



Introduction to DB2 LUW Performance

*IM Partner Technologies - Americas
IBM Toronto Lab*

Winter/Spring 2010

DB2 Performance Clinic Modules - Agenda

- **Introduction to DB2 LUW Performance & Hands on Lab**
- **DB2 Performance - Monitoring Essentials & Hands on Lab**
- **DB2 Performance - SQL Query Tuning & Hands on Lab**
- **DB2 Performance – Locking & Hands on Lab**
- **DB2 Performance – Logging & Hands on Lab**
- **DB2 Performance – Utilities & Hands on Lab**

Introduction to DB2 Performance - Agenda

- **Performance Tuning Overview**
- **DB2 Architecture Fundamentals**
- **Hardware Planning Rules of Thumb**
- **Database Design & Recommendations**

Agenda

- **Performance Tuning Overview**
- DB2 Architecture Fundamentals
- Hardware Planning Rules of Thumb
- Database Design & Recommendations

Why have a DB2 Performance Clinic?

- **Performance problems are trickier than functional problems**
 - Symptoms may have no clues about the problem source, e.g. you observe general slowdown and excessive lock timeouts. Source of problem might be outdated stats leading to bad access plans that are using table scans (instead of index scans)
 - A Performance Problem can be intermittent
 - Performance Problems can be avoided
- **Some common reactions to performance problems (especially if you're new at this...)**
 - Panic
 - Buy more hardware (CPU, Memory, Disk, etc.)
 - Blame DB2...
 - or AIX / Windows / Linux...
 - or IBM / HP / Sun /...
 - Take “shots in the dark” at the problem
 - Making almost random changes based on not much data

What is Performance?

- **The way a system behaves in response to a particular workload**
- **Measured in terms of system response time, throughput, and resource utilization**
- **Affected by:**
 - Resources available on the system
 - How well those resources are used and shared
- **Typically tuned to improve cost-benefit ratio**
 - Processing larger, or more demanding workloads without increasing processing costs
 - Obtaining faster system response times, or higher throughput, without increasing processing costs
 - Reducing processing costs without degrading service to users

Performance Tuning Limits

- **How much time and money should be spent?**
 - Assess the degree to which the investment will help the users
- **Tuning can often help improve**
 - Response times
 - Throughput problems
- **More significant problems may require**
 - More disk storage
 - Faster/additional CPUs
 - More memory
 - Faster network



Benchmarking: A measured approach to tuning

- **Normal part of the application development life cycle**
 - Involves application developers and database administrators
- **Determines current performance and can be used to improve application performance**
- **Based upon controlled conditions**
 - Repeatedly running SQL from your application
 - Change some of the following, for example, between iterations:
 - System configuration
 - SQL
 - Indexes
 - Table space configurations
 - Hardware configurations
 - Repeat until the application runs as efficiently as possible
- **Characteristics of good benchmarks include:**
 - Repeatable tests
 - Each iteration starts in the same system state
 - No other applications are unintentionally active in the system
 - Hardware and software used match production environment

How to avoid performance problems

- **Know your environment**

- OS
- Hardware
 - Processing
 - Memory
 - Storage

- **Understand how DB2 works and how to work DB2**

- Architecture
- Requirements

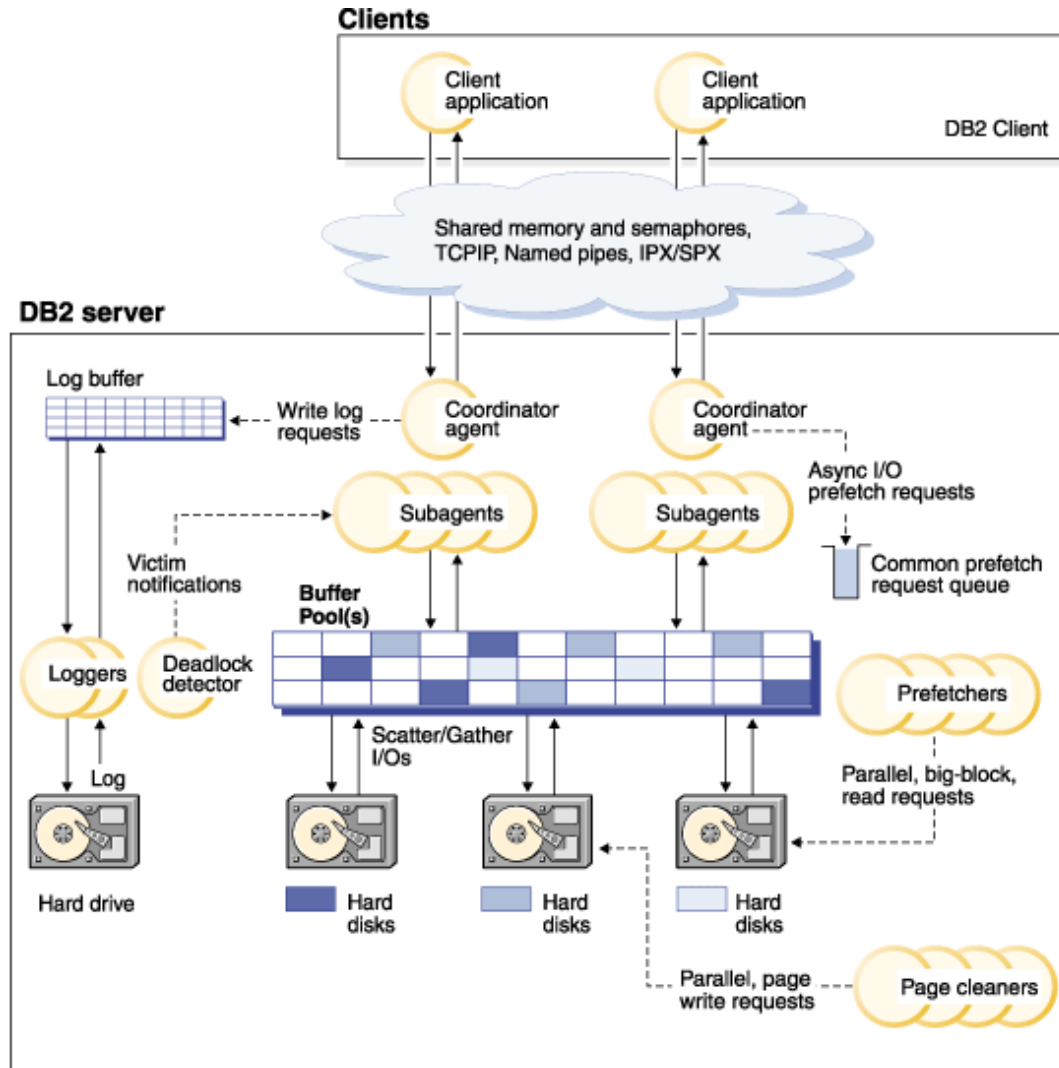
- **Start with a strong foundation**

- Integrated Data Management over the life of your system = IBM Optim
 - <http://www-01.ibm.com/software/data/data-management/optim-solutions>
- Best Practices
 - <http://www.ibm.com/developerworks/data/bestpractices>
- Autonomics

Agenda

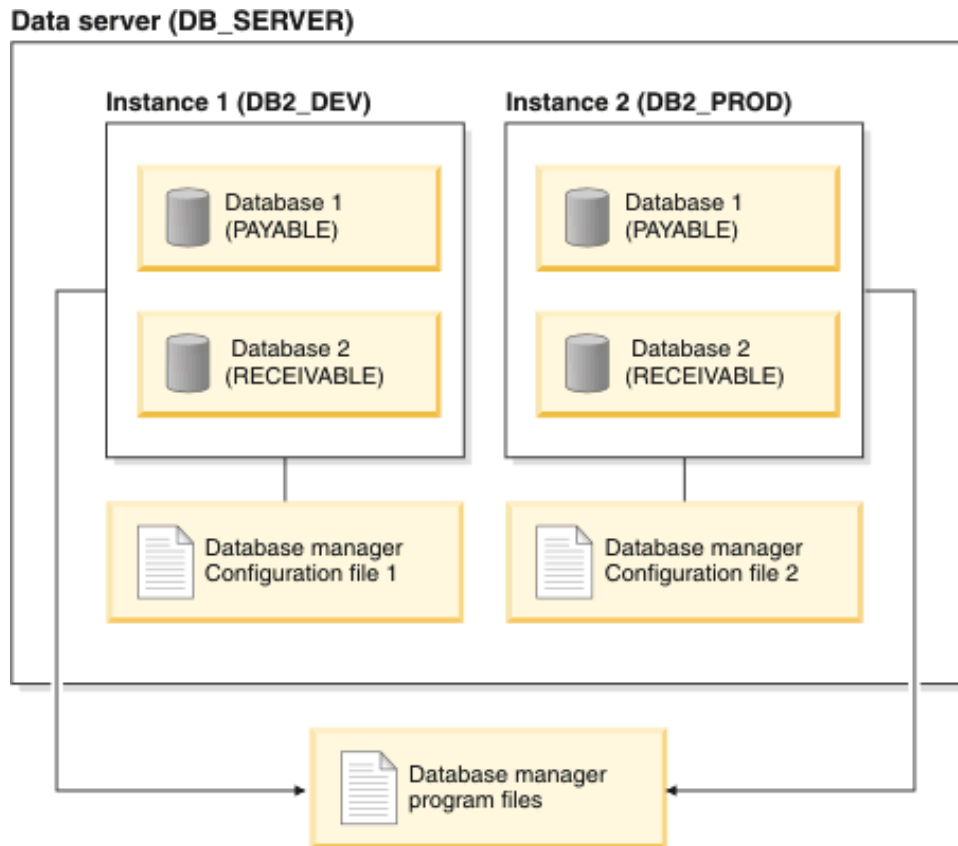
- Performance Tuning Overview
- **DB2 Architecture Fundamentals**
- Hardware Planning Rules of Thumb
- Database Design & Recommendations

Understand DB2's Architecture



DB2 Instances

- Stand-alone DB2 environment
- Can have multiple instances per data server/OS instance
- All instances share the same executable binary files
- Each instance has its own configuration
- Different software level for an instance



DB2 Storage

Instance ↔ Database ↔ Schema ↔ Table ↔ Row/Cell

- **Table space**

- Collection of containers



- **Container**

- Physical storage device



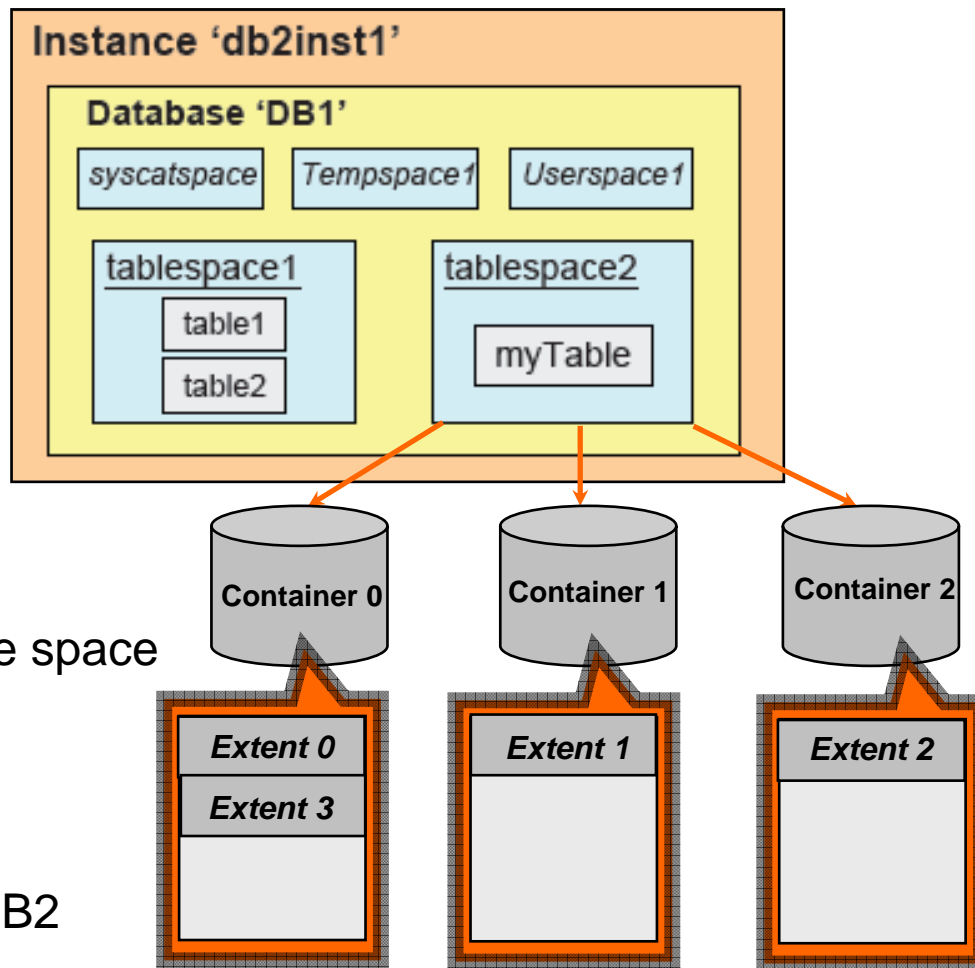
- **Extent:**

- Consecutive pages in a table space



- **Page**

- Smallest unit of storage in DB2

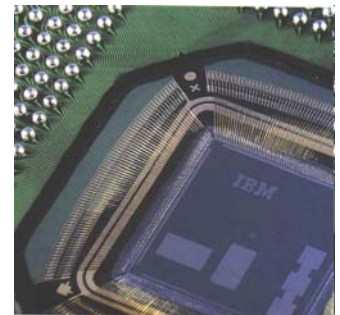


Agenda

- Performance Tuning Overview
- DB2 Architecture Fundamentals
- **Hardware Planning Rules of Thumb**
- Database Design & Recommendations

CPU – the Main Independent Variable in Performance

- **Rule of Thumb for Business Intelligence (BI) environments:**
 - 200-300 GB of active raw data per processor core is a reasonable estimate.
- **Other environments (OLTP):**
 - Try to gauge amount of CPU required based on one or more existing DB2 systems.
 - E.g.: new system to handle 50% more users, SQL is at least as complex as on an existing system → reasonable to assume that 50% more CPU capacity is required.
 - Take a look at www.tpc.org TPC-C DB2 9 results, for example:
 - 8 core IBM POWER5 1.9 GHz = ~430K TPM
 - 8 core IBM POWER6 4.2 GHz = ~630K TPM
 - 64 core IBM POWER6 5.0 GHz = ~6M TPM
 - 8 core Intel Xeon 3.33 GHz = ~314K TPM
 - 16 core Intel Xeon 2.93 GHz = ~517K TPM



Storage

Considerations:

1. I/O Throughput
 - I/Os per Second (IOPS)
 - Megabytes per Second (MPS)
2. Storage Capacity
3. Separate dedicated (unshared) disks for logging

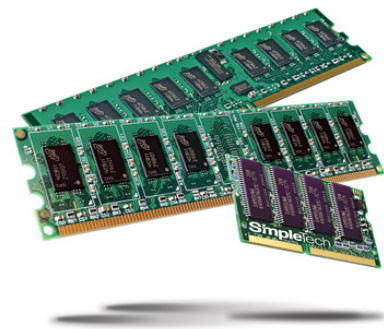


Rules of Thumb:

- **15K RPM drive = ~150-175 IOPS @ 7-8 milliseconds response time**
- **10 to 20 disk per core for average workload**
 - 10 for Intel/AMD (System x)
 - 20 for POWER5 and POWER6 (System p)
- **Logging:**
 - RAID-1 pair of disks enough for up to 400 reasonably write-intensive DB2 transactions per second.
 - Higher volume/greater throughput can be provided by additional disks in RAID-10 configuration connected through a write-caching disk controller

Memory

- **Decouples CPUs and Disks**
- **Limited by addressable shared memory**
 - 32 bit ~4GBs (supported on only Windows and Linux)
 - 64 bit virtually unlimited (17.2 billion gigabytes, 16.8 million terabytes, or 16 exabytes)
- **Not uncommon for some database servers to have 10's to 100's of GB of RAM**
- **Rule of thumb is about 4-8GB RAM per core**
 - 4GB per core for Intel/AMD (System x)
 - 8GB per core for POWER5 and POWER6 (System p)



Agenda

- Performance Tuning Overview
- DB2 Architecture Fundamentals
- Hardware Planning Rules of Thumb
- **Database Design & Recommendations**

Default Database Factors that Affect Performance

- **“CREATE DATABASE <dbname>” results in:**
 - **Configuration Advisor** runs on database only with somewhat conservative options
 - **Automated runstats** is enabled
 - Adaptive **Self Tuning Memory** is enabled
 - **Unicode** database
 - **Automatic Storage** using a single location for containers (defined by DBM CFG DFTDBPATH)
 - Use ON clause to specify multiple paths
 - Database internal files and **log files** stored in same location as table space containers (defined by DBM CFG DFTDBPATH)
 - Use DBPATH ON clause to specify location separate from storage

Default Database Factors that Affect Performance - 2

- **“CREATE DATABASE <dbname>” results in:**
 - Default **page size of 4K** for default buffer pool and all table spaces
 - May not be appropriate for your environment
 - Default Table spaces:
 - SYSCATSPACE
 - TEMPSPACE1
 - USERSPACE1 (DMS Large, NO File system caching)
 - File system caching more suitable for LOB data.
 - **IBMDEFAULTBTP** buffer pool uses automatic tuning with an initial size of 1000 pages
 - May need multiple buffer pools
 - Recommend separate bufferpools for tempspace
 - **Currently Committed** Isolation Level is enabled
 - **DB2DETAILDEADLOCK** event monitor created
 - Drop it, performance overhead – (If new Evm. for locking is to be used)

Database Log Files

- **LOGPATH or NEWLOGPATH (DB CFG)** – defines location of the transaction log
 - update db cfg for <DBNAME> using NEWLOGPATH </path/.../>
 - Or use the DBPATH ON option of CREATE DATABASE
 - Never keep them in default location under the database path
 - Logs should not share disk devices with any other DB2 objects (e.g., table spaces)
 - Placed on dedicated storage with sufficient throughput capacity to ensure that a bottleneck will not be created
- **LOGBUFSZ (DB CFG)** – defines size of transaction logger internal buffer (in 4KB pages)
 - In general, a value of 256 (1MB) to 1000 (4MB) pages is a good range

Table Space Management – MANAGED BY

- **System Managed Space (SMS)**
 - Directory container, each object stored as separate in the file system
 - Access to data is controlled using standard I/O functions of the OS
 - Space not allocated by the OS until required
 - Low maintenance and monitoring
 - Useful for small temporary table spaces
- **Database Managed Space (DMS)**
 - Data stored in pre-allocated files or on raw device
 - Access to data controlled by DB2 and can bypass operating systems I/O functions, increasing performance
 - Increased maintenance and monitoring (though not much anymore)
 - Supports separating of table data, LOBs, and indexes into different table spaces for better parallelism
 - Supports use of LARGE table spaces (Large RIDs) for increased table capacity and better I/O

Automatic Storage

- Storage is completely managed by DB2 utilizing **best practices for optimally** managing table spaces
- **Default** is to use **Automatic Storage**
 - Also, by default, is to use a single path for the storage
- Use the ON clause of the CREATE DATABASE command to specify **multiple paths for DB2 to store table space** data on
 - Paths can be added or removed later
- DB2 creates and **extends containers** as needed up the limits imposed by the storage paths
 - Case for creating SMS table spaces is not very strong anymore

If Not Using Automatic Storage

- **Explicitly Use:**

- **User Table Spaces** – use DMS Large Table Spaces and separate table data, indexes, and LOBs
 - Separating data helps decrease I/O contention and increase prefetching performance
- **Temporary Table Spaces** - use SMS for small ones and DMS for large
 - DB2_SMS_TRUNC_TMPTABLE_THRESH can be set to a number of extents, avoids OS overhead for file creation

Table Spaces and page size

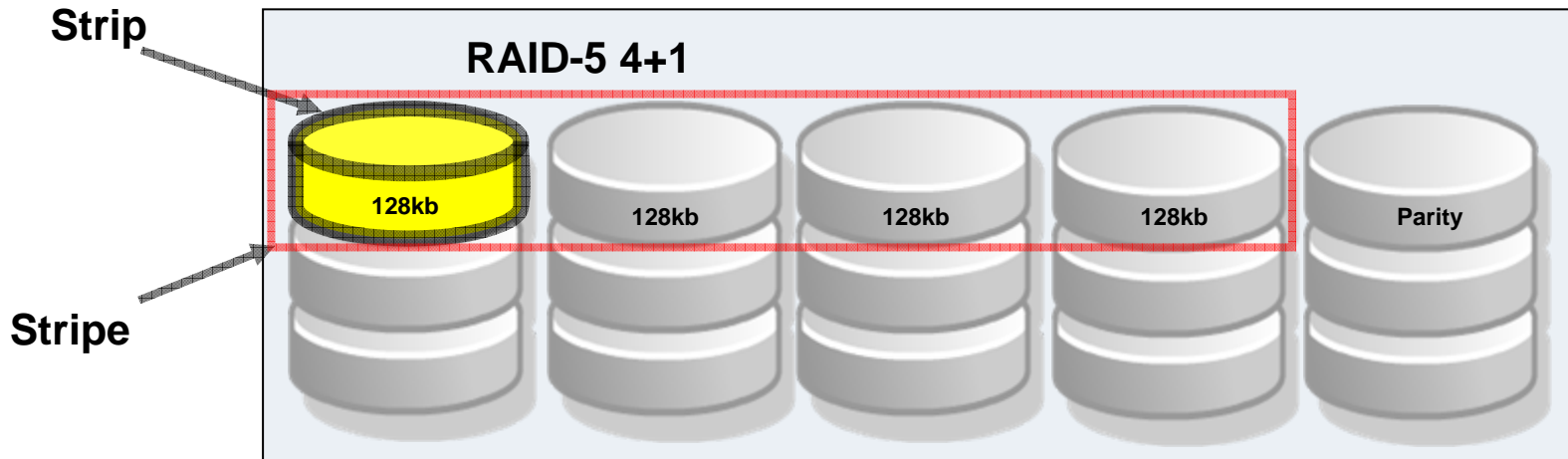
- **Large DMS** table space is the default since DB2 9
- Supports **Large RIDS**, which means it can fit about **2000** rows on a page (DMS only)
- Choose page size for table space based on the **average row size** for tables to be placed in the table space
 - Group tables with smaller rows in smaller page size table space
 - Group tables with larger rows in larger page size table spaces
- **Example: Table with row length of 10 bytes only is placed in a 32k tablespace. It will only use up to about 20k per page wasting 12k of space per page.**
 - Furthermore **data compression** may not be effective
- **There should be a system temporary table space for each page size used, to allow for more efficient REORGs**

Choosing Extent sizes

- **Specified at table space creation time and cannot be changed afterwards**
- **Choosing Extent size:**
 - **Default of 32 Pages** is typically the best choice in most cases
 - **BI/OLAP** type workloads typically benefit from larger extent sizes
 - Improves prefetching performance
 - **OLTP** workloads typically benefit from smaller extent sizes
 - Less wasted space in buffer pools and faster access

Set EXTENTSIZE to RAID stripe size – Best Practices

- The **EXTENTSIZE** for a table space should be set to the number of pages that are included in an entire RAID stripe



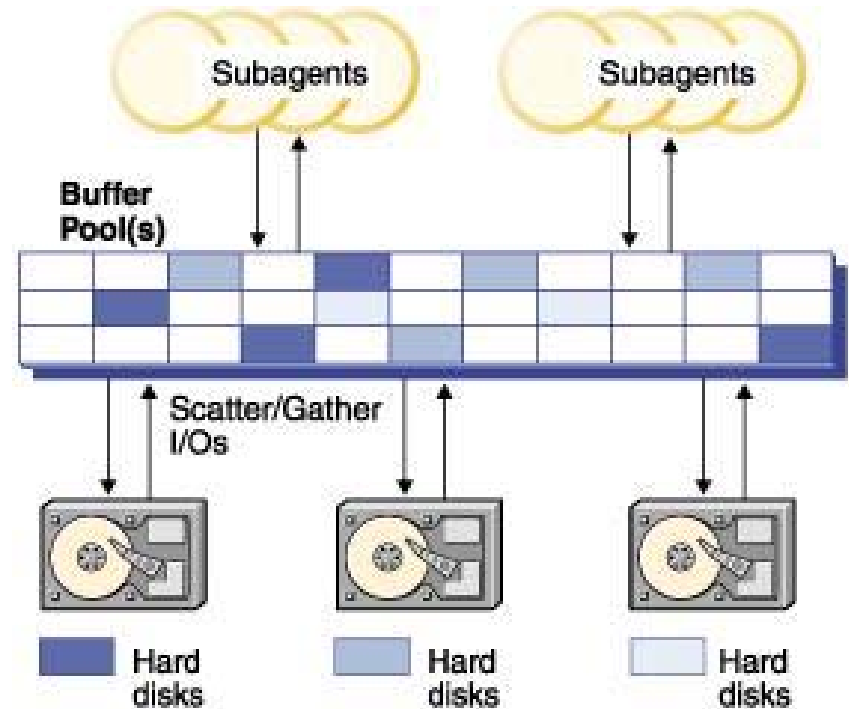
Stripe = 4 (# of data disk) x 128 (Strip Size KB) = **512 KB**

$$\mathbf{EXTENTSIZE} = \frac{\mathbf{Stripe (KB)}}{\mathbf{Page Size (KB)}}$$

<http://www.ibm.com/developerworks/db2/bestpractices/>

Buffer Pools – Major Performance “Knob”

- Area of main memory used to cache tables and indexes
- Purpose is to decrease time required to access data and reduce I/O contention
- Each database must have at least one buffer pool per page size
- Automatic buffer pool tuning with Self-Tuning Memory Manager (STMM)



Buffer Pool Creation

- **CREATE BUFFERPOOL <bpname>**
 - If STMM is active, creates a buffer pool with an initial size of 1000 pages with automatic tuning
 - NPAGES value of -2 in syscat.bufferpools view indicates that buffer pools is enable for automatic tuning
 - Configuration Advisor can be run to start with better initial values for all buffer pools in the database based on available memory (you can simply use the recommendations instead of automatically applying)
- **CREATE BUFFERPOOL <bpname> SIZE <size>**
 - Explicitly providing a size and omitting automatic clause creates a buffer pool that must be manually tuned
 - Non-Automatic Bufferpools can be ALTERED to become AUTOMATIC

Buffer Pool Design

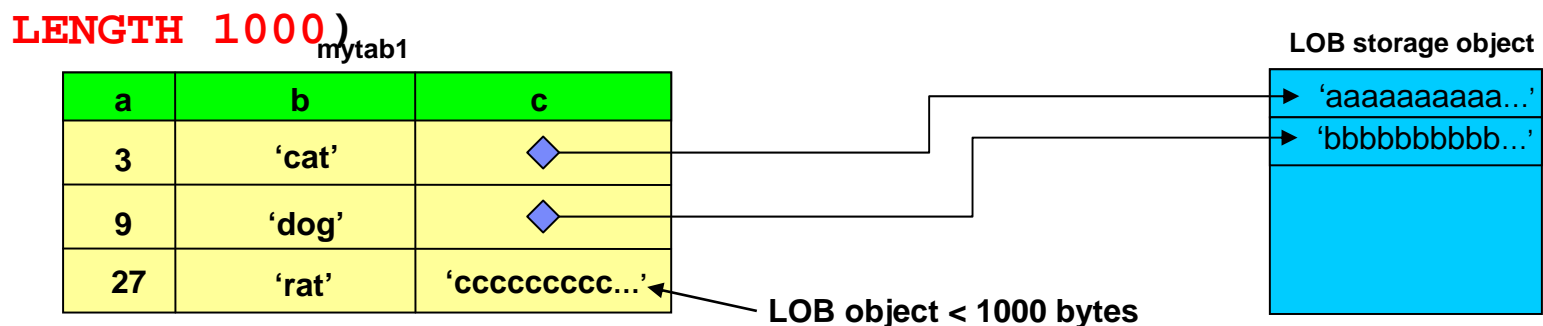
- For each page size used in the database, a buffer pool using the same page size must exist
- **OLTP** – Buffer pools typically consume 75% of available memory
 - Multiple buffer pools can increase performance by caching different amounts of data depending on the purpose of the associated table space(s)
- **BI/OLAP** – Buffer pools typically consume 50% of available memory (50% to SORTHEAP)
 - In most cases only one buffer pool is recommended (assuming one page size) for regular & user temporary tablespaces and one for system temporary tablespaces
 - However, in this case you cannot favor high-priority tables, nor discriminate from low activity tables
 - Exploit's DB2 efficient page management to maintain a high buffer pool hit ratio

Table Design

- **CREATE/ALTER TABLE options**
 - **COMPRESSION** can save up to 70% disk space and provide about 20% performance boost in I/O bound workloads and no performance difference for CPU-bound
 - Fewer I/O operations needed to retrieve same amount of data
 - Accessing data from disk is the slowest database operation
 - Deep Compression w/ Storage Optimization Feature Pack
 - **APPEND ON** for tables which have heavy inserts to avoid searching for free space and instead simply append the row to the end of the table
 - Information about free space on pages will not be kept
 - Alternatively, DB2MAXFSCRSEARCH can be used to improve insert performance
 - **PCTFREE** to maintain free space to help with future INSERTs, LOADs and REORGs
 - default is 10; try 20-35 for tables with clustered indexes and a heavy insert volume
 - If using with APPEND ON, set PCTFREE to 0
 - **PARTITION BY RANGE** to support fast roll-in and roll-out of data

DB2 – XML and LOB In-lining

- XML and LOBs are stored outside the base table in a separate storage object **or if sufficiently sized, can be stored in the formatted rows of the base table**
 - Depending on page size, the maximum length a LOB can be, to qualify for in-lining, is 32 669 bytes
- Descriptors stored in the base table rows are used to keep track of the associated XML/LOB data in the storage object
- Example:
 - `CREATE TABLE mytab1 (a int, b char(5), c clob INLINE LENGTH 1000)`



Why Use XML and LOB In-lining?

- **If a table possesses XML or LOB data that can be in-lined, there are considerable benefits with respect to performance and storage use**

Performance

- In-lined XML or LOB can be buffered in the bufferpool
- LOB data can be compressed saving on IO costs.
- XML data can be compressed in XDA or when in-lined

Storage

- Overall Storage allocated to the storage object is reduced by in-lining XML/LOB data (though base table storage increases)
- In-lining small XML/LOBs can result in a noticeable decrease in net total storage since the decrease in storage size is greater than the increase in base table storage size

TPoX 2.0 testing (August 2009)

- Open Source Benchmark: **TPoX**
- (Transaction Processing over XML Data)
- <http://tpox.sourceforge.net/>
- 1TB raw data, 360M XML Documents
- 70% Queries, 30% Insert/Update/Delete,
- 200 concurrent users
- **The Score Card**
- **DB2 Compression Ratio: 64%**
- **Mixed Workload: 6,763 Transactions per second**
- 1.48x speed-up: Intel® Xeon® 7300 to Intel® Xeon® 7400 Processor
- Insert-only workload (600 users): 11,900 inserts per second Ingestion rate: ~100GB/hour
- All tests performed by Intel at Intel Labs
- **The System Under Test**
- IBM DB2 pureXML 9.5 FP2
- Linux64, SLES 10
- Intel® Xeon® 7400 Processor Server
- 4 CPUs, 6 cores/CPU, 2.67 GHz
- 64GB memory
- 135 disk
- **6,763 TTPS at 95% CPU**
- Intel® Xeon® Processor 7300 Series and Intel® Xeon® Processor 7400 Series are trademarks of Intel Corporation in the U.S. and other countries.
pureXML Presentation Notebook Page 572 of 704

TPoX Performance achievements rely on Compression

- **Storage Consumption and Compression Results**
- **Since the security table is very small (20,833 documents) we examined the space consumption and compression ratio mainly for the two large tables, custacc and order (see Table 3). The 60 million custacc documents are compressed by 64 percent and require 121.4GB in a DB2 table space. The 300 million order documents are compressed by 57 percent and occupy 269.2GB in DB2. Including all data and indexes, the final database size was about 440GB. XML in-lining and compression were critical to avoid I/O bottlenecks.**

Index Design – Best Practices

- **Avoid over indexing!** Excessive number of indexes can slow down update operations and consume extra space
- **It is sometimes possible to have one larger (composite) index that will cover several queries**
 - For composite indexes, place the column which is referenced most in queries first in the definition
- **Columns with high cardinality are good candidates for indexing**
- **Avoid adding an index which is similar to a pre-existing index**, as it creates more work for the optimizer when generating an access plan and will slow down update operations; instead alter the pre-existing index to contain additional columns
- **Use INCLUDE clause to add columns referenced in query to index**
 - Used with unique indexes only and additional columns are not used in enforcing uniqueness
- **Remove unused or no longer used indexes.**
 - Check the SYSCAT.INDEXES.LASTUSED column for possible candidates (v9.7fp1)

Clustering Indexes

- The Design Advisor can be used to recommend clustered indexes
- Clustering indexes can be created to order the rows in the table in the **same physical order** of the **desired result set**
 - One clustering index per table at any one given time
- They are created using the **CLUSTER** option of the CREATE INDEX statement
- For best performance, create the index over small data types (like integer and char(10)), unique columns, and columns most often used in range searches.
- Set the **PCTFREE** CREATE/ALTER TABLE option somewhere between 15-35 for tables with clustered indexes to ensure that the clustered indexes do not become too fragmented

Multidimensional Clustering (MDC)

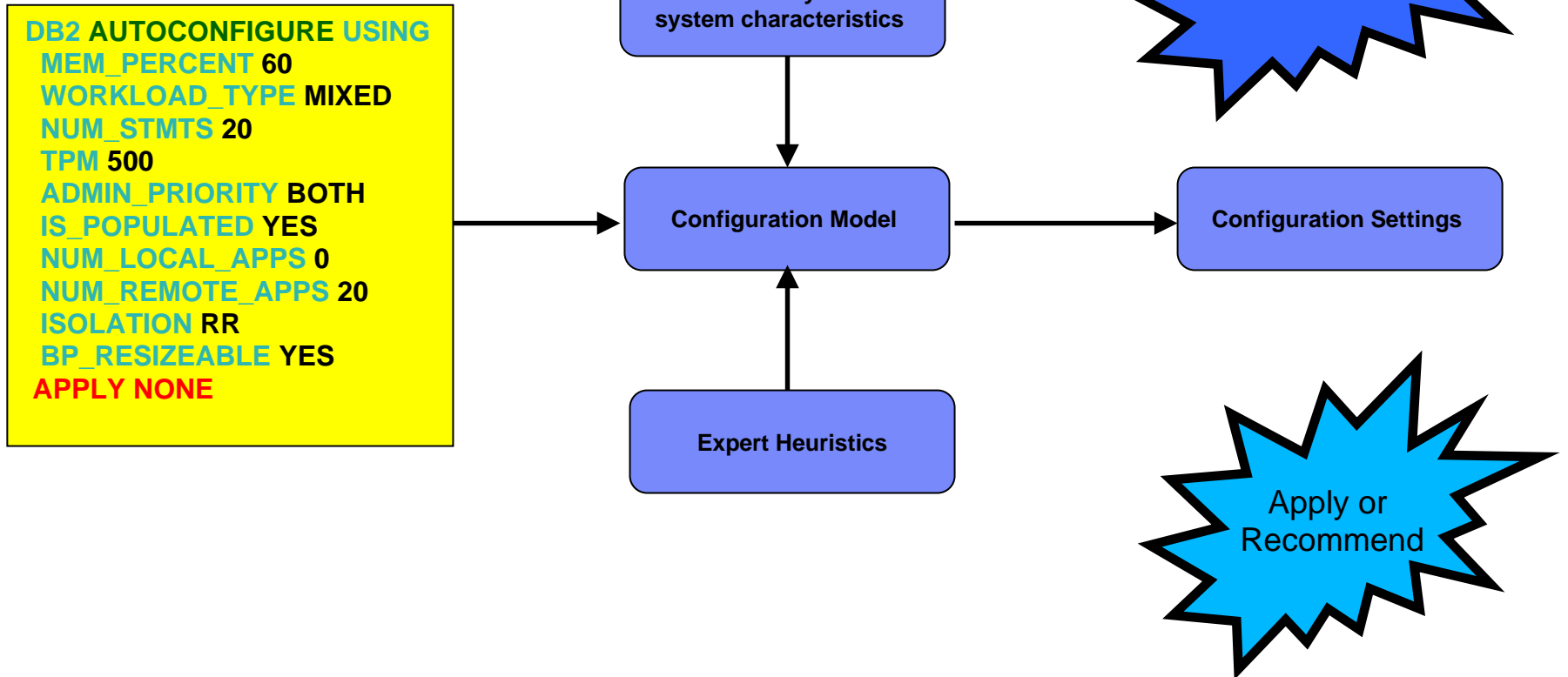
- **The Design Advisor can be used to recommend conversion to MDC tables**
- **Provides for flexible, continuous, and automatic clustering of data along multiple dimensions**
- **Physically clusters the table's data along multiple dimensions simultaneously, which is similar to having multiple, independent clustered indexes on the table**
- **Typically used to help speed up the performance of complex queries on large tables in BI/OLAP environments**
- **Best candidates for an MDC:**
 - **Non-unique columns**, as the dimensions would cause the table to be unnecessarily large
 - For best performance, you will want at least enough rows to fill a full block of each cell, which is equal to the extent size of the table space that the table resides in.

Materialized Query Tables (MQTs)

- The Design Advisor can be used to recommend MQTs
- MQTs are **summary tables** used to improve queries which use the GROUP BY, GROUPING, RANK, or ROLLUP **OLAP functions**
- **Transparent** to the user and DB2 chooses when to use them for optimization purposes
- Used by DB2 to internally maintain summarized results of the required grouping, which allows a user to access a maintained grouping instead of reading through what could be multiple GBs of data to find the answer
- Use **“refresh deferred”** otherwise insert, update, delete operations could take longer

AUTOCONFIGURE Example

Rerun after database has been fully created and populated!



Thank You



DB2 Performance Monitoring Essentials



Welcome

*Data Management Emerging Partnerships and Technologies
IBM Toronto Lab*

Spring 2010

DB2 Performance Monitoring Essentials

Agenda

- **Introduction**
- **Monitoring Interfaces available with DB2**
- **Essential Monitoring Targets**
- **Monitoring tools**

DB2 Performance Monitoring Essentials

Agenda

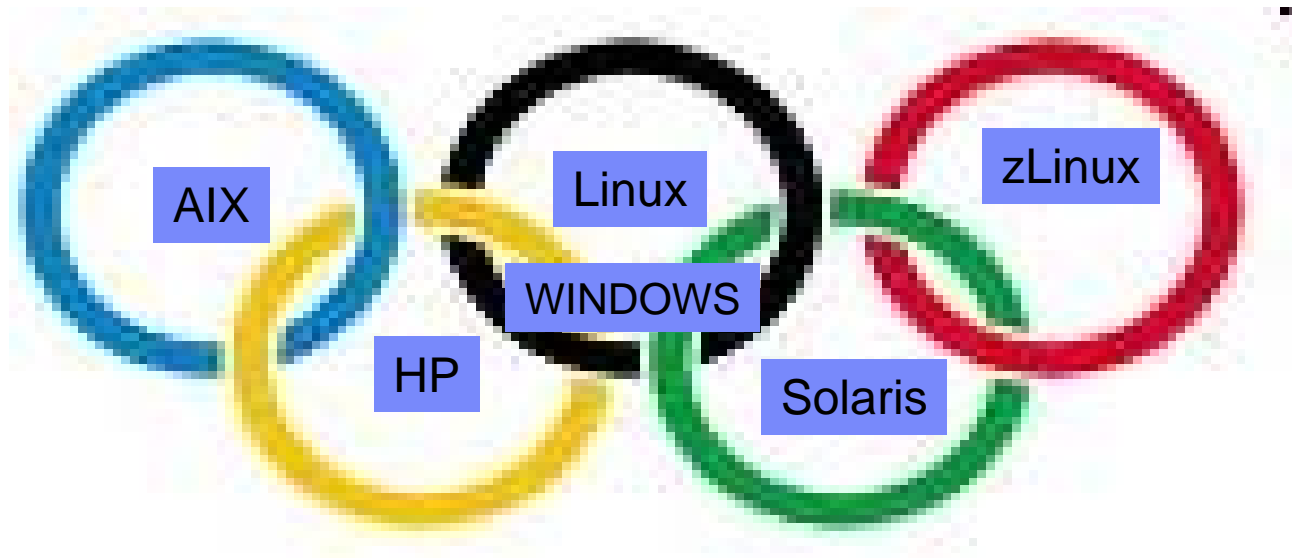
- **Introduction**
 - Learning objectives
 - Before we get started
- Monitoring Interfaces available with DB2
- Essential Monitoring Targets
- Monitoring tools

Module Learning Objectives

- Knowledge of key DB2 performance monitoring targets
- **UNDERSTAND WHAT MONITORING OPTIONS ARE AVAILABLE WITH DB2 AND HOW TO GET STARTED USING THEM EFFECTIVELY.**
- DB2'S new monitoring framework and basic use of new monitoring interfaces
- **KNOWLEDGE OF BASIC SYSTEM PERFORMANCE MONITORING TOOLS FOR LINUX UNIX AND WINDOWS SYSTEMS -TO IDENTIFY SYSTEM RESOURCE BOTTLENECKS**
- db2top tool & Technology Explorer 3.5

DB2 LUW

- Runs on x Operating Systems
- Runs on x Hardware vendors



DB2 is DB2 all functionality at same level for all platforms

DB2 RUNS ON MULTIPLE PLATFORMS

- **CHALLENGE - OS PERFORMANCE TOOLS/PLATFORM**

- Good News: Many OS performance tools very similar across platforms
- More Good News: OS Architectures the same at abstract level
 - Most tuning methods/tools yield like results on all platforms
- More Good News: DB2 monitoring elements and interfaces are the same on all platforms.
- Best News: DB2 Autonomics like STMM reduce the need for production tuning on all platforms.



Good Foundation - KNOW YOUR SYSTEM

What OS level is running ?

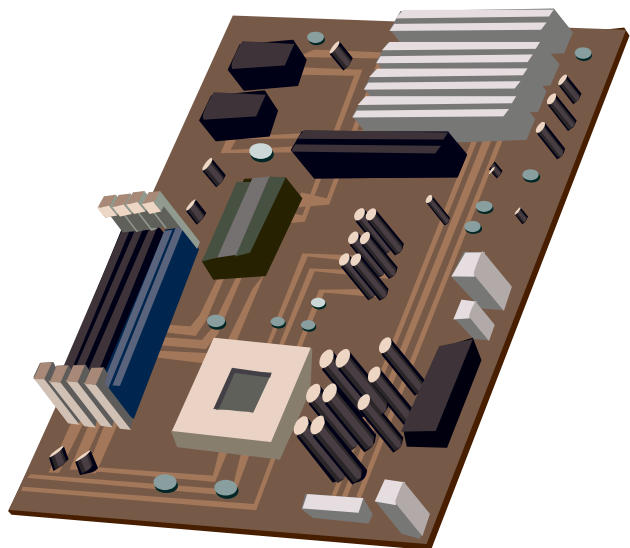
How much swap space is set ?

What are my processors speed ?

How many processor cores are assigned ?

What storage is present/available on the system?

How much memory is available on the system after startup?



Discovery Tools & Commands

- db2pd –osinfo (All)
- systeminfo (Windows)
- free (Linux)
- bootinfo (AIX)
- yast – (Linux)
- smit – (AIX)
- lvm – (AIX, Linux)

DB2 Performance Monitoring Essentials

Agenda

- Introduction
- **Monitoring Interfaces available with DB2**
 - New Monitoring Framework & New Monitoring Table Functions & Views
 - Snapshot Monitoring
 - Event Monitoring
 - db2pd
- Essential Monitoring Targets
- Monitoring tools

Monitoring - Interfaces

- **New Relational Monitoring Interfaces**

- Introduced 9.7
- Offers a Lightweight based method of collecting DB2 metrics with SQL
- Evolved as complimentary extension to Work Load Management Implementation
- Turn on collection at database level (more granularity available with WLM feature)

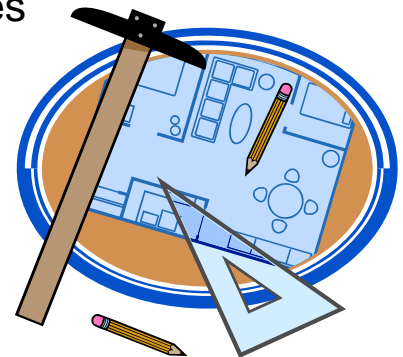
- **Snapshot Monitor**

- Switch Based Monitoring
- Services snapshot command, Health Monitor metrics, most Event Monitors, Administrative Views and Table Functions.
- Integrating/Joining Snapshot Monitor and WLM Monitoring data

- **Event Monitors**

- Event Monitors usage
- New Event Monitor types replacing deprecated Event Monitor types

- **DB2 Problem Determination Tool a.k.a db2pd**

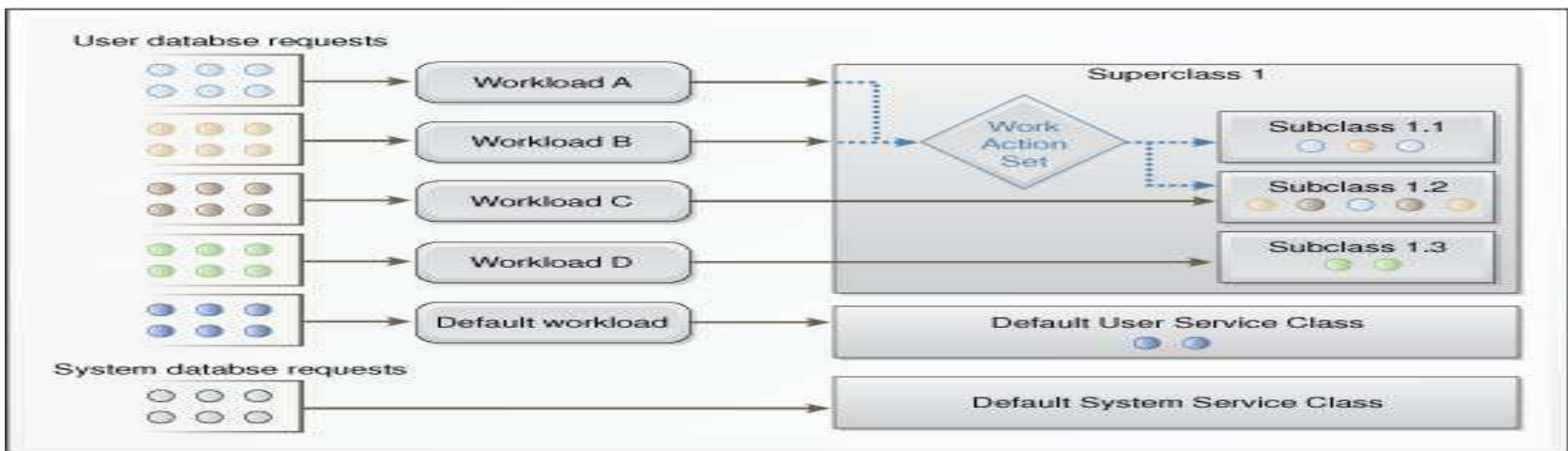


Snapshot Monitoring vs. New Monitoring Framework

- **Many monitoring elements are nominally the same**
 - Examples: pool_data_p_reads, lock_waits
- **More monitoring elements in new framework & more to come**
- **More granularity in new framework (actual vs. wait time)**
- **Overall less overhead with new monitoring framework**
- **Avoid mixing frameworks when monitoring a system**
 - You will incur overhead from both

A Brief Overview of Workload Management in DB2

- **Workload** - an object that is used to properly identify incoming work based on its source so that it can later be properly managed
- **Service Class** - defines an execution environment in which the work can run. This environment can include available resources and various execution thresholds.
- **When you define a workload, you must indicate the service class where work associated with that workload runs.**



New Monitoring Framework - Three Focus Areas

- **System**

- Provide total perspective of application work being done by database system
- Aggregated through the WLM infrastructure

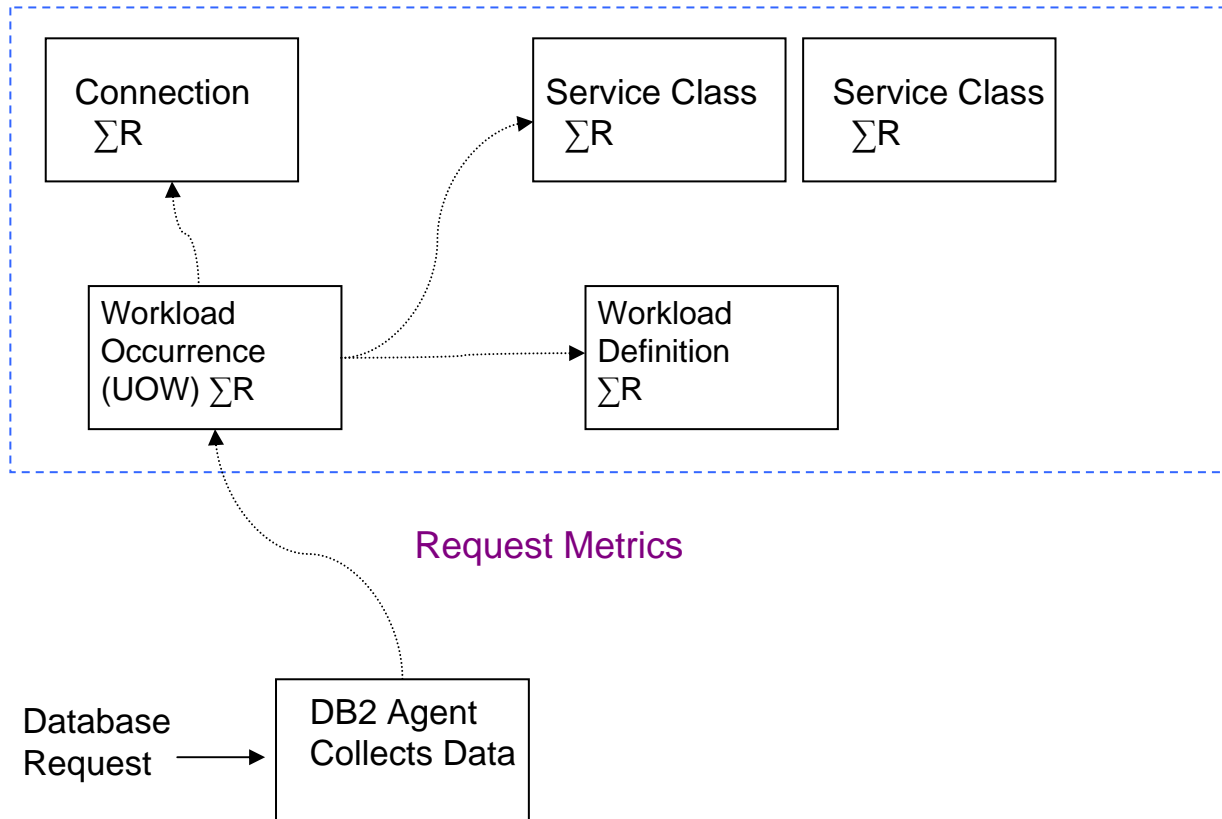
- **Data objects**

- Provide perspective of impact of application work on data objects
- Aggregated through data storage infrastructure

- **Activity**

- Provide perspective of work being done by specific SQL statements
- Aggregated through the package cache infrastructure

In-Memory Metrics: System Perspective



Legend

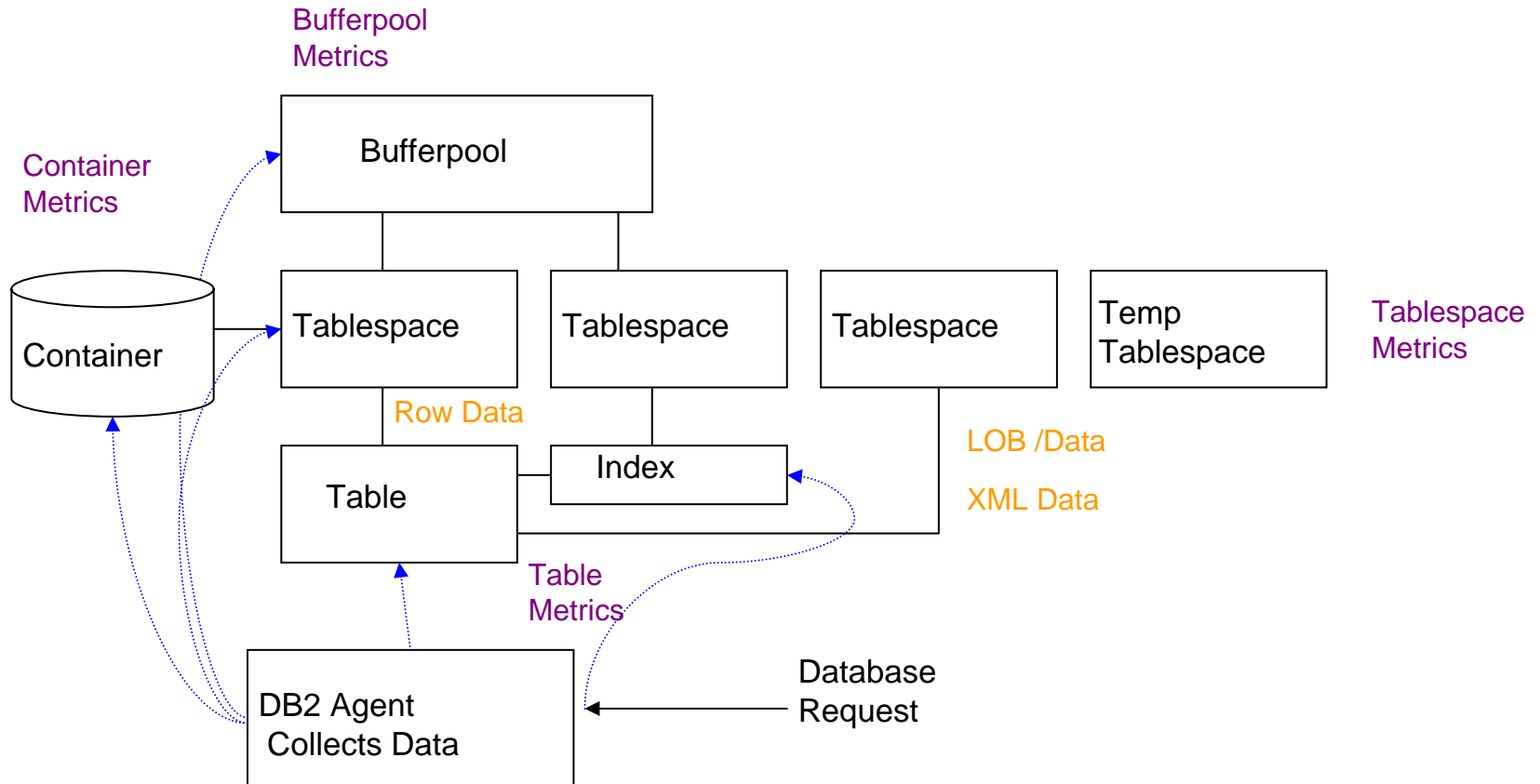
ΣR = Accumulation of request metrics collected by agent

Access Points: System Perspective

- **MON_GET_UNIT_OF_WORK**
- **MON_GET_WORKLOAD**
- **MON_GET_CONNECTION**
- **MON_GET_SERVICE_SUBCLASS**

- **Also provide interfaces that produce XML output:**
 - **MON_GET_UNIT_OF_WORK_DETAILS**
 - **MON_GET_WORKLOAD_DETAILS**
 - **MON_GET_CONNECTION_DETAILS**
 - **MON_GET_SERVICE_SUBCLASS_DETAILS**

In-Memory Metrics: Data Object Perspective



Access Points: Data Object Perspective

• Data Object Monitoring Table Functions

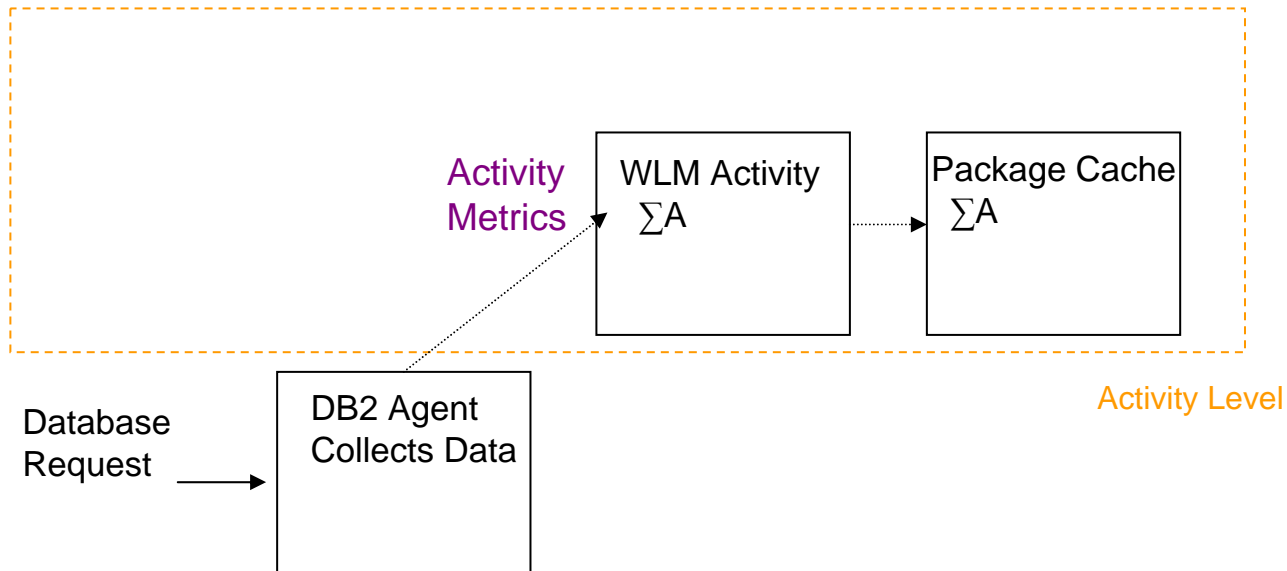
- MON_GET_BUFFERPOOL
 - >>-MON_GET_BUFFERPOOL--(--bp_name--,--member--)-----
 - Monitor bufferpool efficiency, hit ratio, activity
- MON_GET_CONTAINER
 - >>-MON_GET_CONTAINER--(--tbsp_name--,--member--)-----
 - Monitor container activity, rank, enumerate
- MON_GET_INDEX
 - >>-MON_GET_INDEX--(--tabschema--,--tablename--,--member--)-----><
 - Monitor index usage, like number of index scans, how many scans are index only scans
- MON_GET_TABLE (always collected regardless of db cfg)
 - >>-MON_GET_TABLE--(--tabschema--,--tablename--,--member--)-----><
 - Monitor activity on tables, reads, updates, inserts, overflow activity
- MON_GET_TABLESPACE
 - >>-MON_GET_TABLESPACE--(--tbsp_name--,--member--)-----><
 - Monitor tablespace activity, read and writes, bufferpool activity

New monitoring Streams

- Request Metrics
- Activity Metrics

→ **Object Metrics**

In-Memory Metrics: Activity Perspective



Legend

ΣA = Accumulation of metrics from activity execution portion of request

Access Points: Activity Perspective

- **MON_GET_PKG_CACHE_STMT**
 - Both static and dynamic SQL
- **MON_GET_ACTIVITY_DETAILS (XML)**

Monitor Views Added with v9.7 Fix Pack1

- **MON_BP_UTILIZATION**
 - BP Hit Ratios calculated as part of result set (Data, Index, and XDA)
- **MON_CONNECTION_SUMMARY**
 - Includes new elements total_app_commits, total_app_rollbacks
- **MON_CURRENT_SQL**
 - Monitor currently executing SQL
- **MON_CURRENT_UOW**
 - Identify long running units of work and related activity
- **MON_DB_SUMMARY**
 - High level, aggregated metrics, percentage breakdowns, wait vs. active, totals
- **MON_LOCKWAITS**
 - List applications currently waiting to acquire locks, holding applications, elapsed time, and statements
- **MON_PKG_CACHE_SUMMARY**
 - Aggregate metrics for statements both dynamic and static currently in package cache
- **MON_SERVICE_SUBCLASS_SUMMARY**
 - Returns key metrics for all service subclasses in the currently connected database
- **MON_TBSP_UTILIZATION**
 - List tablespace information, state, high watermark, and hit ratios
- **MON_WORKLOAD_SUMMARY**
 - High level, aggregated metrics, percentage breakdowns, wait vs. active, totals

What if I don't have WLM?

- **What if I don't have WLM?**
 - Have Enterprise Server Edition?
 - License WLM with Enterprise Server Edition?
- **There is always...**
 - The default workload and service class
 - All DB2 monitoring functions & event monitors
 - All DB2 monitoring db cfg parms
- **But you can't...**
 - Define new WLM objects (workloads, thresholds, service classes, etc.)
 - Apply priorities (CPU, I/O, bufferpool) to different work
 - Exploit the WLM benefits of granular workload based monitoring and resource control

Comparing the Overhead (OLTP)

- **System Monitor**
 - All switches on = @6%
- **“In-Memory” Metrics**
 - All possible metrics active = @3%

New Monitoring Framework – Database Configurations

- **New DB cfg parameters - ON by default**
- **Collection clauses for further granularity on WLM constructs**
- **Control collection of data for corresponding table functions**
 - **MON_REQ_METRICS** – DB level: Collect request metrics (UOW, Connection, Service Subclass, Workload)
 - **COLLECT REQUEST METRICS** – service superclass level
 - DB level overrides workload level designation when enabled
 - **MON_OBJ_METRICS** – DB level: Collect Object level Metrics (Bufferpool, Container, Index, Table, TableSpace)
 - **MON_ACT_METRICS** – DB level: Collect Activity metrics (Activity Details - xml output, Package Cache)
 - **COLLECT ACTIVITY METRICS** – workload level
 - DB level overrides workload level designation when enabled

Monitoring – New Monitor Table Functions (Examples)

Example: List containers on all database members that have the [highest read time](#).

```
SELECT VARCHAR(CONTAINER_NAME,70) AS CONTAINER_NAME,  
VARCHAR(TBSP_NAME,20) AS TBSP_NAME, POOL_READ_TIME  
FROM TABLE(MON_GET_CONTAINER('',-2)) AS t  
ORDER BY POOL_READ_TIME DESC
```

Example: List top 10 consumers of [cpu time](#) from package cache.

```
SELECT MEMBER,  
SECTION_TYPE ,  
VARCHAR(STMT_TEXT,50) AS STATEMENT,  
NUM_EXEC_WITH_METRICS AS NUMEXEC,  
TOTAL_CPU_TIME/NUM_EXEC_WITH_METRICS AS  
AVG_CPU_TIME,EXECUTABLE_ID  
FROM TABLE(MON_GET_PKG_CACHE_STMT ('D', NULL, NULL, -2)) AS T  
WHERE T.NUM_EXEC_WITH_METRICS <> 0 ORDER BY AVG_CPU_TIME DESC FETCH  
FIRST 10 ROWS ONLY;
```


Snap Shot Monitoring

- **Snapshot based monitoring - still viable**
- **Controlled with monitor switches**
 - In dbm configuration or at session
 - Can reset counters at session level – easy to get current results
 - Higher overhead than new monitoring framework
- **Get snapshot command (CLP)**
 - For database manager, database, bufferpools, locks, dynamic SQL, applications, tables, tablespaces, etc. (the works)
 - Formatted text output by default – “great for greppers”
- **Administrative Views and Table Functions +**
 - Access Snapshot data via views and table functions
 - Easy application interface to use
 - Easy to store in database relational tables

Monitor switch control – for Snapshot Monitoring

- **Monitor Switches globally turned on/off in DBM configuration, locally can be controlled at session.**
 - **get dbm cfg** – to see global settings
 - **update dbm cfg using DFT_MON_monitorswitch on|off**
 - Example: Turn OFF Monitor switch for locking
 - **update dbm cfg using dft_mon_lock off** - restart instance
 - **get snapshot for database on yourdb** – check the “Time database waited on locks” to see that this information is not available.
 - Example: Turn ON Monitor switch for locking at session level
 - **get monitor switches** – to see current settings, locking off
 - **update monitor switches using lock on**
 - **get snapshot for database on yourdb** – check the “Time database waited on locks” to see that this information is now available.
- **Reset counters for a session to isolate metrics**
 - **reset monitor all** – resets all switches at session level
 - otherwise counters start at database startup

Snapshot monitoring Examples - database

- **GET SNAPSHOT**

- get snapshot for database on db2pt97

- **ADMINISTRATIVE VIEW**

- select * from sysibmadm.snapdb

- **ADMINISTRATIVE TABLE FUNCTION**

- select * from table(snap_get_db_v95(' ', -2))

Event Monitoring - more interfaces

- **Event Monitors are used to capture information when an activity occurs as defined by and in an event monitor.**
 - **Created via SQL-DDL, definitions are stored in system catalog tables.**
 - **Output can be directed to file, table, or pipe***
 - **Types of events include: Locking, UOW, statements (SQL), Connections and Tables**
 - **New Monitoring Framework being used for new Event Monitors, existing Event Monitors are being deprecated**
 - DEADLOCKS + DETAILED DEADLOCKS → LOCKING
 - TRANSACTION → UNIT OF WORK (UOW)
- * New event monitors only write output to Unformatted Event Tables*

Event Monitor Example: Statements

```
CREATE EVENT MONITOR GET_SQL_MYAPP  
FOR STATEMENTS  
WHERE (APPL_ID = 'myapp')  
WRITE TO TABLE  
BUFFERSIZE 8 BLOCKED AUTOSTART ;
```

- Capture all statements where `appl_id = myapp`
- Use buffers to reduce overhead
- Place tables in dedicated tablespace to reduce overhead

- Use db2evtbl or Control Center to generate ddl for tables
- `set event monitor state 0` – to deactivate
- Use SQL to query results, find costly and error prone SQL Statements
- Event monitor definitions and status found in SYSCAT schema in EVENTS, EVENTMONITORS, and EVENTTABLES views.

Create New Event Monitor For Locking

- **DB configurations for collecting locking metrics (defaults)**

Lock timeout events	(MON_LOCKTIMEOUT) = NONE
Deadlock events	(MON_DEADLOCK) = WITHOUT_HIST
Lock wait events	(MON_LOCKWAIT) = NONE
Lock wait event threshold	(MON_LW_THRESH) = 5000000

- Update default configurations:

- db2 update db cfg for sample using mon_lockwait with_hist
- db2 update db cfg for sample using mon_lw_thresh 3000000
- db2 update db cfg for sample using mon_locktimeout hist_and_values
- db2 update db cfg for sample using mon_deadlock without_hist

without_hist - collect basic event information

with_hist - collect up to 250 activities within same unit of work

hist_and_values – collect activities and values

3000000 – micro seconds, lock wait condition exists this long before lockwait

Create New Event Monitor For Locking

Create lock event monitor and disable old monitor

- Deactivate & Drop DB2DETAILDEADLOCK event monitor

```
SET EVENT MONITOR DB2DETAILDEADLOCK state 0  
DROP EVENT MONITOR DB2DETAILDEADLOCK
```

- Create event monitor, direct output to table/tablespace

```
CREATE EVENT MONITOR lockevmon FOR LOCKING - ALL LOCKING types  
WRITE TO UNFORMATTED EVENT TABLE - NEW TABLE TYPE must format  
(TABLE MYLOCKS IN TS32K PCTDEACTIVATE 75) - EVM deactivates at 75%
```

Create New Event Monitor For Locking – Cont.

- **Format Unformatted Event Table**

- db2evmonfmt
 - java tool found in samples\java\jdbc (must compile to class file)
 - java **db2evmonfmt** -d *db_name* -ue *table_name* -ftext -u *user_id* -p *password*
 - XML or Formatted text output
- EVMON_FORMAT_UE_TO_XML (table function)
 - Call to format Unformatted Event Table into XML document
- EVMON_FORMAT_UE_TO_TABLES (procedure)
 - Call to format Unformatted Event Table into regular Table
 - Calls evmon_format_ue_to_xml table function
 - Default tables (defined in ../sqllib/misc/DB2EvmonLocking.xsd)
 - SYSIBMADM.LOCK_EVENT
 - SYSIBMADM.LOCK_EVENT_PARTICIPANTS
 - SYSIBMADM.LOCK_PARTICIPANT_ACTIVITIES

db2pd – The DB2 Problem Determination Tool

- **db2pd**
 - Diagnose lockwaits
 - Map application to dynamic SQL statement
 - Monitor Memory usage
 - Map application usage to tablespace
 - Monitor log usage
 - Includes WLM metrics
 - Assess Index usage
 - And more...

db2pd example: Mysterious slowdown every afternoon

- Users report an overall slowdown every afternoon at 1pm, claiming nothing has changed
- Performing runstats does not solve the problem, eliminating out of date statistics as a culprit
- Checked different database indicators such as buffer pool hit ratios, lock waits etc. No bottlenecks found
- “vmstat” shows high **user cpu** and first clue in the mystery:

```
procs -----memory----- ---swap-- -----io----- -system--      -----cpu-----
r b swpd free buff cache si so bi bo in cs us sy id wa st
2 0 520800 60916 3800 438032 32 132 393 183 280 451 53 8 33 6 0
1 0 520800 60636 3808 438032 0 0 0 48 259 376 57 4 37 2 0
3 0 520800 60652 3824 438032 0 0 0 8 263 393 54 8 34 4 0
1 0 520800 60776 3832 438032 0 0 0 26 258 371 60 1 39 0 0
```

Mysterious slowdown every afternoon

- **User cpu is high which indicates that it is due to some activity in the application space (DB2) and not in the operating system**
- **Use “db2pd –edus” to see cpu usage by each thread:**

EDU ID	TID	Kernel TID	EDU Name	USR (s)	SYS (s)
.....					
21	2946493344	10227	db2agent (DS2)	2.200000	0.900000
20	2947541920	10226	db2lfr (DS2)	0.000000	0.000000
19	2948590496	10225	db2loggw (DS2)	1.700000	0.600000
18	2949639072	10224	db2loggr (DS2)	2.170000	1.680000
17	2950687648	10214	db2agent (DS2)	29.310000	4.600000
16	2951736224	10201	db2resync	0.000000	0.000000
15	2952784800	10200	db2tcpcom	0.310000	0.450000

EDU id 17 stands out because it is using significantly more user cpu

Please note: These values are cumulative over the life of EDU so it may be important to know the relative rate change of the CPU usage

Mysterious slowdown every afternoon

- Use “**db2pd –db ds2 –applications**” to find out the application being served by EDU ID 17:

Address	App- Handl	[nod-index]	Num- Agents	Coor- EDUID	Status	L-Anch- ID	L-Stmt- UID	Appid
0x20A55FD0	12	[000-00012]	1	30	ConnectCompleted	0	0	*LOCAL.DB2.091015124129
0x20A50060	11	[000-00011]	1	29	ConnectCompleted	0	0	*LOCAL.DB2.091015124128
0x20A45FD0	10	[000-00010]	1	28	ConnectCompleted	0	0	*LOCAL.DB2.091015124127
0x20A40060	9	[000-00009]	1	27	ConnectCompleted	0	0	*LOCAL.DB2.091015124126
0x20A06C00	8	[000-00008]	1	26	ConnectCompleted	0	0	*LOCAL.DB2.091015124125
0x20A00060	7	[000-00007]	3	17	PerformingLoad	937	1	*LOCAL.johnh.091015130001

** *Some columns are not shown to fit the output*

- Application handle 7 is being serviced by EDU id 17.
- Status filed shows that it is performing a LOAD which explains the problem.
- Novice user johnh has loads scheduled to run at 1pm.
- Load is a resource intensive operation and running it during production hours causes everything else to run slower!

DB2 Performance Monitoring Essentials

Agenda

- Introduction
- Monitoring Interfaces available with DB2
- **Essential Monitoring Targets**
- Monitoring tools

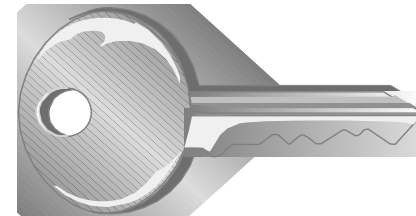
Essential Monitoring Targets

- **Track key performance indicators**
 - Practical approach, too many = diminished returns
 - Key DB2 monitoring elements as indicators
 - Stand Alone Elements
 - Calculate Ratios
 - DB2 monitoring elements can be captured from:
 - Snapshot Monitoring
 - Switch based
 - Some CPU overhead (1-10%)
 - Easy to reset counters
 - New Monitor Table functions and Views via SQL
 - SQL based, easy to tabularize
 - Monitoring elements queried from memory, lower overhead than Snapshot based monitoring

Essential Monitoring Targets – Database Level

- **Key areas**

- TOTAL TRANSACTIONS
- BUFFER POOL HIT RATIOS (DATA, INDEX, TEMP)
- READS AND READ EFFICIENCIES
- SORTING
- LOCKS
- PAGE CLEANING
- PACKAGE CACHE CAPACITY
- TRANSACTION LOGS

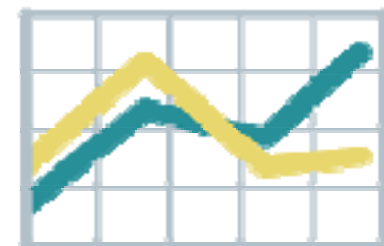


- **Save data to tables**

- Aggregate, differentiate, interpolate, extrapolate → modulate

- **Look for trends**

- Avoid catastrophes
- Easier to fix before broken



Essential Monitoring Targets – Total Transactions

- **The number of transactions executed by applications**
 - Use snapshot monitoring to retrieve
 - Database Level, Application Level
 - COMMIT_SQL_STMTS + ROLLBACK_SQL_STMTS
 - Comparable WLM elements (v9.7 fp1)
 - TOTAL_APP_COMMITS + TOTAL_APP_ROLLBACKS
 - Useful for creating key ratios like reads/commit
 - Adds element of “relativity” to monitoring

Essential Monitoring Targets – BP Hit Ratio

- **Buffer pool hit ratios**
 - With **MON_BP_UTILIZATION** monitor view

Hit Ratio for All Data, Index, and XDA for **each** Bufferpool

```
SELECT  
SUBSTR(bp_name ,1,30) as BPNAME,  
data_hit_ratio_percent as DATA_HR,  
index_hit_ratio_percent as INDEX_HR,  
xda_hit_ratio_percent as XDA_HR  
FROM SYSIBMADM.MON_BP_UTILIZATION
```

Essential Monitoring Targets – BP Hit Ratio

- **Buffer pool hit ratios, measured separately for data, index, and temporary data**

- **With BP_HITRATIO** administrative view

Example: Returns bufferpool hit ratios, including total hit ratio, data hit ratio, XDA hit ratio and index hit ratio, for all bufferpools and all database partitions in the currently connected database.

```
SELECT SUBSTR(db_name,1,8) AS DB_NAME, SUBSTR(bp_name,1,14) AS BP_NAME,  
total_hit_ratio_percent as TOTAL_HR, data_hit_ratio_percent as DATA_HR,  
Index_hit_ratio_percent as INDEX_HR, xda_hit_ratio_percent as XDA_HR,  
dbpartitionnum as DBPARTNUM  
FROM SYSIBMADM.BP_HITRATIO  
ORDER BY dbpartitionnum
```

- **SNAP_GET_BP_V95()** administrative table function
 - Use to aggregate results from all partitions or report on single partition
- **GET SNAPSHOT FOR DATABASE**
 - Use from command line

Essential Monitoring Targets – I/O efficiency

- The number of Rows Read per Transaction
 - With **SYSIBMADM.SNAPDB**

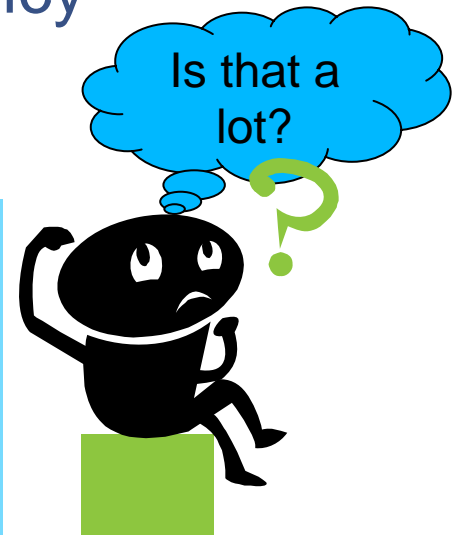
OLTP

< 10 Excellent

10-20 Very Good

20-40 Fair

> 50 Tune



```
SELECT VARCHAR(db_name,10),
  CASE WHEN (commit_sql_stmts + rollback_sql_stmts) > 0
  THEN DEC(((rows_read) / commit_sql_stmts + rollback_sql_stmts),13,2)
  ELSE NULL
  END AS READS_PER_TRANSACTION,
  rows_read as ROWS_READ,
  commit_sql_stmts + rollback_sql_stmts as TOTAL_TRX,
  db_conn_time as FIRSTDB_CONN,
  last_reset as LAST_RESET
FROM SYSIBMADM.SNAPDB;
```

Essential Monitoring Targets – I/O Efficiency

- The number of Rows Read to Rows Returned
 - With **MON_PKG_CACHE_SUMMARY** view

```
select varchar(stmt_text,1024),  
rows_read_per_rows_returned  
FROM  
SYSIBMADM.MON_PKG_CACHE_SUMMARY  
WHERE rows_read_per_rows_returned > 50  
ORDER BY rows_read_per_rows_returned DESC  
FETCH FIRST 20 ROWS ONLY
```



Essential Monitoring Targets - Sorting

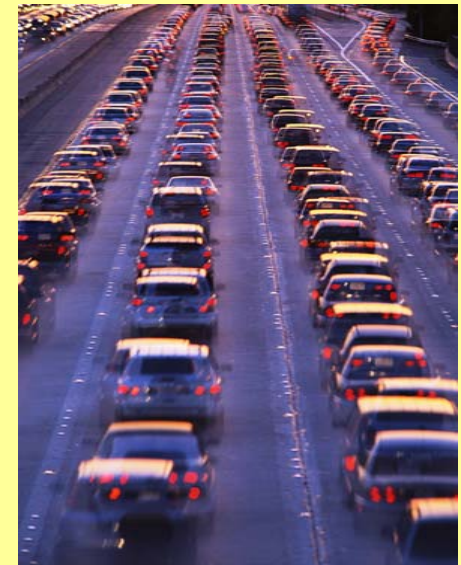
- **Sorting metrics:**
 - Sorts/Trx, Avg. Sort Time
 - Sort Overflows
 - With **SYSPROC.MON_GET_WORKLOAD()**

```
SELECT VARCHAR(workload_name,30),
CASE WHEN (total_app_commits + total_app_rollbacks) > 0
THEN DEC((total_section_sort_time) / ((total_app_commits) + (total_app_rollbacks)),8,5)
ELSE NULL
END AS SORTTIME_PER_TRX,
CASE WHEN total_sorts > 0
THEN ((total_section_sort_time) *.001)/(total_sorts)
ELSE NULL
END as AVG_SORTTIME,
total_sorts as TOTAL_SORTS,
total_section_sort_time as TOTAL_SORTTIME,
sort_overflows as TOTALSORTOVERFL,
(total_app_commits + total_app_rollbacks) as TotalTransactions
FROM TABLE(SYSPROC.MON_GET_WORKLOAD("",-2)) AS T;
```

Essential Monitoring Targets - Locking

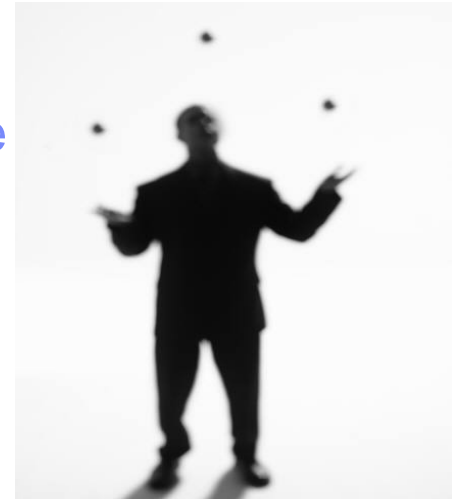
1. The Deadlocks and Lock Timeouts per trx
2. Average Lock Wait Time per Lock Wait
 - With **SYSPROC.MON_GET_WORKLOAD()**

```
SELECT VARCHAR(workload_name,30),  
  CASE WHEN (total_app_commits + total_app_rollbacks) > 0  
  THEN DEC((((deadlocks + lock_timeouts) / (total_app_commits + total_app_rollbacks)),5,0)  
  ELSE NULL  
  END AS DEADLOCK_PLUS_LOCKTIMEOUT_PER_TRX,  
  CASE WHEN (lock_waits) > 0  
  THEN DEC(((lock_wait_time *.001)/lock_waits) ,8,2)  
  ELSE NULL  
  END AS LOCKWAIT_TIME_SEC_PER_LOCKWAIT,  
  lock_waits ,  
  lock_wait_time,  
  deadlocks,  
  lock_timeouts,  
  (total_app_commits + total_app_rollbacks) as TOTALTRX  
FROM TABLE(SYSPROC.MON_GET_WORKLOAD('',-2)) AS T;
```



Essential Monitoring Targets – #Page Clean Triggers

- **Three conditions pages are cleaned from Bufferpools**
 - chngpgs_thresh, % dirty pages in BP is exceeded
 - There is no room in BP dirty pages written to disk
 - softmax, oldest dirty page exceeds desired log size recovery
- **DB2_USE_ALTERNATE_PAGE_CLEANING ?**
 - Page cleaners always active
 - Nullifies these metrics



```
SELECT SUBSTR(bp_name,1,20),
       pool_drty_pg_steal_clns as PAGESTEALTRIGGERS,
       pool_drty_pg_thrsh_clns as PAGETHRESHOLDTRIGGERS,
       pool_lsn_gap_clns as LOGBUFFERPOOLGAP,
CASE WHEN (pool_drty_pg_steal_clns + pool_drty_pg_thrsh_clns + pool_lsn_gap_clns) > 0
THEN DEC((FLOAT(pool_drty_pg_steal_clns) / (FLOAT(pool_drty_pg_steal_clns) + FLOAT(pool_drty_pg_thrsh_clns) +
FLOAT(pool_lsn_gap_clns))),10,5)
ELSE NULL
END as DIRTYSTEALRATIO
FROM TABLE(MON_GET_BUFFERPOOL("-",2)) AS METRICS;
```

Essential Monitoring Targets – Average Log Disk Wait time in milliseconds

- **Time agent waits for log records to be flushed to disk per number of log write waits**
 - With **MON_GET_WORKLOAD()**

```
SELECT varchar(workload_name,30) as WORKLOAD_NAME,  
CASE WHEN log_disk_wait_time > 0  
THEN DEC(FLOAT(log_disk_waits_total)/FLOAT(log_disk_wait_time),10,7)  
ELSE NULL END as AVG_LOGDISK_WAIT_TIME_MS,  
log_disk_wait_time as LOG_DISK_WAIT_TIME,  
log_disk_waits_total as LOG_WAITS_TOTAL  
FROM TABLE(MON_GET_WORKLOAD(",-2)) AS T ;
```


DB2 Performance Monitoring Essentials

Agenda

- Introduction
- Monitoring Interfaces available with DB2
- Essential Monitoring Targets
- **Monitoring tools**
 - System Monitoring Tools
 - DB2 Monitoring Tools

System Monitoring tools \leftrightarrow System Bottlenecks

• Common System Monitoring Tools

- iostat \leftrightarrow Disk Bottlenecks
- vmstat/sar \leftrightarrow CPU, Memory, Disk Bottlenecks
- Perfmon (windows) \leftrightarrow CPU, Memory, Disk Bottlenecks
- nmon (aix et.al) \leftrightarrow CPU, Memory, Disk Bottlenecks
- netstat \leftrightarrow network Bottlenecks
- top \leftrightarrow CPU, Memory Bottlenecks

iostat

- Are you I/O bound?
- Run at regular intervals 3-5 seconds
- Which devices are nearing 100% busy (% tm_act)?
- Where is DB2 making use of this device?

Disks:	% tm_act	Kbps	tps	Kb_read	Kb_wrtn
hdisk6	28.0	73.8	4.2	8999945	3314136
hdisk7	98.7	44486.4	6.1	13790876	629660
hdisk8	0.0	0.6	0.0	90469	2176
hdisk9	0.8	74.4	3.1	9875281	2532636
hdisk10	0.5	54.6	1.7	9108228	3616
hdisk11	0.1	13.7	0.4	2279793	244

Example: List containers on all database members that have the **highest read time**.

```
SELECT varchar(container_name,70) as container_name,
varchar(tbsp_name,20) as tbsp_name, pool_read_time
FROM TABLE(MON_GET_CONTAINER(',-2)) AS t
ORDER BY pool_read_time DESC
```

perfmon (windows)

- **Integrated DB2 Metrics**

- Application level
- DB Level
- DBM Level

The screenshot shows the Windows Performance Monitor interface. On the left, the 'Add Counters' dialog box is open, showing the configuration for monitoring DB2 metrics on the computer \\JPHEHIR. The 'Performance object' is set to 'DB2 Databases' and the instance 'DB2PT97' is selected. The 'Select counters from list' section includes 'Lock Waits', 'Locks Held', 'Number of Lo...', and 'Static SQL St...'. The 'Add' button is highlighted.

The main Performance Monitor window displays a graph with a Y-axis ranging from 0 to 6000. A red vertical line indicates the current time. Below the graph, summary statistics are shown: Last: 2691, Average: 2682, Minimum: 2672, Maximum: 2691, Duration: 1:40.

Color	Scale	Counter	Instance	Parent	Object	Computer
Yellow	1.000	Pages/sec	---	---	Memory	\\JPHEHIR
Blue	100....	Avg. Disk Qu... _Total	---	---	Physic...	\\JPHEHIR
Green	1.000	% Processor ... _Total	---	---	Proces...	\\JPHEHIR
Red	1.000	Lock Waits	DB2PT97...	---	DB2 D...	\\JPHEHIR
Yellow	1.000	Locks Held	DB2PT97...	---	DB2 D...	\\JPHEHIR
Purple	1.000	Number of Lo...	DB2PT97...	---	DB2 D...	\\JPHEHIR
Cyan	1.000	Static SQL St...	DB2PT97...	---	DB2 D...	\\JPHEHIR

top

- CPU consumption
- Configurable output

```

Terminal
File Edit View Terminal Tabs Help
top - 11:11:16 up 12:08, 4 users, load average: 2.04, 1.31, 0.66
Tasks: 108 total, 2 running, 106 sleeping, 0 stopped, 0 zombie
Cpu(s): 23.3%us, 22.4%sy, 0.0%ni, 0.5%id, 50.4%wa, 1.6%hi, 1.8%si, 0.0%st
Mem: 1036540k total, 1021740k used, 14800k free, 103844k buffers
Swap: 2096440k total, 20k used, 2096420k free, 437776k cached

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
 11152 db2inst1  25   0  399m 160m  99m  S  29.2  15.8   1:28.85 db2sysc
   5739 root      14  -1 35476  11m 6272  S   3.6   1.2   2:08.87 X
 19758 db2inst1  16   0 54256  17m 5500  S   1.4   1.7   0:00.14 db2
   8309 root      15   0  102m  16m  10m  S   0.8   1.6   0:22.74 gnome-terminal
   5787 root      15   0 14648  9456 7748  S   0.6   0.9   0:06.58 metacity
 19803 joe       21   0 44684  16m 5500  S   0.6   1.6   0:00.06 db2fm
   5824 root      15   0  102m  26m  18m  S   0.4   2.7   1:06.01 nautilus
   5767 root      15   0 28992  9516 7740  S   0.3   0.9   0:13.70 gnome-settings-
   5822 root      15   0  5912  2436 1968  S   0.3   0.2   0:41.43 vmware-user
   5881 root      15   0 18484  5852 4132  S   0.3   0.6   0:06.58 gnome-power-man
 19805 joe       25   0 35500  15m 4536  R   0.3   1.5   0:00.03 db2set
   3166 root      18   0 35428  16m 5316  S   0.2   1.6   1:38.63 db2fmc
   5809 root      16   0 93076  13m  11m  S   0.2   1.4   0:02.78 gnome-panel
   2974 root      15   0  2376  896  716  S   0.1   0.1   0:45.63 vmware-guestd
   5832 root      15   0 94220  15m  12m  S   0.1   1.5   0:27.09 main-menu
   5843 root      15   0 46092  5452 4472  S   0.1   0.5   0:09.63 gnome-vfs-daemo
   8382 db2inst1  16   0  4412 1916 1428  S   0.1   0.2   0:01.62 bash
 16952 db2inst1  16   0  2192 1040  784  R   0.1   0.1   0:02.53 top
     1 root      16   0   720  284  248  S   0.0   0.0   0:01.16 init
     2 root      34  19   0    0    0  S   0.0   0.0   0:00.03 ksoftirqd/0
     3 root      10  -5   0    0    0  S   0.0   0.0   0:01.06 events/0

```

More system tools

- **netstat**

- Monitor network activity, open ports

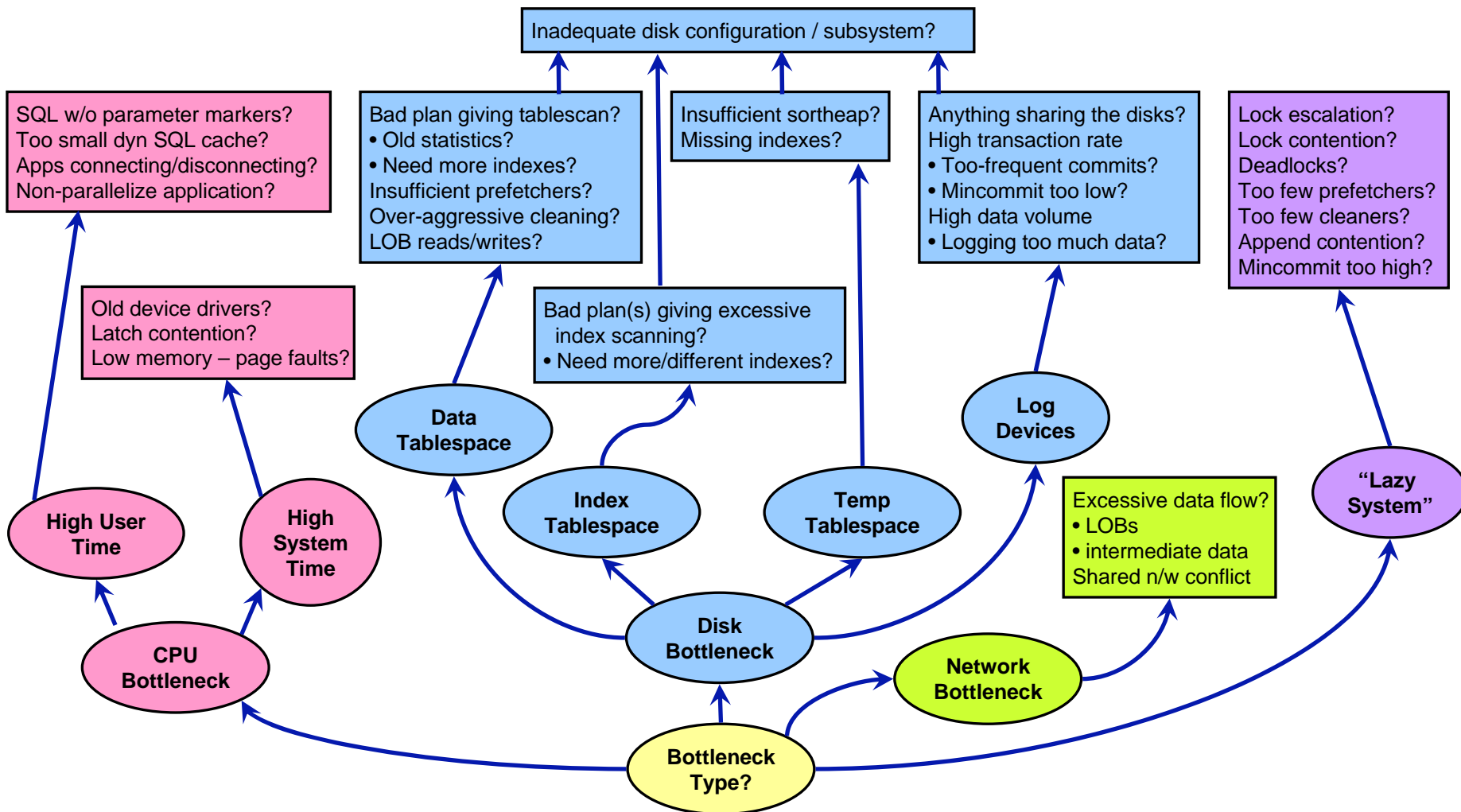
- **nmon**

- Free from sourceforge.net, Open Source
- Comprehensive system monitoring
- Formatted output (CSV) or Screen interface
- Easy to make reports, trend performance data

- **vmstat/sar**

- Command line tools to monitor memory, paging
- CPU usage by user, system, idle, wait times
 - > 25% wait could indicate I/O bottleneck
 - User CPU → compilation and execution of SQL
 - System CPU → calls to OS kernel → Is microcode and drivers current?

How can System Monitoring lead you to the root activity?



DB2 Monitoring Tools

- **db2top**
- **Technology Explorer**



db2top

- **Linux, AIX and Solaris Systems**
- **Single system view**
- **View delta or cumulative snapshot counters**
- **Monitor interactively or collect data for analysis on:**
 - Database
 - Tablespace (t)
 - Dynamic SQL (D)
 - Sessions (I)
 - Bufferpool (b)
 - Lock (U)
 - Table (T)
 - Bottlenecks (B)

Technology Explorer v3.5

More information: <http://sourceforge.net/projects/db2mc/>

	Schema	Table	Data Partition	Type	1 Rows Read	2 Rows Written	Data Object Pages	Index Object Pages
1	SYSIBM	SYSPLAN	null	CATALOG_TABLE	321	0	21	14
2	SYSIBM	SYSHISTOGRAMTEMPLATEBINS	null	CATALOG_TABLE	40	0	1	4
3	SYSIBM	SYSHISTOGRAMTEMPLATEUSE	null	CATALOG_TABLE	28	0	1	4
4	SYSIBM	SYSTABLESPACES	null	CATALOG_TABLE	6	0	1	6
5	SYSIBM	SYSSERVICECLASSES	null	CATALOG_TABLE	6	0	1	4
6	SYSIBM	SYSWORKLOADS	null	CATALOG_TABLE	4	0	1	6
7	SYSIBM	SYSDBAUTH	null	CATALOG_TABLE	2	0	1	6
8	SYSIBM	SYSROUTINES	null	CATALOG_TABLE	2	0	58	87
9	SYSIBM	SYSNODEGROUPS	null	CATALOG_TABLE	2	0	1	4
10	SYSIBM	SYSROLEAUTH	null	CATALOG_TABLE	2	0	1	6
11	SYSIBM	SYSWORKLOADAUTH	null	CATALOG_TABLE	2	0	1	6
12	SYSIBM	SYSEVENTMONITORS	null	CATALOG_TABLE	1	0	1	4
13	SYSIBM	SYSVERSIONS	null	CATALOG_TABLE	1	0	1	3
14	SYSIBM	SYSXMLSTRINGS	null	CATALOG_TABLE	1	0	1	6
15	SYSIBM	SYSXMLPATHS	null	CATALOG_TABLE	1	0	1	6
16	SYSIBM	SYSSURROGATEAUTHIDS	null	CATALOG_TABLE	1	0	1	5
17	SYSIBM	SYSCONTEXTS	null	CATALOG_TABLE	1	0	1	8
18	SYSIBM	SYSCONTEXTATTRIBUTES	null	CATALOG_TABLE	1	0	1	3
19	SYSIBM	SYSHISTOGRAMTEMPLATES	null	CATALOG_TABLE	1	0	1	4

New content for Technology Explorer 3.5

- **New 9.7 tutorials and feature demonstrations**
 - Dynamic Schema Change
 - Created Global Temporary Table
 - Inline Lobs
 - 9.7 Automatic Compression
 - PL SQL Support
 - Truncate Table Command
- **Tools->Monitor (DB2 9.7) - new DB2 9.7 monitoring**
- **Tools->Storage Management (DB2 9.5+)**
- **Health - Monitoring views have been added**
- **View->Catalog - new views have been added**
- **View->Dictionary - views have been added**
- **View->Configuration->Compatibility - will display the compatibility mode of your DB2 9.7 database**

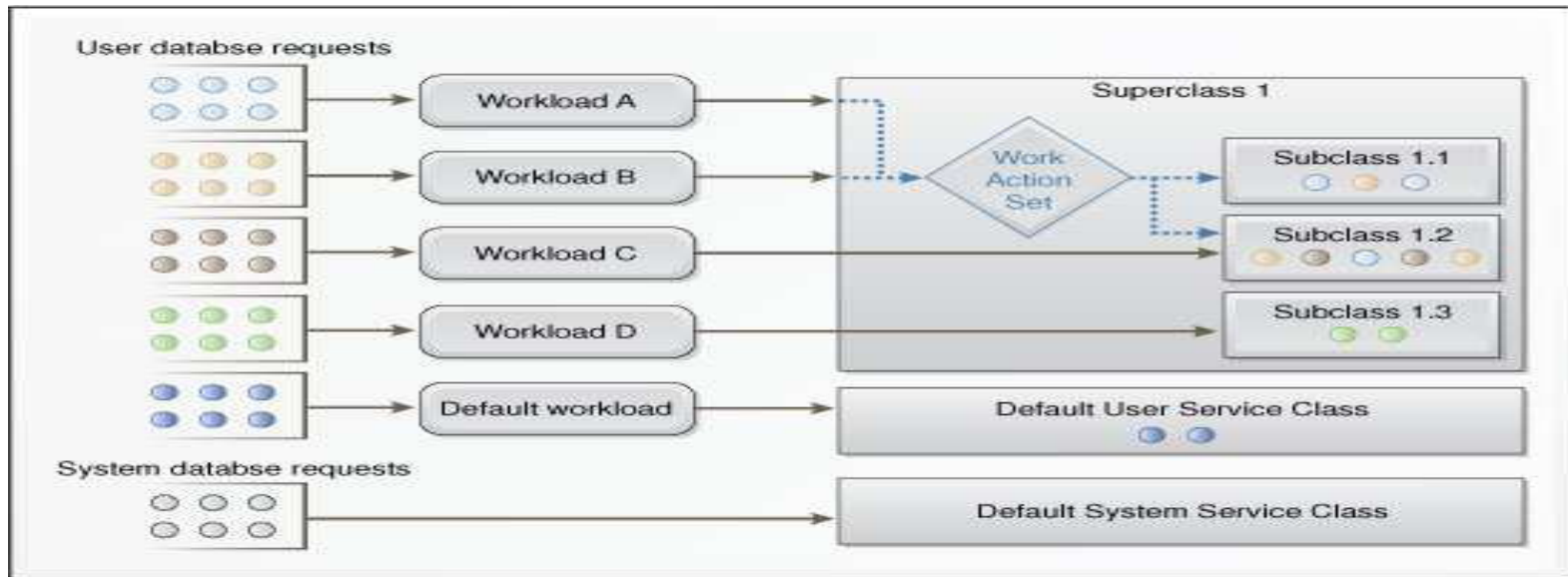
ALL DONE

Thank You



Without Workload Management Feature you can still...

- You can use and alter the default service classes and workloads.
- You can create, alter and drop histogram templates.
- **You can use the DB2 workload manager table functions and stored procedures.**
- **You can create, activate, stop and drop WLM event monitors.**
- You can grant, alter and drop workload privileges.





SQL Query Tuning and DB2 Performance

September 2009

Problematic SQL

- Problematic SQL statements slow down application
- DB2 has many features that can help
- Problematic SQL statements can be detected with:
 - Snapshot monitors
 - Event monitors
 - SQL monitoring interfaces
 - Administrative Views
 - Table functions

Contd.

Problematic SQL

Contd.

- Problematic SQL statements can be analyzed with:
 - Visual Explain
 - Text Explain Facility
- Problematic SQL statements can be improved with:
 - Design Advisor
 - Indexes
 - Materialized Query Tables
 - Multi-dimensional Clustering
 - Partitioning
 - Better coding
 - Statement Concentrator (New 9.7)

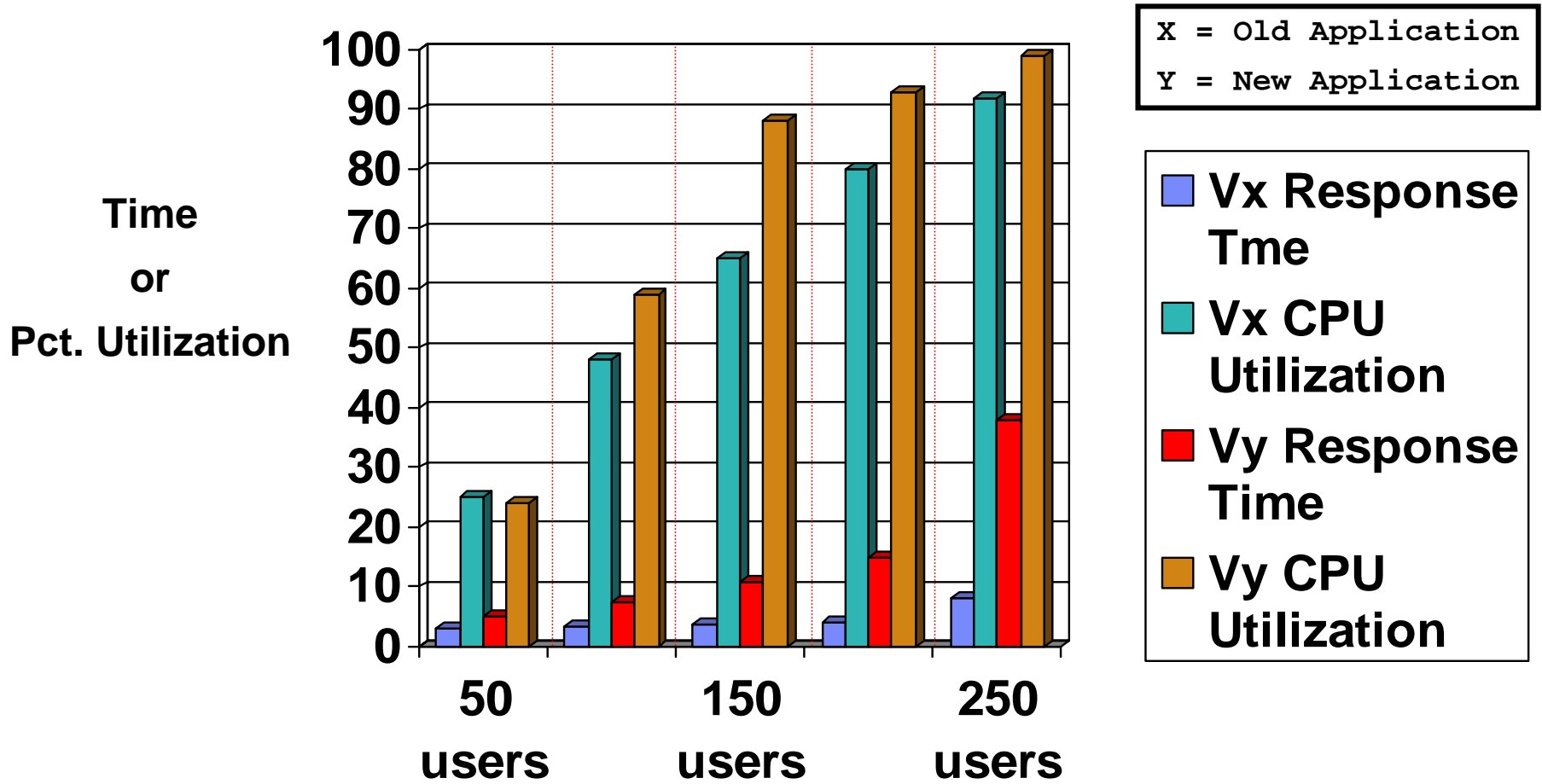
SCENARIO TO FOLLOW: NEW RELEASE OF APP.

- **Foobar Enterprise SW – NEW MAJOR RELEASE + DB2 9.7**
- **EXPECTATION: Increase Value: Reduce Response, Reduce CPU, Increase user capacity**
- **USE DB2ADVIS with -wlm option to capture and advise**
 - Follow recommendations for Indexes, MDC and MQT's
 - Index
 - MDC
 - MQT
- **Capture Additional High Cost SQL for further analysis and tuning**
- **Further analyze and tune with Visual Explain**
- **STAT_CONC – Third party component not using parameter markers**

FooBar Enterprise Software Situation

- **FooBar's benchmark for new release slower than previous release**
- **DB2 is configured well**
 - Large part of DB2 tuning done in previous release
 - Autoconfigure has been run on new schema
- **New SQL may need tuning**
 - Complementary database objects (indexes, MDC, MQT, partitioning)
 - Coding
- **New version of DB2 (9.7)**
 - Explore new performance enhancing features

Original Comparative Results Version X vs. Version Y (New version is slower, consumes more resources)



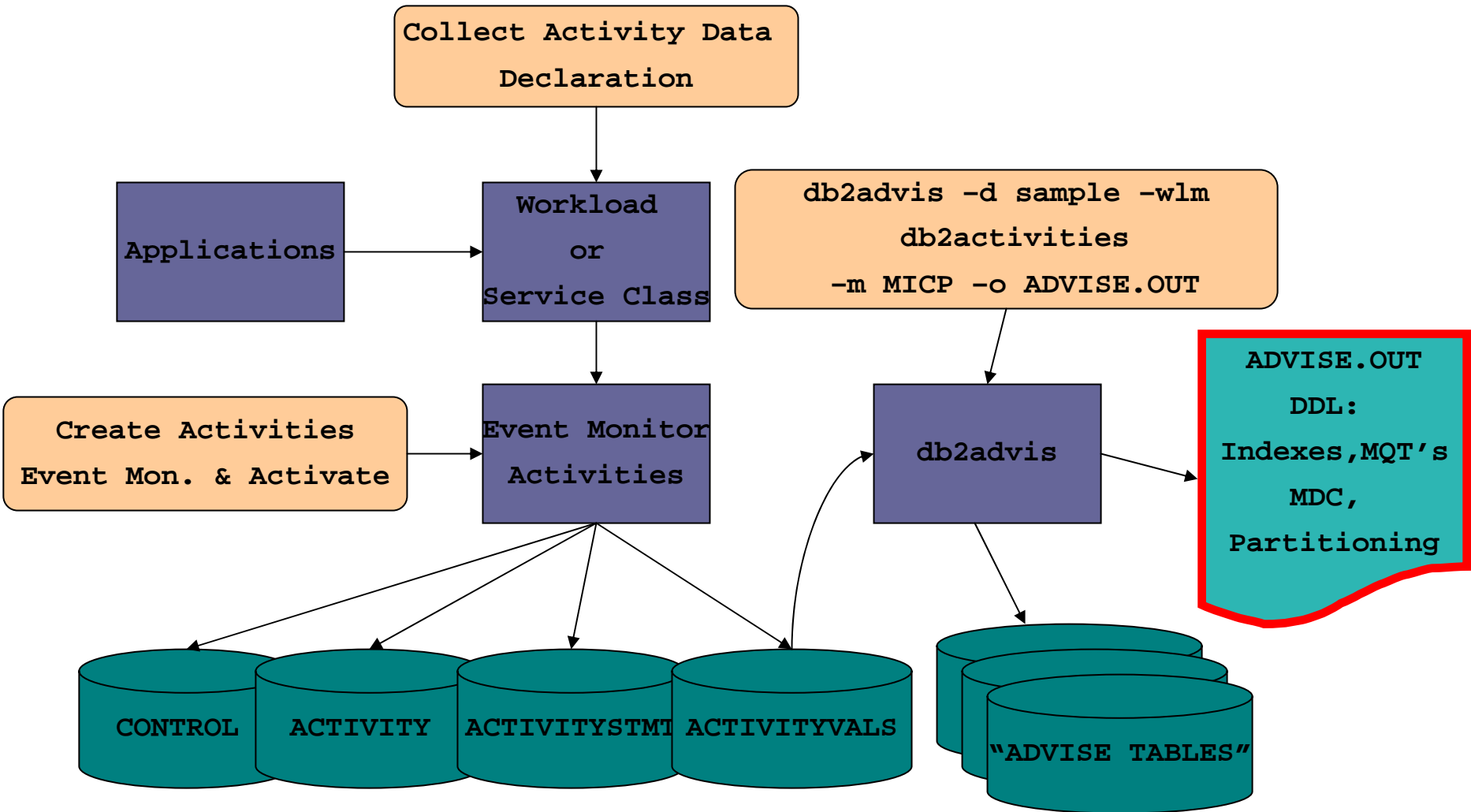
First: Foobar Enterprise Software uses Event Monitor for Activities (9.7) & DB2ADVIS to initially tune workload

- Integrates seamlessly with design advisor (db2advis)
 - Index and other database object recommendations
 - Indexes
 - MQTs
 - MDCs
 - Ranged Table
- Captures both Dynamic and Static SQL
- WLM feature can focus on specific workloads and service classes (w/ Performance Optimization Feature Pack)
- Optionally – Snapshot monitoring can be used in place of Event Monitor for Activities

Steps to take for EVM for Activities & db2advis analysis

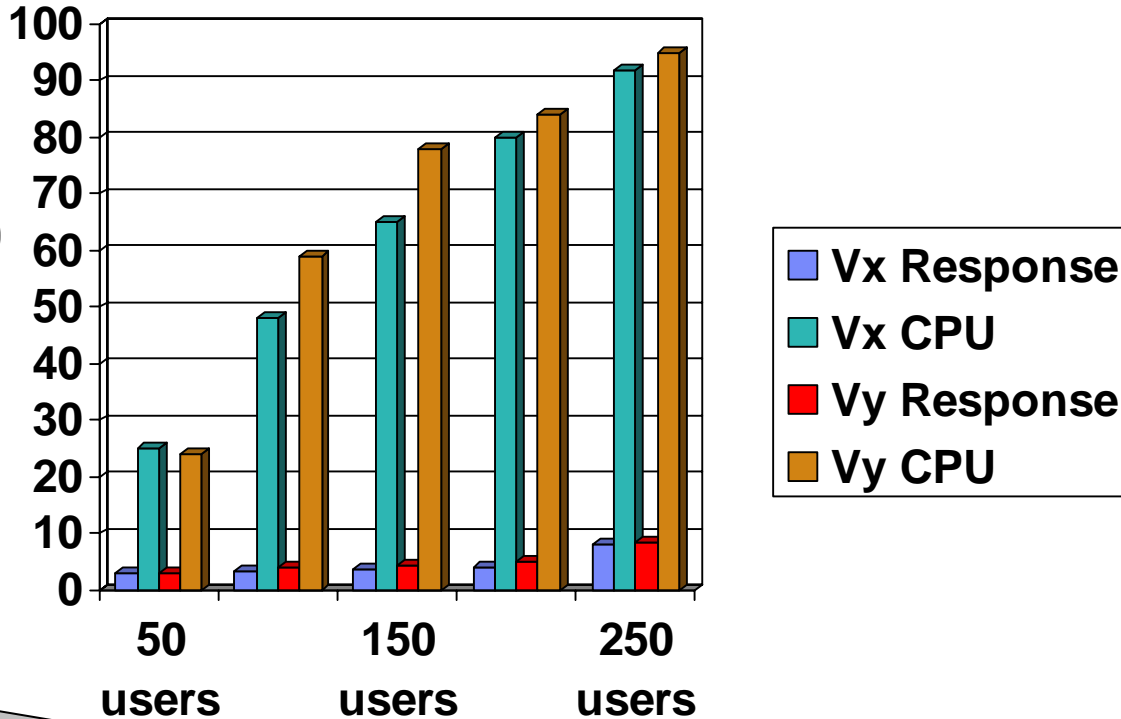
1. Create event monitor db2activities for activities - write to table option
2. Set event monitor db2activities state 1
3. Alter workload sysdefaultuserworkload collect activity data on coordinator with details and values
4. Work is run in sysdefaultuserworkload
5. db2advis -d sample -wlm db2activities -m MICP -o advise.out

db2advis and EVM for Activities flow



After db2advis recommendations and implementation Comparative Results Version X vs. Version Y

Better, but I don't get it? Our new App should be much faster.



FOOBAR's

Application Architect

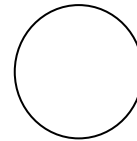
BRING THE COSTLY SQL TO ME NOW!....For further analysis.

Must Detect Costly SQL

Snapshot
Monitoring?

Event
Monitoring?

SQL
Interfaces?



FOOBAR'S

Senior DBA

SNAPSHOT MONITORING

- To request snapshot information about the SQL running on SAMPLE database, issue:
 - **GET SNAPSHOT FOR DYNAMIC SQL ON sample**
- Returns a point-in-time picture of the contents of the SQL statement cache for the database
- **Only for DYNAMIC SQLs**
- Formatted text output
- Returns snapshot information (must have switches on)

Output for SNAPSHOT FOR DYNAMIC SQL

```

db2inst1@db2server...sentinal/db2scripts
File Edit View Terminal Tabs Help

Number of executions           = 17
Number of compilations        = 1
Worst preparation time (ms)   = 6
Best preparation time (ms)    = 6
Internal rows deleted         = 0
Internal rows inserted        = 0
Rows read                     = 72
Internal rows updated         = 0
Rows written                  = 0
Statement sorts               = 17
Statement sort overflows     = 0
Total sort time               = 0
Buffer pool data logical reads = 29
Buffer pool data physical reads = 0
Buffer pool temporary data logical reads = 0
Buffer pool temporary data physical reads = 0
Buffer pool index logical reads = 16
Buffer pool index physical reads = 0
Buffer pool temporary index logical reads = 0
Buffer pool temporary index physical reads = 0
Buffer pool xda logical reads = 0
Buffer pool xda physical reads = 0
Buffer pool temporary xda logical reads = 0
Buffer pool temporary xda physical reads = 0
Total execution time (sec.microsec)= 0.000000
Total user cpu time (sec.microsec) = 0.013410
Total system cpu time (sec.microsec)= 0.000000
Total statistic fabrication time (milliseconds) = 1
Total synchronous runstats time (milliseconds) = 0
Statement text                 = SELECT * FROM WE_VGUSN5J13.WE_OGO_42785 ORDER BY RAND() FETCH FIRST 1 ROWS ONLY

Number of executions           = 1
Number of compilations        = 1
Worst preparation time (ms)   = 5
Best preparation time (ms)    = 5
Internal rows deleted         = 0
Internal rows inserted        = 0
Rows read                     = 641
Internal rows updated         = 0
Rows written                  = 0

```

STATEMENT CACHE AND SNAPSHOT DATA

Event Monitors

- Event monitors are used to collect information about the database and any connected applications when specified events occur
- Filter events on APPL_ID, AUTH_ID and APPL_NAME
- Event type to capture SQL statements
 - STATEMENTS - Statement start/stop time, CPU used, dynamic SQL
- **High overhead**

Creating an Event Monitor for Statements

- A table event monitor streams event records to SQL tables, this makes capture, parsing, and management of event monitoring data easy
- Need SYSADM or DBADM to create a table event monitor
- Syntax:

```
CREATE EVENT MONITOR stmtmon
FOR STATEMENTS
WHERE APPL_NAME = 'NEWAPP' AND
      AUTH_ID = 'BBDS'
WRITE TO TABLE IN event_tblspace
      CONNHEADER(TABLE STMT_EVT_CH, IN TBS_EVMON),
      STMT(TABLE STMT_EVT_STMT, IN TBS_EVMON, TRUNC),
      CONTROL(TABLE STMT_EVT_CTRL, IN TBS_EVMON)
BUFFERSIZE 2000
NONBLOCKED
OR
WRITE TO FILE '/tmp/dlevents'
OR
WRITE TO PIPE '/home/riihi/dlevents'
```

 = REDUCE IMPACT

Sample Output: Statement Event Monitor Query

Control Center

Control Center Selected Edit View Tools Help

Object View Command Editor 1 X

Commands Query Results Access Plan

Edits to these results are performed as positioned UPDATES and DELETES. Use the Tools Settings notebook to change the form of editing.

ROWS_READ	SQLCODE	TOTAL_SORT_TIME	USER_CPU_TIME	5	
0	0	0	0	34	
0	0	0	0	138	SELECT * FROM W...
0	0	0	0	29	SELECT * FROM W...
45	100	0	0	260	SELECT * FROM W...
1	0	0	0	553	DELETE FROM WE...
0	0	0	0	82	
2	0	0	0	434	UPDATE WE_VGU...
0	0	0	0	156	INSERT INTO WE_...
0	0	0	0	44	
0	0	0	0	70	
0	0	0	0	35	
0	0	0	0	122	SELECT * FROM W...
0	0	0	0	32	SELECT * FROM W...
15	100	0	0	234	SELECT * FROM W...
1	0	0	0	482	DELETE FROM WE...
0	0	0	0	56	
0	0	0	0	302	INSERT INTO WE_...

Commit Roll Back Fetch More Rows

Automatically commit updates

100 row(s) in memory

Add Row Delete Row

SQL Monitoring Interfaces

- Administrative Views
 - Easy-to-use application programming interface
 - Execute administrative functions through SQL
- Table functions MON_GET_...
 - Introduced in DB2 9.7
 - Enhanced reporting and monitoring of the database system, data objects, and the package cache
 - **Lightweight** - has a lower impact on the system than existing system monitor and snapshot interfaces

Examples of SQL interfaces – Finding Costly SQL

- Administrative Views
 - LONG_RUNNING_SQL (Time, Statement, Status)
 - QUERY_PREP_COST (High Prep Times, % of Exec)
 - TOP_DYNAMIC_SQL (Exec Time, Sorts)
- Table Functions
 - MON_GET_ACTIVITY_DETAILS (Executing v. Waiting)
 - MON_GET_PKG_CACHE_STMT (Filtering Options)

SQL – High CPU TIME

Example: List top 10 SQL statements by **cpu_time**

```
SELECT MEMBER,  
SECTION_TYPE ,  
varchar(stmt_text,200) as statement,  
num_exec_with_metrics as numExec,  
TOTAL_CPU_TIME/NUM_EXEC_WITH_METRICS as AVG_CPU_TIME,  
TOTAL_CPU_TIME  
FROM TABLE(MON_GET_PKG_CACHE_STMT ('D', NULL, NULL, -2)) as T  
WHERE T.NUM_EXEC_WITH_METRICS <> 0  
ORDER BY AVG_CPU_TIME desc  
fetch first 10 rows only;
```


Results from Top Ten CPU consumers.

Database Administration - .sqlxeditor_project/Script1.sql - IBM Optim Database Administrator

File Edit Navigate Search Project Data Run Script Window Help

SQL Results

Type query expression here

Status	Operation	MEMBER	SECTION_TYPE	STATEMENT	NUMEXEC	AVG_CPU_TIME
✓ Succeed	Run SQL	1 0	D	SELECT MEMBER, SECTION_TYPE, varchar(stmt_text,	3	84556
✓ Succeed	Run SQL	2 0	D	UPDATE WE_Y9NKFDJ12.WE_0G0_47 SET value=200 V 1497		25354
✗ Failed	Run SQL	3 0	D	UPDATE WE_Y9NKFDJ12.WE_0G0_83460 SET value=2(1477		25289
✗ Failed	Run SQL	4 0	D	UPDATE WE_Y9NKFDJ12.WE_0G0_25099 SET value=2(1487		25224
✓ Succeed	Run SQL	5 0	D	UP		
		6 0	D	UP		
		7 0	D	UP		
		8 0	D	UP		
		9 0	D	UP		
		10 0	D	UP		

Long data

Column name: STATEMENT
Column data type: VARCHAR

```
UPDATE WE_Y9NKFDJ12.WE_0G0_47 SET value=200 WHERE id=(SELECT id FROM WE_Y9NKFDJ12.WE_0G0_47 ORDER BY RAND() FETCH FIRST 1 ROWS ONLY)
```

OK Cancel

Total 10 records shown

Displayed 6 of 6 results: 4 succeeded, 2 failed, 0 terminated, 0 warning, 0 critical error

Analyzing SQL

- EXPLAIN TOOLS
- VISUAL EXPLAIN

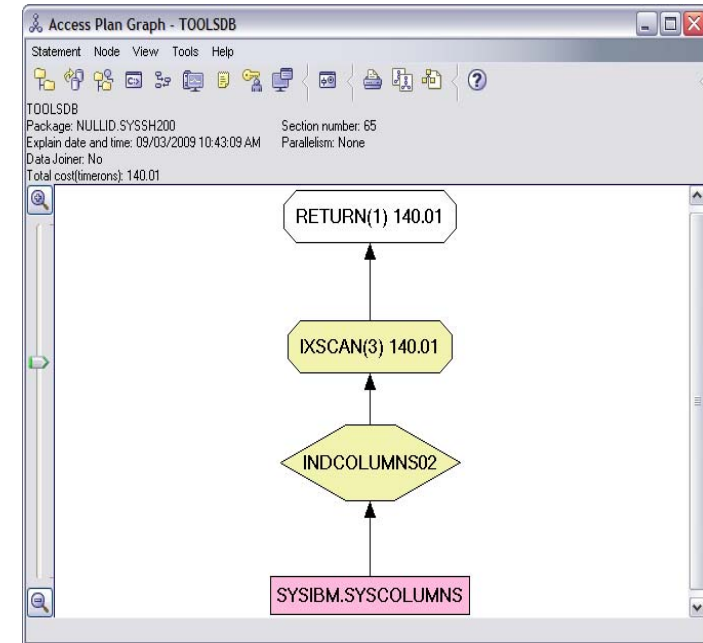
Now we will see,
precisely how, and
how well the
costly SQL
executes



SQL Explain Tools

- Graphical
 - Easy to quickly spot the problem
 - Provides drill down functionality
 - Multiple images can be stored for comparison

- Text Based
 - Can be used with any interface
 - All the information is contained on a single screen
 - Available on all platforms
 - Format output with db2exfmt



```

Optimized Statement:
SELECT Q1.PROD_CODE AS "PROD_CODE", Q1.PROD_NAME AS "PROD_NAME",
       Q1.PROD_BRAND AS "PROD_BRAND", Q1.PROD_CAT AS "PROD_CAT"
FROM DB2INST1.PRODUCT_DIM AS Q1

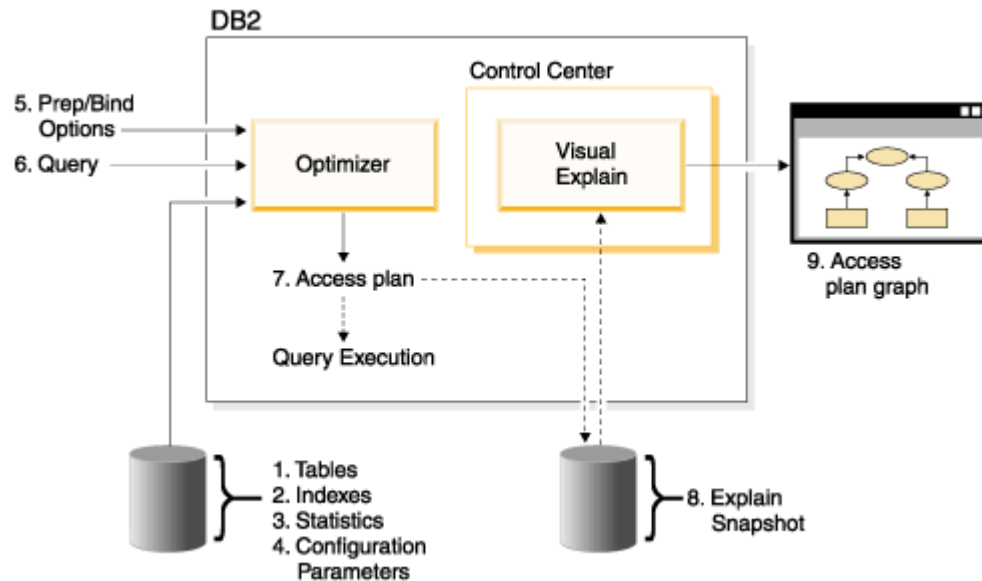
Access Plan:
-----
Total Cost:          7.6874
Query Degree:        1

      Rows
      RETURN
      ( 1)
      Cost
      1/0
      |
      55
      FETCH
      ( 2)
      7.6874
      1
      -----
      55          55
      IXSCAN      TABLE: DB2INST1
      ( 3)        PRODUCT_DIM
      0.0863678   Q1
      0
      |
      55
      INDEX: DB2INST1
      IDX909162256070000
      -----(182)
  
```

Why use Explain?

- To seek performance tuning opportunities
 - How are tables being accessed?
 - How useful are additional indexes?
 - Does rewriting the query help?
- Comparisons: To understand changes in query performance due to:
 - Changes in the data model
 - Changes in the data
 - Changes in configuration parameters
- View statistics used at time of optimization and current performance

VISUAL EXPLAIN



How to use Visual Explain

- Invoke from
 - Data Studio toolset
 - Control Center
- Enter SQL to be analyzed
 - Trap the poor running SQL statement either from your program, performance monitors or create a brand new statement
 - The text can then just be typed or copied into the input box
- Output
 - Explain Information stored in Explain Tables
 - Detailed information
 - Manipulate Explain information using SQL
 - Access Plan Graph
- For dynamic and static SQL statements

Visual Explain Interface

- Every object in the visual explain interface can be drilled down for additional information
- Cost
 - The estimated total resource usage necessary to execute the access plan for a statement. The unit of cost is the timeron
- Timeron
 - Timeron is a combination of CPU cost (in number of instructions) and I/O (in numbers of seeks and page transfers)
 - In general if you have a larger number of timerons your query will run slower
- All of the run times of the individual components are cumulative and are measured in timerons

Visual Explain Interface – Access Plan Diagram

Window

CLICK TO FIND PREFERENCES FOR VISUAL EXPLAIN

Access Plan Diagram

CONTROLS: OPEN, SAVE ACCESS PLAN GRAPH
ZOOM
SCALE GRAPH TO FIT
INVERT GRAPH
PRINT

Canvas

Overview of Diagram

Basic Information
Information that identifies the diagram.

Database Platform: LUW
Database Version: DB2 v09.07.1
Explain Timestamp: 2010-02-11T06:26:41
[View the SQL Statement](#)
[Save Diagram...](#)

Diagram Overview
Display the selected diagram overview.

Query

ISOLATE SPECIFIC SECTIONS OF ACCESS PLAN GRAPH

(1)RETURN
6981.31

(2)HSJOIN
6981.31

(3)TBSCAN
6954.37

(00)SALES_FACT
DB2INST1

(5) |>
0.0!

Description of Selected Node

Search for Node

Query

Visual Explain Interface – View SQL Statement

The screenshot displays the Visual Explain Interface. On the left, the 'Basic Information' panel includes fields for Database Platform (LUW), Database Version (DB2 v09.07.1), and Explain Timestamp (2010-02-11T05:29:51). A red box highlights the 'View the SQL Statement' link. A red arrow points from this link to the 'View SQL Statement Link' label. Below the 'Diagram Overview' section, a red box highlights the 'SQL Statement 2' tab in the 'SQL Statement' window. A red arrow points from this tab to the 'Optimized SQL Tab' label. The 'SQL Statement' window displays the following SQL query:

```

SELECT
  D.day_quarter_name,
  S.state,
  P.prod_name,
  SUM (F.quantity)
FROM
  sales_fact F,
  product_dim P,
  store_dim S,
  date_dim D
WHERE
  F.day_date = D.day_date and
  F.prod_code = P.prod_code and
  F.store_num = S.store_num
    
```

On the right side of the interface, a vertical flow diagram shows three nodes: (1) RETURN (8421.97), (2) GRPBY (8421.93), and (3) TBS CAN (8421.91). The nodes are connected by arrows pointing upwards, indicating a data flow from the bottom node to the top node.

Drilling into Nodes - Index Scan Object Data

Selected Node Descriptor: IXSCAN

Description of Selected Node

Displays information about the node that is highlighted in the diagram.

ixscan

Attributes

NAME	VALUE
Operator Identifier	5
Operator Type	IXSCAN
Output Cardinality	55
Cumulative Total Cost	0.0996144
Cumulative CPU Cost	134613
Cumulative IO cost	0
Cumulative Re-execution Total Cost	0.0733731
Cumulative Re-execution CPU Cost	99152
Cumulative Re-execution IO cost	0
Cumulative First Row Total Cost	0.0295692
Estimated Bufferpool Buffers	1
Maximum pages for prefetch	ALL
Type of Prefetch	NONE
Row Lock intent	NONE
Scan Direction	FORWARD

Description of the Selected Attribute
Cumulative total number of instructions

DESCRIPTION of ATTRIBUTE

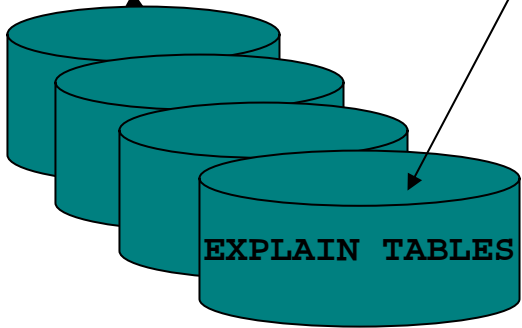
Overview: Costly SQL → Explain

Costly SQL

DB2
Explain

<u>Resource Requirements</u> #CPU Instructions #IOs required #Buffer pages
<u>Runtime State</u> Heap Allocations Database Configurations Statistics
<u>Operations</u> Join Techniques Sorting Predicate Evaluations
<u>Access Methods</u> Table Scans Index Access Index Scans

db2 -tvf EXPLAIN.DDL



At the status meeting....



Stand, take heed, and come hither. Tell me....how goes it?

By using DB2's monitoring and tuning tools, the new version is now seeing a 5% decrease in cpu usage compared to the old version.



FOOBAR'S

V.P. of Engineering



But the new reporting component from CCCZZZ, is still using far too much CPU. They don't use parameter markers, and prep-time is killing us.

Wait! I just had a thought. DB2 V9.7 features the STATEMENT CONCENTRATOR

Statement Concentrator

Introduced in DB2 9.7



Statement Concentrator

- Specifies whether dynamic statements that contain literal values use the statement cache
- Database configuration parameter
stmt_conc **OFF** | LITERALS
- CLI/ODBC configuration keyword
StmtConcentrator = OFF | WITHLITERALS
- Environment or connection attribute
SQL_ATTR_STMT_CONCENTRATOR
SQL_STMT_CONCENTRATOR_OFF
SQL_STMT_CONCENTRATOR_WITH_LITERALS
- The default setting for the configuration keyword or environment attribute is the one that's specified for statement concentration on the server

* Because statement concentration alters the statement text, it has an impact on access plan selection. The statement concentrator should be used when similar statements in the package cache have similar access plans. If different literal values in a statement result in significantly different access plans, the statement concentrator should not be enabled for that statement

Statement Concentrator Example

- SELECT FIRSTNME, LASTNAME FROM EMPLOYEE
WHERE EMPNO='000020'

and

- SELECT FIRSTNME, LASTNAME FROM EMPLOYEE
WHERE EMPNO='000070'

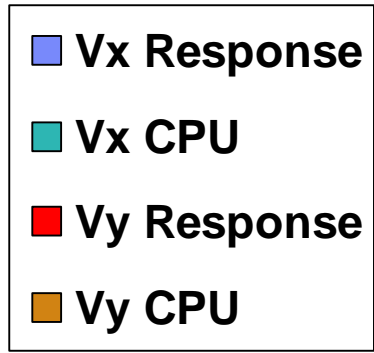
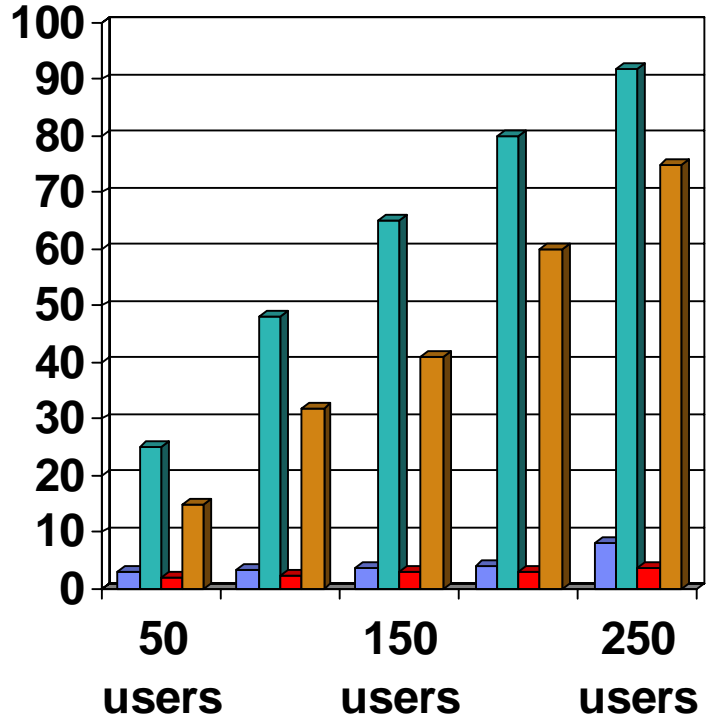
Share the
same
entry in
the
package
cache

- SELECT FIRSTNME, LASTNAME FROM EMPLOYEE
WHERE EMPNO=:L0

Package
cache will
use the
statement

DB2 will provide the value for :L0 (either '000020' or '000070')
based on the literal used in the original statements.

And there was much rejoicing !



Making Performance Improvements

- Database Objects
- Better coding
- Design Advisor
- Other Considerations

Database Objects



Indexes – Benefits and uses

- Apply predicates to provide rapid look-up of the location of data in a database,
- Reduce the number of rows navigated
- To avoid sorts for ORDER BY and GROUP BY clauses
- To induce order for joins
- To provide index-only access, which avoids the cost of accessing data pages
 - CREATE UNIQUE INDEX EMP_IX ON
EMPLOYEE(EMPNO)
INCLUDE(FIRSTNAME, JOB)
- As the only way to enforce uniqueness in a relational database

Indexes – Overhead

- They add extra CPU and I/O cost to UPDATE, INSERT, DELETE, and LOAD operations
- They add to “query prepare time” because they provide more choices for the optimizer
- They can use a significant amount of disk storage

Indexes – Best Practices

Utilize the following index design best practices:

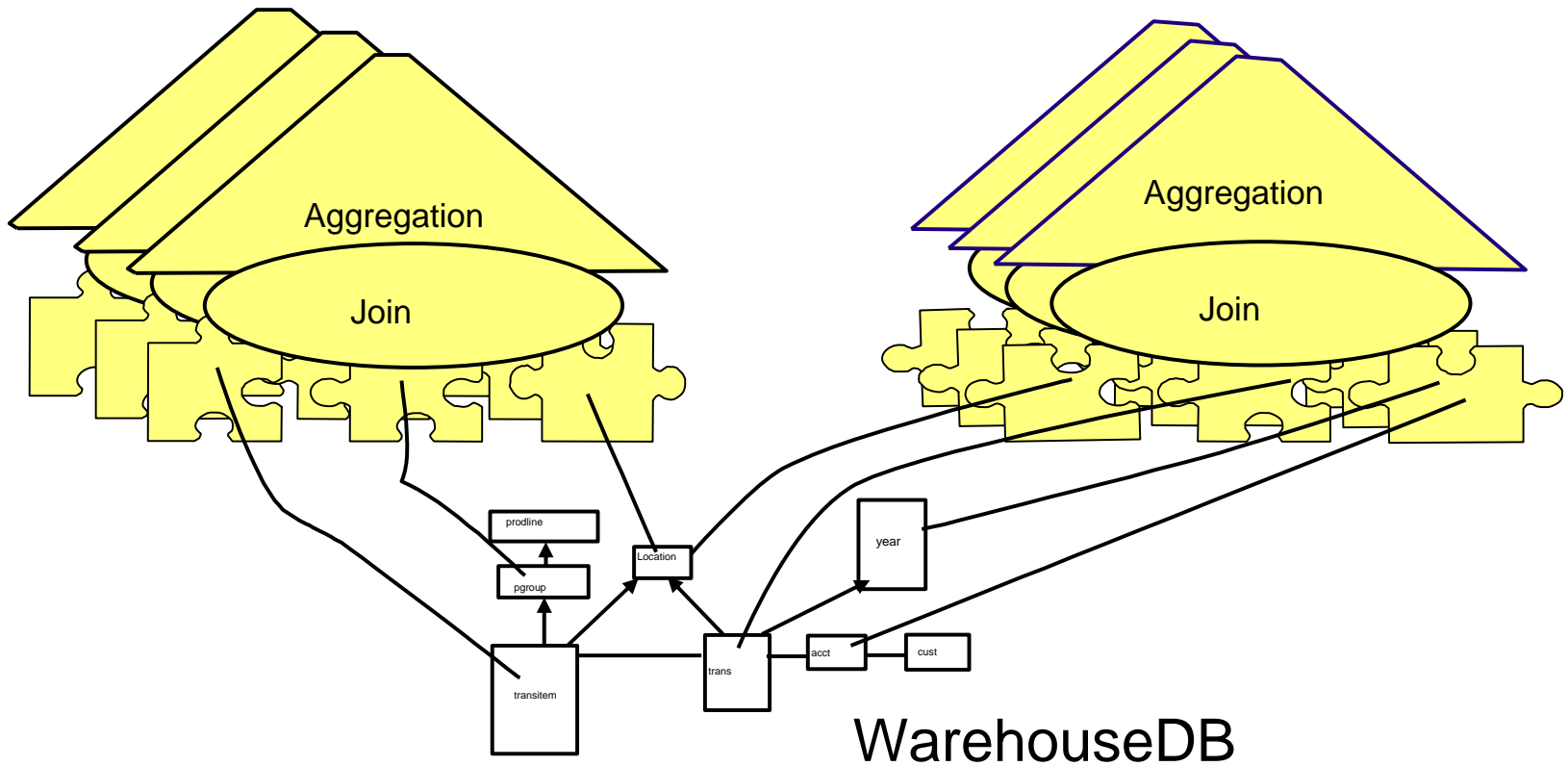
- Index every PK and most FKs in a database. Most joins occur between PKs and FKs, so it is important to build indexes on all PKs and FKs whenever possible.
- Indexes on FKs also improve the performance of RI checking.
- Explicitly provide an index for the PK. The DB2 database manager indexes the PK automatically with a system-generated name if one is not specified. The system-generated name for an automatically-generated index is difficult to administer.
- Columns frequently referenced in WHERE clauses are good candidates for an index. An exception to this rule is when the predicate provides minimal filtering.
 - An example is an inequality such as WHERE cost <> 4. Indexes are seldom useful for inequalities because of the limited filtering provided.
- Specify indexes on columns used for equality and range queries.

Materialized Query Tables (MQTs)

Precomputed results and subsections

Q11, Q12, ...

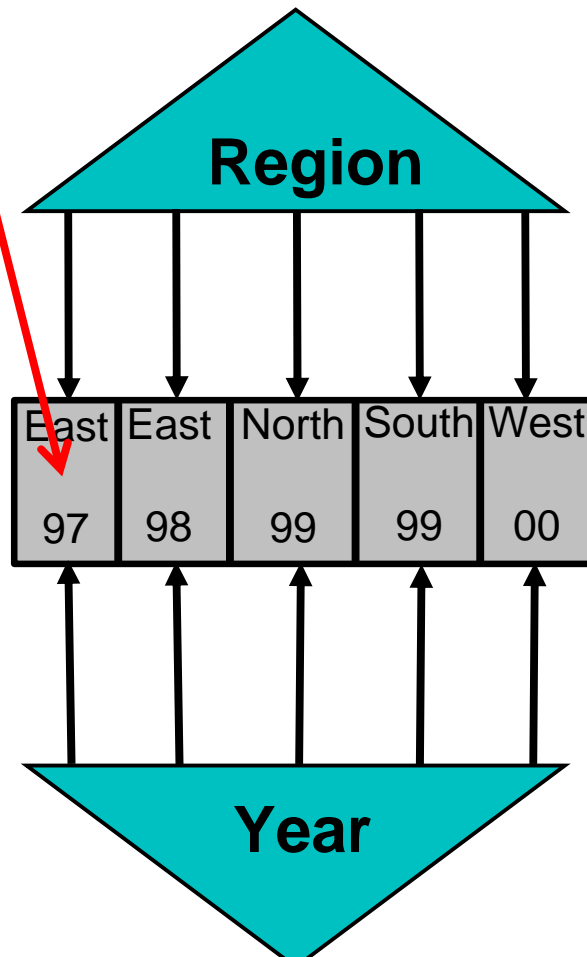
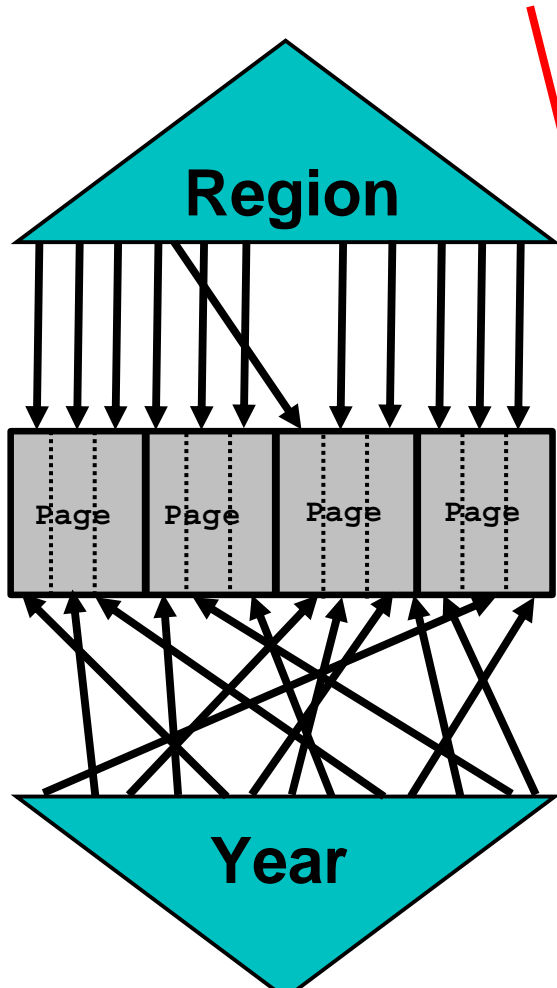
Q21, Q22, ...



What is Multi-Dimensional Clustering?

All records in this block are from the **East** region and from the year **97**

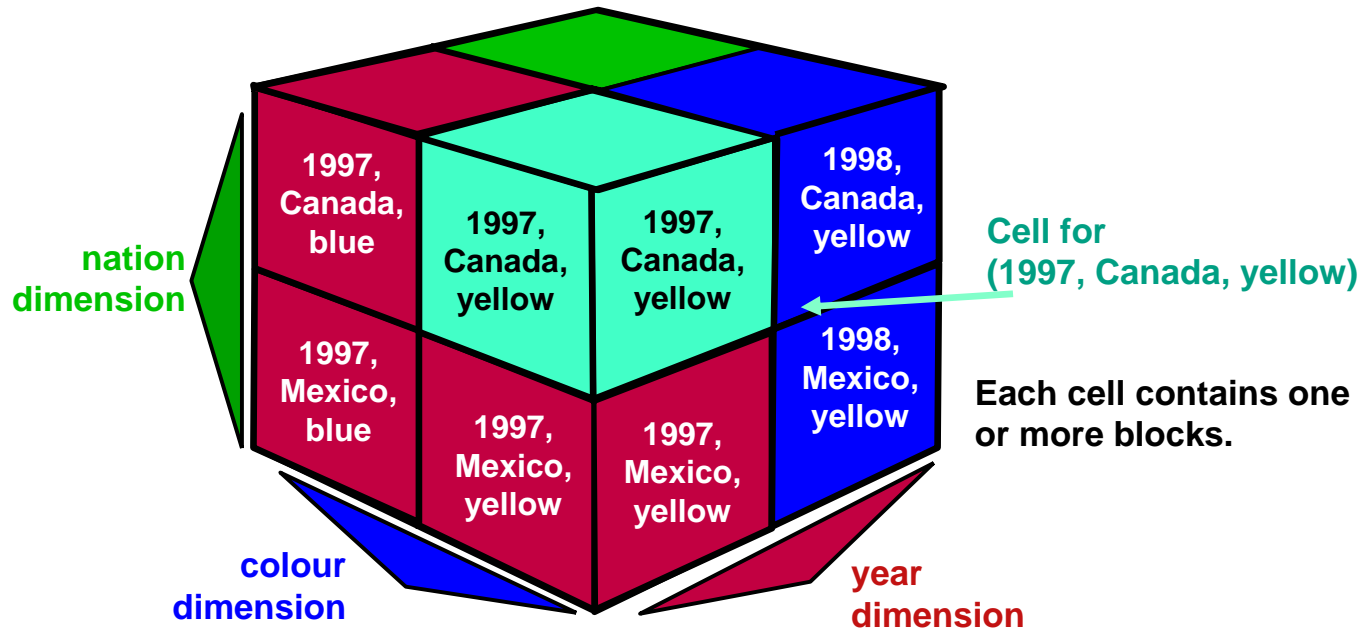
- Traditional indexes refer to **records**.
- Traditional tables are managed by **page**.
- Traditional tables can have only *one* clustering index!
- Access via the clustering index reduces the number of pages that need to be read.



- MDC indexes refer to **blocks**.
- MDC tables are managed by **block**.
- Each row in a block has the same values for the MDC dimensions.
- MDC tables can be clustered on more than one key
- MDC tables can have MDC indexes and ordinary (RID-based) indexes.

Multidimensional Clustering (MDC)

- Faster DELETE along cell or slice boundaries
 - Immediate Index Cleanup Rollout
 - Deferred Index Cleanup Rollout



Better Coding



Writing Efficient SELECT statements

- Specify only columns that you need
- Use predicates that limit to those rows that you need
- Avoid numeric data type conversions
- Avoid data type mismatches
- Preferred data types
 - CHAR instead of VARCHAR for short columns
 - INTEGER instead of FLOAT, DECIMAL or DECFLOAT
 - DECFLOAT instead of DECIMAL
 - DATETIME instead of CHAR
 - NUMERIC instead of CHAR
- Avoid DISTINCT or ORDER BY if not required
- Use IN list if same column used in multiple predicates

Writing Efficient SELECT statements

- Use OPTIMIZE FOR n ROWS clause
- Use FETCH FIRST n ROWS ONLY clause
- Use OPTIMIZE FOR n ROWS clause with the FETCH FIRST n ROWS ONLY clause
- Use FOR READ ONLY or FOR FETCH ONLY clause
- Use FOR UPDATE OF clause
- Use FOR READ ONLY clause along with USE AND KEEP UPDATE LOCKS clause

DB2 Design Advisor



Using Design Advisor

Command Line Usage

```
db2advis -d sample -m MICP -i da.sql
```

-d database name

-m M-MQT I-Indexes C-MDC tables P-Partitioning

Workload type keyword: (choose one)

-s Single SQL statement

-i SQL from input file

-qp SQL from Query Patroller table

-w SQL from ADVISE_WORKLOAD table by workload name

-g Get workload from dynamic SQL snapshot

Other keywords:

-l number of MB available for indexes and MQTs (-1 for unlimited)

-t specifies the maximum time, in minutes, to complete the operation

Create Explain Tables first

Using Design Advisor

Command Line Examples

Example: All object types using an input file of SQL statements

```
db2advis -d sample -m MICP -i da.sql
```

Example: For a single SQL statement

```
db2advis -d TPCD -s "Select * from part where partkey = 1"
```

Example: Use the workload table to determine MQT recommendations

```
db2advis -d TPCD -w wk1 -m M -c sim_space -b mqt_space -q  
newschema
```

Example: Workload from WLM activities event monitor

```
db2advis -d TPCD -wlm db2activities -m MICP -o advise.out
```

Sample Recommendations for Indexes

-- LIST OF RECOMMENDED INDEXES

```
CREATE INDEX "DB2ADMIN"."IDX509062043470000"  
ON "TPCD"."LINEITEM"  
("L_RETURNFLAG" ASC, "L_DISCOUNT" ASC, "L_EXTENDEDPRICE" ASC,  
"L_ORDERKEY" ASC)  
ALLOW REVERSE SCANS ;
```

```
RUNSTATS ON TABLE "TPCD"."LINEITEM"  
FOR INDEX "DB2ADMIN"."IDX509062043470000" ;
```

```
CREATE UNIQUE INDEX "DB2ADMIN"."IDX509062044160000"  
ON "TPCD"."ORDERS"  
("O_ORDERDATE" ASC, "O_ORDERKEY" ASC, "O_CUSTKEY" ASC)  
ALLOW REVERSE SCANS ;
```

```
RUNSTATS ON TABLE "TPCD"."ORDERS"  
FOR INDEX "DB2ADMIN"."IDX509062044160000" ;
```

Sample Recommendations for Materialized Query Table

-- LIST OF RECOMMENDED MQTs

```
CREATE SUMMARY TABLE "DB2INST1"."MQT909112030210000"  
AS (SELECT Q6.C0 AS "C0", Q6.C1 AS "C1", Q6.C2 AS  
"C2", Q6.C3 AS "C3", Q6.C5 AS "C4", Q6.C4 AS "C5"  
FROM TABLE(SELECT Q5.C0 AS "C0", Q5.C1 AS "C1", Q5.C2  
AS "C2", SUM(Q5.C3) AS "C3", COUNT(*) AS "C4", COUNT(Q5.C3)  
AS "C5" FROM TABLE(SELECT Q1.DAY_QUARTER_NAME AS "C0",  
Q2.STATE AS "C1", Q3.PROD_NAME AS "C2", Q4.QUANTITY  
AS "C3" FROM DB2INST1.DATE_DIM AS Q1, DB2INST1.STORE_DIM  
AS Q2, DB2INST1.PRODUCT_DIM AS Q3, DB2INST1.SALES_FACT  
AS Q4 WHERE (Q4.STORE_NUM = Q2.STORE_NUM) AND (Q4.PROD_CODE  
= Q3.PROD_CODE) AND (Q4.DAY_DATE = Q1.DAY_DATE)) AS  
Q5 GROUP BY Q5.C2, Q5.C1, Q5.C0) AS Q6) DATA INITIALLY  
DEFERRED REFRESH IMMEDIATE IN USERSPACE1 ;
```


Other Considerations



Other factors influencing query performance

- Accurate database statistics – RUNSTATS
- Defining [Informational] constraints
- Use the REOPT bind option when host variable's values affect access plan.
- Using parameter markers to reduce statement compilation
- Specifying Row Blocking for better cursor processing
 - Blocking All
 - Blocking No
 - Blocking Unambig
- Data sampling for statistics
 - Row-level Bernoulli sampling
 - System page-level sampling

If all tuning options fail

- Use Optimization profiles
 - Explicit Optimization guideline to DB2 optimizer
 - XML document
 - Define SQL statement
 - Define optimization guideline
 - No application or database configuration changes
- Things to consider about Optimization profiles
 - Requires effort to maintain
 - For existing SQL statements only
 - Optimizer still considers other possible access plans
 - Optimizer ignores invalid or inapplicable guidelines

End of Presentation





DB2 Performance Clinic - Locking

Information Management - Partner Technologies
IBM Software Group

Agenda

- **Locking and Performance**
- **Identifying Locking Scenarios**
- **Using Isolation Levels**
- **Monitoring Locking Issues**
- **Avoiding Locking Scenarios**



Agenda

- **Locking and Performance**
- Identifying Locking Scenarios
- Using Isolation Levels
- Monitoring Locking Issues
- Avoiding Locking Scenarios



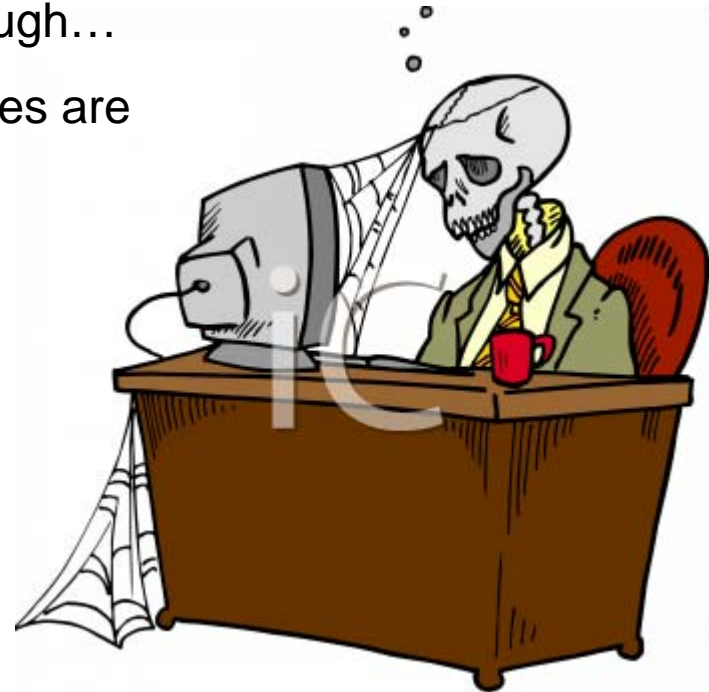
Locking and Performance

To the users, a locking issue can appear to be a performance issue!

While users wait, the perception is the database cannot retrieve data fast enough...

... when in fact, the issue may be queries are encountering:

- **Lock Waits**
- **Lock Timeouts**
- **Lock Escalation**
- **Deadlocks**



You may have a locking problem, if...

- **You are experiencing:**
 - Application failures
 - Frequent retries of application processing
 - General “slow down” in performance of SQL processing
- **Your objectives:**
 - Reduce or eliminate all lock timeouts and deadlocks
 - Both result in application failures
 - Extra processing time
 - Wasted system resources
 - Eliminate all lock escalations

Monitoring Locking

- **Use Snapshots, Event Monitors, Administrative Views and db2pd**
 - Key monitoring elements can be retrieved through multiple ways
- **Review Administrative Notification logs**
- **Need to know:**
 - Type of lock event causing performance slowdown
 - Identify the SQL statement(s) involved
 - Lock requested and encountered by applications
- **Data collection can be difficult**
 - Locking information is extremely transient
 - Most lock information is gone once the lock is released
 - Baseline data is needed for comparison and evaluation

Recommended to monitor for lock waits, lock timeouts and deadlock events at all times

Agenda

- Locking and Performance
- **Identifying Locking Scenarios**
- Using Isolation Levels
- Monitoring Locking Issues
- Avoiding Locking Scenarios



Lock Scenarios

- **Lock waits**
 - Typical, expected event
 - Excessive time spent waiting for a lock is not typical
 - Lock waits slow down performance and completion
 - Excessive lock waits can become lock timeouts
- **Lock Escalations**
 - Small number acceptable – only if no adverse effects
 - Contributes to other locking issues (e.g. lock timeouts)
 - Objective should be to eliminate all lock escalations
- **Lock timeout**
 - Lock timeouts result in the application not completing the transaction and performing a rollback
- **Deadlocks**
 - Resolution deadlock detector arbitrarily selects one deadlocked process as the *victim process* to roll back

Lock Wait Scenario

- Application requests a lock whose mode conflicts with lock held by another
- Requesting application is suspended in a “lock wait” mode until:
 - Transaction causing conflict releases lock (i.e., COMMIT, ROLLBACK or FORCE APPLICATION)
 - Lock timeout (or deadlock) occurs

Application 1

```
T1: update department
      set mgrno = '000350'
      where deptno = 'F01'
```

```
T2: select * from employee
      where empno = '000350'
```

T₃: lock-wait

LOCK-WAIT

DEPARTMENT

Deptno	Name	Mgrno	Loc
...
F01	xxx	xxx	xxx
...

EMPLOYEE

Empno	Name	Job	Sal
...
000350	xxx	xxx	xxx
...

Application 2

```
T1: update employee
      set job = 'MANAGER'
      where empno = '000350'
```

Deadlock Scenario – Example – Cursor Stability

Transaction A	Transaction B
update T1 set col1 = ? where col2 = 2	
	update T2 set col1 = ? where col2 = ?
select * from T2 where col2 >= ?	
	select * from T1 where col5 = ? and col2 = ?

Waiting because is
reading uncommitted data

Waiting because is
reading uncommitted data

DEADLOCK!! 

Deadlock

- **All deadlocks are considered abnormal**
- **Indicators include processing delays and poor performance**
- **Deadlock slows down the participant transaction while it waits for deadlock detection and resolution**
 - Wastes system resources by rolling back victim transaction
 - Causes extra system work
 - Transaction log access
- **Deadlocks or retry logic in the application causing transactions being re-executed**
 - The victim application has to re-execute the transaction from the beginning after ROLLBACK

DB2 Deadlock Detector

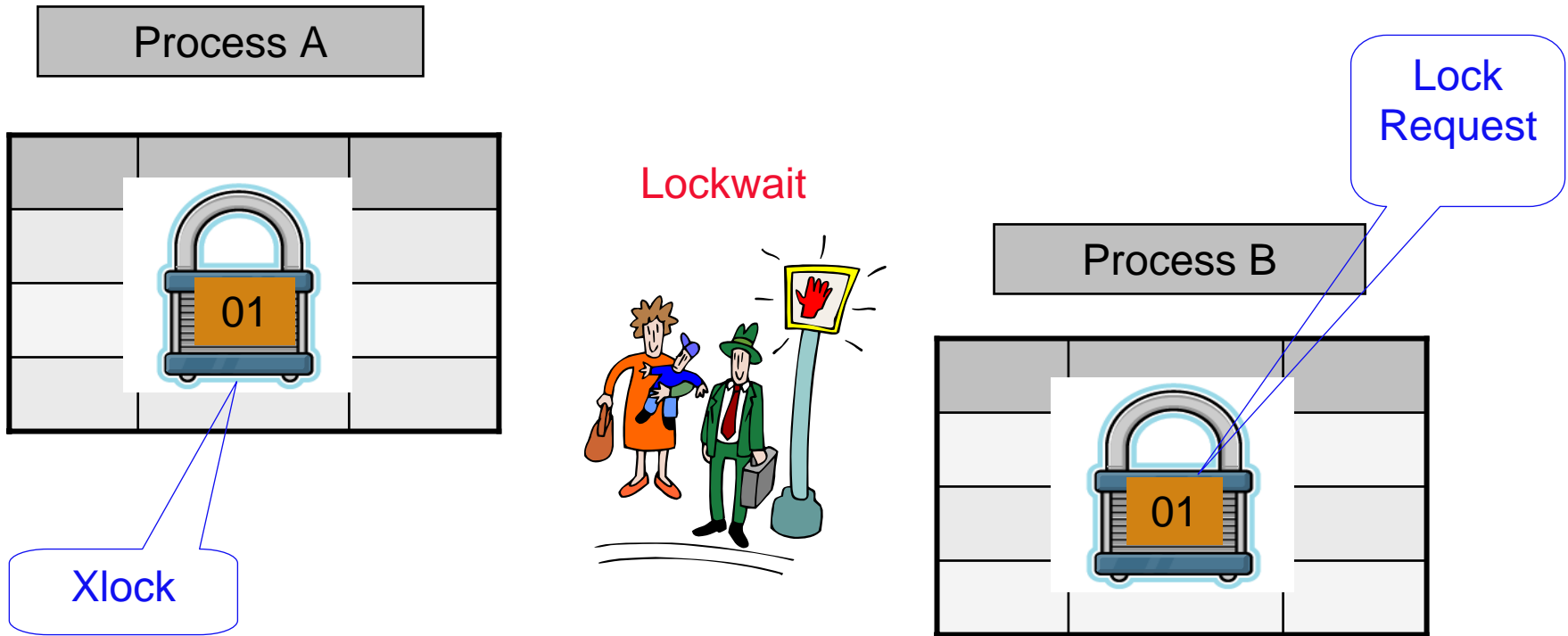
- **Responsible resolving deadlocks**

- When a deadlock is detected, the deadlock detector will choose a victim that will be automatically rolled back
 - Rolling back the victim, the lock conflict is removed, and the other application can continue processing.
 - The victim is chosen “arbitrarily”
 - DB2 deadlock resolution will return a SQL error code of SQL0911N with a reason code of “2” to the victim process

- **DB CFG parameter DLCHKTIME**

- Defines frequency the database manager checks for deadlocks
 - A high value increases the deadlock check time and reduces overhead of checking but could result in applications being stuck in deadlock for longer periods of time
 - A low value allows for deadlocks to be detected sooner, however it also introduces additional overhead for checking more frequently

Lock Timeout Scenario



DB2 waits for a specified period of time, then returns a SQL error code of SQL0911N with a reason code of "68" to the waiting process

LOCKTIMEOUT and SET LOCK MODE

- You can set lock wait behavior on a session level rather than using the global value specified by the DB CFG parameter LOCKTIMEOUT

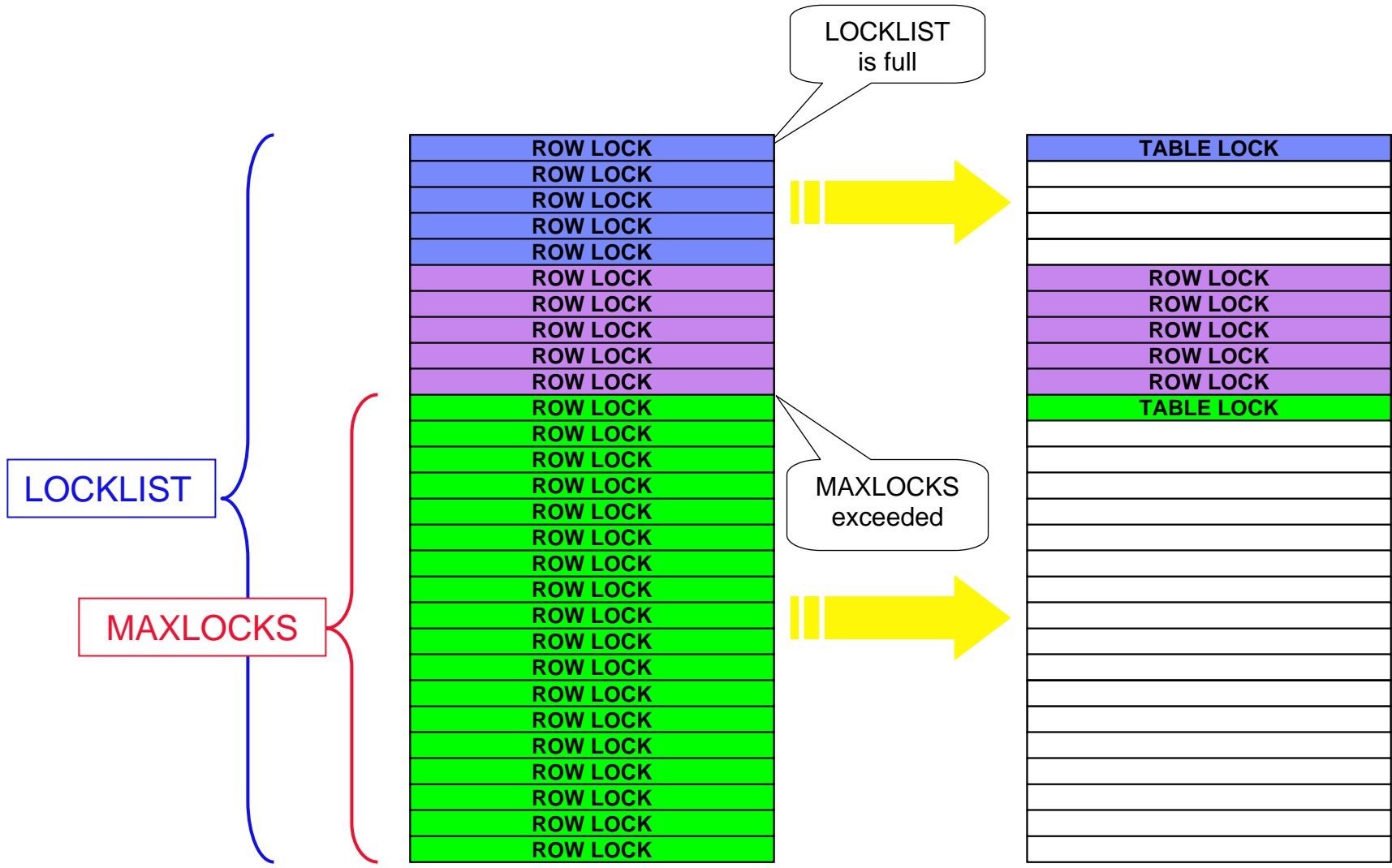
```
SET LOCK MODE TO ---+-- NOT WAIT-----+---|
                |                               |
                +-- WAIT -----+-----+
                               |               |
                               +-integer-constant--+
```



Lock Escalations

- **Lock escalation can occur in two different scenarios:**
 - A single application requests a lock that will exceed its allowable number of locks
 - An application triggers lock escalation because the maximum number of database locks for the entire database exhausted
- **The database manager will attempt obtaining table locks and releasing the existing row locks**
 - The desired effect is to make more lock memory available for other applications
- **The following database parameters have a direct affect on lock escalation:**
 - LOCKLIST - total number of 4k pages allocated for lock storage (please refer to the memory model)
 - MAXLOCKS - allowable percentage of locklist that can be used by a single application
- **Tuning and monitoring may be necessary**
 - Less of an issue if STMM is managing memory for locks
- **Workload and query behavior dictate locking patterns**

Lock Escalation Scenarios



Monitoring/Identifying Locking Issues in General

Monitoring:

- Create/configure/enable locking event monitors to capture details on lock event data for a workload or database.
- Use db2pd –locks (wait)
- Review administration notification log with DIAGLEVEL set to 4 (DB2 Administration Notification Log (<instance>.nfy) or DB2 Diagnostic Log (db2diag.log))
 - Find the agent that is waiting (event monitor), and corresponding requested lock

Indicative Signs of Locks:

- Due to the relatively transient nature of locking events, lock event data is most valuable if collected periodically over time
- One or more applications are occasionally re-executing transactions
- Key indicator monitoring elements:
 - lock_timeouts value is increasing of lock wait time
 - int_rollbacks value is increasing
 - int_deadlock_rollbacks shows increase in number of roll backs due to deadlock events
 - log_disk_wait_time shows increase in amount of time agents spend waiting for logs to be flushed to disk
 - Check monitoring element lock_escals for indications that lock escalations may be a contributing factor

Reducing Locking Occurrences in General

Commit the following actions as soon as possible:

- Write actions such as DELETE, INSERT, and UPDATE
- Data definition language (DDL) statements (e.g. ALTER, CREATE, and DROP statements)
- BIND and REBIND commands
- **Avoid fetching result sets that are larger than necessary**
 - The more that rows are touched, the more that locks are held, the greater the opportunity to run into a locking problem
 - Push down row selection criteria into a WHERE clause of the SELECT statement
 - As opposed to returning rows and filtering them at the application
- **Avoid using a higher isolation level than necessary**
- **Use WITH RELEASE clause with CLOSE CURSOR statement**

Resolving Deadlock Issues

- **Deadlock frequency can be reduced by ensuring that all applications access common data in the same order**
 - When two applications take incompatible locks on the same objects in different order, they run a much larger risk of deadlocking
- **Avoid concurrent DDL operations if possible**
 - For example, DROP TABLE statements can result in a number of catalog updates as rows might have to be deleted for the table indexes, primary keys, check constraints in addition to the table
 - If other DDL operations are dropping or creating objects, there can be lock conflicts and even occasional deadlocks

Eliminating Lock Escalations

- **Combination of good application design and database configuration can minimize or eliminate lock escalations**
 - If possible, acquire an explicit table lock with LOCK TABLE statement
 - Minimizes the DB2 workload and reduces the associated system resources needed
- **If not using STMM, manually adjust MAXLOCKS or LOCKLIST**
 - Their values may be too small for your current workload
 - If multiple applications are experiencing lock escalation, this could be an indication that the LOCKLIST needs to be increased
 - If only one application is experiencing lock escalations, then adjusting MAXLOCKS could resolve this issue

Agenda

- Locking and Performance
- Identifying Locking Scenarios
- **Using Isolation Levels**
- Monitoring Locking Issues
- Avoiding Locking Scenarios



Isolation Levels

- DB2 provides different levels of protection to isolate data

Isolation Level	Dirty Read “Uncommitted Read”	Non-repeatable Read	Phantom Read
Repeatable Read (RR)	-	-	-
Read Stability (RS)	-	-	Possible
Cursor Stability (CS)	-	Possible	Possible
Uncommitted read (UR)	Possible	Possible	Possible

DEFAULT

DEFAULT

- Isolation level can be specified at many levels
 - Connection
 - Session (application)
 - Statement
- For Embedded SQL, the level is set at bind time
- For Dynamic SQL, the level is set at run time

Concurrency Control in DB2 9.7

- Currently Committed is a **variation on DB2's Cursor Stability isolation**
 - If uncommitted row-change found, use currently committed version of data
 - Avoids timeouts and deadlocks
 - Log based:
 - No management overhead

Cursor Stability

Situation	Result
Reader blocks Reader	No
Reader blocks Writer	No
Writer blocks Reader	Yes
Writer blocks Writer	Yes



Currently Committed

Situation	Result
Reader blocks Reader	No
Reader blocks Writer	No
Writer blocks Reader	No
Writer blocks Writer	Yes

Currently Committed – How does it work?

Transaction A	Transaction B
update T1 set col1 = ? where col2 = 2	
	update T2 set col1 = ? where col2 = ?
select * from T2 where col2 >= ?	
	select * from T1 where col5 = ? and col2 = ?
	commit
commit	

No locking

→ Reads last committed version
of the data

No locking

← Reads last committed version
of the data

No deadlocks, no timeouts in this scenario!

DB2 9.7 upgrade considerations for CUR_COMMIT

- For new databases, the default is set to ON
 - By default, a query will return the currently committed value of the data at the time when your query is submitted
- During database upgrade, the **cur_commit** configuration parameter is set to DISABLED to maintain the same behavior as in previous releases
- If you want to use currently committed on cursor stability scans, you need to set the **cur_commit** configuration parameter to **ON** after the upgrade

Agenda

- Locking and Performance
- Identifying Locking Issues
- Using Isolation Levels
- **Monitoring Locking Issues**
- Avoiding Locking Scenarios



Monitoring Locking, Deadlocks and Escalations

- **To help identify locking issues, use:**
 - Administrative Table Functions and Views
 - DBM CFG for DFT_MON_LOCKS must be ON for Snapshot table functions to report accurately
 - Application and Database Lock Snapshots
 - Lock and Deadlock Event Monitors
 - db2pd
- **Check Administration Notification Log for:**
 - Lock Escalations
 - Lock Waits

Administrative View – SYSIBMADM.SNAPDB

```
SELECT LOCKS_HELD, LOCK_WAITS, LOCK_WAIT_TIME,  
       DEADLOCKS, LOCK_ESCALS, LOCKS_WAITING,  
       LOCK_TIMEOUTS, INT_DEADLOCK_ROLLBACKS  
FROM SYSIBMADM.SNAPDB
```

LOCKS_HELD	LOCK_WAITS	LOCK_WAIT_TIME	DEADLOCKS	LOCK_ESCALS	LOCKS_WAITING	LOCK_TIMEOUTS	INT_DEADLOCK_ROLLBACKS
11	16	817243	3	0	1	8	3

1 record(s) selected.

Total Locks
waits

Total
Deadlocks

Total Lock
Escalations

Current
Locks
waiting

Total Lock
Timeouts

Administrative View – SYSIBMADM.LOCKWAITS

SYSIBMADM.LOCKWAITS contains information on locks

```
SELECT SUBSTR(TABSCHEMA,1,8) AS TABSCHEMA,  
       SUBSTR(TABNAME,1,15) AS TABNAME,  
       LOCK_OBJECT_TYPE,  
       LOCK_MODE,  
       LOCK_MODE_REQUESTED,  
       AGENT_ID_HOLDING_LK  
       FROM SYSIBMADM.LOCKWAITS;
```

TABSCHEMA	TABNAME	LOCK_OBJECT_TYPE	LOCK_MODE	LOCK_MODE_REQUESTED	AGENT_ID_HOLDING_LK
DB2INST1	DEPARTMENT	TABLE_LOCK	X	S	40

1 record(s) selected.

Administrative View – SYSIBMADM.LOCKS_HELD

This administrative view contains information on current locks held

```
SELECT SUBSTR(TABSCHEMA,1,8) AS TABSCHEMA,  
       SUBSTR(TABNAME,1,15) AS TABNAME,  
       AGENT_ID,  
       LOCK_OBJECT_TYPE,  
       LOCK_MODE,  
       FROM SYSIBMADM.LOCKS_HELD;
```

TABSCHEMA	TABNAME	AGENT_ID	LOCK_OBJECT_TYPE	LOCK_MODE
DB2INST1	DEPARTMENT	21	TABLE_LOCK	S

Table
Level

Share
Lock

Deadlock Event Monitor – File Output

```
CREATE EVENT MONITOR dead_events FOR DEADLOCKS WITH DETAILS WRITE TO FILE 'path'
```

```
EVENT LOG HEADER
```

```
Event Monitor name: DEAD_EVENTS
```

```
Server instance name: db2inst1
```

```
Database Name: SAMPLE
```

What Instance?

What Database?

```
3) Deadlock Event ...
```

```
Deadlock ID: 1
```

```
Number of applications deadlocked: 2
```

```
Deadlock detection time: 08/27/2009 10:17:31.280624
```

```
Rolled back Appl participant no: 2
```

When did the deadlock occur?

```
4) Connection Header Event ...
```

```
Client Process Id: 10749
```

Victim's process id

```
5) Deadlocked Connection
```

```
Participant no.: 2
```

```
Deadlock detection time: 08/27/2009 10:17:31.282780
```

```
Table of lock waited on : EMPLOYEE
```

```
Schema of lock waited on : DB2INST1
```

```
Type of lock: Row
```

```
Mode of lock: S - Share
```

```
Mode application requested on lock: X - Exclusive4
```

Victim

Requested Resources

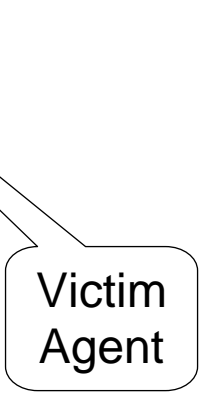
Deadlock Event Monitor – Table Output

```
CREATE EVENT MONITOR DEAD_EVENTS FOR DEADLOCKS WITH DETAILS WRITE TO TABLE
```

```
SELECT DEADLOCK_ID,  
       ROLLED_BACK_AGENT_ID,  
       START_TIME  
       FROM DEADLOCK_DEAD_EVENTS;
```

DEADLOCK_ID	ROLLED_BACK_AGENT_ID	START_TIME
4	117	2009-08-26-19.50.47.030134

1 record(s) selected.



Victim
Agent

Monitoring db2diag.log for Locks Waits

DBM CFG parameter DIAGLEVEL set to 4 records lock timeouts

```
2009-09-04-10.16.41.126755-240 E5543901G631 LEVEL: Info
PID      : 6974          TID   : 2950687648  PROC  : db2sysc 0
INSTANCE: db2inst1     NODE  : 000        DB    : SAMPLE
```

```
...
...
...
```

```
Request for lock "TAB: (2, 6)" in mode ".IX" timed out
Application caused the lock wait is
"*LOCAL.db2inst1.090904141554"
```

Statement:

DATA #2 : String with size, 41 bytes

```
update employee set edlevel = edlevel + 1
```

Attempt to acquire
a table lock

Lock requested
Intent eXclusive
(IX)

Statement causing
the error

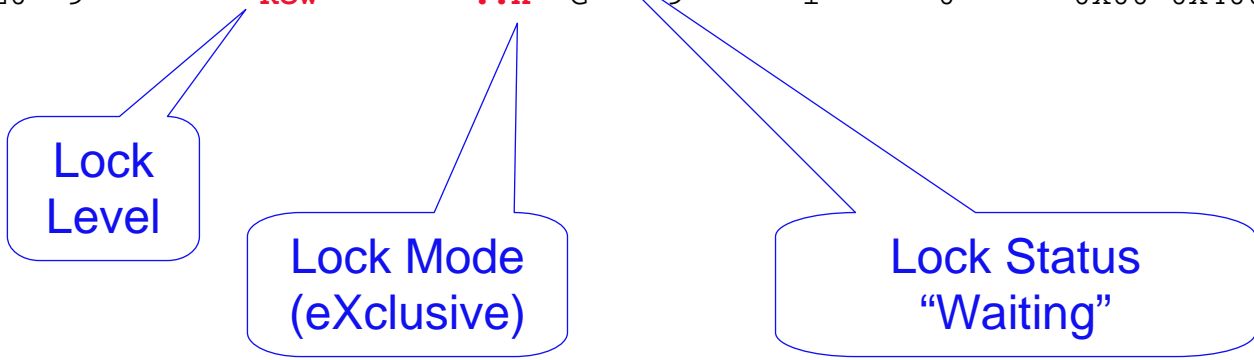
Monitoring database locks using db2pd

```
db2pd -db sample -locks wait
```

```
Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 07:00:28
```

Locks:

Address	TranHdl	Type	Mode	Sts	Owner	Dur	HoldCount	Att	ReleaseFlg
0xA5AB63F0	8	Row	..S	W	9	1	0	0x10	0x00000002
0xA5AB61E0	9	Row	..X	G	9	1	0	0x00	0x40000000



Monitoring Currently Committed using db2pd

```
db2pd -db sample -logs
```

```
Database Partition 4294967295 -- Database SAMPLE -- Active -- Up 0 days
00:08:19
```

Logs:

```
Current Log Number          0
Pages Written                0
Cur Commit Disk Log Reads  0
Cur Commit Total Log Reads 9
Method 1 Archive Status     n/a
Method 1 Next Log to Archive n/a
Method 1 First Failure      n/a
Method 2 Archive Status     n/a
Method 2 Next Log to Archive n/a
Method 2 First Failure      n/a
```

Currently Committed
Log Records read

Address	StartLSN	State	Size	Pages	Filename
0xA363D598	0000000003A98010	0x00000000	1000	1000	S0000000.LOG
0xA363DDD8	0000000003E80010	0x00000000	1000	1000	S0000001.LOG
0xA363E618	0000000004268010	0x00000000	1000	1000	S0000002.LOG

Monitoring - Lock Escalation

- **With DBM CFG parameter DIAGLEVEL set to 3 (default) or 4, Lock Escalations are reported in the db2diag.log:**

```
2009-07-13-16.27.49.115000-240 E1444H457          LEVEL: Warning
PID       : 2800                TID  : 3444          PROC  : db2syscs.exe
INSTANCE: DB2                  NODE  : 000          DB    : SAMPLE
APPHDL   : 0-146               APPID: *LOCAL.DB2.050713222002
FUNCTION: DB2 UDB, data management, sqlEscalateLocks, probe:3
MESSAGE  : ADM5502W The escalation of "240671" locks on table "DB2ADMIN.EMPLOYEE"
           to lock intent "X" was successful.
```

- **Additionally:**
 - Health Center: "lock escalation rate" and "lock list utilization" indicators
 - Snapshot Monitor: Database and Application Snapshots' "Lock escalations" and "Exclusive lock escalations"
 - Event Monitor: Connections' "Lock escalations" and Databases' "Lock escalations", "X lock escalations"

Agenda

- Locking and Performance
- Identifying Locking Issues
- Using Isolation Levels
- Monitoring Locking Issues
- **Avoiding Locking Scenarios**



Avoiding Locking Scenarios

- **Best Practices – Application**

- Use least restrictive isolation level that maintains the data integrity requirements of the application
- Reduce Isolation level of specific statements by using statement level isolation (i.e., WITH clause)
- CLOSE cursors WITH RELEASE to free locks prior to end of transaction
- Perform updates as close to the end of the transaction as possible, to reduce exclusive lock duration
- COMMIT frequently to release locks

- Avoid multiple applications accessing the same tables, but acquiring locks in different orders (Access patterns should be similar)
- Avoid having multiple processes that access the same table for both reads and writes within the same transaction

Avoiding Locking Scenarios

- **Best Practices – Database**

- Avoid lock escalations by increasing DB CFG parameters LOCKLIST and/or MAXLOCKS
- Avoid lock timeouts:
 - Adjust the DB CFG parameter LOCKTIMEOUT or use the SET CURRENT LOCK TIMEOUT command
- Avoid deadlocks:
 - Reduce row blocking during index and table scans:
 - DB2_SKIPINSERTED to skip/ignore uncommitted inserted rows
 - DB2_SKIPDELETED to skip/ignore uncommitted deleted rows
 - DB2_EVALUNCOMMITTED to defer locking until row is known to satisfy query. Uncommitted data will be evaluated. Skips deleted rows on table scans.

More Useful Registry Variables for Locking

- **DB2_KEEPTABLELOCK**
 - allows DB2 to maintain the table lock when an uncommitted read or cursor stability isolation level is closed. The table lock is released at the end of the transaction
- **DB2_MAX_NON_TABLE_LOCKS**
 - defines the maximum number of NON table locks a transaction can have before it releases these locks. Because transactions often access the same table more than once, retaining locks and changing their state to NON can improve performance
- **DB2LOCK_TO_RB**
 - specifies whether lock timeouts cause the entire transaction to be rolled back, or only the current statement



DB2 Performance Clinic - Locking

Information Management - Partner Technologies
IBM Software Group



DB2 Performance Clinic – Logging

Information Management Partner Technologies

Agenda

- **Logging Concepts**
- **Configuration and Performance**
- **Logging Bottlenecks**
- **Reducing Logging**
- **Monitoring and Tuning**

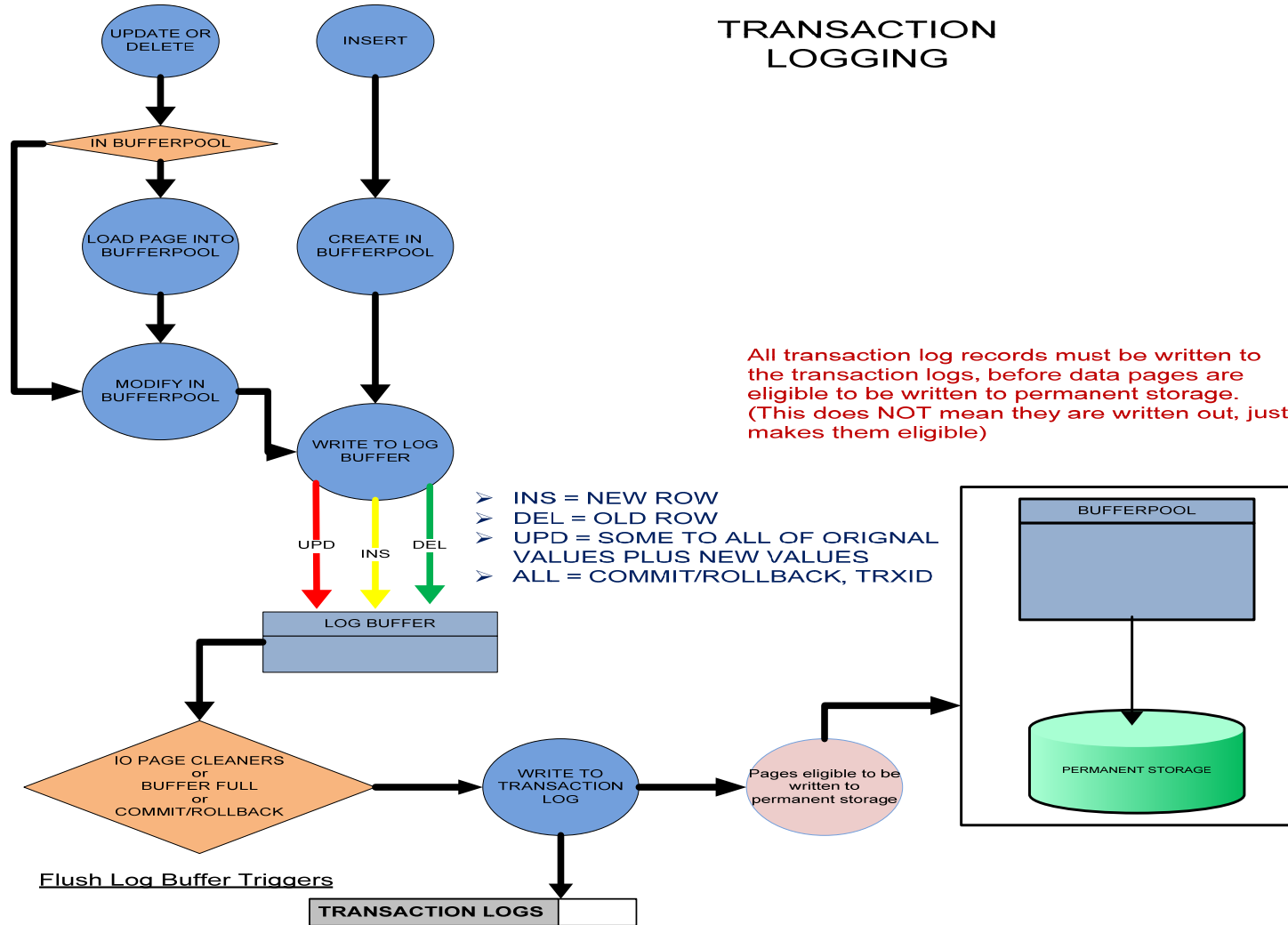


Agenda

- **Logging Concepts**
- Configuration and Performance
- Logging Bottlenecks
- Reducing Logging
- Monitoring and Tuning



TRANSACTION LOGGING OVERVIEW

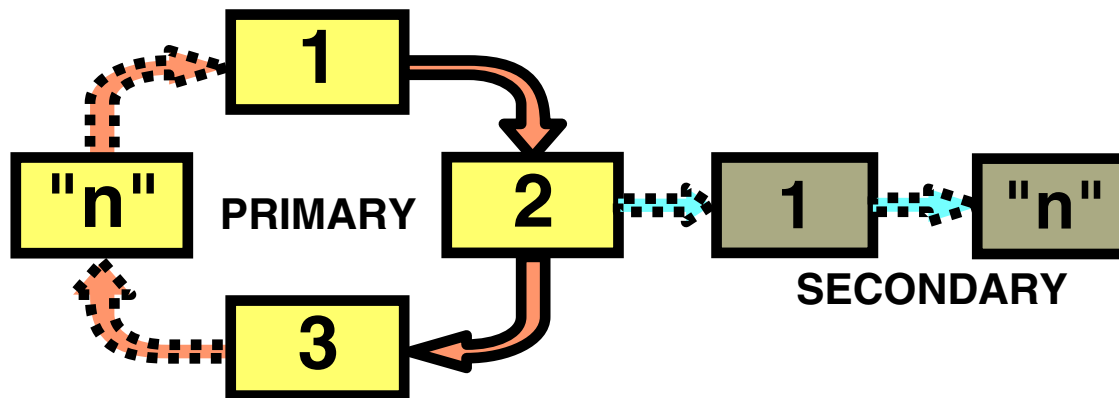


DB2 Logging Concepts

- **Records database transactions**
 - If there is a crash, the Logs are used to playback/redo committed transactions during recovery
- **Logging is always “ON” for regular tables in DB2**
 - Possible to mark some tables or columns as NOT LOGGED
 - Possible to log USER temporary tables
- **Transaction or Unit of Work (UOW)**
 - A sequence of one or more SQL statements
 - Initiated by the first executable SQL statement after connecting to the database
 - UOW terminates with a COMMIT or ROLLBACK
- **DB2 implements two types of logging scenarios:**
 - Circular logging (default)
 - Archive logging

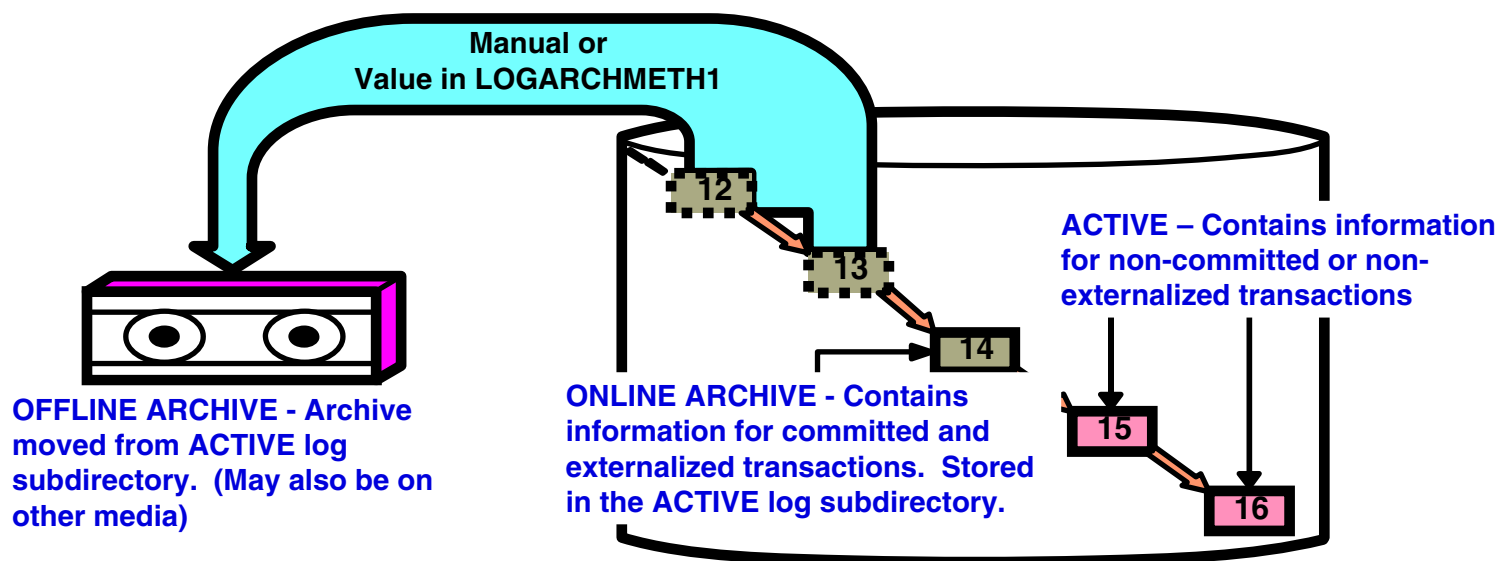
Circular Logging

- Primary log files (LOGPRIMARY) used to record all changes; reused when changes are committed
- Secondary log files (LOGSECOND) allocated when limit of primary log files reached
- If both primary and secondary log limit is reached, an error code is returned. If the file system (LOGPATH) has insufficient space, a “log disk full error” will be raised
- Crash and version recovery possible; roll-forward recovery not possible



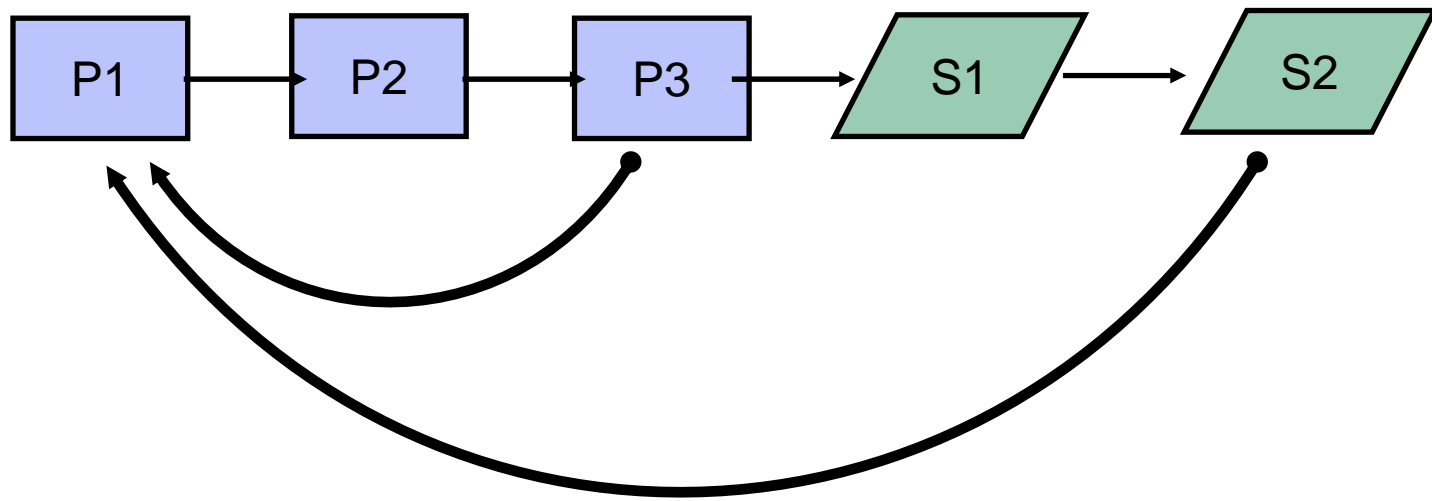
Archive Logging

- Used for roll forward recovery. It is enabled with the LOGARCHMETH1 database configuration parameter
- Logs are archived when no longer active (i.e. not required for crash recovery). They are not deleted; they are kept online or offline
- Rollforward recovery and online backup are possible



Primary and Secondary Logs

- Primary logs are PREALLOCATED
- Secondary logs are ALLOCATED as needed to handle spikes in workload
- For day to day operations, ensure that you stay within your primary log allocation



Agenda

- Logging Concepts
- **Configuration and Performance**
- Logging Bottlenecks
- Reducing Logging
- Monitoring and Tuning



Logging Configuration and Performance

LOGARCHMETH1 and LOGARCHMETH2

LOGARCHOPT1 and LOGARCHOPT2

LOGPATH and NEWLOGPATH

MIRRORLOGPATH

OVERFLOWLOGPATH

BLK_LOG_DSK_FUL

MAX_LOG

MINCOMMIT

NUM_LOG_SPAN

FAILARCHPATH

NUMARCHRETRY

ARCHRETRYDELAY

LOGPRIMARY

LOGSECOND

LOGBUFSZ

LOGFILSIZ

**LOGPATH
NEWLOGPATH
LOGPRIMARY
LOGSECONDARY
LOGBUFSZ
LOGFILSIZ
MINCOMMIT**

Primary logs (LOGPRIMARY)

- **Characteristics**

- The primary log files establish a fixed amount of storage allocated to the transaction log files
- A primary log requires the same amount of disk space whether it is full or empty
- The maximum number of primary logs is 256, the default is 3

- **Impact**

- One can encounter a “log-full” condition if configured with an insufficient number

Secondary logs (LOGSECOND)

- **Characteristics**

- If the primary log files become full, secondary log files are allocated, as needed up to the maximum specified by LOGSECOND
- Once allocated, they are not deleted until the database is deactivated
- The maximum number of primary and secondary log files allowed (logprimary + logsecond), gives an upper limit of 1024 GB of active log space
- Setting of -1 for LOGSECOND enables “infinite logging”

- **Impact**

- Infinite logging could impact performance if a log file has to be brought back from an archive for ROLLBACK

Log file size (LOGFILSIZ)

- **Characteristics**

- DB CFG parameter defines the size of each primary and secondary log file in 4K pages

- **Impact**

- The size of the log file has a direct bearing on performance
 - “Too small”
 - Overhead of archiving more old log files
 - Allocating new log files more frequently
 - “Too big”
 - Logistics of managing large files during archival process
 - Risking the loss of a greater quantity of transactions if a log cannot be recovered

Log buffer size (LOGBUFSZ)

- **DB CFG parameter specifying the amount of memory allocated as a buffer for more efficient log file I/O**
 - Not managed by Self-Tuning Memory Manager (STMM)
- **The log buffer is written to disk when:**
 - A transaction commits or a group of transactions commit, as defined by the mincommit configuration parameter
 - The log buffer becomes full
 - LSN Gap trigger – too many dirty pages in buffer pool
- **Default value usually not large enough for OLTP databases, 256 is a good starting point:**

```
db2 UPDATE DB CFG FOR sample USING LOGBUFSZ 256
```

Number of Commits to Group (MINCOMMIT)

- **DB CFG parameter specifying the number of COMMIT statements to group before triggering the writing of the Log Buffer contents to the Log Files**
 - Improve performance when you have multiple applications performing commits within short time frame
 - Number of I/O operations reduced by grouping multiple commits together.
- **Grouping will only occur when the value of this parameter is greater than 1 (default) AND the number of applications connected to the database is greater than or equal to this parameter**
- **CAUTION: Should only be changed in very specialized situations of very high OLTP batch transactional activity and should not be set very high**

```
db2 UPDATE DB CFG FOR sample USING MINCOMMIT 3
```

New Log Path (NEWLOGPATH)

- **DB CFG parameter which is used to specify a new location of the log files**
 - Set only when relocating the log files, otherwise the parameter has no value
- **Ideally, the log files will be on a physical disk not shared with the database or other applications**
 - Recommended to use RAID-10 for logs to reduce the chance of losing log files due to disk failures
- **I/O characteristics of logs are very different from DB2 containers**
 - Heavy serial write activity

```
db2 UPDATE DB CFG FOR sample USING NEWLOGPATH <path>
```

General Recommendations

- **Location of Database logs**
 - Need to be on their own physical disk
 - A fast I/O device for the log is recommended
- **LOGFILSIZ**
 - Increase beyond default, e.g. 5000 4K pages or more
- **LOGPRIMARY**
 - Allocate all logs as primary logs
 - Use LOGSECOND for “emergency” space only
- **LOGBUFSZ**
 - Increase to 256 or greater
- **MINCOMMIT**
 - Keep this at default of 1, typically no higher than 2-3

Use DB2 and Operating System tools to monitor logging activity

Agenda

- Logging Concepts
- Configuration and Performance
- **Logging Bottlenecks**
- Reducing Logging
- Monitoring and Tuning



Log Bottleneck

- **Anything sharing the disks?**
- **High transaction rate?**
- **High data volume?**
- **Too frequent commits?**
- **Mincommit too low?**
- **Logging too much data?**

Logging Bottlenecks – Disk

- Will interfere with **all** DML activity and cause **COMMITs** and **ROLLBACKs** to take longer
- Can be very performance sensitive, especially in an OLTP environment – a good place to use your best hardware
 - Dedicated disks – separate from tablespaces, etc.
 - Fast disks
 - RAID parallelization with small (e.g. 8k) stripe size
 - Fast controller with write caching
- **Is anything using the same disks?**
 - Can be difficult to determine conclusively
 - Partitioned disks, logical volumes make it difficult to be sure what's on the same disk spindles
 - Check tablespace container paths, database directory, other utilities, etc.

Logging Bottlenecks – High Data Volume

- **What is High Data Volume?**

- iostat (or perfmon) shows larger average I/O (e.g. > 8k), < 50 IO/s

- **Possible Remedy**

- Can you reduce amount of logged data?

- Alter table design (i.e., group frequently updated columns, ideally at end of the row)
 - Use ALTER TABLE ... NOT LOGGED INITIALLY for “bulk” operations
 - Use LOAD utility to insert large amounts of data.
 - Use TRUNCATE command instead of DELETE to empty a table
 - Use Data Compression of data and indexes.
 - Log records are also compressed when using compression which can help reduce I/O traffic
 - If DGTG/CGTTs are being used set NOT LOGGED
 - Larger I/Os can also be due to a poor performing logging disk

Agenda

- Logging Concepts
- Configuration and Performance
- Logging Bottlenecks
- **Monitoring and Tuning**
- Reducing Logging



Identifying the Log Files location

1. `df -k`

```
Filesystem            1K-blocks      Used Available Use% Mounted on
/dev/sda2             10847448    9537288    759132   93% /
/dev                  517576        96    517480    1% /dev
/dev/sdb1             38456308   1837808   34665000    6% /db2fs
```

2. Examine the database configuration parameters 'Path to log files'

```
db2 get db config for sample | grep -i 'path to log files'
Path to log files = /db2fs/db2inst1/NODE0000/SQL00006/SQLLOGDIR/
```

3. Verify that the transaction logs are not sharing filesystems or logical devices.

In this example the transaction logs are sharing the same location as table space containers

```
SELECT SUBSTR(TBSP_NAME,1,20) AS TBSP_NAME, INT(TBSP_ID) AS TBSP_ID,
       SUBSTR(CONTAINER_NAME,1,45) AS CONTAINER_NAME
FROM SYSIBMADM.CONTAINER_UTILIZATION
```

TBSP_NAME	TBSP_ID	CONTAINER_NAME
SYSCATSPACE	0	/db2fs/db2inst1/NODE0000/SAMPLE/T0000000/C000
TEMPSPACE1	1	/db2fs/db2inst1/NODE0000/SAMPLE/T0000001/C000
USERSPACE1	2	/db2fs/db2inst1/NODE0000/SAMPLE/T0000002/C000

SNAPSHOT – Commits and Rollbacks

`db2 get snapshot for all on sample`

Commit statements attempted	= 11
Rollback statements attempted	= 0
Dynamic statements attempted	= 524
Static statements attempted	= 16
Failed statement operations	= 0
Select SQL statements executed	= 171
Xquery statements executed	= 0
Update/Insert/Delete statements executed	= 9
DDL statements executed	= 0
Inactive stmt history memory usage (bytes)	= 0

How many
Commits

How many
Dynamic
statements

1. GET SNAPSHOT FOR All ON sample
2. Locate Log section with Commits/Rollback
3. Reference Commit, Rollback, Dynamic, Static, etc.
4. Trend log information

SNAPSHOT – Log Pages

db2 get snapshot for database on sample

```

Log space available to the database (Bytes)= 8286039447
Log space used by the database (Bytes)      = 37160553
Maximum secondary log space used (Bytes)   = 0
Maximum total log space used (Bytes)      = 7720996823
Secondary logs allocated currently         = 0
Log pages read                             = 13000
Log read time (sec.ns)                    = 0.000000004
Log pages written                          = 12646941
Log write time (sec.ns)                   = 875.000000004
Number write log IOs                      = 1167739
Number read log IOs                      = 5
Number partial page log IOs              = 105768
Number log buffer full                    = 221
Log data found in buffer                  = 200
  
```

How many log reads

Read
latency

How many
log writes

Write
latency

1. Locate log section
2. GET SNAPSHOT FOR DATABASE ON
3. Reference log reads and writes
4. Trend log information:
 - If there are a large 'Number Read Log IOs' relative to 'Log Data found in buffer', you need to tune up the LOGBUFSZ
 - If 'Number of log buffer full' is high, increase LOGBUFSZ
 - 'Log write time/'Number write log IOs' is important. <=2ms is desirable

Administrative View – LOG_UTILIZATION

```
SELECT substr(db_name, 1,10) DB_NAME,  
       log_utilization_percent, total_log_used_kb,  
       total_log_available_kb  
       FROM SYSIBMADM.LOG_UTILIZATION;
```

DB_NAME	LOG_UTILIZATION_PERCENT	TOTAL_LOG_USED_KB	TOTAL_LOG_AVAILABLE_KB
SAMPLE	21.65	0	19902

1 record(s) selected.

Percent utilization of
total log space!

This administrative view contains information about log utilization.

Administrative View – SNAPDB

```

SELECT VARCHAR(DB_NAME,20) AS DBNAME,
       CASE WHEN (commit_sql_stmts + rollback_sql_stmts) > 0
       THEN DEC((1000 * (log_write_time_s / (commit_sql_stmts +
       rollback_sql_stmts))),5,0)
       ELSE NULL
       END AS LogWriteTime_PER_1000TRX,
       log_write_time_s AS LOGWTIME,
       commit_sql_stmts + rollback_sql_stmts AS TOTALTRANSACTIONS
FROM sysibmadm.snapdb;

```

DBNAME	LOGWRITETIME_PER_1000TRX	LOGWTIME	TOTALTRANSACTIONS
SAMPLE	10	20	2000

1 record(s) selected.

Cumulative average time the agent waited per 1000 transactions

This administrative view contains amount of time an agent waits for log buffer to be flushed.

Agenda

- Logging Concepts
- Configuration and Performance
- Logging Bottlenecks
- Monitoring
- **Reducing Logging**



Reducing the overhead of Transaction Logging

- **NOT LOGGED option for LOB and CLOB data**
 - Large object (CLOB) columns are logged by default, if the data they contain is recoverable from outside of the database mark these columns as NOT LOGGED, to reduce the volume of data being logged during insert, update, or delete operations
- **ALTER TABLE... NOT LOGGED INITIALLY**
 - If the excessive log volume is correlated with bulk SQL operations, the target table can be set to NOT LOGGED INITIALLY
- **NOT LOGGED option for temporary tables**
 - Declared Global Temporary Tables (DGTTs)
 - Created Global Temporary Tables (CGTTs)
- **Use LOAD utility for inserting large amounts of data**
 - Load does not go through the SQL engine, therefore it is high speed and logging can be avoided

Reducing the overhead (continued...)

- **Reduce the number of COMMITs**
 - Modify the applications such that commits are performed less often
- **Grouping frequently updated columns**
 - Placing columns that are frequently modified next to one another in the table definition
 - This can reduce the volume of data that is logged during updates
 - They are ideally defined at the end of the row's definition
- **Increase MINCOMMIT**
 - MINCOMMIT directs the database manager to delay the writing of the log records to disk until a minimum number of commits have been performed
- **Use TRUNCATE to empty a table**
 - Truncating a table will avoid the logging activity of DELETE *
- **Use Data Compression to compress data and indexes**
 - Log records are also compressed when using compression which reduces I/O traffic

Log Disk Configuration

- **Rules of thumb:**

- Consider a pair of dedicated RAID-1 log disks for up to around 400 (moderately write-heavy) DB2 transactions per second
- Consider RAID-10 striping multiple disks with 8k stripe size for greater
- Number of log disks and the need for write buffering on the controller is affected by the transaction rate and the amount of data written per transaction



DB2 Performance Clinic – Logging

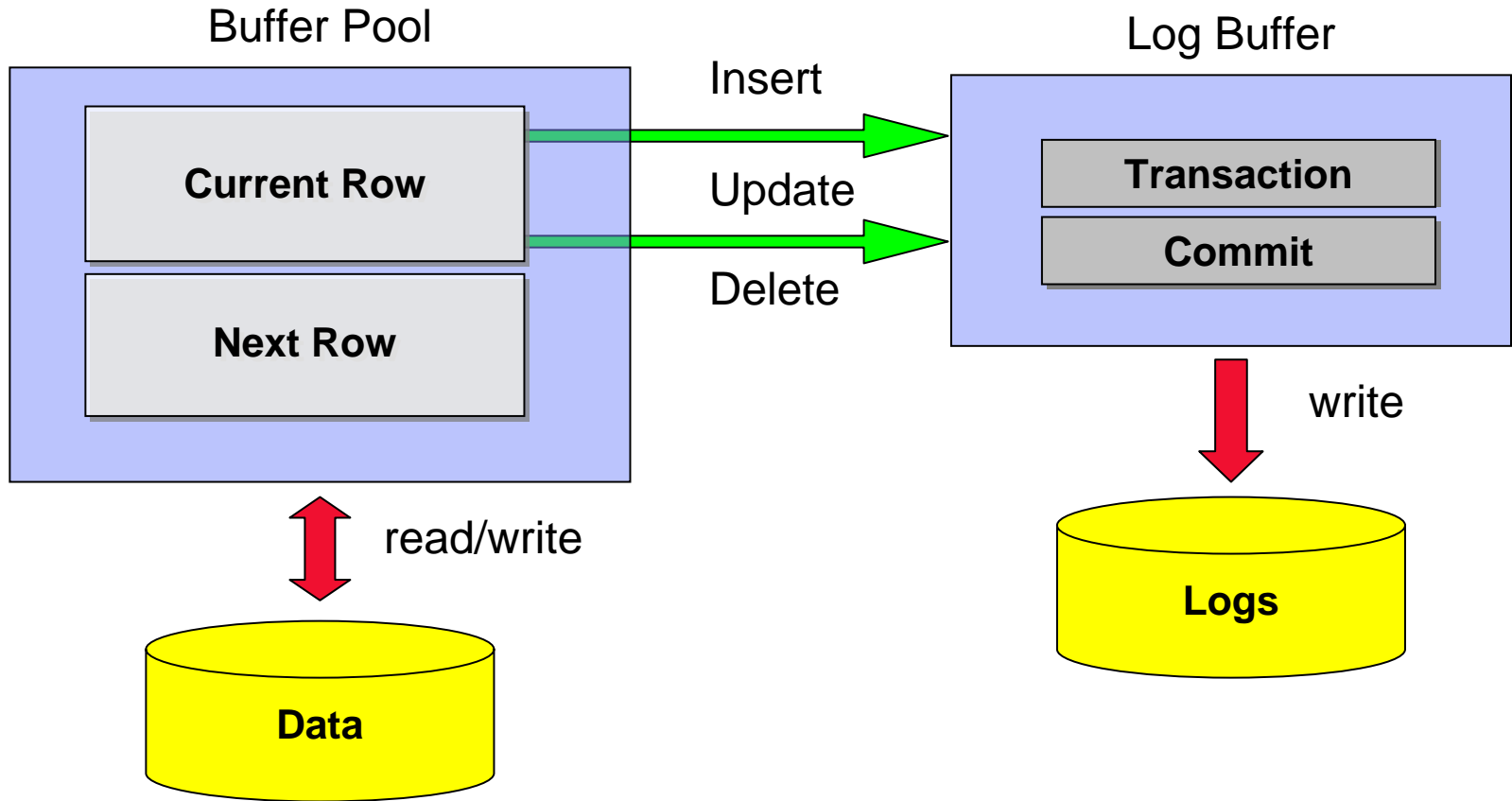
Information Management Partner Technologies

How large should LOGBUFSZ be? (new w/b 17)

- **minimize "number log buffer full" in db snapshot**
- **minimize percentage of log reads from disk for currently committed (seen with db2pd -logs)**
- **don't make unnecessarily large**

Transaction Logging

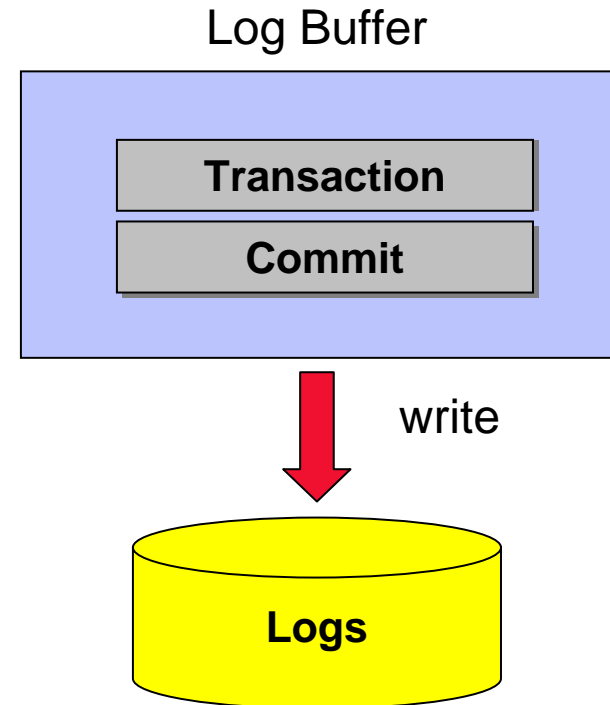
The process of recording changes to database objects and data



When are Log Records written to disk?

Transaction log records are written from log buffer to log files

- **Transaction committed**
 - application commits or a group of transactions commit, as defined by the mincommit configuration parameter
- **Log Buffer full**
 - once the internal log buffer becomes full log records are externalized to the log files on disk
- **LSN Gap Trigger**
 - When the amount of log space between the log record that updated the oldest page in the buffer pool and the current log position exceeds that allowed by the softmax database configuration parameter, the database is said to be experiencing an LSN gap
- **SOFTMAX reached**
 - Softmax is a DB CFG parameter which forces a write to disk when exceeded



This guarantees recoverability during crash recovery

Logging Bottlenecks – High Transaction Rate

- **What is a High Transaction Rate?**
 - iostat (or perfmon) shows log device performing greater than 80-100 I/O requests per second and average size ~4 KB
- **Possible Remedy:**
 - Can you reduce commit frequency?
 - Database snapshot to verify commits are high
 - Application snapshot to find out who is committing so frequently
 - Increase log buffer size
 - May be under-sized
 - # times log buffer filled, etc.
 - Possibly increase MINCOMMIT
 - Effective only when many applications (hundreds!) are all committing frequently

DB2 Maintenance Utilities and Performance



DB2 Maintenance Utilities and Performance

Version 3_2, 02/11/2010

*Data Management Emerging Partnerships and Technologies
IBM Toronto Lab*

Agenda

- **Features that impact utility performance**
- **Maintenance commands, utilities, and stored procedures**
 - Working with data at rest
 - Working with online, active data
- **Automating maintenance**
- **Workload Management**
 - A brief overview

Acknowledgement: The material for this presentation comes directly from the DB2 9.7 InfoCenter and DB2 licenses. It may or may not have been modified. Refer to <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp>

Logical and physical data organization impacts performance.

■ Database partitioning

- Split database into multiple logical/physical partitions: DPF

■ Range partitioning

- Split table into multiple table partitions within a database partition.

■ Compression

- Relational and XML data can be compressed to save storage space and log space
- Index Compression (NEW in DB2 9.7)
- Temporary table compression (NEW in DB2 9.7)

■ Related utilities:

- REORGCHK
- REORG
- RUNSTATS

Disk I/O performance impacts applications and utilities.

- **Prefetching is used to read data from disk to the bufferpools.**
- **I/O servers are the processes/threads that read the data.**
- **Database configuration parameters**
 - NUM_IOSERVERS
- **Related DB2 registry variable**
 - DB2_AVOID_PREFETCH
- **Related operations**
 - ALTER TABLESPACE
- **Related utilities**

All utilities benefit from better disk I/O performance. These utilities in particular should be reviewed for their addition

 - workload management (service classes)
 - REORGCHK, REORG, RUNSTATS

Parallel I/O can be defined for hardware configuration and runtime utility behavior.

- **Set these parameters to AUTOMATIC**
 - Database manager parameter (optional): DFT_PREFETCH_SZ
 - Tablespace parameter: PREFETCHSIZE
- **State the exact number of drives for one or more tablespace's underlying disk arrays with the DB2 registry variable DB2_PARALLEL_IO**
 - Example, all table space containers reside on a RAID-5 array of 4 disks (1 drive is for parity):
`db2set "DB2_PARALLEL_IO=* : 3"`
where * = all tablespaces and 3 is the # of drives data resides on.
- **Related utilities:**
 - **Data movement utilities:**
 - Automatically take advantage of parallel I/O.
 - **Backup and Restore commands:**
 - Also automatically take advantage of parallel operations, but can be explicitly set with the PARALLELISM command option.

Data Server memory allocation impacts performance.

- **DB2's Self Tuning Memory Management (STMM) handles runtime memory allocation and management.**
 - No database administrator intervention required.
 - Adapts to changing workload and environment
 - Automatically enabled for newly created databases since 9.5
 - INSTANCE_MEMORY is a combination of DATABASE_MEMORY and APPL_MEMORY.

- **Related utilities**

All utilities benefit from STMM.

Utility throttling provides a method to control the resource allocation of certain utilities and operations.

- **Database manager configuration parameter:**
 - UTIL_IMPACT_LIM
- **Related utilities:** [BACKUP](#), [RUNSTATS](#)
 - Parameter: UTIL_IMPACT_PRIORITY
- **Related operations:**
 - rebalancing operations (ALTER TABLESPACE command)
 - asynchronous index cleanup (AIC; example: ALTER TABLE command)
- **Related commands:**
 - [LIST UTILITIES](#)
 - [SET UTIL_IMPACT_PRIORITY](#)

DB2's Workload Management is provided by the Performance Optimization Feature.

■ Available for

- DB2 Enterprise Server Edition Version 9.7
- IBM InfoSphere Warehouse Enterprise Base Edition Version 9.7
- IBM Base Warehouse Feature for DB2 Version 9.7

■ Components

- DB2 Performance Expert for Multiplatforms
- DB2 Query Patroller
- DB2 Workload Management

■ Benefits

- WLM can be used to control resources used by the **LOAD** utility and agents used for other operations.

DB2's Deep Compression is provided by the Storage Optimization Feature.

- **Available for**
 - DB2 Enterprise Server Edition Version 9.7
 - IBM InfoSphere Warehouse Enterprise Base Edition Version 9.7
 - IBM Base Warehouse Feature for DB2 Version 9.7
- **Components**
 - Row Level Compression
 - Table Compression
 - Index Compression
- **Benefits**
 - Data is compressed on disk and in memory.
 - Compressed data means fewer I/Os to read/write.
 - More data in memory means better bufferpool hit ratios
- **Related utilities**
 - [RUNSTATS](#), [BACKUP](#), [RESTORE](#)

Agenda

- **Features that impact utility performance**
- **Maintenance commands, utilities, and stored procedures**
 - **Working with data at rest**
 - Working with online, active data
- **Automating maintenance**
- **Workload Management**
 - A brief overview

DB2 provides commands and utilities to move data into and out of a database.

- **Commands and utilities**

- **DB2MOVE** : encapsulates EXPORT, IMPORT and LOAD
- **EXPORT** : moves data out of database into flat files
- **IMPORT** : inserts data into database from flat files
- **LOAD** : fast method to load large amounts of data
- **LOAD QUERY** : check state/phase of LOAD operation or state of a table

- **SQL statements**

- **SET INTEGRITY** : may be required after a LOAD operation

- **Stored Procedures**

- **ADMIN_MOVE_TABLE** : move a table, online or offline

The EXPORT command is used to extract data from a database.

- **Cross-platform compatible**
- **An embedded SQL application executing SQL fetches**
- **Does not move aliases, views, triggers, user defined types or user defined functions.**
- **Can store data in ASCII or PC/IXF format. (WSF is deprecated.)**
- **Optimization techniques**
 - Create indexes for data ([Design Advisor](#) can help with this.)
 - Use large buffer pools ([STMM](#) can help with this.)
 - Allocate enough memory for sort heaps ([STMM](#) can help with this.)
 - Minimize device contention on/for output files by placing them away from the containers and log devices.
- **Optimization parameters**
 - None. DB2 automatically calculates bufferpools, bufferpool sizes, etc.

The **IMPORT** command is used to **INSERT** data into a database.

- **Cross-platform compatible**
- **Can read data in ASCII or PC/IXF format. (WSF is deprecated.)**
- **Can insert data into a table, hierarchy, view or nickname.**
- **An alternative to the LOAD command if the target table**
 - is a view
 - has constraints and you don't want the table put in the “set integrity pending” state
 - has triggers and you want them fired
- **Optimization techniques**
 - Minimize device contention on/for input files by placing them away from the containers and log devices.
- **Optimization parameters**
 - ALLOW NO|WRITE ACCESS
 - COMMITCOUNT *n* | AUTOMATIC

Use the LOAD command if you need a faster option than the IMPORT command.

- **Cross-platform compatible**
- **Can read data in ASCII or PC/IXF format or from a CURSOR, from either a local or remote system.**
- **Restrictions:**
 - Does not support loading data at the hierarchy level.
 - Is not compatible with range-clustered tables.
- **Optimization techniques**
 - Minimize device contention on/for input files by placing them away from the containers and log devices.
 - Use hardware and network performance tuning techniques when not loading from a local system or disk.
 - Customize optimization parameters ([see next slide](#)).

DB2 automatically calculates values for the LOAD command parameters if you do not specify a value.

▪ LOAD command optimization parameters

- savecount *n* : useful for LOAD QUERY command
- statistics *no* | *use profile* : relates to RUNSTATS (discussed later)
- data buffer *buffer-size* : number of 4KB pages used for data transfer within the utility
- sort buffer *buffer-size* : override sortheap db cfg parameter
- cpu_parallelism *n* : number of processes or threads used to build table objects
- disk_parallelism *n* : number of processes or threads used to write to tablespace containers
- fetch_parallelism *no* | *yes* : parallelize fetching from remote cursor
- sourceuserexit ... parallelize : invoke multiple user exit processes simultaneously in multipartition databases
- open *num-sess* sessions : number of I/O sessions to be used with TSM or other vendor product.

The DB2MOVE command simplifies moving large numbers of tables.

- **Cross-platform compatible**
- **Exports tables in PC/IXF format.**
- **Can duplicate a schema**
- **Does not move tables with structured type columns**
- **Optimization techniques**
 - Use the same techniques you would use for EXPORT, IMPORT or LOAD.
 - For IMPORT or LOAD actions:
 - alter the default buffer pool, IBMDEFAULTBP
 - update the database configuration parameters
 - sortheap
 - util_heap_sz
 - logfilsiz
 - logprimary

DB2 data recovery utilities can be integrated with 3rd party software and specific hardware technologies.

- **BACKUP**

- To disk
- To 3rd party software (Tivoli Storage Manager, etc.)
- Full database backup or table space backups
- Incremental backups

- **Snapshot backup (DB2 Advanced Copy Services)**

- Used to take fast snapshots of database that can be used as:
 - Hot standby
 - Mirror copy to be used as Backup image
 - Clone of primary database to be used for read activity

- **RECOVER DATABASE – performs following actions under the cover:**

- RESTORE DATABASE
- ROLLFORWARD DATABASE

- **Crash Recovery**

DB2 automatically configures BACKUP performance parameters based on system resources and database design.

■ Optimization parameters

- PARALLELISM *n*
- WITH *num-buffers* BUFFERS
- BUFFER *buffer-size*
- UTIL_IMPACT_PRIORITY

Set the UTIL_IMPACT_LIM database manager configuration parameter.

Modify with the SET UTIL_IMPACT_PRIORITY command.

■ Optimization techniques

- Increase the value of the PARALLELISM so that it reflects the number of table spaces being backed up.
- Increase the backup buffer size.
- Increase the number of buffers.
- Increase the utility heap size (UTIL_HEAP_SZ).
- Specify the table space backup option on the BACKUP DATABASE command.
- Use multiple target devices.
- Do not overload the I/O device controller bandwidth.

DB2 automatically configures RESTORE performance parameters based on system resources and database design.

- **Optimization parameters**

- PARALLELISM *n*
- WITH *num-buffers* BUFFERS
- BUFFER *buffer-size*

- **Optimization techniques**

- Increase the value of the PARALLELISM parameter.
- Increase the restore buffer size.
- Increase the number of buffers.
- Increase the utility heap size (UTIL_HEAP_SZ).
- For tables with large amount of Long field and LOB, store the LOBs in a separate table space
- Use NOT LOGGED option for the LOB data where possible.
- Use multiple target devices.
- Do not overload the I/O device controller bandwidth.

The RECOVER DATABASE command does not provide any parameters to modify its performance.

- **Optimization parameters**

- None. DB2 will try to optimize the RESTORE operation (see previous slide.)

- **Optimization techniques**

- Increase the utility heap size (UTIL_HEAP_SZ).
- Place the logs on a separate device.
- Use multiple source devices.
- Use multiple target devices.
- Do not overload the I/O device controller bandwidth.
- Restore selected table spaces if they contain large amounts of long field and LOB data.

Table data can be moved online or offline using the ADMIN_MOVE_TABLE procedure.

■ Optimization parameters

- COPY_USE_LOAD
- COPY_WITH_INDEXES
- REORG
- NO_STATS
- COPY_STATS

■ Optimization techniques

- Run this procedure when table activity is low.
- Avoid mass data loads or deletes so that parallel read access is not a problem.
- Use a multi-step move operation. The INIT and COPY phases can be called at any time. Execute the REPLAY phase multiple times in order to keep the staging table size small, and then issue the SWAP during a time of low activity on the table.
- Check if offline methods are a better choice for your table move, especially when considering tables without unique indexes and for tables with no index.

The ADMIN_MOVE_TABLE_UTIL procedure modifies the performance characteristics of the ADMIN_MOVE_TABLE procedure.

- **ADMIN_MOVE_TABLE_UTIL** procedure “key” parameter values:
 - COMMIT_AFTER_N_ROWS
 - DEEPCOMPRESSSION_SAMPLE
 - COPY_ARRAY_SIZE
 - REORG_USE_TEMPSPACE

Agenda

- Features that impact utility performance
- **Maintenance commands, utilities, and stored procedures**
 - Working with data at rest
 - **Working with online, active data**
- Automating maintenance
- Workload Management
 - A brief overview

DB2 provides several utilities that can be used to optimize online, active data.

- **Optimizing data**
 - REORGCHK
 - REORG
 - RUNSTATS

The REORGCHK command is used to identify tables and indexes that could benefit from a REORG.

Evaluation of tables and indexes can be based on the current statistics or the statistics can be updated prior to the check.

The tables and/or indexes which have been identified as needing a **REORG will show one or more asterisks in the reorg columns of the REORGCHK output.**

- **Optimization parameters:**
 - None.
- **Optimization techniques:**
 - None.

An excerpt of REORGCHK output is shown below.

Table statistics:

F1: ... < 5
 F2: ... > 70
 F3: ... > 80

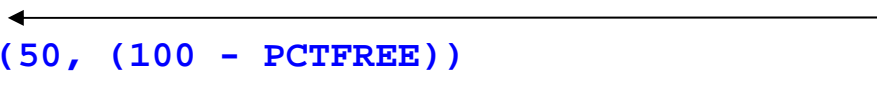
SCHEMA.NAME	...	F1	F2	F3	REORG
Table: DB2INST1.XMLFILES	...	0	89	98	---

Index statistics:

F4: ... > 80
 F5: ... > MIN(50, (100 - PCTFREE))
 F6: ... < 100
 F7: ... < 20
 F8: ... < 20

Does not meet evaluation criteria and would benefit from a REORG.

SCHEMA.NAME	...	PCT_PAGES_SAVED	F4	F5	F6	F7	F8	REORG
Table: DB2INST1.XMLFILES	...							
Index: DB2INST1.IDX1	...	50	98	82	-	0	0	-----
Index: DB2INST1.IDX2	...	0	71	-	-	0	0	*-----
Index: DB2INST1.IDX3	...	33	98	220	-	0	0	-----



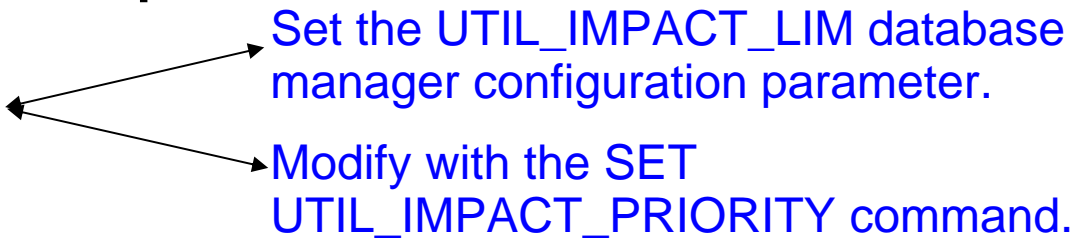
The REORG command should be run after there have been significant changes to data or indexes are added.

- **Eliminates overflow rows, reclaims space from deleted data, and places data into physically contiguous pages**
- **Can order the data in a table in physical sequence according to a specific index.**

- **Optimization sub-parameters**
 - ALLOW READ|WRITE|NO ACCESS
The default value depends on which parameter is being used.

- **Optimization techniques**
 - REORG only the components identified by the REORGCHK command.

RUNSTATS is a utility that can be throttled.

- **Collects statistics on tables, indexes and statistical views for the optimizer to use when creating an access plan.**
 - **Can register and use a statistics profile.**
 - **Optimization parameters**
 - UTIL_IMPACT_PRIORITY
 - **Optimization techniques**
 - Limit statistics collection to the set of columns used in predicates.
 - Limit statistics collection to “KEY” columns.
 - Use the TABLESAMPLE option to collect statistics on a subset of the table data.
 - Use the SAMPLED option if there are many indexes on the table and DETAILED (extended) information on the indexes might improve access plans.
 - Do not specify high NUM_FREQVALUES or NUM_QUANTILES values for columns that are not used in predicates.
- Set the UTIL_IMPACT_LIM database manager configuration parameter.
- Modify with the SET UTIL_IMPACT_PRIORITY command.
- 

Agenda

- **Features that impact utility performance**
- **Maintenance commands, utilities, and stored procedures**
- **Maintenance commands, utilities, and stored procedures**
 - Working with data at rest
 - Working with online, active data
- **Automating maintenance**
- **Workload Management**
 - A brief overview

Automatic maintenance is enabled for a database when it is created.

Database Configuration for Database

...

Automatic maintenance	(AUTO_MAINT) = ON
Automatic database backup	(AUTO_DB_BACKUP) = OFF
Automatic table maintenance	(AUTO_TBL_MAINT) = ON
Automatic runstats	(AUTO_RUNSTATS) = ON
Automatic statement statistics	(AUTO_STMT_STATS) = ON
Automatic statistics profiling	(AUTO_STATS_PROF) = OFF
Automatic profile updates	(AUTO_PROF_UPD) = OFF
Automatic reorganization	(AUTO_REORG) = OFF

...

You can define automatic maintenance windows and customize the maintenance utilities.

- **Sample policies are in `~/sqllib/samples/automaintcfg`**
 - DB2MaintenanceWindowPolicySample.xml
 - DB2AutoRunstatsPolicySample.xml
 - DB2AutoReorgPolicySample.xml
 - DB2AutoBackupPolicySample.xml
- **Define windows for both online and offline maintenance.**
- **Automatic maintenance runs only if the requirements in the policy are met.**
- **Maintenance utilities will run to completion if they are started.**
- **Optimization parameters**
 - None.

Automatic statistics can be collected during maintenance windows or real-time.

- **Optimization parameters:**

- None.

- **Procedure:**

- Set the `auto_maint` and the `auto_tbl_maint` database configuration parameters to ON. This is the default.
- To enable real-time statistics collection, set both `auto_stmt_stats` and `auto_runstats` database configuration parameters to ON. This is the default.
- To enable background statistics collection, set the `auto_runstats` database configuration parameter to ON. This is the default.
- To enable automatic statistics profile generation, set both `auto_stats_prof` and `auto_prof_upd` database configuration parameters to ON. If the `auto_runstats` database configuration parameter is also set to ON, statistics are collected automatically using the generated profiles. **This should not be used in production environments unless performance problems point to bad statistics.**

Agenda

- **Features that impact utility performance**
- **Maintenance commands, utilities, and stored procedures**
- **Maintenance commands, utilities, and stored procedures**
 - Working with data at rest
 - Working with online, active data
- **Automating maintenance**
- **Workload Management**
 - A brief overview

DB2's workload management feature provides a way to monitor and manage certain database resources.

- **Query patroller and DB2 Governor are deprecated.**
 - WLM is the strategic future direction.
- **Provides light-weight, granular way to monitor active work.**
- **Better resource management:**
 - Explicitly allocate resources
 - Limit excessive, unexpected resource consumption
 - Can explicitly manage the **LOAD** utility
- **Better request management**
 - **Manage work based on business priority**
 - Track work performance
- **WLM is not a performance tuning tool.**

END OF PRESENTATION

Questions?

