IDUG
The Worldwide DB2 User Community

IDUG DB2 Tech Conference
Denver, Colorado USA | May 2012

Recommended Best Practices from Customer DB2 Health Check Studies

John Campbell & Florence Dubois
IBM DB2 for z/OS Development

One-Day Seminar
Monday 14 May, 2012 | Platform: DB2 for z/OS

Disclaimer

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS AND/OR SOFTWARE

Objectives

- Introduce and discuss the issues commonly found
- Share the experience from customer health check studies
- Share the experience from live customer production incidents
- Provide proven recommended best practice
- Encourage proactive behaviour

Agenda

- Part I - Continuous availability
  - Continuous availability vs. Fast failover
  - Coupling Facility structure duplexing
  - Automated restart
  - Multiple DB2 data sharing groups
  - Application considerations
- Part II - Special considerations for WLM, CICS and DDF
  - WLM policy settings
  - Balanced design across CICS and DB2
  - Distributed workload and DDF

Agenda …

- Part III - Speed of recovery
  - Disaster recovery
  - Mass data recovery
- Part IV - Operations
  - Preventative software maintenance
  - DB2 virtual and real storage
  - Performance and exception-based monitoring

Part I – Continuous Availability

Continuous Availability vs. Fast Failover

- Common problems
  - Frequent disconnect between IT and business
    - Lack of transparency about what IT can actually deliver
    - Lack of IT understanding of the real business needs and expectations
    - Fuzziness between high availability with fast failover vs. continuous availability
      - Fast failover = restart the failed component as fast as possible
        - Still an application service outage!
      - 'True' continuous availability = avoid or mask both planned and unplanned outages by automatically routing work around the failed component
  - Big investment in z/OS Parallel Sysplex and DB2 for z/OS data sharing, but still cannot achieve 'true' continuous availability
    - 2-way active-passive failover model
    - System SPOFs and application affinities
    - Critical business applications not sysplex-enabled
    - Static workload routing and/or turning off workload distribution

Continuous Availability vs. Fast Failover …

- *Problem: Frequent disconnect between IT and business*
- Recommendation: Establish a list of mission-critical applications (including application inter-dependencies)
  - Ruthlessly prioritise the list based on business importance
  - Understand the true business requirements for application services
    - How aggressive is the availability requirement?
    - What is good enough?
    - High availability with fast failover within x time vs. true continuous availability
  - Be honest and transparent about what can actually be achieved based on the applications and the infrastructure
  - Put a program in place to fill the gaps
    - Or change the expectations…

Continuous Availability vs. Fast Failover …

- *Problem: Big investment in z/OS Parallel Sysplex and DB2 for z/OS data sharing, but still cannot achieve 'true' continuous availability*
- z/OS Parallel Sysplex and DB2 Data Sharing are the 'Gold Standard' in terms of continuous availability
  - Provide the base infrastructure to build upon
- But additional ingredients are required
  - Active read-write data sharing
  - Application redundancy/cloning
  - Fine-grained, dynamic transaction routing (DTR)
  - CF structure duplexing
  - Automated restart

Continuous Availability vs. Fast Failover …

- Trade-off between cost of data sharing vs. continuous availability
- DB2 for z/OS data sharing was designed as a 'pay-as-you-go' system
  - You only pay the cost associated with global locking and GBP dependent protocols when there is inter-system read-write interest
- Some data sharing customers try to avoid inter-system read-write interest altogether to minimise cost …
  - 2-way active-passive failover or spill-over model
  - Affinity routing of application workload to a single DB2 member
- … but as a result, they aggravate
  - Availability: Retained x-mode pageset p-locks will block application failover
    - Cannot be negotiated away until end of the forward log recovery during DB2 warm restart of the failed DB2 member (at the earliest)
  - Potential scalability issues if an application workload is bound to a single DB2 member (e.g. virtual storage, DB2 internal latch contention)

Continuous Availability vs. Fast Failover ...

- Additional risks of avoiding active data sharing
  - Inter-DB2 read-write interest cannot always be prevented
    - Planned failover
    - Batch processing running late into the peak online period
    - Occasional spill onto the second member to absorb workload peaks
  - Latent issues are likely to be discovered at the worst possible time and potentially affect critical business services
    - System not configured for active data sharing (e.g. Shared CFs, CF structures sizing)
    - Badly behaved applications
- Should plan for active data sharing, and not try to avoid it
  - Extensive positive real world customer experience with active data sharing
  - Considerable progress has been made by DB2 and z/OS to reduce CPU resource consumption and lock contention
    - MEMBER CLUSTER, TRACKMOD NO, larger data and index page size, DPSIs, asymmetric leaf page split, etc.

Continuous Availability vs. Fast Failover ...

- General recommendations
  - Minimum of 4-way active single site data sharing across two CPCs (boxes) is the right direction in most cases for
    - True continuous availability
    - Staying away from single image constraint
  - All critical business applications should be sysplex-enabled and deployed for redundancy (cloned)
  - Dynamic fine-grained workload balancing to route around failure and balance the workload
    - CICSPlex System Manager
    - Distributed connections: DVIPA, System Distributor, workload balancing and automatic client reroute
  - Symmetry of group brings operational simplicity, makes it easy to scale out horizontally, and easier to roll software maintenance

Coupling Facility Structure Duplexing

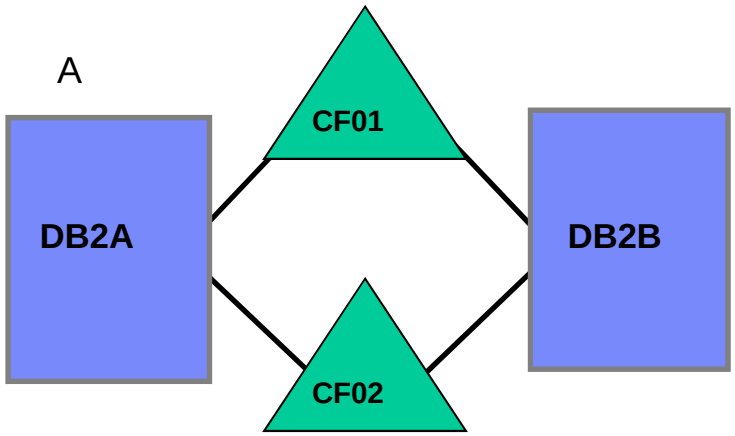
- Common problems
  - No duplexing of the group buffer pools
    - Can result in long outage times in case of a CF failure
      - GRECP/LPL recovery is required
  - No failure-isolated CF for LOCK1 and SCA structures, and no use of system-managed CF structure duplexing
    - A failure of the 'wrong CF' will result in a group-wide outage
      - LOCK1 or SCA can be dynamically rebuilt only if ALL the DB2 members survive the failure

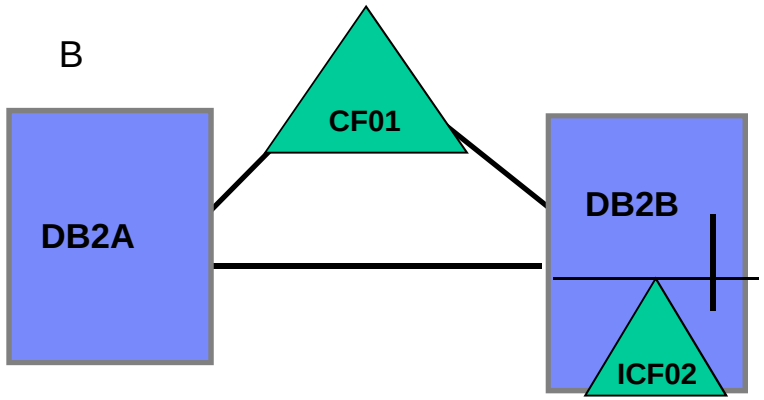
Coupling Facility Structure Duplexing

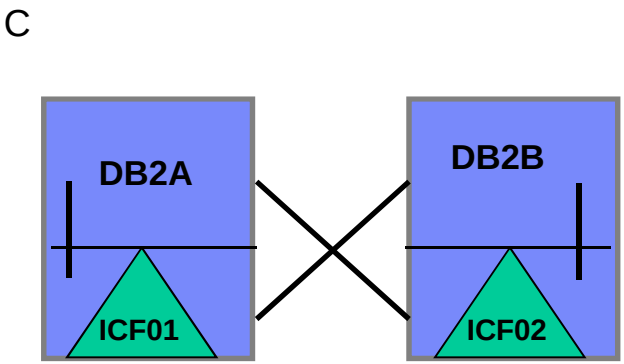
- Recommendations
  - Enable DB2-managed duplexing for the GBPs
    - Small performance overhead
      - Only changed pages are written to the secondary GBP structure
      - Async request to the secondary GBP structure overlaps with the sync request to the primary GBP structure (which may itself be converted to async)
        - But both requests need to complete successfully before the processing can continue
        - Need to recognise that defining remote GBP secondary structures over long distances may impact the throughput of batch programs running against GBP-dependent objects
  - Understand the trade-offs between system-managed CF structure duplexing vs. use of failure-isolated CF and very fast structure rebuild into an alternate CF
    - Option 1 (preferred): LOCK1 and SCA are failure-isolated from the DB2 members
    - Option 2: Use system-managed CF structure duplexing for LOCK1 and SCA
      - Performance overhead for LOCK1 requests
      - Should only be used over very short distances and where CF access intensity is low
        - Objective: keep performance overhead and latency low

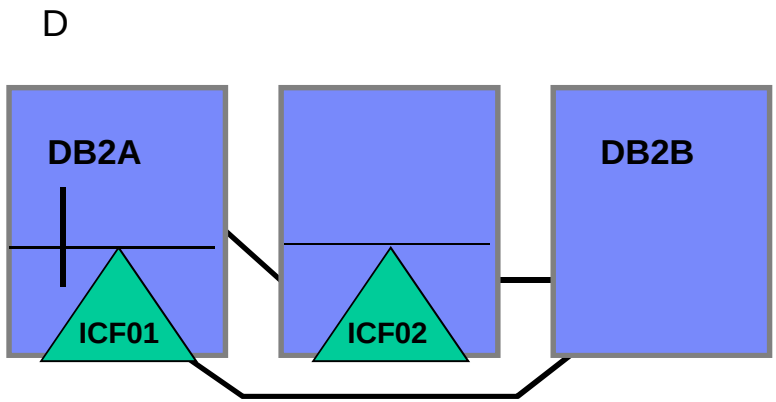
Coupling Facility Structure Duplexing …

- Parallel Sysplex configurations



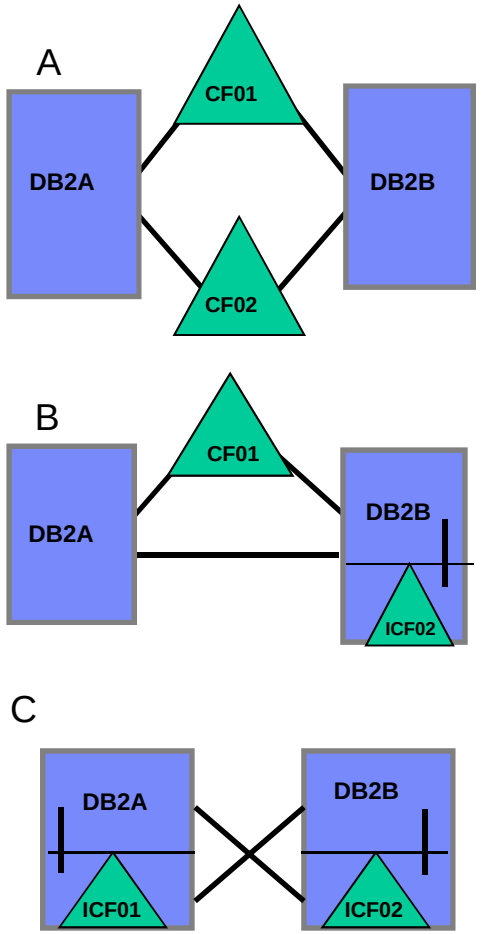






Coupling Facility Structure Duplexing …

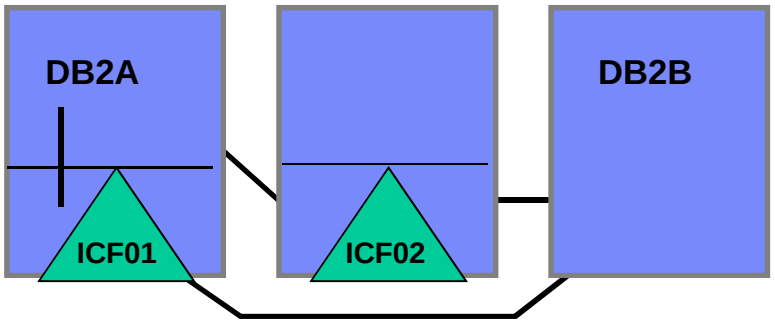
- Parallel Sysplex configurations …
  - A: 'Traditional' configuration
    - LOCK1 and SCA in one CF
    - Duplexed GBPs spread across both CFs
      - Primary and secondary GBPs balanced based on load
  - B: One Integrated CF (ICF), one external CF
    - LOCK1 and SCA in external CF
    - Duplexed GBPs spread across both CFs
      - Primary GBP in ICF has advantages for 'local' DB2
  - C: Two ICF configuration
    - LOCK1 and SCA duplexed; allocated in both CFs
      - 'System-managed' duplexing
      - Performance implication for LOCK1 requests
    - Duplexed GBPs spread across both CFs



Coupling Facility Structure Duplexing …

- Parallel Sysplex configurations
  - D: Two ICF configuration – similar to C but…
    - LOCK1 and SCA duplexed if allocated in ICF01
      - 'System-managed' duplexing
      - Performance implication for LOCK1 requests
    - LOCK1 and SCA do not have to be duplexed if allocated in ICF02
      - Be aware of planned maintenance when ICF02 needs to come down and LOCK1 and SCA will need to live in ICF01 for a small window (or if DB2A/DB2B ever moves to CEC where ICF02 is)
    - Duplexed GBPs spread across both CFs

D



Automated Restart

- Common problems
  - Manual intervention is required to restart a failed DB2 member
    - Failed DB2 member probably holds retained locks
    - Other DB2 members cannot access data protected by retained locks
    - Can result in a long outage, especially if a management decision is involved
  - In case of an LPAR outage, need to wait for re-IPL of z/OS LPAR to restart DB2
    - Can result in an even longer outage
  - Overuse of Restart Light
  - Common misconception is that Restart Light is ‘Restart Fast’ or ‘Backout Fast’
  - No tuning for fast DB2 restart

Automated Restart …

- Recommendations
  - Use automation to detect failures quickly and restart DB2 immediately
    - Objective: Release the retained locks as soon as possible
    - NO manual intervention
    - NO management decision involved
    - Automated Restart Manager (ARM) or automated operators (e.g. Tivoli System Automation family) can be used
  - Two failure scenarios to distinguish
    - DB2 or IRLM failure: restart DB2 in place using a normal DB2 warm restart
    - LPAR or CEC failure: restart DB2 on an alternative LPAR using DB2 Restart Light

```
** Very basic ARM Policy **
RESTART_GROUP(DB2)
          ELEMENT(DSNDB10DB1A)
          TERMTYPE(ALLTERM)
          RESTART_METHOD(ELEMTERM,PERSIST)
          RESTART_METHOD(SYSTERM,STC,'-DB1A STA DB2,LIGHT(YES)')
```

Automated Restart …

- Recommendations …
  - Design point of Restart Light is ONLY for cross-system restart (LPAR/CEC failure)
    - Avoid ECSA/CSA shortage on the alternative LPAR
    - Avoid real storage shortage on the alternative LPAR
  - Restart Light is likely to be slower than normal DB2 crash restart
  - Restart Light will not clean up all retained locks
    - Will NOT free up retained IX/SIX mode pageset p-locks
      - No SQL processing will be blocked
      - No DB2 utility that is a claimer will be blocked e.g., COPY SHRLEVEL(CHANGE) will work
      - But drainers will be blocked e.g, REORG including REORG SHRLEVEL(CHANGE)
  - Restart Light and in-doubt URs
    - By default, DB2 will not terminate automatically until all in-doubt URs are resolved
    - With option LIGHT(NOINDOUBT), DB2 will terminate and not resolve in-doubts
    - But keep in mind that retained locks cannot be freed until the in-doubt UR is resolved

Automated Restart …

- Tuning for fast DB2 restart
  - Take frequent system checkpoints
    - Every 2-5 minutes generally works well
  - Commit frequently!
  - Use DB2 Consistent restart (Postponed Abort) to limit backout of long-running URs
    - Controlled via ZPARM for normal DB2 warm restart
      - LBACKOUT=AUTO
      - BACKODUR=5 (interval is 500K log records if time-based checkpoint frequency)
    - New LBACKOUT options to allow Restart Light with postponed abort
      - See PM34474 (V9) and PM37675 (V10)
    - Postponed Abort is not a 'get-out-of-jail-free' card
      - Some retained locks persist through restart when postponed backout processing is active
        - Retained locks held on page sets for which backout work has not been completed
        - Retained locks held on tables, pages, rows, or LOBs of those table spaces or partitions

Automated Restart …

- Tuning for fast DB2 restart …
  - Track and eliminate long-running URs
    - Long-running URs can have a big impact on overall system performance and availability
      - Elongated DB2 restart and recovery times
      - Reduced availability due to retained locks held for a long time (data sharing)
      - Potential long rollback times in case of application abends
      - Lock contention due to extended lock duration >> timeouts for other applications
      - Ineffective log avoidance (data sharing)
      - Problems getting DB2 utilities executed
    - DB2 provides two system parameters to help identify ‘rogue’ applications
      - URCHKTH for long running URs
      - URLGWTH for heavy updaters updates that do not commit

Automated Restart …

- Tuning for fast DB2 restart …
  - Track and eliminate long-running URs …
    - Aggressively monitor long-running URs
      - Start conservatively and adjust the ZPARM values downwards progressively
      - Initial recommendations
        - URLGWTH = 10 (K log records)
        - URCHKTH = 5 (system checkpoints) – based on a 3-minute checkpoint interval
      - Automatically capture warning messages
        - DSNJ031I (URLGWTH)
        - DSNR035I (URCHKTH)
        - and/or post process IFCID 0313 records (if Statistics Class 3 is on)
    - Get ‘rogue’ applications upgraded so that they commit more frequently
      - Need management ownership and process

Multiple DB2 Data Sharing Groups

- Introduced for technical and political reasons, claiming
  - Improved availability
  - Simplified operations
  - Application isolation
  - Perceived architectural limits
- Common problems
  - Data Sharing Groups are not completely independent
  - Synchronous remote SQL calls over DRDA inter-connected groups
  - Resulting in a more complex environment
  - Major cause of slow-downs and sympathy sickness
  - ‘Nightmare’ for problem determination and problem source identification

Multiple DB2 Data Sharing Groups ...

- Strong recommendations
  - Each data sharing group should be loosely coupled from the others
  - Eliminate synchronous remote SQL calls
  - Implement low-latency data replication to provide loose coupling and independence
  - Connect directly to the target group and avoid ‘hopping’
- Other points
  - Near low-latency data replication is a huge price to pay for independence of data sharing groups
  - If the main data sharing group was managed better, there would be no need to break it into multiple independent data sharing groups
  - DB2 10 provides significant DBM1 31-bit virtual storage constraint relief and allows much greater vertical scalability of individual DB2 subsystems
    - Opens up possibility to consolidate LPARs and DB2 members and/or
    - Enhances scaling of the main data sharing group

Application Considerations

- Common problems
  - Critical business applications not sysplex-enabled
  - Single points of serialisation
    - Scalability constraint
      - Will probably cause a locking ‘hot spot’ or ‘choke points’ due to increased activity
    - Single point of failure when covered by retained lock held by failed DB2 member
  - High cost of data sharing
  - Workloads not scaling
  - Late running batch impacting online day

Application Considerations …

- Recommendations for applications requiring continuous availability
  - Remove all transaction or system affinities
    - Inter-transaction affinity
      - Do not assume that subsequent executions of a transaction will run on the same system/image as the first transaction
      - Do not assume that the order of arrival will be the order of execution for transactions (work units)
    - System/Image affinity
      - Logical by LPAR or DB2 member e.g., to reduce MLC software cost
      - Physical e.g., hardware resource (crypto device, special printer)
  - Clone/replicate application across DB2 members to provide redundancy
  - Avoid single points of control and serialisation -> blocking retained locks
    - Single row tables with update
    - Lock escalation
    - SQL LOCK TABLE
    - SQL DELETE without a WHERE CLAUSE

Application Considerations …

- Use a light-weight locking protocol (isolation level) and exploit lock avoidance
  - Benefits of lock avoidance
    - Increased concurrency by reducing lock contention
    - Decreased lock and unlock activity and associated CPU resource consumption
    - In data sharing, decreased number of CF requests and associated CPU overhead
  - Use ISOLATION(CS) CURRENTDATA(NO) or use ISOLATION(UR)
  - Define cursors with their intended use (e.g., FOR UPDATE OF|READ ONLY)
  - Use a timestamp column on each table
    - To record the last update
    - Use as predicate on searched UPDATE (or DELETE) to enforce data currency
  - Commit frequently to improve the effectiveness of global lock avoidance
  - How to monitor lock avoidance?
    - General ROT: Unlock requests per commit should be less than 5

Application Considerations …

- Reduce lock contention
  - Use LOCKSIZE PAGE as design default
  - Only use LOCKSIZE ROW where appropriate and beneficial
    - Consider LOCKSIZE PAGE MAXROWS 1 as an alternative to LOCKSIZE ROW
  - Commit frequently
  - Avoid use of SQL LOCK TABLE
  - Avoid any dependence on lock escalation
  - Issue CLOSE CURSOR as soon as possible
  - Close open CURSOR WITH HOLD ahead of commit
  - Implement partitioning to potentially reduce inter-DB2 lock contention
  - Access data rows and tables in consistent order
    - Helps reduce the possibility of deadlocks

Application Considerations …

- Sequentially-assigned dumb number keys
  - User-defined solution can compromise performance scalability and continuous availability e.g., single row table used as a ‘counter’
  - Use IDENTITY column or pull value from SEQUENCE object
    - Superior in performance/availability than user defined especially with caching
    - Special non-modify P-lock to serialise update to the SYSSQUENCES page across DB2s
      - DB2 forces the updated page to the GBP prior to releasing the P-lock
      - If DB2 fails in the middle, the P-lock is not retained
    - SQL DDL changes for IDENTITY column definition or SEQUENCE object
      - GENERATED ALWAYS | BY DEFAULT
      - AS IDENTITY
      - START WITH numeric-constant
      - INCREMENT BY numeric-constant
      - NO CACHE | CACHE integer
    - CACHE can cause non-sequential value assignment
    - Not used 'cached' values will be lost after DB2 termination
  - GENERATE_UNIQUE() scalar function to generate sysplex unique key

Application Considerations …

- High-volume concurrent insert
  - Selective use of MEMBER CLUSTER
    - Reduces page p-lock and page latch contention especially when using APPEND mode or when 'insert at end' of pageset/partition
      - Space map pages
      - Data pages when using LOCKSIZE ROW
    - Each member inserts into its own set of pages
      - Space maps and associated data pages
    - Sacrifice clustering of data rows
      - REORG required if data row clustering is important
        - Use RTS SYSINDEXSPACESTATS.REORGCLUSTERSENS column in DB2 V10 to determine if queries are clustering-sensitive
    - Available for UTS PBG and PBR in DB2 V10
      - Still not allowed for segmented
    - Augment with use of TRACKMOD NO
      - When no use of Incremental COPY or performance of Incremental COPY not important

Application Considerations …

- High-volume concurrent insert …
  - Reduce index overhead
    - Keep the number of indexes on a table to a minimum to reduce data sharing overhead
      - Detect use of unused indexes so they can be dropped
        - Use RTS SYSINDEXSPACESTATS.LASTUSED column in DB2 V9
      - Consider dropping low value or low use indexes
    - Reduce painful index leaf page splits
      - Implement data and index partitioning
      - V9 NFM enhancements
        - Larger index page size option
          - Consider especially for sequential key insert pattern
          - Can be used without index compression
        - Asymmetric index split depending on insert pattern

Application Considerations …

- High-volume concurrent insert …
  - Logging considerations
    - In some extreme cases, logging bandwidth can be a constraint
    - Consult DB2 Statistics Trace
      - Physical writes
      - LC19 latch contention rates
    - V9 NFM and V10 NFM provide significant relief in most cases for LC19 latch contention
      - Significantly reduce LRSN ‘spinning’
      - More of a consideration when running on newer faster CPUs
    - I/O bandwidth bottlenecks can be relieved
      - Use of table compression
      - Dataset striping
      - Faster devices

Application Considerations …

- High-volume concurrent insert …
  - Affinity routing
    - A given partition is inserted to by only one DB2 member
    - Only consider for special circumstances
    - Can improve batch performance and reduce global lock contention, but …
      - May compromise the benefits of data sharing
        - Operational simplicity, continuous availability, easy scale-out
    - DPSIs can be useful with this technique
      - Note carefully
        - DPSIs can only be unique within partition
        - Good query performance may be compromised if no WHERE predicates on partitioning key columns

Application Considerations …

- Batch processing
  - Design for ‘parallel batch’
    - Avoid serial processing, full or part
    - Saturate available processor and sysplex capacity
    - Determine how many parallel streams are needed to meet elapsed time requirement
      - Determine how many partitions required
    - Sort input records to drive dynamic sequential prefetch
      - Avoid random I/O by exploiting data row clustering
  - Intermediate commit/restart function in batch should be mandatory
    - Flexible criteria based on CPU consumption (no. of calls) and/or elapsed time
    - Criteria externalized in DB2 table and should be easy to change
    - New criteria should be picked up ‘on the fly’ by the application
    - Intelligent commit point processing based on prevailing operating conditions
    - Definite requirement to allow batch to run into the ‘online day’
    - Evaluate and acquire a robust reliable product

Part II – Special Considerations for WLM, CICS and DDF

WLM Policy Setup

- Common problems
  - Systems are run at very high CPU utilisation for elongated periods of time
  - Little or no non-DB2 work to pre-empt when the system becomes 100% busy i.e., no non-DB2 work that can be sacrificed
  - Sporadic slowdowns or hangs of a DB2 subsystem or an entire DB2 data sharing group as a result of poorly managed WLM policy settings
    - DB2 applications workloads are allowed to ‘fall asleep’
      - If a thread is starved while holding a major DB2 internal latch or other vital resource, this can cause an entire DB2 subsystem or data sharing group to stall
        - Especially if other work running on the LPAR has latent demand for CPU
    - DB2 system address spaces are not protected from being pre-empted under severe load
      - Any delays in critical system tasks as a result of a DB2 system address space being pre-empted can lead to slowdowns and potential ‘sympathy sickness’ across the data sharing group
        - See next slide

WLM Policy Setup

- Why protect DB2 system address spaces from being pre-empted?
  - Answer: These address spaces are critical for efficient system operation and should be defined with an aggressive goal and a very high importance
    - MSTR contains the DB2 system monitor task
      - Requires an aggressive WLM goal so it can monitor CPU stalls and virtual storage constraints
    - DBM1 manages DB2 threads and is critical for both local DB2 latch and cross-system locking negotiation
      - Any delay in negotiating a critical system or application resource (e.g. P-lock on a space map page) can lead to a slowdown of the whole DB2 data sharing group
    - DIST and WLM-managed stored procedure AS only run the DB2 service tasks i.e. work performed for DB2 not attributable to a single user
      - Classification of incoming workload, scheduling of external stored procedures, etc.
      - Typically means these address spaces place a minimal CPU load on the system
        - BUT… they do require minimal CPU delay to ensure good system wide performance
      - The higher CPU demands to run DDF and/or SP workloads are controlled by the WLM service class definitions for the DDF enclave workloads or the other workloads calling the SP
        - Clear separation between DB2 services which are long-running started tasks used to manage the environment and transaction workloads that run and use DB2 services

WLM Policy Setup ...

- Recommendation
  - Configure the WLM policy defensively to deal with situations when the system is over-committed
- General guidelines
  - VTAM, IRLM, and RRS must be mapped to service class SYSSTC
  - All DB2 system address spaces should be isolated into a unique user-defined service class defined with Importance 1 and a very high velocity goal (e.g. 85-90)
    - MSTR, DBM1, DIST, WLM-managed stored procedure address spaces
    - As of V9, recommendation was to put DB2 MSTR into SYSSTC
      - When z/OS WLM Blocked Workload Support is enabled (see later slides), Imp1 with a very high velocity goal is generally sufficient for MSTR
    - Do not generally recommend putting DB2 DBM1 into SYSSTC because of risk of DB2 misclassifying incoming work
      - Special case: If experiencing recurring DB2 slowdowns when running at very high CPU utilisation for elongated periods of time, move all DB2 system AS (MSTR, DBM1, DIST, WLM-managed store procedure AS) to service class SYSSTC to help with PD/PSI

WLM Policy Setup ...

- General guidelines ...
  - No DB2 application workload should run higher than the DB2 system AS
  - Try to have non-DB2 work ready to pre-empt
  - Remove any discretionary definitions for any DB2-related work
  - Do not use Resource Group Capping e.g., for batch
  - Let WLM decide on how to best manage mixed workloads
  - For CICS and IMS, use 80th or 90th percentile response time goals as transactions are typically non-uniform
    - The transaction response time goals must be practically achievable
    - Use RMF Workload Activity Report to validate
  - Online workloads that share DB2 objects should have similar WLM performance goals to prevent interference slowdowns
    - Example: An online DDF enclave-based workload classified much lower in importance than an online CICS workload
      - If they share any of the DB2 objects, a low running DDF enclave may end up causing the CICS online workload to become stalled

WLM Policy Setup ...

- General guidelines ...
  - Processing capacity should be planned at no more than 92-95% average at peak
    - Especially if there is little non-DB2 work that can be sacrificed at high CPU utilisation
    - Risks of running at very high system utilisation over long periods of time
      - Far more likely to expose latent stress-related software defects and/or user setup problems
      - Transaction path length increases at very high system utilisation
        - With a corresponding increase in CPU resource consumption
      - Application locks as well as DB2 internal locks and latches will be held for longer periods
        - Will increase lock/latch contention, which will behave in a non-linear fashion
        - Will lead to greatly extended In-DB2 transit time (response time)
  - Extra processing capacity activated using IBM Capacity on Demand offering should be turned on ahead of anticipated peaks – not when hitting the wall at 100%
    - Otherwise the extra processing is used to manage contention and not net new workload
    - Always need some spare capacity for automated and service recovery actions
  - Enable defensive mechanisms to help with stalled workloads
    - -DISPLAY THREAD(*) SERVICE(WAIT) command
    - z/OS WLM Blocked Workload Support

-DIS THREAD(*) SERVICE(WAIT)

- Identifies allied agents and distributed DBATs that have been suspended for more than x seconds
  - x = MAX(60, 2x IRLM resource timeout interval)
- If the thread is suspended due to IRLM resource contention or DB2 latch contention, additional information is displayed to help identify the problem

```
19:23:01.31  STC42353   DSNV401I  -DT45 DISPLAY THREAD REPORT FOLLOWS -
                        DSNV402I  -DT45 ACTIVE THREADS -
                        NAME     ST A   REQ ID           AUTHID   PLAN     ASID TOKEN
                        CT45A    T  * 12635 ENTRU1030014 ADMF010  REPN603  00C6 90847
                        V490-SUSPENDED 04061-19:20:04.62 DSNTLSUS +00000546 14.21
```

- Note that DISTSERV may return false positives for DBATs that have not been in use for more than the x seconds

-DIS THREAD(*) SERVICE(WAIT) …

- Will also attempt to dynamically boost priority for any latch holder that appears to be stuck
  - Targeted boost via WLM services
- Can run with SCOPE(GROUP)
  - Output piped back to the originating member via notify message
- Recommendations
  - V8 – Strongly recommend to run at one-minute interval through automation
  - V9/V10 – Driven at one-minute interval by the internal DB2 System Monitor
    - But the diagnostic information is not written out
    - Still recommend to issue the command via automation on a one-minute interval and save away the output as diagnostics
    - See also DSNV507I message on -DISPLAY THREAD(*) TYPE(SYSTEM) output

```
V507-ACTIVE MONITOR, INTERVALS=1235, STG=47%, BOOSTS=0, HEALTH=100
      REGION=1633M, AVAIL=1521M, CUSHION=375M
```

z/OS WLM Blocked Workload Support

- Allows small amounts of CPU to be leaked onto stalled dispatchable units of work on the system ready queue
  - Not specific to DB2 workloads
  - Allows even frozen discretionary jobs to get some small amount of CPU
  - But will not help jobs that are stalled because of resource group capping
- Controlled by two parameters in IEAOPTxx parmlib member
  - BLWLINTHD – Threshold time interval for which a blocked address space or enclave must wait before being considered for promotion
    - Default: 20 seconds
  - BLWLTRPCT – How much of the CPU capacity on the LPAR is to be used to promote blocked workloads
    - Default: 5 (i.e. 0.5%)
  - For more information, see APARs OA17735, OA22443, and techdoc FLASH10609
    - http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/FLASH10609

z/OS WLM Blocked Workload Support …

- Recommendations
  - All customers should run with this function ENABLED and reasonably aggressive
    - In a CPU-saturated environment where DB2 application workloads are running with low dispatching priority, it may prove very beneficial to be more aggressive than the defaults
    - Changing BLWLINTHD to 3-5 seconds may provide better overall system throughput at very high CPU utilisation
  - Regularly review the statistics on this function provided in RMF CPU Activity and Workload Activity reports

Balanced Design across CICS and DB2

- Common problems
  - Imbalance across configuration of CICS and DB2, lack of fixed topology across CICS and DB2, and CICS transaction affinities
    - Sporadic unpredictable response times
    - Complex debug of performance issues
    - Single points of failure
    - Increased risk of DB2 crashes
  - Lack of strategy for the resolution of in-doubt threads
  - Poor thread reuse
  - Over-use of protected ENTRY threads especially with RELEASE(DEALLOCATE)
  - Misuse of TCB Priority on DB2ENTRY
  - Missing performance improvement opportunities

Balanced Design across CICS and DB2 …

- Support for Group Attach in CICS
  - Parameter DB2GROUPID on DB2CONN
    - Mutually exclusive to DB2ID parameter
    - CICS will connect to any member of the group
  - May simplify operational procedures BUT …
    - Can lead to sympathy sickness when CICS AORs are moved away to different LPARs
      - Risk of DB2 member overload with excessive queuing if CTHREAD reached
        - Can cause unpredictable response times and wildly varying response times which can lead to severe z/OS Sysplex-wide issues
      - Increased risk of DB2 crashes
    - Significantly complicates DBM1 31-bit virtual storage and DB2 real storage planning
    - Significantly complicates load balancing across DB2 members of group
  - Fixes a simple problem by creating a complex problem
    - Complexity of problem would be even worse if multiple DB2 members of the same data sharing group were introduced on the same LPAR

Balanced Design across CICS and DB2 …

- Recommendation: Implement a fixed CICS/DB2 landscape topology
  - Do not use the support for Group Attach in CICS
  - Configure number of CICS AORs based on LPAR size
    - In non-THREADSAFE environment, configure one CICS per defined CP (processor) since CICS QR TCB is bottlenecked on a single processor
  - Configure DB2 CTHREAD value based on static CICS environment to allow for better balancing of DB2 threads and better DBM1 31-bit virtual storage usage
  - Configure ARM policy so that CICS+DB2 are allowed to restart as a unit
    - Allow in-doubt URs to be automatically processed and resolved
    - Data integrity can be maintained
    - See next slide…

Balanced Design across CICS and DB2 …

- Resolution of in-doubt URs with DB2 restart light
  - If you do not specify LIGHT(NOINDOUBT) option on DB2 start – default since V8
    - If there are in-doubts URs then DB2 will wait for CICS to reconnect
    - If CICS reconnects to DB2, DB2 informs CICS it is a restart-light subsystem
    - CICS-DB2 Attach connection status remains as 'connecting‘
      - Stops any new work accessing DB2
    - CICS allows resynchronisation tasks (CRSY) to access DB2
    - DB2 automatically terminates when resync is complete
  - If you specify the LIGHT(NOINDOUBT) option on DB2 start
    - DB2 subsystem will shut down and not resolve in-doubts
    - If you abort all the in-doubts, then, when CICS is warm started, CICS will try to re-sync for in-doubts and there will be error messages

CICS/DB2 – Thread Reuse

- Thread Reuse
  - Opportunity to provide performance boost
  - Benefits
    - Lower CPU resource consumption
    - Faster transaction throughput
    - Fewer DB2 threads for the same throughput
    - Delays the need for additional DB2 members
  - Techniques for CICS
    - Protected ENTRY threads
      - PROTECTNUM>0 on DB2ENTRY
    - Tailgating on ‘unprotected’ ENTRY threads or on POOL threads
      - THREADWAIT=YES on DB2ENTRY or DB2CONN for pool entries
    - Conditions for reuse: both tasks must use the same DB2 plan

CICS/DB2 – Thread Reuse …

- Recommendations
  - Encourage thread reuse for high-volume simple online transactions
  - Configure protected threads based on arrival rate – not on a binary decision
  - Number of protected ENTRY threads should be balanced against the affordable CTHREAD budget of the DB2 subsystem
    - Difficult to do when CICS AORs are floating across LPARs…
  - Reuse of unprotected ENTRY (and POOL) threads using natural arrival rate can provide a better alternative
    - Remove queuing for threads due to reaching CTHREAD
    - Achieve more even transaction response times
    - Lower CPU cost per transaction at higher CPU utilization
    - Achieve better use of DBM1 ASID 31-bit virtual storage due to threads going through de-allocation on a regular basis
  - Use of single plan for CICS would allow for maximum thread reuse in the POOL

CICS/DB2 – Thread Reuse …

- Recommendations …
  - Consider collapsing multiple transactions onto a single ENTRY thread to drive up thread reuse
  - Avoid use of CICS Dynamic Plan Selection Exit and plan switching
    - Makes thread reuse difficult to achieve
  - How to measure the cost of thread create and terminate, or avoidance thereof?
    - Not captured in DB2 Accounting Class 2 TCB time
    - Cost of thread create and terminate will clock against the L8 TCB and will be in the CICS SMF type 110 record
      - With OTE, CICS uses the L8 TCB to access DB2 no matter if the application is threadsafe or not
      - Note: prior to OTE, it was uncaptured time
  - Use thread storage contraction (CONTSTOR=YES)

CICS/DB2 – Thread Reuse …

- BIND option RELEASE(DEALLOCATE)
  - Potential for even more savings in terms of CPU resource consumption
    - Avoiding the repetitive cost per transaction of allocating and freeing private/stack storage, EDM storage, parent L-locks, etc.
  - Investigate selective use of BIND option RELEASE(DEALLOCATE) for high-volume packages within high-volume transactions achieving thread reuse
  - BUT…
    - Trade off with BIND/REBIND and DDL concurrency
    - Watch out for increased virtual and real storage consumption
    - Some locks are held beyond commit until thread termination
      - Mass delete locks (SQL DELETE without WHERE clause)
      - Gross level lock acquired on behalf of a SQL LOCK TABLE
      - Note: no longer a problem for gross level lock acquired by lock escalation

CICS/DB2 – Thread Reuse …

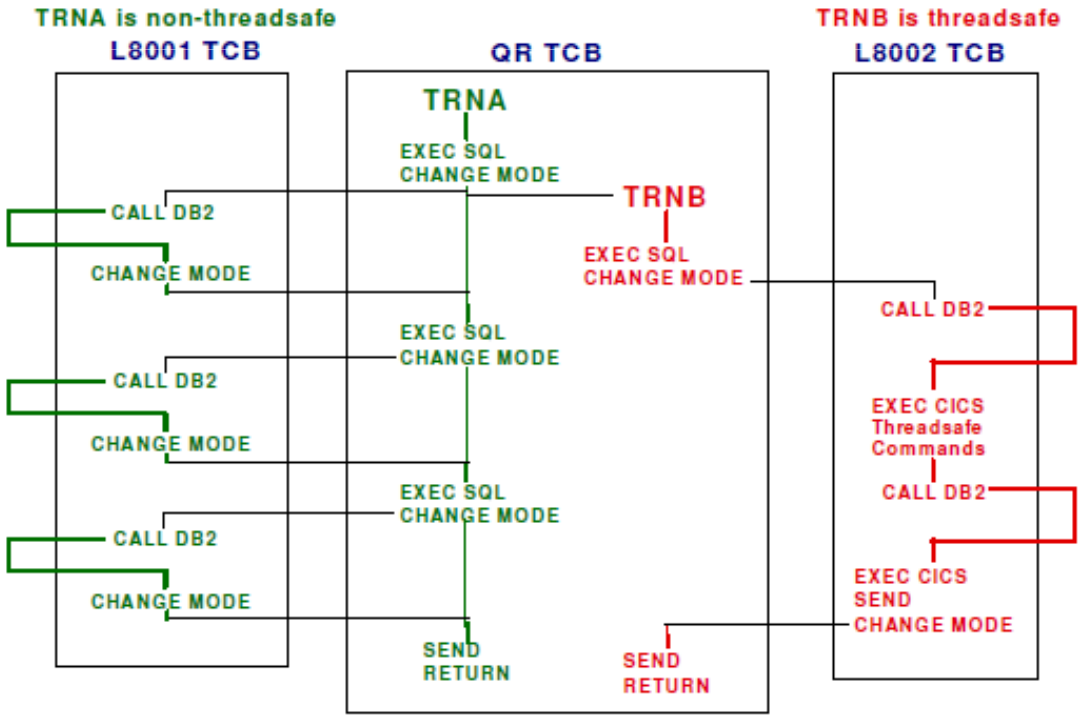
- BIND option RELEASE(DEALLOCATE) …
  - Note:
    - Long-running, performance-critical batch processes that commit frequently can also benefit from RELEASE(DEALLOCATE)
      - Keep generated xPROCs for fast column processing across commit
      - Sustain dynamic prefetch and index look-aside across commit
  - How to measure the benefit of RELEASE(DEALLOCATE)?
    - Captured in DB2 Accounting Class 2 TCB time

CICS/DB2 – TCB Priority

- TCB Priority
  - Parameter PRIORITY (HIGH|EQUAL|LOW) on DB2ENTRY
    - Specifies the MVS dispatching priory of the pool thread subtask related to the CICS main task (QR TCB)
  - HIGH should always be the preferred configuration default
    - With PRIORITY set to EQUAL (or even LOW) for CICS-DB2 transactions, the CICS QR TCB can go to sleep while waiting for DB2 SQL TCB work to catch up ('no work' wait)
      - Particularly significant for transactions running as quasi-reentrant
        - Sleep/wake cycle can increase CPU cost by up to 30%
      - Less of a problem for transactions running as threadsafe
    - Special consideration if running at very high CPU utilisation
      - At a minimum, PRIORITY=HIGH should be used for transactions that are high volume and have many simple SQL statements
        - Weighted approach to determine which transactions may be the best candidates
        - (#SQL statements per transaction) * (frequency of transaction)

CICS/DB2 – Missed Performance Opportunities

- THREADSAFE
  - Can deliver much better CPU performance – up to 22%
  - Eliminates switching between CICS QR TCB and DB2 thread TCB
  - IBM Redbook: 'Threadsafe Considerations for CICS', SG24-6351



CICS/DB2 – Missed Performance Opportunities …

- PKLIST search optimisation (e.g. “COL_a.*, COL_b.*, COL_c.*”)
  - Efficient matching index access to find the package within each collection but DB2 goes serially through the PKLIST entries

PLAN/PACKAGE PROCESSING	QUANTITY	/SECOND	/THREAD	/COMMIT
PACKAGE ALLOCATION ATTEMPT	593.0K	165.31	140.49	21.61
PACKAGE ALLOCATION SUCCESS	43809.00	12.21	10.38	1.60

Success rate = PACKAGE ALLOC. SUCCESS / PACKAGE ALLOC. ATTEMPT

  - Impact of long PKLIST search
    - Additional CPU resource consumption, catalog accesses, and elapsed time
    - Can aggravate DB2 internal latch (LC32) contention
  - Recommendations
    - Reduce the number of collections on the PKLIST – even just one
      - Scrub all inactive or unused collections on PKLIST
      - Fold in and collapse the number of collections on PKLIST
    - Ruthlessly prioritise and reorder the collections on PKLIST to reflect frequency of access
      - Put the collections that are most likely to contain the packages first in the PKLIST
    - Use SET CURRENT PACKAGESET to direct the search to a specific collection

Distributed Workload and DDF

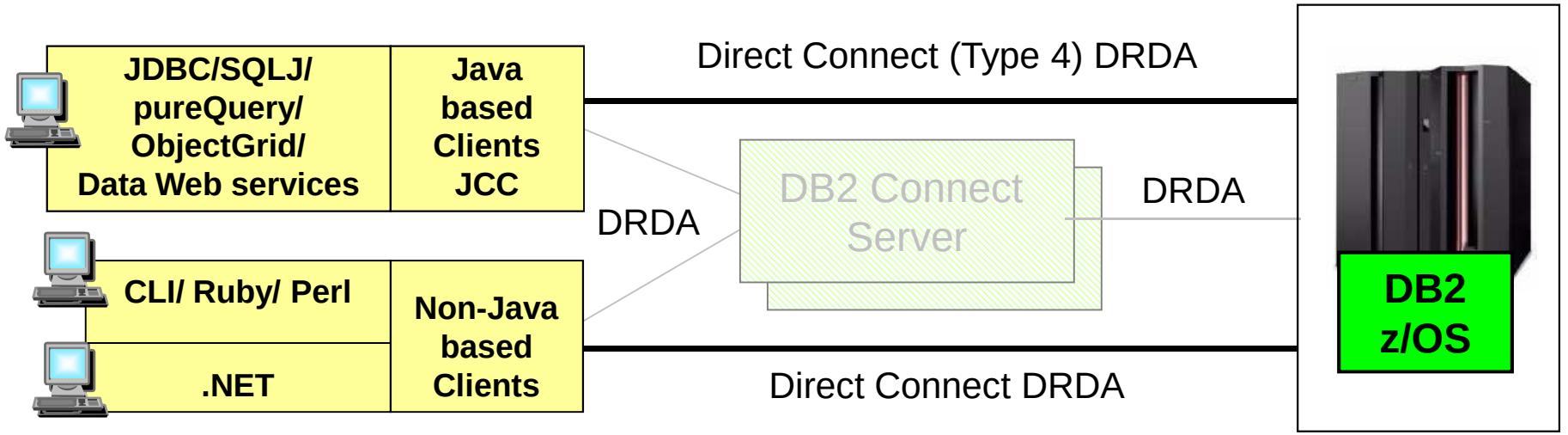
- Common problems
  - Applications not fully sysplex enabled
    - May be exploiting TCP/IP Sysplex Distributor
      - Does provide availability as client will always find a DB2 member (as long as one is up)
      - But only provides connection load distribution
        - Most applications use connection pooling and there is no frequent connect/disconnect
        - Therefore not an appropriate solution to distribute work
    - But often not exploiting transaction-level DB2 sysplex workload balancing (WLB)
      - Applications will be unavailable when
        - DB2 member is quiesced (e.g., subsystem maintenance)
        - DB2 member has crashed
      - Little resiliency to DB2 member slowdowns and/or DBM1 ASID 31-bit virtual storage shortages
        - For example, lock/latch contention creating queuing for database access threads (DBATs)
        - Shortage of DBATs
        - High level of number of connections
        - DB2 subsystem under stress (full system contraction, abort of application threads)
        - DB2 subsystem may crash out leading to unavailable applications

Distributed Workload and DDF ...

- Common problems ...
  - Low utilization of distributed threads (DBATs)
    - Driving up the high MAXDBAT requirement
    - Possible causes of low distributed thread utilization
      - Applications not committing frequently
      - Applications not explicitly closing result set cursors opened with the WITH HOLD clause
      - Applications not dropping declared global temporary tables when no longer needed
      - Over use of high performance DBATs in V10
  - No standards for the IBM Data Server Driver and/or DB2 ConnectClient/Server to ensure well-performing highly-available applications
  - No effective strategy to upgrade IBM Data Server Driver and/or DB2 ConnectClient/Server levels
  - No effective DB2 for z/OS server capacity planning to ensure future growth can be absorbed safely
  - Very limited ‘negative’ testing

Distributed Workload and DDF ...

- Recommendations
  - Migrate to exploit IBM Data Server Driver package as your direct connection to DB2 for z/OS Server
    - Fewer points of failure
    - Improved elapsed time
    - Improved serviceability
    - Significant reduction of client footprint
    - Much easier to stay current



Distributed Workload and DDF ...

- Recommendations ...
  - TCP/IP Sysplex Distributor should continue to be used for availability
  - Enable DB2 Sysplex WLB on all distributed applications to ensure workload distribution at the transaction level and seamless route around failure
    - There is no penalty
  - V9 - Tune MAXDBAT down based on available DBM1 ASID 31-bit virtual storage budget to reduce exposure to the DB2 out-of-storage condition
  - V10 - Tune MAXDBAT down based on available real storage budget to reduce exposure to excessive paging
  - Set CONDBAT to total number of concurrent requests once drivers are at V9.5 level or higher
    - Improved MAXDBAT queue drain

Distributed Workload and DDF ...

- Recommendations ...
  - Sysplex WLB relies on the inactive connection support
    - Enabled by DB2 system ZPARM CMTSTAT=INACTIVE
  - Application guidelines for effective Sysplex WLB
    - Commit frequently (but not with auto commit)
    - Explicitly close result set cursors opened with the WITH HOLD clause
    - Use ON COMMIT DROP TABLE or explicit DROP TABLE for DGTTs
    - Do not use KEEPDYNAMIC YES

Distributed Workload and DDF ...

- Recommendations ...
  - Standardize on IBM Data Server Driver Configuration
    - Best practice is often NOT to use default settings
    - Design recommendations for the configuration settings
      - Automatically close all open result set cursors on commit boundaries *(1)*
      - Implicitly close result set cursors after all rows are fetched *(2)*
      - Disable auto commit *(3)*
      - Drop connection to prematurely terminate long-running request *(4)*
      - Enable sysplex workload balancing *(5)*
      - Set max number of transports based on expected maximum number of concurrent requests across the DB2 data sharing group *(6)*
        - Should be agreed by DB2 for z/OS server team and included in proper capacity planning
        - If a DB2 member fails, the surviving DB2 members should be able to handle the load
      - Set login timeout based on how long you want to wait for a new connection before rerouting to another member e.g., 3 seconds *(7)*
      - Set low TCP/IP keepalive value to quickly detect network or member outages *(8)*
    - Deviation from design recommendations should only occur on a very exceptional basis

Distributed Workload and DDF ...

- Recommendations ...
  - Design recommendations for the configuration settings …

	Java driver	**Non-java driver**
(1)	setResultHoldability=2	CursorHold=0
(2)	queryCloseImplicit property set to Query_Close_Implicit_Yes (1) (default)	CursorTypes set to static forward-only cursors
(3)	setAutoCommit method	Autocommit configuration keyword
(4)	interruptProcessingMode property set to Interrupt_Processing_Mode_Close_Socket (2)	Interrupt=2
(5)	EnableSysplexWLB datasource property	enableWLB in the <WLB> section
(6)	MaxTransportObjects datasource property	maxTransports in the <WLB> section
(7)	LoginTimeout=3-5 seconds	MemberConnectTimeout=1 second (default)
(8)	TCP/IP Keepalive settings are set at the OS level	keepAliveTimeout

Distributed Workload and DDF ...

- Recommendations ...
  - Set unique ApplicationName client info in major application function
    - Better accountability
    - Detailed level monitoring
    - Facilitate problem determination and root cause analysis
    - Granular WLM definitions
  - Develop a set of common routines and enforce the usage

Java driver client info properties	Non-java driver client info properties
ApplicationName	SQLE_CLIENT_INFO_APPLNAME
ClientAccountingInformation	SQLE_CLIENT_INFO_ACCTSTR
ClientHostname	SQLE_CLIENT_INFO_WRKSTNNAME
ClientUser	SQLE_CLIENT_INFO_USERID
	sqleseti API callable service can set client information

Distributed Workload and DDF ...

- Recommendations ...
  - High Performance DBATs in V10
    - Reintroduces thread reuse and RELEASE(DEALLOCATE) for distributed threads
    - Potential to reduce CPU for DRDA transactions
      - Must be using CMTSTAT=INACTIVE so that threads can be pooled and reused
      - Packages must be bound with RELEASE(DEALLOCATE) to get reuse for same connection
        - Default on standard ODBC/JDBC packages supplied with latest drivers/connect packages
      - MODIFY DDF PKGREL(BNDOPT) must also be in effect
    - BUT …
      - Will drive up the MAXDBAT requirement
      - Will need additional real storage to support increased number of threads

Distributed Workload and DDF ...

- Recommendations ...
  - High Performance DBATs in V10 …
    - Objective: Selective use of High Performance DBATs
      - High-performance well-behaved applications running on distributed application servers
    - Standard ODBC and JDBC packages supplied with drivers/connect packages should be bound twice into two different package collections e.g.,
      - The CS package in collection1 will be bound with RELEASE(DEALLOCATE) so that the applications using that package will be eligible to use high performance DBATs
      - The CS package in collection2 (e.g., NULLID) will be bound with RELEASE(COMMIT) and will not use high performance DBATs
    - For JDBC applications
      - Set the currentPackageSet property in the respective datasource
    - For .NET and ODBC / CLI applications
      - Set CurrentPackageSet parameter in the db2dsdriver.cfg configuration

Distributed Workload and DDF ...

- Recommendations ...
  - Develop software migration strategy for IBM Data Server Driver
    - Maintain currency by applying preventative maintenance and upgrading releases on a regular basis to take advantage of new features, improvements in availability features, and workload management
  - Strongly encourage 'negative' testing
    - DB2 shutdown, DB2 failure, 'CPU-soaker' program, over-commitment of MAXDBAT, etc.
    - Objectives:
      - Evaluate readiness
      - Evaluate infrastructure for availability

Part III – Speed of Recovery

Disaster Recovery

- Context
  - Disaster recovery = complete site failure
- Common problems
  - Disconnect between IT and business
    - Lack of transparency about what IT can actually deliver
    - Lack of IT understanding of the real business needs and expectations
    - Lack of consistency between Continuous Availability and Disaster Recovery objectives
  - Compromised recoverability
    - Consistency of the remote copy with disk-based replication is not guaranteed
    - Forgetting to delete all CF structures owned by the data sharing group
  - Lack of automation and optimisation to recover in a timely manner
  - Limited or no testing
  - No Plan B

Disaster Recovery …

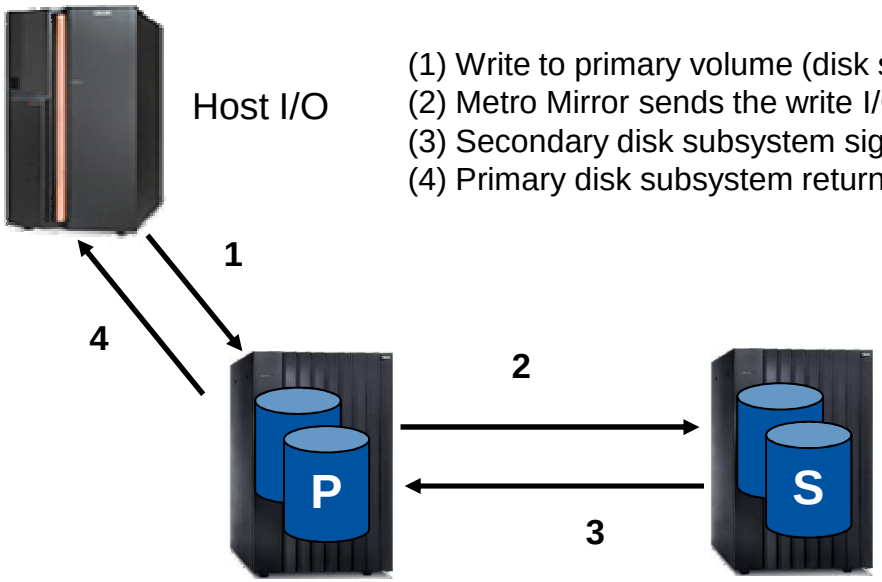
- Need to understand the real business requirements and expectations
  - These should drive the infrastructure, not the other way round
  - 'Quality of Service' requirements for applications
    - Availability
      - High availability? Continuous operations? Continuous availability?
      - Restart quickly? Mask failures?
    - Performance
  - In case of a Disaster
    - Recovery Time Objective (RTO)
      - How long can your business afford to wait for IT services to be resumed following a disaster?
    - Recovery Point Objective (RPO)
      - What is the acceptable time difference between the data in your production system and the data at the recovery site (consistent copy)?
      - In other words, how much data is your company willing to lose/recreate following a disaster?

Disaster Recovery …

- Critical concept for disk-based replication solutions
  - Dependent writes
    - The start of a write operation is dependent upon the completion of a previous write to a disk in the same storage subsystem or a different storage subsystem
      - For example, typical sequence of write operations for a database update transaction:
        1. An application makes an update and the data page is updated in the buffer pool
        2. The application commits and the log record is written to the log on storage subsystem 1
        3. At a later time, the update to the table space is externalized to storage subsystem 2
        4. A diagnostics log record is written to mark that the page has been externalized successfully
  - Data consistency for secondary DASD copy = I/O consistency
    - Order of the dependent writes is preserved
  - DB2 data consistency = application consistency
    - Re-established through normal DB2 warm restart recovery mechanisms
      - Requires an I/O consistent base

Metro Mirror

- a.k.a. PPRC (Peer-to-Peer Remote Copy)
- Disk-subsystem-based synchronous replication



(1) Write to primary volume (disk subsystem cache and NVS)
(2) Metro Mirror sends the write I/O to secondary disk subsystem
(3) Secondary disk subsystem signals write complete when the updated data is in its cache and NVS
(4) Primary disk subsystem returns Device End (DE) status to the application

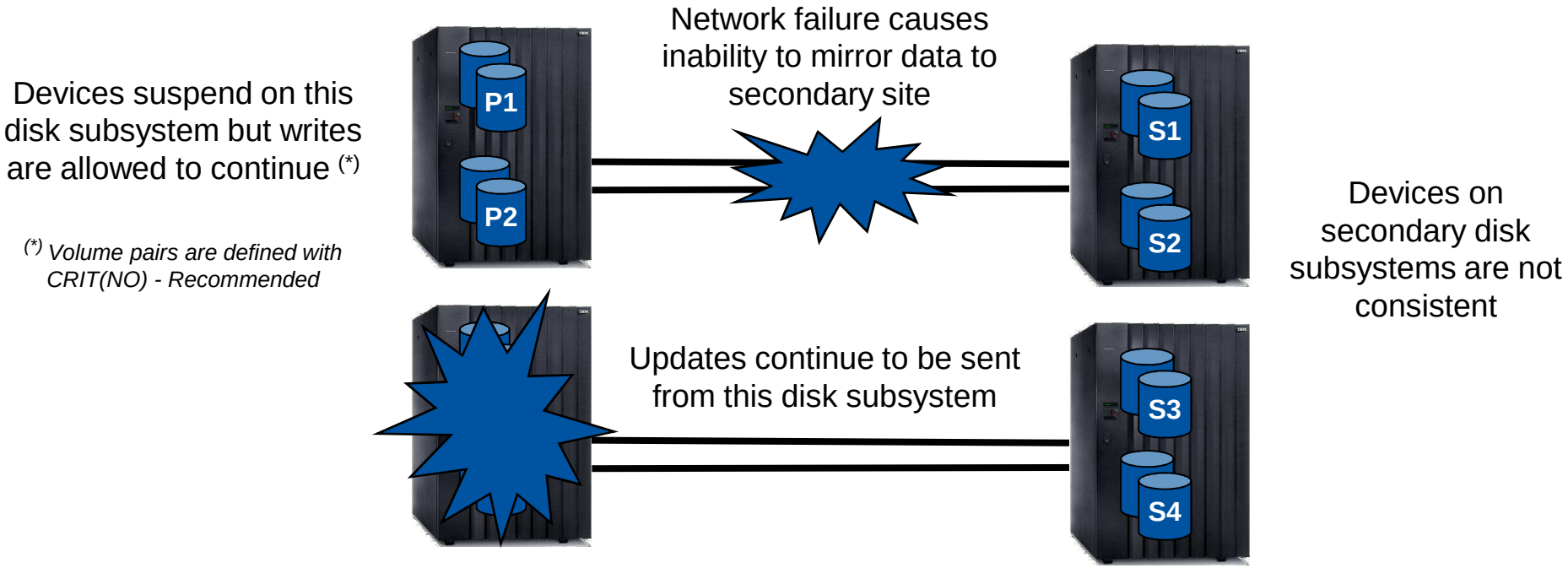
- Limited distance
  - Over-extended distance can impact the performance of production running on the primary site

Metro Mirror …

- Misconception #1: Synchronous replication always guarantees data consistency of the remote copy
- Answer:
  - Not by itself…
  - Metro Mirror operates at the device level (like any other DASD replication function)
    - Volume pairs are always consistent
  - But in a rolling disaster, cross-device (or boxes) consistency is not guaranteed
    - An external management method is required to maintain consistency

Metro Mirror …

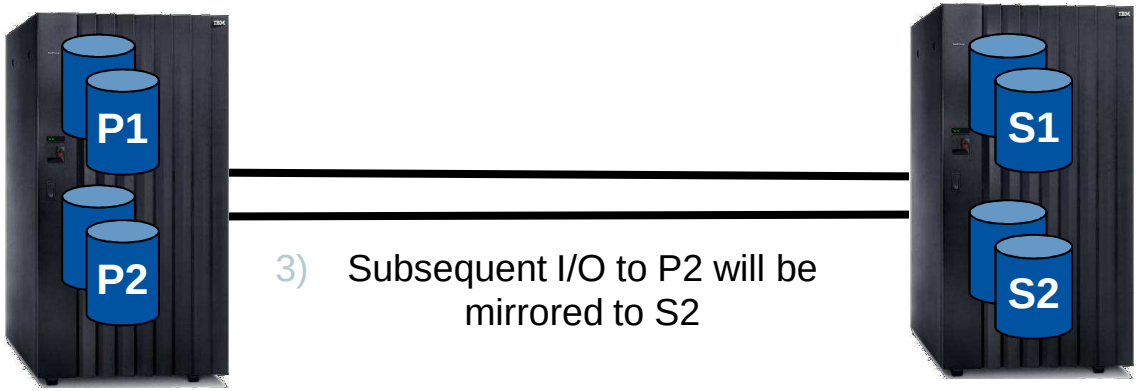
- Traditional example of multiple disk subsystems
  - In a real disaster (fire, explosion, earthquake), you can not expect your complex to fail at the same moment. Failures will be intermittent, gradual, and the disaster will occur over seconds or even minutes. This is known as the Rolling Disaster.



Metro Mirror …

- Example of single disk subsystem with intermittent problem with communication links or problem with the secondary DASD configuration

1) Temporary communications problem causes P1-S1 pair to suspend e.g. network or SAN event



3) Subsequent I/O to P2 will be mirrored to S2

2) During this time no I/O occurs to P2 so it remains duplex

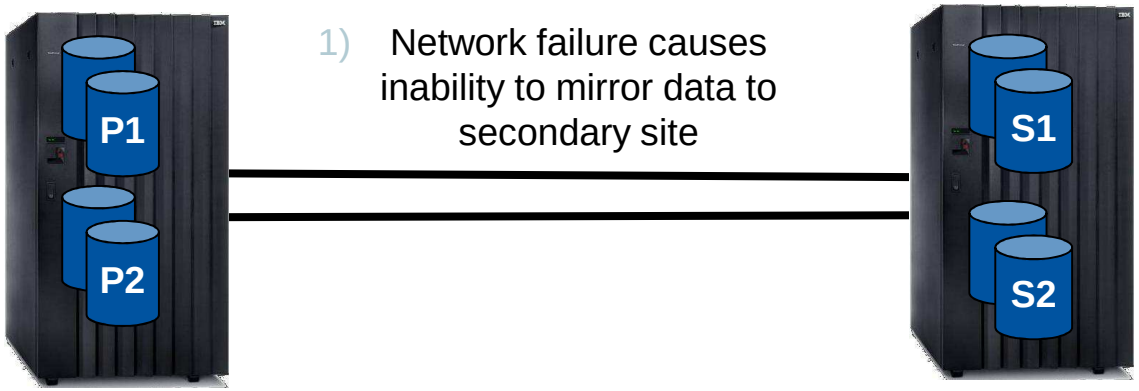
4) S1 and S2 are now not consistent and if the problem was the first indication of a primary disaster we are not recoverable

Metro Mirror …

- Consistency Group function combined with external automation

2) The volume pair defined with CGROUP(Y) that first detects the error will go into an 'extended long busy' state and IEA494I is issued

3) Automation is used to detect the alert and issue the CGROUP FREEZE command(*) to all LSS pairs



1) Network failure causes inability to mirror data to secondary site

(*) *or equivalent*

5) Automation issues the CGROUP RUN(*) command to all LSS pairs releasing the long busy. Secondary devices are still suspended at a point-in-time

4) CGROUP FREEZE(*) deletes Metro Mirror paths, puts all primary devices in long busy and suspends primary devices

Without automation, it can take a very long time to suspend all devices with intermittent impact on applications over this whole time

Metro Mirror …

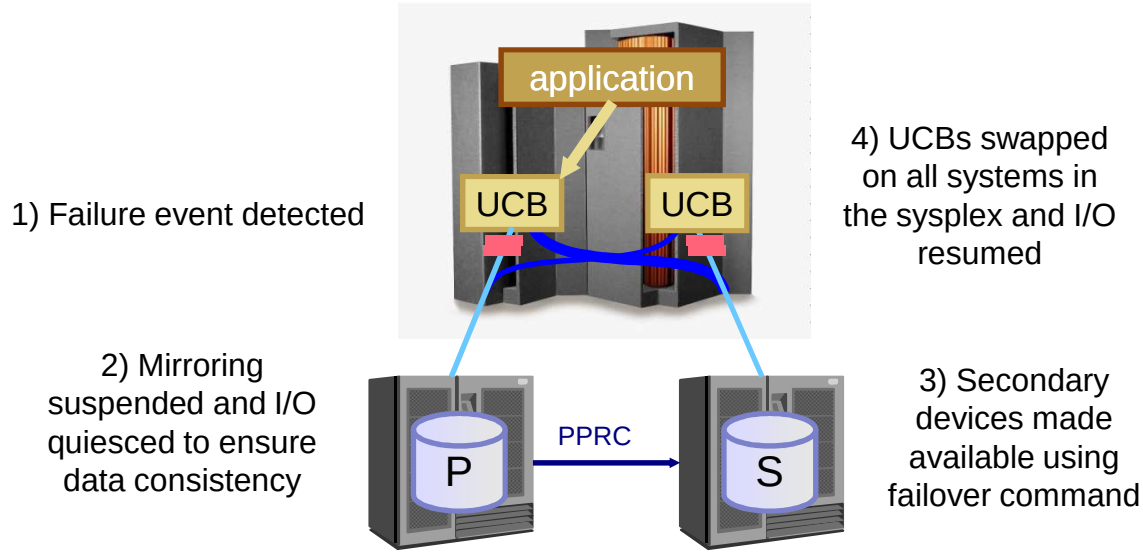
- Misconception #1: Synchronous replication always guarantees data consistency of the remote copy
- Answer:
  - In case of a rolling disaster, Metro Mirror alone does not guarantee data consistency at the remote site
  - Need to exploit the Consistency Group function AND external automation to guarantee data consistency at the remote site
    - Ensure that if there is a suspension of a Metro Mirror device pair, the whole environment is suspended and all the secondary devices are consistent with each other
    - Supported for both planned and unplanned situations

Metro Mirror …

- Misconception #2: Synchronous replication guarantees zero data loss in a disaster (RPO=0)
- Answer
  - Not by itself…
  - The only way to ensure zero data loss is to immediately stop all I/O to the primary disks when a suspend happens
    - e.g. if you lose connectivity between the primary and secondary devices
    - FREEZE and STOP policy in GDPS/PPRC
      - GDPS will reset the production systems while I/O is suspended
  - Choosing to have zero data loss really means that
    - You have automation in place that will stop all I/O activity in the appropriate circumstances
    - You accept a possible impact on continuous availability at the primary site
      - Systems could be stopped for a reason other than a real disaster (e.g. broken remote copy link rather than a fire in the computer room)

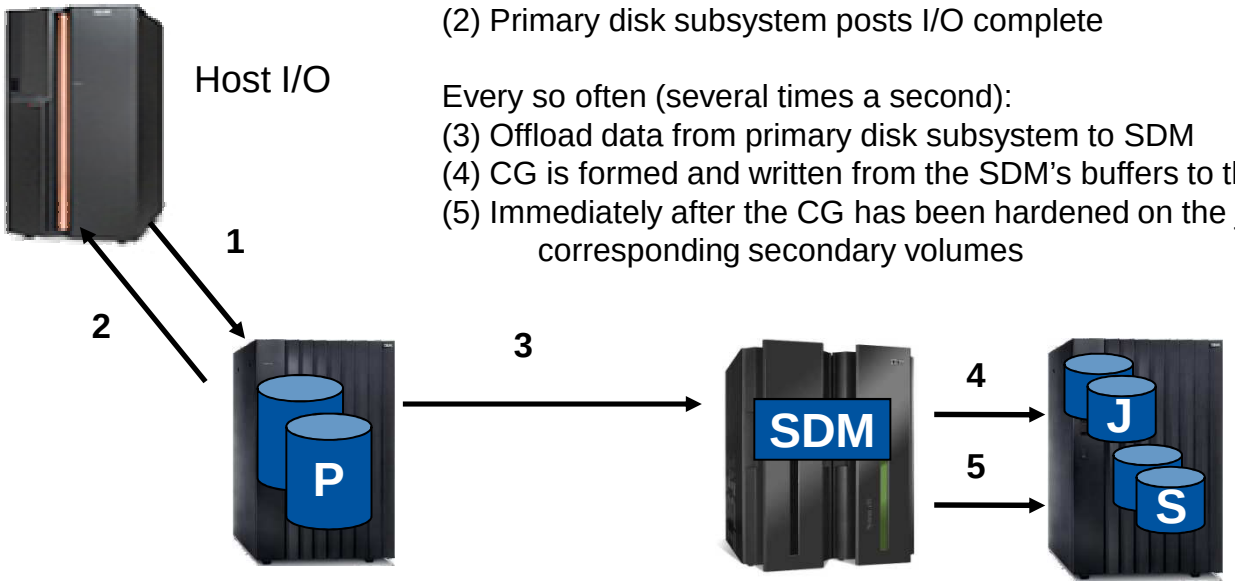
Metro Mirror …

- Misconception #3: Metro Mirror eliminates DASD subsystem as single point of failure (SPOF)
- Answer:
  - Not by itself...
  - Needs to be complemented by a non-disruptive failover HyperSwap capability e.g.,
    - GDPS/PPRC Hyperswap Manager
    - Basic Hyperswap in TPC-R



z/OS Global Mirror

- a.k.a. XRC (eXtended Remote Copy)
- Combination of software and hardware functions for asynchronous replication WITH consistency
  - Involves a System Data Mover (SDM) on z/OS in conjunction with disk subsystem microcode



(1) Write to primary volume
(2) Primary disk subsystem posts I/O complete

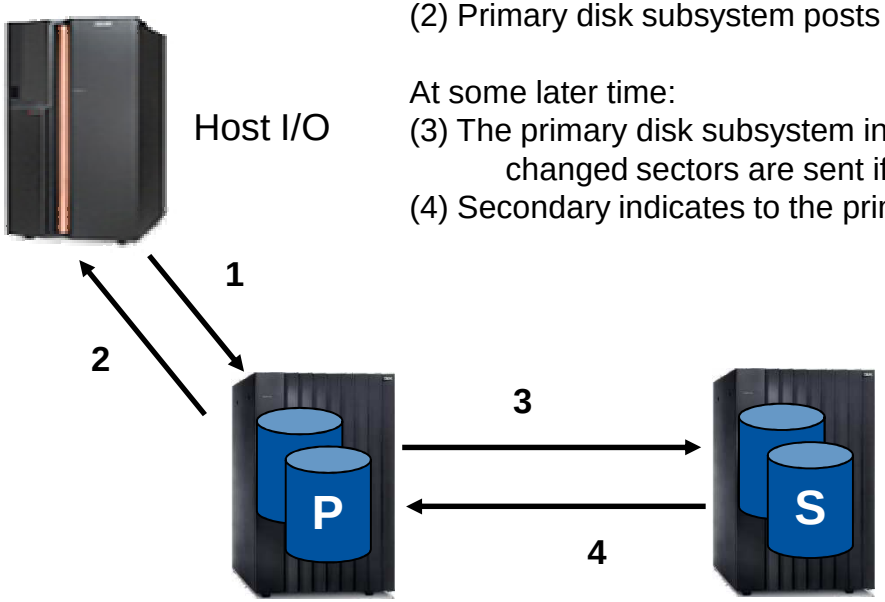
Every so often (several times a second):
(3) Offload data from primary disk subsystem to SDM
(4) CG is formed and written from the SDM's buffers to the SDM's journal data sets
(5) Immediately after the CG has been hardened on the journal data sets, the records are written to their corresponding secondary volumes

z/OS Global Mirror …

- Use of Time-stamped Writes and Consistency Groups to ensure data consistency
  - All records being written to z/OS Global Mirror primary volumes are time stamped
  - Consistency Groups are created by the SDM
    - A Consistency Group contains records that the SDM has determined can be safely written to the secondary site without risk of out-of-sequence updates
    - Order of update is preserved across multiple disk subsystems in the same XRC session
- Recovery Point Objective
  - Amount of time that secondary volumes lag behind the primary depends mainly on
    - Performance of the SDM (MIPS, storage, I/O configuration)
    - Amount of bandwidth
    - Use of device blocking or write pacing
      - Pause (blocking) or slow down (pacing) I/O write activity for devices with very high update rates
      - Objective: maintain a guaranteed maximum RPO
      - Should not be used on the DB2 active logs (increased risk of system slowdowns)

Global Copy

- a.k.a. PPRC-XD (Peer-to-Peer Remote Copy Extended Distance)
- Disk-subsystem-based asynchronous replication <u>WITHOUT consistency</u>



(1) Write to primary volume
(2) Primary disk subsystem posts I/O complete

At some later time:
(3) The primary disk subsystem initiates an I/O to the secondary disk subsystem to transfer the data (only changed sectors are sent if the data is still in cache)
(4) Secondary indicates to the primary that the write is complete - primary resets indication of modified track

Global Copy …

- Misconception #4: Global Copy provides a remote copy that would be usable in a disaster
- Answer:
  - Not by itself…
  - Global Copy does NOT guarantee that the arriving writes at the local site are applied to the remote site in the same sequence
    - Secondary copy is a ‘fuzzy’ copy that is just not consistent
  - Global Copy is primarily intended for migrating data between sites or between disk subsystems
  - To create a consistent point-in-time copy, you need to pause all updates to the primaries and allow the updates to drain to the secondaries
    - E.g., Use the -SET LOG SUSPEND command for DB2 data

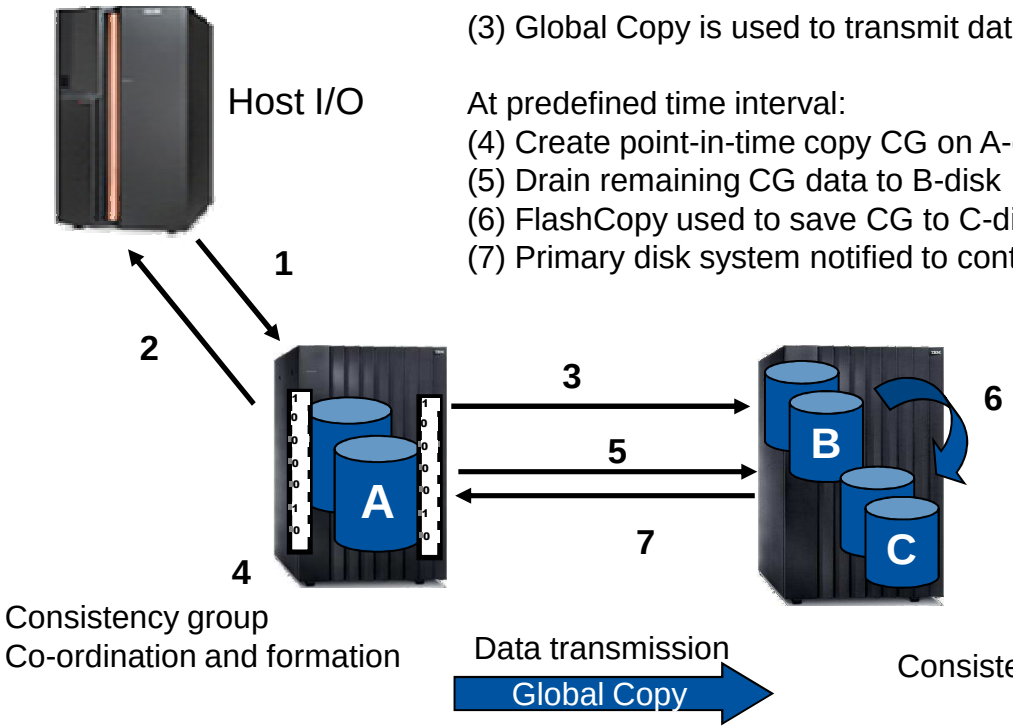
Global Mirror

- Combines Global Copy and FlashCopy Consistency Groups
- Disk-subsystem-based asynchronous replication WITH consistency

(1) Write to primary volume
(2) Primary disk subsystem posts I/O complete

At some later time:
(3) Global Copy is used to transmit data asynchronously between primary and secondary

At predefined time interval:
(4) Create point-in-time copy CG on A-disks – Write I/Os queued for short period of time (usually < 1 ms)
(5) Drain remaining CG data to B-disk
(6) FlashCopy used to save CG to C-disks
(7) Primary disk system notified to continue with the Global Copy process



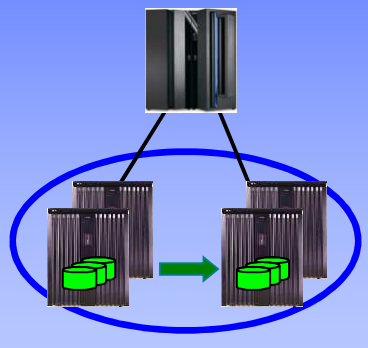
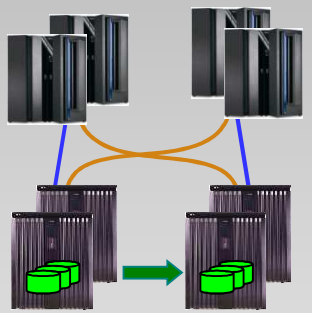
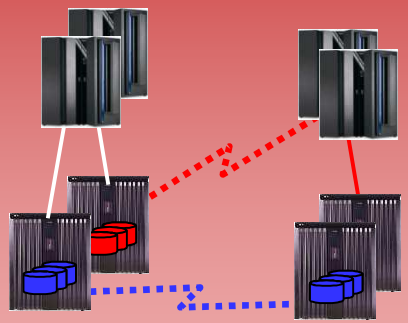
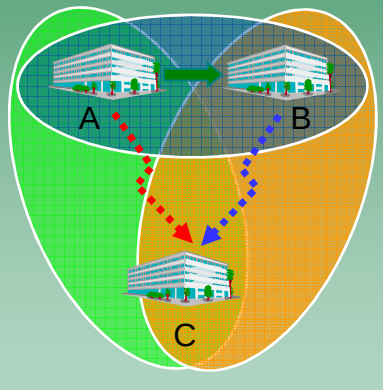
Global Mirror …

- Recovery Point Objective
  - Amount of time that the FlashCopy target volumes lag behind the primary depends mainly on
    - Bandwidth and links between primary and secondary disk subsystems
    - Distance/latency between primary and secondary
    - Hotspots on secondary in write intensive environments
  - No pacing mechanism in this solution
    - Designed to protect production performance at the expense of the mirror currency
    - RPO can increase significantly if production write rates exceed the available resources

Remote Copy Services Summary

	Metro Mirror	z/OS Global Mirror	Global Copy	Global Mirror
	Synchronous	Asynchronous with consistency	Asynchronous without consistency	Asynchronous with consistency
	Hardware	Hardware and software SDM on z/OS only	Hardware	Hardware
Used for	HA and DR	DR	Data Migration	DR
Distance	Metro distances, with impact to the response time of the primary devices (10 microsec/km)	Virtually unlimited distances, with minimal impact to the response time of the primary devices	Virtually unlimited distances, with minimal impact to the response time of the primary devices	Virtually unlimited distances, with minimal impact to the response time of the primary devices
RPO	0 if FREEZE and STOP (no data loss, but no more production running) > 0 if FREEZE and RUN (data loss if real disaster)	1-5 seconds or better with sufficient bandwidth and resources Max RPO can be guaranteed with pacing	Depends on user-managed procedures and consistency creation interval	3-5 seconds or better with sufficient bandwidth and resources No pacing
Supported devices	CKD (System z) and FB (Open)	CKD only (z/OS, z/Linux, z/VM)	CKD (System z) and FB (Open)	CKD (System z) and FB (Open)
Supported disk subsystems	Any enterprise disk subsystem (ESS, DS6000, DS8000, HDS/EMC)	Any enterprise disk subsystem (ESS, DS8000, HDS/EMC)	IBM enterprise disk subsystem (ESS, DS6000, DS8000)	IBM enterprise disk subsystem (ESS, DS6000, DS8000)

CA and DR Topologies and the GDPS Family

Continuous Availability of Data within a Data Center	Continuous Availability / Disaster Recovery within a Metropolitan Region	Disaster Recovery at Extended Distance	Continuous Availability Regionally and Disaster Recovery Extended Distance
Single Data Center Applications remain active Continuous access to data in the event of a storage subsystem outage	Two Data Centers Systems remain active Multi-site workloads can withstand site and/or storage failures	Two Data Centers Rapid Systems Disaster Recovery with 'seconds' of Data Loss Disaster recovery for out of region interruptions	Three Data Centers High availability for site disasters Disaster recovery for regional disasters
			
GDPS/HyperSwap Mgr	GDPS/HyperSwap Mgr GDPS/PPRC	GDPS/GM GDPS/XRC	GDPS/MGM GDPS/MzGM

DB2 Restart Recovery

- DB2 restart is the key ingredient for disk-based DR solutions
  - Normal DB2 warm restart
  - Re-establishes DB2 data consistency through restart recovery mechanisms
  - Directly impact the ability to meet the RTO
    - See tuning recommendations for fast DB2 restart in Part I
- DB2 restart MUST NOT be attempted on a mirrored copy that is not consistent
  - Guaranteed inconsistent data that will have to be fixed up
  - No way to estimate the damage
  - After the restart it is too late if the damage is extensive
  - Damage may be detected weeks and months after the event
- DB2 cold start or any form of conditional restart WILL lead to data corruption and loss of data

DB2 Restart Recovery …

- All DB2 CF structures MUST be purged before attempting a disaster restart
  - Otherwise, guaranteed logical data corruption
  - Required even if using Metro Mirror with FREEZE/GO policy
    - No consistency between ‘frozen’ secondary copy and content of the CF structures
  - Necessary actions and checks should be automated
  - New DB2 ZPARM in V10
    - DEL_CFSTRUCTS_ON_RESTART=YES/NO
    - Auto-delete of DB2 CF structures on restart if no active connections exist
  - GRECP/LPL recovery will be required as a result of deleting the GBPs
    - Likely to be the largest contributor to DB2 restart time

DB2 Restart Recovery …

- Optimise GRECP/LPL Recovery
  - Frequent castout
    - Low CLASST (0-5)
    - Low GBPOOLT (5-25)
    - Low GBPCHKPT (2-4)
  - Switch all objects to CLOSE YES to limit the number of objects to recover
    - Also has the potential to reduce the data sharing overhead
    - Additional recommendations
      - Tune DSMAX to avoid hitting it too often
      - Adjust PCLOSEN/T to avoid too frequent pseudo closes
        - ROT: #DSETS CONVERTED R/W -> R/O < 10-15 per minute
      - Apply APAR PM5672
        - Fixed issue: Candidate list for physical close becomes stale and may result in incorrectly closing datasets that are currently being frequently accessed, rather than others which are not currently being used

DB2 Restart Recovery …

- Optimise GRECP/LPL Recovery …
  - V9 – Set ZPARM LOGAPSTG (Fast Log Apply) to maximum buffer size of 100MB
  - V10 – LOGAPSTG is removed and the implicit size is 510MB
  - Build an automated GRECP/LPL recovery procedure that is optimised
    - Identify the list of objects in GRECP/LPL status
    - Start with DB2 Catalog and Directory objects first
    - For the rest, generate optimal set of jobs to drive GRECP/LPL recovery
      - Spread the -START DATABASE commands across all available members
        - V9 – 10 commands maximum per member (based on 100MB of FLA storage)
        - V10 – 51 commands maximum per member
      - 20-30 objects maximum per -START DATABASE command

DB2 Restart Recovery …

- Optimise GRECP/LPL Recovery …
  - DB2 9 can automatically initiate GRECP/LPL recovery at the end of both normal restart and disaster restart when a GBP failure condition is detected
  - Why would I still need a procedure for GRECP/LPL recovery?
    - If for any reason the GRECP recovery fails
      - Objects that could not be recovered will remain in GRECP status and will need to be recovered by issuing -START DATABASE commands
    - If the GBP failure occurs after DB2 restart is complete
      - GRECP/LPL recovery will have to be initiated by the user
    - Highly optimised user-written procedures can still outperform and be more robust than the automated GRECP/LPL recovery
      - May want to turn off automated GRECP/LPL recovery if aggressive RTO
        - DB2 -ALTER GBPOOL …. AUTOREC(NO)

Disaster Recovery – Testing

- Need to periodically rehearse DR
- Objectives
  - Validate recovery procedures and maintain readiness
  - Ensure that RTO/RPO can be achieved
  - Flush out issues and get them fixed
    - Any sign of inconsistency found in your testing should be driven to root cause
      - Broken pages, data vs.index mismatches, etc.
- Recommendations
  - Allocate resources to test DR without impacting production
    - E.g. Splitting/suspending the DASD replication and using Flashcopy
  - Testing should be end-to-end, not based on individual application silos
  - Test in anger and not simulated
    - Need representative application workload running concurrently

Disaster Recovery – Testing

- Recommendations …
  - Verify data consistency
    - DBD integrity
      - REPAIR DBD DIAGNOSE DATABASE dbname for each user data base
    - DB2 RI and table check constraint violations + consistency between base table and corresponding LOB/XML table spaces
      - CHECK DATA TABLESPACE dbname.tsname part SCOPE ALL
    - Consistency between table spaces and indexes
      - CHECK INDEX ALL TABLESPACE or CHECK INDEX LIST (with appropriate LISTDEF)
    - Invalid LOB values or structural problems with the pages in a LOB table space
      - CHECK LOB TABLESPACE dbname.tsname for each LOB table space
    - Integrity of the DB2 catalog
      - IBM-provided DSNTESQ queries (should always return zero rows)
    - Options to check each page of a table space for consistency
      - DSN1COPY with CHECK option
      - SELECT * from tbname (throw away output)
      - Basic RUNSTATS to touch each row

Mass Data Recovery

- Context
  - Widespread logical data corruption e.g. introduced by
    - Operational error, CF microcode, DASD microcode, DB2 bug, etc.
  - Plan B for disaster recovery if
    - The mirror is damaged/inconsistent
    - Bad Disaster Restart e.g., using stale CF structures
  - Mass data recovery would be based on ‘traditional’ DB2 log-based forward recovery (image copies + logs)
  - Very rare event but if not prepared, practiced and optimised, it will lead to extended application service downtimes (possibly many hours to several days)
  - In real world mass data recovery is more frequent than a true DR event

Mass Data Recovery ...

- Common problems
  - Lack of planning, design, optimisation, practice & maintenance
  - No prioritised list of application objects and inter-dependencies
    - Limited use of DB2 referential integrity
    - Logical model, data dependencies, integrity management are buried in the applications
    - Heavily dependant on application knowledge and support
  - Procedures for taking backups and executing recovery compromised by lack of investment in technical configuration
  - Use of tape including VTS
    - Cannot share tape volumes across multiple jobs
    - Relatively small number of read devices
    - Concurrent recall can be a serious bottleneck
    - Even though VTS has a disk cache, it is known to z/OS as tape device
      - Same serialization characteristics as all tape devices
      - A single virtual volume cannot be shared by different jobs or systems at the same time

Mass Data Recovery ...

- Results: any or all of the following
  - No estimate of elapsed time to complete
  - Elongated elapsed time to complete recovery
  - Performance bottlenecks so that recovery performance does not scale
  - Breakage in procedures
  - Surprises caused by changing technical configuration
  - Unrecoverable objects

Logging Environment

- Design for performance
  - Keep a minimum of 6h of recovery log data in the active log pairs at any time
    - Objective: provide some reaction time in case of archiving problem
    - Adjust number/size of the DB2 active log pairs (limit is 93 log pairs)
    - Active logs can be added dynamically in DB2 10
      - New -SET LOG NEWLOG option
      - New active log must be IDCAMS defined & preformatted by DSNJLOGF
      - Only a single log dataset at a time >> Issue command twice for dual logging
      - No dynamic delete of active logs
  - Keep 48-72h of recovery log data on DASD
    - Keep archive log COPY1 on DASD for 48-72h before migrating it to tape
    - Archive log COPY2 can be migrated away to tape at any point
  - Be ready to extend the amount of recovery log beyond what is available on DASD
    - Set BLKSIZE=24576 to optimise reads on DASD
    - Prepare a procedure to copy archive logs from tape or VTS to DASD

Logging Environment …

- Design for resiliency
  - Separate COPY1 and COPY2 of the active log pairs and BSDS across different DASD controllers
  - Isolate objects into separate ICF catalogs to achieve more resiliency from an ICF catalog failure
    - Separate out the datasets for each DB2 member into separate ICF catalogs
      - Active logs, archive logs and BSDS for member DB2A away from those for member DB2B
      - Result: an ICF catalog failure would only affect one DB2 member
    - Should also consider further isolation
      - COPY1 of active log, archive log and BSDS into one ICF catalog
      - COPY2 of active log, archive log and BSDS into an alternate ICF catalog
      - Result: an ICF catalog failure would not affect DB2
    - Additional ICF catalog considerations for better performance and resilience
      - Isolate DB2 catalog/directory objects into a separate ICF catalog
      - Use multiple ICF catalogs for the DB2 user objects
      - Separate ICF catalogs for DB2 objects and DB2 image copy backups

Logging Environment …

- Design for serviceability
  - Retain archive log data for 30 days
  - At first sign of logical data corruption, stop the deletion of image copies and archive log datasets

DB2 Catalog and Directory

- Keep the DB2 catalog/directory FICs on DASD to speed up recovery
- Need to build and maintain a JCL to recover the DB2 catalog/directory
  - Sizes of the underlying data sets must be documented and readily available
  - Must be maintained over time when number of datasets and/or sizes are changed
- Recovery of DB2 catalog/directory should be periodically rehearsed
- Need a process to check periodically the DB2 catalog/directory integrity
  - Using a cloned copy of the DB2 catalog/directory into an auxiliary DB2 subsystem
- Periodic REORG of the DB2 catalog/directory outside of release migration
  - Most importantly, SYSLGRNX should be reorganised every quarter
    - Can be run as SHRLEVEL(CHANGE) at a time of low activity
    - Will speed up online REORG, MODIFY RECOVERY, GRECP/LPL recovery

Mass Data Recovery

- The following information should be documented, kept current and made readily available to the DBAs
  - Consolidated logical/physical database design view
  - Application RI relationships
  - Dependencies across application domains
- Need to agree on a prioritised list of business-critical applications and related data required by these applications
  - This list should be used to drive the recovery order
- Critical information needed during a recovery event
  - Objective: Bring back critical application services as soon as possible
  - Without these lists, either have to wait for the whole world to be recovered, or take a risk in bringing back some of the application services earlier
  - Should not rely exclusively on application expertise

Mass Data Recovery ...

- Schedule a daily production job to check for unrecoverable objects
- Build, test and optimise a set of recovery jobs exploiting capacity of the entire DB2 data sharing group
  - Up to 10 (V9) or 51 (V10) RECOVER jobs per DB2 subsystem
  - 20-30 objects per RECOVER
  - Spread the jobs across all DB2 data sharing members
  - If elapsed time needs to be improved further, look for possible optimisations e.g.,
    - Change the recovery order to factor in the constraining elapsed time of the critical objects (RECOVER TABLESPACE+REBUILD index)
    - Consider possible use of BACKUP SYSTEM / RESTORE SYSTEM
    - Increase image copy frequency for critical objects
    - Keep more image copies on DASD
    - Optimise copy of archive logs to DASD
    - Consider use of COPY/RECOVER instead of REBUILD for very large NPIs
    - Additional buffer pool tuning for DB2 catalog objects

Mass Data Recovery ...

- Practise regular full-scale ‘fire drills’ for mass recovery of the entire system
  - Find out what the actual service level is
  - Validate that procedures are in working order
    - Both for local and remote DR recovery
  - Maintain readiness on mass recovery execution
- Need to invest in procedures to validate the success of the recovery and possibly compensate for lost work
  - Requires active cooperation and buy-in from the application teams
  - Develop and maintain data reconciliation programs
  - Consider cloning the data into an auxiliary DB2 subsystem, turn on DB2 RI and use CHECK DATA to validate the relationships
  - Requires investment in a Log Analysis tool to support this critical step
    - Needs regular practice on the tool

Mass Data Recovery ...

- FlashCopy provides interesting new options to help with mass data recovery
  - System-wide point-in-time backup
    - Can restart off a previous PIT copy
    - Can use RESTORE SYSTEM LOGONLY or RECOVER to come forward in time
  - Quick cloning of the environment to create a ‘forensic’ system
- I/O consistency of the FlashCopy backup is key to allow quick restart
  - When using FlashCopy to take backups outside of DB2’s control
    - Option 1: Use -SET LOG SUSPEND to temporarily ‘freeze’ all DB2 update activity
      - For Data Sharing, you must issue the command on each data sharing member and receive DSNJ372I before you begin the FlashCopy
    - Option 2: Use the FlashCopy Consistency Group function
      - Not 100% transparent to the applications!
        - All volumes in the group are put in long busy state (i.e. all writes are suspended) until they are all copied or the timeout value is reached (2 minutes with default settings)
      - Use option FCCGVERIFY to determine if the copies of the group of volumes are consistent
    - DB2 warm restart recovery will re-establish data consistency

Part IV – Operations

Preventative Software Maintenance

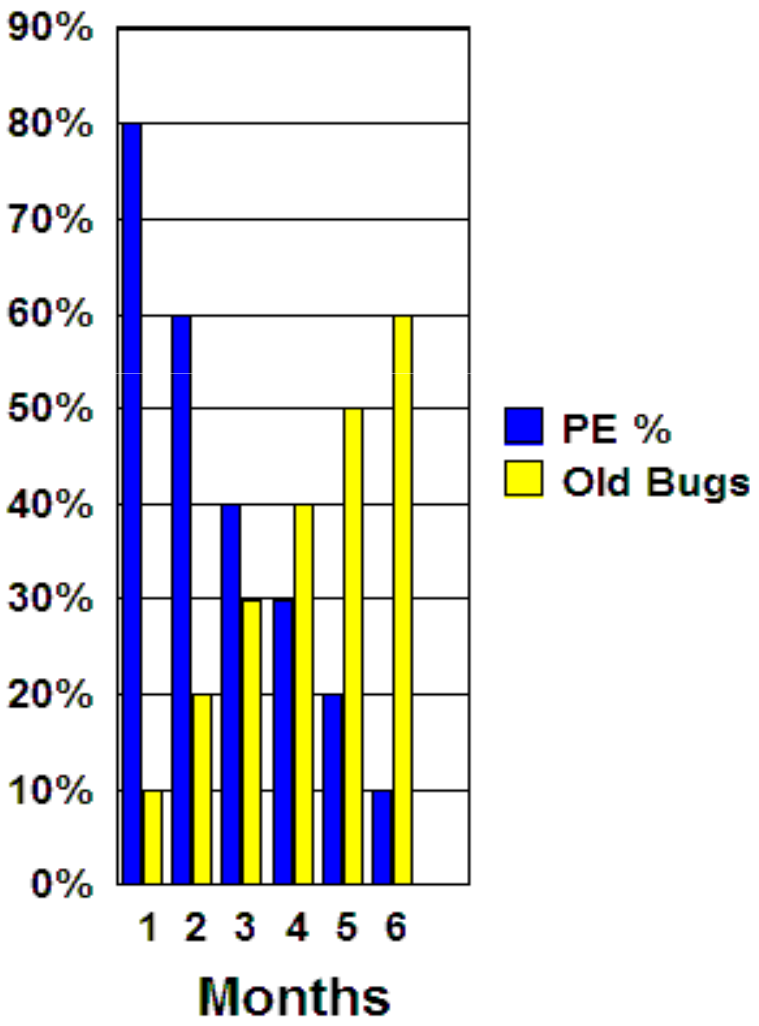
- Common problems
  - Too many customers are very back level on preventative service
  - No HIPERs or PE fixes applied since the last preventative service upgrade
  - High profile production incidents could have been avoided by missing HIPER
  - ‘Fix-on-failure’ culture introduces the probability of long prerequisite chain when having to apply emergency corrective service
  - Not exploiting DB2 Data Sharing technology to avoid planned outages and remove dependency on change windows
  - Delay in exploiting new availability functions
  - Delay in applying DB2 serviceability enhancements to prevent outages
  - ‘One-size-fits-all’ approach across multiple different application environment leads to escalating maintenance costs

Why Install Preventive Software Maintenance?

- Achieving the highest availability depends on a having an adaptive preventive maintenance process
- Applying preventive maintenance can and will avoid outages
  - Up to 20% of multi-system outages could have been avoided by regularly installing 'critical' (e.g., HIPER and PE fixing) PTFs
- Executing a preventive maintenance process requires an understanding of trade-offs in achieving high system availability

Maintenance Trade-Off

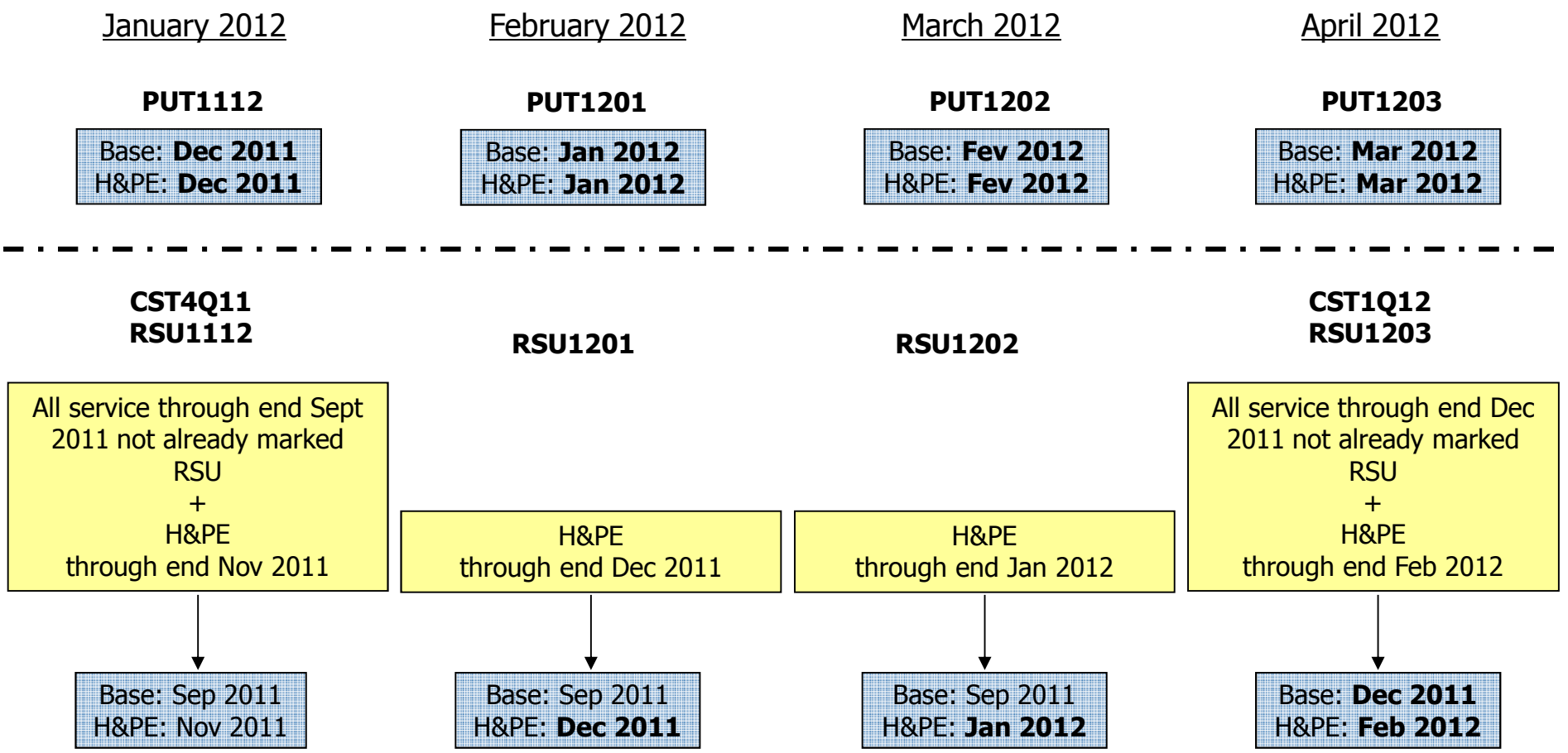
- Must balance for severity vs. risk
  - Problems encountered vs. problems avoided
  - Potential for PTF in Error (PE)
  - Application work load type
  - Windows available for installing service
- Need adaptive service strategy that is adjusted based on
  - Experience over previous 12-18 months
  - Attitude to risk in changing environment and exploiting new DB2 releases and associated products, and new feature/function
  - DB2 product and service plans



Consolidated Service Test (CST)

- Goals
  - Enhance the way service is tested and delivered for z/OS, by providing a single coordinated service recommendation
    - Provides cross-product testing for participating products
    - List of products tested is continually expanding
    - Performed in addition to existing testing programs and does not replace any current testing performed by the products
    - Standardize maintenance recommendation on z/OS platform
- Results
  - Quarterly report available on CST website
    - http://www.ibm.com/systems/z/os/zos/support/servicetest
  - Monthly addendum with update on tested HIPERs and PE fixes
  - After service has passed CST testing
    - Marked with RSU (Recommended Service Upgrade) RSUyymm SOURCEID notation

CST/RSU vs. PUT Calendar



H&PE = HIPER/Security/Integrity/Pervasive PTFs + PE resolution (and associated requisites and supersedes)

Enhanced HOLDDATA

- http://service.software.ibm.com/holddata/390holddata.html
- Key element of the CST/RSU best practices process
  - Simplifies service management
  - Can be used to identify missing PE fixes and HIPER PTFs
    - SMP/E REPORT ERRSYSMODS
  - Produces a summary report
    - Includes the fixing PTF number when the PTF is available
    - Includes HIPER reason flags
      - IPL, DAL (data loss), FUL(major function loss), PRF (performance), PRV (pervasive)
    - Identifies whether any fixing PTFs are in RECEIVE status (available for installation) and if the chain of PTFs to fix the error has any outstanding PEs
  - Enhanced HOLDDATA is updated daily
    - A single set of HOLDDATA is cumulative and complete
    - Up to 3 years of history is available

Fix Category HOLDDATA (FIXCAT)

- http://www.ibm.com/systems/z/os/zos/smpe/fixcategory.html
- Fix categories can be used to identify a group of fixes that are required to support a particular HW device, or to provide a particular software function
  - Supplied in the form of SMP/E FIXCAT HOLDDATA statements
    - Each FIXCAT HOLDDATA statement associates an APAR to one or more fix categories
- Current list of FIXCAT HOLDs for DB2 for z/OS

SYSPLEXDS	Fixes that enable the sysplex data sharing function or fixes required for data sharing
DB2COEXIST/K	Fixes that enable DB2 releases to coexist when in data sharing mode
DB2PARALL/K	Fixes for problems with the DB2 parallelism function
DB2STGLK/K	Fixes for DB2 storage leak problems
DB2OVRLAY/K	Fixes for DB2 storage overlay problems
DB2INCORR/K	Fixes for DB2 SQL incorrect output problems
DB2MIGV9/K	Fixes that allow the prior release of DB2 to migrate to or fall back from DB2 V9
DB2MIGV10/K	Fixes that allow prior releases of DB2 to migrate to or fall back from DB2 V10
ISAOPT/K	Fixes required for z/OS software products to support the IBM Smart Analytics Optimizer

Preventive Software Maintenance …

- Recommendations
  - Change Management process should balance the risk of making 'no change' vs. making 'a change'
  - Apply preventative maintenance every 3 months
    - Use RSU instead of PUT to be less aggressive on applying non-HIPER maintenance
    - Sample strategy based on two 'major' and two 'minor' releases
      - Refresh of the base every 6 months ('major')
      - Each base upgrade should be based on latest quarterly RSU
        - Ensure that RSU-only service is installed by adding the SOURCEID (RSU*) option in the supplied APPLY and ACCEPT jobs
      - In addition, two mini packages covering HIPERs and PEs in between ('minor')
    - Early adopters of new releases and/or new functions should be more aggressive about applying preventative service
  - Review Enhanced HOLDDATA on a weekly basis
    - Expedite critical fixes to production after 1-2 weeks in test
    - Others can be deferred until the next major or minor maintenance drop

Preventive Software Maintenance …

- Recommendations …
  - Develop processes/procedures and technical changes to implement ‘rolling’ maintenance outside of heavily constrained change windows
    - Separate SDSNLOAD per DB2 member
    - Separate ICF User Catalog Alias per DB2 member
    - Benefits of ‘rolling’ maintenance
      - One DB2 member stopped at a time
      - DB2 data is continuously available via the N-1 members
      - Fallback to the prior level is fully supported if necessary
      - Difficult if your applications have affinities!
  - Aim for company wide certification of new release/maintenance
    - Shared test environment with collaborative use by systems programmer, DBA, and all the application teams
    - Additional ‘insurance policy’ before starting to roll out new DB2 maintenance package

Preventive Software Maintenance …

- Recommendations …
  - Separated DB2 maintenance environments
    - Have single 'master' SMP/E environment as base
    - Have one additional SMP/E environment to match each DB2 application environment
    - Certification testing based on 'master' before starting to promote new maintenance to Test and Production environments
    - Benefits of separated DB2 maintenance environments
      - No 'one size fits all'
      - Can be more aggressive on applying maintenance to specific DB2 environment(s)
      - While protecting the stability of other DB2 application environments
      - Very flexible approach that meets application development requirements to use new functions
      - Supports the migration of new DB2 releases perfectly as DB2 application environments can be treated independently

DB2 Virtual and Real Storage

- Common problems
  - CTHREAD and/or MAXDBAT set too high
    - Inflated size of storage cushion (V9)
      - DBM1 full system storage contraction
    - No denial of service (V9/V10)
      - EDM 31-bit pool full condition (V9) > SQLCODE -904 returned to the application
      - DBM1 full system contraction > Potentially degrading performance
      - DBM1 storage critical > Individual DB2 threads may abend with 04E/RC=00E200xx
      - DB2 crashes out
      - Excessive paging
      - LPAR crashes out
  - Aggravated by
    - Lack of balance across CICS/WAS and DB2
    - Workload failover
    - Abnormal slow downs
    - Workload moves

DB2 Virtual and Real Storage …

- Problems
  - Shortage of real storage
    - Can lead to excessive paging and severe performance issues
    - Ultimately, can take the LPAR out
      - Once all AUX is consumed, the LPAR goes into a wait state
    - Can lead to long DUMP processing times and cause major disruption
      - DUMP should complete in seconds to make sure no performance problems ensue
      - Once paging begins, it is possible to have DUMP processing take 10s of minutes with high risk of system-wide or even sysplex-wide slowdowns
        - Could be running ‘one DUMP away from a disaster’
    - Wasted opportunities for CPU reduction
      - Reluctance to use bigger or more buffer pools
      - Reluctance to use buffer pool long-term page fix
      - Many performance opportunities in DB2 10 require real storage

DB2 Virtual and Real Storage …

- Recommendations
  - Collect IFCID 225 data from DB2 start to DB2 shutdown
  - Use SPREADSHEETDD subcommand option of OMPE to post process SMF data
    - Note: SPREADSHEETDD support in OMPE has not been enhanced to support V10
  - Use sample REXX program MEMU2 to pull the data via IFI
    - Support both V9 and V10
    - Available on the DB2 for z/OS Exchange community website on IBM My developerWorks
      - http://www.ibm.com/developerworks/software/exchange/db2zos
      - Click on 'View and download examples‘ and search on tag with 'memu2
  - Plan on a basic storage cushion (free) to avoid full system storage contraction plus approx 100MB to allow for some growth and margin for error
  - Project how many active threads can be supported
  - Set CTHREAD and MAXDBAT to realistic values that can be supported
  - Balance the design across CICS AORs connecting to the DB2 subsystem

DB2 Virtual and Real Storage …

- Recommendations …
  - Track messages issued by the DB2 Internal Monitor
    - Automatically issues console messages (DSNV508I) when DBM1 31-bit virtual storage crosses (increasing or decreasing) thresholds
      - Threshold augmented in PM38435 to account for the storage cushion

```
DSNV508I -SE20 DSNVMON - DB2 DBM1 BELOW-THE-BAR
STORAGE NOTIFICATION
          77% CONSUMED
          76% CONSUMED BY DB2
          352M AVAILABLE OUT OF REGION SIZE 1553M
          WITH A 274M STORAGE CUSHION
```

    - Identifies the agents that consume the most storage
    - Can get status at any time using -DIS THREAD(*) TYPE(SYSTEM)

```
NAME     ST A   REQ ID          AUTHID   PLAN     ASID   TOKEN
VA1A     N  *   0 002.VMON 01   SYSOPR            002A   0
V507-ACTIVE MONITOR, INTERVALS=8216, STG=77%, BOOSTS=0, HEALTH=100
     REGION=1553M, AVAIL=352M, CUSHION=274M
```

DB2 Virtual and Real Storage …

- Recommendations …
  - Not good practice to configure REAL storage based on normal operating conditions and rely on DASD paging space to absorb the peaks
  - Need to provision sufficient REAL storage to cover both the normal DB2 working set size and MAXSPACE requirement
    - MAXSPACE can typically be up to 8-9GB in V9
    - Should plan for approx 16GB for MAXSPACE in V10
    - Undersizing MAXSPACE will result in partial dumps
      - Seriously compromises problem determination
  - Dumps should be taken very quickly (<10 secs) almost without any one noticing i.e., little or no disruption to the subject LPAR and to the rest of the Parallel Sysplex
    - Consider automation to kill dumps taking longer than 10 secs

DB2 Virtual and Real Storage …

- Recommendations …
  - V9/V10 – Enable thread storage contraction to help protect the system against excessive DBM1 31-bit virtual storage usage
    - Run with CONTSTOR=YES (not the default in V9)
      - With YES, DB2 compresses out part of Agent Local Non-System storage based on number of commits and thread size
      - Maximum of 1 compress every 5 commits so very cheap to implement
      - Ineffective for long-running persistent threads with use of RELEASE(DEALLOCATE)
  - V10 – Enable DISCARD mode to contract storage and protect the system against excessive paging and use of AUX
    - Apply DB2 APAR PM24723 and prereq z/OS APAR OA35885
    - Run with REALSTORAGE_MANAGEMENT=AUTO (default)
      - With AUTO, DB2 detects whether paging is imminent and reduces the frame counts to avoid system paging - switches between ON/OFF
    - Note: z/OS APAR OA37821 and corresponding DB2 APAR PM49816 address a potential CPU increase especially when multiple DB2 subsystems are running on the same LPAR

DB2 Virtual and Real Storage …

- Recommendations …
  - Real storage needs to be monitored as much as virtual storage
    - Important subsystems such as DB2 should not be paging IN from AUX (DASD)
      - Recommendation to keep page-in rates near zero
      - Monitor using RMF Mon III
      - Monitor in DB2 page in for reads/writes and output log buffer paged in
    - Collect IFCID 225 data from DB2 start to DB2 shutdown
    - High auxiliary storage in use should be investigated
      - Indicates that there are periods when the DB2 working set is pushed out to AUX (DASD)
      - Need to understand what is the driver

V9

```
REAL AND AUXILIARY STORAGE                      QUANTITY
-------------------------------------  ---------------
REAL STORAGE IN USE                (MB)         5958.66
AUXILIARY STORAGE IN USE           (MB)            0.00
```

DB2 Virtual and Real Storage …

V10

REAL AND AUXILIARY STORAGE FOR DBM1		QUANTITY
REAL STORAGE IN USE	(MB)	2525.20
31 BIT IN USE	(MB)	258.09
64 BIT IN USE	(MB)	2267.10
64 BIT THREAD AND SYSTEM ONLY	(MB)	0.00
HWM 64 BIT REAL STORAGE IN USE	(MB)	2321.07
AVERAGE THREAD FOOTPRINT	(MB)	5.70
AUXILIARY STORAGE IN USE	(MB)	0.00
31 BIT IN USE	(MB)	0.00
64 BIT IN USE	(MB)	0.00
64 BIT THREAD AND SYSTEM ONLY	(MB)	0.00
HWM 64 BIT AUX STORAGE IN USE	(MB)	0.00

REAL AND AUXILIARY STORAGE FOR DIST		QUANTITY
REAL STORAGE IN USE	(MB)	49.87
31 BIT IN USE	(MB)	38.91
64 BIT IN USE	(MB)	10.96
64 BIT THREAD AND SYSTEM ONLY	(MB)	0.00
HWM 64 BIT REAL STORAGE IN USE	(MB)	10.97
AVERAGE DBAT FOOTPRINT	(MB)	3.46
AUXILIARY STORAGE IN USE	(MB)	0.00
31 BIT IN USE	(MB)	0.00
64 BIT IN USE	(MB)	0.00
64 BIT THREAD AND SYSTEM ONLY	(MB)	0.00
HWM 64 BIT AUX STORAGE IN USE	(MB)	0.00

SUBSYSTEM SHARED STORAGE ABOVE 2 GB		QUANTITY
REAL STORAGE IN USE	(MB)	0.00
SHARED THREAD AND SYSTEM	(MB)	0.00
SHARED STACK STORAGE	(MB)	0.00
AUXILIARY STORAGE IN USE	(MB)	0.00
SHARED THREAD AND SYSTEM	(MB)	0.00
SHARED STACK STORAGE	(MB)	0.00

MVS LPAR SHARED STORAGE ABOVE 2 GB		QUANTITY
SHARED MEMORY OBJECTS		4.00
64 BIT SHARED STORAGE	(MB)	327680.00
HWM FOR 64 BIT SHARED STORAGE	(MB)	327680.00
64 BIT SHARED STORAGE BACKED IN REAL	(MB)	1777.25
AUX STORAGE USED FOR 64 BIT SHARED	(MB)	0.00
64 BIT SHARED STORAGE PAGED IN FROM AUX	(MB)	0.00
64 BIT SHARED STORAGE PAGED OUT TO AUX	(MB)	0.00

COMMON STORAGE BELOW AND ABOVE 2 GB		QUANTITY
(...)		
REAL STORAGE IN USE	(MB)	0.00
AUXILIARY STORAGE IN USE	(MB)	0.00

LPAR level instrumentation for REAL and AUX storage use
Warning: Numbers reported are inaccurate if more than one DB2 subsystem on the LPAR or if other subsystems use 64-bit shared
Solution: DB2 APAR PM24723 and prereq z/OS APAR OA35885

DB2 Virtual and Real Storage …

- How to limit REAL storage
  - V9 – Hidden ZPARM SPRMRSMX (‘real storage kill switch’)
    - Delivered in APAR PK18354
    - Not widely broadcasted
    - Prevents a runaway DB2 subsystem from taking the LPAR down
      - Should be used when there is more than one DB2 subsystem running on the same LPAR
      - Aim is to prevent multiple outages being caused by a single DB2 outage
      - Should be set to 1.5x to 2x normal DB2 subsystem usage
      - Kills the DB2 subsystem when SPRMRSMX value is reached
  - V10 – SPRMRSMX becomes an opaque ZPARM REALSTORAGE_MAX
    - Need to factor in 64-bit shared and common use to establish new footprint

Performance and Exception-Based Monitoring

- Common problems
  - Operating mostly in ‘fire-fighting’ mode, reacting to performance problems
  - Missing performance data and/or lack of granularity
    - Limits ability to drive problem to root cause
    - May need to wait for a reoccurrence of the problem
  - No performance database and/or limited use of it
    - No ‘baseline’ for DB2 system and application performance
    - No trend analysis
  - Not near-term history capability
  - No DB2 exception monitoring
    - Early signs that the system is in trouble go undetected
    - No one knows there is an issue until the situation escalates
    - Delayed PD/PSI (problem determination/problem source identification)
  - No control over dynamic SQL for .NET, ODBC and JDBC applications

Performance and Exception-Based Monitoring …

- Recommendations for data collection
  - SMFSTAT = YES (default)
    - Starts the trace for the default classes (1,3,4,5)
    - Consider adding Stats Class 8 for data set I/O statistics
  - V9 – STATIME=1 **** Important recommendation ****
    - Highly valuable / essential to study evolutional trend that led to complex system problems, e.g. slowdowns etc.
  - V10 – IFCIDs 2, 202, 217, 225, 230 are always cut at a fixed 1-minute interval
    - STATIME controls the frequency of IFCIDs 105, 106, 199
  - Copy SMF 100 records and to keep them separately
    - Relatively small amount of the total SMF data volume
    - Improved elapsed time performance to post process

Performance and Exception-Based Monitoring …

- Recommendations for data collection …
  - Should try to run with SMFACCT = (1,2,3,7,8)
    - Accounting Class 10 should not be active on a permanent basis in production
  - Options to consider if SMF data volume is an issue
    - Record SMF data to System Logger log streams rather than VSAM files
      - Improved performance and throughput
    - V10 – Enable DB2 compression of SMF data to reduce SMF data volume
      - Experience is that Accounting records compress 70-80%
      - Tiny CPU overhead at ~1%
    - Accounting ROLLUP (ACCUMACC)
      - Only for DDF and RRS Attach
      - No package-level rollup in V9
      - Lose the granularity – if you have a problem that is not pervasive, you've lost the data

Performance and Exception-Based Monitoring …

- Proactively monitor and review key DB2 metrics on a regular basis
  - Develop an automated process to store away DB2 performance data in DB2 tables
    - At least DB2 stats data (with no aggregation)
  - Invest in a set of 'canned' reports to transform the DB2 stats data into information
    - Track the evolutionary trend of key performance indicators at DB2 system level
    - Alert on 'out of normal' conditions at DB2 system level
    - Establish a baseline for analysis
    - Could also be used as additional input for management reporting

Additional resources

Optimising DB2 for z/OS System Performance Using DB2 Statistics Trace

One-day seminar by John Campbell and Florence Dubois

Provides a list of performance metrics and rules-of-thumbs in the areas of buffer pools, group buffer pools, lock/latch contention, global lock contention, logging, system address space CPU, EDM pool, DBM1 virtual storage, etc.

Webcasts available on the DB2 for z/OS Best Practices website:

https://www.ibm.com/developerworks/data/bestpractices/db2zos/#MONIT270

Performance and Exception-Based Monitoring …

- Enable near-term history collection in your DB2 online monitor
  - Provide the ability to retrieve and review DB2 statistics and accounting records from the past few hours of DB2 processing, and intercept adverse changing trends
- Effective exception monitoring at DB2 system and application levels
  - Track key statistics performance indicators in your DB2 online monitor and generate alerts on ‘out-of-range’ conditions
    - Performance metrics and rules-of-thumbs in the performance one-day seminar can be used as a starting point to implement an initial set of alert thresholds
  - Keep a log/history of the alerts and analyse for trends
  - Objectives
    - Get the production support teams engaged sooner
      - Help avoid a problem escalating into a serious incident
      - Help avoid extended recovery times, performance problems and application availability issues
    - Narrow down the scope of a problem to speed up reaction time during PD/PSI
    - Understand ‘weak points’ and drive strategic remedial actions

Performance and Exception-Based Monitoring …

- Enhance the capture of diagnostics data
  - Issue at 1-minute intervals via automation and save away the outputs
    - -DISPLAY THREAD(*) SERVICE(WAIT)
  - Issue at 15-minute intervals via automation and save away the outputs
    - -DISPLAY THREAD(*) SCOPE(GROUP) TYPE(INDOUBT)
    - -DISPLAY DATABASE(*) SPACENAM(*) RESTRICT LIMIT(*)
    - -DISPLAY UTIL(*)
    - -DISPLAY THREAD(*) TYPE(SYSTEM)
    - -DISPLAY THREAD(*) SERVICE(STORAGE)
    - -DISPLAY BPOOL(*) DETAIL(INTERVAL)
    - -DISPLAY GBPOOL(*) MDETAIL(INTERVAL)
    - -DISPLAY GROUP
    - -DISPLAY ARCHIVE
    - -DISPLAY DDF
    - -DISPLAY LOCATION(*) DETAIL
    - F irlmproc,STATUS, ALLD
    - D XCF, STR
    - D GRS,C

Should also be used to trigger alerts based on defined thresholds

Objective: Detect and correct problems as fast as possible, and avoid, where possible, long-running recovery actions

Performance and Exception-Based Monitoring …

- Exploit the dynamic statement cache for .NET, ODBC and JDBC applications
  - Sample procedure to explain statements from the DB2 statement cache
    1. Create table DSN_STATEMENT_CACHE_TABLE to hold the execution statistics data, and explain tables DSN_FUNCTION_TABLE, DSN_STATEMENT_TABLE and PLAN_TABLE to hold access path details
       - See member DSNTESC of the SDSNSAMP library
    2. Start the collection of dynamic statement cache performance statistics
       - -START TRACE(P) CLASS(30) IFCID(316, 317, 318)
    3. Use EXPLAIN STMTCACHE ALL to extract the SQL statements from the global cache and dump the statistics information to DSN_STATEMENT_CACHE_TABLE
       - All statements included if EXPLAIN executed by SYSADM
       - Otherwise only statements with matching authorization
    4. Generate individual EXPLAIN statements for each SQL statement in the cache

```
SELECT 'EXPLAIN STMTCACHE STMTID '||STRIP(CHAR(STMT_ID))||' ;'
FROM USERID.DSN_STATEMENT_CACHE_TABLE;
```

    5. Import the content of the 4 tables into a spreadsheet for post-processing

John Campbell & Florence Dubois

IBM DB2 for z/OS Development

campbelj@uk.ibm.com

fldubois@uk.ibm.com

One-Day Seminar

Recommended Best Practices from Customer DB2 Health Check Studies