシティーを使い始めることができます。

注:

1. 14 日間が経過したとき、または新規のプロセッサー・オン・デマンド (POD) 活動化コードが入力されると、試行は自動的に終了します。
2. ご使用のサーバーが区画に分割されている場合、試行期間の終わりに待機プロセッサーを基本区画に戻さなければなりません。

待機プロセッサーの永続的活動化: プロセッサー・オン・デマンド (POD) 活動化フィーチャーを 1 つ以上購入すると、待機プロセッサーを活動化するための POD 活動化コードが提供されます。

待機プロセッサーの一部またはすべてを永続的に活動化するには、以下のステップを実行します。

1. POD 活動化コードを次のようにして検索します。
   a. 次の iSeries Capacity Upgrade on Demand Web サイトにアクセスします。

      http://www.ibm.com/servers/eserver/iseries/hardware/ondemand 
   b. ご使用のサーバーのシステム・タイプとシリアル番号を入力します。
   c. Web サイトで表示される POD 活動化コードを記録します。
2. コマンド行で STRSST と入力して、システム保守ツール (SST) を開始します。 SST にサインオンします。

   注: システム保守ツールを使用するには、システム・キャパシティー管理者特権がある有効な保守ツール・ユーザー ID が必要です。
3. オプション 6 (システム・キャパシティーの処理 (Work with system capacity)) を選択して Enter を押します。
4. オプション 2 (永続的システム・キャパシティーの活動化 (Activate permanent system capacity)) を選択して Enter を押します。
5. プロセッサー・オン・デマンド活動化コードのフィールドに POD 活動化コードを入力して、 Enter を押します。「システム・キャパシティーの活動化の確認 (Confirm Activate System Capacity)」画面が表示されます。
6. Enter を押してシステム・キャパシティーの活動化を確認します。
7. 「システム・キャパシティーの処理 (Work with System Capacity)」画面を終了します。
8. サーバーが区画に分割されている場合は、新規キャパシティーを使用する前に、論理区画に対して使用可能なプロセッサーを新規に割り当てなければなりません。

   活動化されたプロセッサーを論理区画に割り当てる方法については、『処理能力の動的な移動』を参照してください。
9. サーバーが区画に分割されていない場合は、次のステップを行います。
   a. オプション 5 (システム区画の処理 (Work with system partitions)) を選択して、 Enter を押します。
   b. オプション 3 (区画構成の処理 (Work with partition configuration)) を選択して Enter を押します。
   c. PRIMARY の隣に 2 (区画処理資源の変更 (Change partition processing resources)) を入力して、 Enter を押します。
   d. プロセッサーの新規数の値を入力します。これは活動状態のプロセッサーの総数を表すものです。 **注:** 画面の下部に構成エラーが表示された場合は、デフォルトの基本区画が変更されている (論理区分化が活動状態で使用されている) ことを示しており、この場合は論理区画計画を参照して、活動化されたプロセッサーをサーバーに適切に割り当てる必要があります。
   e. Enter を押して変更を確認します。
10. SST を終了します。

注: POD 活動化コードを入力すると、待機プロセッサーは即時に活動化されます。ただし、サーバーが POD 活動化コードを保管するためには、サーバーを 15 分間実行しなければなりません。 POD 活動化コードを入力してサーバーを 15 分間より短時間実行してシャットダウンすると、サーバーを開始するときにその POD 活動化コードを再入力しなければならない場合があります。

これで新規キャパシティーを使い始めることができます。

シナリオ: Capacity Upgrade on Demand (CUoD)

お客様のワークロードの必要に応じて、 Capacity Upgrade on Demand を使用して待機プロセッサーを活動化することができます。次のシナリオは、このフィーチャーの計画、注文、および利用のステップを示したものです。

1. フィーチャー・コード 2416 の 840 モデル・サーバーが、 8 つのアクティブ・プロセッサーと 4 つの待機プロセッサーで運用されています。サーバーのワークロードが増大するに従って、使用可能な CPU 資源の稼働率が使用可能キャパシティーの 70% 近くなったりそれを超えるようになっています。資源を追加する必要があると見込んだ管理者は、待機プロセッサーのいくつかを活動化することにします。
2. 管理者はプロセッサーを活動化する前に、Capacity Upgrade on Demand のためのサーバーの準備を行います。これには、追加プロセッサーがいくつ必要になるかを知るための傾向分析、追加プロセッサーを利用するためのサーバーの準備、および新規キャパシティーの注文の準備を行うことが必要です。
3. 管理者は、追加プロセッサーを活動化することによる利点を調査するために、試行期間としてプロセッサーを活動化することにします。試行期間は、連続する 14 日間です。
4. 管理者は、追加プロセッサーを活動化したことによりパフォーマンスが向上したのでプロセッサーを永続的に購入しても大丈夫であると判断し、 IBM 営業担当員またはビジネス・パートナーに連絡、あるいは www.ibm.com にアクセスして、プロセッサー・オン・デマンド (POD) 活動化コードを 4 つ注文します。
5. IBM 営業担当員はその注文を IBM コンフィギュレーターに入力し、その注文の対象のサーバーの重要プロダクト・データ (VPD) の送信を求める通知を受け取ります。 VPD データは IBM にファックスで送ることができますし、 Electronic Service Agent を使用して電子的に送信することもできます。
6. 管理者は Web サイトから POD 活動化コードを検索して、 永続的キャパシティーを活動化します。これには、ターゲット・サーバーの POD 活動化コードを入力して、各プロセッサーを論理区画に割り当てることが必要です。

これでこの 840 モデルの 12 プロセッサーすべてが使用可能になります。

パフォーマンス管理ストラテジーの選択

よいパフォーマンス管理ストラテジーを開発することは、システムのパフォーマンスを管理するのに役立ちます。パフォーマンス管理ストラテジーは、ユーザーがパフォーマンスの管理に費やすことのできる時間的な余裕がどれくらいあるかに大きく依存します。小さな会社で働いている場合は、ユーザーはさまざまな異なるビジネスの局面を管理することがあり、パフォーマンスの管理に多くの時間を費やすことができません。多くの大きな企業では、システムの調整を行い、効率的に動作するようにするため、パフォーマンスの専門家を採用します。

基本的なパフォーマンス管理ストラテジーを決定し、どのパフォーマンス・アプリケーションを使用するかを識別するには、ユーザーの企業を、小規模、中規模、大規模の 3 つのカテゴリーに分類します。ビジネス・リソースはそれぞれのサイズによってさまざまで、管理戦略もそれに応じて異なります。

小規模ビジネス

小規模ビジネスでは多くの場合、パフォーマンスの管理に費やすことのできるリソースは、大規模ビジネスの場合よりも少なくなります。そのため、可能であればなるべく自動化します。パフォーマンス・データのコンパイルおよびレポートの生成を行う IBM にデータを直接送信するために、PM/400 を使用します。これは時間の節約になるだけでなく、ユーザーの iSeries サーバーにアップグレードが必要な場合、IBM が提案することもできます。

小規模ビジネスで推奨されるパフォーマンス・アプリケーションを以下にリストします。

収集サービス
今後の分析のために、ユーザーが定義した間隔でサンプル・データを収集します。

グラフ・ヒストリー
収集サービスで収集されたパフォーマンス・データを表示します。

PM/400
システム・パフォーマンス・データの収集、アーカイブ、および分析を自動化します。

Performance Tools
システム・パフォーマンス情報の取得、分析、および保守を行います。

モニター
iSeries システム・パフォーマンスのグラフィカル表現を監視し、事前定義されたイベントまたは状態に対して自動応答します。

中規模ビジネス

ほとんどの場合、中規模ビジネスは小規模ビジネスに比較して、多くのリソースをパフォーマンスの管理に費やすことができます。それでも、可能な限り自動化することが望ましく、 PM/400 を使用することによって利益が得られます。

中規模ビジネスで推奨されるパフォーマンス・アプリケーションを以下にリストします。

収集サービス
今後の分析のために、ユーザーが定義した間隔でサンプル・データを収集します。

グラフ・ヒストリー
収集サービスで収集されたパフォーマンス・データを表示します。

PM/400
システム・パフォーマンス・データの収集、アーカイブ、および分析を自動化します。

Performance Tools
システム・パフォーマンス情報の取得、分析、および保守を行います。

モニター
iSeries システム・パフォーマンスのグラフィカル表現を監視し、事前定義されたイベントまたは状態に対して自動応答します。

Performance Explorer
特定のアプリケーションまたはシステム・リソースに関する詳細な情報を収集します。

大規模ビジネス

大規模ビジネスでは、パフォーマンスの管理にリソースを費やすことができます。

大規模ビジネスで推奨されるパフォーマンス・アプリケーションを以下にリストします。

収集サービス
今後の分析のために、ユーザーが定義した間隔でサンプル・データを収集します。

グラフ・ヒストリー
収集サービスで収集されたパフォーマンス・データを表示します。

PM/400
システム・パフォーマンス・データの収集、アーカイブ、および分析を自動化します。

Performance Tools
システム・パフォーマンス情報の取得、分析、および保守を行います。

モニター
iSeries システム・パフォーマンスのグラフィカル表現を監視し、事前定義されたイベントまたは状態に対して自動応答します。

Performance Explorer
特定のアプリケーションまたはシステム・リソースに関する詳細な情報を収集します。

iDoctor for iSeries
システムおよびアプリケーションのパフォーマンスを向上させるため、トレース・データを分析します。

Performance Trace Data Visualizer (PTDV)
Java アプリケーションからのトレース・データを表示します。

パフォーマンスを管理するための環境のセットアップ

iSeries サーバーには、システム・パフォーマンス・データを定期的に収集し、システムのパフォーマンスの傾向および潜在的な問題をモニターするためのいくつかのツールが含まれています。ユーザーの個別の要件や環境によって、投入するために選択するツール、および設定する構成の選択が決定されます。システムを効果的に設定すると、システムの成長に合わせた正確なキャパシティー・プランニングが可能になり、パフォーマンス上の問題が発生したときにも解決することができます。

システム・パフォーマンスのデータを収集、モニター、および分析するためのツールとその構成について、下記のトピックを使用して確認してください。

収集サービス
収集サービスは、システム・パフォーマンス・データの周期的な収集を管理します。このツールは定期的にデータを収集し、収集オブジェクトと呼ばれるアーカイブを作成します。これらの収集オブジェクトは、いくつかのツールから直接アクセスされたり、または、独自のカスタム照会または他のツールおよびレポートによる分析のためのデータベース・ファイルのセットに変換されます。収集サービスは主に他のアプリケーションに対してデータを提供するため、使用するそのアプリケーションは、データをどの程度の周期で収集するか、収集するデータのタイプ、およびシステム上にデータを保持する時間を含めた、構成の選択に大きく影響します。

Performance Management/400
PM/400 は、所有権のないパフォーマンス・データを取得し、保管および専門的な分析を行うために IBM にデータを送信するために、収集サービスを使用します。これでお客様が保存し保持する必要はなくなります。その後、Web ブラウザーを使用して、システム・パフォーマンスに関する詳細な報告書、および推奨にアクセスすることができます。

iSeries ナビゲーターのモニター
iSeries ナビゲーターに含まれているモニターは、関心のある特定のシステム・パフォーマンスの要素を追跡するために、収集サービスのデータを使用します。さらに、CPU 稼働率のパーセンテージやジ

ョブの状況など、特定のイベントが生じた場合に、指定されたジョブを実行できます。このトピックを使用して、これらのモニターの使い方、およびシステムにセットアップする方法を確認してください。

iSeries パフォーマンスの管理

パフォーマンスの管理を適切に行うことは、システムが効率的にリソースを使用し、 iSeries サーバーがユーザー・ニーズや業務ニーズに対して最良のサービスを提供することを確実にします。さらに、効果的なパフォーマンス管理は、システム内の変更に迅速に対応することを可能にし、高価なアップグレードや保守費用を先送りすることにより、コストを節約することができます。

システムのパフォーマンスに影響のある要素を理解することは、問題に対応し、より良い長期計画を作成するのに役立ちます。効果的な計画により、開発によるパフォーマンスの問題の可能性を回避し、現在および増大するワークロードを処理するシステム能力を確保することができます。

システムのパフォーマンスの保守、およびパフォーマンス上の問題を解決する方法について、下記のトピックを使用して確認してください。

パフォーマンスの追跡
時間をかけてシステム・パフォーマンスを追跡すると、ユーザーのシステムの発展を計画することが可能になり、パフォーマンス上の問題の原因を切り分けて、原因を識別するのに役立つデータを得ることができます。どのアプリケーションを使用するのか、また定期的にパフォーマンス・データを収集する方法を確認してください。

パフォーマンスの問題の調査
パフォーマンス上の問題の識別、および解決に役立つさまざまな選択可能なオプションがあります。パフォーマンス上の問題の原因を見つけるのに役立つ、使用可能なツールおよびレポートの使用方法について確認してください。

パフォーマンス・データの表示
パフォーマンス・データを収集した後に、ユーザーの目的に合った最も適切なツールを使用してデータを表示する方法を確認してください。

パフォーマンスのチューニング
パフォーマンス上の問題を識別したなら、問題を修正するためにシステムを調整します。

e-business パフォーマンスの管理
e-business 環境のパフォーマンスの管理には、 iSeries 管理者にとっての新しい問題がいくつか伴います。 e-business アプリケーションのパフォーマンスの計画、追跡、および改善に役立つ情報と資源については、このトピックを参照してください。

パフォーマンスの追跡

iSeries サーバーのシステム・パフォーマンスを追跡すると、傾向を見極めて、システム構成の調整とシステムのアップグレードの時期と方法について最良の選択をする手掛かりとすることができます。さらに、問題が発生したときには、パフォーマンス上の問題の原因の範囲を絞り込み、適切な解決策を見つけるために、その前後のパフォーマンス・データを手に入れることは不可欠です。

iSeries サーバーには、パフォーマンスの傾向を追跡し、 iSeries パフォーマンス・データのヒストリー・レコードを保持するいくつかのアプリケーションがあります。こうしたアプリケーションのほとんどは収集サービスが収集したデータを使用します。収集サービスを使用して、以下の領域で傾向を監視できます。

- システム・リソースの使用状況の傾向。この情報を使用して、システム構成の変更やアップグレードを計画し、明確に調整することができます。
- 構成中の物理的構成要素に対するストレスの識別
- ピーク時と通常時、対話型ジョブとバッチ・ジョブのシステム・リソース使用量のバランス
- 構成変更。収集サービスのデータを使用して、ユーザー・グループの追加、対話型ジョブの増加、およびその他の変更の影響を正確に予測することができます。
- システム上の他の活動に問題を引き起こしている恐れのあるジョブの識別

以下のツールがシステム・パフォーマンスをモニターするのに役立ちます。

収集サービス
収集サービスはユーザーが定義した時間間隔でパフォーマンス・データを集め、この情報をシステムの収集オブジェクトに保管します。モニター、グラフ・ヒストリー、PM/400、および Performance Tools ライセンス・プログラムの多くの機能など、他のツールの多くが収集オブジェクトのデータに依存しています。

グラフ・ヒストリー
グラフ・ヒストリーは指定された期間にわたって収集サービスが収集したパフォーマンス・データをグラフィカル・ユーザー・インターフェース (GUI) で表示します。表示できる期間の長さは収集オブジェクトをどれだけ長く保存しているかと PM/400 を使用しているかどうかによります。

PM/400
PM/400 はシステム・パフォーマンス・データの収集、保存、および分析を自動化し、システム資源およびキャパシティーを管理するのに役立つ明確な報告書を戻します。

パフォーマンスの問題の調査

パフォーマンスを収集または分析するツールの多くはトレース・データとサンプル・データのいずれかを使用します。収集サービスはさまざまなシステム・リソース上で定期的にサンプル・データを収集します。いくつかのツールがこのサンプル・データを分析し、それに基づいて報告し、これを使用してシステム・リソースの使用状況の全体像をつかみ、多くの共通なパフォーマンスに関する質問に答えることができます。より詳細なパフォーマンス情報については、いくつかのツールがトレース・レベルのデータを生成します。しばしば、トレース・レベルのデータは、システム上のジョブとアプリケーションの動作とリソース使用量について詳細な情報を提供してくれます。 Performance Explorer および STRPFRTRC コマンドがトレース・データを生成する 2 つの共通ツールです。

たとえば、システムの実行が遅い場合、システム・モニターを使用して問題を探します。 CPU 稼働率が高いことが分かれば、異常に大量のリソースを使用しているように見えるジョブを識別できるかもしれません。そうすれば、構成を変更することによって、問題を正せるかもしれません。しかしながら、問題によっては追加情報が必要です。そのジョブのパフォーマンスについて詳細な情報を得るため、 Performance Explorer セッションを開始し、iSeries システム上でのそのジョブの動作について詳細な情報を集め、もしかすると問題を引き起こしているプログラムに変更を加えることができます。

パフォーマンス・データの収集についてもっと知るために、以下のトピックを使用してパフォーマンス管理アプリケーションをいつどのように使うか学んでください。

パフォーマンスの問題の識別
パフォーマンス上の問題の識別に関係した共通ステップを考えます。

共通のパフォーマンスの問題の識別と解決
多くのさまざまなパフォーマンス上の問題が iSeries システムの共通域にしばしば影響します。共通域の問題の調査解決方法を考えます。

システム・パフォーマンス情報の収集
収集サービスは定期的にシステム・パフォーマンスに関する情報を収集します。しばしば、パフォーマンス・データの分析はこの情報から始まります。

システム・リソース使用状況についての情報の収集
いくつかのツールが CPU、ディスク・スペース、対話式能力、および多くのほかの要素のようなリソースがどう使用されているかをモニターします。問題のある領域を識別するのにこれらのツールを使用できます。

アプリケーションのパフォーマンスについての情報の収集
いくつかの理由でアプリケーションの実行が遅くなることがあります。 OS/400 に組み込まれているいくつかのツールや他のライセンス・プログラムを使用してもっと情報を得ることができます。

シナリオ:アップグレードまたはマイグレーション後にシステム・パフォーマンスを改善する
このシナリオは、システムをアップグレードまたは移行したところ、以前よりも実行速度が遅くなったように思える、というものです。このシナリオはパフォーマンス上の問題を識別して修正するのに役立ちます。

パフォーマンスの問題の識別
パフォーマンスの問題を識別しようとする際には、ハードウェア構成がそのワークロードをサポートするのに適切かどうかについて評価することが重要になります。 CPU の能力は充分ですか ? 主記憶域はさまざまなアプリケーションを処理するのに充分ですか ? これらの疑問に対して、たとえばシステム負荷のモデル化技法を使用してまず回答を出しておくと、あとで不必要な作業をしないで済みます。

問題の症状と達成目標を理解することにより、分析担当者は問題の原因を説明できる仮説を立てることができます。分析担当者は、システム・パフォーマンスを測定するために、OS/400 オペレーティング・システムおよび Performance Tools ライセンス・プログラムで使用可能なコマンドおよびツールを使用することができます。

測定したデータを検討することは、問題をさらに定義し、仮説が妥当であるか破棄すべきものであるかを判断するのに役立ちます。 1 つまたは複数の明白な原因が分離されると、ソリューションを提案することができます。一度に 1 つのソリューションを処理するようにすると、プログラムを再設計してテストすることが可能です。分析担当者のツールは多くの場合、ソリューションの効果を測定し、副次作用があるかどうかについても探し出すことができます。

最適なパフォーマンスを達成するには、重要なシステム・リソースの相互の関係を知り、それらのリソース（つまり CPU、ディスク、主記憶装置、および通信の場合のリモート回線）の間のバランスがとれるようにしなければなりません。これらのリソースは、それぞれがパフォーマンス低下の原因となる可能性があります。

システム・パフォーマンスを改善することは、それが対話式スループット、対話式応答時間、バッチ・スループット、あるいはそれらの組み合わせに対する改善であっても、単に活動レベルまたはプール・サイズを調整することからアプリケーション・コード自体を変更することまで、多くの形態を取ることが考えられます。この場合、活動レベルは、処理装置を獲得するために同時に競合し得るジョブの最大数を指定するサブシステムの 1 つの特性です。

共通のパフォーマンスの問題の識別と解決

iSeries サーバーでパフォーマンスの問題が発生すると、しばしば最初に影響を受けるのはシステムのある領域です。このようなシステム領域のパフォーマンスを調査するのに使用可能ないくつかの方法について次の表を参照してください。このような領域の多くはシステム・モニターのメトリックとして使用可能です。しかしながら、それらについての情報を見る他の方法もいくつかあります。

領域	説明	使用可能なツール
プロセッサー負荷	システム上にジョブが多すぎないかどうか、あるいは一部のジョブがプロセッサー時間を独占していないかどうかを判別します。	• 活動ジョブの処理 (WRKACTJOB) コマンド • システム活動の処理 (WRKSYSACT) コマンド (Performance Tools ライセンス・プログラムに含まれています) • iSeries ナビゲーターの実行管理機能 • iSeries ナビゲーター・システム・モニター内の CPU 使用状況メトリック
主記憶装置	ページ不在や待ち状態から不適格状態に移行しているトランザクションを調査します。	• ディスク状況の処理 (WRKDSKSTS) コマンド • iSeries ナビゲーター・システム・モニター内のディスク装置メトリック • システム状況の処理 (WRKSYSSTS) コマンド • iSeries ナビゲーターの実行管理機能の下にあるメモリー・プール機能
ディスク	アームが少なすぎないかどうか、あるいはアームが遅すぎないかどうかを判別します。	• ディスク状況の処理 (WRKDSKSTS) コマンド • iSeries ナビゲーター・システム・モニター内のディスク・アーム使用状況メトリック • Performance Tools のシステムと構成要素報告書
通信	遅い回線、回線上のエラー、あるいは特定回線への過剰なユーザーの集中を検出します。	• Performance Tools の構成要素報告書 • iSeries ナビゲーター・システム・モニター内の LAN 使用状況メトリック
IOP	使用率に偏りのある IOP がないかどうか、あるいは IOP が不足していないかどうかを判別します。	• Performance Tools の構成要素報告書 • iSeries ナビゲーター・システム・モニター内の IOP 使用状況メトリック
ソフトウェア	ロックと相互排他 (mutex) を調べます。	• Performance Tools のロック報告書 • Performance Tools のトレース報告書 • オブジェクト・ロックの処理 (WRKOBJLCK) コマンド • iSeries ナビゲーターの実行管理機能で疑わしいジョブを右マウス・ボタン・クリックし、**「詳細」** —> **「ロックされたオブジェクト」**を選択。

システム・パフォーマンス・データの収集

データの収集は、パフォーマンスを改善するための重要なステップです。パフォーマンス・データを収集するとき、応答時間やスループットを理解するために使用できるサーバーに関する情報を収集します。データ

を収集することは、ユーザーの作業を行うために関係のあるサーバー、または一連のサーバーのパフォーマンス状況を把握する手段です。データ収集により、後に行われるすべての比較および分析に備えた、コンテキストや開始点が提供されます。 初めてデータ収集を使用する時、ユーザーは将来の改善に対するベンチマークと、現在のパフォーマンスを改善する出発点を手にすることになります。 調整を行い、応答時間を改善し、システムが最高のパフォーマンスを得るのを助けるために収集するパフォーマンス・データを使用することができます。パフォーマンス上の問題の分析は、多くの場合 「何か変わったのか」という単純な質問から始まります。パフォーマンス・データはその質問に答えるのに役立ちます。

収集サービスを使用して、パフォーマンス・データの収集、パフォーマンス・データの作成 (CRTPFRDTA) コマンドでパフォーマンス・ファイルの作成、パフォーマンス・データの変換 (CVTPFRDTA) コマンドまたは iSeries の Performance Tools プラグインで現行リリースに交換、そしてパフォーマンス・データベース・ファイルの情報を使用して報告書作成または独自の照会を行うことができます。

パフォーマンス・データの詳細については、以下のものを参照してください。

収集サービス
分析のためにパフォーマンス・データを収集する方法と収集をカスタマイズする方法を参照してください。

パフォーマンス・データベース・ファイル
使用可能なパフォーマンス・データベース・ファイルの概要を調べ、各パフォーマンス・データベース・ファイルの詳細なフィールド・データを参照してください。

さらに Performance Management API を使用して、収集を開始、終了したり、コレクションを循環させたりすることができます。また、収集されたデータと関連するシステム・パラメーターを変更したり、検索したりすることもできます。他の iSeries パフォーマンス・ツールおよび技法の詳細については、『パフォーマンス』のトピックを参照してください。

システム・リソース使用状況についての情報の収集

iSeries サーバーとアプリケーションが使用可能なリソースを使用している状態をモニターおよび追跡するのに役立つツールが数多くあります。この情報を問題分析の出発点として使用し、傾向を識別し、キャパシティー・プランニングとシステムの成長を管理するのに役立ててください。

以下のトピックを参照して、これらのツールをいつどのように使うかを調べてください。

iSeries ナビゲーターのモニター
iSeries ナビゲーターに組み込まれたモニターはさまざまなメトリックの現在および最近のデータを提供します。さらにある種のイベントが発生したときに特定のアクションをとるように構成することができます。

OS/400 パフォーマンス・コマンド
OS/400 には、システム・パフォーマンスの管理および保守に役立つ複数の重要な機能があります。

PM/400
PM/400 は収集サービスを使用して所有権を主張されないパフォーマンス・データを集め、保存と専門家の分析のために IBM に送信します。これにより、ユーザー自身が保管および分析する必要がなくなります。ご使用のシステムのパフォーマンスおよび傾向分析について詳細な報告書と推奨事項を Web ブラウザーでアクセスできます。

アプリケーションのパフォーマンスについての情報の収集

アプリケーションのパフォーマンスについての情報の収集は、システム・パフォーマンスについての情報の収集と大きく異なります。アプリケーション情報の収集は Performance Explorer、Performance Trace Data Visualizer (PTDV)、および iDoctor のようなパフォーマンス・アプリケーションによってのみ行えます。別の方法として、 ジョブ・モニターを使用して個々のサーバーのパフォーマンスを追跡したり、Performance Tools を使用してサーバーのジョブを追跡および分析することにより、アプリケーション・パフォーマンスの概要を取得することもできます。

注: アプリケーションのパフォーマンス・データを収集するとシステムのパフォーマンスに目立って影響が出る場合があります。収集を開始する前に、他の収集オプションをすべて試したか確認してください。

Performance Explorer

このツールは、一般的なパフォーマンス・モニターを行うツールを使用していたのでは識別できないパフォーマンス問題の原因を見つけ出すのに役立ちます。コンピューター環境がサイズと複雑さの両面で拡大すると、当然パフォーマンス分析も同様に複雑になります。 Performance Explorer は、複合したパフォーマンスの問題に関するデータを収集することにより、そのような複雑さの拡大に対処しています。

Performance Explorer はプログラムのパフォーマンスを理解し改善したいと思っているアプリケーション開発者のために設計されています。複雑なパフォーマンスの問題を識別し明確にできるようにすることによって、パフォーマンス管理の知識のあるユーザーにも役に立ちます。

Performance Trace Data Visualizer for iSeries (PTDV)

このツールは、iSeries で稼働するアプリケーションのパフォーマンス分析に使用できる Java アプリケーションです。 PTDV は OS/400 オペレーティング・システムの Performance Explorer 機能と共に働き、分析者がプログラム・フローを見てトレース、ジョブ、スレッド、およびプロシージャーごとにまとめられた詳細 (CPU 時間、現在のシステム時刻、サイクル数、命令数など) を入手できるようにします。 Java アプリケーション・トレースをビジュアル化するとき、Java ロック動作についての情報と共に作成されたオブジェクトの数とタイプなどのような追加の詳細も表示できます。また、WebSphere Application Server が生成する Performance Explorer イベントもサポートします。 PTDV では、カラムのソート、データのエクスポートおよびさまざまなレベルでのデータ要約が可能です。

詳しくは、 Performance Trace Data Visualizer  Web サイトに進んでください。

iDoctor for iSeries

iDoctor の Performance Explorer Analyzer 機能には、システムおよびアプリケーションのパフォーマンスを改善しようとトレース・データを分析するために特に調整されたソフトウェア・ツールが入っています。この詳細な分析により、ディスク操作、CPU 稼働率、ファイル・オープン操作、マシン・インターフェース (MI) プログラム、待ち状態、ディスク・スペース使用量などの低レベル要約を取得できます。クライアント構成要素は、iSeries トレース・データを圧縮したりグラフィックに表示する機能を持つ iSeries ナビゲーター・プラグインです。

パフォーマンス・トレース開始 (STRPFRTRC) コマンド

OS/400 にはマルチプログラミングおよびトランザクション・データを収集するコマンドがあります。このコマンドは前のリリースで STRPFRMON が収集していたデータを収集します。このコマンドを実行後、トレース・ダンプ (DMPTRC) コマンドでデータをデータベース・ファイルにエクスポートできます。

トレース・データのダンプ: ダンプはシステムのパフォーマンスに影響を与えるため、いつトレース・データのダンプを行うかを決定するのは重要な問題です。 トレースのダンプ (DMPTRC) コマンドは、内部

トレース表内の情報をデータベース・ファイルに書き込みます。負荷のかかったシステムや、高優先順位の (対話式の) ジョブで、活動がピークに達しているときにダンプを行うのは好ましくありません。トレース・ダンプを後で行うこともできますが、データの存在を忘れないうちにダンプしておきたいものです。もし、何らかの理由でトレース表がクリアされるようなことがあれば、トレース・データが失われてしまうからです。しかし、ダンプをわずかに遅らせて、トレースのダンプ (DMPTRC) コマンドを使用すれば、ユーザーのパフォーマンスを維持することが可能です。

トレース・データをダンプするには、次のコマンドを実行します。

```
DMPTRC MBR(member-name) LIB(library-name)
```

コマンドを実行する際には、データの保管先としてメンバー名とライブラリー名を指定する必要があります。 収集サービスを使えば、トレース情報と同時に、サンプル・ベース・データの収集を行うことができます。これと同じように、サンプル・データとトレース・データを一緒に収集する場合には、それらのデータを保管するメンバーの名前が一致している必要があります。つまり、CRTPFRDTA TOMBR および TOLIB パラメーターで指定した名前と、 DMPTRC MBR および LIB パラメーターで指定した名前が一致していなければなりません。

シナリオ: アップグレードまたはマイグレーション後にシステム・パフォーマンスを改善する

最近 iSeries サーバーを最新のリリースにアップグレードしたとします。アップグレードが完了し、通常操作を再開した後は、システム・パフォーマンスは著しく低下しています。パフォーマンス上の問題の原因を突き止め、システムを通常レベルに復元したいとします。

変更点を分離する

オペレーティング・システムのアップグレード後にパフォーマンスの低下を招く問題がいくつかあります。OS/400 および Performance Tools ライセンス・プログラム (5722-PT1) に組み込まれているパフォーマンス管理ツールを使用して、パフォーマンス上の問題についての詳細情報を入手し、疑わしい問題を可能性のある原因へと絞り込むことができます。

1. CPU 稼働率を調べる。アップグレード後に、ジョブは、必要なリソースの一部にアクセスできなくなる場合があります。これは、許容できないほどの量の CPU 資源が 1 つのジョブで消費されるという結果を招く場合があります。
   - WRKSYSACT、WRKSYSSTS、WRKACTJOB、または iSeries ナビゲーターのシステム・モニターを使用して、 CPU の合計稼働率を検出する。
   - CPU 稼働率が高い (例: 90% を超えている) 場合は、アクティブ・ジョブが使用している CPU の量を調べる。 1 つのジョブで 30% を超える CPU 資源が消費されている場合は、欠落ファイルを呼び出しているか、オブジェクトが欠落していることが考えられます。その場合は、ベンダーに連絡してベンダー提供のプログラムを入手するか、ジョブの所有者またはプログラマーに連絡して追加プログラムを入手してください。
2. STRPFRTRC コマンドを使用してパフォーマンス・トレースを開始し、次いでシステム報告書と構成要素報告書を使用して、以下の考えられる問題を突き止めて訂正する。
   - マシン・プールの 1 秒あたりのページ不在率が 10 より高い場合は、マシン・プールに割り当てるメモリーを増やして、不在率がこのレベルより低くなるようにする。
   - ディスク使用率が 40% を超えている場合は、待ち時間およびサービス時間を調べる。これらの値が許容範囲内の場合は、優先順位を管理するためのワークロードを削減しなければならない場合があります。
   - IOP 使用率が 60% を超えている場合は、 IOP を追加し、いくらかのディスク資源を割り当てる。

- ユーザー・プールのページ不在率が許容できないほど高い場合は、トピック『パフォーマンスの自動調整』を参照してください。

3. ジョブ要約報告書を実行して、**占有ロック競合報告書**を参照する。占有またはロック競合数が高い場合は、アクセス・パス・サイズを 1TB に設定してください。占有またはロック競合がユーザー・プロファイルで起こっている場合で、参照されたユーザー・プロファイルが多くのオブジェクトを所有している場合は、そのプロファイルが所有するオブジェクトの数を減らしてください。
4. **「タスク切り替え (Task switch)」** オプションを指定して iDoctor を 5 分間実行する。その後、タスク切り替えモニターを使用して結果トレース・データを分析します。以下を確認して解決します。
   - CPU 待ちのジョブ数
   - 不在ジョブ数
   - 占有競合数

パフォーマンス・データの表示

パフォーマンス・データを表示すると、システムのパフォーマンスをより正確に分析するのに役立ちます。パフォーマンス・データはさまざまな方法で表示できますが、状況によって特定のパフォーマンス・アプリケーションが適している場合があります。ほとんどのアプリケーションは、 収集サービスを使用して収集されたデータか、パフォーマンス・トレースから収集されたデータを表示します。データにアクセスする最善の方法は、パフォーマンス上の問題を解決しようとしているのか、今後の成長を考慮してシステム・パフォーマンスをモニターするのか、傾向を見分けるのかによって異なります。

近況リアルタイム・パフォーマンス・データを表示する

現在または最近のパフォーマンス情報を表示するには、以下のツールを使用します。

OS/400 コマンド
基本オペレーティング・システムには、システム・パフォーマンスの特定の領域についての現行情報を表示できる多くのコマンドがあります。

Performance Tools の画面
Performance Tools ライセンス・プログラムには、収集サービスの収集オブジェクトのパフォーマンス・データを表示する、 iSeries ナビゲーター用のプラグインが組み込まれています。システム上のジョブについての詳細情報を表示したり、 Performance Tools の報告書を印刷することもできます。

システム・モニターおよびジョブ・モニター
これらのモニターは、多くのシステム要素のパフォーマンス・データを表示します。モニター・データは、収集オブジェクトに基づいており、収集サービスの収集間隔に従ってデータが収集されたときに表示されます。

ヒストリー・パフォーマンス・データを表示する

システム上に保管されているデータを表示するには、以下のツールを使用します。

PM/400
PM/400 はシステム・パフォーマンス・データの収集、保存、および分析を自動化し、システム資源およびキャパシティーを管理するのに役立つ明確な報告書を戻します。

グラフ・ヒストリー
グラフ・ヒストリーは、収集サービスの保存期間に基づいて、最大で 1 週間相当のパフォーマンス・データをグラフィカルに表示します。 PM/400 を使用すると、グラフ・ヒストリーはより長い期間のデータ収集を表示できます。

パフォーマンスのチューニング

パフォーマンス調整の主な目的は、サーバーがシステム資源を最大限に活用できるようにし、ワークロードを可能な限り効果的に管理することにあります。パフォーマンス調整は、システムのパフォーマンスを調整する 1 つの方法であり、手動で、あるいは自動的に行うことができます。システムを調整するためのオプションはたくさんあります。それぞれのシステム環境はどれも固有なものであるため、そのパフォーマンスを観察して、その環境にとって最も良い調整を施すことが必要です。言い換えれば、パフォーマンス・モニターを定期的に実行する必要があります。パフォーマンスを調整する前に実行すべきパフォーマンス・モニターのステップの詳細については、『iSeries パフォーマンスの管理』を参照してください。

IBM では、ディスクから読み取られる物理入出力要求の数を減らすことによって入出力サブシステムとシステム応答時間の両方を向上させるツールも提供しています。 拡張キャッシュを使用してシステム・パフォーマンスを向上させる方法を学んでください。

パフォーマンス調整の詳細については、以下のトピックを参照してください。

パフォーマンス調整の基本
システムのパフォーマンスを調整するには、初期調整値をセットアップし、システム・パフォーマンスを監視し、値を検討し、調整対象を判別する必要があります。

パフォーマンスの自動調整
ほとんどのユーザーの場合は、自動的にパフォーマンス調整を行うようにシステムをセットアップできます。出荷される時点の新しいシステムには、あらかじめ、自動的に調整を行うように構成が行われています。

パフォーマンス調整の基本

パフォーマンスの調整を開始するには、まず、初期マシン・プール・サイズと初期ユーザー・プール・サイズを決定することによって、初期調整値を設定する必要があります。その後、システム・パフォーマンスの監視を開始できます。

初期調整値を設定する
初期調整値の設定には、効果的にシステムを調整するために、最初にシステム・プール・サイズと活動レベルを構成するステップが含まれます。初期値は推定値に基づいています。したがって、システムが活動状態にある間に、推定値をさらに調整しなければならない場合があります。初期調整値は、以下のステップで設定します。

- 初期マシン・プール・サイズを決定する
- 初期ユーザー・プール・サイズを決定する

システム・パフォーマンスを監視する
システム・パフォーマンスを監視するために、システム状況の処理 (WRKSYSSTS)、ディスク状況の処理 (WRKDSKSTS)、および活動ジョブの処理 (WRKACTJOB) コマンドを使用できます。各監視期間について、パフォーマンスの目標値に照らしてシステム・パフォーマンスの測定値を考察および評価する必要があります。

1. 不規則なシステム活動を除去する。重度のパフォーマンス低下の原因となりうる不規則な活動には、たとえば、対話式プログラム・コンパイル、通信エラー・リカバリー手順 (ERP)、 QUERY ファイルのオープン (OPNQRYF)、サインオフ活動などがあります。
2. WRKSYSSTS、WRKDSKSTS、および WRKACTJOB コマンドを使用して、パフォーマンス・データを表示する。パフォーマンス・データは、 Performance Tools コマンド (システム活動の処理 (WRKSYSACT)) を使用して表示することもできます。
3. システムのデータ収集を最低 5 分行う。
4. パフォーマンスの目標値に照らしてパフォーマンスの測定値を評価する。通常の測定値は以下のとおりです。
   - 対話式スループットおよび応答時間。これは WRKACTJOB 画面に表示されます。
   - バッチ・スループット。活動状態のバッチ・ジョブの補助入出力 (AuxIO) 値と CPU 稼働率 (CPU%) 値を監視します。
   - スプール・スループット。活動状態の書き出しプログラムの補助入出力 (AuxIO) 値と CPU 稼働率 (CPU%) 値を監視します。
5. 期待に沿わないパフォーマンス・データがある場合は、システムを新しいデータに基づいて調整する。以下の点に気を付けてください。
   - すべての重要なパフォーマンス測定値を測定および比較する。
   - 調整の実施と評価を同時に行う。

パフォーマンスを検討する

適切な調整値を設定したなら、システムが順調に機能しつづけるようにするために、定期的に調整値を検討する必要があります。継続調整には、システム・パフォーマンスの各面の監視と、推奨されるガイドラインへの調整が含まれます。

意味のある統計を収集するには、標準的な活動レベルの時にシステム・パフォーマンスを監視する必要があります。たとえば、システムでジョブが実行されていない間に収集される統計は、システム・パフォーマンスを評価する点ではほとんど価値がありません。最大限の努力を払ってもパフォーマンスが納得のいくものではない場合は、現在の構成の能力を評価する必要があります。目標を達成するために、以下を考慮してください。

- プロセッサーのアップグレード
- 記憶装置およびコントローラーの追加
- 主記憶装置の追加
- アプリケーションの修正

上記の 1 つ以上の方法を適用することによって、目標を達成できます。適切な方法を適用した後に、まだ目標を達成できない場合は、実行する作業の種類にとって現実的な目標を設定しているかどうかを判断する必要があります。

調整対象を判別する

システム・パフォーマンスが低下して調整が必要な場合は、パフォーマンス上の問題の原因を突き止めて具体的な訂正を行うために、『パフォーマンスの問題の調査』を参照してください。

パフォーマンスの自動調整

システム資源を効果的に使用するために、システムはパフォーマンス値を自動的に設定できます。以下を行うことによって、システム・パフォーマンスを自動的に調整するようにシステムをセットアップできます。

- 記憶域プール・サイズと活動レベルの調整

- 記憶域プール・ページングの調整

記憶域プール・サイズと活動レベルの調整

QPFRADJ システム値を使用して、記憶域プールと活動レベルの自動調整を制御します。この値は、システムがシステム再始動 (IPL) 時に値を調整するのか、再始動後に定期的に調整するのかを指示します。

パフォーマンスを IPL 時に調整する、動的に調整する、または IPL 時にも動的にも調整するようにシステムをセットアップできます。

- システム再始動 (IPL) 時にのみ調整を行うようシステムをセットアップするには、 iSeries ナビゲーターで「構成およびサービス」->「システム値」->「パフォーマンス」を選択する。「メモリー・プール」タブをクリックして、**「メモリー・プールと活動レベルを自動的に調整する (Automatically adjust memory pools and activity levels)」**の下で**「システム再始動時 (At system restart)」**を選択します。これは、QPFRADJ システム値を 1 に設定することと同じです。
- 記憶域プールの調整をシステム再始動 (IPL) 時に行い、再始動後に記憶域プールの調整を定期的に行うようシステムをセットアップするには、 iSeries ナビゲーターで「構成およびサービス」->「システム値」->「パフォーマンス」を選択する。「メモリー・プール」タブをクリックして、**「メモリー・プールと活動レベルを自動的に調整する (Automatically adjust memory pools and activity levels)」**の下で**「システム再始動時 (At system restart)」**と**「再始動後定期的に (Periodically after restart)」**を選択します。これは、QPFRADJ システム値を 2 に設定することと同じです。
- 記憶域プールの調整をシステム再始動 (IPL) 時に行わず、再始動後に定期的に行うようシステムをセットアップするには、 iSeries ナビゲーターで「構成およびサービス」->「システム値」->「パフォーマンス」を選択する。「メモリー・プール」タブをクリックして、**「メモリー・プールと活動レベルを自動的に調整する (Automatically adjust memory pools and activity levels)」**の下で**「再始動後定期的に (Periodically after restart)」**を選択します。これは、QPFRADJ システム値を 3 に設定することと同じです。

記憶域プール値は、システム再始動 (IPL) 時に初期値にリセットされることはありません。

記憶域プール・ページングの調整

システムによって提供される動的調整は、システムのパフォーマンスを改善するために、共用プールのプール・サイズと活動レベルを自動的に調整します。この調整は、使用率が最も低い記憶域プールから、より多くの記憶域があることで利点があるプールに記憶域を移動することによって行われます。この調整では、プール内のスレッドの数のバランスを取るために、プールに割り振られた記憶域を使用して、活動レベルの設定も行われます。システムを調整するために、調整プログラムは、スレッドの数に基づいて計算されるガイドラインを使用します。

動的調整が有効になっている場合は、以下のパフォーマンス値が適切な設定値に自動的に変更されます。

- マシン (*MACHINE) メモリー・プール・サイズ (QMCHPOOL システム値)
- ベース (*BASE) メモリー・プール・レベル (QBASACTLVL システム値)
- 共用プール *INTERACT のプール・サイズと活動レベル
- 共用プール *SHRPOOL のプール・サイズと活動レベル
- 共用プール *SHRPOOL1-*SHRPOOL60 のプール・サイズと活動レベル

動的調整が有効になっている (QPFRADJ システム値が 2 または 3 に設定されている) 場合は、プロファイル QSYS の下で実行されるジョブ QPFRADJ はシステム上で活動状態として表示されます。

メモリー・プールの詳細については、『メモリー・プール』を参照してください。

e-business パフォーマンスの管理

e-business 環境におけるパフォーマンスには、 iSeries システム管理者にとって複雑な問題がいくつか含まれます。 iSeries サーバーの日常的な調整に加え、管理者は e-business トランザクションをサポートしているハードウェアとサービスもモニターして最適化する必要があります。

以下のトピックは、サーバーの e-business パフォーマンスを最大化するためのいくつかの重要な考慮事項を理解するのに役立つもので、詳細な推奨事項と例がある追加情報資源へのリンクが含まれています。

クライアント・パフォーマンス
多くの場合、 iSeries システム管理者が e-business ネットワークのクライアント・サイドの制御を行うことはほとんどありませんが、ここでの推奨事項を基に、クライアント装置が e-business 環境に合わせて最適化されていることを確認することができます。

ネットワーク・パフォーマンス
多くの場合、ネットワーク設計、ハードウェア資源、およびトラフィック・プレッシャーは、e-business アプリケーションに大きな影響を与えます。ネットワーク・パフォーマンスの最適化の方法と iSeries 通信資源の調整の方法については、このトピックを参照してください。

OS/400 における Java パフォーマンス
OS/400 には、iSeries サーバー上の Java のアプリケーションやサービスのパフォーマンスを最適化するための構成オプションと資源がいくつかあります。 OS/400 における Java 環境について、および Java ベースのアプリケーションで可能な限り最高のパフォーマンスを得る方法については、このトピックを参照してください。

HTTP サーバー・パフォーマンス
HTTP サーバーは多くの場合、iSeries サーバーでの e-business パフォーマンスの重要な部分です。IBM は、このサーバーを最大限に生かすことができるオプションと構成の選択項目をいくつか用意しています。

WebSphere パフォーマンス
WebSphere Application Server は、 iSeries サーバーにとって最適な e-business アプリケーション配置環境です。 WebSphere 環境でのパフォーマンスの計画と最適化の方法については、このトピックを参照してください。

管理者は、これらの特定の推奨トピックのほかに、次のトピックも理解する必要があります。

- 実行管理機能
- Java for iSeries
- HTTP サーバー
- Domino for iSeries sizing and performance tuning 

クライアント・パフォーマンス

Web ブラウザーを搭載している PC から成るクライアントは多くの場合、管理者が直接制御を行うことが最も少ない e-business 構成要素になります。しかし、この構成要素は Web アプリケーションのエンドツーエンドの応答時間にも大きく影響します。

高パフォーマンスが得られるようにするには、クライアント PC を次のようにする必要があります。

- 十分なメモリー容量にします。資源集中アプレット、および複雑なフォームとグラフィックスを使用するインターフェースによって、クライアントのプロセッサーに対しても要求が行われる場合があります。
- 高速で最適化されたネットワーク接続を使用します。クライアント PC 上の通信アダプターの多くが、それらのネットワーク環境に合わせて最適化されないまま機能していることがあります。詳しくは、ご使用の通信ハードウェアの資料を参照してください。
- 必要なテクノロジーを完全にサポートしているブラウザーを使用します。また、ブラウザーのサポートとパフォーマンスは、 Web インターフェースを設計するときの主要な関心事にすべきことです。

ネットワーク・パフォーマンス

多くの場合、ネットワークは Web アプリケーションの応答時間において主要な役割を果たします。その上、ネットワーク構成要素に対するパフォーマンスの影響はたいていは複雑で測定が困難です。ネットワーク・トラフィックと使用可能帯域幅はたびたび変化することがあり、システム管理者が直接制御できない作用の影響を受けるためです。しかし、ご使用の iSeries サーバーの通信資源のモニターと調整に使用できる資源がいくつかあります。

詳しくは、以下のトピックを参照してください。

収集サービス

収集サービスは通信資源のパフォーマンス・データを定期的に収集します。特に関心がある TCP サーバーに関する情報は、パフォーマンス・データ・ファイルの QAPMTCP と QAPMTCPIFC に保管されます。このデータはそれらのファイルを直接照会するか、あるいは Performance Tools ライセンス・プログラムに含まれる報告書を使用して参照することができます。

システム・モニター

iSeries サーバーの通信ハードウェアを含むシステム資源がどのように使用されているかという情報を、システム・モニターを使用して得ることができます。システム・モニターでの回線稼働率および IOP メトリックは、ネットワーク・パフォーマンスに関する特に有用なデータになります。

パフォーマンスの追跡

いくつかのアプリケーションとツールを使用すると、 iSeries サーバーの通信資源のパフォーマンス・データを定期的に収集して、そのパフォーマンスを長期間に渡ってモニターすることができます。

iSeries Performance Capabilities Reference 

Performance Capabilities Reference には、ご使用の iSeries サーバーを最適なパフォーマンスに構成したり調整するときに役立つ詳細な情報、報告書、および例が記載されています。特に iSeries 通信資源の計画と管理について、第 5 章『Communications Performance』を参照してください。

iSeries Network.com 

この Web サイトには、ネットワーク計画と資源を最適化するための多数の情報資源があります。特に、『Cultivate your AS/400 Networks』と『8 tools for better network performance』を参照してください。

OS/400 における Java パフォーマンス

Java は多くの場合、Web ベース・アプリケーションに最適な言語です。しかし Java アプリケーションで最適なパフォーマンスを得るには、 OS/400 実行環境と Java アプリケーションの両方で幾らかの最適化が必要な場合があります。

OS/400 における Java 環境と、 Java パフォーマンスの分析と改善のための有効なヒントとツールについては、以下の情報資源を利用してください。

Java パフォーマンス
Java ベースのアプリケーションで最高のパフォーマンスを得るのに役立つ、いくつかの重要な構成の選択とツールがあります。

アプリケーションのパフォーマンスについての情報の収集
OS/400 におけるアプリケーションのパフォーマンスのモニターと調整に使用できるツールがいくつかあります。アプリケーション・パフォーマンスの測定と改善に役立つパフォーマンスのトレース、Performance Explorer (PEX)、および類似ツールの使用方法については、このトピックをご覧ください。

iSeries Performance Capabilities Reference
Performance Capabilities Reference には、ご使用の iSeries サーバーを最適なパフォーマンスに構成したり調整するときに役立つ詳細な情報、報告書、および例が記載されています。特に Java アプリケーションのパフォーマンスの最適化と Java プログラミングでのパフォーマンスのヒントについて、第 7 章『Java Performance』を参照してください。

Java and WebSphere performance in OS/400
Java および WebSphere のパフォーマンスを最高にするための、およびパフォーマンス・データの収集と分析を行うための稼働環境の計画および構成の方法については、この Redbook をお読みください。

WebSphere J2EE application development for the IBM eServer iSeries server
この Redbook には、 J2EE の概要と、 iSeries サーバーに J2EE アプリケーションを正しくインプリメントするのに役立つ提案と例が記載されています。

パフォーマンス情報のほかに、 Java トピックには iSeries サーバーに Java アプリケーションを開発して配置する際の情報資源が記載されています。

HTTP サーバー・パフォーマンス
HTTP サーバーは Web ベース・アプリケーションのエンドツーエンドのパフォーマンスで重要な役割を果たします。また新規改良点によって Web サーバーのパフォーマンスの効果的なモニターと改善が可能になりました。新規の Fast Response Caching Accelerator (FRCA) を使用することによって、特に大部分が静的な環境では、 HTTP サーバーのパフォーマンスを大幅に向上させることができる場合があります。

HTTP サーバーのパフォーマンスを最大限に高める方法については、以下の情報資源を参照してください。

収集サービス
収集サービスを使用して HTTP サーバーのパフォーマンス・データを収集し、その結果を長期間に渡ってモニターすることができます。 HTTP サーバーのデータは、収集間隔ごとにパフォーマンス・データ・ファイル QAPMHTTPB および QAPMHTTPD に保管されます。 QAPMHHTB には基本情報があり、 QAPMHTTPD にはより詳細な統計情報があります。これらのデータ・ファイルは直接照会することができますが、 Performance Tools ライセンス・プログラムのシステム報告書と構成要素報告書を参照することもできます。

IBM HTTP Server for iSeries
iSeries の HTTP サーバーのセットアップ、構成、および管理については、このトピックを参照してください。このトピックには、 Fast Response Caching Accelerator (FRCA) のような本製品への最新の機能強化の説明も含まれています。

iSeries Performance Capabilities Reference
Performance Capabilities Reference には、ご使用の iSeries サーバーを最適なパフォーマンスに構成したり調整するときに役立つ詳細な情報、報告書、および例が記載されています。特に、HTTP サーバーのパフォーマンスの詳細な説明、計画に関する情報、およびパフォーマンスのヒントについて、第 6 章『Web Server and Web Commerce』を参照してください。

HTTP server (Powered by Apache)
OS/400 での HTTP Server (Powered by Apache) の詳細な解説は、この Redbook をお読みください。一般的な使用シナリオで HTTP Server を構成する例が含まれています。

AS/400 HTTP Server Performance and Capacity Planning
パフォーマンスの調整と計画における HTTP サーバーの影響については、この Redbook を参照してください。この資料には、 iSeries パフォーマンス管理ツールを使用して Web サーバーのパフォーマンス・データを収集、解釈、およびそれに対応する場合の提案も記載されています。

WebSphere パフォーマンス

WebSphere 環境の iSeries サーバーのパフォーマンスの管理には、 iSeries 管理者にとってのいくつかの課題が伴います。 Web ベースのトランザクションによって消費される資源は多くなることがあり、しかもそこで消費される資源はいつもの通信ワークロードとは異なる場合があります。

WebSphere 環境での最適パフォーマンスの計画方法およびサーバー資源の調整方法については、以下のトピックと情報資源を参照してください。

WebSphere Application Server performance considerations
この Web サイトには、 iSeries サーバー上の WebSphere Application Server の各バージョンの、パフォーマンスに関する多数の有用なヒントと推奨事項を含む情報資源があります。この情報資源は、サーブレット、Java Server Pages (JSP)、および Enterprise Java Beans (EJB) を使用する環境では特に有用です。

DB2 UDB/WebSphere Performance Tuning Guide
この Redbook には WebSphere 環境と DB2 環境のそれぞれの概要が記載されており、パフォーマンス上の共通の問題に対する提案、例、およびソリューションが示されています。これらを基に WebSphere と DB2 のパフォーマンスを最適化することができます。

Java and WebSphere performance in OS/400
Java および WebSphere のパフォーマンスを最高にするための、およびパフォーマンス・データの収集と分析を行うための稼働環境の計画および構成の方法については、この Redbook をお読みください。

WebSphere V3 Performance Tuning Guide
この Redbook には、 iSeries サーバー上の WebSphere V3 のパフォーマンスを最適化するための推奨事項と例が詳細に示されています。

iSeries Performance Capabilities Reference
Performance Capabilities Reference には、ご使用の iSeries サーバーを最適なパフォーマンスに構成したり調整するときに役立つ詳細な情報、報告書、および例が記載されています。特に WebSphere 特有のパフォーマンス上のヒントについて、第 6 章『Web Server and Web Commerce』を参照してください。

WebSphere および e-business に関するその他の情報資源については、トピック『WebSphere e-business の管理』を参照してください。

パフォーマンス管理用のアプリケーション

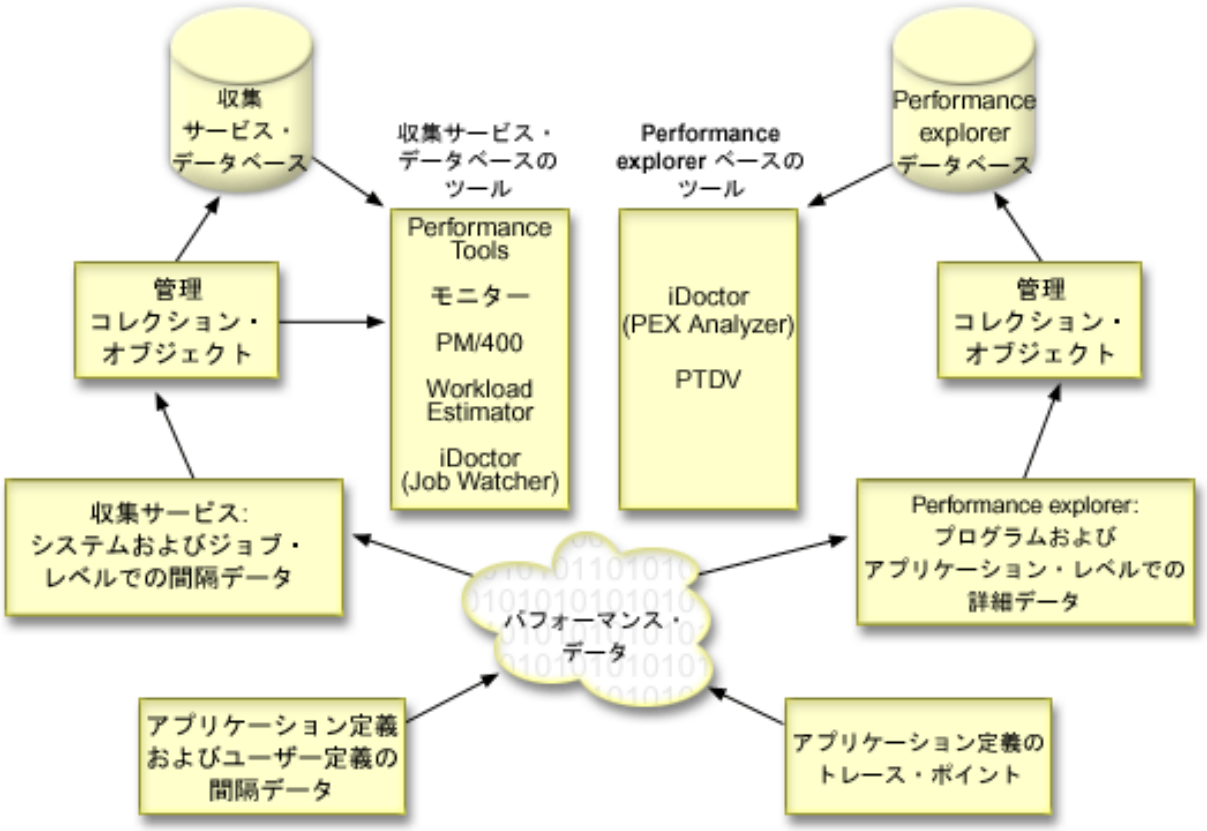
パフォーマンス管理用の多くのアプリケーションには、複数の機能があります。使用可能な一組のアプリケーションのどの構成要素が、特定の状況に最適であるかを正確に把握するのは困難です。以下のトピックでは、各パフォーマンス管理アプリケーションの選択、使用、および構成を含めた詳細が説明されています。

以下の図に示されているとおり、iSeries サーバーには、基本的に 2 つのパフォーマンス収集機能があります。

- 収集サービス。システム・レベルおよびジョブ・レベルで間隔データを収集します。システムの状況を把握するために、このサービスを継続的に実行できます。収集される内部データは、アプリケーション定義データまたはユーザー定義データです。
- Performance Explorer。プログラム・レベルおよびアプリケーション・レベルで詳細データを収集します。アプリケーションでの作業の流れもトレースするので、パフォーマンス上の難しい問題を診断するために使用することもできます。データは、ドミノ、NetServer、または WebSphere などの、アプリケーションで定義された Performance Explorer トレース・ポイントに基づいて収集されます。

これらの収集機能は両方とも、データを管理収集オブジェクトに保管します。収集サービス・データの場合はパフォーマンス・データの作成 (CRTPFRDTA) コマンド、 Performance Explorer データの場合は Performance Explorer データの作成 (CRTPEXDTA) コマンドを使用して、管理収集オブジェクトのデータを変換できます。

このトピックでは、収集サービス・データまたは Performance Explorer データに使用できるパフォーマンス管理アプリケーションを紹介します。



収集サービス
収集サービスは、ユーザー定義の時間間隔でパフォーマンス・データを収集し、この情報をシステム上の収集オブジェクトに保管します。モニター、グラフ・ヒストリー、PM/400 などの他の多くのツールや、 Performance Tools ライセンス・プログラムの多くの機能は、これらの収集オブジェクトからデータを入手します。

パフォーマンス・データベース・ファイル
収集サービスが管理する収集オブジェクトからデータベース・ファイルを生成できます。これらのデータベース・ファイルの名前、説明、および属性については、このトピックを参照してください。

モニター
モニターは、システムのパフォーマンスについての現行情報を表示します。特定のイベントの発生時に事前定義アクションを実行するために、モニターを使用することもできます。システム、メッセージ、ジョブ、ファイル、および B2B トランザクション・モニターを使用して、システムについての情報を表示したりモニターすることができます。システム・モニターとジョブ・モニターは、収集サービスによって収集されたパフォーマンス・データを使用します。

グラフ・ヒストリー
グラフ・ヒストリーは、指定した期間に渡って収集サービスによって収集されたパフォーマンス・データをグラフィカルに表示します。

PM/400
PM/400 はシステム・パフォーマンス・データの収集、保存、および分析を自動化し、システム資源およびキャパシティーを管理するのに役立つ報告書を戻します。 PM/400 は、収集サービスによって収集されたデータを使用します。

Performance Tools
Performance Tools ライセンス・プログラムには、システム・パフォーマンス情報を収集、分析、および保守するのに役立つ多くの機能があります。これには、分散ネットワーク上でのパフォーマンスの管理、要約データおよびトレース・データの収集と報告、およびキャパシティーの計画に役立つ機能が含まれています。 Performance Tools は、収集サービスによって収集されるパフォーマンス・データと、パフォーマンス・トレースの開始 (STRPFRTRC) コマンドおよびパフォーマンス・トレースの終了 (ENDPFRTRC) コマンドから取得されるトレース・データを使用します。

Performance Explorer
Performance Explorer は、特定のアプリケーション、プログラムまたはシステム資源に関するより詳細な情報を収集し、特定のパフォーマンス上の問題を詳しく洞察します。これには、複数のタイプおよびレベルのトレースを実行する機能と、明細報告書を実行する機能が含まれます。

iDoctor for iSeries
iDoctor for iSeries プラグインは、パフォーマンスを管理するための 3 つのソフトウェア・ツールで構成されます。つまり、Performance Explorer Analyzer (トレース・データの詳細分析用)、 Job Watcher (ジョブの動作についてのトレース・レベルの情報用)、および Object Explorer (システム上のオブジェクトの照会および管理用) です。

Performance Trace Data Visualizer (PTDV)
Performance Trace Data Visualizer for iSeries (PTDV) は、 iSeries 上で実行されるアプリケーションのパフォーマンス分析に使用できる Java アプリケーションです。

Performance Management API
Performance Management API は、収集を管理するためのサービスを提供します。これらの API は、収集を開始、終了、および循環させ、収集されたデータのシステム・パラメーターを変更します。多くの Performance Management API は、収集サービスによって収集されたパフォーマンス・データを使用します。

OS/400 パフォーマンス・コマンド
OS/400 には、システム・パフォーマンスの管理および保守に役立つ複数の重要な機能があります。

拡張最適キャッシュ
拡張最適キャッシュは、ディスク使用量データを収集し、それらの統計を使用して大規模なキャッシュを作成し、ディスクの物理入出力要求を効果的に削減することによって、システム・パフォーマンスを改善できます。

Workload Estimator for iSeries
Workload Estimator は、次のアップグレードのサイズおよびタイミング要件を計画するのに役立ちます。このツールは、システム・パフォーマンスの傾向を分析するために PM/400 と一緒に使用されることがよくあり、iSeries サーバーの成長と拡張を効果的に管理するのに役立ちます。

iSeries ナビゲーター (ワイヤレス対応)
iSeries ナビゲーター (ワイヤレス対応) を使用すると、携帯情報端末 (PDA)、インターネット電話、

または旧来の Web ブラウザーを使用した無線接続を介して、パフォーマンス・データをモニターすることができます。 iSeries ナビゲーター (ワイヤレス対応) は、収集サービスによって収集されたデータを使用します。

PATROL for iSeries (AS/400) - Predict
PATROL for iSeries (AS/400) - Predict は、高可用性および最適なパフォーマンスに必要な多くの定期管理作業を自動化することにより、 iSeries パフォーマンスの管理に役立ちます。さらに、この製品は、iSeries 環境の成長を計画するのに役立つ、詳細なキャパシティー・プランニング情報を提供します。

収集サービス

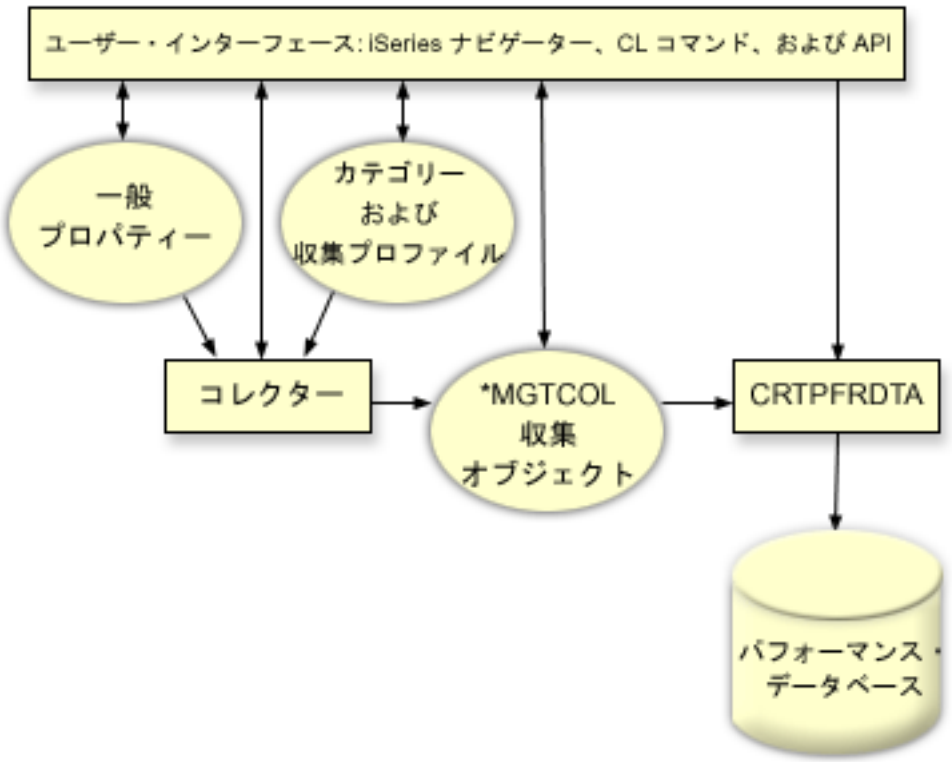
収集サービスを使用して、 Performance Tools for iSeries ライセンス・プログラムまたはその他のパフォーマンス報告書アプリケーション、iSeries ナビゲーター・モニター、およびグラフ・ヒストリー機能による今後の分析のために、パフォーマンス・データを収集します。 (リアルタイムにパフォーマンス・データを表示させたい場合、システム・モニターには、システム・パフォーマンスをモニターするための簡単に使えるグラフィカル・インターフェースもあります。) 収集サービスは、システムのさまざまな分野ごとに使用されるシステム・リソースの相対的な量を示すデータを収集します。収集サービスを使用して、以下のことを行うことができます。

- 収集オブジェクトを容易に管理する。
- 最小のシステム・オーバーヘッドを使用してパフォーマンス・データを継続および自動的に収集する。
- どのようなデータを収集するか、またデータをどのように使用するかを制御する。
- データを変換することなく、リリース間でパフォーマンス・データを移動する。
- Performance Tools によって使用されるパフォーマンス・データ・ファイルを作成する。
- ユーザー定義パフォーマンス・データを収集するためのプログラムを収集サービスに統合する。

収集サービスの動作方法

収集サービスは、OS/400 パフォーマンス・モニターの代わりとなります。これは、パフォーマンス・モニターの開始 (STRPFRMON) コマンドによって呼び出されました。パフォーマンス・モニター (STRPFRMON コマンド) は、V4R5 以降は使用可能ではありません。 OS/400 パフォーマンス・モニターを使用したとき、収集されたデータは 30 個ものデータベース・ファイルになります。

収集サービス機能では、パフォーマンス・データを収集するための新しいプロセスを紹介します。収集サービスであれば、データは収集ごとに単一の収集オブジェクトに入れられ、そこからデータベース・ファイルのさまざまな集合を必要に応じて作成することができます。つまり、パフォーマンス・データを収集する間も、システムのオーバーヘッドを低く押さえることができます。 また、収集の実行時にデータベース・ファイルを作成する場合でも、収集サービスは、低い優先順位 (50) のバッチ・ジョブを使用してこれらのファイルを更新するため、 OS/400 パフォーマンス・モニターよりも高いパフォーマンスを維持することができます。 このようにして、収集によるオーバーヘッドを減らすことにより、パフォーマンス・データをより詳細に、また、より短い間隔で継続的に収集することが可能になります。収集サービスを使用すれば、パフォーマンス・データの収集と保存に関してネットワーク規模のシステム・ポリシーを設け、そのポリシーを自動的に実装することができます。 これらの管理収集オブジェクトが保管されている限り、必要が生じた場合でも、収集したデータから過去のパフォーマンスに関連したイベントを振り返り、詳細なレベルまでそれを分析することができるのです。



収集サービスを使用すると、システム・パフォーマンスにほとんど影響を与えることなく、またははっきり分かるほどの影響を与えることなく、パフォーマンス・データを収集できます。 iSeries ナビゲーターを使用して、任意の頻度でデータを収集するよう収集サービスを構成することができます。収集オブジェクト *MGTCOL は、大量のパフォーマンス・データを保持するための効果的なストレージ・メディアとしての機能を果たします。収集サービスを構成して開始すると、パフォーマンス・データは継続的に収集されます。パフォーマンス・データを処理する必要がある場合は、必要なデータを一連のパフォーマンス・データベース・ファイルにコピーできます。

上の図は、以下の収集サービス要素の概要を示しています。

ユーザー・インターフェース
収集サービスの異なる要素にアクセスできる複数の方法が用意されています。たとえば、CL コマンド、API、および iSeries ナビゲーター・インターフェースを使用できます。

一般プロパティー
一般プロパティーは、収集がどのように実行されるかを定義し、自動収集属性を制御します。

データ・カテゴリー
データ・カテゴリーは、収集するデータのタイプを識別します。各カテゴリーを別々に構成し、収集するデータと、データ収集の頻度を制御することができます。

収集プロファイル
収集プロファイルは、特定のカテゴリー構成を保管および活動化するための手段を提供します。

パフォーマンス・コレクター
パフォーマンス・コレクターは、一般プロパティーとカテゴリー情報を使用して、パフォーマンス・データの収集を制御します。パフォーマンス・コレクターは手動で開始および停止できますし、自動的に実行されるよう構成することもできます。

収集オブジェクト
収集オブジェクト *MGTCOL は、大量のパフォーマンス・データを保持するための効果的なストレージ・メディアとしての機能を果たします。

パフォーマンス・データの作成 (CRTPRFDTA) コマンド
CRTPFRDTA コマンドは、管理収集オブジェクトに保管されているデータを処理し、パフォーマンス・データベース・ファイルを生成します。

パフォーマンス・データベース
データベース・ファイルには、 CRTPFRDTA コマンドによって処理されるデータが保管されます。このファイルは、時間間隔データが入っているパフォーマンス・データ・ファイル、構成データ・ファイル、トレース・データ・ファイルというカテゴリーに分けることができます。

収集サービスの開始方法

以下の方法のいずれかを使用して、収集サービスを開始することができます。ただし、パフォーマンスのトピック内にある情報は、 iSeries ナビゲーターの方法に焦点を当てています。

開始方法	説明
iSeries ナビゲーター	続くセクションでは、 iSeries ナビゲーターを使用して、さまざまな収集サービス・タスクを行う方法を示します。
Performance Management API	Performance Management API を使用して、収集の開始、カスタマイズ、終了、循環を行うことができます。さらに、API を使用して、管理収集オブジェクトを処理したり独自のトランザクションを定義することができます。
従来のメニュー・オプション	文字ベースのインターフェースで **GO PERFORM** と入力して、 Performance Tools のメイン・メニューでオプション 2 (パフォーマンス・データの収集) を選択します。追加情報については、 Performance Tools for iSeries  に進んでください。
Performance Management/400	収集サービスの開始を自動化し、収集中にデータベース・ファイルを作成する PM/400 を活動化することができます。

収集サービスのタスク

収集サービスおよび iSeries ナビゲーターを使用して、下の表に示されているように、さまざまなデータ収集タスクを実行することができます。

タスク	説明
さまざまな方法での収集サービスの開始	特定のパフォーマンス・メトリックを使用する個々のシステムまたはシステム・グループ上に、カスタマイズされたパフォーマンス・データ・コレクションを作成します。パフォーマンス・データ・コレクションを自動的に開始するために、始動プログラム内の Start Collector API を使用することもできます。これらのタスクを実行する方法の詳細については、オンライン・ヘルプを参照してください。作業の詳細ヘルプは、「iSeries ナビゲーター」ウィンドウから使用できます。メニュー・バーから**「ヘルプ」**をクリックして、**「ヘルプ・トピック」**を選択します。**「. . . によって実行できる処理」**を選択して、実行できる処理と、「iSeries ナビゲーター」ウィンドウのどこから実行するのかを調べます。
データベース・ファイルの作成	収集サービスを使用して、パフォーマンス・データベース・ファイルの作成を自動化します。データベース・ファイルは、収集後にデータが保管されるコレクション・オブジェクトから作成することもできます。 PM/400、Performance Tools ライセンス・プログラムとともにこれらのデータベース・ファイルを使用するか、またはこれらのファイルに対して実行するための独自の照会を作成することができます。 データベース・ファイルを作成するときに、どのデータを収集するかを制御する方法を学習します。 『パフォーマンス・データベース・ファイル 』を参照し、それぞれのファイルに含まれているフィールド・レベルのデータだけでなく、どのデータベース・ファイルが使用可能であるかについても調べます。
データ収集のカスタマイズ	データ収集をカスタマイズします。収集するパフォーマンス・データおよびデータが収集される頻度の制御に関する情報を調べます。重要な時間帯の考慮事項についての情報も検索できます。
ユーザー定義カテゴリーの収集サービスへの追加	出口プログラムを作成し、それを収集サービスに統合することによって、ユーザー・アプリケーションからパフォーマンス・データを収集できます。定期収集間隔中にこのデータを収集し、収集オブジェクトに保管することができます。これを実行する方法については、『ユーザー定義カテゴリー』を参照してください。
収集オブジェクトの管理	収集オブジェクトの内容、収集オブジェクトの保管期間、および収集オブジェクトを使用して実行できる事柄を含め、 収集オブジェクトの管理の必要がある情報を検索します。
トレース・データの収集	収集サービスでは、サンプル・データを収集します。しかし、トレース・データは収集しません。 トレース・データの収集方法を調べます。
ユーザー定義トランザクション用のパフォーマンス・データの収集	収集サービスには、独自のトランザクションを定義できる API が用意されています。この作業を行う方法については、『ユーザー定義トランザクション』を調べてください。

収集サービスのデータからのデータベース・ファイルの作成

収集サービスは、収集されたデータを管理収集オブジェクトに保管します。このデータを使用するには、まず、データをデータベース・ファイルの特別なセットの中に入れる必要があります。データ収集時に自動的にデータベース・ファイルが作成されるようにするには、ただ**「収集サービスの開始」**ダイアログから**「データベース・ファイルの作成」**を選択します。また、既存の管理収集オブジェクトからデータをエクスポートしたいときに、 後からデータベース・ファイルを作成することもできます。

データベース・ファイルは、いろいろなオプションで作成できます。

- 収集サービスを使用してパフォーマンス・データを収集する場合、データが収集されるにつれて自動的にデータベース・ファイルが作成されるようにすることができます。

- 収集された後のデータが保管されている管理収集オブジェクトからデータベース・ファイルを作成できます。 パフォーマンス・データの作成 (CRTPFRDTA) コマンドを使用すると、管理収集 (*MGTCOL) オブジェクトに保管されているパフォーマンス情報から一連のパフォーマンス・データベース・ファイルを作成できます。これには、iSeries ナビゲーターのインターフェースを使用することもできますし、CRTPFRDTA コマンドを使用しても構いません。
- PM/400 を活動化して、自動的に収集サービスを開始させ、収集の際にデータベース・ファイルを作成させることもできます。

作成したデータベース・ファイルは、 Performance Tools for iSeries ライセンス・プログラムやパフォーマンス報告生成用の他のアプリケーションで使用できます。その際には、まず 1 つのシステムでパフォーマンス・データの収集を行い、それからその管理収集 (*MGTCOL) オブジェクトを別のシステムに移して、パフォーマンス・データ・ファイルの生成と Performance Tools 報告書の実行を行います。このようにすることで、起動側のシステムのパフォーマンスには影響を与えることなく、別のシステムでパフォーマンス・データを分析できます。 Performance Tools についての詳細は、 Performance Tools for iSeries を参照してください。

データベース・ファイルではなく管理収集オブジェクトにデータを保管する理由

なぜ、報告書の実行に必要なデータベース・ファイルではなく管理収集オブジェクトにデータを保管する必要があるのでしょうか。それは、管理収集オブジェクトとデータベース・ファイルを別個に管理して、パフォーマンス・データの収集間隔は短く (例: 5 分間隔) しておき、データベース・ファイルはもっと長いサンプリング間隔 (例: 15 分間隔) で作成するということができるからです。

さまざまなデータ・カテゴリー、さまざまな時間範囲、さまざまなサンプリング間隔を指定することによって、 1 つの管理収集オブジェクトから、さまざまな目的のための異なるデータベース・ファイルのセットを作成できます。

たとえば、全カテゴリーのセット (すべてのデータ、または**「標準 + プロトコル」**プロファイル) について、 5 分間隔で 24 時間、パフォーマンス・データを収集するようにできますが、その 1 つの管理収集オブジェクトから、さまざまな目的のためのさまざまなデータベース・ファイルのセットを作成することができます。通常の 1 日ごとのパフォーマンス報告を実行するためのデータベース・ファイルのセットを 1 つ作成できます。それらのファイルには、サンプリング間隔を 15 分とした全カテゴリーのデータを含めることができます。特定のパフォーマンス上の問題を分析する場合には、データベース・ファイルの別のセットを作成できます。それらのファイルには、分析の対象となる単一のカテゴリーで、24 時間以内の特定の期間、そしてサンプリング間隔を 5 分に細分したデータだけを含めることができます。

さらに、単一の管理収集オブジェクトにより、データは多数のファイルとしてではなく、単一のオブジェクトとして管理することができます。この単一の収集オブジェクトによって、リリース間でパフォーマンス・データをデータ変換せずに移動することができます。収集オブジェクトを保持していれば、パフォーマンスに関連したイベントを振り返って、収集できた範囲での詳細な分析が行えます。

収集したデータのエクスポート

パフォーマンス・データを管理収集オブジェクトからデータベース・ファイルにエクスポートするには、次のようにします。

1. iSeries ナビゲーターで、**「マネージメント・セントラル」**の下にあるエンドポイント・システムか、**「現在の接続 (My Connections)」**(または活動状態の環境) の下にある、直接接続されているシステムを選択します。
2. **「構成およびサービス」**を展開します。

3. **「収集サービス」**をクリックします。
4. データベース・ファイルにエクスポートする管理収集オブジェクトを右クリックして、**「データベース・ファイルの作成」**を選択します。
5. **「データベース・ファイルの作成」**ダイアログで、収集オブジェクトのうちデータベース・ファイルに含めるカテゴリーを選択します。また、収集オブジェクトに含まれているデータが、指定しようとしている期間、サンプリング間隔に対応するものである限り、異なる期間、そして異なるサンプリング間隔を選択することもできます。
6. **「OK」**をクリックします。

既存の収集オブジェクトからのデータベース・ファイルの作成: 既存の管理収集オブジェクトからデータベース・ファイルにパフォーマンス・データをエクスポートできます。次のステップを実行します。

1. パフォーマンス・データが収集されるシステムの**「構成およびサービス」**を展開します。
2. **「収集サービス」**を選択します。
3. データベース・ファイルにデータをエクスポートする管理収集オブジェクトを右クリックします。
4. 初めに、**「プロパティー」**を選択すると、収集オブジェクトに含まれているデータの特性を表示することができます。「データ・プロパティー」ページには、この収集オブジェクトで収集されたデータのカテゴリーや、データが収集された間隔が示されています。これらの情報を利用して、エクスポートするデータを選択することができます。この情報を検討した後、**「OK」**をクリックします。
5. 管理収集オブジェクトをもう一度右クリックして、**「データベース・ファイルの作成」**を選択します。オンライン・ヘルプを使用しながらフィールドに必要な情報を入力します。
6. **「OK」**をクリックします。

データベース・ファイル内のデータを変換した後、 Performance Tools for iSeries ライセンス・プログラムやパフォーマンス報告生成用の他のアプリケーションを使用できます。

データ収集のカスタマイズ

収集サービスを使用してパフォーマンス・データを収集するときは、どのようなデータを収集するか、またどれほどの頻度で収集するかを制御します。これは、提供されているコレクション・プロファイルから選択できます。**「標準 (Standard)」**プロファイルは、システム・データ用の OS/400 パフォーマンス・モニター機能 (STRPFRMON コマンド) のシステム・データの設定に相当します。**「標準 + プロトコル」**プロファイルは、全データ用の STRPFRMON コマンドの設定に相当します。あるいは、**「カスタム」**を選択して、独自にカスタマイズしたプロファイルを作成することもできます。使用可能なプロファイルは他にもいくつかあります。詳細な説明については、オンライン・ヘルプを参照してください。プロファイルのカスタマイズにおいては、システム CPU、ローカル応答時間、ディスク装置、IOP (入出力処理装置) などを、使用できるデータ・カテゴリーのリストから選択できます。

データを収集する頻度は、収集するデータの各カテゴリーごとに指定できます。多くのカテゴリーでは、デフォルトの収集間隔を選択できます。これは、15 秒～60 分の範囲の事前設定値から設定できます。 (推奨される設定値は 15 分です。)

注: デフォルト値に何らかの特定時間が設定されている場合、ディスク・ストレージ、入出力処理機構、および通信関連のカテゴリーなど、明示的な時間間隔を持つカテゴリー以外のカテゴリーはすべてこの特定時間を使用します。

収集されたデータは、コレクションと呼ばれる管理収集オブジェクト (タイプ *MGTCOL) に保管されます。管理収集オブジェクトが大きくなりすぎるのを防ぐため、収集作業は一定の間隔でサイクルとして実行されるようにしてください。収集のサイクルとは、元の収集オブジェクトへのデータ収集が停止すると同時

に、新しい収集オブジェクトを作成してそこにデータを保管し始める、ということです。この間隔は、データの用途に応じて 1 ～ 24 時間の範囲で自由に指定できます。

システムに合わせて収集サービスをカスタマイズするには、次のようにします。

1. iSeries ナビゲーターで、**「マネージメント・セントラル」**の下にあるエンドポイント・システムか、**「現在の接続 (My Connections)」**(または活動状態の環境) の下にある、直接接続されているシステムを選択します。
2. **「構成およびサービス」**を展開します。
3. **「収集サービス」**を右クリックして、**「プロパティー」**を選択します。
4. 保存期間をデフォルトの 1 日より長くする場合は、**「一般」**ページでそれを指定します。保存期間が過ぎると、収集サービスは、管理収集オブジェクトとその中のデータをシステムから削除することになります。管理収集オブジェクトが作成されると、それに有効期限が割り当てられます。それで、その管理収集オブジェクトが別のライブラリーに移動されていても、有効期限が満了すれば、収集サービスはそのオブジェクトを削除します。収集サービスによって新しい収集オブジェクトに有効期限が割り当てられることを望まない場合は、**「永続」**を指定してください。その後は、これらの収集オブジェクトの削除は、手動で行わなければなりません。
   Graph History ウィンドウを表示する場合は、「グラフ (Graph)」ないし「要約 (Summary)」のコレクション保存期間を指定する必要があります。これらのオプションを使用すると、ヒストリー報告の機能が利用できるようになり、この機能によって、より長い期間における複数のシステムのメトリックを比較できるようになります。
   さらに、コレクションの保管先のパス、収集のサイクルの頻度、デフォルトの収集間隔も指定できます。収集時に自動的にデータベース・ファイルが作成されるよう選択することもできます。
5. **「収集するデータ」**タブをクリックします。
6. **「使用するコレクション・プロファイル」**に**「カスタム」**を選択します。収集の間隔は、カスタマイズ・リストに選択するカテゴリーごとに指定できます。
7. **「OK」**をクリックすると、カスタマイズの値が保管されます。

収集サービスの設定値のカスタマイズが完了したなら、もう一度**「収集サービス」**を右クリックして、**「収集サービスの開始」**を選択すると、パフォーマンス・データの収集が開始します。

収集サービスに関連した時間帯の考慮事項

パフォーマンス・データを検討して分析する際、収集が行われた現地の実時間は重要な要素となります。たとえば、1 日のピークとなる時にどのようなデータが収集されたのかを検討すれば、システムが処理した最も大きなワークロードを判断することができます。パフォーマンス・データを収集するシステムの中に、異なった時間帯に置かれているシステムがある場合は、これらの点について考慮する必要があります。

- システム・グループに対して収集サービスを開始するときは、グループ内のすべてのシステムで同時に収集サービスを開始する必要があります。一部のシステムが別の時間帯にあるために生じるシステム時間や日付の設定のずれは、一切考慮されません。
- マネージメント・セントラル・スケジューラーを使用して収集サービスを開始する場合、スケジューラーは、マネージメント・セントラルのセントラル・システムになっているマシンのシステム時間と日付に基づいてタスクを開始します。
- 各エンドポイント・システムの管理収集オブジェクトは、そのエンドポイント・システムと、使用するセントラル・システムの QTIME および QUTCOFFSET (協定世界時オフセット) システム値に基づいた開始および終了の時刻を反映します。このエンドポイント・システムとセントラル・システムが異なる時間帯にあり、両方のシステムでこれらのシステム値が正しく設定されている場合、収集オブジェクト

に報告される開始および終了の時刻は、エンドポイント・システムがある時間帯での実時間になります。つまり、開始および終了の時刻は、エンドポイント・システムの QTIME の値を、それらのイベントが起きた実際の時刻として反映します。

- パフォーマンスの収集をスケジューリングする場合には、標準時からサマータイムへ、あるいはサマータイムから標準時への境界を通過することが考えられます。そのような場合は、開始時刻をスケジューリングする際に、この時間のずれを考慮に入れる必要があります。このことを考慮に入れないでスケジューリングを行ってしまうと、実際の開始と終了の時刻が、期待していたよりも 1 時間早く、または 1 時間遅くなってしまう可能性があります。これらに加えて、管理収集オブジェクトに報告される開始と終了の時刻もこの時間のずれの影響を受けるため、サマータイムの開始と終了に合わせて QUTCOFFSET システム値を調整する必要があります。

収集サービスを使用したパフォーマンス・データの収集についての詳細は、『収集サービス』を参照してください。

収集サービスのユーザー定義カテゴリー

収集サービスのユーザー定義カテゴリー機能によって、アプリケーションはパフォーマンス・データ収集を収集サービスに統合することができます。この機能では、データ収集プログラムを作成して登録し、それを収集サービスと統合することによって、データをアプリケーションから収集できます。そして、収集サービスは収集間隔ごとにデータ収集プログラムを呼び出し、収集オブジェクトにデータを保管します。収集オブジェクトに保管されているデータにアクセスするには、以下にリストする収集オブジェクト API を使用する必要があります。データには、そのデータが収集されている間にリアルタイムでアクセスでき、また、収集オブジェクトが保持されている限りアクセスできます。

この機能をインプリメントするには、以下のようにします。

1. 収集サービスで新しいカテゴリー用のパフォーマンス・データ収集プログラムを作成します。詳しくは、『収集プログラムに関する勧告事項と要件』を参照してください。
2. 収集プログラムのジョブ記述を作成します。 QGPL のジョブ記述 QPMUSRCAT は 1 つの例ですが、デフォルト値や勧告事項を示すものではありません。
3. 新しいカテゴリーを登録して、データ収集プログラムを指定します。詳しくは以下の API の説明を参照してください。
   - 登録: QypsRegCollectorDataCategory
   - 登録取り消し: QypsDeregCollectorDataCategory

   カテゴリーを登録すると、それは収集サービスによって使用可能な収集カテゴリーのリストに含められます。
4. カテゴリーを収集サービス・プロファイルに追加し、収集サービスを循環させます。
5. 収集オブジェクトを照会するプログラムを作成します。詳しくは以下の API の説明を参照してください。
   - 活動状態の収集オブジェクト名の検索: QpmRtvActiveMgtcolName (リアルタイムでの収集オブジェクトの照会にのみ使用されます。)
   - 管理収集オブジェクト属性の検索: QpmRtvMgtcolAttrs
   - 管理収集オブジェクトのオープン: QpmOpenMgtcol
   - 管理収集オブジェクトのクローズ: QpmCloseMgtcol
   - 管理収集オブジェクト・リポジトリーのオープン: QpmOpenMgtcolRepo
   - 管理収集オブジェクト・リポジトリーのクローズ: QpmCloseMgtcolRepo
   - 管理収集オブジェクト・データの読み取り: QpmReadMgtcolData

これで、カスタマイズした収集プログラムは各収集間隔ごとに実行し、収集されたデータがコレクション・オブジェクト内にアーカイブされるようになります。

さらに、これらの API の Java バージョンもインプリメントできます。必要な Java クラスは、 IFS ディレクトリー QIBM/ProdData/OS400/CollectionServices/lib の ColSrv.jar に組み込まれています。 Java アプリケーションには、そのクラスパスにこのファイルが含まれていなければなりません。 Java インプリメンテーションについて詳しくは、 javadocs を参照してください。

インプリメンテーション例については、『例: ユーザー定義カテゴリーのインプリメント』を参照してください。

リアルタイムでの収集オブジェクトの照会

アプリケーションで、収集オブジェクトをリアルタイムで照会する必要がある場合は、照会を収集サービスと同期する必要があります。そうするには、アプリケーションでデータ・キューを作成し、それを収集サービスに登録する必要があります。一度登録されると、コレクターによって各収集間隔ごとと収集サイクルの終了に通知が送信されるようになります。アプリケーションは、データ・キューを保守する必要があります。保守には、終了時のデータ・キューの除去と異常終了の処理が含まれます。データ・キューを登録および登録取り消しするには、以下の API の説明を参照してください。

- コレクター通知の追加: QypsAddCollectorNotification
- コレクター通知の除去: QypsRmvCollectorNotification

収集プログラムに関する勧告事項と要件: 収集サービスはデータ収集プログラムを、収集サービスを開始するときに 1 回、各収集間隔ごとに 1 回、そして収集サイクルの終了にもう 1 回呼び出します。データ収集プログラムは任意のデータ収集を実行し、そのデータを収集サービスによって用意されているデータ・バッファーに戻さなければなりません。収集サービスはデータ・バッファーの提供に加えて、データ収集プログラムが収集間隔の間にいくつかの状態情報を保持することのできる作業域を用意しています。

データ収集プログラムはできるだけ速くデータを収集し、最小限の書式設定を実行します。このプログラムはデータ処理やソートは実行しません。ユーザー定義カテゴリーのデータはデータベース・ファイルには変換されませんが、収集サービスが各収集間隔の終了に自動的に CRTPFRDTA コマンドを実行して、収集オブジェクトのデータをデータベース・ファイルに追加できます。データ収集プログラムが収集間隔以内にそのタスクを完了できない場合は、それに順応して CRTPFRDTA コマンドは実行されません。

データ収集プログラムは以下のいくつかの環境で作成できます。

- OPM 言語用の *PGM。この環境は、収集オブジェクトの照会には使用できず、パフォーマンスを低下させる可能性があります。しかし、これより古いプログラミング言語ではサポートされます。
- *SRVPGM。サービス・プログラムのエントリー・ポイント。これは ILE 言語用です。
- *JVAPGM。必要な Java クラスは ColSrv.jar に含まれています。このファイルは、QIBM/ProdData/OS400/CollectionServices/lib の IFS にあります。 API の Java インプリメンテーションの説明については、 javadocs.zip ファイルをダウンロードして index.html をオープンしてください。

収集サービスは、以下の要求をデータ収集プログラムに送信します。

要求	説明
収集の開始	データ収集プログラムは、データ収集時に使用する任意のインターフェースまたは資源を初期化します。また、任意で、収集サービスによって用意されていて、収集間隔の間に状態情報が保持される作業域も初期化します。収集データの前に制御レコードを含めたいなら、データ収集プログラムがデータ・バッファーに少量のデータを書き込むことも可能です。通常、この制御レコードは、データ処理の際にデータを解釈するために使用されます。
収集間隔	収集サービスは、各収集間隔ごとに区間要求を送信します。データ収集プログラムはデータを収集して、それをデータ・バッファーに戻します。その後、収集サービスは、そのデータを収集オブジェクト内の間隔レコードに書き込みます。 データ・バッファーに対してデータの量が大きすぎる場合、データ収集プログラムは「More data」フラグを設定します。この処置が取られると、収集サービスは、別の区間要求を、それが続きであることを示す修飾子を含めて送信します。収集サービスは、各呼び出しの前に more data フラグをリセットします。このプロセスが、すべてのデータが収集オブジェクト内に移動するまで繰り返されます。
収集の終了	データ収集プログラムが帰属するカテゴリーの収集が終了すると、収集サービスはこの要求を送信します。データ収集プログラムは終結処理を実行します。そして任意で収集制御レコードを戻すことができます。また、データ収集プログラムは、収集の結果を表す戻りコードも送信します。
終結処理と終了 (シャットダウン)	収集サービスは、異常終了が必要な場合にこの要求を送信します。オペレーティング・システム資源は、データ収集プログラムのジョブが終了すると自動的に解放されますが、その他のシャットダウン操作は、データ収集プログラムによって実行されなければなりません。データ収集プログラムはいつでもこの要求を受け付けることができます。

これらのパラメーター、作業域、データ・バッファー、および戻りコードの詳細については、 QSYSINC にあるヘッダー・ファイル QPMDCPRM を参照してください。

収集オブジェクト内のデータ・ストレージ

収集オブジェクトには、各データ収集カテゴリー用のリポジトリーがあります。このリポジトリーは、収集サービスによって、そのカテゴリーの収集が開始するときに作成されます。各リポジトリーは、以下のレコードによって構成されます。

レコード	説明
制御	このオプションのレコードは、データ収集プログラムの結果として生じる最初または最後のレコードとなる可能性があり、その両方の位置に存在することがあります。普通、これにはレコード・データを解釈するために必要な任意の情報が含まれている必要があります。
間隔	各収集間隔には、間隔レコードが作成されます。これは空であっても作成されます。間隔レコードには、その収集間隔内にデータ・バッファーに書き込まれたデータが含まれています。このサイズは 4 GB 以内でなければなりません。
停止	収集サービスは自動的にこのレコードを作成してデータ収集セッションの終了を示します。収集サービスを終了または循環させずにユーザー定義カテゴリーの収集を再開した場合は、任意で終了レコードの後、追加の間隔レコードの前に制御レコードを含めることができます。

例: ユーザー定義カテゴリーのインプリメント: 以下のサンプル・プログラムは、提供されている API を使用してカスタマイズしたデータ収集を収集サービスに統合する方法を示しています。

- サンプル・データ収集プログラム (C++)
- データ収集プログラムを登録するためのサンプル・プログラム (C++)

- 収集オブジェクトを照会するためのサンプル・プログラム (Java)

コードの特記事項情報

IBM は、お客様に、すべてのプログラミング・コード・サンプルを使用することができる非独占的な使用権を許諾します。お客様は、このサンプル・コードから、お客様独自の特別のニーズに合わせた類似のプログラムを作成することができます。

すべてのサンプル・コードは、例として示す目的でのみ、IBM により提供されます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。

ここに含まれるすべてのプログラムは、現存するままの状態で提供され、いかなる保証条件も適用されません。第三者の権利の不侵害、商品性、特定目的適合性に関する黙示の保証の適用もいっさいありません。

例: データ収集プログラム: 以下のプログラムはいくつかのテスト・データを収集してデータ・バッファーに保管し、それを収集サービスは収集オブジェクトにコピーします。収集プログラムについて詳しくは、『収集プログラムに関する勧告事項と要件』を参照してください。

注: 法律上の重要な情報に関しては、『コードの特記事項情報』をお読みください。

C++ サンプル・コード

```
#include "string.h"                     // memcpy(), memset(), strlen()
#include "stdio.h"                     // printf()
#include "qpmdcprm.h"                   // data collection program interface
#include "time.h"

extern "C"
void DCPentry( Qpm_DC_Parm_t *request, char *dataBuffer,
                                                        char *workArea, int *returnCode )
{
  static  char  testData[21] = "Just some test stuff";
  int           i;



/* Print contents of request structure */

  printf( "DCP called with parameters:¥n" );
  printf( "  format name: ¥"%8.8s¥";  category name: ¥"%10.10s¥";¥n",
          request->formatName, request->categoryName );
  printf( "  rsvd1: %4.4X; req type: %d; req mod: %d; buffer len: %d;¥n",
          *(short *)(request->rsvd1), request->requestType,
          request->requestModifier, request->dataBufferLength );
  printf( "  prm offset: %d; prm len: %d; work len: %d; rsvd2: %8.8X;¥n",
          request->parmOffset, request->parmLength, request->workAreaLength,
          *(int *)(request->rsvd2) );
  printf( "  rec key: ¥"%8.8s¥"; timestamp: %8.8X %8.8X;¥n",
          request->intervalKey,
          *(int *)(request->intervalTimestamp),
          *(int *)(request->intervalTimestamp + 4) );
  printf( "  return len: %d; more data: %d; rsvd3: %8.8X %8.8X;¥n",
          request->bytesProvided, request->moreData,
          *(int *)(request->rsvd3),
          *(int *)(request->rsvd3 + 4) );

  switch ( request->requestType )
  {
  /* Write control record in the beginning of collection */
```

```
   case PM_DOBEGIN:
     printf( "doBegin(%d)¥n", request->requestModifier );
     switch ( request->requestModifier)
     {
       case PM_CALL_NORMAL:
           memcpy( dataBuffer, testData, 20 );
           *(int *)workArea = 20;
           request->moreData = PM_MORE_DATA;
           request->bytesProvided = 20;
         break;

       case PM_CALL_CONTINUE:
         if ( *(int *)workArea < 200 )
         {
           memcpy( dataBuffer, testData, 20 );
           *(int *)workArea += 20;
           request->moreData = PM_MORE_DATA;
           request->bytesProvided = 20;
         }
         else
         {
           *(int *)workArea = 0;
           request->moreData = PM_NO_MORE_DATA;
           request->bytesProvided = 0;
         }
         break;

       default:
         *returnCode = -1;
         return;
     }
     break;
/* Write control record in the end of collection */
   case PM_DOEND:
     printf( "doEnd(%d)¥n", request->requestModifier );
     switch ( request->requestModifier)
     {
       case PM_CALL_NORMAL:
           memcpy( dataBuffer, testData, 20 );
           *(int *)workArea = 20;
           request->moreData = PM_MORE_DATA;
           request->bytesProvided = 20;
         break;

       case PM_CALL_CONTINUE:
         if ( *(int *)workArea < 200 )
         {
           memcpy( dataBuffer, testData, 20 );
           *(int *)workArea += 20;
           request->moreData = PM_MORE_DATA;
           request->bytesProvided = 20;
         }
         else
         {
           *(int *)workArea = 0;
           request->moreData = PM_NO_MORE_DATA;
           request->bytesProvided = 0;
         }
         break;

       default:
         *returnCode = -1;
         return;
     }
     break;

/*Write interval record */
```

```
    case PM_DOCOLLECT:
      printf( "doCollect(%d)¥n", request->requestModifier );
      for ( i = 0; i < 10000; i++ )
        dataBuffer[i] = i % 256;
      request->bytesProvided = 10000;

      switch ( request->requestModifier)
      {
        case PM_CALL_NORMAL:
            *(time_t *)(workArea + 4) = time( NULL );
            *(int *)workArea = 1;
            request->moreData = PM_MORE_DATA;
          break;

        case PM_CALL_CONTINUE:
          *(int *)workArea += 1;
          if ( *(int *)workArea < 20 )
            request->moreData = PM_MORE_DATA;
          else
          {
            *(time_t *)(workArea + 8) = time( NULL );
            printf( "doCollect() complete in %d secs (%d bytes transferred)¥n",
                    *(time_t *)(workArea + 8) - *(time_t *)(workArea + 4), 10000 * 20 );
            request->moreData = PM_NO_MORE_DATA;
          }
          break;

        default:
          *returnCode = -1;
          return;
      }
      break;
 /* Clean-up and terminate */
    case PM_DOSHUTDOWN:
      printf( "doShutdown¥n" );
      *returnCode = 0;
      return;
      break;

    default:
      *returnCode = -1;
      return;
      break;
  }

}/* DCPentry() */
```

例: データ収集プログラムを登録するためのプログラム: 以下のプログラムは、前の例のデータ収集プログラムを収集サービスに登録します。実行すると、収集サービスのデータ収集カテゴリーのリストに、このデータ収集プログラムが表示されます。

C++ サンプル・コード

```
#include "stdlib.h"
#include "stdio.h"
#include "string.h"
#include "qypscoll.cleinc"


int main( int argc, char *argv[] )
{
    int    CCSID = 0;
    int    RC  = 0;
    Qyps_USER_CAT_PROGRAM_ATTR    *pgmAttr;
    Qyps_USER_CAT_ATTR             catAttr;
    char   collectorName[11] = "*PFR      ";
```

```
    char   categoryName[11]  = "TESTCAT   ";
    char   collectorDefn[11] = "*CUSTOM   ";  /* Register to *CUSTOM profile only */

      if ( argc > 2 )
      {
        int len = strlen( argv[2] );

        if ( len > 10 ) len = 10;
        memset( categoryName, ' ', 10 );
        memcpy( categoryName, argv[2], len );
      }

      if ( argc < 2 || *argv[1] == 'R' )
      {
        pgmAttr = (Qyps_USER_CAT_PROGRAM_ATTR *)malloc( 4096 );
        memset( pgmAttr, 0x00, sizeof(pgmAttr) );
        pgmAttr->fixedPortionSize = sizeof( Qyps_USER_CAT_PROGRAM_ATTR );
        memcpy( pgmAttr->programType,     "*SRVPGM   ", 10 );
        memcpy( pgmAttr->parameterFormat, "PMDC0100", 8 );
        memcpy( pgmAttr->ownerUserId,     "USERID    ", 10 );
        memcpy( pgmAttr->jobDescription,  "QPMUSRCAT QGPL      ", 20 );
        memcpy( pgmAttr->qualPgmSrvpgmName, "DCPTEST   LIBRARY    ", 20 );
        pgmAttr->workAreaSize = 123;
        pgmAttr->srvpgmEntrypointOffset = pgmAttr->fixedPortionSize;
        pgmAttr->srvpgmEntrypointLength = 8;
        pgmAttr->categoryParameterOffset = pgmAttr->srvpgmEntrypointOffset +
                                           pgmAttr->srvpgmEntrypointLength;
        pgmAttr->categoryParameterLength = 10;
    /* Set entry point name */
        memcpy( (char *)(pgmAttr) + pgmAttr->srvpgmEntrypointOffset,
                "DCPentry", pgmAttr->srvpgmEntrypointLength );  /* Set parameter string */
        memcpy( (char *)(pgmAttr) + pgmAttr->categoryParameterOffset,
                "1234567890", pgmAttr->categoryParameterLength );

        memset( &catAttr, 0x00, sizeof(catAttr) );
        catAttr.structureSize = sizeof( Qyps_USER_CAT_ATTR );
        catAttr.minCollectionInterval = 0;
        catAttr.maxCollectionInterval = 0;
        catAttr.defaultCollectionInterval = 30;   /* Collect at 30 second interval */
        memset( catAttr.qualifiedMsgId, ' ', sizeof(catAttr.qualifiedMsgId) );
        memcpy( catAttr.categoryDesc,
                "12345678901234567890123456789012345678901234567890", sizeof(catAttr.categoryDesc) );

        QypsRegCollectorDataCategory( collectorName,
                                      categoryName,
                                      collectorDefn,
                                      &CCSID,
                                      (char*)pgmAttr,
                                      (char*)&catAttr,
                                      &RC
                                      );
      }
      else
      if( argc >= 2 && *argv[1] == 'D' )
        QypsDeregCollectorDataCategory( collectorName, categoryName, &RC );
      else
        printf("Unrecognized option¥n");

}/* main() */
```

例: 収集オブジェクトを照会するためのプログラム: 以下のサンプル・プログラムは、QIBM/ProdData/OS400/CollectionServices/lib の ColSrv.jar ファイルに入って出荷されている Java クラスを使用して収集オブジェクト内に保管されているデータを照会する方法を示しています。

Java サンプル・コード

```
import com.ibm.iseries.collectionservices.*;

class testmco2
{
  public static void main( String argv[] )
  {
    String    objectName = null;
    String    libraryName = null;
    String    repoName = null;
    MgtcolObj mco = null;
    int       repoHandle = 0;
    int       argc = argv.length;
    MgtcolObjAttributes
              attr = null;
    MgtcolObjRepositoryEntry
              repoE = null;
    MgtcolObjCollectionEntry
              collE = null;
    int       i,j;

    if ( argc <  3 )
    {
      System.out.println("testmco2  objectName libraryName repoName");
      System.exit(1);
    }

    objectName  = argv[0];
    libraryName = argv[1];
    repoName    = argv[2];

    if ( ! objectName.equals( "*ACTIVE" ) )
      mco = new MgtcolObj( objectName, libraryName );
    else
      try
      {
        mco = MgtcolObj.rtvActive();
      } catch ( Exception e)
      {
        System.out.println("rtvActive(): Exception " + e );
        System.exit(1);
      }
    System.out.println("Object name = " + mco.getName() );
    System.out.println("Library name = " + mco.getLibrary() );

    try
    {
      attr = mco.rtvAttributes( "MCOA0100" );
    } catch ( Exception e)
    {
      System.out.println("rtvAttributes(): MCOA0100: Exception " + e );
      System.exit(1);
    }

    System.out.println("MCOA0100:  Object " + mco.getLibrary() + "/" + mco.getName() );
    System.out.println("   size = " + attr.size + " retention = " + attr.retentionPeriod +
         " interval = " + attr.dftInterval + " time created = " + attr.timeCreated +
         " time updated = " + attr.timeUpdated );
    System.out.println("   serial = " + attr.logicalPSN + " active = " + attr.isActive +
         " repaired = " + attr.isRepaired + " summary = " + attr.sumStatus +
         " repo count = " + attr.repositoryCount );
    if ( attr.repositoryInfo != null )
      for(i = 0; i < attr.repositoryCount; i++ )
      {
 repoE = attr.repositoryInfo[ i ];
 System.out.println("      name = " + repoE.name + " category = " + repoE.categoryName +
      " size = " + repoE.size );
 for( j = 0; j < repoE.collectionInfo.length; j++ )
```

```
{
  collE = repoE.collectionInfo[ j ];
  System.out.println("         startTime = " + collE.startTime + " endTime = " + collE.endTime +
       " interval = " + collE.interval );
}
     }

   try
   {
     attr = mco.rtvAttributes( "MCOA0200" );
   } catch ( Exception e)
   {
     System.out.println("rtvAttributes(): MCOA0200: Exception " + e );
     System.exit(1);
   }

   System.out.println("MCOA0200: Object " + mco.getLibrary() + "/" + mco.getName() );
   System.out.println("   size = " + attr.size + " retention = " + attr.retentionPeriod +
        " interval = " + attr.dftInterval + " time created = " + attr.timeCreated +
        " time updated = " + attr.timeUpdated );
   System.out.println("   serial = " + attr.logicalPSN + " active = " + attr.isActive +
        " repaired = " + attr.isRepaired + " summary = " + attr.sumStatus +
        " repo count = " + attr.repositoryCount );
   if ( attr.repositoryInfo != null )
     for(i = 0; i < attr.repositoryCount; i++ )
     {
repoE = attr.repositoryInfo[ i ];
System.out.println("      name = " + repoE.name + " category = " + repoE.categoryName +
     " size = " + repoE.size );
for( j = 0; j < repoE.collectionInfo.length; j++ )
{
  collE = repoE.collectionInfo[ j ];
  System.out.println("         startTime = " + collE.startTime + " endTime = " + collE.endTime +
       " interval = " + collE.interval );
}
     }

   if ( repoName.equals("NONE") )
return;

   try
   {
     mco.open();
   } catch ( Exception e)
   {
     System.out.println("open(): Exception " + e );
     System.exit(1);
   }

   try
   {
     repoHandle = mco.openRepository( repoName, "MCOD0100" );
   } catch ( Exception e)
   {
     System.out.println("openRepository(): Exception " + e );
     mco.close();
     System.exit(1);
   }
   System.out.println("repoHandle = " + repoHandle );

   MgtcolObjReadOptions  readOptions = new MgtcolObjReadOptions();
   MgtcolObjRecInfo recInfo = new MgtcolObjRecInfo();

   readOptions.option = MgtcolObjReadOptions.READ_NEXT;
   readOptions.recKey = null;
   readOptions.offset = 0;
   readOptions.length = 0;
```

```
    while ( recInfo.recStatus == MgtcolObjRecInfo.RECORD_OK )
    {
      try
      {
        mco.readData( repoHandle, readOptions, recInfo, null );
      } catch ( Exception e)
      {
        System.out.println("readData(): Exception " + e );
        mco.close();
        System.exit(1);
      }

      if( recInfo.recStatus == MgtcolObjRecInfo.RECORD_OK )
      {
        System.out.print("Type = " + recInfo.recType );
        System.out.print(" Key = " + recInfo.recKey );
        System.out.println(" Length = " + recInfo.recLength );
      }

    }/* while ... */

    mco.closeRepository( repoHandle );
    mco.close();

  }/* main() */

}/* class testmco2 */
```

収集オブジェクトの管理

収集サービスを使用してパフォーマンス・データを収集する場合、それぞれの収集の結果は 1 つのオブジェクトの中に保管されます。管理収集オブジェクトに入っているデータの要約は、次のようにして調べることができます。

1. iSeries ナビゲーターで、**「マネージメント・セントラル」**の下にあるエンドポイント・システムか、**「現在の接続 (My Connections)」**(または活動状態の環境) の下にある、直接接続されているシステムを選択します。
2. **「構成およびサービス」**を展開します。
3. **「収集サービス」**を選択します。
4. リストの中から任意の管理収集オブジェクトを右クリックし、**「プロパティー」**を選択します。そのコレクションについての一般的な情報と、そこに入っているデータの要約情報が表示されます。

任意の収集オブジェクトを右クリックして、「データベース・ファイルの作成」を選択すると、データベース・ファイルに含めるデータのカテゴリー、コレクション期間内の時間範囲、およびサンプリング間隔を指定することができます。

任意の収集オブジェクトを右クリックして「グラフ・ヒストリー (Graph History)」を選択すると、管理対象オブジェクト内のデータをグラフで表示することができます。

古い管理収集オブジェクトの削除または保存

システムから収集オブジェクトを削除するには、オブジェクトを右クリックして**「削除」**を選択します。手動で削除されないオブジェクトは、有効期限の満了後、収集サービスによって自動的に削除されます。

収集サービスは、**循環済み**の管理収集オブジェクトしか削除しません。**循環済み**とは、収集サービスがデータ収集とそのオブジェクトへの保管を停止したことを意味しています。各管理収集オブジェクトの状況は、**「構成およびサービス」**を展開して**「収集サービス」**を選択すると表示される、収集オブジェクトのリストに示されます。

有効期限の満了日に達した循環済み収集オブジェクトは、次に収集サービスが開始された時点で削除されます。有効期限は、管理収集オブジェクトに関連付けられています。それで、その管理収集オブジェクトが別のライブラリーに移動されていても、有効期限が満了すれば、収集サービスはそのオブジェクトを削除します。

各管理収集オブジェクトの有効期限の満了日は、そのコレクション・オブジェクトの「プロパティー」の中に示されます。その「プロパティー」ページで日付を変更するだけで、オブジェクトがシステムに存在する時間をもっと長くすることができます。リストの中から任意の管理収集オブジェクトを右クリックし、**「プロパティー」**を選択すると、そのコレクションについての情報が表示されます。収集サービスによってその管理収集オブジェクトが自動的に削除されることを望まない場合は、**「永続」**を指定してください。

ユーザー定義トランザクション

収集サービスおよび Performance Explorer は現在、ユーザーのアプリケーションで定義するパフォーマンス・データの収集ができます。備えられている API を用いるならば、収集サービスを使用してトランザクション・データを定期的にスケジュールされているサンプル・データに統合することができ、さらに Performance Explorer を実行することによりトランザクションに関するトレース・レベルのデータを取得できます。

詳細記述および使用上の注意については、以下の API の説明を参照してください。

- トランザクション開始 : QYPESTRT、qypeStartTransaction
- トランザクション終了 : QYPEENDT、qypeEndTransaction
- トランザクション・ログ : QYPELOGT、qypeLogTransaction (Performance Explorer のみによる使用)
- トレース・ポイント追加 : QYPEADDT、qypeAddTracePoint (Performance Explorer のみによる使用)

注 : アプリケーションを一度装備するだけで十分です。収集サービスおよび Performance Explorer は同じ API 呼び出しを使用して、さまざまなタイプのパフォーマンス・データを収集します。

ユーザー定義トランザクション・データを収集サービスに統合する

収集サービスの構成で、ユーザー定義トランザクションを収集用カテゴリーとして選択することができます。収集サービスはその後、すべての収集間隔においてトランザクション・データを収集し、そのデータを収集オブジェクトに保管します。 CRTPFRDTA コマンドは、このデータをユーザー定義のトランザクション・データベース・ファイル、 QAPMUSRTNS にエクスポートします。収集サービスは、トランザクション・タイプに基づいてデータを編成します。必要な数のトランザクション・タイプを指定することができますが、収集サービスは、最初の 15 のトランザクション・タイプのみを報告します。それ以外のトランザクション・タイプのデータは、*OTHER トランザクション・タイプとして結合および保管されます。すべての収集間隔において、収集サービスは、それぞれの固有のジョブのトランザクション・タイプごとに 1 つのレコードを作成します。詳細記述については、『トランザクションの開始 API』にある使用上の注意を参照してください。

収集サービスは、トランザクション応答時間などの一般トランザクション・データを収集します。トランザクションのために使用される SQL ステートメントの数、またはその他の増分測定機能のような、アプリケーションの特定のデータを追跡することのできるオプションのアプリケーション定義によるカウンターを、最大 16 まで組み込むこともできます。アプリケーションは、トランザクション開始 API を使用して新規トランザクションの開始を示す必要があり、収集サービスにトランザクション・データを引き渡すために、対応するトランザクション終了 API を組み込む必要があります。詳しくは、QAMUSRTNS ファイル記述および API 記述を参照してください。

実装例については、『C++』または『Java』の例を参照してください。

注: 法律上の重要な情報に関しては、『コードの特記事項情報』をお読みください。

Performance Explorer によるユーザー定義トランザクションのトレース情報の収集

Performance Explorer セッションの時に、トランザクションの開始、終了、およびログ API を使用して、トレース・レコードを作成することができます。 Performance Explorer は、現行のスレッドのシステム・リソース使用状況 (CPU 稼働率、入出力、捕獲/ロック活動など) を、それらのトレース・レコードに保管します。加えて、アプリケーション固有のパフォーマンス・データを組み込む選択をするならば、その後、それらの各 API にある Performance Explorer にそれを送信することができます。さらに、トレース・ポイント追加 API を使用して、 Performance Explorer がトレース・データを収集する必要があるアプリケーション固有のイベントを識別します。

トランザクションのための Performance Explorer セッションを開始するには、 Performance Explorer 定義の (OSEVT) パラメーターで *USRTRNS を指定します。 ENDPEX コマンドを入力した後、Performance Explorer は、アプリケーションによって提供されたデータを、 QAYPEMIUSR Performance Explorer データベース・ファイルの QMUDTA フィールドに書き込みます。開始、終了、およびすべてのログ・レコードのためにシステムにより備えられたデータは、 QAYPEMIUSR および QAYPETIDX データベース・ファイルに保管されます。

詳細記述については、『トランザクションの開始 API』の記述にある API 記述および使用上の注意を参照してください。

C++ の例 : ユーザー定義トランザクションを収集サービスに統合する: 以下の C++ のプログラム例では、トランザクション開始およびトランザクション終了 API を使用して、ユーザー定義のトランザクション・パフォーマンス・データを収集サービスに統合する方法を示しています。

注: 法律上の重要な情報に関しては、『コードの特記事項情報』をお読みください。

```
//*********************************************************************
// tnstst.C
//
// This example program illustrates the use
// of the Start/End Transaction APIs (qypeStartTransaction,
// qypeEndTransaction).
//
//
// This program can be invoked as follows:
//   CALL lib/TNSTST PARM('threads' 'types' 'transactions' 'delay')
//     where
//       threads      = number of threads to create (10000 max)
//       types        = number of transaction types for each thread
//       transactions = number of transactions for each transaction
//                      type
//       delay        = delay time (millisecs) between starting and
//                      ending the transaction
//
// This program will create "threads" number of threads. Each thread
// will generate transactions in the same way. A thread will do
// "transactions" number of transactions for each transaction type,
// where a transaction is defined as a call to Start Transaction API,
// then a delay of  "delay" millisecs, then a call to End Transaction
// API. Thus, each thread will do a total of "transactions" * "types"
// number of transactions. Each transaction type will be named
// "TRANSACTION_TYPE_nnn" where nnn ranges from 001 to "types". For
// transaction type n, there will be  n-1 (16 max) user-provided
// counters reported, with counter m reporting m counts for each
// transaction.
//
// This program must be run in a job that allows multiple threads
// (interactive jobs typically do not allow multiple threads). One
```

```
// way to do this is to invoke the program using the SBMJOB command
// specifying ALWMLTTHD(*YES).
//
//*********************************************************************

#define _MULTI_THREADED

// Includes
#include "pthread.h"
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "qusec.h"
#include "lbcpynv.h"
#include "qypesvpg.h"

// Constants
#define maxThreads 10000

// Transaction pgm parm structure
typedef struct
{
  int types;
  int trans;
  int delay;
} tnsPgmParm_t;

// Error code structure
typedef struct
{
  Qus_EC_t error;
  char     Exception_Data[100];
} error_code_t;


//*********************************************************************
//
// Transaction program to run in each secondary thread
//
//*********************************************************************

void *tnsPgm(void *parm)
{
  tnsPgmParm_t *p = (tnsPgmParm_t *)parm;

  char tnsTyp[] = "TRANSACTION_TYPE_XXX";
  char pexData[] = "PEX";
  unsigned int pexDataL = sizeof(pexData) - 1;
  unsigned long long colSrvData[16] = {1,2,3,4,5,6,7,8,
                                       9,10,11,12,13,14,15,16};
  unsigned int colSrvDataL;
  char tnsStrTim[8];

  struct timespec ts = {0, 0};

  error_code_t errCode;

  _DPA_Template_T target, source; // Used for LBCPYNV MI instr

  unsigned int typCnt;
  unsigned int tnsCnt;
  int rc;


  // Initialize error code
  memset(&errCode, 0, sizeof(errCode));
  errCode.error.Bytes_Provided = sizeof(errCode);
```

```
  // Initialize delay time
  ts.tv_sec  = p->delay / 1000;
  ts.tv_nsec = (p->delay % 1000) * 1000000;

  // Loop doing transactions
  for (tnsCnt = 1; tnsCnt <= p->trans; tnsCnt++)
  {
    for (typCnt = 1; typCnt <= p->types; typCnt++)
    {
      // Set number field in transaction type
      source.Type = _T_UNSIGNED;
      source.Length = 4;
      source.reserved = 0;
      target.Type = _T_ZONED;
      target.Length = 3;
      target.reserved = 0;
      _LBCPYNV(tnsTyp + 17, &target, &typCnt, &source);

      // Set Coll Svcs data length in bytes
      colSrvDataL = (typCnt <= 16) ? (typCnt - 1) : 16;
      colSrvDataL = colSrvDataL * 8;

      // Call Start Transaction API
      qypeStartTransaction(tnsTyp,
                           (unsigned int *)&tnsCnt,
                           pexData,
                           (unsigned int *)&pexDataL,
                           tnsStrTim,
                             &errCode);

      // Delay specified amount
      rc = pthread_delay_np(&ts);

      // Call End Transaction API
      qypeEndTransaction(tnsTyp,
                           (unsigned int *)&tnsCnt,
                           pexData,
                           (unsigned int *)&pexDataL,
                           tnsStrTim,
                           (unsigned long long *)&colSrvData[0],
                           (unsigned int *)&colSrvDataL,
                             &errCode);
    }
  }

  return NULL;
}


//*********************************************************************
//
// Main program to run in primary thread
//
//*********************************************************************

void main(int argc, char *argv[])
{
  // Integer version of parms
  int threads;  // # of threads
  int types;    // # of types
  int trans;    // # of transactions
  int delay;    // Delay in millisecs

  pthread_t threadHandle[maxThreads];
  tnsPgmParm_t tnsPgmParm;
  int rc;
```

```
  int           i;


  // Verify 4 parms passed
  if (argc != 5)
  {
    printf("Did not pass 4 parms¥n");
    return;
  }

  // Copy parms into integer variables
  threads = atoi(argv[1]);
  types   = atoi(argv[2]);
  trans   = atoi(argv[3]);
  delay   = atoi(argv[4]);

  // Verify parms
  if (threads > maxThreads)
  {
    printf("Too many threads requested¥n");
    return;
  }

  // Initialize transaction pgm parms (do not modify
  // these while threads are running)
  tnsPgmParm.types = types;
  tnsPgmParm.trans = trans;
  tnsPgmParm.delay = delay;

  // Create threads that will run transaction pgm
  for (i=0; i < threads; i++)
  {
    // Clear thread handle
    memset(&threadHandle[i], 0, sizeof(pthread_t));
    // Create thread
    rc = pthread_create(&threadHandle[i],      // Thread handle
                        NULL,                  // Default attributes
                        tnsPgm,                // Start routine
                        (void *)&tnsPgmParm);  // Start routine parms
    if (rc != 0)
      printf("pthread_create() failed, rc = %d¥n", rc);
  }

  // Wait for each thread to terminate
  for (i=0; i < threads; i++)
  {
    rc=pthread_join(threadHandle[i],  // Thread handle
                    NULL);            // No exit status
  }

}  /* end of Main */
```

Java の例 : ユーザー定義トランザクションを収集サービスに統合する: 以下の Java のプログラム例では、トランザクション開始およびトランザクション終了 API を使用して、ユーザー定義のトランザクション・パフォーマンス・データを収集サービスに統合する方法を示しています。

注: 法律上の重要な情報に関しては、『コードの特記事項情報』をお読みください。

```
import com.ibm.iseries.collectionservices.PerformanceDataReporter;


// parameters:
//   number of TXs per thread
//   number of threads
//   log|nolog
//   enable|disable
//   transaction seconds
```

```
public class TestTXApi
{
  static TestTXApiThread[]    thread;

  static private String[] TxTypeString;
  static private byte[][] TxTypeArray;

  static private String   TxEventString;
  static private byte[]   TxEventArray;

  static
  {
    int i;

    // initialize transaction type strings and byte arrays

      TxTypeString = new String[20];
      TxTypeString[ 0] = "Transaction type  00";
      TxTypeString[ 1] = "Transaction type  01";
      TxTypeString[ 2] = "Transaction type  02";
      TxTypeString[ 3] = "Transaction type  03";
      TxTypeString[ 4] = "Transaction type  04";
      TxTypeString[ 5] = "Transaction type  05";
      TxTypeString[ 6] = "Transaction type  06";
      TxTypeString[ 7] = "Transaction type  07";
      TxTypeString[ 8] = "Transaction type  08";
      TxTypeString[ 9] = "Transaction type  09";
      TxTypeString[10] = "Transaction type  10";
      TxTypeString[11] = "Transaction type  11";
      TxTypeString[12] = "Transaction type  12";
      TxTypeString[13] = "Transaction type  13";
      TxTypeString[14] = "Transaction type  14";
      TxTypeString[15] = "Transaction type  15";
      TxTypeString[16] = "Transaction type  16";
      TxTypeString[17] = "Transaction type  17";
      TxTypeString[18] = "Transaction type  18";
      TxTypeString[19] = "Transaction type  19";

      TxTypeArray = new byte[20][];
      for ( i = 0; i < 20; i++ )
        try
        {
          TxTypeArray[i] = TxTypeString[i].getBytes("Cp037");
        } catch(Exception e)
        {
          System.out.println("Exception ¥"" + e + "¥" when converting");
        }

  }/* static */



  public static void main( String[] args )
  {
    int     numberOfTXPerThread;
    int     numberOfThreads;
    boolean log;
    boolean enable;
    int     secsToDelay;

      // process parameters
      if ( args.length >= 5 )
 try
        {
          numberOfTXPerThread = Integer.parseInt( args[0] );
          numberOfThreads     = Integer.parseInt( args[1] );

          if ( args[2].equalsIgnoreCase( "log" ) )
     log = true;
          else
          if ( args[2].equalsIgnoreCase( "nolog" ) )
            log = false;
          else
          {
            System.out.println( "Wrong value for 3rd parameter!" );
```

```
          System.out.println( "¥tshould be log|nolog" );
          return;
        }

        if ( args[3].equalsIgnoreCase( "enable" ) )
    enable = true;
        else
        if ( args[3].equalsIgnoreCase( "disable" ) )
          enable = false;
        else
        {
          System.out.println( "Wrong value for 4th parameter!" );
          System.out.println( "¥tshould be enable|disable" );
          return;
        }

        secsToDelay = Integer.parseInt( args[4] );

      } catch (Exception e)
      {
        System.out.println( "Oops! Cannot process parameters!" );
        return;
      }
    else
    {
      System.out.println( "Incorrect Usage." );
      System.out.println( "The correct usage is:" );
      System.out.println( "java TestTXApi numberOfTXPerThread numberOfThreads log|nolog enable|disable secsToDelay");
      System.out.println("¥tlog will make the program cut 1 log transaction per start / end pair");
      System.out.println("¥tdisable will disable performance collection to minimize overhead");
      System.out.print("¥nExample: ¥"java TestTXApi 10000 100 log enable 3¥" will call " );
      System.out.println("cause 10000 transactions for each of 100 threads");
      System.out.println("with 3 seconds between start and end of transaction");
      System.out.println("Plus it will place additional log call and will enable reporting." );
      return;
    }

    System.out.println( "Parameters are processed:" );
    System.out.println( "¥tnumberOfTxPerThread = " + numberOfTXPerThread );
    System.out.println( "¥tnumberOfThreads = " + numberOfThreads );
    System.out.println( "¥tlog = " + log );
    System.out.println( "¥tenable = " + enable );
    System.out.println( "¥tsecsToDelay = " + secsToDelay );

  // cause initialization of a PerformanceDataReporter class
    {
      PerformanceDataReporter pReporter = new PerformanceDataReporter();
pReporter.enableReporting();
    }

  TestTXApi t = new TestTXApi( );

    System.out.println( "¥nAbout to start ..." );
    t.prepareTests( numberOfTXPerThread, numberOfThreads, log, enable, secsToDelay );

  long startTime = System.currentTimeMillis();

    t.runTests( numberOfThreads );

    // wait for threads to complete
    for ( int i = 0; i < numberOfThreads; i++ )
      try
      {
        thread[i].join( );
      } catch(Exception e)
      {
        System.out.println( "***Exception ¥"" + e + "¥" while joining thread " + i );
      }

  long endTime = System.currentTimeMillis();

    System.out.println( "¥nTest runtime for " + ( numberOfTXPerThread * numberOfThreads) +
                        " TXs was " + ( endTime - startTime ) + " msec" );

 }/* main() */
```

```
  private void prepareTests( int numberOfTxPerThread,
                             int numberOfThreads, boolean log, boolean enable, int secsToDelay )
  {
    System.out.println( "Creating " + numberOfThreads + " threads");
    thread = new TestTXApiThread[numberOfThreads];
    for ( int i = 0; i < numberOfThreads; i++ )
      thread[i] = new TestTXApiThread( i, numberOfTxPerThread,
                                       log, enable, secsToDelay );

  }/* prepareTests() */

  private void runTests( int numberOfThreads )
  {
    for ( int i = 0; i < numberOfThreads; i++ )
      thread[i].start( );

  }/* runTests() */

  private class TestTXApiThread extends Thread
  {
    private int     ordinal;
    private int     numberOfTxPerThread;
    private boolean log;
    private boolean enable;
    private int     secsToDelay;

    private PerformanceDataReporter     pReporter;

    private long    timeStamp[];
    private long    userCounters[];


      public TestTXApiThread( int ordinal, int numberOfTxPerThread,
                              boolean log, boolean enable, int secsToDelay )
      {
        super();
        this.ordinal             = ordinal;
        this.numberOfTxPerThread = numberOfTxPerThread;
        this.log                 = log;
        this.enable              = enable;
        this.secsToDelay         = secsToDelay;

          pReporter = new PerformanceDataReporter( false );
          if ( enable )
            pReporter.enableReporting();
          timeStamp = new long[1];
          userCounters = new long[16];
          for ( int i = 0; i < 16; i++ )
              userCounters[i] = i;

      }/* constructor */

      public void run()
      {
        int i;

              for ( i = 0; i < numberOfTxPerThread; i++ )
              {
                pReporter.startTransaction( TxTypeArray[i%20], i, TxTypeArray[i%20], 20, timeStamp );
//                pReporter.startTransaction( TxTypeArray[i%20], i, TxTypeString[i%20], timeStamp );
                if ( log )
                  pReporter.logTransaction( TxTypeArray[i%20], i, TxTypeArray[i%20], 20 );
//                  pReporter.logTransaction( TxTypeArray[i%20], i, TxTypeString[i%20] );
                if (secsToDelay > 0)
                  try
                  {
                    Thread.sleep(secsToDelay * 1000);
                  } catch(Exception e) { }
                pReporter.endTransaction( TxTypeArray[i%20], i, TxTypeArray[i%20], 20, timeStamp,
                                          userCounters );
//                pReporter.endTransaction( TxTypeArray[i%20], i, TxTypeString[i%20], timeStamp,
//                                            userCounters );
              }

      }/* run() */
```

```
 }/* class TestTXApiThread */

}/* class TestTXApi */
```

パフォーマンス・データ・ファイル

パフォーマンス・データとは、応答時間およびスループットを理解するために使用できるシステム (システムのネットワーク) の操作に関する一連の情報です。パフォーマンス・データを使用することにより、プログラム、システム属性、および操作に調整を加えることができます。これらの調整を行うことにより応答時間とスループットを向上させることができます。また調整は、システム、操作、またはプログラムへの特定の変更の影響を予測する上で役立ちます。

収集サービスは、管理収集オブジェクト (*MGTCOL) にパフォーマンス・データを収集します。パフォーマンス・データの作成 (CRTPFRDTA) コマンドは、その収集オブジェクトからデータを処理し、パフォーマンス・データベース・ファイルに結果を保管します。データベース・ファイルは、以下のカテゴリーに分けられます。

時間間隔データを含むパフォーマンス・データ・ファイル
これらのファイルには、それぞれの間隔を収集されるパフォーマンス・データが含まれています。それらのファイルのリストについては、『時間間隔データを含むパフォーマンス・データ・ファイル』を参照してください。そこには、各ファイルに関する要旨および完全な情報へのリンクも含まれています。これらのファイル内のデータがどこから収集されるかを理解するには、『システム・カテゴリーとファイルの関係』を参照してください。これらのファイルを表示するときに、 ファイルの略語が役立つことにお気付きになることでしょう。

構成データ・ファイル
構成データは、セッションごとに一度収集されます。これらのファイル内のデータがどこから収集されるかを理解するには、『システム・カテゴリーとファイルの関係』を参照してください。 構成データ・ファイル内に QAPMCONF、QAPMHDWR、および QAPMSBSD ファイルを見つけることができます。

トレース・データ・ファイル
トレース・データは、データのトレースを選択する場合に限り収集されます。 トレース・データ・ファイル内に QAPMDMPT を見つけることができます。

バイト数およびバッファー位置などの追加のフィールド情報は、システム上で使用可能なファイル・フィールド記述の表示 (DSPFFD) コマンドを使用すれば、利用可能になります。たとえば、コマンド行で以下のコマンドを入力します。

```
DSPFFD file(QSYS/QAPMCONF)
```

iSeries のパフォーマンスについての詳細は、『パフォーマンス』を参照してください。

時間間隔データを含むパフォーマンス・データ・ファイル

パフォーマンス・データ・ファイルに関する完全な情報を表示するには、以下のリストから表示したいファイルを選択します (アルファベット順で示されています) 。

ファイル	説明
QAPMAPPN	APPN データ
QAPMASYN	非同期統計 (リンクにつき 1 つ)
QAPMBSC	2 進データ同期統計 (リンクにつき 1 つ)

ファイル	説明
QAPMBUS	バス・カウンター (バスにつき 1 つ)
QAPMCIOP	通信 IOP データ (IOP につき 1 つ)
QAPMDDI	Distributed Digital Interface (DDI) データ
QAPMDIOP	ストレージ・デバイス IOP データ (IOP につき 1 つ)
QAPMDISK	ディスク装置データ (読取/書込ヘッドにつき 1 つ)
QAPMDOMINO	ドミノ (iSeries 版) データ (ドミノ・サーバーにつき 1 つのレコード)
QAPMECL	トークンリング・ファイル項目 (リンクにつき 1 つ)
QAPMETH	イーサネット統計 (リンクにつき 1 つ)
QAPMFRLY	フレーム・リレー・データ (リンクにつき 1 つ)
QAPMHDLC	HDLC 統計 (リンクにつき 1 つ)
QAPMHTTPB	IBM HTTP server (Apache 駆動) の基本データ (サーバーにつき 1 つ)
QAPMHTTPD	IBM HTTP server (Apache 駆動) の詳細データ (サーバー構成要素につき 1 つ)
QAPMIDLC	統合サービスのディジタル・ネットワークのデータ・リンク制御ファイル項目
QAPMIOPD	命令拡張 IOP データ (特殊 IOP 用の追加データ) (IOP につき 1 つ)
QAPMJOBMI	MI ジョブ・データ (ジョブ、タスク、またはスレッドにつき 1 つのレコード)。この文書を使用するときに、タスク・タイプ・エクステンダーに関する情報が役立ちます。
QAPMJOBOS	ジョブ・オペレーティング・システム・データ (ジョブにつき 1 つのレコード)
QAPMJOBS および QAPMJOBL	ジョブ・データ (ジョブ、タスク、またはスレッドにつき 1 つのレコード)
QAPMJOBWT	ジョブ、タスク、およびスレッド待ち条件
QAPMJOBWTD	ファイル QAPMJOBWT にあるカウンター設定の説明。
QAPMJSUM	ジョブ・グループごとのジョブの要約データ (ジョブ・グループにつき 1 つのレコード)
QAPMLAPD	統合サービス・ディジタル・ネットワークの LAPD ファイル項目 (リンクにつき 1 つ)
QAPMLIOP	平衡型ワークステーション制御装置データ (物理制御装置につき 1 つ)
QAPMMIOP	多機能 IOP (IOP につき 1 つ)
QAPMPOOL および QAPMPOOLL	主記憶装置データ (システム記憶域プールにつき 1 つ)
QAPMPOOLB	記憶域プール・データ (リンクにつき 1 つ)
QAPMPOOLT	記憶域プールの調整データ (リンクにつき 1 つ)
QAPMPPP	Point-to-Point Protocol データ (リンクにつき 1 つ)
QAPMRESP	ローカル・ワークステーションの応答時間 (ワークステーションにつき 1 つ)
QAPMRWS	リモート・ワークステーションの応答時間
QAPMSAP	TRLAN、イーサネット、DDI、およびフレーム・リレー SAP ファイル項目
QAPMSNA	SNA データ
QAPMSNADS	SNADS データ (SNADS ジョブにつき 1 つ)
QAPMSTND	DDI 端末データ
QAPMSTNE	イーサネット端末ファイル項目
QAPMSTNL	トークンリング端末ファイル項目
QAPMSTNY	フレーム・リレー端末ファイル項目

ファイル	説明
QAPMSYS および QAPMSYSL	システム・パフォーマンス・データ
QAPMSYSCPU	システム CPU の使用状況データ
QAPMSYSTEM	システム・レベルのパフォーマンス・データ
QAPMTCP	TCP/IP データ
QAPMTCPIFC	個々の TCP/IP インターフェースに関する TCP/IP データ
QAPMUSRTNS	ユーザー定義トランザクション・データ (各ジョブは、トランザクションのタイプごとに 1 つのレコードを持つ)
QAPMX25	X.25 統計 (リンクにつき 1 つ)

パフォーマンス・データ・ファイル: ファイルの略語

パフォーマンス・データ・ファイルは、フィールドおよびバイト・データ表内で略語を使用します。これらの略語には、以下のものが含まれます。

略語	説明
1 次ファイル	これらのファイルは、カテゴリーと関連付けられ、カテゴリーから生成されます。
C	属性列内の文字
PD	属性列内のパック 10 進数
Z	属性列内のゾーン 10 進数
IOP	入出力処理機構。ディスク、ディスプレイ装置、および通信回線など、ホスト・システムとその他の装置との間の活動を制御する処理機構。
DCE	データ回線終端装置。
MAC	中間アクセス制御。通信 IOP 内のエンティティー。
LLC	論理リンク制御。通信 IOP 内のエンティティー。
ビーコン・フレーム	リングが実行不能のときに送信されるフレーム。
タイプ II フレーム	システム・ネットワーク体系 (SNA) によって使用される接続指向のフレーム (情報フレーム)。
I フレーム	情報フレーム。

パフォーマンス・データ・ファイル: 収集サービス・システム・カテゴリーとファイルの関係

収集サービスを使用してパフォーマンス・データを収集する場合、そのデータは管理収集 (*MGTCOL) オブジェクトに保管されます。 CRTPFRDTA コマンドは、管理収集オブジェクトからデータをエクスポートしてから、 パフォーマンス・データ・ファイルにデータを書き込みます。収集サービスが別個に制御して収集するそれぞれのデータ・タイプは、データ・カテゴリーと表現します。それぞれのデータ・カテゴリーには、1 つまたは複数のパフォーマンス・データ・ファイルに書き込まれるデータを含んでいるかまたは提供します。作成されるデータベース・ファイルまたはメンバーの場合、ファイルまたはメンバーが依存しているカテゴリー (またはカテゴリーのグループ) が存在しており、CRTPFRDTA によって処理されなければなりません。下記の表は、カテゴリーとファイルの関係を示しています。以下の 3 つのタイプの関係があります。

関係	説明
1 次ファイル	これらのファイルは、カテゴリーと関連付けられ、カテゴリーから生成されます。
互換性ファイル	これらのファイルは、パフォーマンス・データベース互換性を提供するための 1 次ファイルを、以前のファイル構造と結合する論理ファイルです。システムが、関係するすべてのファイル (1 次カテゴリー) を生成する場合、互換性ファイルも生成されます。
2 次ファイル	これらのファイルは、カテゴリーまたは 1 次ファイル内に含まれているデータから派生した一部のデータと関連付けられており、それらを含んでいます。ただし、これらのファイルは、そのカテゴリーによって制御されません。

ユーザーは、以下のことに注意する必要があります。

1. CRTPFRDTA コマンドは、データベース・ファイルが、選択されたカテゴリー用の 1 次ファイルである場合にのみ、そのファイルを生成します。
2. 1 次ファイルが複数のカテゴリー用にリストされている場合、ファイルを生成するためにそれらの各カテゴリーを選択する必要があります。
3. 1 つのカテゴリー用の 1 次ファイルが別のカテゴリー用の 2 次ファイルとしてリストされている場合、生成されたデータベース・ファイル内の情報を完全なものにするために 2 番目のカテゴリーを選択する必要があります。たとえば、下記の表で示しているように、 QAPMECL 用の完全なデータベース・ファイルを生成するには、*CMNBASE と *CMNSTN の両方を選択する必要があります。
4. 関連したすべての 1 次ファイルを生成する場合にのみ、システムは互換性ファイルを生成します。

下記の表では、システム・カテゴリーとパフォーマンス・データベース・ファイルとの間の関係を示しています。

カテゴリー	1 次ファイル	互換性ファイル	2 次ファイル
*SYSBUS	QAPMBUS		
*POOL	QAPMPOOLB	QAPMPOOLL	
*POOLTUNE	QAPMPOOLT	QAPMPOOLL	
*HDWCFG	QAPMHDWR		
*SUBSYSTEM	QAPMSBSD		
*SYSCPU	QAPMSYSCPU	QAPMSYSL	
*SYSLVL	QAPMSYSTEM	QAPMSYSL	
*JOBMI	QAPMJOBMI QAPMJOBWT QAPMJOBWTD QAPMJSUM	QAPMJOBL QAPMSYSL	QAPMSYSTEM
*JOBOS	QAPMJOBOS QAPMJSUM	QAPMJOBL QAPMSYSL	QAPMSYSTEM
*SNADS	QAPMSNADS		
*DISK	QAPMDISK		QAPMSYSTEM
*IOPBASE	QAPMLIOP QAPMDIOP QAPMCIOP QAPMMIOP		

*IPCS	QAPMIOPD QAPMTSK		
*CMNBASE	QAPMASYN QAPMBSC QAPMDDI QAPMECL QAPMETH QAPMFRLY QAPMHDLC QAPMIDLC QAPMLAPD QAPMPPP QAPMX25		
*CMNSTN	QAPMSTND QAPMSTNE QAPMSTNL QAPMSTNY なし		QAPMDDI QAPMETH QAPMECL QAPMFRLY QAPMX25
*CMNSAP	QAPMSAP		
*LCLRSP	QAPMRESP		
*APPN	QAPMAPPN		
*SNA	QAPMSNA		
*EACACHE	なし		QAPMDISK (注を参照)
*TCPBASE	QAPMTCP		
*TCPIFC	QAPMTCPIFC		
*DOMINO	QAPMDOMINO		
*HTTP	QAPMHTTPB QAPMHTTPD		
*USRTNS	QAPMUSRTNS		
注: このカテゴリーは、CRTPFRDTA には選択可能ではありません。ただし、追加データが *DISK カテゴリーによって報告されるようにします。			

パフォーマンス・データ・ファイル: 構成データベース・ファイルのフィールドのデータ

構成データは、セッションごとに一度収集されます。以下のパフォーマンス・データ・ファイルは、ファイル名、簡単な説明、およびシステム構成データ、サブシステム・データ、およびハードウェア構成データに関するフィールド・データの詳細 (提供される場合) への参照を示しています。 収集サービスがこのファイルを作成する方法、およびデータが収集される元の場所については、『システム・カテゴリーとファイルの関係』を参照してください。

フィールド名	説明
QAPMCONF	システム構成データ。
QAPMHDWR	システム・ハードウェア構成。
QAPMSBSD	サブシステム・データ。フィールドおよびバイト・データはありません。

このトピックに関する詳細情報を見つけるには、『パフォーマンス・データベース・ファイル』の概要を参照してください。

パフォーマンス・データベース・ファイル: トレース・データベース・ファイルのフィールド・データ

トレース・データには、システム内部のトレース・データが含まれます。これは、特定のジョブやトランザクションに関して付加的な情報を得るために収集される詳細データです。この種のデータは、 Performance Tools ライセンス・プログラムを使用して分析を行わない限り、収集されることはありません。パフォーマンス・トレースの開始 (STRPFRTRC) コマンドの使用時にシステムがサポートしているのは、以下のパフォーマンス・データ・ファイルです。

ファイル名	説明
QAPMDMPT	システム・トレース・データ (フィールドやバイトに関する詳細は含まれません)。

このトピックに関する詳細情報を見つけるには、『パフォーマンス・データベース・ファイル』の概要を参照してください。

iSeries ナビゲーターのモニター

iSeries ナビゲーターに組み込まれているモニターは、 収集サービス・データを使用して、特定のシステム・パフォーマンスの要素を追跡します。さらに、CPU 稼働率やジョブのステータスなど、特定のイベントが発生したときに、指定されたアクションを実行することができます。モニターを使用することによって、複数のシステムやグループのシステム・パフォーマンスをリアルタイムに表示および管理できます。

モニターを使用する場合、まずモニターを開始してから、 iSeries ナビゲーターや PC からサーバーに対して他のタスクを実行できます。実際、自分の PC をオフにすることさえ可能です。その間も iSeries ナビゲーターはモニターを継続し、しきい値コマンドやアクションも指定どおりに実行します。停止処置を実行するまでモニターは実行し続けます。さらにモニターを使用して、 iSeries ナビゲーター (ワイヤレス対応) にアクセスし、パフォーマンスをリモート側で管理することもできます。

iSeries ナビゲーターには、以下のタイプの iSeries ナビゲーターが備えられています。

システム・モニター

発生時または最高 1 時間まで、パフォーマンス・データを収集して表示します。詳細グラフは、発生時にサーバーで何が起こっているかを視覚的に示すのに役立ちます。システム・パフォーマンスの特定の局面を正確に示すために、さまざまなメトリック (パフォーマンス測定) から選択してください。たとえば、サーバー上の平均 CPU 稼働率をモニターする場合には、グラフ上の任意の収集ポイントをクリックして、最も高い CPU 稼働率をもつ 20 個のジョブを表示した詳細図表を表示できます。それから、これらのジョブのいずれかを右クリックして、直接そのジョブを処理できます。

ジョブ・モニター

ジョブ名、ジョブ・ユーザー、ジョブ・タイプ、サブシステム、またはサーバー・タイプに基づいてジョブまたはジョブのリストをモニターします。ジョブのパフォーマンス、状況、またはエラー・メッセージをモニターするために、さまざまなメトリックから選択してください。直接ジョブを処理するには、「ジョブ・モニター (Job Monitor)」ウィンドウに表示されたリストから、ジョブを右クリックするだけで行えます。

メッセージ・モニター

アプリケーションが正常に完了するか、または業務上の必要に不可欠な特定のメッセージについてモニターするかどうかを調べます。「メッセージ・モニター」ウィンドウから、メッセージの詳細の表示、メッセージへの応答、メッセージの送信、およびメッセージの削除を行うことができます。

B2B 活動モニター
「iSeries の接続 (Connect for iSeries)」のようなアプリケーションが構成されている場合は、 B2B 活動モニターを使用して B2B トランザクションをモニターすることができます。活動状態のトランザクションのグラフを時間で表示し、さらに、しきい値でトリガーが出される時に自動的にコマンドを実行することができます。特定のトランザクションを検索して表示し、その特定のトランザクションの詳細ステップの棒グラフを表示できます。

ファイル・モニター
1 つ以上の選択したファイルで、指定されたテキスト・ストリング、指定されたサイズ、またはファイルに対する修正をモニターします。

モニターに関する詳細は、以下のトピックを参照してください。

モニターの概念
モニターは、リアルタイムにパフォーマンス・データを表示できます。さらに、その間もシステムのモニターを継続し、指定されたしきい値に達したなら選択されたコマンドを実行できます。モニターの働きと、モニターできる内容、および特定のパフォーマンス状態に応答する方法について学ぶことができます。

モニターの構成
iSeries モニター中でモニターを構成できます。このトピックを使用すると、モニターをセットアップする方法や、使用可能なオプションを最大限に活用できるように構成する方法を学べます。

シナリオ
このトピックにはシナリオが記載されており、さまざまなタイプのモニターの一部を使用してシステム・パフォーマンスの特定の側面を参照する方法が説明されています。

モニターの概念
システム・モニターは、収集サービスによって生成されて保守される収集オブジェクト中に保管されるデータを表示します。システム・モニターは、データが収集されるつど、最大 1 時間分表示します。 1 時間より長い期間のデータを表示するには、 グラフ・ヒストリーを使用する必要があります。モニター・プロパティー中でデータ収集の頻度を変更できます。この場合、収集サービス中の設定はオーバーライドされます。

モニターを使用して、システム・パフォーマンスの多種多様な要素を追跡したり調べたりすることができます。また多種多様なモニターを同時に実行することもできます。複数のモニターを同時に使用すると、システム・パフォーマンスの監視や管理を行う精巧なツールが得られます。たとえば、新しい対話式アプリケーションを実装する場合に、システム・モニターを使用してジョブのリソース使用率を優先順位付けし、ジョブ・モニターを使用して問題のあるジョブの監視や処理を行い、メッセージ・モニターを使用して指定されたメッセージがいずれかのシステムに表示された場合にアラートを出すことができます。

しきい値とアクションの設定

新しいモニターを作成する際には、システム・メトリックが指定されたしきい値レベルに達した場合や、イベントが起きた場合に実行させるアクションを指定できます。しきい値レベルに達したりイベントが起きたりすると、メッセージの送信やジョブ待ち行列の保持などの、 OS/400 コマンドをエンドポイント・システム上で実行するよう選択できます。さらに、イベント・ログを更新したり、 PC 上でアラーム音を鳴らすかモニターを立ち上げるかしてアラートを出したりするなどの、複数の事前定義されたアクションをモニターが実行するように選択することもできます。最後に、2 番目のしきい値を指定してモニターを自動的にリセットできます。このしきい値に達すると、モニターは通常の活動を再開します。

モニターの構成

システム・モニターは、高度な対話性を備えたツールで、エンドポイント・システムからのリアルタイム・パフォーマンス・データを収集および表示します。新しいモニターを作成する手順は、短時間でできる簡単なもので、**「新しいモニター」**ウィンドウから始めます。

1. iSeries ナビゲーターで、「マネージメント・セントラル」を展開し、**「モニター」**を選択して**「システム」**を右クリックし、次いで**「新しいモニター」**を選択します。
2. モニター名を指定します。**「新しいモニター」**-**「一般」**ページで、モニターの名前を指定します。モニターのリストからモニターを見付けられるように、簡単な説明も加えてください。
3. メトリックを選択します。**「新しいモニター」**-**「メトリック」**ページで、メトリックを選択します。任意の数のエンドポイント・システムまたはシステム・グループについて、任意の数のメトリックをモニターすることができます。
4. メトリック情報を表示して変更を加えます。**「新しいモニター」**-**「メトリック」**ページで、各メトリックのプロパティーを編集します。選択した各メトリックについて、収集間隔、グラフの最大値、および表示時間を編集できます。
5. しきい値コマンドを設定します。**「メトリック」**ページの**「しきい値」**タブで、しきい値を有効にし、そのトリガーしきい値またはリセットしきい値に達したときにエンドポイント・システムで実行するコマンドを指定します。
6. しきい値アクションを設定します。**「新しいモニター」**-**「アクション」**ページで、メトリックがトリガーしきい値またはリセットしきい値に達したときに実行するアクションを指定します。
7. システムとグループを選択します。**「新しいモニター」**-**「システムとグループ」**ページで、モニターを開始したいエンドポイント・システムとシステムのグループを選択します。

モニターを作成した後、モニター名を右クリックして**「開始」**をクリックすると、モニターを実行してモニター・グラフでの作業を開始することができます。

モニターのメトリック

効果的なシステム・パフォーマンスのモニターを行うためには、どのような視点からシステム・パフォーマンスをモニターするかを決めなければなりません。マネージメント・セントラルには、**メトリック**と呼ばれるさまざまなパフォーマンスの測定法がありますが、このメトリックを使用するとシステム・パフォーマンスのさまざまな側面に焦点を合わせることができます。

「新しいモニター」ウィンドウの**「メトリック」**ページでは、モニターに関する一般的な情報 (メトリックを含む) を表示および変更できます。このページを表示するには、**「モニター」**を選択して**「システム」**を右クリックし、次いで**「新しいモニター」**を選択します。フィールドに必要な情報を入力し、**「メトリック (Metrics)」**タブをクリックします。

モニターを構成する際に、モニターに組み込むリストから任意のメトリック、メトリックのグループ、またはすべてのメトリックを選択して使用することができます。モニターに使用できるメトリックのタイプには、次のものがあります。

メトリックのグループ:	メトリックの説明:
CPU 稼働率	処理装置時間のうち、システム上のジョブによって消費される時間のパーセンテージ。以下の中から、モニターで使用する CPU 稼働率メトリックのタイプを選択します。 • CPU 稼働率 (平均) • CPU 稼働率 (対話型ジョブ) • CPU 稼働率 (対話型のフィーチャー) • CPU 稼働率 (データベース能力) • CPU 稼働率 (2 次ワークロード) • CPU 稼働率基本 (平均) これらのメトリックに関する詳細と使用法については、 iSeries ナビゲーターの **「新しいモニター」**ウィンドウまたは**「モニター・プロパティー」**ウィンドウで、**「一般」**タブからオンライン・ヘルプを参照してください。
対話式応答時間 (平均および最大)	そのシステムでの対話型ジョブの応答時間。
トランザクション率 (平均)	システム上のジョブによって完了したトランザクションの 1 秒当たりの数。
トランザクション率 (対話式)	以下のタイプのジョブによってシステム上で完了されたトランザクションの 1 秒当たりの数。 • 対話式 • 複数要求端末 (MRT) • システム/36 環境対話式 • パススルー
バッチ論理データベース I/O	システム上のバッチ・ジョブによって現在実行されている論理データベース入出力 (I/O) 操作の平均数。
ディスク・アーム稼働率 (平均および最大)	データ収集中にシステム上で使用されているディスク・アーム・キャパシティーのパーセンテージ。
ディスク装置 (平均および最大)	データ収集中にシステム上でいっぱいになっているディスク・アーム記憶域のパーセンテージ。
ディスク IOP 稼働率 (平均および最大)	データ収集中のシステム上のディスク入出力処理装置 (IOP) が使用されている程度。
通信 IOP 稼働率 (最大および平均)	データ収集中のシステム上の通信入出力処理装置 (IOP) が使用されている程度。
通信回線稼働率 (平均および最大)	すべてのシステム通信回線から実際に送信および受信したデータの量。
LAN 稼働率 (最大および平均)	すべてのローカル・エリア・ネットワーク (LAN) 通信回線から実際に送信および受信したデータの量。
マシン・プール・ページ不在	システム上のマシン・プールで発生する 1 秒当たりのページ不在の数。

ユーザー・プール・ページ不在 (最大および平均)	システムのすべてのユーザー・プールの中で発生する 1 秒当たりのページ不在の数。

より詳細なヘルプを表示させるには、**「新しいモニター」**-**「メトリック」**ウィンドウで、**「ヘルプ」**ボタンをクリックします。マネージメント・セントラル・メトリックの使用に慣れたら、ご使用のコンピューティング環境に必要な情報に合わせてメトリックを選択できます。必要な情報に的を絞ってメトリックを選択した後、モニター用に選択した各メトリックごとにメトリックの詳細情報を表示および変更することができます。

シナリオ: iSeries ナビゲーターのモニター

iSeries ナビゲーターに組み込まれているモニターには、システム・パフォーマンスの調査や管理を行う強力なツールの集合が備えられています。 iSeries ナビゲーターに備えられているモニターのタイプの概要については、『iSeries ナビゲーターのモニター』を参照してください。

詳細な使用例とサンプルの構成については、以下のシナリオを参照してください。

シナリオ: システム・モニター
この例のシステム・モニターは、 CPU 稼働率が高すぎるために使用可能なリソースが増えるまで優先順位の低いジョブを一時的に保留する場合に、アラートを出します。

シナリオ: メッセージ・モニター
この例のメッセージ・モニターは、iSeries サーバー上で生じた、メッセージ待ち行列に関する照会メッセージを表示します。このモニターは、メッセージを検出すると即時にそのメッセージをオープンして表示します。

シナリオ: ジョブ・モニター
指定されたジョブの CPU 稼働率を追跡し、CPU 稼働率が高くなりすぎたときにジョブの所有者に警告する、ジョブ・モニターの例を考慮します。

シナリオ: システム・モニター:

状態

システム管理者は、ユーザーの要件や業務上の要件に基づく現在の要求を満たせるだけの資源が、確実に iSeries システム上にあるようにする必要があります。ご使用のシステムでは、CPU 稼働率が特に重要な関心事です。 CPU 稼働率が高すぎるために、使用可能なリソースが増えるまで優先順位の低いジョブを一時的に保留する場合に、システムがアラートを出すようにしたいと思っています。

そのためには、CPU 稼働率が 80% を超えたらメッセージを送信するように、システム・モニターをセットアップできます。さらに、CPU 稼働率が 60% に下がるまですべてのジョブを QBATCH ジョブ待ち行列中に保留し、 60% になったらジョブを保留解除して通常の操作を再開することもできます。

構成の例

システム・モニターをセットアップするには、追跡したいメトリックと、そのメトリックが指定のレベルに達した場合にモニターが行う処理を定義する必要があります。この目標を達成するようにシステム・モニターを定義するには、以下のステップを完了してください。

1. iSeries ナビゲーターで、**「マネージメント・セントラル」**>**「モニター」**を展開し、**「システム・モニター (System Monitor)」**を右クリックし、**「新しいモニター... (New Monitor...)」**を選択します。
2. **「一般」**ページで、このモニターの名前と説明を入力します。
3. **「メトリック」**タブで、以下の値を入力します。
   a. 「使用可能なメトリック」のリストから**「CPU 稼働率基本 (平均)」**を選択して、**「追加」**を選択します。「CPU 稼働率基本 (平均)」が「モニターするメトリック」の下にリストされるようになり、ウィンドウの下部にこのメトリックの設定が表示されます。
   b. **「収集間隔」**で、このデータを収集する頻度を指定します。この値は、収集サービスの設定をオーバーライドします。この例では、**「30 秒 (30 seconds)」**を例示します。
   c. このメトリックに関するモニターのグラフの縦軸の目盛りを変更するには、**「最大グラフ値」**を変更します。このメトリックに関するモニターのグラフの横軸の目盛りを変更するには、**「表示時間」**の値を変更します。
   d. メトリック設定の**「しきい値 1」**タブをクリックし、以下の値を入力して、 CPU 稼働率が 80% 以上の場合に照会メッセージを送信するようにします。
      1) **「しきい値を使用可能にする」**を選択します。
      2) しきい値トリガーの値として、**「>= 80」**(80 % 以上の稼働率) を指定します。
      3) **「期間」**に、間隔**「1」**を指定します。
      4) **「OS/400 コマンド」**に、以下の値を指定します。

         ```
         SNDMSG MSG('Warning,CPU...') TOUSR(*SYSOPR) MSGTYPE(*INQ)
         ```

      5) しきい値リセットの値として、**「< 60」**(60 % 未満の稼働率) を指定します。この場合、CPU 稼働率が 60% 未満に下がるとモニターがリセットされます。
   e. **「しきい値 2」**タブをクリックし、以下の値を入力して、 5 回の収集間隔の間 CPU 稼働率が 80% を超える状態が続いたら、すべてのジョブを QBATCH ジョブ待ち行列中に保留するようにします。
      1) **「しきい値を使用可能にする」**を選択します。
      2) しきい値トリガーの値として、**「>= 80」**(80 % 以上の稼働率) を指定します。
      3) **「期間」**に、間隔**「5」**を指定します。
      4) **「OS/400 コマンド」**に、以下の値を指定します。

         ```
         HLDJOBQ JOBQ(QBATCH)
         ```

      5) しきい値リセットの値として、**「< 60」**(60 % 未満の稼働率) を指定します。この場合、CPU 稼働率が 60% 未満に下がるとモニターがリセットされます。
      6) **「期間」**に、間隔**「5」**を指定します。
      7) **「OS/400 コマンド」**に、以下の値を指定します。

         ```
         RLSJOBQ JOBQ(QBATCH)
         ```

         このコマンドは、5 回の収集間隔の間 CPU 稼働率が 60% 未満の状態が続いたら、 QBATCH ジョブ待ち行列を保留解除します。
4. **「アクション」**タブをクリックして、**「トリガー」**と**「リセット」**の両方の列で、**「イベントのログ」**を選択します。このアクションを選択すると、しきい値が起動したりリセットされたりする際に、イベント・ログ中に項目が作成されます。
5. **「システムおよびグループ」**タブをクリックして、モニターしたいシステムとグループを指定します。
6. **「OK」**をクリックして、モニターを保管します。
7. システム・モニターのリストから、新しいモニターを右クリックして、**「開始」**を選択します。

結果

新しいモニターは、CPU 稼働率を表示し、指定された収集間隔に従って 30 秒ごとに新しいデータ・ポイントを追加します。 CPU 稼働率が 80% に達すると、PC がオフになっている場合も含めて、必ずモニターは指定されたしきい値アクションを実行します。

注: このモニターは、CPU 稼働率のみを追跡します。しかしながら、同一のモニターに使用可能なメトリックをいくつでも組み込むことができ、個々のメトリックに独自のしきい値とアクションを指定できます。さらに、複数のシステム・モニターを同時に実行することもできます。

シナリオ: ジョブ・モニター:

状態

現在 iSeries サーバー上で新しいアプリケーションを実行しており、一部の新しい対話型ジョブが許容量を超えるリソースを使用していることに着目しています。ジョブが使用する CPU キャパシティーが多すぎる場合に、常にその問題のジョブの所有者に通知するようにしたいと思っています。

新しいアプリケーション中のジョブを監視し、ジョブが使用する CPU キャパシティーが 30% を超えたらメッセージを送信するように、ジョブ・モニターをセットアップできます。

構成の例

ジョブ・モニターをセットアップするには、監視対象のジョブ、監視対象のジョブ属性、および指定したジョブ属性が検出された場合にモニターが行う処理を定義する必要があります。この目標を達成するようにジョブ・モニターをセットアップするには、以下のステップを完了してください。

1. iSeries ナビゲーターで、**「マネージメント・セントラル」**>**「モニター」**を展開し、**「ジョブ・モニター (Job monitor)」**を右クリックし、**「新しいモニター... (New Monitor...)」**を選択します。
2. **「一般」**ページで、以下の値を入力します。
   a. このモニターの名前と説明を指定します。
   b. **「モニターするジョブ (Jobs to monitor)」**タブで、以下の値を入力します。
      1) **「ジョブ名 (Job name)」**で、監視したいジョブの名前 (MKWIDGET など) を指定します。
      2) **「サブシステム」**で、QINTER を指定します。
      3) **「追加」**をクリックします。
3. **「メトリック」**タブで、以下の情報を入力します。
   a. **「使用可能なメトリック」**リストで、**「合計数値 (Summary Numeric Values)」**を展開し、**「CPU 稼働率のパーセンテージ (CPU Percent Utilization)」**を選択して、**「追加」**をクリックします。
   b. メトリック設定の**「しきい値 1」**タブで、以下の値を入力します。
      1) **「トリガーを使用可能にする」**を選択します。
      2) しきい値トリガーの値として、**「>= 30」**(30 % 以上の稼働率) を指定します。
      3) **「期間」**に、間隔**「1」**を指定します。
      4) **「OS/400 トリガー・コマンド (OS/400 trigger command)」**に、以下の値を指定します。

         ```
         SNDMSG MSG('Your job is exceeding 30% CPU capacity') TOUSR(&OWNER)
         ```

      5) **「リセットを使用可能にする (Enable reset)」**をクリックします。
      6) しきい値リセットの値として、**「< 20」**(20 % 未満の稼働率) を指定します。

4. **「収集間隔」**タブをクリックして、**「15 秒 (15 seconds)」**を選択します。この値は、収集サービスの設定をオーバーライドします。
5. **「アクション」**タブをクリックして、**「トリガー」**と**「リセット」**の両方の列で、**「イベントのログ」**を選択します。
6. **「サーバーおよびグループ (Servers and groups)」**タブをクリックして、このジョブをモニターする対象にしたいサーバーとグループを選択します。
7. **「OK」**をクリックして、新しいモニターを保管します。
8. ジョブ・モニターのリストから、新しいモニターを右クリックして、**「開始」**を選択します。

結果

新しいモニターは、15 分ごとに QINTER サブシステムをチェックし、ジョブ MKWIDGET の CPU 稼働率が 30 % を超えると、このジョブの所有者にメッセージを送信します。このジョブが使用する CPU のキャパシティーが 20% 未満の場合は、このモニターはリセットします。

シナリオ: メッセージ・モニター:

状態

貴社で複数の iSeries サーバーを実行しており、個々のシステムのメッセージ待ち行列をチェックするのに時間がかかります。システム管理者は、システム全体のどこでも照会メッセージが生成されたら、そのことに気付く必要があります。

いずれかの iSeries システムで生じた、メッセージ待ち行列に関する照会メッセージを表示するように、メッセージ・モニターをセットアップできます。このモニターは、メッセージを検出すると即時にそのメッセージをオープンして表示します。

構成の例

メッセージ・モニターをセットアップするには、監視したいメッセージのタイプと、それらのメッセージが生成された場合にモニターが行う処理を定義する必要があります。この目標を達成するようにメッセージ・モニターをセットアップするには、以下のステップを完了してください。

1. iSeries ナビゲーターで、**「マネージメント・セントラル」**>**「モニター」**を展開し、**「メッセージ・モニター (Message monitor)」**を右クリックし、**「新しいモニター... (New Monitor...)」**を選択します。
2. **「一般」**ページで、このモニターの名前と説明を入力します。
3. **「メッセージ」**タブで、以下の値を入力します。
   a. **「モニターするメッセージ待ち行列 (Message queue to monitor)」**で、**「QSYSOPR」**を指定します。
   b. **「メッセージ・セット 1 (Message set 1)」**タブ上で、**「タイプ」**で**「照会」**を選択して、**「追加」**をクリックします。
   c. **「このメッセージ数で起動 (Trigger at the following message count)」**を選択して、メッセージ数**「1」**を指定します。
4. **「収集間隔」**タブをクリックして、**「15 秒 (15 seconds)」**を選択します。
5. **「アクション」**タブをクリックして、**「モニターを開く」**を選択します。

6. **「システムおよびグループ」**タブをクリックして、照会メッセージをモニターしたいシステムとグループを指定します。
7. **「OK」**をクリックして、新しいモニターを保管します。
8. メッセージ・モニターのリストから、新しいモニターを右クリックして、**「開始」**を選択します。

結果

新しいメッセージ・モニターは、モニターされているいずれかの iSeries サーバー上の QSYSOPR に送信された照会メッセージを表示します。

注: このモニターは、 QSYSOPR に送信される照会メッセージだけに応答します。しかしながら、1 つのモニターに 2 種類のメッセージの集合を組み込んだり、複数のメッセージ・モニターを同時に実行したりできます。また、指定されたメッセージが受信された時点で、メッセージ・モニターが OS/400 コマンドを実行することもできます。

グラフ・ヒストリー

グラフ・ヒストリーには、 収集サービスを使用して数日、数週間、数か月、または数年にわたって収集されたパフォーマンス・データのグラフィカル・ビューが備えられています。システム・モニターを実行してパフォーマンス・データを表示する必要はありません。収集サービスを使用してデータを収集する限り、「グラフ・ヒストリー」ウィンドウを表示することができます。

- **グラフ・ヒストリーの概念**
  グラフ・ヒストリーで使用できるヒストリー・データの量は、収集サービス中の収集保存期間の値と、PM/400 が使用可能かどうかによって、大きく変わります。パフォーマンス・データのレコードの管理や表示を行うのに使用できるオプションの説明については、このトピックを参照してください。
- **グラフ・ヒストリーの使用**
  グラフ・ヒストリーには、iSeries ナビゲーターからアクセスできます。ステップ形式の指示については、このトピックを参照してください。

システム・パフォーマンスのモニターについて詳しくは、『パフォーマンスの追跡』のトピックを参照してください。

グラフ・ヒストリーの概念

「グラフ・ヒストリー」には、 収集サービスで作成された収集オブジェクトに入っているデータが表示されます。したがって、使用可能なデータのタイプと量は、収集サービスの構成によって異なります。

グラフ化が可能なデータの量は、「収集サービス」プロパティーから選択した設定値 (特に収集保存期間) によって決まります。複数のシステムにまたがって PM/400 を活動化するには iSeries ナビゲーターを使用します。 PM/400 をアクティブにすると、グラフ・ヒストリー機能を使用して、数日前、数週間前、または数か月前に収集されたデータを表示することができます。これはリアルタイムのモニター機能を超えて、要約または詳細データにアクセスすることができます。 PM/400 が使用可能になっていないと、グラフ・データ・フィールドは 1 ～ 7 日をサポートします。 PM/400 が使用可能になっていると、システム上での管理収集オブジェクトの保存期間を定義します。

- **詳細データ**
  管理収集オブジェクトが削除される前に、ファイル・システム内にそれらが保存される時間の長さ。特定の時間間隔を時間数または日数で選択するか、あるいは **「永続」**を選択することができます。**「永続」**を選択する場合、管理収集オブジェクトは自動的に削除されません。

- **グラフ・データ**
  「グラフ・ヒストリー」ウィンドウに示される詳細およびプロパティー・データのデータが、表示されてから削除されるまでのシステムに留まっている時間の長さです。 PM/400 を開始しない場合、1 ～ 7 日を指定することができます。 PM/400 を開始する場合、1 ～ 30 日を指定することができます。デフォルト値は 1 時間です。
- **要約データ**
  グラフのデータ収集ポイントが削除される前に、「グラフ・ヒストリー (Graph History)」ウィンドウ内にそれらが表示されるか、またはシステムに保存される時間の長さ。詳細データまたはプロパティー・データは使用できません。 PM/400 を開始して、要約データ・フィールドを使用可能にする必要があります。デフォルトは 1 か月です。

グラフ・ヒストリーの使用

グラフ・ヒストリーは iSeries ナビゲーターに組み込まれています。 収集サービスでモニターしているデータのグラフ・ヒストリーを表示するには、以下のステップを行ってください。

1. 単一システムまたはシステム・グループでの収集サービスの開始の方法の詳細は、 iSeries ナビゲーターのオンライン・ヘルプを参照してください。
2. 必要な場合、**「収集サービスの開始 - 一般」**ページから、**「PM/400 を開始」**を選択します。
3. 収集保存期間の残りの値に変更を加えます。
4. **「OK」**をクリックします。
5. システム・モニターまたは収集サービス・オブジェクトのどちらかを右マウス・ボタンでクリックして**「グラフ・ヒストリー」**を選択することによって、グラフ・ヒストリーを表示することができます。
6. グラフィカルに表示するには、**「最新表示」**をクリックしてください。

グラフ・ヒストリーの立ち上げが完了したなら、グラフ化された一連の収集ポイントを示したウィンドウが表示されます。グラフ線上の収集ポイントは、使用できるデータの 3 つのレベルと対応する 3 つの異なるグラフで表示されます。

- 四角の収集ポイントは、詳細情報とプロパティー情報の両方がデータ内にあることを意味します。
- 三角形の収集ポイントは、詳細情報を含む要約データを表しています。
- 円形の収集ポイントは、詳細情報またはプロパティー情報を含まないデータを表しています。

Performance Management/400

Performance Management/400 (PM/400) は自動化および自己管理されているため、簡単に使用することができます。 PM/400 は、収集サービスを自動的に起動して、所有権の付いていないパフォーマンス・データおよびキャパシティー・データをサーバーから収集してから、そのデータを IBM に送信します。すべてのコレクション・サイトはネットワーク保護されており、転送時間は、ユーザーによって完全に制御されます。 IBM にデータを送信すると、すべての傾向データを自分で保管する必要がなくなります。 IBM がユーザーのためにデータを保管し、サーバーの拡張およびパフォーマンスを示す一連の報告書およびグラフを提供します。従来のブラウザーを使用して、電子的に報告書にアクセスすることができます。そのような報告書を参考にして、かぎとなるパフォーマンス指標の継続的分析を通して、システム資源を計画して管理することができます。

IBM Operational Support Services for PM/400e には一連の報告書、グラフ、およびプロファイルが組み込まれています。これらは、現行アプリケーションおよびハードウェア・パフォーマンスを最大化する (パフォーマンス傾向分析を使用して) のに役立ちます。またこの製品を使用すると、 CPU またはディスクなどのハードウェアに必要なアップグレードのタイミングと業務傾向がどのように関連しあっているかが分かりやすくなります (キャパシティー・プランニングを使用して) 。キャパシティー・プランニング情報はシステ

ム稼働率資源およびスループット・データの傾向から得ることができ、ご使用のサーバーについての早期警戒システムと見なすことができます。 PM/400e は、システムの「正常性」を知らせる仮想資源であると考えてください。

PM/400 は、中央処理装置 (CPU) の 1 % 未満しか使用しません。 PM/400 は、約 58 MB のディスク・スペースを使用します。これはハードウェア・モデルおよび収集間隔のサイズによって異なります。

PM/400 の概念
PM/400 で利用できる機能と長所についてと、実装時の重要な考慮事項について考察してみてください。

PM/400 の構成
PM/400 の使用を開始するには、それを活動化し、データの送信と報告書の受信のための伝送方式をセットアップしてから、最後にデータの収集とストレージをカスタマイズする必要があります。

PM/400 の管理
ネットワークのセットアップが完了したので、 PM/400 でのさまざまなタスクを実行することができます。

PM/400 報告書
PM/400 を使用して収集サービス・データを IBM へ直接送信するように iSeries サーバーを構成することができます。その後 IBM でいくつかの報告書が生成されます。これは、 Web 上で表示できますが、直接ユーザーに返送することも可能です。 PM/400 を活動化して自動的にレポートを生成するようにすれば、時間と資源の節約になるだけでなく、将来の巨大化時点でのニーズを見込んで事前に計画をたてることができます。

PM/400 の概念

PM/400 は収集サービスを使用して、所有権の付いていないパフォーマンス・データおよびキャパシティー・データをサーバーから収集してから、そのデータを IBM に送信します。そのような情報には、 CPU 稼働率とディスク・キャパシティー、応答時間、スループット、アプリケーションとユーザーの使用量などがあります。 IBM にデータを送信すると、すべての傾向データを自分で保管する必要がなくなります。 IBM がユーザーのためにデータを保管し、サーバーの拡大とパフォーマンスを示す一連の報告書とグラフを提供します。従来のブラウザーを使用して、電子的に報告書にアクセスすることができます。

PM/400 の利点
PM/400 を使用すると、システム資源の管理とキャパシティー・プランニングがかなり簡素化されます。 PM/400 を使用する具体的な方法を考察してください。

PM/400 に備わったオプション
PM/400 には、広範囲にわたるオプションが用意されています。以下の情報を使用して、各自のニーズに最も合ったサービスの組み合わせを決めてください。

データ収集に関する考慮事項
PM/400 では、パフォーマンス・データの収集には収集サービスを使用します。 PM/400 と収集サービスがどのように連携して、必要なデータを提供するかを確認してください。

PM/400 の利点: PM/400 サービスを使用すると、次のような利点が得られます。

- **不測の事態が起きないようにするのに役立ちます。**
  思いがけない失策を免れます。ユーザーがシステムの拡張およびパフォーマンスの管理を制御しますが、これは、ユーザーがシステムを管理するのであって、システムがユーザーを管理するのではないことを意味します。

- **時間を節約します。**
  パフォーマンス・データの収集および報告を自動的に行うことによって、それらの作業にかかる労力や費用を節約します。これにより、ユーザー資源をシステムおよびアプリケーションの管理に集中できるという利点があります。
- **最大限の効率を得られるように前もって計画することができます。**
  システムを最大効率で実行し続けるための財政的な要件を前もって計画することができます。
- **情報を理解することが容易になります。**
  情報を理解すれば、上司から「なぜアップグレードする必要があるのか」と質問されたときに、容易に返答することができます。
- **将来を予測することができます。**
  実際の傾向情報に基づいてデータ処理の拡張を予測することができます。
- **システム問題を識別することができます。**
  PM/400 データを使用してパフォーマンス障害を識別することができます。
- **次のアップグレードのサイズを見積もるときの参考になります。**

  PM/400 データを Workload Estimator for iSeries  にアップロードすれば、次のアップグレードのサイズを設定することができます。

PM/400 を使用する前に何をすればよいかの詳細については、 Performance Management/400 を参照してください。

PM/400e 製品の操作サポート・サービス: グラフおよび報告書は電子的または印刷形式のいずれかで受け取ることができます。電子グラフは毎月受け取ることができます。印刷グラフは毎月または 3 か月ごとに受け取ります。 PM/400e サービス料金は、パフォーマンス情報を受け取る回数とその形式 (電子的または印刷形式) の選択によって変わります。これらの報告書オプションのいくつかは無料ですが、有料のものもあります。それぞれの国のマーケティングおよびサービスに関係する組織は、使用可能なサポートの詳細を提供することができます。無償および有償のオプションの詳細は、 PM/400e の Web サイト  を参照してください。

PM/400 を使用する前に何をすればよいかの詳細については、 Performance Management/400 を参照してください。

PM/400 のデータ収集に関する考慮事項: システムの稼働率、ワークロード、およびパフォーマンス測定の正確な傾向を設定するための最も重要な要件は、整合性です。理想的に言って、パフォーマンス・データは 1 日に 24 時間収集される必要があります。 PM/400 と収集サービスとの間の関係のため、 PM/400 の使用時に起こり得る事柄を理解している必要があります。

ここで、PM/400 の使用時に収集の定義を助けるためのいくつかの指針があります。

- **QMPGDATA ライブラリーを選択して、データを保管します。**
  PM/400 が活動状態にある場合、**「コレクションを保管する場所」**フィールドには、デフォルト値 /QSYS.LIB/QMPGDATA.LIB が使用されます。 QMPGDATA が他の値に変更されると、PM/400 はすぐにコレクションを循環させ、その値を QMPGDATA に戻します。異なるライブラリーにデータを収集したい場合には、PM/400 がデータを検索する場所を変更します。コマンド行で **GO PM400** と入力し、オプション 3 (カスタマイズの処理) を選択して、ライブラリー名を変更します。
- **収集サービスを使用して、継続的にデータを収集します。**
  PM/400 は、収集サービスを使用して 1 日に 24 時間データを収集することにより、この要件を満たします。 PM/400 は、15 分間隔でパフォーマンス・データを収集します。 PM/400 は、デフォルトでは 15 分間隔を使用しますが、設定されている間隔は変更しません。推奨される間隔は 15 分間隔です。

- **「標準 + プロトコル」プロファイルを選択します。**
  標準 + プロトコルは、収集プロファイルのデフォルト値です。収集プロファイルは、どのデータが収集されるかを示します。「標準 + プロトコル」プロファイル内のデータ・カテゴリーは、パフォーマンス・モニターの開始 (STRPFRMON) コマンド上の DATA パラメーターの *ALL 値に対応しています。この値が他の値に変更されると、PM/400 はすぐに値を元に戻します。これは、「カスタム」を選択し、すべてのカテゴリーを含めている場合も同じです。変更は直ちに有効になります。コレクションは、(他の理由で必要とされない限り) 循環しません。このアクションは、PM/400 報告書に関する十分な情報を収集するために行います。
- **PM/400 が活動状態にあるときは、収集のパラメーターに一時変更を加えないようにします。**
  たとえば、PM/400 を活動化したときは、**「収集時にデータベース・ファイルを作成」**フィールドにデフォルト値のチェックが付いています。この値が変更されると、PM/400 はすぐにデフォルト値を元に戻します。変更は直ちに有効になります。コレクションは、(他の理由で必要とされない限り) 循環しません。
- **収集サービスを終了します。**
  オペレーション・ナビゲーターからいつでも収集サービスを終了することができます。収集サービスを終了する場合、 PM/400 が実行しているときに以下の考慮事項が適用されます。
  - PM/400 スケジューラーは、次の時間の始めに収集サービスを開始します。
  - 収集されるデータがほとんどない日は、傾向の計算に含まれません。したがって、収集サービスを頻繁に中断しないでください。

収集サービスを開始しない場合には、 PM/400 を一時的にオフにすることができます。

PM/400 の構成

PM/400 では、収集サービスを通してパフォーマンス・データの収集が自動化されています。どのライブラリーにそのデータを入れるかを指定できますが、基本補助記憶域プール (ASP) に置かれているライブラリーであることが前提になります。そのライブラリーを、独立した補助記憶域プールに移動してはなりません。独立した補助記憶域プールはオフに変更される可能性があり、もしオフに変更されると、PM/400 の収集プロセスは停止するからです。ライブラリーが存在しないと、PM/400 が起動時に作成します。

PM/400 の使用を開始するには、以下のタスクを実行する必要があります。

PM/400 の活動化
PM/400 は OS/400 の付属製品ですが、収集機能を利用するには活動化する必要があります。

使用する送信メソッドの判別
データの送信方法を決定します。マネージメント・セントラルを使用してデータを収集してから、Electronic Service Agent (エクストリーム・サポート) を使用してデータを送信することができますが、 PM/400 を使用してデータを収集して SNA プロトコルを介してデータを送信してもかまいません。

PM/400 のカスタマイズ
ネットワークのセットアップが完了したなら、ユーザーの必要を満たすために PM/400 をカスタマイズする必要があります。

Performance Management/400 の活動化: データ収集機能を利用するには、PM/400 を開始しなければなりません。以下の方法のいずれかを使用して、PM/400 を開始することができます。

iSeries ナビゲーターを使用します。
複数のシステムにまたがって PM/400 を活動化するには iSeries ナビゲーターを使用します。 PM/400 を

アクティブにすると、グラフ・ヒストリー機能を使用して、数日前、数週間前、または数か月前に収集されたデータを表示することができます。リアルタイム・モニター機能を超えています。要約データまたは詳細データにアクセスすることができます。 PM/400 が使用可能になっていないと、グラフ・データ・フィールドは 1 ～ 7 日をサポートします。 PM/400 が使用可能な場合は、データ保存の時間の長さを選択します。

iSeries ナビゲーターから PM/400 を開始するには、以下のステップを実行します。

1. PM/400 を開始したいシステムを iSeries ナビゲーターで拡張表示します。
2. **「構成およびサービス」**を展開します。
3. **「収集サービス」**を右クリックします。
4. **PM/400** を選択します。
5. **「開始」**を選択します。
6. PM/400 を開始したいシステムを選択します。
7. **「OK」**をクリックします。

QSYSOPR メッセージ待ち行列内のメッセージ CPAB02A への応答

QSYSWRK サブシステムが開始するとき、このメッセージによって PM/400 を活動化するかどうかを尋ねられます。

1. 文字ベースのインターフェースから、 QSYSOPR 内のメッセージ “Do you want to activate PM/400? (I G C)” に G と応答します。 QSYSOPR メッセージ待ち行列は PM/400 を活動化するというメッセージを受信します。
2. 連絡先情報を更新します。 **GO PM400** コマンドを発行して、オプション 1 を指定します。

PM/400 の構成 (CFGPM400) コマンドの発行

文字ベースのインターフェースから、PM/400 の構成 (CFGPM400) コマンドを発行することができます。

セットアップ・プロセス内の次のステップ『IBM にデータを送信するのにどの送信方式を使用するかの決定』に進むことができます。

iSeries のパフォーマンスの概要の詳細は、『パフォーマンス』を参照してください。

どの PM/400 送信方式を使用するかの決定: V5R1 以降では、PM/400 送信プロセスでは、セントラル・システムとエンドポイント・システムをセットアップするためにマネージメント・セントラルを使用して実行するネットワーク構成の利点が生かされています。ただし、引き続き文字ベースのインターフェースを使用して PM/400 を構成することもできます。使用したい送信方式を選択します。

- エクストリーム・サポートでの Electronic Service Agent によるデータの送信
  この送信方式を選択する場合、マネージメント・セントラルによってデータを収集するように PM/400 を構成する必要があります。サーバーに V4R5 以降のオペレーティング・システムがインストールされている場合 (ユニバーサル・コネクションの修正も適用しなければなりません)、 PM/400 に対してこの構成を実行します。エクストリーム・サポートを使用する場合は、このメソッドを選択します。
- SNA プロトコルでのデータの送信
  この送信方式を選択する場合、文字ベースのインターフェースを使用して PM/400 を構成する必要があります。 PM/400 は SNA を使用してデータを収集し、それを送信します。サーバーに OS/400 V4R5 またはそれ以前がインストールされている場合、 PM/400 に対してこの構成を実行します。

使用したい送信方式の実装が完了したなら、次に PM/400 の管理を実行するためのその他のタスクを行うことができます。

エクストリーム・サポートでの Service Agent によるデータの送信 (ユニバーサル・コネクション):
PM/400 は、収集サービスを使用して、所有権の付いていないパフォーマンス・データおよびキャパシティー・データをサーバーから収集します。このデータを収集した後、エクストリーム・サポートで Service Agent を使用して、 IBM にデータを送信することができます。

これらの機能を利用するには、サーバー上に V5R1 または V5R2 か、あるいはユニバーサル・コネクションの修正を適用された V4R5 がインストールされていなければなりません。以下に、PM/400 を構成するためのステップを示します。

1. PM/400 を活動化します。
   データ収集機能を利用するには、PM/400 を開始しなければなりません。
2. マネージメント・セントラル・ネットワークをセットアップします。
   どのサーバーがセントラル・システムであり、どのサーバーがエンドポイント・システムであるかを定義します。 IBM にデータを送信する前に、このネットワーク階層を使用して、エンドポイント・システムから中央設置場所にデータを送信することができます。
3. IBM に接続し、ユニバーサル・コネクションを使用してデータを送信します。
   これは、マネージメント・セントラルが IBM に PM/400 データを送信するために使用する接続です。以前のリリースでは、SNA 上で実行したエレクトロニック支援 (ECS) 接続を使用していました。ユニバーサル・コネクションを使用する際には、TCP/IP 上でデータを送信することができます。
4. PM/400 パフォーマンス・データを収集します。
   マネージメント・セントラル・インベントリー機能を使用して、データを収集します。
5. IBM へデータを送信します。
   Electronic Service Agent (マネージメント・セントラル階層内のエクストリーム・サポートの下で使用可能) を使用して、 IBM にデータを送信します。 Electronic Service Agent はユニバーサル・コネクションを使用します。

また、SNA プロトコルでデータを送信することもできます。

PM/400 を構成し終わったら、PM/400 の管理を実行するためのその他のタスクを行う準備ができました。

PM/400 パフォーマンス・データの収集: 以下のタスクを完了した場合、マネージメント・セントラルを使用して、 PM/400 パフォーマンス・データを収集することができます。

1. PM/400 の活動化
2. ユニバーサル・コネクションの構成
3. マネージメント・セントラル・ネットワークのセットアップ
4. Electronic Service Agent がシステム上にインストールされていることの確認

エンドポイント・システムまたはシステム・グループ上に PM/400 パフォーマンス・データを収集するには、以下のステップを実行します。

1. iSeries ナビゲーターで、**「マネージメント・セントラル」**を展開します。
2. **「エンドポイント・システム」**または**「システム・グループ」**を展開します。
3. エンドポイント・システムまたはシステム・グループを右クリックして、**「インベントリー」**を選択します。
4. **収集**を選択します。
5. 収集する 1 つまたは複数のインベントリーを選択します。この場合、**「PM/400 パフォーマンス・データ (PM/400 performance data)」**を選択します。

6. 収集の完了時にセントラル・システム上でアクションを実行する場合には、リストからアクションを選択します。
7. すぐにデータの収集を開始する場合は**「OK」**をクリックします。あるいは**「スケジュール」**をクリックして、データの収集をいつ実行するかを指定します。

サーバーを構成し終わったら、PM/400 の管理を実行するためのその他のタスクを行う準備ができました。

SNA プロトコルでのデータの送信: エクストリーム・サポートでの Electronic Service Agent によるデータの送信を利用しないことにした場合でも、やはり文字ベースのインターフェースを使用してデータを送信することができます。 PM/400 は、サーバーの構成および使用に関する一連の質問を尋ねてきます。「PM/400 の構成 (Configure PM/400)」画面では、サーバーが PM/400 パフォーマンス・データを送受信する方法に関する質問を尋ねられます。プロセスの最初の部分には、ネットワークのセットアップが含まれています。 2 番目の部分は、データの送信方法に関するものです。文字ベースのインターフェースを使用する際には、直接ダイヤル回線を使用してデータを送信することができます。

SNA を使用してデータを送信するには、以下のタスクを行います。

1. PM/400 を活動化します。
   データ収集機能を利用するには、PM/400 を開始しなければなりません。
2. 使用するネットワーク構成を選択します。

   データの送信に使用するネットワーク構成を決定します。 直接ダイヤル回線、既存のインターネット・サービス・プロバイダー (ISP)、または仮想私設ネットワーク (VPN) を使用することによって、 IBM に接続する方法を選択します。 ISP または VPN を使用する場合には、 ユニバーサル・コネクションを構成しなければなりません。

   直接ダイヤル回線を使用して IBM にデータを報告することにした場合、ネットワークを構成する方法としていくつかの選択項目があります。ご使用のネットワークに適切な構成を選択し、「PM/400 の構成 (Configure PM/400)」画面からその特定の構成について概説しているステップを実行します。

   - 単一サーバーとして、 IBM に直接そのデータを送信します。
   - ホスト・サーバーとして、ご使用のサーバーが他のサーバー (リモート・サーバー) からパフォーマンス・データを受信してから、 IBM にデータを転送します。ホスト・サーバーは、その他のサーバーより前のリリース・レベルであってはなりません。つまり、ホスト・サーバーは、その他のサーバーと同じかまたはそれ以降のリリース・レベルでなければなりません。
   - リモート・サーバーとして、パフォーマンス・データをホスト・サーバーに送信することができます。「PM/400 の構成 (Configure PM/400)」画面で、リモート・サーバーが必要であることを確認し、 PM/400 メニューのオプション 5 (「リモート iSeries システムでの処理 (Work with remote iSeries systems)」) を使用してリモート・サーバーを定義します。
3. リモート・サーバーでの処理
   ご使用のネットワークをホスト・サーバー用にセットアップすることにした場合、ホスト・サーバーにそれらのデータを送信するサーバーを識別する必要があります。単一サーバーまたはリモート・サーバーを使用する場合には、このステップを無視することができます。
4. PM/400 のカスタマイズ
   ネットワークを構成した後、PM/400 ソフトウェアの操作用のグローバル・パラメーターを設定する必要があります。直接ダイヤル回線で IBM に接続する場合には、PM/400 データ電話番号を定義する必要があります。

サーバーを構成し終わったら、PM/400 の管理を実行するためのその他のタスクを行う準備ができました。

単一サーバー用の PM/400 ネットワーク: 単一サーバーは、IBM に直接そのデータを送信します。以下に示すステップは、PM/400 がデータを収集し、SNA 上でデータを送信する場合にのみ、単一サーバー用の PM/400 を構成するために実行しなければなりません。サーバーの「PM/400 の構成 (CFGPM400) (Configure PM/400 (CFGPM400))」画面から:

1. コマンド行に **CFGPM400** と入力します。
2. **「IBM へのパフォーマンス・データの送信 (Send performance data to IBM)」**フィールドに *YES を指定します。
3. **「パフォーマンス・データの受信 (Receive performance data)」**フィールドに *NO を指定します。
4. QMPGDATA のデフォルトのライブラリーを受け入れます。
5. 「IBM へのパフォーマンス・データの送信 (Send performance data to IBM)」フィールドに *YES を指定した場合、該当する通信オブジェクトが存在するかどうかを示す追加情報が表示されます。そのようなオブジェクトが存在しないと、伝送用の通信オブジェクトが PM/400 によって作成されます。その他の表示に対しては、適宜応答してください。
6. 「連絡先情報の処理 (Work with Contact Information)」画面に会社の連絡先情報を入力します。

単一サーバーのセットアップは必要ないと判断した場合、別の SNA 構成オプションを選択することができます。

サーバーを構成し終わったら、PM/400 の管理を実行するためのその他のタスクを行う準備ができました。

ホスト・サーバー用の PM/400 ネットワーク: ホスト・サーバーは、その他のサーバーからパフォーマンス・データを受信してから、IBM にデータを転送します。以下に示すステップは、PM/400 がデータを収集し、SNA 上でデータを送信する場合にのみ、ホスト・サーバー用に PM/400 を構成するために実行しなければなりません。

1. ホスト・サーバー上の「PM/400 の構成 (Configure PM/400)」画面から
   - コマンド行に **CFGPM400** と入力します。
   - **「IBM へのパフォーマンス・データの送信 (Send performance data to IBM)」**フィールドに *YES を指定します。
   - **「パフォーマンス・データの受信」**フィールドに *YES を指定します。
   - QMPGDATA のデフォルトのライブラリーを受け入れます。
2. ホスト・サーバー上の「リモート iSeries システムでの処理 (Work with Remote iSeries Systems)」画面から、次のようにします。
   - F6 (作成) を押して、どのサーバーがホスト・サーバーにそれらのデータを送信するかを識別します。
   - フィールドを完成させて、Enter を押します。

注: 以下の状態は、 PM/400 がデータを収集し、SNA 上でデータを送信する場合にのみ生じます。ネットワーク・システムを利用している場合、 iSeries ナビゲーターにおいてユニバーサル・コネクションとマネージメント・セントラルを使用して、それらのシステム用のデータの収集と伝送を行うことをお勧めします。

PM/400 は、データがリモート・サーバーから受信された後、 1 次サーバーから IBM へのデータの送信を自動的にスケジュールします。自動スケジューリングがご使用の作業管理体系に合わない場合には、 1 次サーバーからデータの送信を手動でスケジュールすることができます。

ここに、データの送信をスケジュールするときに覚えておく必要のあるヒントがあります。週全体で、1 次サーバーへのデータの送信を均等にスケジュールします。このアクションにより、1 次サーバーのパフォー

マンスの影響を最小限にします。たとえば、12 のサーバーから成るネットワークで、4 つのシステムの 3 つのグループを持っているとします。月曜日、水曜日、および金曜日にデータを送信するようにそれぞれのグループをスケジュールすることができます。これにより、1 次サーバーに送信されるデータの量は均等に分散されます。

ホスト・サーバーのセットアップは必要ないと判断した場合、別の SNA 構成オプションを選択することができます。

サーバーを構成し終わったら、PM/400 の管理を実行するためのその他のタスクを行う準備ができました。

リモート・サーバー用の PM/400 ネットワーク: リモート・サーバーは、ホスト・サーバーにそのパフォーマンス・データを送信します。以下に示すステップは、PM/400 がデータを収集し、SNA 上でデータを送信する場合にのみ、リモート・サーバー用の PM/400 を構成するために実行しなければなりません。リモート・サーバー上の「PM/400 の構成 (Configure PM/400)」画面 (CFGPM400) で、次のようなステップを行います。

1. コマンド行に **CFGPM400** と入力します。
2. **「IBM へのパフォーマンス・データの送信 (Send performance data to IBM)」**フィールドに *NO を指定します。
3. **「パフォーマンス・データの受信 (Receive performance data)」**フィールドに *NO を指定します。
4. QMPGDATA のデフォルトのライブラリーを受け入れます。

注: ネットワーク・システムを利用している場合、 iSeries ナビゲーターのインベントリー機能を使用してデータを収集して、ユニバーサル・コネクションを介してそれらのシステム用のデータを伝送することをお勧めします。

リモート・サーバーのセットアップは必要ないと判断した場合、別の SNA 構成オプションを選択することができます。

サーバーを構成し終わったら、PM/400 の管理を実行するためのその他のタスクを行う準備ができました。

リモート・サーバーでの処理: サイトによっては、処理に必要なパフォーマンス・データはネットワーク内のホスト・サーバーから IBM に送信されます。ホスト・サーバー・ネットワークを使用すると、ネットワーク内の他のサーバーがこのホスト・サーバーにパフォーマンス・データを送信して IBM に送信します。ホスト・サーバーを使用するようにネットワークをセットアップするには、その他のリモート・サーバーを識別し、それらのデータ送信用のスケジュールを設定しなければなりません。「リモート iSeries システムでの処理 (Work with Remote iSeries Systems)」画面では、その他のサーバーを定義することができます。

注:

1. リモート・サーバーまたは単一サーバーとしてネットワークをセットアップしている場合、この画面を使用する必要はありません。 PM/400 がデータを収集し、 SNA 上でデータを送信する場合にのみこのタスクを実行します。
2. ネットワーク・システムを利用している場合、 iSeries ナビゲーターのインベントリー機能を使用してデータを収集して、ユニバーサル・コネクションを介してそれらのシステム用のデータを伝送することをお勧めします。

リモート・サーバーを定義するには、以下のステップを実行します。

1. コマンド行に **GO PM400** と入力します。

2. PM/400 メニューで 5 (AS/400 リモート・システムでの処理) と入力して、Enter を押します。最初に表示されたリモート・サーバーは表示されません。新規リモート・ロケーションを作成する必要があります。
3. F6 (作成) を押して、新規リモート・ロケーションを作成します。
4. 以下の情報に関する値を報告します。ネットワーク属性の表示 (DSPNETA) コマンドを使用して、リモート・システムからこれらの値を表示します。
   - ローカル・ネットワーク ID
   - デフォルトのローカル・ロケーション

   「リモート iSeries システムでの処理 (Work with Remote iSeries Systems)」画面は、リモート・サーバーのリストを表示します。このリストには、サーバーの状況 (活動状態または非活動状態) およびそれぞれのサーバーごとの説明が含まれています。
5. 「PM/400 リモート・サイトの保守 (PM/400 Remote Site Maintenance)」画面または「リモート・サイト iSeries の変更 (Change Remote Site AS/400)」画面を使用して、リモート・サイト・サーバーに関する説明を作成または変更します。リモート・ロケーション名は、すべてのリモート・サーバーを通して固有でなければなりません。

PM/400 は、データがリモート・サーバーから受信された後、 1 次サーバーから IBM へのデータの送信を自動的にスケジュールします。自動スケジューリングがご使用の作業管理体系に合わない場合には、 1 次サーバーからデータの送信を手動でスケジュールすることができます。データの送信を手動でスケジュールするには、『PM/400 スケジューラー』を参照してください。

PM/400 ソフトウェアは、データを受信するサーバー (ホスト・サーバー) とデータを送信するサーバー (リモート・サーバー) との間で拡張分散ネットワーク機能 (APPN) リンクを定義してあることを前提とします。システム値 QCRTAUT (デフォルトの共通認可の作成) が *EXCLUDE または *USE に設定されている場合、制御装置記述を定義する方法については、『リモート・サーバー用の装置記述の作成』を参照する必要があります。ネットワークがこれらの前提事項を満たしていない場合、それぞれのリモート・サーバーへの接続をサポートするための装置のペアの作成については、『非 APPN ネットワークに関する考慮事項』を参照してください。

リモート・サーバーを定義し終わったら、特定の回線接続を使用するために PM/400 のカスタマイズを行う準備ができています。

非 APPN ネットワーク内のリモート・サーバーの処理: 1 次サーバーは、その他のサーバーから PM/400 データを受信してから、IBM にデータを送信します。リモート・サーバーは、1 次サーバーに PM/400 データを送信します。以下の情報は、参照している制御装置が前もって定義されていることを前提としています。

PM/400 がデータを収集し、SNA 上でデータを送信する場合にのみ、それぞれのリモート・サーバーへの接続をサポートするための装置のペアを作成する必要があります。

1. 装置記述の作成 (APPC) (CRTDEVAPPC) コマンドを使用します。リモート・サーバー上で、CRTDEVAPPC と入力します。 F4 を押してパラメーターを入力するように求め、以下の情報を持つ値を定義します。

リモート・システム

DEVD(Q1PLOC)	装置記述の名前を指定します。
RMTLOCNAME(Q1PLOC)	リモート・ロケーションの名前を指定します。

ONLINE(*YES)	システムの始動または再始動時にこの装置がオンラインに変更されるかどうかを指定します。
LCLLOCNAME(Q1PRMxxx)	ローカル・ロケーション名を指定します。 Q1PRMxxx は、1 次サーバーの RMTLOCNAME と一致します。 xxx は、それぞれのリモート・ロケーションごとに固有です。
CTL(yyyyyy)	接続された制御装置の名前を指定します。 yyyyyy は、1 次サーバーに接続する制御装置です。
MODE(Q1PMOD)	モード名を指定します。
APPN(*NO)	装置が APPN 対応であるかどうかを指定します。

2. 1 次サーバーに関する以下の情報を指定します。コマンド行に、CRTDEVAPPC と入力します。 F4 を押してパラメーターを入力するように求め、以下の情報を持つ値を定義します。

1 次サーバー

DEVD(Q1PRMxxx)	装置記述の名前を指定します。ここで使用されている名前は、リモート・システムの装置記述名と一致します。
RMTLOCNAME(Q1PRMxxx)	リモート・ロケーションの名前を指定します。ここで使用されている名前は、リモート・サーバーの LCLLOCNAME 値と一致します。 xxx は、それぞれのリモート・ロケーションごとに固有です。
ONLINE(*YES)	システムの始動または再始動時にこの装置がオンラインに変更されるかどうかを指定します。
LCLLOCNAME(Q1PLOC)	ローカル・ロケーション名を指定します。この値は、リモート・サーバーの RMTLOCNAME と一致します。
CTL(aaaaaa)	接続された制御装置の名前を指定します。 aaaaaa は、リモート・サーバーに接続する制御装置です。
MODE(Q1PMOD)	モード名を指定します。
APPN(*NO)	装置が APPN 対応であるかどうかを指定します。

3. APPC 装置の定義が完了したなら、装置をオンに変更します (構成の変更 (VRYCFG) コマンド)。リモート・サーバー上で、VRYCFG と入力します。 F4 を押して、パラメーターを入力するように求めます。

リモート・システムをオンに変更

CFGOBJ(Q1PLOC)	構成オブジェクトを指定します。
CFGTYPE(*DEV)	構成オブジェクトのタイプを指定します。
STATUS(*ON)	状況を指定します。

4. PM/400 メニューでオプション 5 を入力して、リモート・サーバーとして Q1PRMxxx を追加します。リモート・サーバーの追加方法については、『リモート・サーバーの処理』を参照してください。

これで PM/400 の構成が完了しました。 PM/400 で実行できるその他のタスクについては、『PM/400 の管理』を参照してください。

PM/400 用の装置記述の作成: 以下のステップでは、それぞれのリモート・サーバーで、デフォルトの共通認可の作成 (QCRTAUT) システム値が *EXCLUDE または *USE に設定されている必要があります。QUSER が装置記述 Q1PLOC への *CHANGE 権限を持っていない場合、リモート送信は失敗します。これらのステップでは、装置が自動的に作成または削除されないようにします。

注: このタスクは、PM/400 がデータを収集し、SNA 上でデータを送信する場合にのみ必要です。

装置を自動的に作成できる場合、装置記述は、QCRTAUT に設定されている値に応じて、 PUBLIC *EXCLUDE または *USE 権限付きで作成されます。装置を自動的に作成または削除できるかどうかは、制御装置によって制御されます。これらのパラメーターがシステム上で定義される方法を決定するには、以下のコマンドを参照してください。

- 制御装置記述の作成 (APPC) (CRTCTLAPPC) コマンド
- 制御装置記述の変更 (APPC) (CHGCTLAPPC) コマンド
- 制御装置記述の表示 (DSPCTLD) コマンド

APPN を使用するように構成されていないシステムの場合、装置記述の作成方法については、『非 APPN 環境内でのリモート・サーバーの処理』を参照してください。

以下の情報は、ホスト・サーバーと通信するために使用される制御装置がリモート・サーバー上で前もって定義されていることを前提とします。

リモート・サーバー上で、装置記述 Q1PLOC を再作成します。

```
VRYCFG  CFGOBJ(Q1PLOC)
        CFGTYPE(*DEV)
        STATUS(*OFF)


DLTDEVD   DEVD(Q1PLOC)


CRTDEVAPPC DEVD(Q1PLOC)
           RMTLOCNAME(Q1PLOC)
           ONLINE(*NO)
           LCLLOCNAME(name of remote system)
           RMTNETID(remote netid of primary (or central) system)
           CTL(name of controller that the device will be attached to)
           AUT(*EXCLUDE)



CRTOBJAUT  OBJ(Q1PLOC)
           OBJTYPE(*DEVD)
           USER(QUSER)
           AUT(*CHANGE)


VRYCFG     CFGOBJ(Q1PLOC)
           CFGTYPE(*DEV)
           STATUS(*ON)
```

これで PM/400 の構成が完了しました。 PM/400 で実行できるその他のタスクについては、『PM/400 の管理』を参照してください。

PM/400 のカスタマイズ: 「PM/400 カスタマイズの処理 (Work with PM/400 Customization)」画面では、以下の機能を提供します。

PM/400 ソフトウェアの操作用のグローバル・パラメーターの設定
グローバル・パラメーターにより、以下の項目をカスタマイズすることができます。以下のフィールドの説明については、オンライン・ヘルプを参照してください。

- 優先順位の限界
- 傾向およびシフト・スケジュール
- パフォーマンス・データ・ライブラリー
- 除去仕様

PM/400 データ電話番号の定義
米国およびカナダ以外ではデータの受け取り先の IBM の所在地の電話番号を PM/400 に指定しておかなければなりません。ほとんどのロケーションでは、PM/400 の構成を開始すると PM/400 はユーザーのロケーションに合った正しいデータ電話番号の選択を試みます。

PM/400 での回線のオンおよびオフの変更
「PM/400 回線制御 (PM/400 Line Control)」画面では、 PM/400 を使用して、回線をオフに変更し、PM/400 データを送信してから、回線を接続保留状態に戻すことができます。

グローバル・パラメーターをカスタマイズするには、以下のステップを実行します。

1. コマンド行に **GO PM400** と入力します。
2. 「PM/400 カスタマイズの処理 (Work with PM/400 Customization)」を表示するには、 PM/400 メニューで a 3 と入力して、Enter を押します。

収集サービスを使用して PM/400 データを収集する場合には、 PM/400 のデータ収集の考慮事項に配慮する必要があります。

PM/400 で実行できるその他のタスクについては、『PM/400 の管理』を参照してください。

PM/400 データ番号の確認: サーバーが IBM への直接ダイヤル接続を使用している場合、PM/400 電話番号が正しいことを確認する必要があります。電話番号には、ユーザー回線用の正しい接頭部が含まれていなければなりません。

注: これは、SNA 送信にのみ当てはまります。

エレクトロニック支援の電話番号の形式をチェックするには、以下のステップを実行します。

1. `DSPDTAARA DTAARA(QUSRSYS/QESTELE)`
   と入力して Enter キーを押します。
2. オフセット 0 にある接続番号の接頭部を判別します。たとえば、オフセット 0 が **'T9:1800xxxxxxx'** である場合、接頭部は **T9:** です。
3. `DSPDTAARA DTAARA(QUSRSYS/Q1PGTELE)`
   と入力して Enter キーを押します。
4. オフセット 0 (ゼロ) が、使用するダイヤリング・ストリングです。 (他の番号は使用しません。)

5. ECS 回線を使用して PTF をオーダーする場合、オフセット 0 (ゼロ) のフォーマットを、ECS 回線用に使われるフォーマットである CALL QESPHONE に対して比較し、使用するストリングを書き留めて、それをステップ 2 にある値と比較することができます。

   電話番号は異なっても、接頭部は同じ (つまり、SST9:1800...、SST:1800... など) でなければなりません。

電話番号を変更する必要が生じた場合、データ域の変更 (CHGDTAARA) コマンドを次のように使用します。

CHGDTAARA と入力します。ただし DTAARA は Q1PGTELE、LIB は QUSRSYS、サブストリングの開始位置は *ALL、そして New の値は 'SST:18005475497' です。

注: New の値は、各自のダイヤル接頭部の後に 18005475497 (米国およびカナダの場合) が続いている値でなければなりません。

これで PM/400 構成が完了しました。次に実行できるタスクについては、『PM/400 の管理』を参照してください。

PM/400 用の直接ダイヤル回線の設定: ほとんどのロケーションでは、PM/400 は、ユーザーのロケーションに合った正しいデータ電話番号を選択しようとします。常に、PM/400 データ電話番号が正しいことの確認をする必要があります。 PM/400 データ電話番号および PM/400 サポート番号を含む情報を持っていない場合には、 IBM サポート担当者に連絡してください。 IBM サポート担当者が正しい電話番号をお知らせします。

注: この電話番号は、ユニバーサル・コネクションを介してデータを送信する場合には必要ではありません。この電話番号は、直接ダイヤル回線を使用している場合にのみ必要です。

PM/400 データ電話番号を定義するか、または PM/400 データ電話番号を変更するには、以下のステップを実行します。

1. コマンド行に **GO PM400** と入力します。
2. 「PM/400 カスタマイズの処理 (Work with PM/400 Customization)」を表示するには、 PM/400 メニューで a 3 と入力して、Enter を押します。
3. この画面で、電話番号フィールドを示す画面のセクションが表示されるまで前方にスクロールします。
4. **「IBM PM/400 電話番号 (IBM PM/400 phone number)」**フィールドに正しいダイヤル順序を入力します。多くの IBM モデムの場合、ダイヤル音用のコロン (:) 文字を使用する必要があります。

PM/400 での回線のオンおよびオフの変更: 時折、PM/400 が使用する回線が接続保留状態になることがあります。この状態では、PM/400 が回線にアクセスしてデータを送信することができません。「PM/400 回線制御 (PM/400 Line Control)」画面では、 PM/400 を使用して、回線をオフに変更し、データを送信してから、回線を接続保留状態に戻すことができます。この画面を使用すると、 PM/400 送信タスク (Q1PCM1) を変更して回線状態をチェックしたり、該当する回線をオフに変更することができます。いったん送信が完了すると、同じ回線は接続保留状態に入れられます。

注: このタスクは、PM/400 がデータを収集し、SNA 上でデータを送信する場合にのみ必要です。

回線をオンおよびオフに変更するには、以下のステップを実行します。

1. コマンド行で **PMLINMON** と入力して、PM/400 回線モニター機能を開始します。「PM/400 回線制御 (PM/400 Line Control)」画面が表示されます。
2. 最初の画面上に示されている警告を読んでから、Enter を押します。

3. PM/400 がオフに変更する必要のある回線、制御装置、および装置の組み合わせを定義します。
4. 機能のマスター制御スイッチとして、プロンプト **Do you want PM/400 automatic line control active?** を使用します。 **YES** を指定する場合、PM/400 機能は活動状態になります。 **NO** を指定する場合、機能は使用不可になります。

   NO を指定する場合、 **YES** を指定したときに再び回線制御リストを定義する必要はありません。回線だけを指定することによって、回線をオンおよびオフに変更することができます。 3 つの説明すべてを指定することによって、回線、制御装置、および装置をオンおよびオフに変更することができます。
5. 定義した回線、制御装置、および装置を確認します。 Enter を押して、選択項目の要約を表示します。
6. Enter を押して選択項目を確認するか、F12 を押して、項目を変更するための前の画面に戻ります。

PM/400 の構成 (CFGPM400) コマンドを使用して、PM/400 回線制御を設定することもできます。

PM/400 で実行できるその他のタスクについては、『PM/400 の管理』を参照してください。

PM/400 の管理

PM/400 を使用するようにネットワークをセットアップした後、以下のタスクを実行することができます。

PM/400 の非活動化
PM/400 を停止する方法を解説します。

連絡先情報の変更
連絡先情報の元の設定を変更する方法について学習します。

PM/400 を使ったジョブのスケジュール
PM/400 を使用してジョブをスケジュールする方法を学習します。

PM/400 分析での項目の省略
PM/400 を使用して分析を実行するときに、ジョブ、ユーザー、および通信回線を省略する方法について学習します。

PM/400 の暫時オフ
PM/400 を暫時停止する方法について学習します。

PM/400 の状況の表示
iSeries ナビゲーターまたは PM/400 メニューを使用して、 PM/400 状況を表示する方法について学習します。

PM/400 報告書の表示
PM/400 報告書の例およびそれらの報告書を解釈する方法に関する説明が示されます。

データのグラフ・ヒストリーの表示
グラフ・ヒストリーには、指定の期間中に収集されたパフォーマンス・データがグラフィック表示で示されます。このデータを表示する方法が解説されています。

PM/400 の非活動化: PM/400 の実行を停止するには、以下の方法のいずれかを使用することができます。

iSeries ナビゲーターを使用する場合

以下のステップを実行します。

1. iSeries ナビゲーターで、PM/400 を実行しているシステムを拡張表示します。

2. **「構成およびサービス」**を展開します。
3. **「収集サービス」**を右クリックします。
4. **PM/400** を選択します。
5. **停止**を選択します。
6. PM/400 を停止したいシステムを選択します。
7. **「OK」**をクリックします。

API の使用

PM/400 (Q1PENDPM) API の終了 (End PM/400 (Q1PENDPM) API) を使用して、 PM/400 を非活動化します。

PM/400 で実行できるその他のタスクについては、『PM/400 の管理』を参照してください。

PM/400 連絡先情報の変更: PM/400 ソフトウェアの構成中に、連絡先の相手を識別し、所属の組織に関するメール情報を入力しました。後で情報を更新する必要がある場合には、「連絡先情報の処理 (Work with Contact Information)」オプションを使用します。連絡先情報を変更するには、以下のステップを実行します。

1. コマンド行に **GO PM400** と入力します。
2. PM/400 メニューで 1 と入力して、Enter を押します。「連絡先情報の処理 (Work with Contact Information)」画面が表示されます。
3. 連絡先情報を変更し、適当であれば Enter を押します。

PM/400 で実行できるその他のタスクについては、『PM/400 の管理』を参照してください。

PM/400 を使ったジョブのスケジュール: PM/400 ソフトウェアに不可欠なのは、 PM/400 パフォーマンス・データの収集および分析をサポートするのに必要なジョブを自動的に開始するスケジューラーです。

PM/400 ソフトウェア活動化プロセスの一環として、Q1PSCH というジョブも開始することになります。次にこのジョブは、以下の表に示されている他のジョブを開始します。

PM/400 のスケジュール・ジョブにアクセスするには、次のようにします。

1. コマンド行で **GO PM400** と入力します。
2. PM/400 メニューで 2 を入力して、Enter を押します。「自動スケジュールされたジョブの処理 (Work with Automatically Scheduled Jobs)」画面が現れます。
3. 各ジョブの状況をアクティブから非アクティブに変更することができます。変更したいジョブの隣に 2 (変更) を入力します。「自動スケジュールされたジョブの変更 (Change Automatically Scheduled Jobs)」画面が示されます。

以下の表は、選択できる PM/400 ジョブを一覧で示しています。

PM/400 スケジュール済みジョブ

ジョブ	スケジュール	機能
Q1PTEST	活動化時	PM/400 が活動化されてから非活動化状態になることを確認します。

Q1PCM1	毎週	削減したパフォーマンス・データを IBM に送信します。このジョブは、直接ダイヤル回線を使用している場合にのみ活動状態です。
Q1PCM2	毎日	通信をオフラインに変更します。
Q1PPMSUB	1 時間ごと	収集サービスがデータを収集していることを確認します。
Q1PDR	毎日	データ削減を実行し、パフォーマンス・データを除去します。
Q1PPG	毎月	削減されたパフォーマンス・データを除去します。
Q1PCM3	必要時	直接ダイヤル送信が回線をオフに変更できなかった後で、通信をオフラインに変更します。
Q1PCM4	必要時	リモート・サーバーから PM/400 データにアクセスします。このジョブが開始するのは、PM/400 メニューでオプション 5 を使用してリモート・システムを追加していた場合のみです。
Q1PPMCHK	4 時間ごと	データ収集が活動状態であることを確認します。
Q1PMONTH	毎月	その月中に送信を追加する必要がある場合に毎月送信できるようにします。デフォルト値は、非活動状態に設定されます。このジョブは、直接ダイヤル回線を使用している場合にのみ使用可能です。

PM/400 で実行できるその他のタスクについては、『PM/400 の管理』を参照してください。

PM/400 分析での項目の省略: PM/400 ソフトウェア・アプリケーションの要約には、バッチ・ジョブ、ユーザー、および通信回線に関する上位 10 項目の分析が含まれています。ただし、ジョブ、ユーザー、または通信回線によってはそのような分析に該当しないものがあります。たとえば、実行時カテゴリー内の通常より長い実行時間を設定したジョブ (自動開始ジョブなど) を除外することができます。

汎用除外機能を使用することによって上位 10 項目の分析からバッチ・ジョブおよびユーザーのグループを省略することができます。たとえば、MYAPP で始まるすべてのジョブを省略するには、MYAPP* を指定します。

省略項目を処理するには、以下のステップを実行します。

1. コマンド行に **GO PM400** と入力します。
2. PM/400 メニューで 4 と入力して、Enter を押します。「上位 10 項目の省略の処理 (Work with Top Ten Omissions)」画面が表示されます。

3. 省略したい項目によって適当なオプション番号を入力します。
   - ジョブを処理するには 1 を入力します。
   - ユーザーを処理するには 2 を入力します。
   - 通信回線を処理するには 3 を入力します。
4. 該当するフィールドに 1 を入力すると、特定のカテゴリーからユーザーまたはジョブのどちらかを省略します。通信回線の場合には、回線の名前を入力してから、該当するフィールドに 1 を入力します。

PM/400 で実行できるその他のタスクについては、『PM/400 の管理』を参照してください。

PM/400 の暫時オフ: 収集サービスがデータを収集していることを確認するために PM/400 を停止する必要がある場合、スケジューラー・ジョブを使用して、日付を Q1PPMSUB ジョブを行う今後の日付に変更することができます。

1. コマンド行に **GO PM400** と入力します。
2. 2 (自動的にスケジュール済みジョブを処理する) と入力します。
3. Q1PPMSUB ジョブの隣に 2 (変更) と入力します。
4. 今後の日付および時刻に日付または時刻を変更します。
5. Enter を押します。この変更により、収集サービスがデータを収集していることを確認するために PM/400 を一瞬停止します。現在収集されているものを終了する必要があります。

注: Q1PPMSUB ジョブに設定された日付および時刻に達するまで、 PM/400 は収集サービスを開始、循環、または変更しません。

スケジューラーを使用して行うことのできるその他の事柄については、『PM/400 を使ったジョブのスケジュール』を参照してください。

PM/400 で実行できるその他のタスクについては、『PM/400 の管理』を参照してください。

PM/400 の状況の表示: iSeries ナビゲーターまたはサーバー上の PM/400 メニューのいずれかを使用して、 PM/400 の状況を表示することができます。「Performance Management/400 状況」ダイアログを使用して、 1 つ以上のサーバーまたはグループ上の PM/400 の状況全体を表示します。たとえば、PM/400 が活動状態であるかどうかに関する詳細が表示されます。 PM/400 メニューを使用して、収集サービス状況、PM/400 スケジューラー状況、パフォーマンス・データのリリース、最後の送信の試み、パフォーマンス・データ・メンバー、およびパフォーマンス・データ・サイズを表示します。

iSeries ナビゲーターから PM/400 の状況全体を表示するには、以下のステップを実行します。

1. iSeries ナビゲーターのエンドポイント・システムまたはシステム・グループを拡張表示します。
2. **「構成およびサービス」**を展開します。
3. **「収集サービス」**を右クリックします。
4. **PM/400** を選択します。
5. **状況**を選択します。

PM/400 メニューから PM/400 の詳細状況を表示するには、以下のステップを実行します。

1. コマンド行に **GO PM400** と入力します。
2. コマンド行から 6 と入力して、Enter を押します。それぞれのフィールドの説明については、オンライン・ヘルプを参照してください。

PM/400 で実行できるその他のタスクについては、『PM/400 の管理』を参照してください。

PM/400 報告書の表示: PM/400e 関連の出力は、月単位または四半期単位の管理報告書およびグラフのセットです。 PM/400e オファリングの報告書には、2 つのオプションがあります。

報告書およびグラフの目的は、サーバーの現行パフォーマンスおよび正確な拡張傾向を管理者が明確に理解できるようにすることです。それぞれの報告書およびグラフを詳しく表示し、それらの利点および使用方法を確かめるには、 PM/400 の Web サイト を参照してください。

PM/400 で実行できるその他のタスクについては、『PM/400 の管理』を参照してください。

Performance Tools

Performance Tools for iSeries ライセンス・プログラムを使用すると、パフォーマンス・データをさまざまな方法で分析することができます。 Performance Tools は、パフォーマンス・データの表示、報告、およびグラフ化を行うためのツールとコマンドを集めたものです。 Performance Tools for iSeries を使用することにより、 収集サービスによって収集されたパフォーマンス・データを表示したり、パフォーマンス・トレースの開始 (STRPFRTRC) コマンドによって収集されたトレース・データを表示することができます。そしてデータを報告書に要約して、システムのパフォーマンス上の問題を調べることができます。パフォーマンス・データのグラフを作成して、ある時間帯の資源の稼働率を調べることもできます。

Performance Tools for iSeries には、基本プロダクトと 2 つのフィーチャー (マネージャーとエージェント) があります。基本に加えていずれかのフィーチャーが必要です。 Performance Tools のマネージャーおよびエージェント・フィーチャーについての詳細は、『マネージャーおよびエージェント・フィーチャーの比較』トピックを参照してください。

Performance Tools の概念
パフォーマンス情報の収集と分析に役立つさまざまなツールを説明しています。具体的にどのツールにどの機能があって、それらがどのように動作するかという詳細情報があります。

Performance Tools のインストールと構成
このトピックにはインストールとセットアップの手順が記載されています。

Performance Tools 報告書
Performance Tools 報告書には、ある時間帯に収集されたデータの情報が示されます。この報告書を使用して、システム資源のパフォーマンスと使用状況についての追加情報を得ることができます。

システム、ジョブ、またはプログラムのパフォーマンスに関するデータを Performance Tools を使用して収集する方法の詳細は、Performance Tools for iSeries を参照してください。そこでは、問題の識別と訂正に役立つ、データの分析と印刷の方法についても説明しています。『HVLPTASK タスク (HVLPTASK task)』で、このタスクが消費する CPU 時間が Performance Tools によってどのように示されるかを参照できます。

Performance Tools の概念

Performance Tools for iSeries ライセンス・プログラムは、サンプル・データとトレース・データという別個の 2 つのタイプのパフォーマンス・データを分析します。 収集サービスはサンプル・データを収集します。サンプル・データは一定の時間間隔で取り込まれる要約データのことです。サンプル・データは、傾向分析やパフォーマンスの分析を行うために収集されます。このデータは、記憶域プールや応答時間といった事柄と関係があります。しかし、収集サービスは、トレース・データの収集をサポートしていません。トレ

ース・データというのは、特定のジョブやトランザクションについての付加的な情報を得るために収集される詳細データです。トレース・データを収集するには、パフォーマンス・トレースの開始 (STRPFRTRC) コマンドか Performance Explorer を使用します。

Performance Tools に組み込まれている機能

Performance Tools には、パフォーマンス・データの収集、分析、および報告のための種々のアプリケーションが組み込まれています。特定のタスクに対してどの機能が使用可能でどれが最適かを識別するのは複雑です。このライセンス・プログラムに組み込まれている機能の説明は、このトピックを参照してください。

マネージャーおよびエージェント・フィーチャーの比較

マネージャー・フィーチャーとエージェント・フィーチャーを使用して、分散環境での Performance Tools の必要な機能を効果的に分割することができます。このトピックでは、これらの 2 つのフィーチャー、それぞれに含まれる機能、およびそれらを最も効果的に使用する方法について記述されています。

パフォーマンス情報の表示

システム資源の使用率のデータを iSeries ナビゲーターで見ることができます。データの表示、そのデータのグラフ化および報告書への要約を行うことができます。この機能の利用方法については、このトピックを参照してください。

Performance Tools が提供する機能: Performance Tools は、報告書、対話式コマンド、およびその他の機能などで構成されています。 Performance Tools にはたとえば以下のものが含まれます。

ツール	説明
システム活動の処理 (WRKSYSACT) コマンド	システム活動の処理 (WRKSYSACT) コマンドを使用すると、現在システムで実行されているジョブ、スレッド、およびタスクを対話式で処理することができます。 WRKSYSACT コマンドは、共用処理プールを使用する区画でのタスクごとの CPU 使用量などのシステム・リソースの使用状況を報告します。該当するタスクで使用されている CPU 時間の表示のためのこのコマンドの使用法について詳しくは、 HVLPTASK タスクを参照してください。
パフォーマンス・データの表示	「パフォーマンス・データの表示 (Display Performance Data)」グラフィカル・インターフェースを使用して、パフォーマンス・データを表示し、データをレポートに要約し、傾向を表示するためのグラフを作成し、システム・パフォーマンスの詳細を分析します。
報告書	報告書は、論理的で役立つ形式で、収集サービス・パフォーマンスを編成します。このツールについては、Performance Tools for iSeries  に詳しく説明されています。
グラフィックス機能	Performance Tools のグラフィックス機能を使用すると、グラフ様式でパフォーマンス・データを処理することができます。グラフを対話式に表示したり、印刷や作図を行うことができます。また、データを図形データ形式 (GDF) ファイルに保管し、他のユーティリティーで使用することができます。このツールについては、Performance Tools for iSeries  に詳しく説明されています。

Performance Explorer	Performance Explorer はデータ収集のツールであり、これを用いて、収集サービスによって収集されたサンプル・データや、一般的な傾向分析によっては識別できないパフォーマンスの問題の原因を究明することができます。 Performance Explorer を使用して、プログラム、プロシージャー、モジュール、またはメソッド・レベルで詳細なアプリケーション分析を行ってください。たとえば、個々のプログラムまたはプロシージャー CPU および入出力統計、または個々のオブジェクト入出力の特性に関するトレース・データを収集することができます。このツールについては、Performance Tools for iSeries  に詳しく説明されています。

マネージャーおよびエージェント・フィーチャーの比較: Performance Tools は、別々にインストール可能な 2 つのフィーチャーで使用できます。このトピックでは、ユーザーのアプリケーションにとってどちらのフィーチャーが適しているかを判断できるように、その 2 つのフィーチャーの違いを説明します。

マネージャー・フィーチャー
Performance Tools マネージャー・フィーチャーは、分散環境での中央側システムまたは単一システムでの使用を目的とした全機能を持つパッケージです。トレース・データの分析、データのグラフ化表示、システムの活動のリアルタイム表示、またはシステムの拡大の管理と追跡が必要な場合は、 Performance Tools ライセンス・プログラムのマネージャー・フィーチャーの方が役に立ちます。

エージェント・フィーチャー
Performance Tools エージェント・フィーチャーはマネージャー機能のサブセットを持ち、より基本的な機能を備えた低価格パッケージです。分散環境では、詳細な分析が必要な場合はデータをマネージャーに送信できるため、エージェント・フィーチャーはネットワーク内の管理対象システム用に適しています。適度なレベルの自己完結性が必要だがエキスパートのスキルを利用できないサイトにとっては、これも有効なツールです。

Performance Tools のエージェント・フィーチャーには、パフォーマンス・データの収集、管理、オンライン表示、データ削減、および分析を単純化する機能があります。 Performance Explorer の報告機能とその関連コマンドは、 Performance Tools for iSeries ライセンス・プログラムの基本オプションに組み込まれているため、マネージャー・フィーチャーまたはエージェント・フィーチャーで使用することができます。エージェント・フィーチャーに含まれていない Performance Tools の主要機能は、パフォーマンスとトレースの報告、パフォーマンス・ユーティリティー (ジョブ・トレース、ファイルおよびアクセス・グループの選択)、システム活動のモニター、およびパフォーマンス・グラフです。

パフォーマンス情報の表示: Performance Tools は iSeries ナビゲーターからパフォーマンス・データを表示することができます。このグラフィカル・インターフェースから、パフォーマンス・データを表示し、データを報告書に要約し、傾向を示すグラフを作成し、システム・パフォーマンスの詳細を分析することができます。

メトリック
iSeries ナビゲーターは選択した時間間隔でのパフォーマンス・メトリックを表示します。「パフォーマンス・データの表示 (Display Performance Data)」ウィンドウの「グラフ」ペインに表示できるパフォーマンス・メトリックには、次のものがあります。

- トランザクション・カウント
- トランザクション応答時間
- 合計 CPU 稼働率
- 対話型 CPU 稼働率

- バッチ CPU 稼働率
- 高ディスク稼働率
- マシン・プール・ページ不在/秒
- ユーザー・プール・ページ不在/秒
- 例外

「詳細」ペインには、選択した時間間隔での詳細なパフォーマンス・データをさまざまな方法で表示することができます。システム・パフォーマンスを分析するために、ジョブ・データ、サブシステム・データ、プール・データ、またはディスク装置のデータを表示できます。

報告書

「パフォーマンス・データの表示 (Display Performance Data)」ウィンドウからは、グラフと詳細データの表示だけでなく、報告書の印刷も行えます。パフォーマンス上の問題の原因になっているシステムの領域をパフォーマンス報告書から調べることができます。さまざまな報告書を実行してシステム資源が使われている場所を調べることができます。 Performance Tools の報告書を印刷できるのは、 Performance Tools for iSeries (5722-PT1) のオプション 1 (マネージャー・フィーチャー) がセントラル・システムにインストールされている場合のみです。マネージャー・フィーチャーについての詳細は、『マネージャーおよびエージェント・フィーチャーの比較』トピックを参照してください。

「パフォーマンス・データの表示 (Display Performance Data)」ウィンドウから印刷できる報告書には、次のものがあります。

- システム
- 構成要素
- ジョブ
- プール
- 資源

iSeries ナビゲーターを介したアクセス

「パフォーマンス・データの表示 (Display Performance Data)」ウィンドウは、 iSeries ナビゲーターで次のステップでアクセスすることができます。

1. iSeries ナビゲーターで**「ユーザー接続」** (またはアクティブ環境) を展開します。
2. 表示したいパフォーマンス・データがあるサーバーを展開します。
3. **「構成およびサービス」**を展開します。
4. **「収集サービス」**を右クリックして**「Performance Tools」**を選択し、**「パフォーマンス・データ (Performance Data)」**を選択します。
5. 表示したいパフォーマンス・データ・ファイルを選択します。
6. **「表示 (Display)」**をクリックします。

iSeries ナビゲーターでの「パフォーマンス・データの表示 (Display Performance Data)」ウィンドウの使用方法についての詳細は、 iSeries ナビゲーターのオンライン・ヘルプを参照してください。

Performance Tools のインストールと構成

Performance Tools をインストールするには、システムの保管 (*SAVSYS) 権限があるユーザー・プロファイルが必要です。システム・オペレーター・プロファイルを使用してこの権限を得ることができます。

Performance Tools は QPFR という名前のライブラリー内で実行する必要があります。ご使用のシステムにこの名前のライブラリーがある場合は、 Performance Tools をインストールする前に、オブジェクトの名前変更 (RNMOBJ) コマンドを使用してそれを名前変更します。このステップによって Performance Tools を正しく操作できるようになります。

次のコマンドを使用して Performance Tools をライブラリー QPFR 内に置きます。

```
RSTLICPGM LICPGM(5722PT1) DEV(NAME) OPTION(*BASE)
```

次に、以下のいずれかを実行します。

- マネージャー・フィーチャーを購入している場合は、次のコマンドを使用します。

  ```
  RSTLICPGM LICPGM(5722PT1) DEV(tape-device-name) OPTION(1)
  ```

- エージェント・フィーチャーを購入している場合は、次のコマンドを使用します。

  ```
  RSTLICPGM LICPGM(5722PT1) DEV(NAME) OPTION(2)
  ```

インストールする CD-ROM が数枚ある場合は、次の状態になることがあります。最初の 1 枚をインストールした後、ライセンス・プログラムは復元されたが言語オブジェクトが復元されていないというメッセージが出されることがあります。このような場合は次の CD-ROM を挿入して、以下を入力します。

```
RSTLICPGM LICPGM(5722PT1) DEV(NAME) RSTOBJ(*LNG) OPTION(*BASE)
```

Performance Tools プログラムをインストールする別の方法として、 GO LICPGM と入力してメニュー・オプションを使用する方法があります。

Performance Tools はプロセッサー・ベースのプログラムです。使用タイプは「同時」で、このプログラムは使用法制限 *NOMAX としてインストールされます。

このプログラムについては、 Performance Tools for iSeries  に詳しく説明されています。

Performance Tools 報告書

Performance Tools には、収集されたデータの調査を容易にする手段が用意されています。 収集サービスは、ほとんどの Performance Tools 報告書のデータにトランザクション、ロック、およびトレース報告書の例外を提供します。これらの 3 つの報告書のトレース情報を収集するには、STRPFRTRC および ENDPFRTRC コマンドを使用する必要があります。 Performance Tools 報告書を使用して、パフォーマンス上の問題を切り分けることができます。ある時間帯のパフォーマンス・データを収集した後、報告書を印刷してシステム資源の使われ方と使われている場所を調べることができます。全体の応答時間が遅くなっている原因の特定のアプリケーション・プログラム、ユーザー、または非効率なワークロードが、報告書から分かります。

以下は、各報告書の説明とそれぞれの報告書を使用する理由の簡単な概要です。各報告書については、

Performance Tools for iSeries  に詳しく説明されています。

注: パフォーマンス・モニターの開始 (STRPFRMON) コマンドを使用して、以前のリリースで収集されたデータの報告書を印刷することができます。

Performance Tools 報告書の概要

報告書	説明	表示される内容	情報の使用方法
システム	収集サービスのデータを使用して、システムの稼働状態の概要を示します。この報告書には、ワークロード、資源の使用、記憶域プール稼働率、ディスク稼働率、および通信に関する要約情報が記載されます。この報告書をしばしば実行して印刷し、システムの使用に関して全般的に把握します。	システム・ワークロード。この報告書にはデータベース権限データが記載されます。	ワークロードの予測
構成要素	収集サービスのデータを使用してシステム報告書の場合と同じシステム・パフォーマンス構成要素についての情報を示しますが、レベルが詳細になります。この報告書は、CPUやディスクなどのシステム資源を大量に消費しているジョブを見つけるのに役立ちます。	資源の使用、通信、システム、およびユーザー・ジョブ。この報告書には、データベース権限データおよび対話型のフィーチャー稼働率も含まれます。	ハードウェアの拡張および構成処理の傾向
トランザクション	トレース・データを使用して、パフォーマンス・データ収集時に起きたトランザクションについての詳細情報を示します。	CPU のワークロードおよび稼働率、ディスク、主記憶装置、トランザクション・ワークロード、オブジェクト競合	ワークロードの予測、プール構成、アプリケーション設計、ファイル競合、およびプログラムの使用
ロック	トレース・データを使用して、システム操作時のロックおよび占有の競合についての情報を示します。この情報をもとに、不十分なロック要求や内部のマシン占有の競合が原因でジョブの処理が遅れているかどうかを判別することができます。このような状態を待機とも呼びます。このような状態が起きている場合は、ジョブがどのジョブを待機しているかということと、待機の長さを判別することができます。	時間別のファイル、レコード、またはオブジェクト競合; 保留ジョブまたはオブジェクト名; 要求ジョブまたはオブジェクト名	問題分析。オブジェクト競合の低減または除去。

トレース	トレース・データを使用して、時間を通してトレースされたさまざまなジョブ・タイプ (例: バッチ・ジョブ) の経過を示します。使用された資源、例外、および状態遷移が報告されます。	ジョブ・クラスのタイム・スライス終了およびトレース・データ	問題分析およびバッチ・ジョブの進行
ジョブ	収集サービスのデータを使用して、すべてのまたは選択した間隔およびジョブに関する情報を示します。これには対話型ジョブと非対話型ジョブの詳細および要約情報が含まれます。報告書が長くなる場合は、含めたい間隔とジョブを選択して出力を制限することもできます。	間隔別のジョブ	ジョブ・データ
プール	収集サービスのデータを使用して、サブシステム活動のセクションとプール活動のセクションを示します。データはサンプル間隔ごとに示されます。報告書が長くなる場合は、含めたい間隔とジョブを選択して出力を制限することもできます。	間隔別のプール	プール・データ
資源間隔	収集サービスのデータを使用して、すべてのまたは選択した間隔についての資源の情報を示します。報告書が長くなる場合は、含めたい間隔を選択して出力を制限することもできます。	間隔別のリソース	システム・リソースの使用

Performance Explorer と収集サービスは、別々の収集エージェントです。それぞれは、グループ化された収集データのセットを含む独自のデータベース・ファイルのセットを生成します。同時に両方のコレクションを実行することができます。

他のツールでの報告書のリストについては、次を参照してください。

- Performance Explorer 報告書
- Performance Management/400 報告書

Performance Explorer

Performance Explorer はデータ収集ツールであり、これを使用すると、収集サービスを使用したデータの収集や一般的な傾向分析では特定できないパフォーマンスの問題の原因を特定することに役立ちます。Performance Explorer を使用する理由として次の 2 つがあります。

- パフォーマンス上の問題を問題の原因になっている、システム資源、アプリケーション、プログラム、プロシージャー、またはメソッドに分離する。
- アプリケーションのパフォーマンスを分析する。

Performance Explorer の収集機能および関連コマンドは、 OS/400 ライセンス・プログラムの一部です。報告機能とその関連コマンドは、 Performance Tools for iSeries ライセンス・プログラムの基本オプションの一部であるため、マネージャー・フィーチャーまたはエージェント・フィーチャーで使用することができます。 AS/400 Performance Explorer Tips and Techniques book には、 Performance Explorer の機能の補足例と拡張 Performance Explorer のトレース・サポートの例が記載されています。

Performance Explorer は、一般的なパフォーマンスのモニターを行うツールを使用しても特定できないパフォーマンスの問題の原因を見つけるのに役立つツールです。コンピューター環境がサイズと複雑さの両面で拡大すると、当然パフォーマンス分析も同様に複雑になります。 Performance Explorer は、複合したパフォーマンスの問題に関するデータを収集することにより、そのような複雑さの拡大に対処しています。

注: Performance Explorer は、他のツールで試行した後に使用するツールです。このツールは、パフォーマンスの問題に関与する要因を容易に切り分けることができる特定の形式のデータを収集しますが、そのデータを収集するときはシステムのパフォーマンスに著しい影響を与えることがあります。

このツールは、自身のプログラムのパフォーマンスの理解や改良に関心があるアプリケーション開発者を対象に設計されています。これはまた、パフォーマンス管理を十分理解しているユーザーが、複合したパフォーマンスの問題を識別して切り分ける場合にも役立ちます。

Performance Explorer についての詳細は、 Performance Explorer に関する次のトピックを参照してください。

Performance Explorer の概念
Performance Explorer は、指定されたシステムのプロセスまたは資源についての詳細情報を収集します。このトピックでは Performance Explorer の動作と最適な使用方法を説明しています。

Performance Explorer の構成
詳細なトレース情報を収集するには、トレース対象のアプリケーション・プロセスと最適に連動できるように Performance Explorer を調整する必要があります。

Performance Explorer 報告書
Performance Explorer セッションでパフォーマンス・データを収集した後、そこに含まれる報告書を実行するか、またはデータベース・ファイルを直接照会してそのデータを表示することができます。

詳しくは、Performance Tools for iSeries を参照してください。

Performance Explorer の概念

Performance Explorer は、 収集サービスのように、あとで分析を行うためのデータを収集します。しかし収集するデータのタイプは大きく異なります。収集サービスは、システム資源の消費を最小限にして、一定のスケジュール間隔で広範囲のシステム・データを収集します。一方 Performance Explorer は、トレー

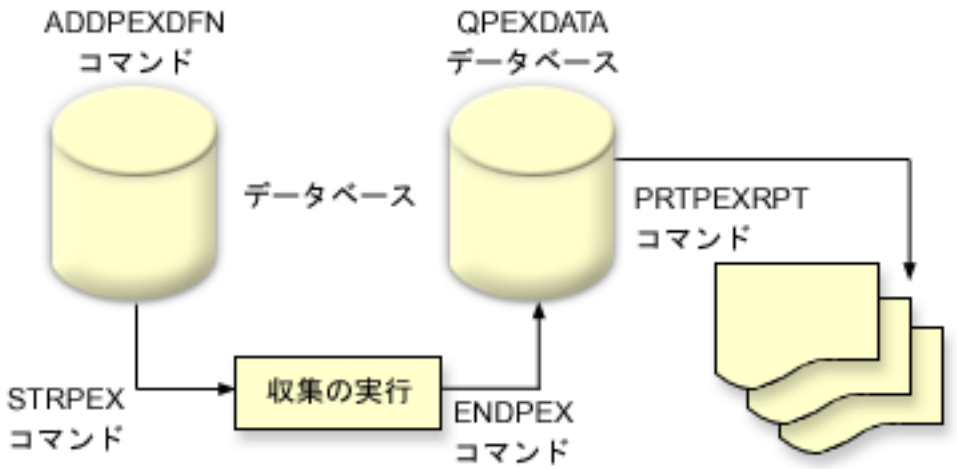
ス・レベルのデータを収集するセッションを開始します。このトレースによって、アプリケーション、ジョブ、またはスレッドが消費する資源についての詳細情報が大量に生成されます。具体的には、Performance Explorer を使用して、システム生成のディスク入出力、プロシージャー呼び出し、 Java メソッド呼び出し、ページ不在、および他のトレース・イベント のような領域についての特定の疑問を解くことができます。 Performance Explorer でパフォーマンスの問題の切り分けを効率的に行えるのは、これが非常に限定された非常に詳細な情報を収集することができるためです。たとえば、収集サービスを使用すると、ディスク記憶域が急速に消費されているということが分かります。 Performance Explorer を使用すると、ディスク・スペースの消費が多すぎるプログラムやオブジェクトとその理由を特定することができます。

注: Performance Explorer のデータと収集サービスのデータを同時に収集することができます。

Performance Explorer の動作

次に示す図は、Performance Explorer での典型的なパスを示したものです。これらのステップのそれぞれの詳細は、『Performance Explorer の構成』を参照してください。この図は基本動作サイクルを示しており、次のステップで構成されます。

1. Performance Explorer のデータ収集を定義します。特定のイベントについての比較値を指定することによりフィルターを追加して、収集するデータの量を制限することもできます。
2. Performance Explorer を開始して、定義に基づいたデータの収集を行います。
3. ユーザーのプログラム、コマンド、またはワークロードを実行します。
4. 収集を終了します。収集されたデータは一連のデータベース・ファイルに保管されます。
5. データベース・ファイルから報告書を作成して印刷します。



Performance Explorer についての詳細は、 Performance Explorer に関する次のトピックを参照してください。

Performance Explorer の定義

Performance Explorer が収集するデータとその収集方法を決定するパラメーターと条件は、Performance Explorer の定義を使用して構成して保管します。このトピックではそれらの定義の使用方法を説明し、簡単な定義のサンプルを示しています。

Performance Explorer データベース・ファイル
Performance Explorer が収集するデータは、 Performance Explorer データベース・ファイルに保管されます。

Performance Explorer の利点
Performance Explorer には、詳細なパフォーマンス情報の収集と分析に役立つ数々の機能があります。このトピックではそれらのさまざまな機能の概要を説明しています。

Performance Explorer の定義: Performance Explorer データを収集するには、収集するデータについて Performance Explorer に通知する必要があります。これは、 PEX 定義の追加 (ADDPEXDFN) コマンドを使用して Performance Explorer 定義を作成することにより行えます。定義が完了し、保管された後、作業のサイクルの次のタスクを続行することができます。

新しい定義の作成前に、必要な情報の種類、および必要な詳細事項の量を考慮してください。 Performance Explorer は、次のタイプのデータ収集を提供します。

統計タイプ定義
CPU を過度に消費するか、または多くのディスク入出力操作を実行するアプリケーションおよび IBM プログラムまたはモジュールを識別します。通常、統計タイプは、潜在的なパフォーマンス・ボトルネックをさらに調査すべきプログラムを識別するのに使用します。

- OS/400 プログラム、プロシージャー、および MI 複合指示の最初のオーダー分析に適しています。
  - 呼び出しの数を指定する
  - インラインおよび累積 CPU 使用量の両方をマイクロ秒単位で指定する
  - 同期および非同期入出力のインラインおよび累積数を指定する
  - 実行される呼び出しの数を指定する
- 短い、または長い実行に合うように作動します。
- 収集されるデータのサイズは、すべての実行に対してかなり小さく、一定しています。
- ILE プロシージャーのランタイム収集オーバーヘッドは、呼び出しの頻度のための問題である可能性があります。ランタイムのレベルが下がっても、収集される統計は正確です。これは Performance Explorer が、データからほとんどの収集オーバーヘッドを除去するためです。
- 組み合わされた、または個別にされたデータ域を使用します。ADDPEXDFN コマンドの MRGJOB パラメーターは、すべてのプログラム統計が 1 つのデータ域に蓄積されるか、それとも個別にされるか (例: 各ジョブごとに 1 つのデータ域) を指定します。

統計は、階層方式または水平方式のどちらかで構造化されます。

- 階層構造は、ツリー中の各ノードが、ジョブまたはタスクによって実行されるプログラム・プロシージャーを表す呼び出しツリー形式に統計を編成します。
- 水平構造は、統計を、プログラムまたはプロシージャーそれぞれが統計の独自のセットを持つように、単純リストに編成します。

この例は、MYSTATS という名前の Performance Explorer 統計定義の例で、プログラムまたはプロシージャー・レベルごとに CPU およびディスク資源使用量を示します。

```
ADDPEXDFN DFN(MYSTATS) /* The name of the definition. */
TYPE(*STATS) /* The type of definition */
JOB(*ALL) /*All Jobs */
TASKS(*ALL) /*All tasks */
MRGJOB(*YES) /* Merge records to reduce collection overhead */
DTAORG(*FLAT) /* Do not keep track of who calls who */
```

プロファイル・タイプ定義

ソース・プログラム・ステートメント番号に基づき、CPU を過度に消費する高水準言語 (HLL) プログラムを識別します。プログラムの開始とプログラムの終了時のサブルーチンとの間で、絶えず分岐しているプログラムも識別できます。プログラムの大きさが十分なものである場合、この往復の繰り返しによって、主記憶域が限られているシステムにおいて、ページ不在率が極端に増えることがあります。

- プログラム・プロファイル (ADDPEXDFN コマンドで TYPE(*PROFILE) および PRFTYPE(*PGM) を指定する)
  - 特定のジョブ内の一連のプログラムで、時間がかかっている部分の明細を提供します。
  - プログラム、モジュール、プロシージャー、ステートメント、または指示によって、データを要約できます。
  - 実行の長さに関係なく、収集のサイズはかなり小さく、一定です。
  - 16 MI プログラムの制限とは、これを 2 番目のオーダー分析ツールとして使用するべきであるということです。
  - サンプル間隔を変更することにより、オーバーヘッドが変更されます。ベンチマークには、 2 ミリ秒の間隔が最初の選択として適しています。
  - 指定されるプログラムの数、または指定されるプログラムのサイズのため、画面区画サイズには制限がありません。

  この例は、PGMPROF という名前の Performance Explorer プログラム・プロファイル定義の例で、特定のプロシージャーの使用量を示します。

```
ADDPEXDFN DFN(PGMPROF) /* The name of the definition. */
TYPE(*PROFILE) /* The type of definition */
JOB(*ALL) /*All Jobs */
PGM((MYLIB/MYPGM MYMODULE MYPROCEDURE)) /* The name of the program to monitor. */
INTERVAL(1) /* A 1-millisecond sample will be taken. */
```

- ジョブ・プロファイル (ADDPEXDFN コマンドで、TYPE(*PROFILE) および PRFTYPE(*JOB) を指定する)
  - 一連のジョブまたはタスクの収集で、時間がかかっている部分の明細を提供します。
  - 収集のサイズは比較的小さいですが、一定ではありません。サイズは、実行が長くなればなるほど大きくなります。
  - システム上のすべてのジョブおよびタスクのプロファイルを作成できるか、または収集されるデータの範囲を数個のジョブまたは対象となるタスクだけに絞ることができます。
  - サンプル間隔を変更することにより、オーバーヘッドが変更されます。ベンチマークには、 2 ミリ秒の間隔が最初の選択として適しています。

  この例は、ALLJOBPROF という名前の Performance Explorer ジョブ・プロファイル定義の例で、すべてのジョブの使用量を示します。

```
ADDPEXDFN DFN(ALLJOBPROF) /* The name of the definition. */
TYPE(*PROFILE) /* The type of definition */
PRFTYPE(*JOB) /* A job profile type will be monitored. */
JOB(*ALL) /*All Jobs */
TASKS(*ALL) /*All tasks */
INTERVAL(1) /* A 1-millisecond sample will be taken. */
```

トレース定義

システム上の 1 つ以上のジョブによって生成されるパフォーマンス・アクティビティーのヒストリー・トレースを収集します。トレース・タイプは、イベントがいつ発生したか、および発生した順序

についての特定の情報を収集します。トレース・タイプは、プログラム、ライセンス内部コード (LIC) タスク、OS/400 ジョブ、およびオブジェクト参照情報についての、詳細な参照情報を収集します。

- 一部の一般的なトレース・イベントは次のとおりです。
  - プログラムおよびプロシージャー呼び出しおよび戻り。
  - 記憶域 (例: たとえば割り振りおよび割り振り解除)。
  - ディスク入出力 (例: 読み取り操作と書き込み操作)。
  - Java メソッド (例: 入り口と出口)。
  - Java (例: オブジェクト作成とガーベッジ収集)。
  - ジャーナル (例: コミットの開始とコミットの終了)。
  - 同期化 (例: mutex ロックとアンロック、またはセマフォー待機)。
  - 通信 (例: TCP、IP、または UDP)。
- 実行が長いと、より多くのデータを収集します。

この例は、DISKTRACE という名前の Performance Explorer トレース定義で、すべてのディスク・イベントの使用量を示します。

```
 ADDPEXDFN DFN(DISKTRACE) /* The name of the definition. */
 TYPE(*TRACE) /* The type of definition */
 JOB(*ALL) /*All Jobs */
 TASKS(*ALL) /*All tasks */
 TRCTYPE(*SLTEVT) /* Only selected individual events and machine instructions
are included in the trace definition */
 SLTEVT(*YES) /* *SLTEVT allows you to specify individual machine instructions
and events to be specified in addition to the categories of events
available with the TRCTYPE parameter. */
 DSKEVT((*ALL)) /* All disk events are to be traced. */
```

Performance Explorer データベース・ファイル: 次の表は、データ収集コマンドを使用したときにシステムによって収集される Performance Explorer のデータ・ファイルを示したものです。単一のファイルの内容を表示するには、ファイル・フィールド記述の表示 (DSPFFD) コマンドを次のように入力します。

```
DSPFFD FILE(xxxxxxxxx)
```

xxxxxxxxx は表示するファイルの名前です。

ファイルに含まれる情報のタイプ	ファイル名
参照情報	QAYPEREF
一般情報	QAYPERUNI
PMC 選択	QAYPEFQCFG
基本構成情報	QAYPECFGI
収集されたマシン・インターフェース (MI) 複合命令	QAYPELCPLX
収集されたジョブ	QAYPELJOB
データ収集対象のメトリック	QAYPELMET
収集されたマシン・インターフェース (MI) プログラム、モジュール、またはプロシージャー	QAYPELMI
データ収集対象のライセンス内部コード (LIC) モジュール	QAYPELLIC
データ収集対象のタスク名	QAYPELNAMT
データ収集対象のタスク番号	QAYPELNUMT

ファイルに含まれる情報のタイプ	ファイル名
マシン・インターフェース (MI) 複合命令のマッピング	QAYPEMICPX
イベント・タイプとサブタイプのマッピング	QAYPEEVENT
ハードウェア・マッピング・データ	QAYPEHWMAP
ライセンス内部コード (LIC) アドレス解決マッピング	QAYPEPROCI
セグメント・アドレス解決マッピング	QAYPESEGI
プロセスおよびタスク解決マッピング	QAYPETASKI
すべてのイベントについての共通トレース・データ	QAYPETIDX
補助記憶域管理イベント・データ	QAYPEASM
基本イベント・データ	QAYPEBASE
ディスク・イベント・データ	QAYPEDASD
ディスク・サーバー・イベント・データ	QAYPEDSRV
ページ不在イベント・データ	QAYPEPGFLT
資源管理プロセス・イベント・データ	QAYPERMPM
資源管理占有ロック・イベント・データ	QAYPERMSL
拡張 36 イベント・データ	QAYPES36
セグメント・アドレス範囲 (SAR) データ	QAYPESAR
不明イベント・データ	QAYPEUNKWN
基本統計データ	QAYPESTATS
統計プロファイル要約データ	QAYPEPSUM
ライセンス内部コード (LIC) ブラケット化データ	QAYPELBRKT
マシン・インターフェース (MI) ユーザー・イベント・データ	QAYPEMIUSR
マシン・インターフェース (MI) プログラム・ブラケット化データ	QAYPEMBRKT
マシン・インターフェース (MI) ポインターのアドレス	QAYPEMIPTR
ユーザー定義のブラケット化フック・データ	QAYPEUSRDF
ハードウェア・モニター・データ	QAYPEHMON
ハードウェア・モニター合計データ	QAYPEHTOT
リリース、バージョン、修正レベル	QRLVRM
Performance Explorer レベル標識	QRLLVL
Performance Explorer Java イベント・データ	QAYPEJVA
Performance Explorer Java クラス情報データ	QAYPEJVCI
Performance Explorer Java メソッド情報データ	QAYPEJVMI
Performance Explorer Java 名前情報データ	QAYPEJVNI
同期イベント・データ	QAYPESYNC
通信イベント・データ	QAYPECMN
ファイル・サービス・イベント・データ	QAYPEFILSV
ヒープ・イベント・データ	QAYPEHEAP
PASE イベント・データ	QAYEPASE
トレース・ジョブ同等イベント・データ	QAYPETBRKT
タスク切り替えイベント・データ	QAYPETSKSW

ファイルに含まれる情報のタイプ	ファイル名
同期イベント・データ	QAYPESYNC
プログラム・プロファイル・データ	QAYPEPPANE

Performance Explorer 報告書: Performance Explorer は、プログラムやジョブの動作とパフォーマンスについての詳細情報を収集して、その情報をPerformance Explorer データベース・ファイルに保管します。これらのファイルは、SQL を使用して、またはさまざまな報告書のいずれかを実行することにより照会することができます。 Performance Explorer で、統計、プロファイル、トレース、および基本の、 4 つの異なる報告書を生成することができます。特定の定義を使用してこれらの報告書の 1 つを生成する理由については、『Performance Explorer の定義』を参照してください。各報告書については、 Performance Tools for iSeries に詳しく説明されています。

Performance Explorer 報告書の作成と印刷は、 Print Performance Explorer Report (PRTPEXRPT) コマンドを使用して行います。トレース報告書をカスタマイズする場合は、 OUTFILE パラメーターを使用します。 Performance Explorer の各タイプのデータの報告書を印刷する場合のコマンドの例を次に示します。

- *STATS 報告書を、使用された CPU 時間別に分類して印刷する。

```
PRTPEXRPT MBR(MYSTATS) LIB(MYLIB) TYPE(*STATS) STATSOPT(*CPU)
```

- プロファイル報告書を、プロシージャー別に要約して印刷する。

```
PRTPEXRPT MBR(MYPROFILE) LIB(MYLIB) TYPE(*PROFILE) PROFILEOPT(*SAMPLECOUNT *PROCEDURE)
```

- トレースを、タスク ID 別に分類して印刷する。

```
PRTPEXRPT MBR(MYTRACE) LIB(MYLIB) TYPE(*TRACE) TRACEOPT(*TASK)
```

Performance Explorer は収集したデータを QAVPETRCI ファイルに保管します。このファイルは QPFR ライブラリーに置かれています。単一レコードの内容を表示するには、次のコマンドを入力します。

```
DSPFFD FILE(QPFR/QAVPETRCI)
```

Performance Explorer の利点: Performance Explorer は、iSeries サーバーにおいて詳細なパフォーマンス分析が必要な人に役立ちます。 Performance Explorer を使用すると、次のことが可能です。

- ユーザー、ジョブ、ファイル、オブジェクト、スレッド、タスク、プログラム、モジュール、プロシージャー、ステートメント、または命令アドレスのレベルまで下りて、システムにおけるパフォーマンス上の問題の原因を判別できます。
- ユーザーが開発したソフトウェア、およびシステム・ソフトウェアに関するパフォーマンス情報を収集できます。
- システム上の他の操作のパフォーマンスに影響を与えずに、ある特定のジョブの詳細な分析を行えます。
- データの収集元でないシステム上で、そのデータを分析できます。たとえば、ネットワークの管理対象システムでデータを収集した場合、そのデータをセントラル・サイト・システムに送信して分析することができます。

Performance Explorer の構成

Performance Explorer を構成するには、以下のステップを実行します。

1. iSeries サーバーに、収集するパフォーマンス・データを通知するセッション定義を作成します。「PEX 定義の追加 (ADDPEXDFN)」画面で、収集タイプ、および定義の名前を指定します。この定義は、ライブラリー QUSRSYS の QAPEXDFN ファイルに、その名前でデータベース・メンバーとして保管されます。指定される名前は、 Performance Explorer の開始 (STRPEX) コマンドで使用されます。

2. フィルターを追加 (ADDPEXFTR コマンド) します。 Performance Explorer フィルターは、Performance Explorer セッション中に収集されるパフォーマンス・データを識別し、特定のイベントに比較値を指定することによって収集されるデータの量を制限するためのものです。
3. データの収集を開始 (STRPEX コマンド) します。 *PMCO イベントが収集されていない場合、ジョブは複数の Performance Explorer に入っている可能性があります。*PMCO イベントが収集されている場合、すべての収集の間隔指定が同じであれば (ADDPEXDFN INTERVAL() パラメーター)、ジョブが複数の収集に入っていることが考えられます。
4. 分析したいデータでコマンド、プログラム、またはワークロードを実行します。
5. データの収集を停止し、分析を行うためにデータベース・ファイルに保管します。収集を停止するには、 Performance Explorer の終了 (ENDPEX) コマンドを使用します。
6. パフォーマンス・データを分析します。Performance Tools ライセンス・プログラムに入っている Print Performance Explorer Report (PRTPEXRPT) コマンドは、データの各タイプ (統計、プロファイル、トレース・プロファイル、またはトレース) ごとに、固有のレポートを提供します。分析用の他のオプションは、データベース・ファイルのセット全体で、独自の Query を作成するためのものです。

すべての Performance Explorer コマンドは、以下のどちらかの方法でアクセスできます。

- コマンド・インターフェース。コマンド行でコマンドを入力します。 Print Performance Explorer Report (PRTPEXRPT) コマンド以外のすべてのコマンドは、 OS/400 オペレーティング・システムの一部です。
- Performance Tools のメニュー・オプション。

Performance Explorer の作業の周期を確認するには、 Performance Explorer の概念を参照してください。

Performance Explorer の終了

Performance Explorer セッションを終了するには、 Performance Explorer の終了 (ENDPEX) コマンドを使用します。 ENDPEX コマンドは、収集されたデータ上で次の処置を取ります。

- 収集されたデータを、指定されたライブラリー中のファイル QAYPExxx に入れます。
  これには、OPTION(*END) および DTAOPT(*LIB) を使用します。すべての QAYPExxx ファイルのデータベース・メンバー名は、DTAMBR パラメーターに名前を指定しない限り、セッション名をデフォルトとして使用します。RPLDTA(*NO) を指定すると、このセッション名を使用して収集されたデータを消去することができ、RPLDTA(*YES) を指定すると、収集されたデータを既存のデータに追加することができます。このセッションにかなり慣れていない限り、RPLDTA(*NO) を使用してください。
- 収集されたデータを、単一の IBM 定義ファイルに入れます。
  これには、OPTION(*END) および DTAOPT(*MGTCOL) を使用します。通常は、 IBM サービス担当者の指示の下で、*MGTCOL だけを使用します。 DTAOPT パラメーターで *MGTCOL 値を指定すると、収集情報が管理収集オブジェクトに保管されます。管理収集オブジェクト・オプションは、データが IBM に送られる場合のみ使用します。 Performance Tools が分析できるのはデータベース・ファイルのみです。
- 収集したデータを廃棄します。
  データを保管したい場合は OPTION(*END)、収集したデータを廃棄したい場合は DTAOPT(*DLT) を使用します。廃棄するのは、収集されたデータが使用できないと判断した場合です。たとえば、予期していたジョブの 1 つが期待どおりに開始しなかった場合などが考えられます。 *DLT オプションを選択すると、そのセッションに収集されたパフォーマンス・データは保管されません。
- 収集セッションを中断しますが、終了はしません。
  これには OPTION(*SUSPEND) を使用します。特定のセッション ID に、OPTION(*RESUME) を指定した STRPEX コマンドを出すことによって、データ収集を後で再開することができます。

注: 活動収集セッション名を忘れた場合、 ENDPEX SSNID(*SELECT) コマンドを使用してください。

iDoctor for iSeries

iDoctor for iSeries は、Job Watcher、Object Explorer、および Performance Explorer Analyzer の 3 つの構成要素から構成されるツールのセットです。

Job Watcher および Performance Explorer Analyzer は、パフォーマンス分析を行います。これらの構成要素のサーバー・サイド部分は、さまざまなデータ収集、および使用により適したフォーマットでパフォーマンス・データを強化するために設計された分析プログラムから構成されます。Job Watcher および Analyzer は、柔軟なグラフおよび表のビューでサーバー・データを表示するためのグラフィカル・インターフェースから構成されています。

Object Explorer は、グラフィカル・インターフェースを使用して、 iSeries データをよりアクセスしやすくするために設計されたツールです。システム上のどんなオブジェクト・タイプでもリスト、表示、および記述できます。 Object Explorer を使用すると、システム上のすべての物理ファイルおよび論理ファイルにアクセスしたり、簡単に使用できる Query 定義およびグラフ定義インターフェースを使用して、データから独自の Query およびグラフを作成したりできます。

Job Watcher

Job Watcher は、ジョブが実行していること、および実行していない理由を非常に詳細に表す、リアルタイムの表およびグラフィカル・データを表示します。 Job Watcher は、一定間隔ごとに詳細なジョブ統計を提供する、さまざまな報告書をいくつか作成します。これらの統計により、CPU 稼働率、DASD カウンター、待ち、障害、呼び出しスタック情報、対立情報などを判別することができます。

Object Explorer

Object Explorer を使用すると、iSeries サーバー上のオブジェクトのブラウズおよび処理を行えます。Object Explorer の基本的な機能は Data Viewer で、システム上の物理ファイルの内容を表示し、データから独自の Query およびグラフを作成するための機能です。また、次の機能も Object Explorer に含まれています。

- ライブラリーおよびオブジェクトのリストの簡単かつ高速なフィルター操作
- 高速で、使いやすいユーザー・インターフェースを使った、オブジェクトのコピー (CRTDUPOBJ)、切り取り (MOVOBJ)、名前変更 (RENOBJ)、および削除 (DLT*) 機能
- 論理または物理ファイル・メンバーのブラウズ
- 独自の Query を定義し、後で Query 定義インターフェースで使用するために保管する
- 独自のグラフを定義し、後でグラフ定義インターフェースで使用するために保管する

Performance Explorer Analyzer

Performance Explorer Analyzer は、システムのパフォーマンスを全体的に評価する機能で、 Performance Tools ライセンス・プログラムを使用して実行したことに基づいています。Analyzer は、トレース・データのボリュームを、パフォーマンス問題を際立たせるのに役立つ報告書に圧縮し、問題判別にかかる時間全体を削減します。Analyzer は、CPU 稼働率、物理ディスク操作、論理ディスク入出力、データ域、およびデータ待ち行列の分析のために、使いやすいグラフィカル・インターフェースを提供します。また、Analyzer はアプリケーションのスローダウンの原因を見つけるのにも役立ちます。

詳細については、 iDoctor for iSeries  Web サイトを参照してください。

Performance Trace Data Visualizer (PTDV)

Performance Trace Data Visualizer (PTDV) は、iSeries サーバーで実行するアプリケーションのパフォーマンスを分析するのに使用できる、Java アプリケーションです。PTDV は OS/400 基本オペレーティング・

システムで Performance Explorer と協業し、分析者はプログラム・フローを表示し、詳細 (CPU 時間、現行システム時刻、周期の回数、および命令の回数など) を得ることができます。 Java アプリケーション・トレースを視覚化すると、作成されるオブジェクトの数やタイプ、および Java ロックの振る舞いについての情報などの付加的な情報も表示することができます。また、WebSphere Application Server が生成する Performance Explorer イベントもサポートします。 PTDV では、カラムのソート、データのエクスポートおよびさまざまなレベルでのデータ要約が可能です。

詳細については、 Performance Trace Data Visualizer  Web サイトを参照してください。

Performance Management API

Performance Management API を使用すると、収集サービス、 Performance Collector、Performance Explorer および Performance Management/400 (PM/400) を使用してパフォーマンス・データを収集および管理できます。

Performance Management API には以下のものが含まれます。

- Collection Services API
- Performance Collector API
- Performance Explorer (PEX) API
- Performance Management/400 (PM/400) API

OS/400 パフォーマンス用の処理コマンド

OS/400 には、文字ベースのインターフェースからパフォーマンス・データのモニターをリアルタイムで実行するための多数のコマンドがあります。これらのコマンドを使用して、システム・パフォーマンスに関する特定の質問に応答したり、システムの調整に役立てたりすることができます。iSeries ナビゲーターからのリアルタイムのモニターについての詳細は、iSeries Navigator のモニターを参照してください。

コマンド	機能
活動ジョブの処理 (WRKACTJOB)	システム上で実行する、ジョブの属性および資源の使用効率を見直し、変更します。
ディスク状況の処理 (WRKDSKSTS)	システム・ディスク装置のパフォーマンス情報、および属性を表示します。
システム状況の処理 (WRKSYSSTS)	現行システムの活動の概要を示します。特に、システム上のジョブの数、および記憶域プールの使用状況に関する情報を表示します。
システム活動の処理 (WRKSYSACT) (Work with System Activity (WRKSYSACT))	システム上のジョブおよびタスクを処理します。このコマンドは、Performance Tools ライセンス・プログラム (PT1) の一部です。
オブジェクト・ロックの処理 (WRKOBJLCK) (Work with Object Locks (WRKOBJLCK))	適用を待機しているロックも含め、指定されたオブジェクトのロックを処理し、表示します。
共用記憶域プールの処理 (WRKSHRPOOL) (Work with Shared Storage Pools (WRKSHRPOOL))	マシンおよび基本プールを含めた、共用記憶域プールの使用状況に関する情報を表示し、その属性を変更します。

システム・パフォーマンスの向上

注: フィーチャー・コード #4331 および #6831 (CCIN #6731) は、受注停止になりました。ここで提供されている情報は、既存のユーザーのための参照情報です。 iSeries 記憶域入出力のこのキャッシュ機能への機能拡張はすべて、これらのページを通して使用できます。

拡張キャッシュを使用して、現在の iSeries システムのパフォーマンスを向上させます。拡張キャッシュは、ディスクから読み取られる物理入出力要求の数を減らすことによって入出力サブシステムとシステム応答時間の両方を向上させる、先端の大規模読み取りキャッシュ技術です。拡張キャッシュは、データに関する統計情報を生成し、複合的な管理技法を使用してキャッシュに入れるデータを決定します。拡張キャッシュは、多くのタイプのワークロードに対してかなりの効果を持つことが証明されています。

IBM では、 iSeries コンピューティング環境における拡張キャッシュの利点を知っていただけるよう、革新的なツールを提供しています。**拡張キャッシュ・シミュレーター**は、 iSeries ナビゲーター内の収集サービス機能を通して活動化されます。このシミュレーターは、時間を追って見た実際のワークロードについて、エミュレートされたパフォーマンスの結果をディスク単位で示します。拡張キャッシュ・シミュレーターは、記憶域入出力アダプターのレベルで実行され、拡張キャッシュを管理する場合と同じアルゴリズムを使用します。

さらに、以下のトピックも参照してください。

- **拡張キャッシュの概念**
  拡張キャッシュについて紹介しています。このツールの使用を開始する前に考慮すべき、計画、制約事項、および重要な考慮事項に関する情報を扱います。
- **拡張キャッシュ・シミュレーター**
  現在のコンピューティング環境で拡張キャッシュを使った場合に得られる応答時間の向上を、拡張キャッシュ・シミュレーターで判別する方法について説明します。
- **拡張キャッシュの入手**
  拡張キャッシュ・シミュレーターを使用し、このツールがコンピューティング環境に与える益を確認したユーザーのために、拡張キャッシュの入手方法を説明します。

拡張キャッシュの概念

拡張キャッシュを使用してシステム・パフォーマンスを向上させます。拡張キャッシュは、ディスクから読み取られる物理入出力要求の数を減らすことによって入出力サブシステムとシステム応答時間の両方を向上させる、先端の読み取りキャッシュ技術です。拡張キャッシュは、データベースの読み取りアクションに限らず、すべての読み取りアクションのパフォーマンスを向上させます。これには、統合 xSeries サーバーなどの他のシステム構成要素による読み取りアクションも含まれます。また、拡張キャッシュは、 装置パリティー保護やミラー保護のある記憶域サブシステムでも効果的に機能します。拡張キャッシュは、多くのタイプのワークロードに対してかなりの効果を持つことが証明されています。

拡張キャッシュが働く仕組み

拡張キャッシュは、iSeries 入出力サブシステムに統合されています。拡張キャッシュはディスク・サブシステム制御装置のレベルで操作し、 iSeries システム・プロセッサーには影響を与えません。記憶域入出力アダプターは、 Read Cache Device (固形のディスクなど) を使用してキャッシュ・メモリーを提供することにより、この拡張キャッシュを管理しています。

拡張キャッシュは、データに関する統計情報を生成し、混ぜ合わされた管理戦略を使用してキャッシュに入れるデータを決定します。キャッシュの管理は、入出力アダプター内で自動的に行われ、予測のアルゴリズムを使用してデータのキャッシュを行うように設計されています。このアルゴリズムは、前もって定められた範囲のデータに対し、ホストがどれほど最近に、そしてどれほど頻繁にアクセスしたかを考慮します。

拡張キャッシュの設計は、iSeries サーバー・システムの特別なデータ管理戦略に基づいています。ディスクが装置パリティー保護されているか、ミラー保護されているか、あるいは無保護であるかに関係なく、そのディスクに保管されているデータには、バンド状に発生する傾向があります。つまり、ディスク記憶域には物理的な連続区域があって、活動状態で読み取られている区域、頻繁に書き込みが行われる物理的な連続区域、活動状態で読み取られていながらかつ書き込まれている物理的な連続区域、あるいはあまり頻繁にアクセスされない記憶域の物理的な連続区域が存在することを意味します。

この「バンド状の」データは、拡張キャッシュの設計によって説明付けられます。拡張キャッシュの目的は、読み取り/書き込み、および読み取り専用の特徴を持つバンドをキャッシュに入れることです。書き込み専用の特徴をもつバンドは、記憶域サブシステムの書き込みキャッシュに入れられながらも、大部分は、拡張キャッシュの影響を受けることなく残されます。加えて、拡張キャッシュは、大きなブロックでデータを順次に書き込む、または読み取るパフォーマンスに影響を与えないようにも設計されています。この場合は、ディスクの事前取り出し機能が即時応答を保証しています。これは、システム内の他のキャッシュでも同じです。

さらに、以下のトピックも参照してください。

- **拡張キャッシュの制約事項および考慮事項**
  拡張キャッシュに必要な構成要素を示すと共に、何が期待できるかについてさらに詳しく説明します。
- **拡張キャッシュ・シミュレーター**
  現在のコンピューティング環境で拡張キャッシュを使った場合に得られる応答時間の向上を、拡張キャッシュ・シミュレーターで判別する方法について説明します。
- **拡張キャッシュの開始**
  拡張キャッシュを活動化する方法について扱います。

拡張キャッシュの制約事項および考慮事項: 拡張キャッシュの使用を開始する前に、ご使用のコンピューティング環境に関係し得る制約事項や考慮事項を検討するため、いくつかの事前の計画を立てる必要があります。

制約事項

拡張キャッシュを使用するためには、システムが以下を備えている必要があります。

- 拡張キャッシュをサポートしている 1 つ以上の記憶域入出力アダプター (CCIN 2748 (V4R4 以降が稼働するシステム用)、CCIN 2778 (V5R1 以降が稼働するシステム用)、または CCIN 2757 (最新リリースの V5R2 が稼働するシステム用) (情報 APAR II13365 を参照))
- 拡張キャッシュを活動化する、各記憶域入出力アダプター用の Read Cache Device (RCD) (CCIN 6731 (V4R4 以降が稼働するシステム用))
- Performance Tools for iSeries ライセンス・プログラム

拡張キャッシュは、RCD を通して自動的に使用可能になります。このオン/オフを制御するスイッチはありません。 RCD は、システム割り込みがなくても、 並行メインテナンスによって追加される場合があります。 RCD は、内部ディスク・スロットに常駐し、他のすべてのディスク・タイプおよびキャパシティーで作業します。拡張キャッシュ内の全データは、必ずディスク上にも存在します。ほとんどあり得ないイベントですが、RCD が失敗しても、データが失われることはありません。

入出力アダプター内の他のディスクに対する装置パリティー保護やミラー保護に関して、拡張キャッシュに制約事項はありません。ただし、同じ入出力アダプターの中で拡張キャッシュと Integrated Hardware Disk Compression を一緒に使用することはできません。最後に、拡張キャッシュは、特に iSeries エキスパート・キャッシュを補うために設計されており、これと一緒に使用することも、単独で使用することも可能です。

考慮事項

拡張キャッシュを使用すると、ほとんどの環境において、入出力の応答時間をかなり短縮し、システム入出力のスループットを高めることができます。一般のキャッシュと同様、拡張キャッシュの効果は、システムの構成やワークロードに影響を受けます。拡張キャッシュは、記憶域サブシステム・レベルで行われます。そして、特定のサブシステム内にある一連のディスクのデータをキャッシュに入れます。したがって、システム内の活動状態にある記憶域サブシステムやパフォーマンス重視の記憶域サブシステムに対する拡張キャッシュの追加は、ほとんどの場合、論理的なものです。拡張キャッシュは、事前取り出しタイプのキャッシュとは見なされないので、ディスク内の先読み機能を妨げることはありません。

活発に入出力要求を受け取るディスク記憶域の区域が大きいと、新しいデータをキャッシュに入れようとするときを決定する拡張キャッシュの選択肢が広がります。このような適応性を備えているため、 拡張キャッシュは、さまざまなタイプやサイズのワークロードに対して効果的です。キャッシュ全体の効果を知るには、拡張キャッシュ・シミュレーターを通してこの観点から考慮するのが最善です。

なお、同じ記憶域入出力アダプター上で、拡張キャッシュ・シミュレーターと拡張キャッシュの両方を活動状態にすることはできません。

これらの制約事項や考慮事項を理解すると、 拡張キャッシュを開始する準備が整います。

拡張キャッシュの開始: 拡張キャッシュを開始し、システムのパフォーマンスを向上させるためには、Read Cache Device を購入してください。Read Cache Device がサブシステムのディスク・スロットに挿入されると、拡張キャッシュが活動状態になります。このオン/オフをユーザーが制御するためのスイッチはありません。拡張キャッシュがデータ・フローをモニターし、 Read Cache Device に移植するまでには、1 時間ほどかかります。 1 時間ほど拡張キャッシュを実行すると、システムのパフォーマンスの向上 (現在のワークロードによる) や入出力スループットの増加が確認できるようになります。

ご使用の iSeries システムで拡張キャッシュを使用できるかどうかは、『拡張キャッシュの制約事項および考慮事項』を参照してください。

拡張キャッシュ・シミュレーター

注: フィーチャー・コード #4331 および #6831 (CCIN #6731) は、受注停止になりました。ここで提供されている情報は、既存のユーザーのための参照情報です。 iSeries 記憶域入出力のこのキャッシュ機能への機能拡張はすべて、これらのページを通して使用できます。

拡張キャッシュによって得られるシステム・パフォーマンスの向上を見積もるには、拡張キャッシュ・シミュレーターを使用します。拡張キャッシュ・シミュレーターは、 Performance Tools の 1 つであり、拡張キャッシュによってシステムにもたらされる応答時間の向上を判別できます。この判別は、システムの構成とデータのワークロードに基づいて行うことができ、 Read Cache Device の購入に先立って実行されます。

拡張キャッシュ・シミュレーターは収集サービス内で制御され、 CCIN 2757 入出力アダプターを使用する最新リリースの V5R2 システムが稼働するシステムで使用できます。 (CCIN 2748、CCIN 2778、および CCIN 2757 入出力アダプターは、拡張キャッシュそのものをサポートする同じ記憶域入出力アダプターです。) このシミュレーターは柔軟性が高く、特定のシステムやワークロードの必要に最も適したキャパシティーをより適格に判別できるように、複数の異なるキャッシュ・キャパシティーをエミュレートできるようになっています。実際の Read Cache Device のキャパシティーは 1600 MB です。

拡張キャッシュ・シミュレーターの活動化を通して集められたパフォーマンス情報には、拡張キャッシュの使用によって節約できたであろうディスク読み取りの数が示されます。パフォーマンス・データは、ディスク・アクセス時間における向上の可能性を反映します。

拡張キャッシュの入手

拡張キャッシュ・シミュレーターからパフォーマンス・データを取得し、拡張キャッシュを使用してシステムのパフォーマンスを向上させることを決定したなら、 Read Cache Device (RCD) を購入する必要があります。拡張キャッシュは、RCD を通して自動的に使用可能になります。

拡張キャッシュを実際に使用するためには、以下のシステムが必要です。

- 拡張キャッシュをサポートしている 1 つ以上の記憶域入出力アダプター (CCIN 2748 (V4R4 以降が稼働するシステム用)、CCIN 2778 (V5R1 以降が稼働するシステム用)、 CCIN 2757 (最新リリースの V5R2 が稼働するシステム用))
- 拡張キャッシュを活動化する、各記憶域入出力アダプター用の Read Cache Device (RCD) (CCIN 6731 (V4R4 以降が稼働するシステム用))

拡張キャッシュはこの RCD を通して自動的に使用可能になるため、そのオン/オフを制御するスイッチはありません。 RCD は、システム割り込みがなくても、 並行メインテナンスによって追加される場合があります。 RCD は、内部ディスク・スロットに常駐し、他のすべてのディスク・タイプおよびキャパシティーで作業します。拡張キャッシュ内の全データは、必ずディスク上にも存在します。ほとんどあり得ないイベントですが、RCD が失敗しても、データが失われることはありません。

Read Cache Device は、iSeries ハードウェアが置かれている店舗などで購入できます。あるいは、地域の IBM 担当員までご連絡ください。

Workload Estimator for iSeries

Workload Estimator は、特定のワークロード・タイプのワークロードを見積もることにより、それに基づいて行われるシステムの必要の判別を支援します。 PM/400 は、プロセッサーの保証を受けられるユーザーや IBM のメンテナンス契約に加入しているユーザーが、追加の課金なしで利用できる統合 OS/400 機能です。これを利用すると、システムの成長やパフォーマンスを計画および管理するのに便利な、キャパシティーとパフォーマンスの分析グラフが送られてきます。

Workload Estimator と PM/400 は、相互に連携するよう拡張されています。 Web ベース・アプリケーションを通して、既存のシステムの使用状況、パフォーマンスおよび PM/400 によって報告される成長に適合した、必要とされる iSeries システムへのアップグレードのサイジングが行えます。追加オプションとして、ドミノ、Java、および WebSphere などの特定のアプリケーションが追加されたキャパシティー、また、1 システム上で、複数の AS/400 や iSeries の従来の OS/400 を統合したワークロードのキャパシティーも含めたサイジングが行えます。このようなキャパシティーを含めておくなら、使用している独自のシステムから得られた既存の使用状況データに基づいて、将来のシステム要件を計画することが可能です。

iSeries ナビゲーター (ワイヤレス対応)

iSeries ナビゲーター (ワイヤレス対応) では、インターネット対応の電話、ワイヤレス・モデムを備えた個人用ディジタル・アシスタント (PDA)、または従来の Web ブラウザーを使用して、システムのパフォーマンスや状況をリモートにモニターできます。ワイヤレス装置では、以下のことを行えます。

- 複数システム間でのコマンドの実行
- システム、ジョブ、およびメッセージ・モニターの開始と表示
- モニターからのジョブやメッセージの処理 (保留、開放、終了、応答、詳細表示)
- 統合 xSeries サーバーの管理

iSeries ナビゲーター (ワイヤレス対応) がリモート・モニターを始めるのにどのように役立つかについては、トピック『iSeries ナビゲーター (ワイヤレス対応)』を参照してください。

リモート・モニターに関する完全で最新の情報は、 iSeries Navigator for Wireless ホーム・ページを参照してください。

PATROL for iSeries (AS/400) - Predict

PATROL for iSeries (AS/400) - Predict 製品は、多くのパフォーマンス管理タスクを自動化することにより、 iSeries のパフォーマンスの管理を支援します。この製品では、一群の iSeries サーバーに関する現行のシステム・データや過去のシステム・データを詳細に表示し、 CPU およびファイルの使用率や状況といった、特定の詳細情報をドリルダウンしていくことができます。加えて Patrol には、iSeries サーバーで生じるパフォーマンスと可用性の問題に対して未然に積極的な対応ができる、自動化のオプションもいくつか用意されています。

この製品は、詳細なキャパシティー・プランニング情報を示してくれるため、将来のアップグレードを計画したり、iSeries 環境の成長を管理したりするのに便利です。

詳細は、BMC 製品の Web サイト を参照してください。

シナリオ: パフォーマンス

パフォーマンス管理について学ぶ最も良い方法の 1 つは、サンプル・ビジネス環境でいくつのアプリケーションと機能が使用され得るかを示す例を見ることです。次のシナリオと構成例を考慮して、パフォーマンスの管理に関する理解を深めてください。

シナリオ:アップグレードまたはマイグレーション後にシステム・パフォーマンスを改善する
このシナリオは、システムをアップグレードまたは移行したところ、以前よりも実行速度が遅くなったように思える、というものです。このシナリオは、パフォーマンスの問題を識別し、修正するのに役立つでしょう。

シナリオ: システム・モニター
この例のシステム・モニターは、 CPU 稼働率が高すぎるために使用可能なリソースが増えるまで優先順位の低いジョブを一時的に保留する場合に、アラートを出します。

シナリオ: メッセージ・モニター
この例のメッセージ・モニターは、iSeries サーバー上で生じた、メッセージ待ち行列に関する照会メッセージを表示します。このモニターは、メッセージを検出すると即時にそのメッセージをオープンして表示します。

シナリオ: ジョブ・モニター
この例のジョブ・モニターは、指定されたジョブの CPU 稼働率を追跡し、 CPU 稼働率が高すぎる場合はそのジョブの所有者にアラートを出します。

関連情報

以下に、パフォーマンスのトピックと関連がある PDF 形式の iSeries 資料 (「ホワイト・ブック」と呼ばれる) および IBM Redbooks(TM) をリストします。以下の PDF のいずれかを表示または印刷することもできます。

- **マニュアル**

Performance Tools for iSeries
この資料では、システム、ジョブ、またはプログラム・パフォーマンスに関するデータを収集するために必要な情報をプログラマーに提供します。資料には、存在する可能性のある非効率性を識別して訂正するための、パフォーマンス・データの印刷と分析に関するヒントが載せられているほか、マネージャー機能やエージェント機能に関する情報が示されています。

- **Web サイト**

  iSeries Performance Capabilities Reference
  この解説書には、パフォーマンス・ベンチマークに役立つサーバー・パフォーマンス、キャパシティー・プランニング、およびサーバー・パフォーマンスの計画に関する高水準の技術情報を提供しています。

- **Redbooks:**

  - **IBM eserver iSeries Universal Connection for Electronic Support and Services**
    この資料は、ユニバーサル・コネクションについて紹介するものです。またこの中では、マシンのソフトウェアやハードウェアのインベントリーを IBM に報告するさまざまなサポート・ツールを使用して、システム・データに基づいた個別のエレクトロニック支援を受けられるようにする方法も説明されています。

  - **Lotus Domino for AS/400: Performance, Tuning, and Capacity Planning**
    この資料では、パフォーマンス管理のための方法論について説明します。この資料には、パフォーマンス目標の設定、パフォーマンス・データの収集と検討、資源の調整、およびキャパシティー・プランニングが含まれています。パフォーマンスに関する指針およびアプリケーション設計のヒントも提供されます。

  - **AS/400 Performance Management**
    この資料では、パフォーマンス管理のための方法論について説明します。この資料には、パフォーマンス目標の設定、パフォーマンス・データの収集と検討、資源の調整、およびキャパシティー・プランニングが含まれています。パフォーマンスに関する指針およびアプリケーション設計のヒントも提供されます。

  - **AS/400 HTTP Server Performance and Capacity Planning**
    インターネットおよび Web ブラウザー・ベースのアプリケーションは、どのように組織が情報を配布し、ビジネス・プロセスを実行し、顧客にサービスを提供し、新規のマーケットに到達するかについて多大の影響を与えています。この資料は、Web ベースのアプリケーションおよび情報システムの設計、開発、および拡張を担当する iSeries プログラマー、ネットワークおよびシステム管理の専門家、およびその他の情報技術者を対象としています。

  - **Java and WebSphere Performance on IBM eserver iSeries Servers**
    この資料では、Java を使用した作業を行うためのヒント、テクニック、および方法論を扱っているほか、 iSeries サーバーに関係した特定の視点から WebSphere Application Server のパフォーマンス関連の問題に触れます。

  - **Management Central: A Smart Way to Manage AS/400 Systems**
    マネージメント・セントラルの利点を紹介します。この資料の中では、パフォーマンス・モニターに代わるものとしての収集サービスについて扱います。また、この資料では、マネージメント・セントラルにより、オペレーターおよび管理者がリアルタイムのパフォーマンス・モニター機能 (イベント

の通知およびイベントへの自動応答など) を提供することによって、ネットワーク内でサーバーをモニターできるようにする方法についても説明します。

- **Managing AS/400 V4R4 with Operations Navigator** 
  この資料では、iSeries ナビゲーターとして知られるようになったオペレーション・ナビゲーターを介して使用できる、広範な iSeries 機能に関する洞察が得られます。これらの機能に関する完全な説明に加えて、とりわけマネージメント・セントラルと収集サービスを使用したシステム・パフォーマンスのモニターの説明が載せられています。

- **AS/400 Performance Explorer Tips and Techniques** 
  この資料は、V3R6 で使用できた Performance Explorer 機能に関する説明と詳細な例を提供します。この中には、特定のアプリケーションの例やレポートが含まれています。

iSeries のパフォーマンスに関する完全な情報を確認していただくため、必ず『パフォーマンス』のトピックを参照してください。

Printed in Japan

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12