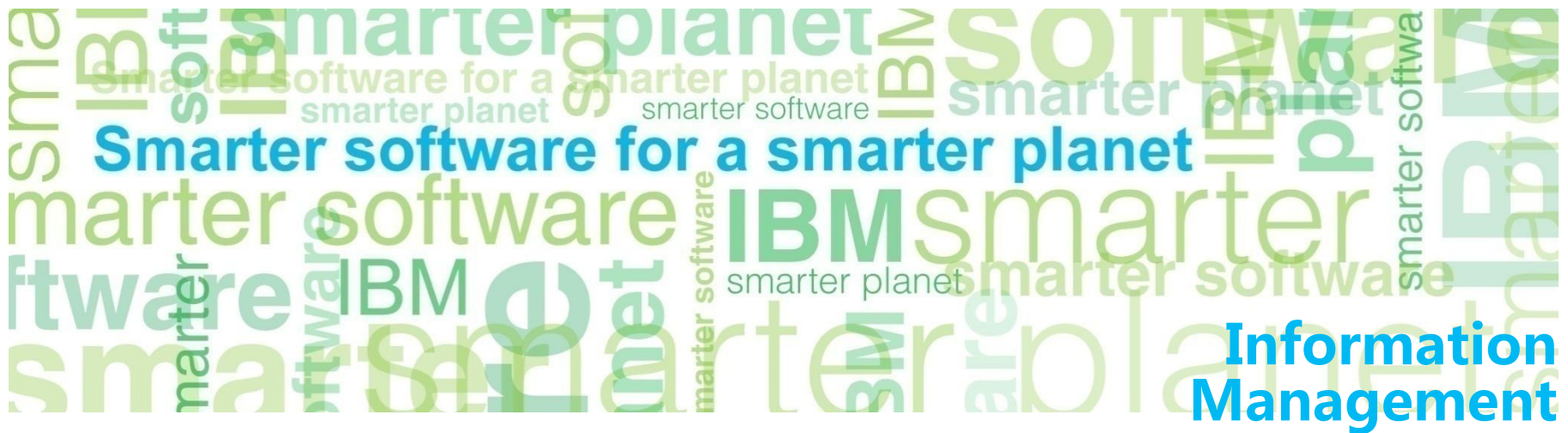


IBM DB2 监控与调优



尤祖喜

Information Management Partner Ecosystem

youzuxi@cn.ibm.com

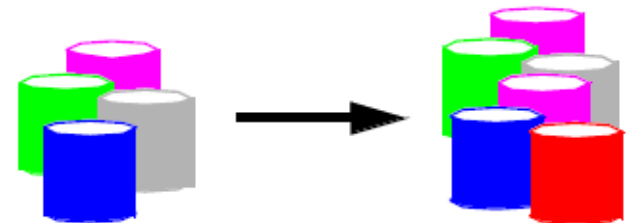
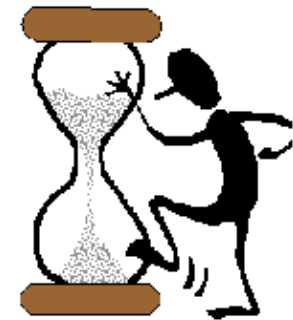
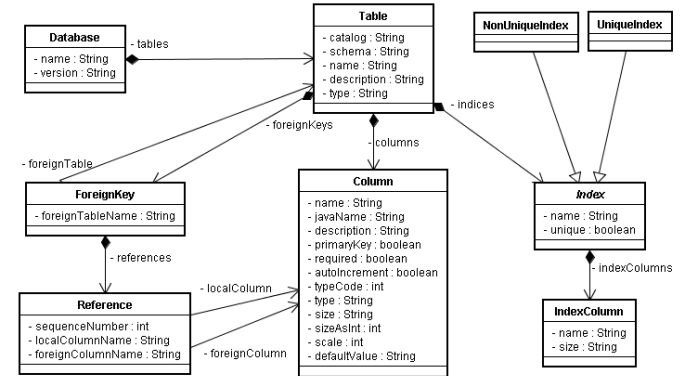
- 数据库性能问题原因
- DB2数据库监控手段
 - 事件监控
 - 监控快照
 - 监控函数与视图
- DB2 SQL监控与调优
- DB2 Lock监控与调控机制
- DB2 Log监控与调控机制
- DB2 监控常用工具

性能调试从来都是由一个问题开始的

- 数据库或者用户的设计问题:
 - *How should we design our application/database for the best performance?*

- 用户的抱怨:
 - *This transaction is taking too long*

- 系统的扩容:
 - *Can the current system handle a 50% increase in the size of our database?*



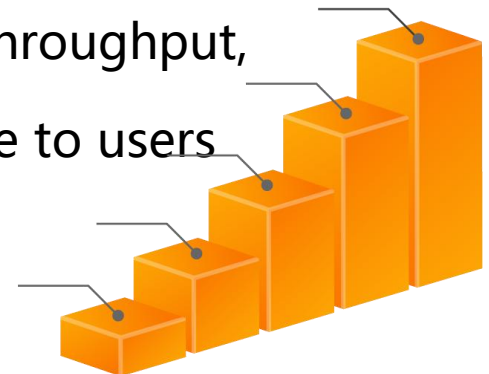
数据库性能调优是一个需要望闻问切、详细诊断的过程

- Performance problems are trickier than functional problems
 - Symptoms may provide no clues about the problem source, e.g. you observe general slowdown and excessive lock timeouts
 - 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

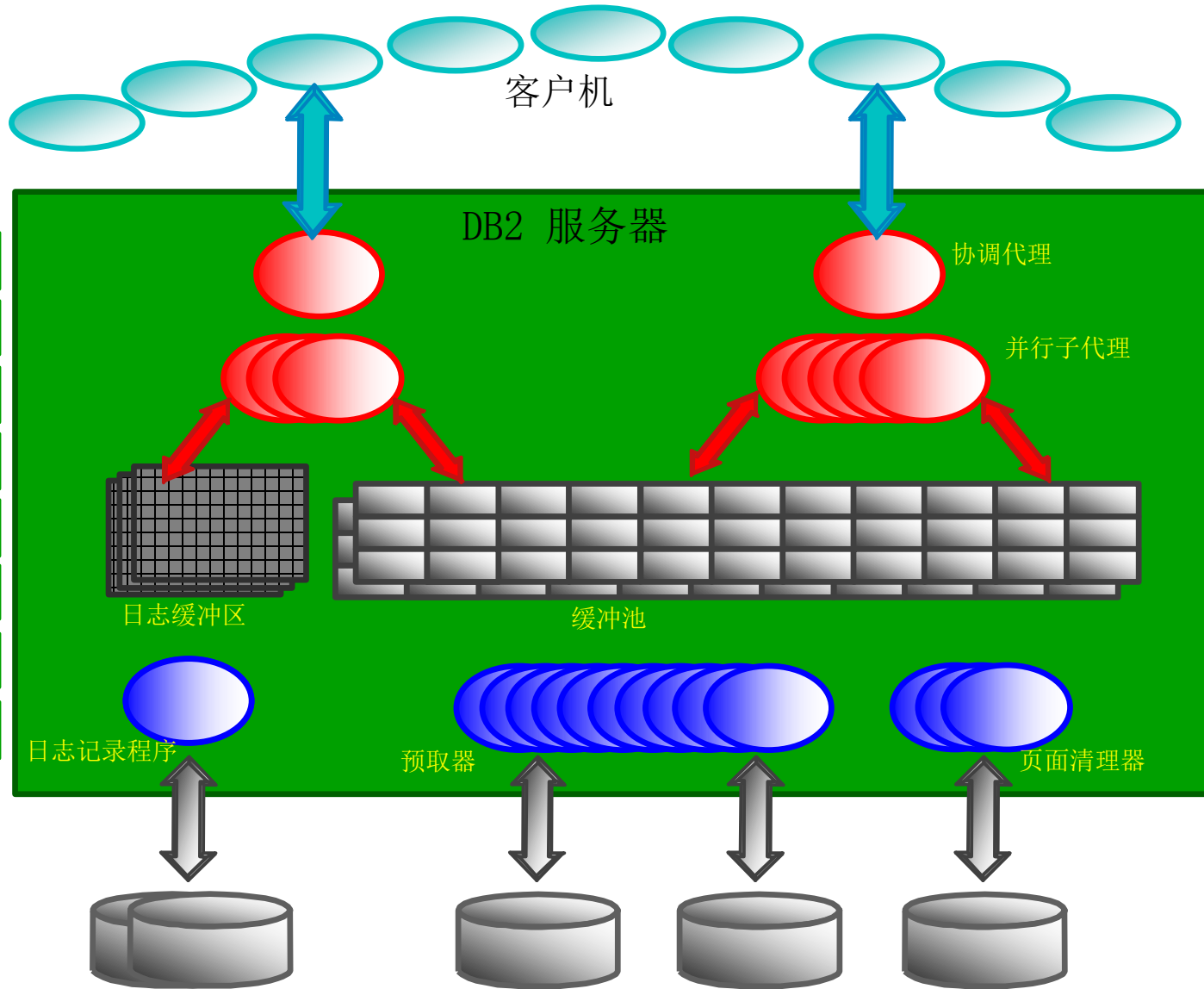


什么是性能问题？

- 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



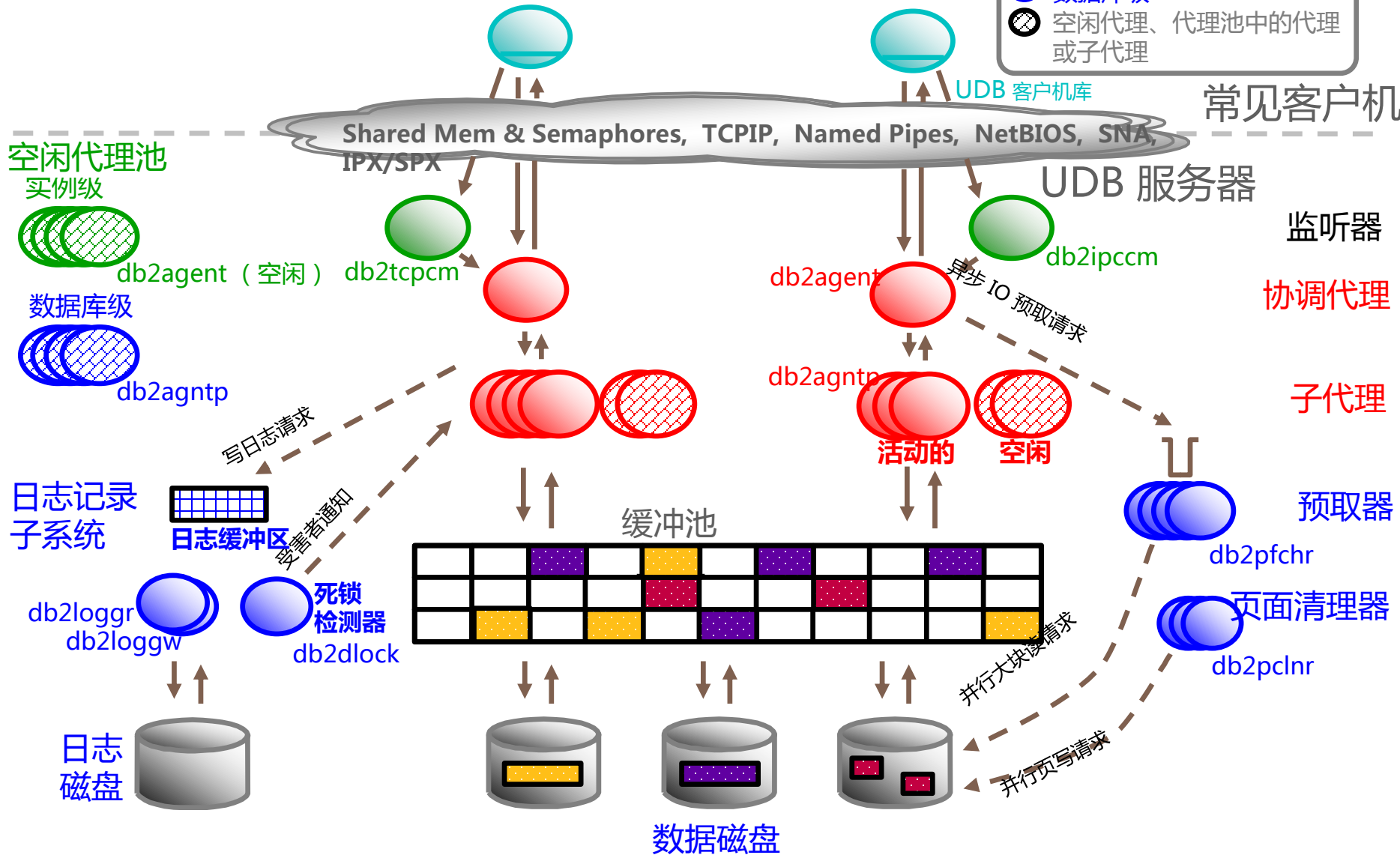
DB2 架构概览



DB2进程模型：详细视图

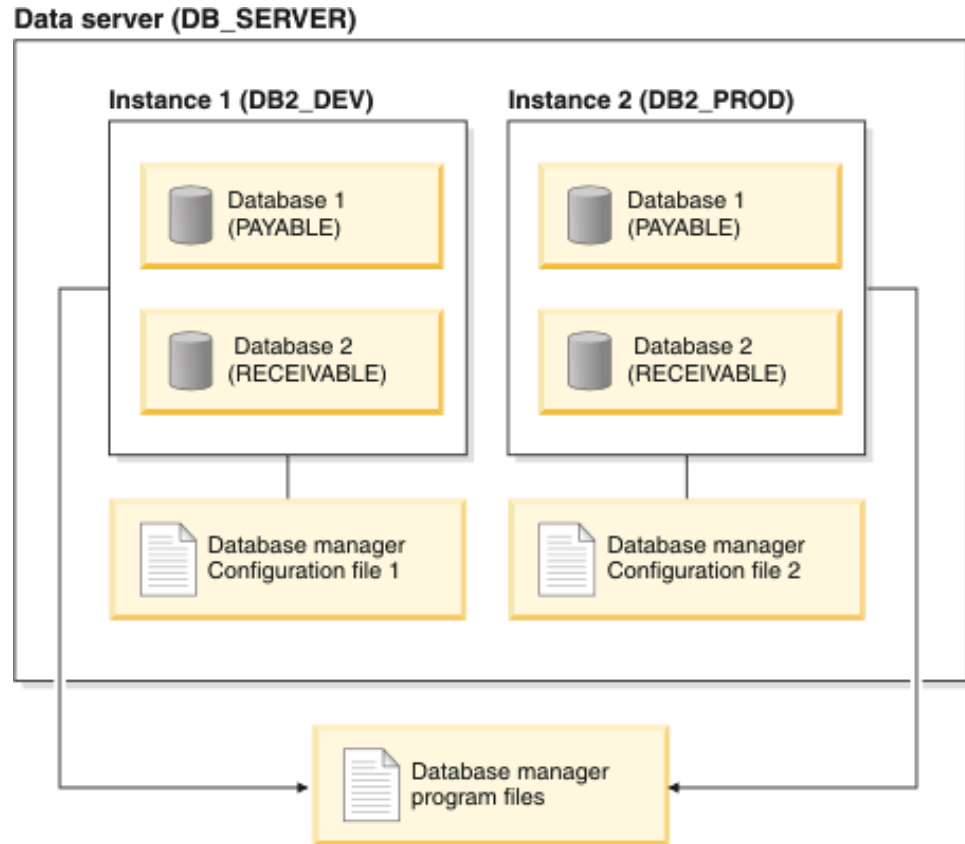
进程/线程组织

- 实例级 (Green circle)
- 应用程序级 (Red circle)
- 数据库级 (Blue circle)
- 空闲代理、代理池中的代理或子代理 (Hatched circle)



DB2 数据库实例

- 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



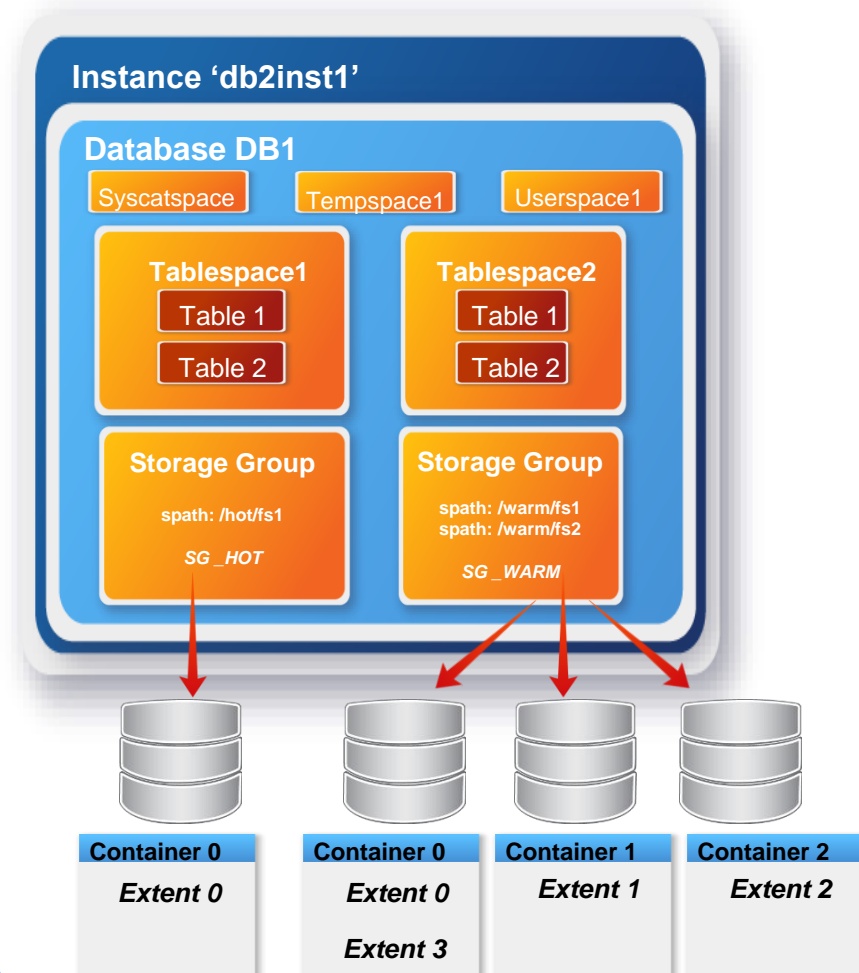
■ **Page**

– Smallest unit of storage in DB2

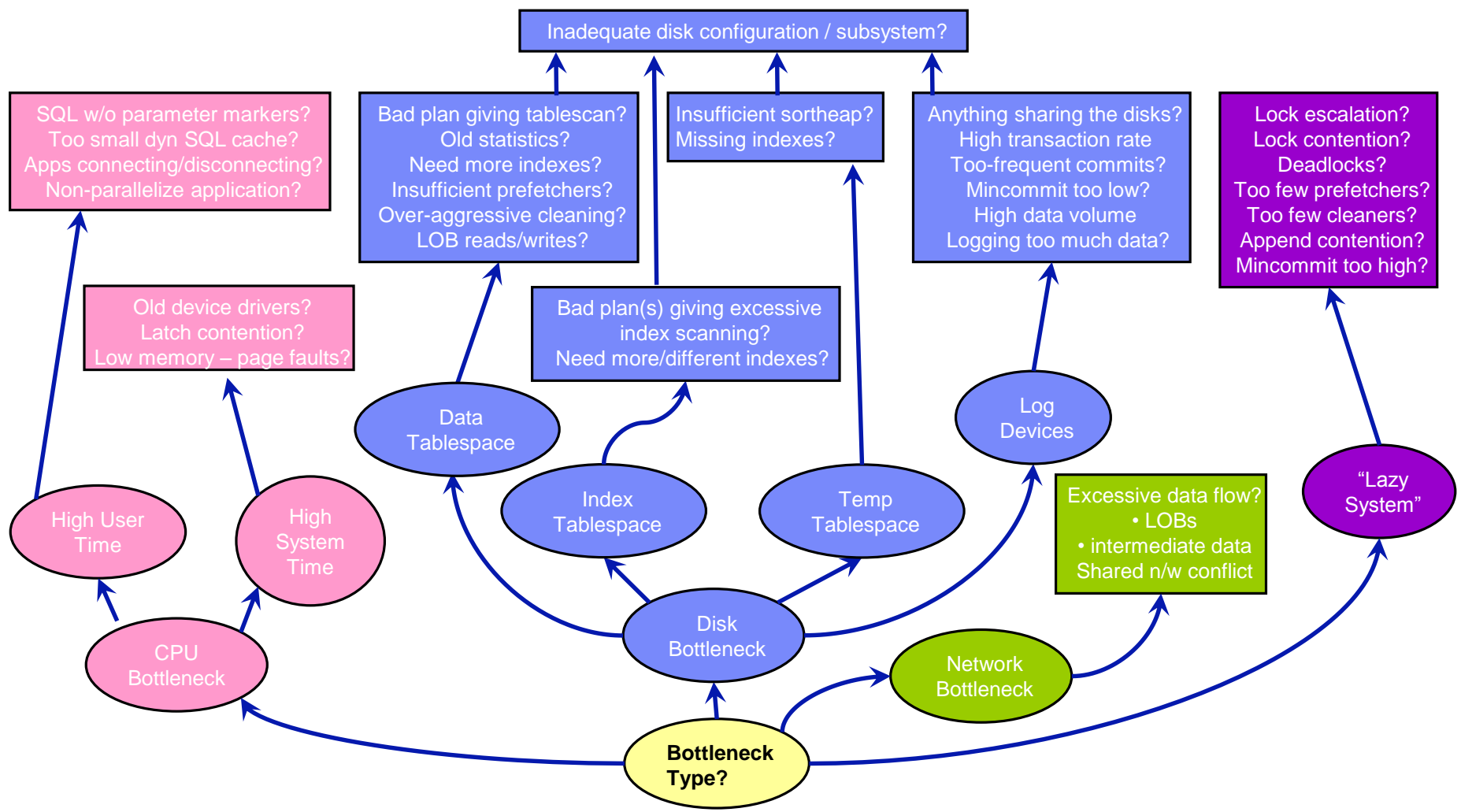


■ **Storage Group**

– New layer of abstraction between logical (table spaces) and physical storage (containers)

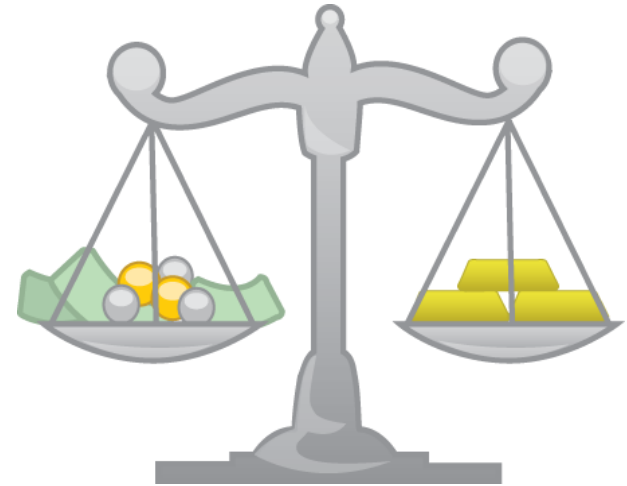


数据库的性能问题可能是多方面的



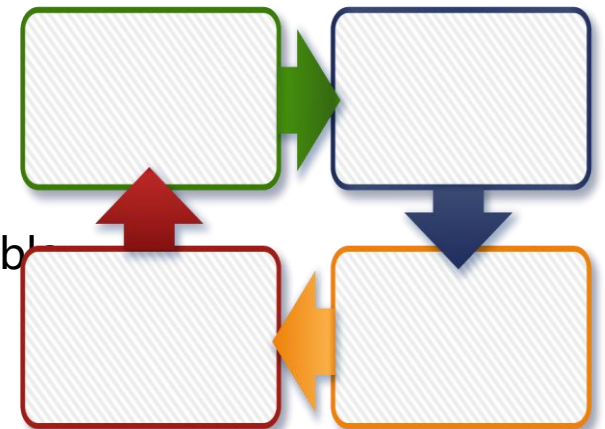
性能调试的限制

- 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



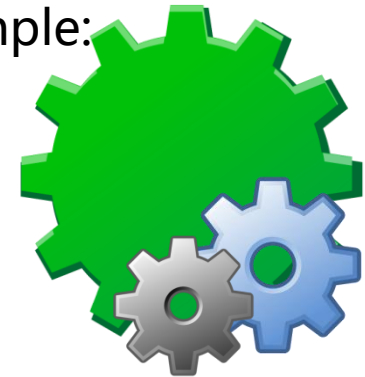
性能调试的方式与基线

- 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



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 results, for example:
 - 8 core IBM POWER7 4.14GHz = ~1.2M TPM
 - 64 core IBM POWER6 5 GHz = ~6M TPM
 - 192 core IBM POWER7 7 3.86 GHz = ~10M TPM
 - 10 core Intel Xeon 2.4 GHz = ~3M TPM
 - 32 core Intel Xeon 2.26 GHz = ~2.3M TPM



Storage

■ Considerations:

- I/O Throughput
 - I/Os per Second (IOPS)
 - Megabytes per Second (MBPS)
- Storage Capacity
- Separate dedicated (unshared) disks for logging



■ Rules of Thumb:

- 15K RPM Fibre Channel disk = ~200 IOPS @ ~10 milliseconds response time
- Solid State Drives (SSDs) I/O service times are typically less than a millisecond, instead of up to approximately 10 ms, for typical small random reads from physical disks

■ Because CPU processing speeds have increased substantially relative to spindle speeds:

- For OLTP, ensure that there are 15 - 20 dedicated physical disks per CPU core.
- For warehousing, ensure that there are 8 - 10 disks per CPU core

■ Logging:

- For OLTP, fast log response time is often more important than I/O service times for data, which is frequently asynchronous

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 POWER (System p)

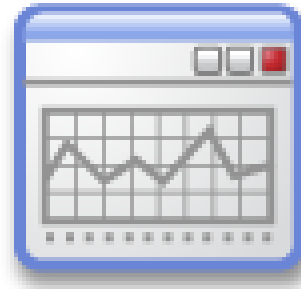


议程

- 数据库性能问题原因
- DB2数据库监控手段
 - 事件监控
 - 监控快照
 - 监控函数与视图
- DB2 SQL监控与调优
- DB2 Lock监控与调控机制
- DB2 Log监控与调控机制
- DB2 监控常用工具

Introduction

- **Database monitoring**
 - Tasks associated with examining the operational status of your database
- **Database monitoring is a vital activity for:**
 - The maintenance of performance
 - Health of your database management system
- **Collects information from:**
 - Database manager
 - Its databases
 - Connected applications



Monitor Elements



▪ Data structures used to store information about a particular aspect of the database system status

- Each monitor element reflects one of the following types of data:
 - **Counter**: the number of times something happens
 - deadlocks*: the total number of deadlocks that have occurred
 - rows_deleted*: the number of row deletions attempted
 - total_sorts*: the total number of sorts that have been executed
 - **Gauge**: a measurement of how much of something is happening or is used
 - total_section_proc_time* or *total_sort_time*: measures of how much time is used in different phases of processing
 - **Watermark**: the highest value reached for a given measurement
 - uow_total_time_top*: the lifetime of the longest-running unit of work since the database was activated
 - **Text**: many monitor elements report text values
 - stmt_text*: the text of an SQL statement
 - **Timestamp**: the time that something happened
 - conn_time*: the time that a connection was made to a database

Monitor Elements - Categories

- **Request monitor elements** (also known as *request metrics*)
 - Measure the volume of work or effort expended by the database server to process requests issued directly by an external application (*application requests*), by a coordinator agent to a subagent or by an agent at a different database member
 - **Overall system processing**
 - total_cpu_time*
 - total_wait_time*
 - total_rqst_time*
 - rqsts_completed_total*
 - **Client-server processing**
 - client_idle_wait_time*
 - tcpip_recv_volume*
 - **Data Server processing**
 - lock_wait_time*
 - pool_read_time*
 - **Specific Data Server environment:**
 - fcm_recv_wait_time*
 - wlm_queue_time_total*
- Available through Table Functions and Event Monitors

Monitor Elements - Categories

- **Activity monitor elements** (also known as *activity metrics*)

 - Subset of request monitor elements:
 - Monitor the work done to execute SQL statement sections, including locking, sorting, and row processing
 - direct_read_time*
 - effective_isolation*
 - STMT_TEXT*
 - Available through Table Functions and Event Monitors

- **Data Object monitor elements**

 - Provide information about operations performed on particular data objects, including tables, indexes, buffer pools, table spaces and containers
 - TABLE_SCANS*
 - ROWS_INSERTED*
 - INDEX_SCANS*
 - Available through Table Functions



Monitor Elements – Collection Levels

- **REQUEST METRICS {NONE | BASE | EXTENDED}**
 - Broadest (highest) collection level specified by:
 - Database configuration parameter: MON_REQ_METRICS
 - CREATE/ALTER SERVICE CLASS... COLLECT REQUEST METRICS
- **ACTIVITY METRICS {NONE | BASE | EXTENDED}**
 - Broadest (highest) collection level specified by:
 - Database configuration parameter: MON_ACT_METRICS
 - CREATE/ALTER WORKLAD... COLLECT ACTIVITY METRICS
- **DATA OBJECT METRICS {NONE | BASE | EXTENDED}**
 - Database configuration parameter: MON_OBJ_METRICS
- **Always collected**

Monitor Elements – DB2 10 Enhancements



NEW IN
DB2 10

▪ Some of the new monitor elements:

- `evmon_wait_time` The amount of time that an agent waited for an event monitor record to become available
- `total_extended_latch_wait_time` The amount of time, in milliseconds, spent in extended latch waits
- `total_extended_latch_waits` The number of extended latch waits
- `intra_parallel_state` The current state of intrapartition parallelism reported at statement, activity, transaction, or workload level
- `total_stats_fabrication_time` Is the statistics collection activity needed to generate statistics during query compilation
- `total_stats_fabrication_proc_time` The total non-wait time spent on statistics fabrications by real-time statistics gathering
- `total_sync_runstats_time` The total time spent on synchronous RUNSTATS activities triggered by real-time statistics gathering
- `total_disp_run_queue_time` The total time that requests, that were run in this service class, spent waiting to access the CPU

Monitoring Framework - Three Focus Areas

- **System**

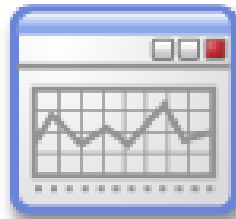
- Provide total perspective of application work being done by database system
- Aggregated through the WLM infrastructure

- **Activity**

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

- **Data objects**

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



Performance Monitoring Methodology

- **Operational monitoring strategy**
 - Needs to be very light weight
 - Analysis and comparison of monitoring data
 - Do not limit yourself to just metrics that the DB2 product provides
- **Types of data are useful to collect**
 - A basic set of DB2 system performance monitoring metrics
 - DB2 configuration information
 - Overall system load
 - Throughput and response time measured at the business logic level



Performance Monitoring Methodology - Continued



▪ **Basic set of system performance monitor elements**

- The number of transactions executed
- Analysis and comparison of monitoring data
- Buffer pool hit ratios, measured separately for data, index, XML storage object, and temporary data
- Buffer pool physical reads and writes per transaction
- The ratio of database rows read to rows selected
- The amount of time spent sorting per transaction
- The amount of lock wait time accumulated per thousand transactions
- The number of deadlocks and lock timeouts per thousand transactions
- The number of dirty steal triggers per thousand transactions
- The number of package cache inserts per thousand transactions
- The time an agent waits for log records to be flushed to disk
- In partitioned database environments, the number of fast communication manager (FCM) buffers sent and received between partitions

Essential Monitoring Targets

▪ Track key performance indicators

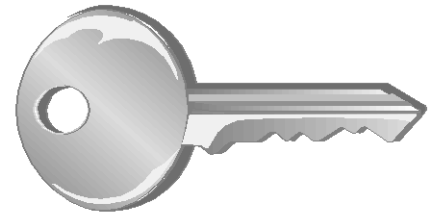
- Practical approach, too many = diminished returns
- Key DB2 monitoring elements as indicators
 - Stand alone elements
 - Calculate ratios
- DB2 monitoring elements can be captured from:
 - Snapshot monitoring
 - Switch based
 - Some CPU overhead (1-10%)
 - Easy to reset counters
 - Monitor table functions 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

- **Total number of transactions executed by applications:**

`COMMIT_SQL_STMTS + ROLLBACK_SQL_STMTS`

- Snapshot monitoring: database or application level
- Event Monitor: database or connection level

- **Total number of units of work:**

`COMMIT_SQL_STMTS + INT_COMMITS +
ROLLBACK_SQL_STMTS + INT_ROLLBACKS`

- **Total number of commit and rollback statements issued by the client application**

- System Monitor Table Functions or Event Monitors:

`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**, measured separately for **DATA**, **INDEX** and **XDA**
 - **MON_BP_UTILIZATION** Administrative View
- **For each Bufferpool:**
 - **DATA_HIT_RATIO_PERCENT**
 - Percentage of time that the database manager did not need to load a page from disk to service a data page request
 - **INDEX_HIT_RATIO_PERCENT**
 - Percentage of time that the database manager did not need to load a page from disk to service an index data page request
 - **XDA_HIT_RATIO_PERCENT**
 - Auxiliary storage objects hit ratio, that is, the percentage of time that the database manager did not need to load a page from disk to service a data page request for XML storage objects (XDAs)

```
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 globally and separately for DATA, INDEX and XDA**

- **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
      , data_hit_ratio_percent
      , index_hit_ratio_percent
      , xda_hit_ratio_percent
      , dbpartitionnum
FROM SYSIBMADM.BP_HITRATIO
ORDER BY dbpartitionnum
```

OLTP

GOOD HIT RATIO:

Data: > 80-85%

Indexes: > 90-95%

- **SNAP_GET_BP Table Function**

- Use to aggregate results from all partitions or report on single partition

- **GET SNAPSHOT FOR ALL BUFFERPOOLS Command**

Essential Monitoring Targets – I/O EFFICIENCY

▪ Number of ROWS READ PER TRANSACTION

NOTE: is not the # of rows that were returned to the calling application, but the # of rows that had to be read from the table in order to return the result set (table scan vs index access only)

- SNAPDB Administrative View
- SNAP_GET_DB Table Function
- GET_SNAPSHOT FOR DATABASE Command

Is that a lot?

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

▪ Total amount of CPU TIME

– `MON_PKG_CACHE_SUMMARY` Administrative View

– Aggregate metrics overall executions of each SQL statement (static or dynamic) in the cache:

- `TOTAL_CPU_TIME`

Total amount of CPU time, in microseconds, used while within the DB2® database manager (combined total of both user and system CPU time)

- `TOTAL_LOCK_WAIT_TIME`

Total elapsed time, in milliseconds, spent waiting for locks

- `TOTAL_IO_WAIT_TIME`

The total elapsed time, in milliseconds, spent on I/O operations

```
SELECT total_cpu_time
       , total_lock_wait_time
       , total_io_wait_time
       , avg_io_wait_time
       , avg_lock_wait_time
FROM   SYSIBMADM.MON_PKG_CACHE_SUMMARY
ORDER BY total_cpu_time DESC
FETCH FIRST 20 ROWS ONLY
```



Essential Monitoring Targets - LOCK WAIT TIME



Information for each workload

– `SYSPROC.MON_GET_WORKLOAD` Table Function

```
SELECT varchar(workload_name,30) as workload_name
      , sum(lock_wait_time) as total_lock_wait_time
      , sum(lock_waits) as total_lock_waits
      , sum(lock_timeouts) as total_lock_timeouts
      , sum(lock_escals) as total_lock_escals
FROM TABLE(MON_GET_WORKLOAD('',-2)) AS t
GROUP BY workload_name
ORDER BY total_lock_wait_time DESC;
```

– `SYSPROC.MON_GET_WORKLOAD_DETAILS` Table Function

```
SELECT varchar(wlmetrics.workload_name,30) as workload_name,
      sum(detmetrics.lock_wait_time) as total_lock_wait_time,
      sum(detmetrics.lock_waits) as total_lock_waits,
      sum(detmetrics.lock_timeouts) as total_lock_timeouts,
      sum(detmetrics.lock_escals) as total_lock_escals
FROM TABLE(MON_GET_WORKLOAD_DETAILS('',-2)) AS WLMETRICS,
XMLTABLE (XMLNAMESPACES( DEFAULT 'http://www.ibm.com/xmlns/prod/db2/mon' ),
          '$detmetric/db2_workload' PASSING
          XMLPARSE(DOCUMENT WLMETRICS.DETAILS)
          as "detmetric"
COLUMNS "LOCK_WAIT_TIME" INTEGER PATH 'system_metrics/lock_wait_time',
        "LOCK_WAITS" INTEGER PATH 'system_metrics/lock_waits',
        "LOCK_TIMEOUTS" INTEGER PATH 'system_metrics/lock_timeouts',
        "LOCK_ESCALS" INTEGER PATH 'system_metrics/lock_escals'
) AS DETMETRICS
```

Essential Monitoring Targets - SORTING

▪ SORTING metrics

- `SYSPROC.MON_GET_WORKLOAD` Table Function
- `TOTAL_SECTION_SORT_TIME / (TOTAL_APP_COMMITS + TOTAL_APP_ROLLBACKS)`
- `SORT_OVERFLOWS / TOTAL_SORTS` = % of sorts that need more heap space
- `TOTAL_SECTION_SORT_TIME / TOTAL_SORTS` = average sort time

```
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 – AVERAGE LOG DISK WAIT TIME

▪ Time an agent spends waiting for log records to be flushed to disk

– **MON_GET_WORKLOAD** Table Function

– **LOG_DISK_WAIT_TIME**

- The amount of time (in milliseconds) an agent spends waiting for log records to be flushed to disk

– **LOG_DISK_WAITS_TOTAL**

- The number of times agents have to wait for log data to write to disk while copying log records into the log buffer

```
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;
```

议程

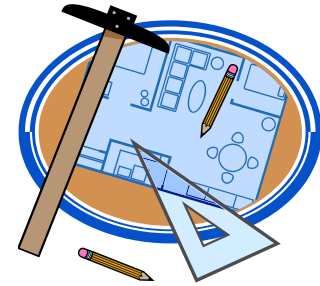
- 数据库性能问题原因
- DB2数据库监控手段
 - 事件监控
 - 快照监控
 - 监控函数与视图
- DB2 SQL监控与调优
- DB2 Lock监控与调控机制
- DB2 Log监控与调控机制
- DB2 监控常用工具

Monitoring Enhancements for Version 10.5

New Monitoring elements for new column-organized tables, for example

- Counters for total logical and physical column-organized data page reads and pages found
 - E,g, POOL_COL_L_READS, POOL_COL_P_READS
- Counter for column-organized data page writes:
 - POOL_COL_WRITES
- Counters for asynchronous column-organized data page reads and writes and pages found:
 - POOL_ASYNC_COL_READS, POOL_ASYNC_COL_WRITES
- Counters for column-organized data page reads per table

Monitoring - Interfaces



▪ **Table Functions and Monitor Views**

- Real-time monitoring accessible through SQL statements
- Newer, lightweight, high-speed monitoring infrastructure
- Evolved as complimentary extension to Workload Manager (WLM) implementation
- Turn on collection at database level (more granularity available with WLM feature)

▪ **Event Monitors**

- Capture information about database operations over time, as specific types of events that take place in your system

▪ **Snapshot Monitor**

- Switch based monitoring
- Services snapshot command, most Event Monitors, Administrative Views and Table Functions
- Snapshots are useful for determining the status of a database system

▪ **DB2 problem determination tool a.k.a db2pd**

Event Monitoring

- To capture point-in-time information related to different kinds of events that take place in the system
- Created via **SQL-DDL**, definitions are stored in system catalog tables
- Output can be directed to:
 - File
 - Table
 - Pipe
- Types of events include:
 - Locking
 - UOW
 - Statements (SQL)
 - Connections
 - Tables
- New **monitoring framework** being used for new **event monitors**
 - **Some event monitors** have been deprecated
 - **DEADLOCKS + DETAILED DEADLOCKS** → **LOCKING**
 - **TRANSACTION** → **UNIT OF WORK (UOW)**



Types Of Events For Which Event Monitors Capture Data



Type of event to monitor	Event monitor name	Details
Locks and deadlocks	LOCKING	To determine when locks or deadlocks occur, and the applications that are involved.
Execution of an SQL statement	ACTIVITIES	To capture activities for diagnostic reasons and to study the resource consumption of SQL
Execution of an SQL statement	STATEMENTS	To track what requests are being made to the database as a result of the execution of SQL statements
Completion of a unit of work (transaction)	UNIT OF WORK	To gather resource usage information and performance metrics for UOWs that run on the system
Eviction of sections from the package cache	PACKAGE CACHE	To capture a history of statements that are no longer in the package cache
Connections to the database by applications	CONNECTIONS	To capture metrics and other monitor elements for each connection to the database by an application
Deactivation of database	DATABASE	To capture metrics and other monitor elements that reflect information about the database as whole, since activation
Deactivation of database	BUFFERPOOLS	To capture metrics related to buffer pools
Deactivation of database	TABLESPACES	To capture metrics related to table spaces
Deactivation of database	TABLES	To capture metrics related to tables that have changed since database activation
Statistics and metrics on WLM objects	STATISTICS	To capture processing metrics related to WLM objects in the database
Exceeding a WLM threshold	THRESHOLD VIOLATIONS	To determine when specific thresholds that you set are exceeded during database operations
Changes to db or database manager configuration	CHANGE HISTORY	To capture change to db and db manager configuration and registry settings, execution of DDL statements and execution of utilities

Working With Event Monitors - Procedure

1) Create the Event Monitor

– (Optional) Activate the Event Monitor

2) Enable the collection of data

– Only for:

- LOCKING
- ACTIVITIES
- STATISTICS
- UNIT OF WORK

3) Run your application or queries

4) (Optional) Deactivate the Event Monitor

5) Examine the data collected by the Event Monitor

6) (Optional) Prune data that is no longer needed from the Event Monitor Tables

Working With Event Monitors – (1) Create the Event Monitor

A. Determine type of Event Monitor

B. Decide type of output from the Event Monitor



Regular Tables

- ✓ Starting in DB2 Version 10, all event monitors can write output to regular tables
- ✓ Examine monitoring data at a later point in time
- ✓ Immediate access to data using SQL
- ✗ CPU, log file, disk storage

Unformatted Event (UE) Tables

- ✓ Data written in binary format
- ✓ New PRUNE_UE_TABLE option for the procedure EVMON_FORMAT_UE_TO_TABLES, to prune data from the UE table after the extraction
- ✓ Examine monitoring data at a later point in time
- ✓ Performance, CPU, log file, disk storage
- ✗ Require a post-processing operation to extract the data and to perform query using SQL

Files

- ✓ Managed by the Operating System
- ✓ Data stored outside of the database being monitored
- ✓ Examine the data offline at later point in time
- ✗ SQL access

Named Pipes

- ✓ Output sent to a named pipe so the data can be used by another application immediately
- ✓ Event Data manipulation in real time
- ✗ Access event data at a later point in time

Working With Event Monitors – (1) Create the Event Monitor

Type of event to monitor	Regular Table	UE Table	File	Named Pipe
LOCKING	✓	✓		
ACTIVITIES	✓		✓	✓
STATEMENTS	✓		✓	✓
UNIT OF WORK	✓	✓		
PACKAGE CACHE	✓	✓		
CONNECTIONS	✓		✓	✓
DATABASE	✓		✓	✓
BUFFERPOOLS	✓		✓	✓
TABLESPACES	✓		✓	✓
TABLES	✓		✓	✓
STATISTICS	✓		✓	✓
THRESHOLD VIOLATIONS	✓		✓	✓
CHANGE HISTORY	✓			

Working With Event Monitors – (1) Create the Event Monitor

C. Issue a **CREATE EVENT MONITOR** statement

```
CREATE EVENT MONITOR evmon-name FOR eventtype
WRITE TO {TABLE | PIPE | FILE | UNFORMATTED EVENT TABLE}
{AUTOSTART | MANUALSTART}
```

- **CONNECTIONS** and **STATEMENTS** Event Monitors support the use of a **WHERE** clause on *application id*, *authorization id* and *application name*, in the **CREATE** or **ALTER EVENT MONITOR** statement
- **BUFFERPOOLS**, **CONNECTIONS**, **DATABASE**, **STATEMENTS**, **TABLES** and **TABLESPACES** Event Monitors can capture different types of events with a single event monitor definition

D. (Optional) If required by the type of event monitor created, activate it by issuing the **SET EVENT MONITOR STATE** statement

```
SET EVENT MONITOR evmon-name STATE 1
```

Working With Event Monitors – (1) Create the Event Monitor

▪ **Recommended Practice:**

- Table Space dedicated and configured to store the output table or tables associated with any event monitor
- For Unformatted Event Table: create Table Spaces with at least an 8K pagesize to ensure that event data is contained within the inlined BLOB column of the UE table. If the BLOB column is not inlined, then the performance of writing and reading the events to the unformatted event table might not be efficient

Working With Event Monitors – (2) Enable Data Collection

PASSIVE EVENT MONITORS:

■ **UNIT OF WORK**

– Database configuration parameter:

- **MON_UOW_DATA** {NONE | BASE}
- **MON_UOW_PKGLIST** {OFF | ON}
- **MON_UOW_EXECLIST** {OFF | ON}

– CREATE/ALTER WORKLOAD ... **COLLECT UNIT OF WORK DATA...**

– **N.B.:** Enable REQUEST METRICS collection

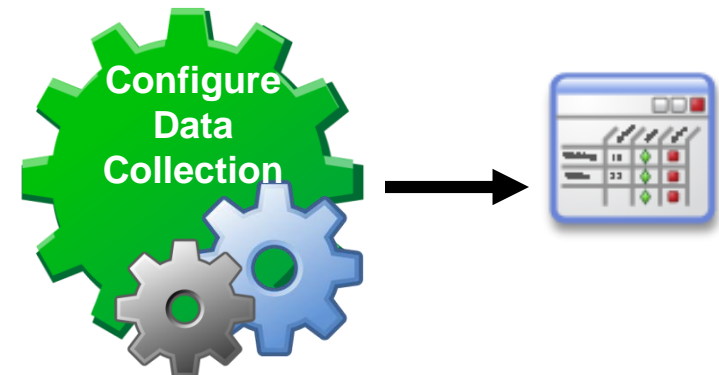
■ **LOCKING**

– Database configuration parameter:

- **MON_LOCKWAIT**
- **MON_LW_THRESH**
- **MON_LOCKTIMEOUT**
- **MON_DEADLOCK**

– CREATE/ALTER WORKLOAD ...

- **COLLECT LOCK WAIT DATA**
- **COLLECT LOCK TIMEOUT DATA**
- **COLLECT DEADLOCK DATA**



Working With Event Monitors – (2) Enable Data Collection

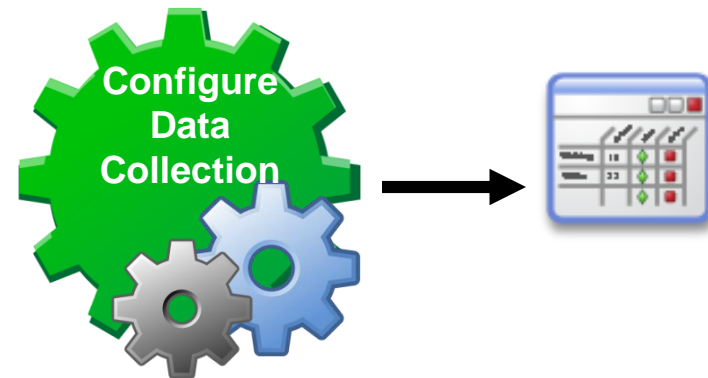
PASSIVE EVENT MONITORS:

■ **ACTIVITIES**

- CREATE THRESHOLD... COLLECT ACTIVITY DATA...
- CREATE/ALTER SERVICE CLASS...COLLECT ACTIVITY DATA...
- CREATE/ALTER WORKLOAD... COLLECT ACTIVITY DATA...

■ **STATISTICS**

- Database configuration parameter
 - Enable REQUEST METRICS collection
- CREATE/ALTER SERVICE CLASS ... COLLECT REQUESTS METRICS...



Working With Event Monitors – (4) Deactivate Event Monitor

- (Optional) Deactivate the Event Monitor by issuing the SET EVENT MONITOR STATE statement:

```
SET EVENT MONITOR evmon-name STATE 0
```

- Event Monitor Status:

```
SELECT EVMONNAME
       , EVENT_MON_STATE(EVMONNAME) STATUS
FROM SYSCAT.EVENTMONITORS;
```



Working With Event Monitors – (5) Examine the data collected by EM

▪ UNFORMATTED EVENT (UE) TABLE:

– db2evmonfmt tool

- Extracts data into a text report or into a formatted XML document
- Limited capabilities to filter data (event ID, application, workload, ...)
- Setup and compilation of the Java source code provided (sqllib/samples/java/jdbc) is required before the tool can be used
- Example:

```
java db2evmonfmt -f lock.xml -ftext -type lockwait -hours 5
```

– EVMON_FORMAT_UE_TO_XML table function

- Extracts data into an XML document
- PureXML features to query data

– EVMON_FORMAT_UE_TO_TABLES procedure

- Extracts data into a set of relational tables
- With PRUNE_UE_TABLES option, data that is successfully inserted into relational tables is deleted from the UE table



Working With Event Monitors – (5) Examine the data collected by EM

▪ **REGULAR TABLE:**

- Run a **SELECT statement** to display the monitor element data

▪ **FILE or PIPELINE**

- db2evmon command

- Formats event monitor file and named pipe output, for display using standard output
- **EXAMPLE:**

Providing the path of the event files

```
db2evmon -path '/tmp/dlevents'
```

Providing the name of the database and the event monitor name

```
db2evmon -db 'sample' -evm 'dlmon'
```

Event Monitor Data Retention From Release To Release



NEW IN
DB2 10

- **You can upgrade Event Monitor Output Tables after you upgrade the DB2 product**
 - To retain any data that might exist in Event Monitor Tables created before the upgrade
- **The `EVMON_UPGRADE_TABLES` procedure upgrades the definitions of existing Event Monitor (REGULAR and UE) Tables to the current level of DB2**
- **Use the `EVMON_FORMAT_UE_TO_TABLES` procedure with the `UPGRADE_TABLES` option to upgrade the set of Relational Tables produced from an UE table**
- **Implications of not upgrading event monitor tables**
 - Any new columns that have been added to the event monitor in the new release will not be populated with data, and will not be available for queries
 - The values for any monitor elements that previously existed in the old release and that increased in size in the new release might be truncated

Working With Event Monitors - Altering An Event Monitor

NEW IN
DB2 10

▪ LOGICAL DATA GROUPS

– Monitor Elements are grouped on logical data group

– EXAMPLE:

- ACTIVITIES logical data groups:

```
event_activity
event_activity_metrics
event_activitystmt
event_activityvals
```

▪ An Event Monitor cannot be changed

– EXCEPTION: one or more logical data groups can be added* using

```
ALTER EVENT MONITOR ... ADD LOGICAL GROUP...
```

– DEFAULT: all logical data groups that are associated with that Event Monitor are captured

```
CREATE EVENT MONITOR myacts FOR ACTIVITIES
WRITE TO TABLE ACTIVITY, ACTIVITYMETRICS;
ALTER EVENT MONITOR myacts
    ADD LOGICAL GROUP ACTIVITYSTMT
    ADD LOGICAL GROUP ACTIVITYVALS;
```

Create New Event Monitor Example :: Statements

- **EXAMPLE:** Capture all statements where appl_id = myapp

```
CREATE EVENT MONITOR GET_SQL_MYAPP FOR STATEMENTS
WHERE (APPL_ID = 'myapp')
WRITE TO TABLE IN MYTBP PCTDEACTIVATE 70 BUFFERSIZE 8
AUTOSTART ;
```

- **RECCOMENDATIONS:**

- For highly active event monitors use larger buffers
- Place Event Monitor Tables in dedicated Tablespace

- It's possible use `db2evtb1` command to generate sample `CREATE EVENT MONITOR` SQL statement that write to SQL tables

- `SET EVENT MONITOR STATE 1` – to activate an Event Monitor

- `SET EVENT MONITOR STATE 0` – to deactivate an Event Monitor

- Use SQL to query results and find costly and error prone SQL statements

- **CATALOG VIEWS**

- `EVENTS`
- `EVENTMONITORS`
- `EVENTTABLES`



Create New Event Monitor Example :: Locking

- **DB configuration parameters 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

- **TODO: Update default configurations:**

```
db2 update db cfg for SAMPLE using
    mon_lockwait history
    mon_lw_thresh 3000000
    mon_locktimeout hist_and_values
    mon_deadlock without_hist
```



- **WITHOUT_HIST** – collect basic event information
- **HISTORY** – collect up to 250 activities within same unit of work
- **HIST_AND_VALUES** – collect activities and values
- **3000000** – [µs], lock wait condition exists this long before lockwait

Create New Event Monitor Example :: Locking - Continued

- Designed to simplify the task of collecting locking data
- DB2DETAILDEADLOCK Event Monitor is DEPRECATED
 - Disable and remove it issuing the following SQL statements:

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

- Create Event Monitor that writes in an UE Table

```
CREATE EVENT MONITOR LOCKEVMON FOR LOCKING  
WRITE TO UNFORMATTED EVENT TABLE  
(TABLE IMRAN.LOCKEVENTS  
IN APPSPACE PCTDEACTIVATE 85)
```

议程

- 数据库性能问题原因
- DB2数据库监控手段
 - 事件监控
 - 快照监控
 - 监控函数与视图
- DB2 SQL监控与调优
- DB2 Lock监控与调控机制
- DB2 Log监控与调控机制
- DB2 监控常用工具

Snapshot Monitoring

- **Snapshot based monitoring - still viable**
- **Higher overhead than new monitoring framework**
- **Controlled with System Monitor switches**
 - DBM level vs SESSION level
 - Can reset counters at session level – easy to get current results
- **System Monitor data accessible through:**
 - Snapshot Monitor APIs (C or C++ application)
 - **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”
 - Snapshot Administrative Views and Snapshot Table Functions
 - Easy application interface to use
 - Easy to store in database relational tables
 - **SYSIBMADM.SNAP*** Views
 - **SYSPROC.SNAP_*** Table Functions

Snapshot Monitoring - System Monitor Switch Control

- **Globally turned ON/OFF in DBM configuration (online)**

```
db2 get DBM CFG | grep DFT_MON
db2 update DBM CFG using
      DFT_MON_monitorswitch {ON | OFF}
db2 get snapshot for database on MYDB
```

- **Locally turned ON/OFF for the current SESSION**

```
db2 get MONITOR SWITCHES
db2 update MONITOR SWITCHES using
      monitorswitch {ON | OFF}
db2 get snapshot for database on MYDB
```

- **Reset counters**

- At SESSION level: **db2 reset monitor all**
- Globally: **DEACTIVATE/ACTIVATE DATABASE MYDB**

Snapshot Monitoring - System Monitor Switches

- **Before capturing a snapshot or using an event monitor, you must determine what data you need the database manager to gather**
 - Buffer pool activity information
 - Lock, lock wait, and time related lock information
 - Sorting information
 - SQL statement information
 - Table activity information
 - Times and timestamp information
 - Unit of work information

Monitor Switch	DBM Parameter	Information Provided
BUFFERPOOL	DFT_MON_BUFPOOL	Number of reads and writes, time taken
LOCK	DFT_MON_LOCK	Lock wait times, deadlocks
SORT	DFT_MON_SORT	Number of heaps used, sort performance
STATEMENT	DFT_MON_STMT	Start/Stop time, statement identification
TABLE	DFT_MON_TABLE	Measure of activity(rows read/written)
UOW	DFT_MON_UOW	Start/end times, completion status
TIMESTAMP	DFT_MON_TIMESTAMP	Timestamps

Snapshot Monitoring Examples - Database

▪ **GET SNAPSHOT**

– Collects status information and formats the output for the user

– **get snapshot for**

- `database on SAMPLE`
- `database manager`
- `application agentid #`
- `dynamic sql on SAMPLE`



▪ **SNAPDB ADMINISTRATIVE VIEW**

– Allows you to retrieve snapshot information from the database (dbase) logical group for the currently connected database

▪ **SNAP_GET_DB TABLE FUNCTION**

– Returns the same information as the SNAPDB administrative view

Snapshot Monitoring Examples - Database

▪ Example: SNAPDB ADMINISTRATIVE VIEW

- Retrieve the status, platform, location, and connect time for all database members of the currently connected database

```
SELECT SUBSTR(DB_NAME, 1, 20) AS DB_NAME
      , DB_STATUS
      , SERVER_PLATFORM
      , DB_LOCATION
      , DB_CONN_TIME
      , DBPARTITIONNUM
FROM SYSIBMADM.SNAPDB
ORDER BY DBPARTITIONNUM;
```

DB_NAME	DB_STATUS	SERVER_PLATFORM	DB_LOCATION	DB_CONN_TIME	DBPARTITIONNUM
TEST	ACTIVE	AIX64	LOCAL	2006-01-08-16.48.30.665477	0
TEST	ACTIVE	AIX64	LOCAL	2006-01-08-16.48.34.005328	1
TEST	ACTIVE	AIX64	LOCAL	2006-01-08-16.48.34.007937	2

Snapshot Monitoring Examples - Database

▪ Example: SNAP_GET_DB TABLE FUNCTION

- Retrieve the status, platform, location, and connect time as an aggregate view across all database members for all active databases in the same instance that contains the currently connected database

```
SELECT SUBSTR(DB_NAME, 1, 20) AS DB_NAME
      , DB_STATUS
      , SERVER_PLATFORM
      , DB_LOCATION
      , DB_CONN_TIME
FROM TABLE (SNAP_GET_DB (CAST (NULL AS VARCHAR(128)), -2)) AS T
```

DB_NAME	DB_STATUS	SERVER_PLATFORM	B_LOCATION	DB_CONN_TIME
TOOLSDB	ACTIVE	AIX64	LOCAL	2005-07-24-22.26.54.396335
SAMPLE	ACTIVE	AIX64	LOCAL	2005-07-24-22.09.22.013196

Snapshot Monitor - db2top

- Most entries in snapshots are cumulative values and show the condition of the system at a point in time
- DB2TOP can be used to calculate the delta values for those snapshot entries in real time
- Run db2top in interactive mode

```
db2top -d SAMPLE
```

- Run db2top in batch mode

```
db2top -d SAMPLE -f collect.file -C -m 480 -i 15  
db2top -d SAMPLE -f collect.file -b l -A
```

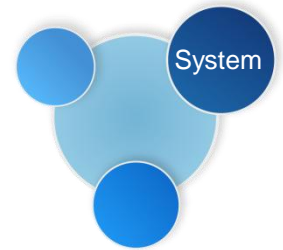
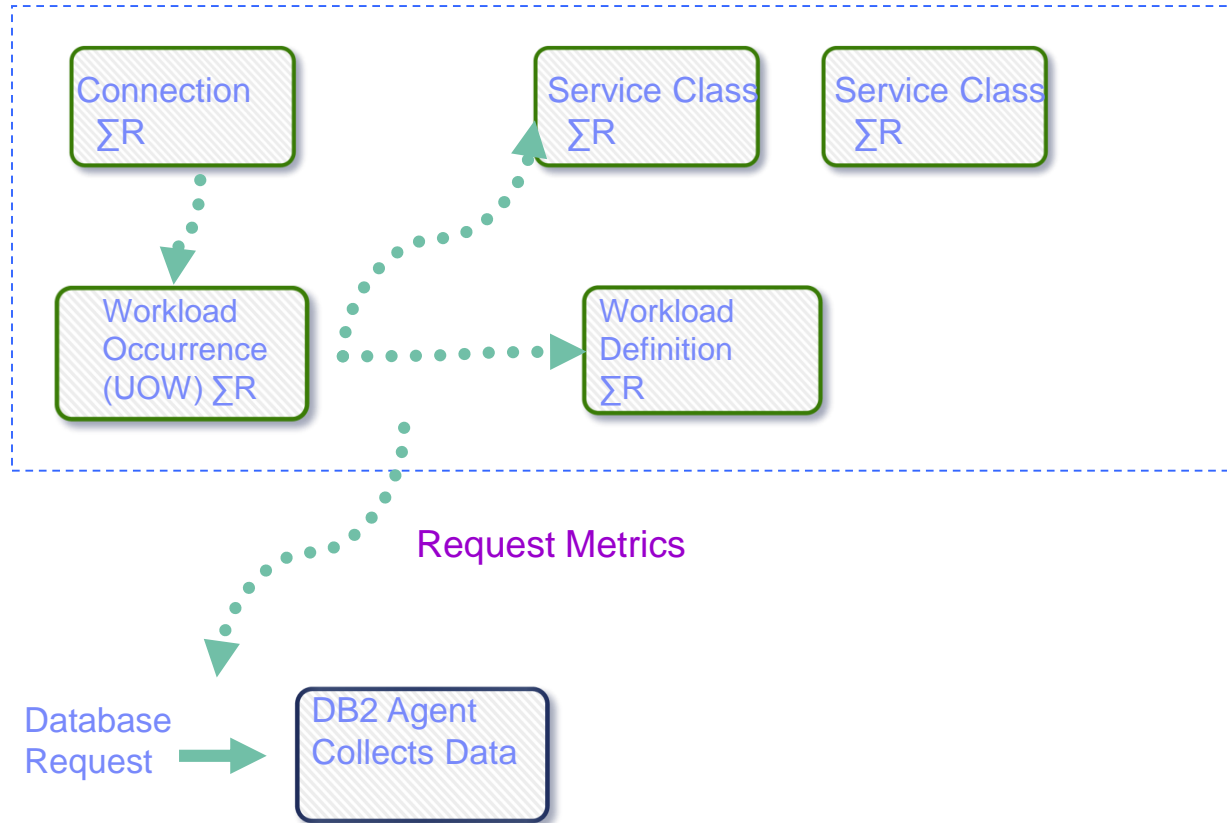
- Object Monitored:

- Database (d)
- Tablespaces (t)
- Dynamic SQL (D)
- Sessions (l)
- Bufferpools (b)
- Locks (U)

议程

- 数据库性能问题原因
- DB2数据库监控手段
 - 事件监控
 - 快照监控
 - 监控函数与视图
- DB2 SQL监控与调优
- DB2 Lock监控与调控机制
- DB2 Log监控与调控机制
- DB2 监控常用工具

In-Memory Metrics :: System Perspective



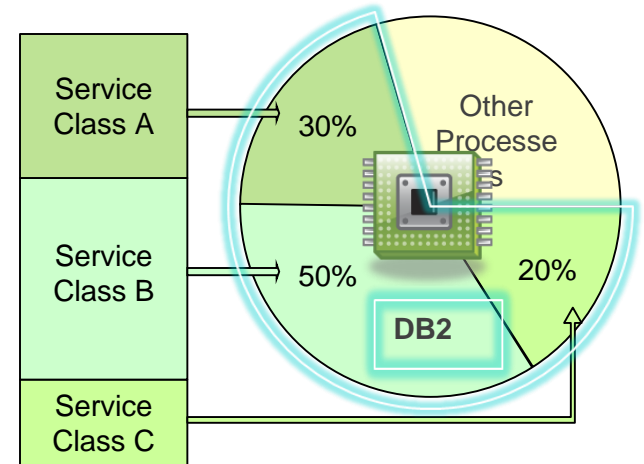
Legend

ΣR = Accumulation of request metrics collected by agent

Access Points :: System Perspective

System Monitoring Table Functions:

- **MON_GET_UNIT_OF_WORK**
 - Returns metrics for one or more units of work
- **MON_GET_WORKLOAD**
 - Returns metrics for one or more workloads
- **MON_GET_CONNECTION**
 - Returns metrics for one or more connections
- **MON_GET_SERVICE_SUBCLASS**
 - Returns metrics for one or more service subclasses
- **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



Access Points :: System Perspective

EXAMPLE:

- Display connections that return the highest volume of data to clients, ordered by rows returned

APPLICATION_HANDLE	ROWS_RETURNED	TCPIP_SEND_VOLUME	EVMON_WAIT_TIME	TOTAL_PEAS	TOTAL_CONNECT_REQUEST_TIME
26	436	341504	0	0	101
20	24	0	0	0	7
22	16	1174	0	0	37
18	1	1422	0	0	852
7	1	0	0	0	4699
25	0	67940	0	0	10
24	0	129842	0	0	6
23	0	97262	0	0	8

```
db2 "SELECT application_handle
      , rows_returned
      , tcpip_send_volume
      , evmon_wait_time
      , total_peas
      , total_connect_request_time
FROM TABLE(MON_GET_CONNECTION(cast(NULL as bigint),-2)) AS t
ORDER BY rows_returned DESC ";
```

Access Points :: System Memory

System Memory Monitoring Table Functions:

- **MON_GET_MEMORY_POOL**

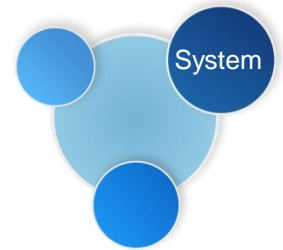
- Retrieves metrics from the memory pools contained within a memory set

- **MON_GET_MEMORY_SET**

- Retrieves metrics from the allocated memory sets, both at the instance level and for all active databases within the instance

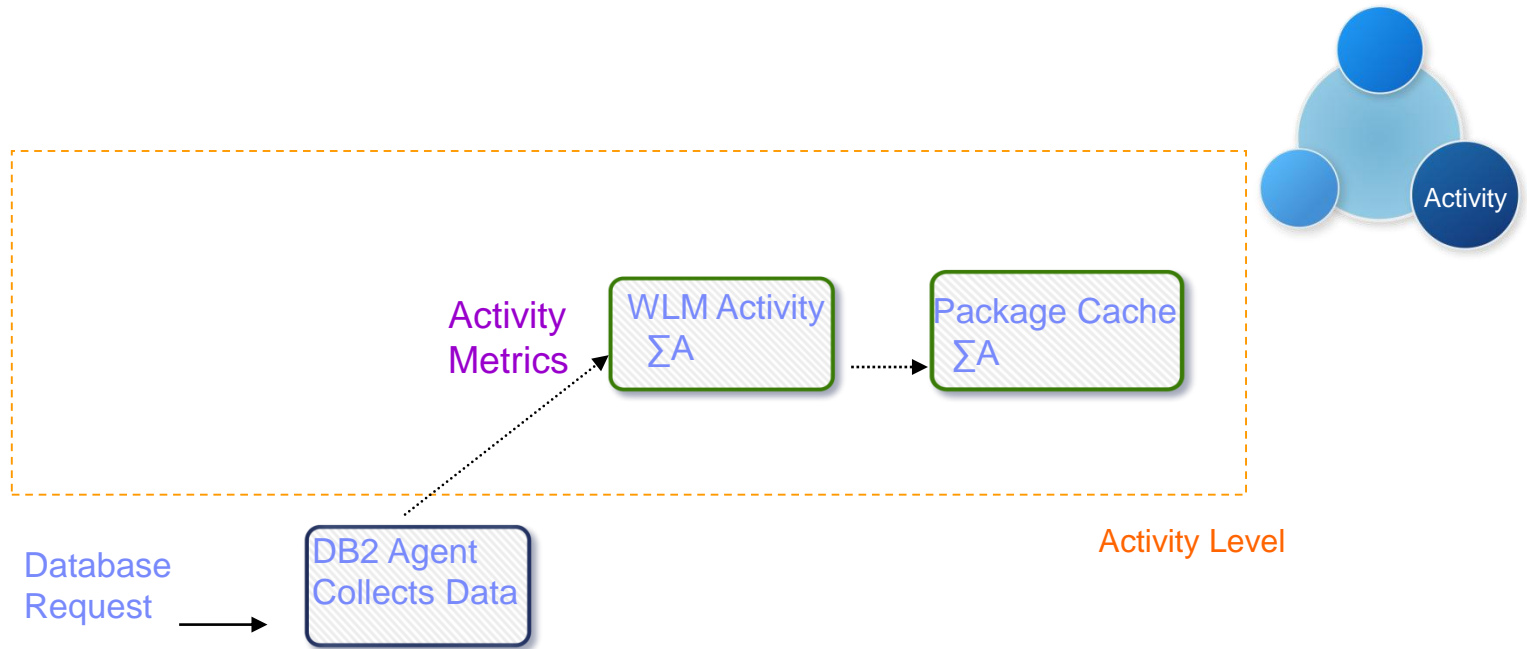
- **Other *Miscellaneous* Monitoring Table Functions:**

- MON_GET_FCM
- MON_GET_FCM_CONNECTION_LIST
- MON_GET_EXTENT_MOVEMENT_STATUS



**Always collected*

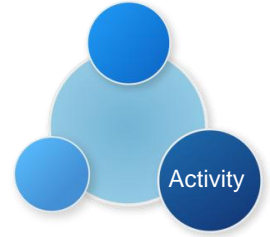
In-Memory Metrics :: Activity Perspective



Legend

ΣA = Accumulation of metrics from activity execution portion of request

Access Points :: Activity Perspective



- **Activities Monitoring Table Functions:**

- **MON_GET_PKG_CACHE_STMT**

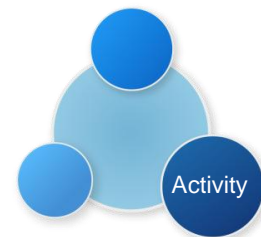
- Aggregates of the metrics gathered during each execution of a particular SQL statement (static or dynamic)

- **MON_GET_PKG_CACHE_STMT_DETAILS**

- **MON_GET_ACTIVITY_DETAILS**

- Information on activities currently running on a system

Access Points :: Activity Perspective



▪ MON_GET_PKG_CACHE_STMT

- Returns a point-in-time view of both **static** and **dynamic** SQL statements in the database package cache

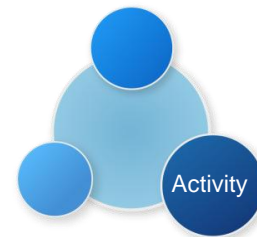
– EXAMPLE:

List all the dynamic SQL statements from the database package cache ordered by the average CPU time:

MEMBER	SECTION_TYPE	AVG_CPU_TIME	LOCK_WAIT_TIME	STMT_TEXT
0 D		11		0 SET CURRENT LOCK TIMEOUT 5
0 D		123		0 INSERT INTO WE_F6QEE8J2Y.WE_0G0_18846 (v
0 D		1753		0 SELECT STATS_FLAG FROM SYSTOOLS.HMON_ATM
0 D		1907		0 INSERT INTO WE_F6QEE8J2Y.WE_0G0_17064 (v
0 D		1920		0 DELETE FROM WE_F6QEE8J2Y.WE_0G0_18846 WH
0 D		2142		0 SELECT TABSCHEMA, TABNAME FROM SYSCAT.TA
0 D		2248		0 SELECT POLICY FROM SYSTOOLS.POLICY WHERE
0 D		2450		0 UPDATE WE_F6QEE8J2Y.WE_0G0_18846 SET val
0 D		33623		0 SELECT * FROM WE_F6QEE8J2Y.WE_0G0_3128 0
0 D		34477		0 UPDATE WE_F6QEE8J2Y.WE_0G0_18536 SET val
0 D		34678		0 DELETE FROM WE_F6QEE8J2Y.WE_0G0_4062 WH

```
db2 "SELECT MEMBER
      , SECTION_TYPE
      , TOTAL_CPU_TIME/NUM_EXEC_WITH_METRICS as AVG_CPU_TIME
      , LOCK_WAIT_TIME
      , SUBSTR(STMT_TEXT,1,40) STMT_TEXT
FROM TABLE(SYSPROC.MON_GET_PKG_CACHE_STMT('D',NULL,NULL,-2)) as T
WHERE T.NUM_EXEC_WITH_METRICS <> 0
ORDER BY AVG_CPU_TIME"
```

Access Points :: Activity Perspective



▪ MON_GET_ACTIVITY_DETAILS (XML)

- Returns details about an activity, including general activity information (like statement text) and a set of metrics for the activity
- EXAMPLE:

Captures information about all the activities currently running on a system:

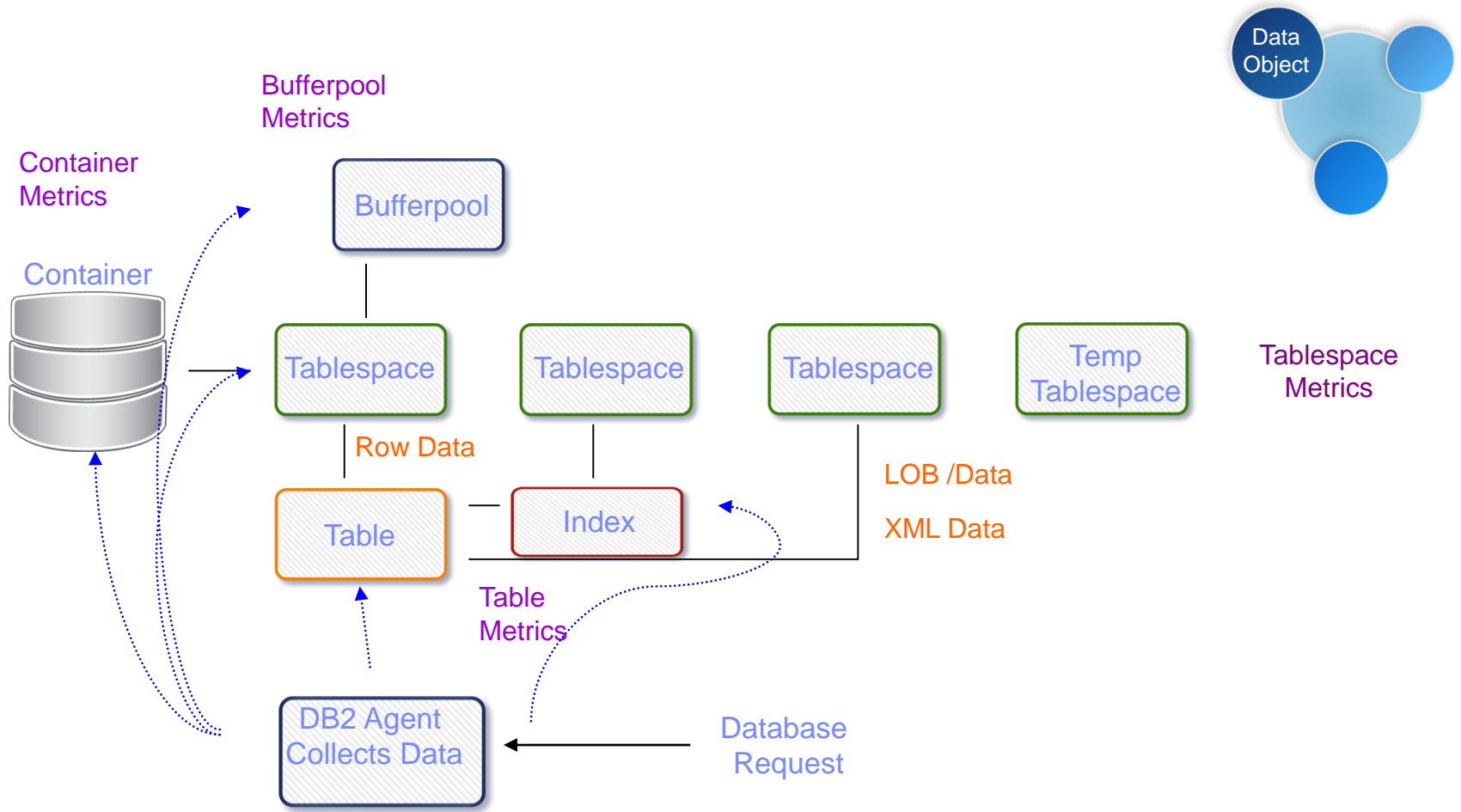
APP...HANDLE	A..._ID	UOW_ID	T...ACT_TIME	T...WAIT_TIME	STMT_TEXT
15	1	5	16	5	select name from sysibm.systables
15	1	3	17	5	select * from sysibm.systables
7	1	49	0	0	with A1 as (select * from)

3 record(s) selected with 1 warning messages printed.

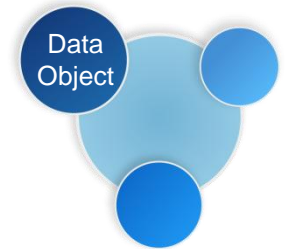
WITH A1 AS

```
(SELECT * FROM TABLE(wlm_get_workload_occurrence_activities(null, -1)) WHERE activity_id > 0)
SELECT A1.application_handle
, A1.activity_id
, A1.uow_id
, total_act_time
, total_act_wait_time
, varchar(actmetrics.stmt_text, 50) AS stmt_text
FROM A1
, TABLE(MON_GET_ACTIVITY_DETAILS(A1.application_handle, A1.uow_id,A1.activity_id, -1)) AS ACTDETAILS
, XMLTABLE ( XMLNAMESPACES( DEFAULT 'http://www.ibm.com/xmlns/prod/db2/mon' )
, '$actmetrics/db2_activity_details'
PASSING XMLPARSE(DOCUMENT ACTDETAILS.DETAILS) AS "actmetrics"
COLUMNS "STMT_TEXT" VARCHAR(1024) PATH 'stmt_text'
, "TOTAL_ACT_TIME" INTEGER PATH 'activity_metrics/total_act_time'
, "TOTAL_ACT_WAIT_TIME" INTEGER PATH 'activity_metrics/total_act_wait_time'
) AS ACTMETRICS
```


In-Memory Metrics :: Data Object Perspective



Access Points :: Data Object Perspective



▪ Data Object Monitoring Table Functions:

– MON_GET_BUFFERPOOL

- Monitor bufferpool efficiency, hit ratio, activity

– MON_GET_CONTAINER

- Monitor container activity, rank, enumerate

– MON_GET_INDEX*

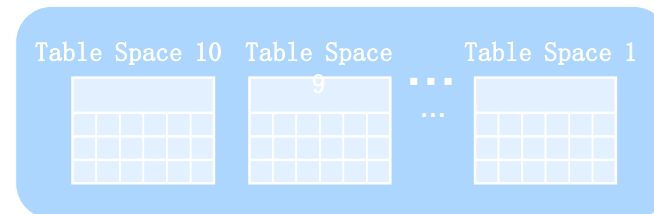
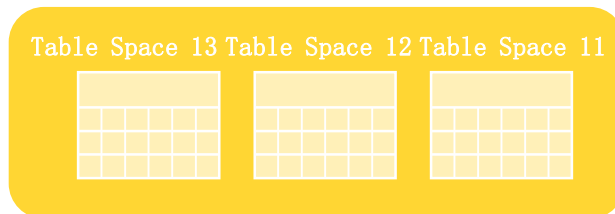
- Monitor index usage, e.g. number of index scans, how many scans are index only scans

– MON_GET_TABLE*

- Monitor activity on table reads, updates, inserts, overflow activity

– MON_GET_TABLESPACE

- Monitor tablespace activity (read and writes), bufferpool activity



**Always collected*

Access Points :: Data Object Perspective

EXAMPLE:

- List utilization of container file systems, ordered by highest utilization

CONTAINER_NAME	FS_ID	FS_USED_SIZE	FS_TOTAL_SIZE	UTILIZATION
/home/db2inst1/db2inst1/NODE0000/DB2PT/T0000000/C0000000.CAT	2050	15056855040	19592417280	76.85
/home/db2inst1/db2inst1/NODE0000/DB2PT/T0000001/C0000000.TMP	2050	15056855040	19592417280	76.85
/home/db2inst1/db2inst1/NODE0000/DB2PT/T0000002/C0000000.LRG	2050	15056855040	19592417280	76.85
/home/db2inst1/db2inst1/NODE0000/DB2PT/T0000003/C0000000.LRG	2050	15056855040	19592417280	76.85

```

SELECT varchar(container_name, 65) as container_name
      , SUBSTR(fs_id,1,10) fs_id
      , fs_used_size
      , fs_total_size
      , CASE WHEN fs_total_size > 0
            THEN DEC(100*(FLOAT(fs_used_size)/FLOAT(fs_total_size)),5,2)
            ELSE DEC(-1,5,2)
            END as utilization
FROM TABLE(MON_GET_CONTAINER('',-1)) AS t
ORDER BY utilization DESC
    
```

In-Memory Metrics :: Lock Perspective

▪ Lock Monitoring Table Functions:

–MON_GET_LOCKS

- List of all locks in the currently connected database

–MON_GET_APPL_LOCKWAIT

- All locks that each application's agents, connected to the current database, are waiting to acquire



**Always collected*

Monitor Views

- **MON_BP_UTILIZATION**
 - Buffer pool efficiency (e.g. hit ratios, average read and write times)
- **MON_CONNECTION_SUMMARY**
 - Incoming work per connection
 - New monitor elements: total_app_commits, total_app_rollbacks
- **MON_CURRENT_SQL**
 - Currently executing SQL statements (both static and dynamic)
- **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 currently in package cache, both dynamic and static
- **MON_SERVICE_SUBCLASS_SUMMARY**
 - Returns key metrics for all service subclasses in the currently connected database
- **MON_WORKLOAD_SUMMARY**
 - High level, aggregated metrics (percentage breakdowns, wait vs. active, totals)
- **MON_TBSP_UTILIZATION**
 - List tablespace information, state, high watermark, and hit ratios



议程

- 数据库性能问题原因
- DB2数据库监控手段
 - 事件监控
 - 监控快照
 - 监控函数与视图
- DB2 SQL监控与调优
- DB2 Lock监控与调控机制
- DB2 Log监控与调控机制
- DB2 监控常用工具

Agenda

- **Problematic SQL & Situation**
- **Response time solution**
- **SQL costs solution**
 - Snapshot Monitoring
 - Event Monitoring
 - SQL Monitoring Interfaces
 - Analyzing SQL
 - Explain Tools
 - Visual Explain
 - Statement Concentrator
- **Making Performance Improvements**
 - Database Objects
 - Better Coding
 - Design Advisor
 - Other Considerations

Agenda

- **Problematic SQL & Situation**
- **Response time solution**
- **SQL costs solution**
 - Snapshot Monitoring
 - Event Monitoring
 - SQL Monitoring Interfaces
 - Analyzing SQL
 - Explain Tools
 - Visual Explain
 - Statement Concentrator
- **Making Performance Improvements**
 - Database Objects
 - Better Coding
 - Design Advisor
 - Other Considerations

Problematic SQL

- Query optimization is the main common factor that affects application performance
- Problematic SQL statements slow down application
- Problematic SQL statements can be **detected** with:
 - Snapshot monitors
 - Event monitors
 - SQL monitoring interfaces
 - Administrative views
 - Table functions
- Problematic SQL statements can be **analyzed** with:
 - Visual Explain
 - Text Explain Facility



Problematic SQL

- Problematic SQL statements can be **improved** with:
 - Design Advisor (database objects)
 - Indexes
 - Materialized Query Tables
 - Multi-dimensional Clustering
 - Database Partitioning
 - Better coding
 - Statement Concentrator
 - DB2 Optim Query Workload Tuner
 - **DB2 10**
 - Explore new performance enhancing features and take advantage of them



A number of performance improvements have been included in DB2 10 to improve the speed of many queries

**NEW IN
DB2 10**

These improvements are automatic; there are no configuration settings or changes to the SQL statements required

A common scenario

- **New major application version**
- Expectations:
 - Increase value
 - Reduce response time
 - Reduce CPU utilization
 - Increase user capacity
- Use `db2adviz` with `-wlm evmonname` or `-w workloadname` option to capture and advise about
 - Index
 - MDC
 - MQT
 - DB Partitioning
- Capture additional high cost SQL for further analysis and tuning
- Further analyze and tune with Visual Explain
- `STMT_CONC` (DB CFG parameter)



Actual Situation

- Benchmark for new application version slower than previous application version
- DB2 is configured well
 - Large part of DB2 tuning done in previous application version
 - Autoconfigure has been run on new schema
- New SQL may need tuning
 - Complementary database objects (indexes, MDC, MQT, partitioning)
 - Coding (apply best practices)

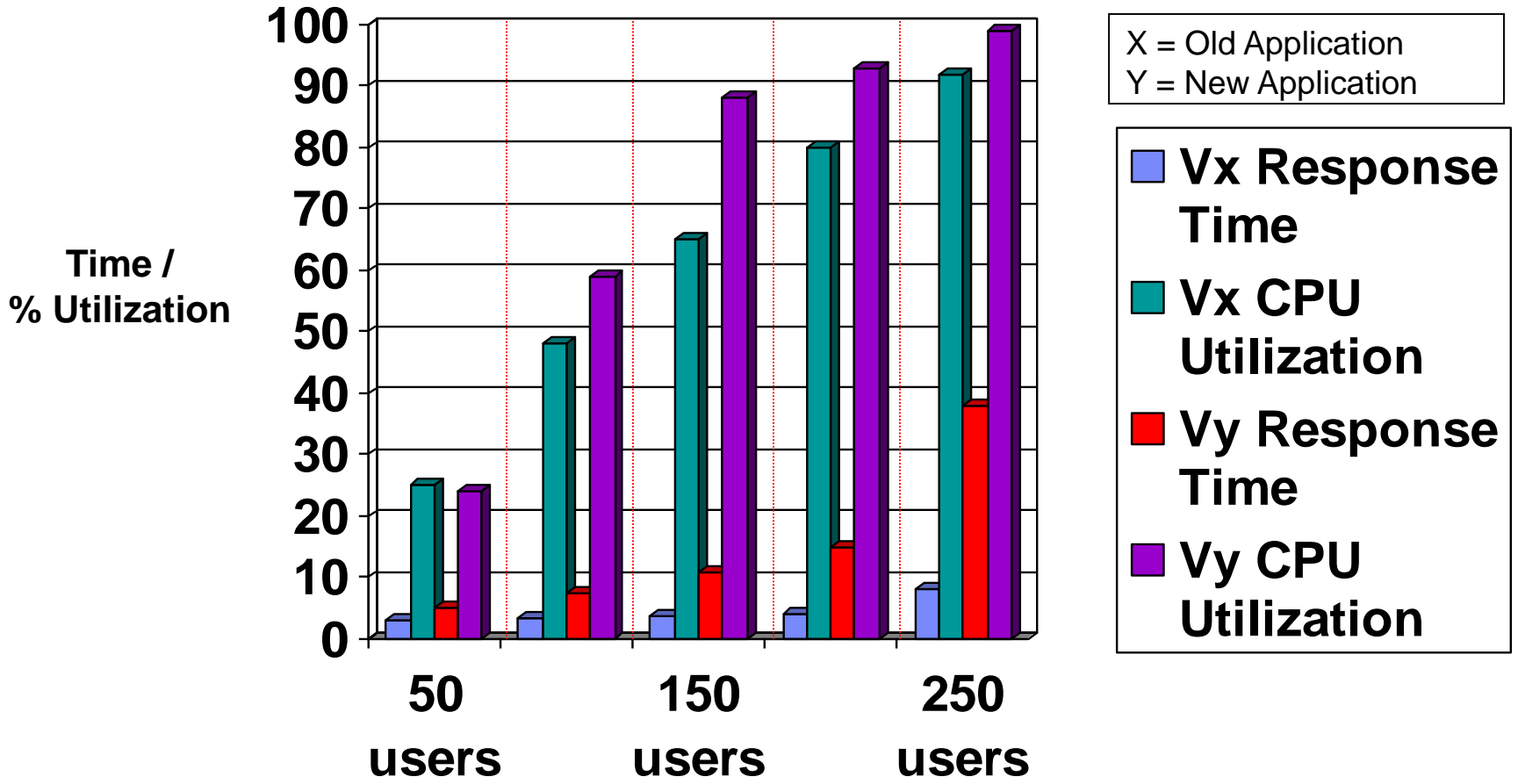


NEW IN
DB2 10

Use the AUTOCONFIGURE command to get recommendations from the configuration advisor. Although the wizard interface for the configuration advisor is discontinued in DB2 10, the configuration advisor is still available by using the AUTOCONFIGURE command

Original Comparative Application Results Version X vs. Version Y

- New version is slower and consumes more resources



Agenda

- **Problematic SQL & Situation**
- **Response time solution**
- **SQL costs solution**
 - Snapshot Monitoring
 - Event Monitoring
 - SQL Monitoring Interfaces
 - Analyzing SQL
 - Explain Tools
 - Visual Explain
 - Statement Concentrator
- **Making Performance Improvements**
 - Database Objects
 - Better Coding
 - Design Advisor
 - Other Considerations

Use Event Monitor for Activities & db2advis To Initially Tune Workload

- Event monitor for activities captures both dynamic and static SQL
- WLM feature can focus on specific workloads and service classes
- Optionally Snapshot Monitoring or Monitoring Table Functions can be used in place of Event Monitor for Activities
- Integrates with design advisor (db2advis) for recommendations about:
 - Indexes
 - MQTs
 - MDCs
 - Database partitioning



**NEW IN
DB2 10**

Use the db2advis command to get recommendations from the design advisor. The wizard interface for the design advisor is discontinued in DB2 10, but the design advisor is still available using the db2advis command

Steps to take for EVM for Activities & db2advis Analysis

1. Alter workload `sysdefaultuserworkload` (or your desired workload) to collect activity data on coordinator with details and values
2. Create event monitor `db2activities` for `activities`
3. Set event monitor `db2activities` state 1

```
ALTER WORKLOAD SYSDEFAULTUSERWORKLOAD COLLECT ACTIVITY DATA
ON ALL WITH DETAILS AND VALUES;

CREATE EVENT MONITOR DB2ACTIVITIES
FOR ACTIVITIES WRITE TO TABLE;

SET EVENT MONITOR DB2ACTIVITIES STATE 1;
```

4. Work is run in `sysdefaultuserworkload` (or your desired workload)
5. Use:

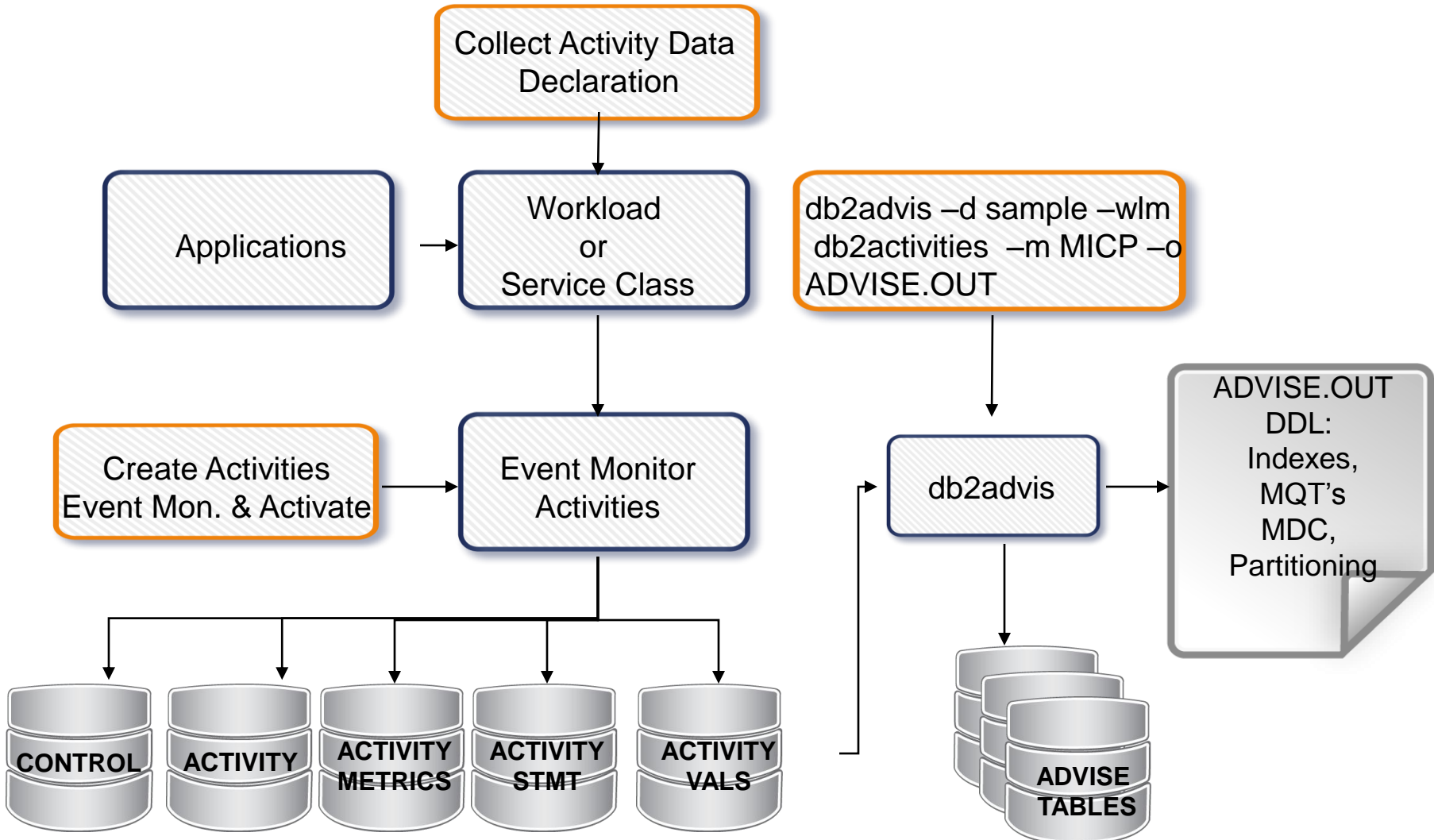
```
db2advis -d sample -wlm db2activities -m MICP -o advise.out
```

New event monitors features:

- All event monitors now support the WRITE TO TABLE target
- Existing event monitors that write to tables can be altered to capture additional logical data groups

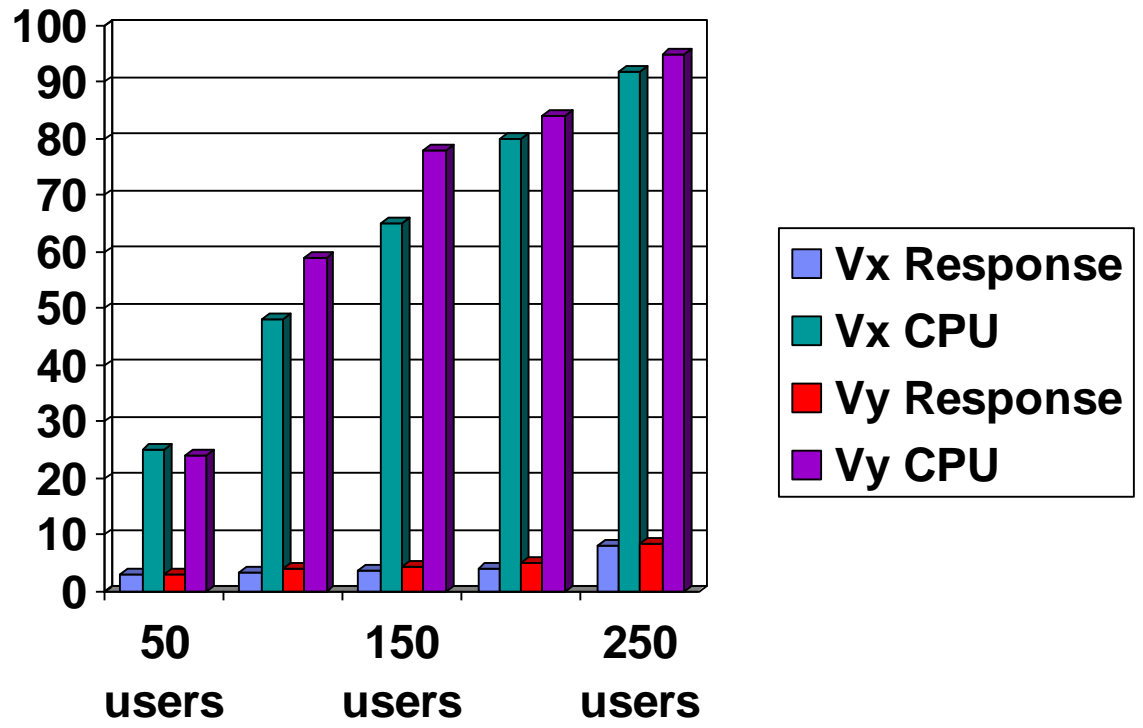
NEW IN
DB2 10

db2advis and EVM for Activities flow



After db2advis Recommendations and Implementation Comparative Application Results Version X vs. Version Y

- Now it's better, but the new application should be much faster
- SQL costs... need further analysis, still very high



Agenda

- **Problematic SQL & Situation**
- **Response time solution**
- **SQL costs solution**
 - Snapshot Monitoring
 - Event Monitoring
 - SQL Monitoring Interfaces
 - Analyzing SQL
 - Explain Tools
 - Visual Explain
 - Statement Concentrator
- **Making Performance Improvements**
 - Database Objects
 - Better Coding
 - Design Advisor
 - Other Considerations

Detect SQL costs

Snapshot Monitoring?



Event Monitoring?

SQL Interfaces?

Agenda

- **Problematic SQL & Situation**
- **Response time solution**
- **SQL costs solution**
 - Snapshot Monitoring
 - Event Monitoring
 - SQL Monitoring Interfaces
 - Analyzing SQL
 - Explain Tools
 - Visual Explain
 - Statement Concentrator
- **Making Performance Improvements**
 - Database Objects
 - Better Coding
 - Design Advisor
 - Other Considerations

Snapshot Monitoring

- To request snapshot information about the dynamic 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 SQL
- Formatted text output



Switches must be ON

Agenda

- **Problematic SQL & Situation**
- **Response time solution**
- **SQL costs solution**
 - Snapshot Monitoring
 - Event Monitoring
 - SQL Monitoring Interfaces
 - Analyzing SQL
 - Explain Tools
 - Visual Explain
 - Statement Concentrator
- **Making Performance Improvements**
 - Database Objects
 - Better Coding
 - Design Advisor
 - Other Considerations

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 and Static 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*

Agenda

- **Problematic SQL & Situation**
- **Response time solution**
- **SQL costs solution**
 - Snapshot Monitoring
 - Event Monitoring
 - SQL Monitoring Interfaces
 - Analyzing SQL
 - Explain Tools
 - Visual Explain
 - Statement Concentrator
- **Making Performance Improvements**
 - Database Objects
 - Better Coding
 - Design Advisor
 - Other Considerations

SQL Monitoring Interfaces

- **Administrative Views**

- Easy-to-use application programming interface
- Execute administrative functions through SQL

- **Table functions MON_GET_...**

- Enhanced reporting and monitoring of the database system, data objects, and the package cache
- **Have a lower impact on the system than existing system monitor and snapshot interfaces**



Examples of SQL interfaces – Finding SQL Costs

▪ Snapshot 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 vs. Waiting)



DB2 10 returns additional columns such as columns that report information about data tags in service class thresholds

- MON_GET_PKG_CACHE_STMT



DB2 10 returns additional columns that report metrics about I/O server efficiency, processing time for authentication, statistics generation, statement execution, high water mark input values, and extended latch waits

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;
```

Agenda

- **Problematic SQL & Situation**
- **Response time solution**
- **SQL costs solution**
 - Snapshot Monitoring
 - Event Monitoring
 - SQL Monitoring Interfaces
 - Analyzing SQL
 - Explain Tools
 - Visual Explain
 - Statement Concentrator
- **Making Performance Improvements**
 - Database Objects
 - Better Coding
 - Design Advisor
 - Other Considerations

SQL Explain Tools

Graphical

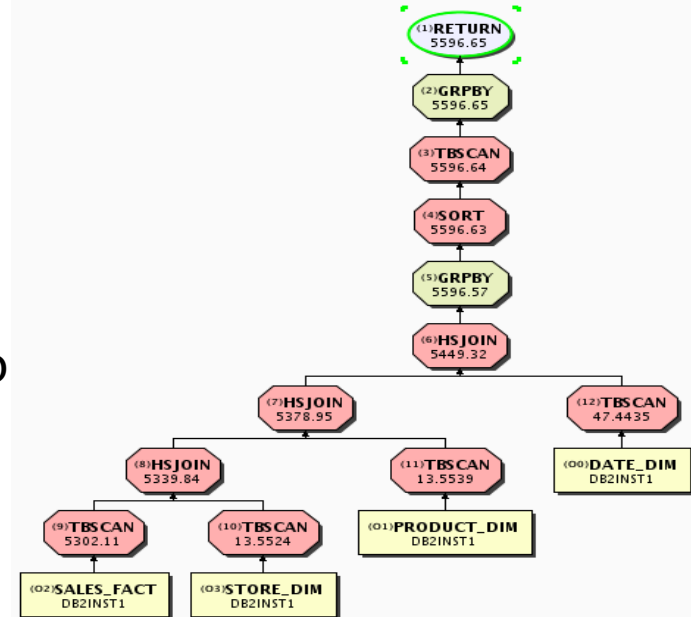
- Easy to quickly spot the problem
- Provides drill down functionality
- Multiple images can be stored for comparison

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 or db2expln



In DB2 10 the db2exfmt command output now shows the table space attributes value for each table space containing a partitioned table



```

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

      Rous
      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
      -----
      Name: (1R2)
    
```

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**

NEW IN
DB2 10

In general Control Center and related wizards and advisors have been discontinued in DB2 10. These have been replaced by a new set of GUI tools: IBM Data Studio and IBM InfoSphere Optim tools

How to use Visual Explain

■ Invoke from

- IBM Data Studio
- IBM Optim Workload Query Tuner

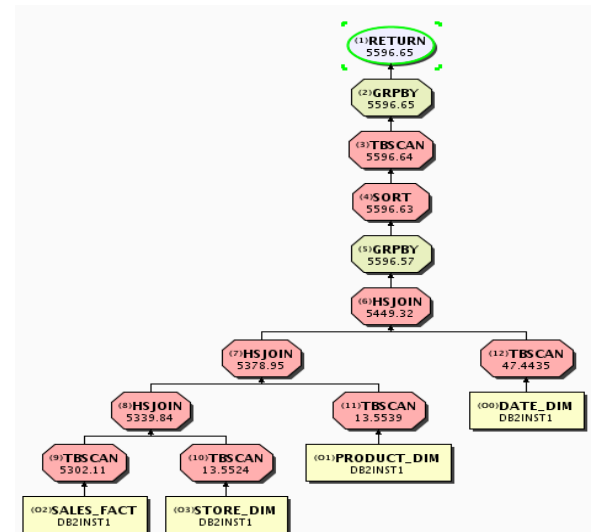
■ Enter SQL to be analyzed

- Trap the poorly-running SQL statement from your program or from performance monitors, or create a brand new statement
- The text can then just be typed or copied into the input box
- Click the Visual Explain button

■ 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**
 - 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**

The screenshot shows a database query tool interface. At the top, a window titled "bp_revenue.sql" displays the following SQL query:

```

select count(*) from bp_revenue WHERE COMPANY_ID IS NULL
;
select imt, sum(db2_q109) db2_q109, sum(db2_q209) db2_q209, sum(db2_q309) db2_q309, s
;
select co.*, ed.* from education ed, courses co where ed.course_id = co.id
;
select distinct name, cast (DESCRIPTION as varchar(100)) desc from courses co
;
    
```

Below the query editor, the "Access Plan Diagram" is visible. It shows a hierarchical execution plan:

- Root node: **(#)RETURN** (Cost: 601.29)
- Intermediate node: **(#)HSJOIN** (Cost: 601.29)
- Leaf nodes: **(#)TBSCAN** (Cost: 570.523) and **(#)TBSCAN** (Cost: 30.2656)
- Source tables: **(#)EDUCATION** (DB2INST1) and **(#)COURSES** (DB2INST1)

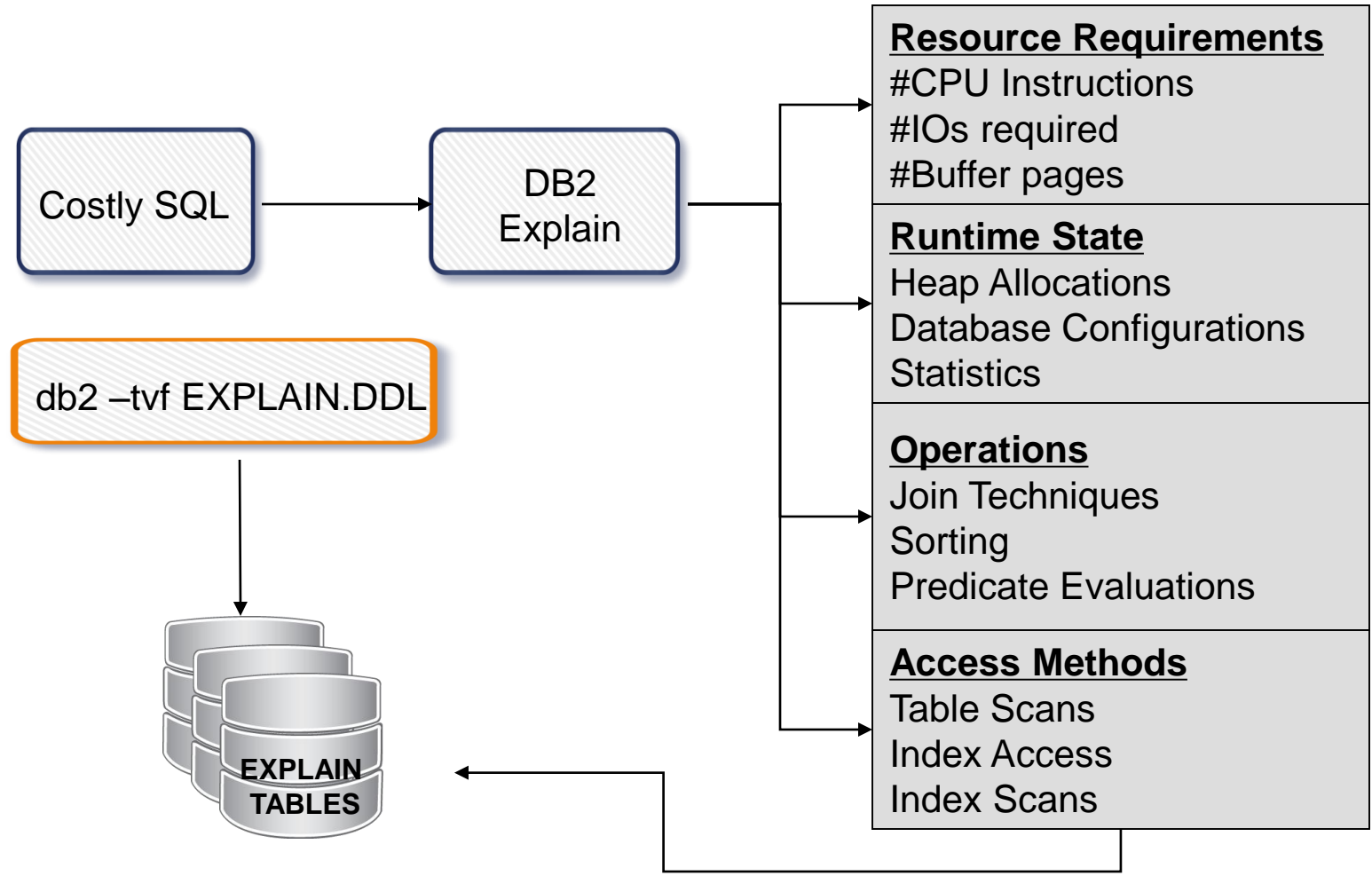
The "Properties" pane on the left provides details for the selected node:

Overview of Diagram
Description of Selected Node
 Description of Selected Node
 Displays information about the node that is highlighted in the diagram.

Attributes

NAME	VALUE
Operator Identifier	1
Operator Type	RETURN
Cumulative Total Cost	601.29

Overview: Costly SQL → Explain



Capturing and accessing section actuals

- **Section actuals are runtime statistics collected during the execution of the section for an access plan**
- **The section actuals values can then be compared with the estimated access plan values generated by the optimizer to assess the validity of the access plan**
- **To capture a section with actuals, you need to enable section actuals:**
 - Enable section actuals for the entire database using the `section_actuals` database configuration parameter

db2 update database configuration using `section_actuals` base

- Enable section actuals for a specific application using the `WLM_SET_CONN_ENV` procedure

CALL `WLM_SET_CONN_ENV(...)`

Capturing and accessing section actuals

- Section actuals can be accessed using the **EXPLAIN_FROM_ACTIVITY** procedure

CALL EXPLAIN_FROM_ACTIVITY (...)

- You can format the explain data using the **db2exfmt** command and specifying, as input, the explain instance key that was returned as output from the **EXPLAIN_FROM_ACTIVITY** procedure



There is graphical interface where you can collect and analyze actuals. This interface is called **Optim Query Workload Tuner (OQWT)**

```

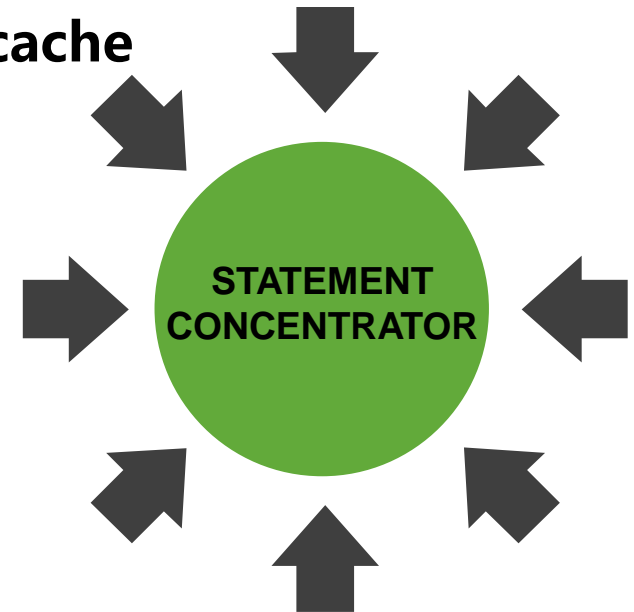
Rows
Rows Actual
RETURN
( 1)
Cost
I/O
|
54
396
>^HSJOIN
( 2)
153.056
NA
/-----+-----\
54                20
396                0
>^HSJOIN          TBSCAN
( 3)              ( 12)
140.872           11.0302
NA                NA
(continued below) |
                   20
                   NA
                   TABLE: SYSIBM
                   SYSAUDITPOLICIES
    
```

Agenda

- **Problematic SQL & Situation**
- **Response time solution**
- **SQL costs solution**
 - Snapshot Monitoring
 - Event Monitoring
 - SQL Monitoring Interfaces
 - Analyzing SQL
 - Explain Tools
 - Visual Explain
 - Statement Concentrator
- **Making Performance Improvements**
 - Database Objects
 - Better Coding
 - Design Advisor
 - Other Considerations

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**



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
```

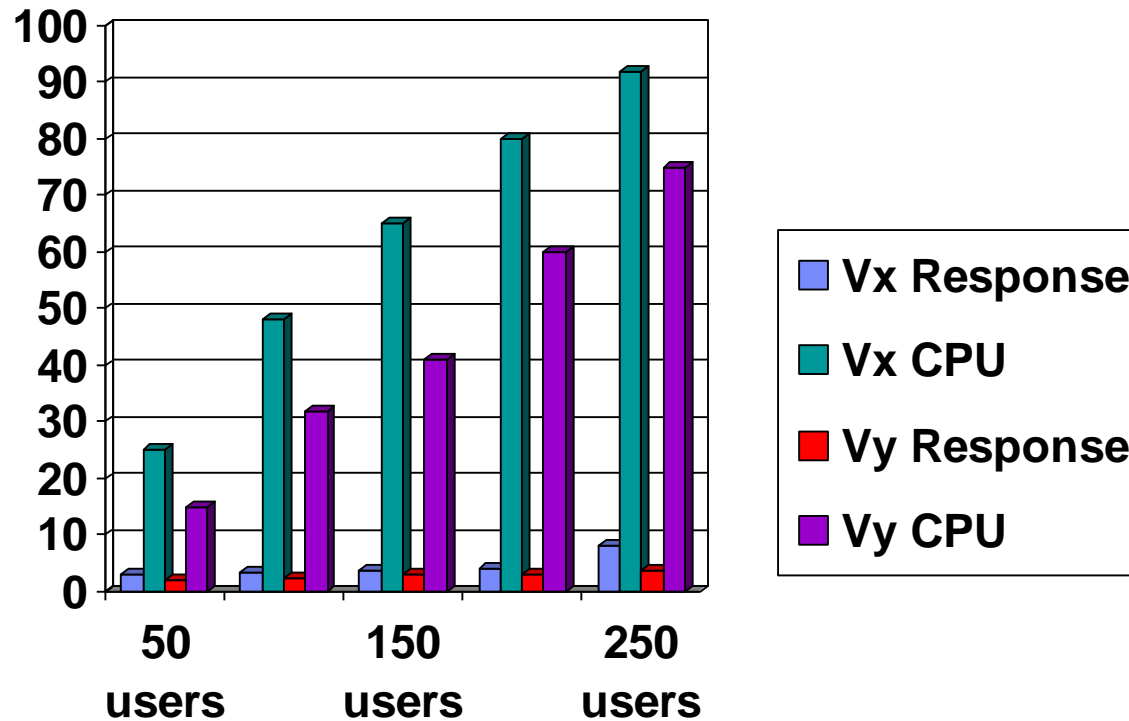
Package
cache will use
the statement

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

After Statement Concentrator

Comparative Application Results Version X vs. Version Y

- Now it's much faster
- SQL costs look much better!



Agenda

- **Problematic SQL & Situation**
- **Response time solution**
- **SQL costs solution**
 - Snapshot Monitoring
 - Event Monitoring
 - SQL Monitoring Interfaces
 - Analyzing SQL
 - Explain Tools
 - Visual Explain
 - Statement Concentrator
- **Making Performance Improvements**
 - Database Objects
 - Better Coding
 - Design Advisor
 - Other Considerations

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
- Try to avoid sorts for ORDER BY and GROUP BY clauses
- Create an index on:
 - Join columns
 - Selective filter columns
 - Columns frequently used for ordering
- To provide index-only access, which avoids the cost of accessing data pages
- As the only automatic way to enforce uniqueness in a relational database

```
CREATE UNIQUE INDEX EMP_IX ON EMPLOYEE (EMPNO) INCLUDE (FIRSTNAME, JOB)
```

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

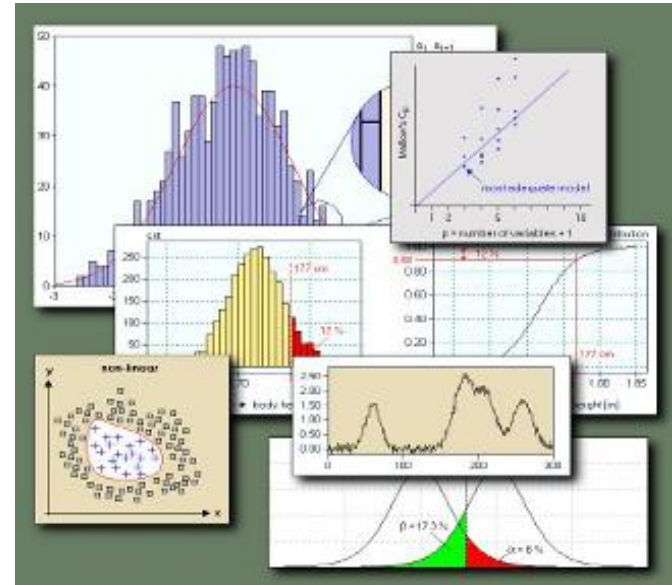
- Index every PK and most FKs in a database
- Indexes on FKs also improve the performance of RI checking
- Explicitly provide an index for the PK
- Columns frequently referenced in WHERE clauses are good candidates for an index
- An exception to this rule is when the predicate provides minimal filtering
 - 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



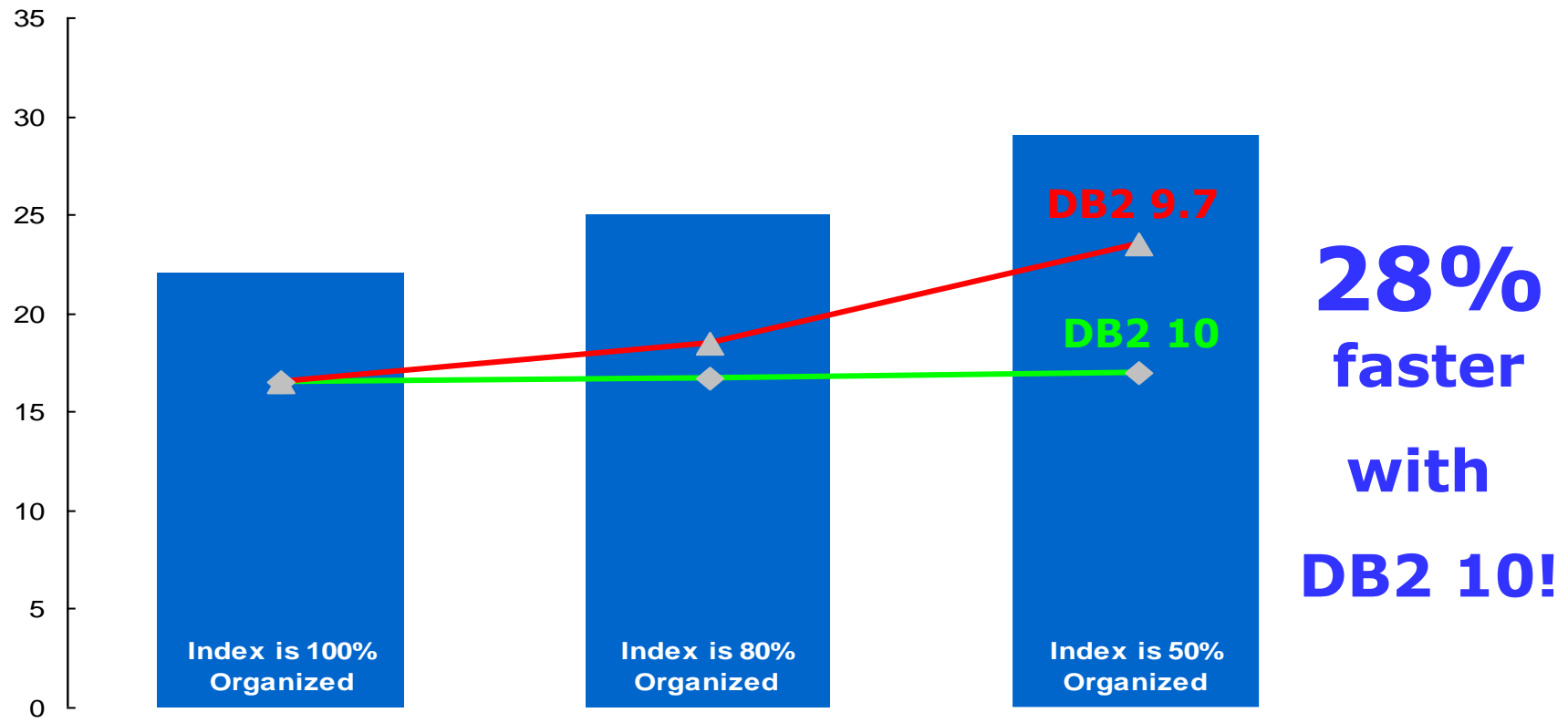
Index Management Redefined in DB2 10

- **Jump scan**
 - Need fewer indexes
- **Smart index prefetching**
 - Faster index access & fewer index reorgs
- **Smart data prefetching**
 - Faster data scans & fewer data reorgs
- **Predicate evaluation avoidance**
 - Faster index scans
- **Higher performance**
 - Faster index performance
- **Lower costs**
 - Fewer indexes to maintain
 - Dramatic reduction in index reorgs

Everything here is NEW in DB2 10!



Smart Index Prefetching = Fewer Index Reorgs



**28%
faster
with
DB2 10!**



DB2 V10.5 Index improvements

- **Expression based indexes improve query performance**
- You can create an index that includes expressions.
- Best suited when you want an efficient evaluation of queries that involve a column expression.
- Values are transformed by the expressions that you specify.
- For indexes created with the UNIQUE option, the uniqueness is enforced against the values that are stored in the index. The uniqueness is not enforced against the original column values.
- **Clause EXCLUDE NULL KEYS from Index**
- Resolves issues around sparsity of key values
- Reduces size of index object

Materialized Query Tables (MQTs)

- **Powerful way to improve response time for complex queries**
- **Queries that might require one or more of the following operations:**
 - Aggregate data over one or more dimensions
 - Join and aggregate data over a group of tables
 - Data from a commonly accessed subset of data
 - Repartitioned data from a table
- **Without MQTs, similar queries that do the same expensive operations may repeatedly compute the same results however:**
 - Consume extra disk space
 - Must be updated to maintain their consistency
 - Requires its own indexes for efficient access
- **The benefit of an MQT relative to its cost is therefore maximized if the MQT benefits many queries, particularly costly queries, or frequently executed queries**

Materialized Query Tables (MQTs)

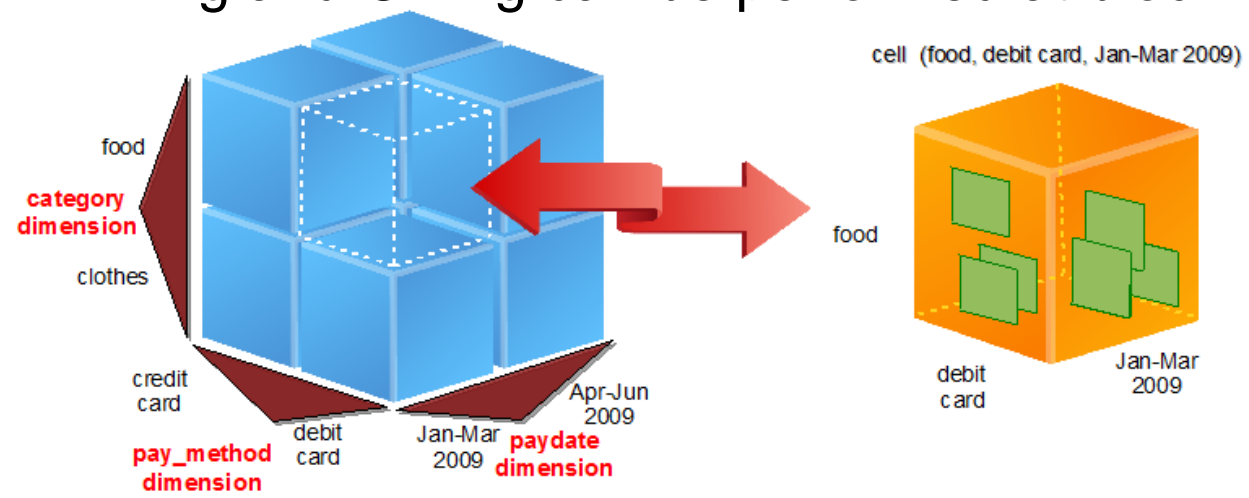
- An example of creating an MQT is shown below. The table definition is written just as it is when you use DDL for creating a normal table, but you do not define the columns and data types. You write an SQL query that describes the structure of the MQT, and then the MQT is filled with the data that the query returns:

```
CREATE TABLE MY_MQT AS (
    SELECT SUM(T1.Sales) as Total_Sales,
           SUM(T1.COGS) as Total_Costs,
           SUM(T1.Expenses) as Total_Expenses,
           T2.Prd_Family as Product_Family,
           T3.Years Year, T3.Quarter as Quarter
    FROM SALES_FACT T1, PRODUCT T2, TIME T3
    WHERE T1.Product_ID = T2.Product_ID AND
           T1.Time_ID = T3.Time_ID
    GROUP BY T2.Prd_Family, T3.Year, T3.Quarter)
DATA INITIALLY DEFERRED REFRESH DEFERRED
ENABLE QUERY OPTIMIZATION MAINTAINED BY SYSTEM
```

- Based on the MQT DDL above, the DB2 Optimizer decides whether to use the base table or the MQT. When a query is run against the database, the DB2 Optimizer rewrites the query if the query matches the MQT definition and using the MQT would reduce the query costs. When the access plan has a better result with the MQT, then the query is run against the MQT to access the data

Multidimensional Clustering (MDC)

- Enables a table to be physically clustered on several dimensions simultaneously
- Primarily intended for data warehousing and large database environments
- DB2 places records that have the same column values in physical locations that are close together
 - Block indexes are smaller than RID indexes
 - Faster lookup
 - Scan only required blocks
 - Index ANDing and ORing can be performed at block level



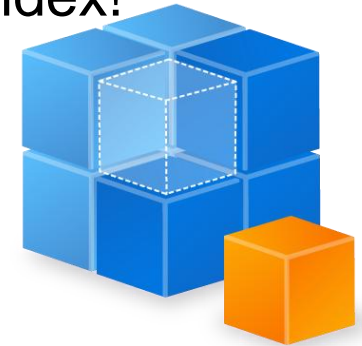
Multidimensional Clustering differences

▪ Without MDC:

- 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

▪ With MDC:

- 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



Agenda

- **Problematic SQL & Situation**
- **Response time solution**
- **SQL costs solution**
 - Snapshot Monitoring
 - Event Monitoring
 - SQL Monitoring Interfaces
 - Analyzing SQL
 - Explain Tools
 - Visual Explain
 - Statement Concentrator
- **Making Performance Improvements**
 - Database Objects
 - Better Coding
 - Design Advisor
 - Other Considerations


Writing Efficient SELECT statements

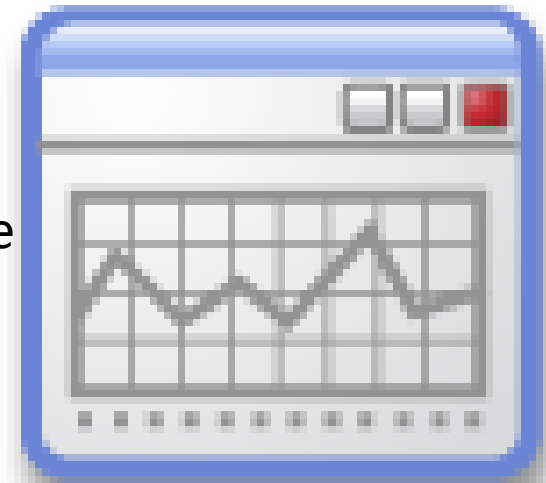
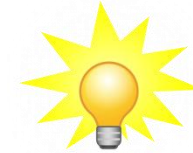
- **Specify only columns that you need**
- **Use predicates that limit to those rows that you need**
- **Avoid numeric data type conversions**
- **Avoid data type mismatches**
- **Preferred data types**
 - CHAR instead of VARCHAR for short columns
 - INTEGER instead of FLOAT, DECIMAL or DECFLOAT
 - DECFLOAT 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 10 Up to 3x Faster Query Performance

- Multi-core parallelism enhancements
- Performance improvements in  for:
 - Queries over star schemas
 - Queries with joins and sorts
 - Queries with aggregation
 - Hash joins
- **Higher performance**
 - Up to **35% faster** out-of-the-box performance
 - Up to **3x faster** than DB2 9.7
- **Lower costs**
 - No need of more/new hw
 - Postpone hardware upgrades



Agenda

- **Problematic SQL & Situation**
- **Response time solution**
- **SQL costs solution**
 - Snapshot Monitoring
 - Event Monitoring
 - SQL Monitoring Interfaces
 - Analyzing SQL
 - Explain Tools
 - Visual Explain
 - Statement Concentrator
- **Making Performance Improvements**
 - Database Objects
 - Better Coding
 - Design Advisor
 - Other Considerations

Using Design Advisor

- **Create the explain tables:**

- The EXPLAIN.DDL file is located in the **\$INSTHOME/sql/lib/misc** directory
- Go to this directory and issue the `db2 -tvf EXPLAIN.DDL` command

- **Run the EXPLAIN.DDL DB2 command file:**

```
db2 CONNECT TO database-name  
db2 -tvf EXPLAIN.DDL
```

- **Command Line Usage**

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

- **Parameters:**

- d database name
- m M-MQT I-Indexes C-MDC P-Partitioning

Workload type keyword: (choose one)

- s single SQL statement
- i SQL from input file
- w SQL from ADVISE_WORLOAD table by workload name
- g Get workload from dynamic SQL snapshot
- wlm SQL from the ACTIVITY Event Monitor tables

Other keywords:

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

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"
```

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

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

- Example: Workload from then Activity Event Monitor tables

Agenda

- **Problematic SQL & Situation**
- **Response time solution**
- **SQL costs solution**
 - Snapshot Monitoring
 - Event Monitoring
 - SQL Monitoring Interfaces
 - Analyzing SQL
 - Explain Tools
 - Visual Explain
 - Statement Concentrator
- **Making Performance Improvements**
 - Database Objects
 - Better Coding
 - Design Advisor
 - Other Considerations

Other factors influencing query performance



- Accurate database statistics – RUNSTATS

A number of improvements have been made to the RUNSTATS command to make statistics gathering faster in some cases. The command parameters have also been simplified

- 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



NEW IN
DB2 10

Optimization profile supports registry variables and inexact matching

Summary

- Many performance improvements have been included in DB2 10 to improve the speed of many queries. These improvements are automatic; there are no configuration settings or changes to the SQL statements required!
- Use event monitors and Design Advisor for better response time results
- Use snapshot monitoring, event monitoring or SQL monitoring interfaces to identify SQL costs
- Visual Explain tools are a good way to review the query access plans
- Statement Concentrator as alternative to reduce SQL costs
- You can make performance improvements after database design and coding phase
- Optim Query Workload Tuner is a good tool when you need to make performance improvements



议程

- 数据库性能问题原因
- DB2数据库监控手段
 - 事件监控
 - 监控快照
 - 监控函数与视图
- DB2 SQL监控与调优
- DB2 Lock监控与调控机制
- DB2 Log监控与调控机制
- DB2 监控常用工具

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

- Lock Waits
- Lock Timeouts
- Lock Escalations
- 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 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**

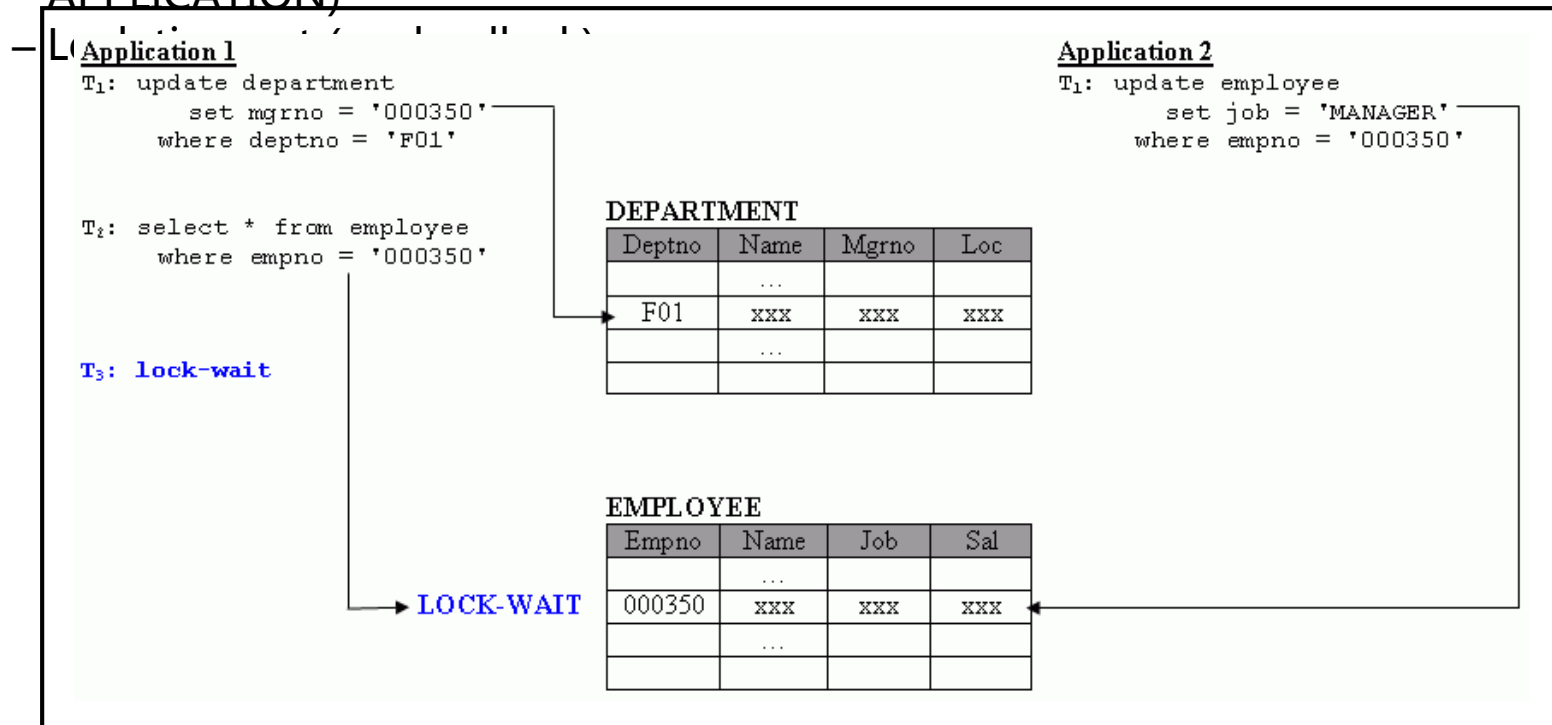
- Lock timeouts result in the application not completing the transaction and performing a rollback

- **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)



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 cause transactions to be re-executed**
 - The victim application has to re-execute the transaction from the beginning after ROLLBACK

DB2 Deadlock Detector

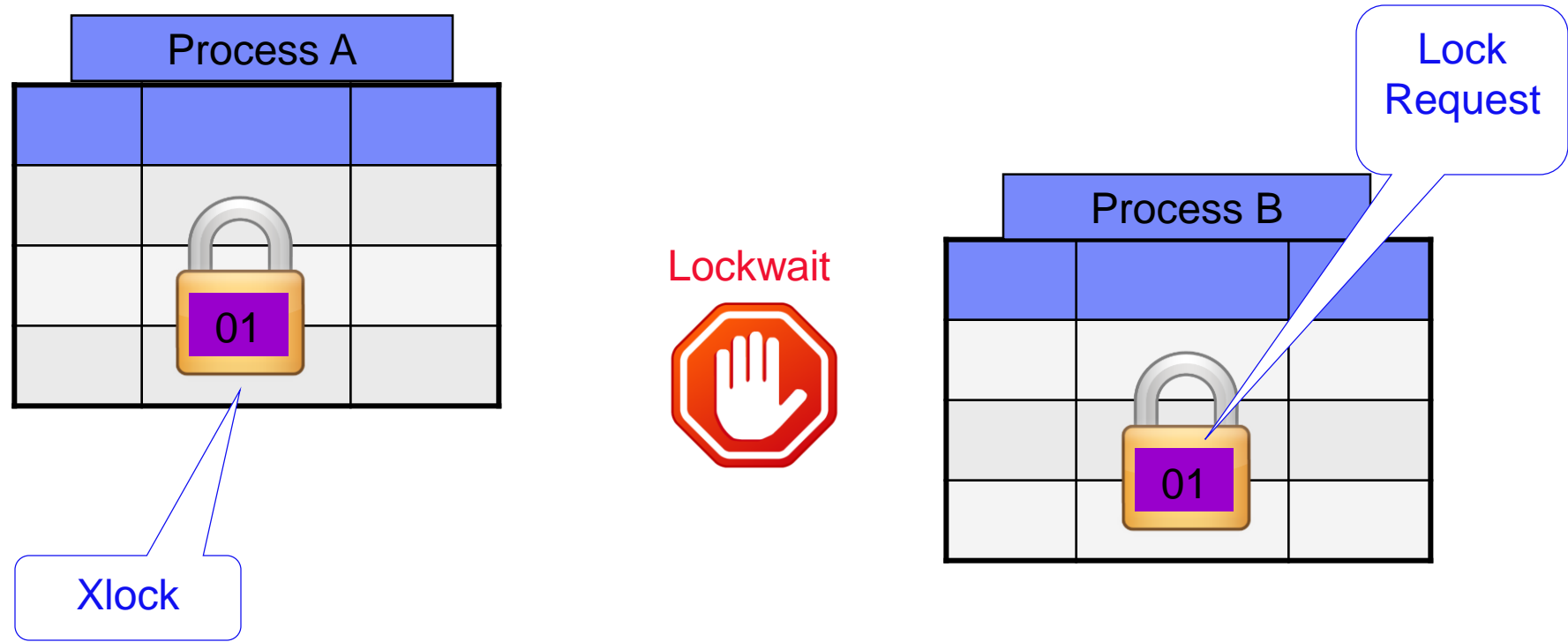
▪ **Responsible for resolving deadlocks**

- When a deadlock is detected, the deadlock detector will choose a victim that will be automatically rolled back
 - Rolling back the victim causes the lock conflict to be removed, and the other application can continue processing
 - If it finds a deadlock, the deadlock detector arbitrarily selects one deadlocked process as the victim process to roll back
 - The victim process is awakened, and returns SQLCODE -911 (SQLSTATE 40001), with reason code 2, to the calling application

▪ **DB CFG parameter DLCHKTIME**

- Defines frequency with which 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 CURRENT LOCK TIMEOUT

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

```

        .-CURRENT-.                .-=-.
>>-SET--+-----+--LOCK TIME OUT--+---+----->
>--+--WAIT-----+-----><
+-NOT WAIT -----+
+-NULL-----+
| .-WAIT-.      |
+-+-----+--integer-constant---+
'-host-variable-----'
    
```

Examples

Example 1: Set the lock timeout value to wait for 30 seconds before returning an error

```
SET CURRENT LOCK TIMEOUT 30
```

Example 2: Unset the lock timeout value, so that the locktimeout database configuration parameter value will be used instead

```
SET CURRENT LOCK TIMEOUT NULL
```

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 is exhausted
- **The database manager will attempt to obtain table locks and release the existing row locks**
 - The desired effect is to make more lock memory available for other applications
- **The following database parameters have a direct effect on lock escalation:**
 - LOCKLIST - total number of 4k pages allocated for lock storage
 - 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**

Monitoring/Identifying Locking Issues in General

▪ **Key indicator monitoring elements:**

- lock_timeouts or lock_wait_time values are increasing
- int_rollbacks value is increasing
- int_deadlock_rollbacks shows increase in number of rollbacks due to deadlock events
- Check monitoring element lock_escals for indications that lock escalations may be a contributing factor

▪ **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
 - MON_GET_LOCKS and MON_GET_APPL_LOCKWAIT table functions
- Application and database lock snapshots
- Lock and deadlock event monitors
- db2pd

▪ **Check Administration Notification Log for:**

- Lock escalations
- Lock waits

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 rows that are touched, the more locks that 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

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

- 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

- **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:

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!

Considerations for CUR_COMMIT

- **For new databases, the default is set to ON**
 - When the default is set to ON your query will return the currently committed value of the data at the time when your query is submitted
- During database upgrade from V9.5 or earlier, 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 Scenarios
- Using Isolation Levels
- **Monitoring Locking Issues**
- Avoiding Locking Scenarios

Monitoring Locking Issues in General

■ Monitoring:

- Create/configure/enable locking event monitors to capture details on lock event data for a workload or database
 - Find the application that is waiting, and information on the lock requested
- Use `db2pd -locks (wait)`
- Review DB2 administration notification log (`<instance>.nfy`) with `DIAGLEVEL` set to 3 or 4 (waits), or DB2 diagnosis log (`<instance>.diag.log`)

NEW IN
DB2 10

New event monitor features:

- All event monitors now support the `WRITE TO TABLE` target
- Existing event monitors that write to tables can be altered to capture additional logical data groups

Administrative View – SYSIBMADM.SNAPDB

```
■ SYSCRMADM.SNAPDB contains relevant information about locks  
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.MON_LOCKWAITS

```
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.MON_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.

Lock Mode

Lock Mode Requested

SYSIBMADM.LOCKWAITS administrative view is deprecated in DB2 10. Use new view like SYSIBMADM.MON_LOCKWAITS

Table Functions – MON_GET_LOCKS and MON_GET_APPL_LOCKWAIT

- Used to investigate locking problems in the current connected database

1. Call the MON_GET_APPL_LOCKWAIT table function to determine all the locks that are waiting to be acquired in the database

```
SELECT lock_name, hld_member, lock_status, hld_application_handle
FROM TABLE (MON_GET_APPL_LOCKWAIT(NULL, -2))
```

- This query returns the following output:

LOCK_NAME	HLD_MEMBER	LOCK_STATUS	HLD_APPLICATION_HANDLE
00030005000000000280000452	-2	W	
00030005000000000280000452	-2	W	
00030005000000000280000452	-2	W	

3 record(s) selected.

Lock status
is waiting

- A HLD_MEMBER value of **-2** indicates that the lock **0x00030005000000000280000452** is being held at a remote member

MON_GET_LOCKS and MON_GET_APPL_LOCKWAIT (Cont.)

- Call the MON_GET_LOCKS table function to determine the holder of the lock, by specifying the lock name, 0x00030005000000000280000452, as the search argument:

```
SELECT lock_name, member, lock_status, application_handle
FROM TABLE (MON_GET_LOCKS
( CLOB('<lock_name>00030005000000000280000452</lock_name>'), -2))
```

- This query returns the following output:

LOCK_NAME	MEMBER	LOCK_STATUS	APPLICATION_HANDLE
00030005000000000280000452	0	W	12562
00030005000000000280000452	1	W	12562
00030005000000000280000452	2	G	65545
00030005000000000280000452	3	W	12562

4 record(s) selected.

Lock status
is Granted

The App "12562" is waiting to
obtain a lock that the App
"65545" has

Monitoring db2diag.log for Lock 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        DR    : 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
```

Lock Level

Lock Mode
(exclusive)

Lock Status
"Waiting"

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, sqldEscalateLocks, probe:3
MESSAGE  : ADM5502W The escalation of "240671" locks on table
"DB2ADMIN.EMPLOYEE"
to lock intent "X" was successful.
```

- Additional tools
 - IBM Data Studio & IBM InfoSphere Optim Performance Manager
 - Snapshot Monitor (Database and Application level) and Event Monitor (Database and Connection type) => "Lock escalations" and "Exclusive lock escalations" (`x_lock_escals`) info
 - Many others DB2 interfaces provide info on "Lock escalations" (`lock_escals`)

In general Control Center and related components as Health Center have been discontinued in DB2 10. These have been replaced by a new set of GUI tools: IBM Data Studio and IBM InfoSphere Optim tools

Agenda

- Locking and Performance
- Identifying Locking Scenarios
- 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, or using STMM
- Avoid lock timeouts:
 - Adjust the DB CFG parameter LOCKTIMEOUT or use the SET CURRENT LOCK TIMEOUT command
- Use CURRENTLY COMMITTED
- If not using CURRENTLY COMMITTED, avoid deadlocks by:
 - Reducing row blocking during index and table scans (CS and RS only):
 - 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**

–If set to **ON** or **TRANSACTION**, this variable allows the DB2 database system to maintain the table lock when an Uncommitted Read or Cursor Stability isolation level is closed. The table lock that is kept is released at the end of the transaction. If set to **CONNECTION**, a table lock is released for an application until the application either rolls back the transaction or the connection is reset

▪ **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

Random ordering for Index columns: New in 10.5

- Lessens index page contention
- When rows are added to a table in Index sequence contention on the leaf page might occur
- For example keys generated using identity column or sequences or timestamps
- Most beneficial in pureScale environments

Summary

- Locking and concurrency issues can have a significant impact on the performance of a DB2 application
- It is necessary to collect information that would help to identify what type of lock event is involved
- Commit DML (insert, update, delete) and DDL actions as soon as possible
- Avoid concurrent DDL operations if possible
- Avoid using higher isolation levels than necessary
- Use DB2 options for monitoring locking



议程

- 数据库性能问题原因
- DB2数据库监控手段
 - 事件监控
 - 监控快照
 - 监控函数与视图
- DB2 SQL监控与调优
- DB2 Lock监控与调控机制
- DB2 Log监控与调控机制
- DB2 监控常用工具

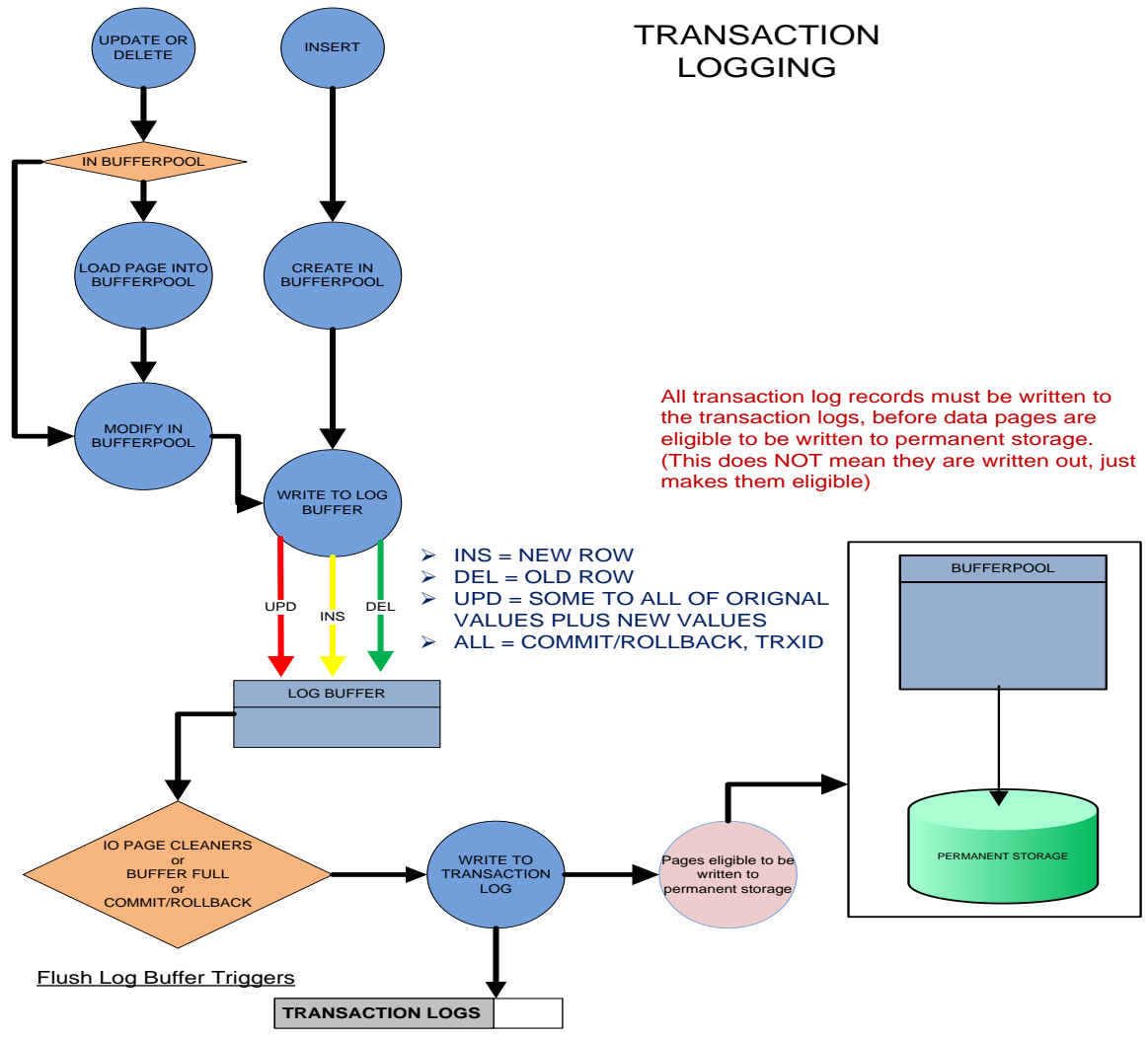
DB2 Performance Clinic Modules - Agenda

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

Agenda

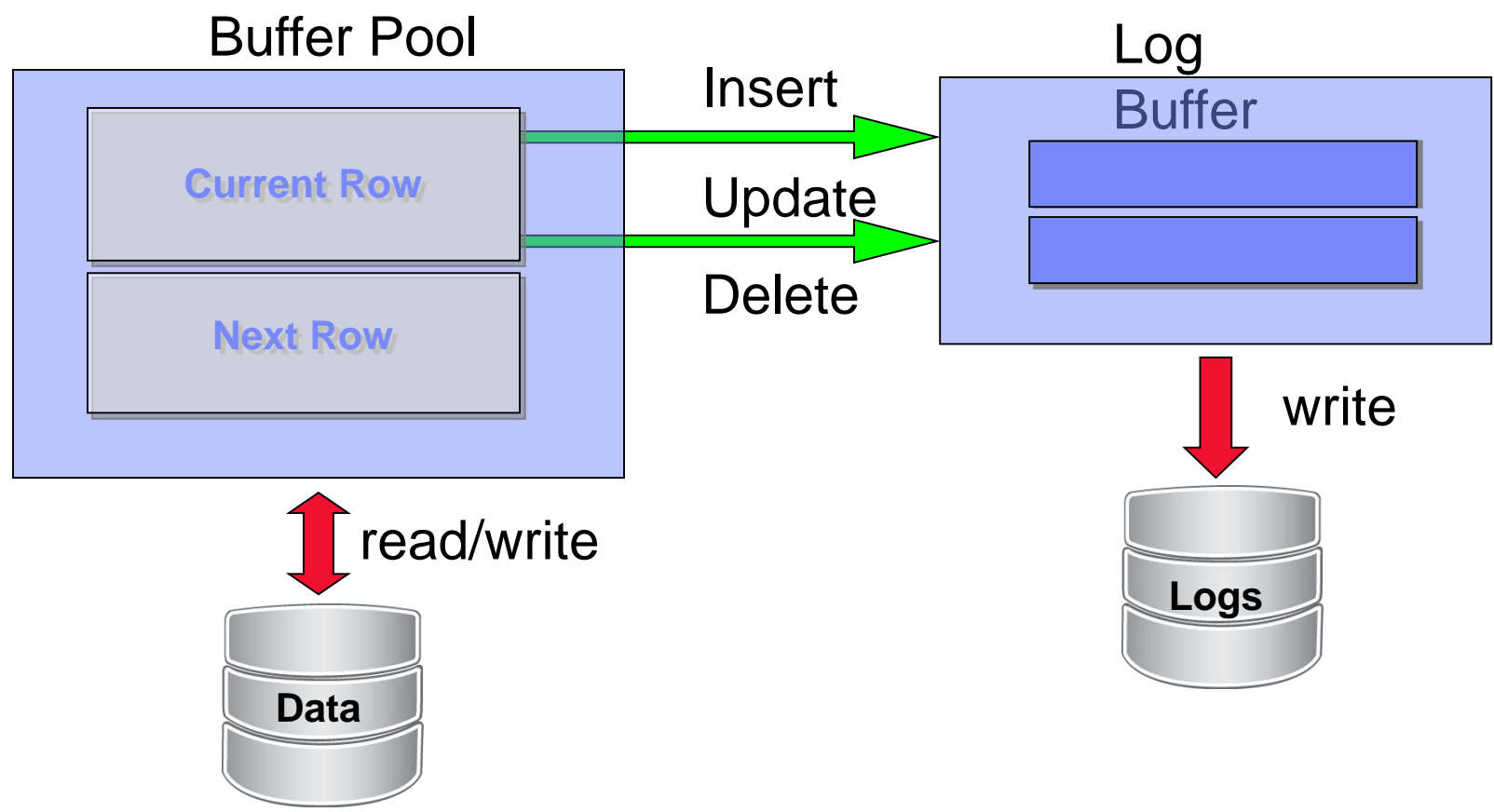
- **Logging Concepts**
- **Configuration and Performance**
- **Logging Bottlenecks**
- **Reducing Logging**
- **Monitoring and Tuning**

Transaction Logging Overview



Transaction Logging Overview

The process of recording changes to database objects and data



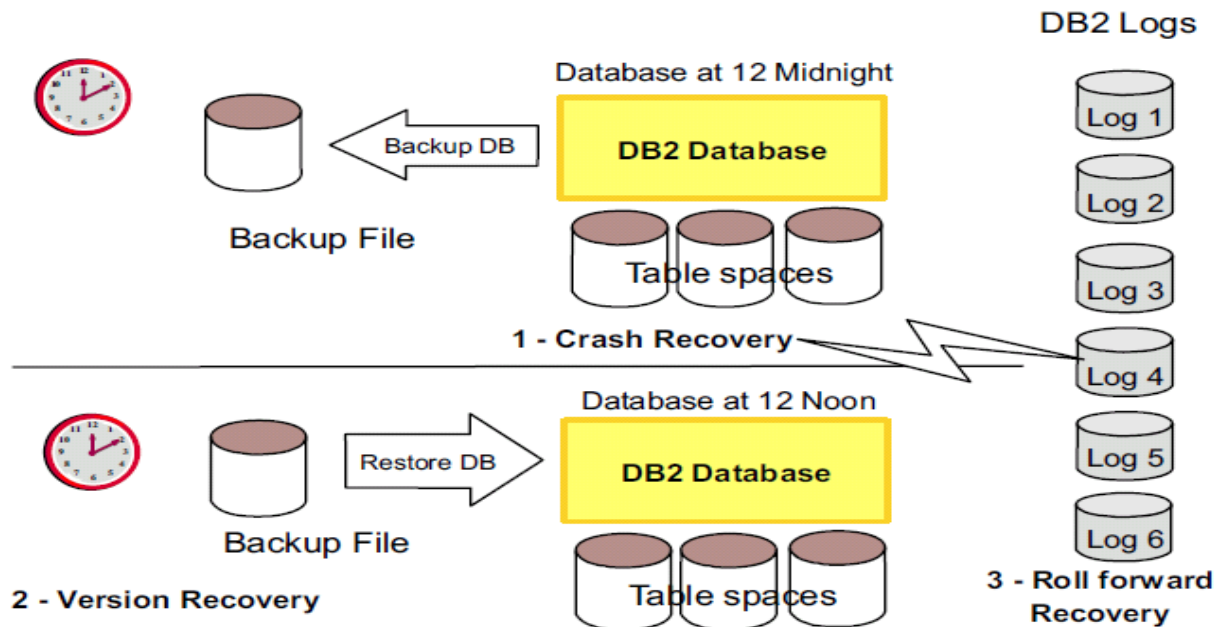
DB2 Logging Concepts

- **Records database transactions**
 - If there is a crash, the Logs are used to undo or redo 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 or not the 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**
 - Circular logging (default)
 - Archive logging



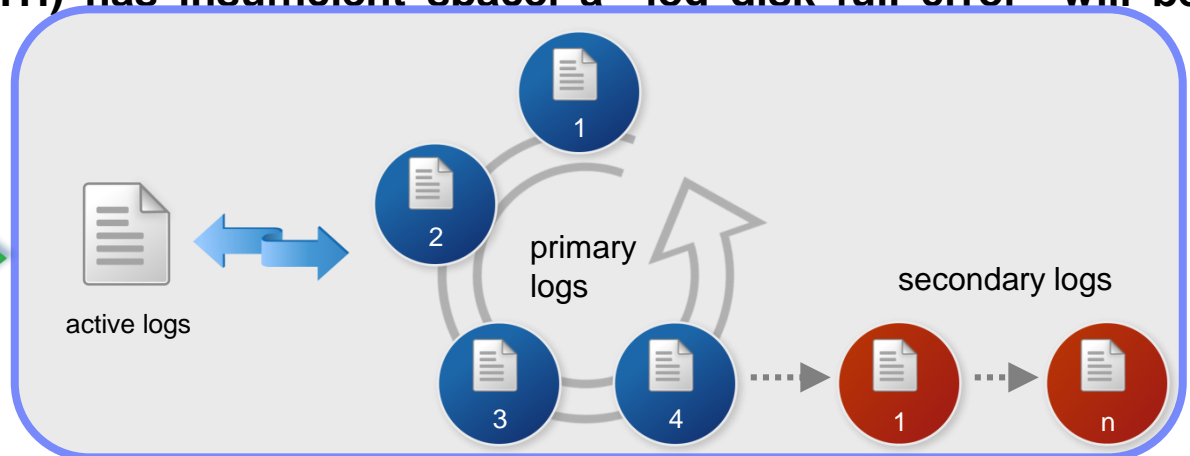
DB2 Logging and Recovery Methods

- **Database Crash Recovery (DB2 Logs)**
 - Recovery from unscheduled outages
 - Uses database logs to Undo or Redo changes
- **Version Recovery (DB2 Backup Image)**
 - Recovery of a database to a previous state using a DB2 backup
- **Roll forward Recovery (DB2 Backup Image + DB2 Logs)**
 - Recovery of database or table space changes using a DB2 backup image and then applying the DB2 logs using roll forward



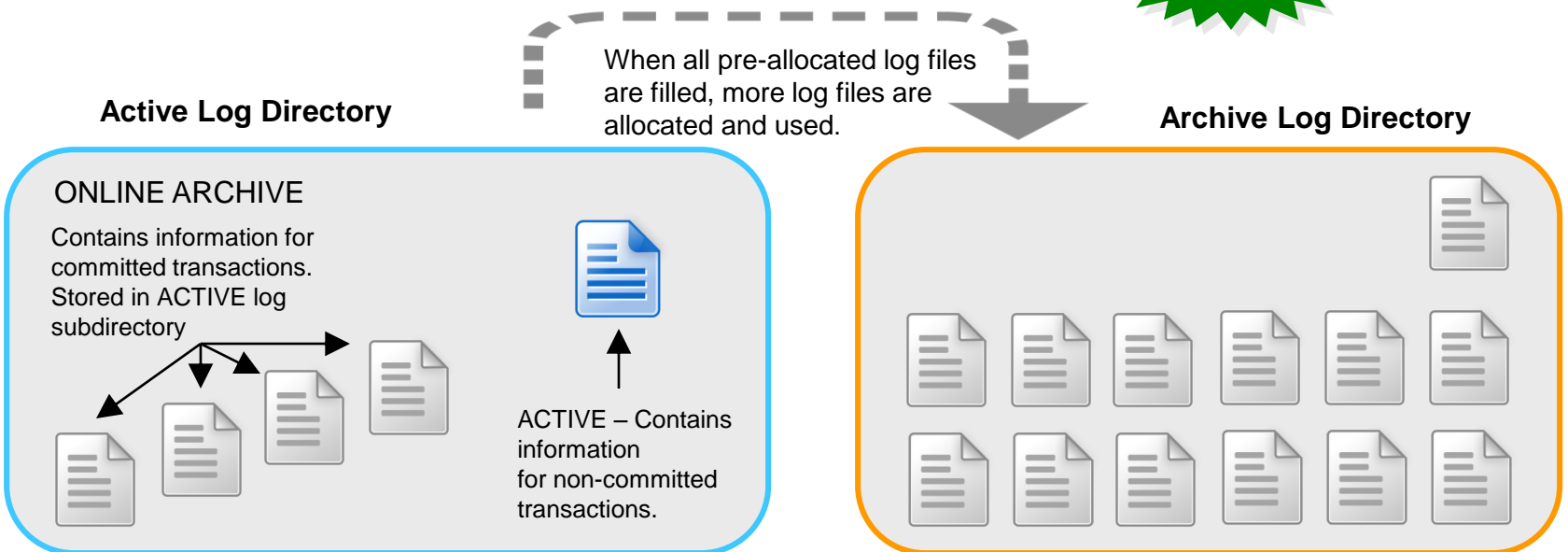
Circular Logging

- **Primary log files (LOGPRIMARY)** are used to record all changes and reused when changes are committed
 - Crash and version recovery possible; roll-forward recovery not possible
 - Only full, offline database backups are allowed
- **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**



Archive Logging

- **Maintain a history of log files**
 - Enable with **LOGARCHMETH1** DB configuration parameter
 - **LOGRETAIN** and **USEREXIT** have been discontinued in DB2 10. They have been replaced with **LOGARCHMETH1**
 - Allows **roll-forward recovery** or **online backup**
- **Logs can be archived externally when no longer active to avoid exhaustion of log directory**
- **As of DB2 10, archived log files can be compressed**



Infinite Logging

- Issue with limited number of logs
 - A long running transaction can exhaust logs allocation, even after secondary log files are allocated
 - The number of primary and secondary log files must comply:
 - If logsecond has a value of -1, logprimary \leq **256**
 - If logsecond does not have a value of -1, (logprimary + logsecond) \leq **256**

- Solution: **Infinite Logging**
- No limit on the size or the number of **in-flight transactions running**
- Enabled by setting logsecond to -1
- Database must be configured to use **archive logging**
 - Can hinder performance for rollback and crash recovery
- Other control parameters
 - num_log_span: number of log files an active transaction can span
 - max_log: percentage of the primary log space that a transaction can consume

Log control files

- **Used to determine which records from the log files need to be applied to the DB when it restarts after a failure**
- **Redundancy for database resilience for protection**
 - Two copies of the each member's log control file, SQLOGCTL.LFH.1 and SQLOGCTL.LFH.2
 - Two copies of the global log control file, SQLOGCTL.GLFH.1 and SQLOGCTL.GLFH.2
- **Performance considerations**
 - Applying the transaction information contained in the log control files contributes to the overhead of restarting a database after a failure
 - Use the **softmax** parameter to configure the frequency at which the database manager writes the buffer pool pages to disk

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

LOGARCHCOMPR1 and

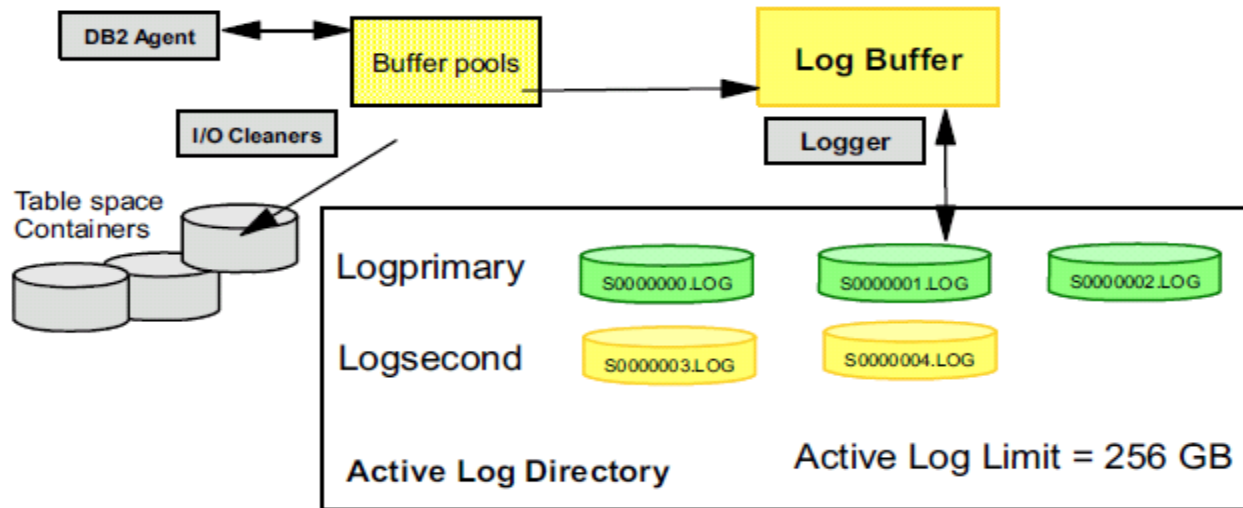
LOGARCHCOMPR2

**LOGPATH
NEWLOGPATH
LOGPRIMARY
LOGSECONDARY
LOGBUFSZ
LOGFILSIZ
MINCOMMIT**



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



	Default	Minimum	Maximum
Logprimary	3	2	256 - Logsecond
Logsecond	2	0	256 - Logprimary

Primary Logs (LOGPRIMARY)

▪ Characteristics

- This parameter specifies the number of primary logs of size *logfilsiz* that will be created
- 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

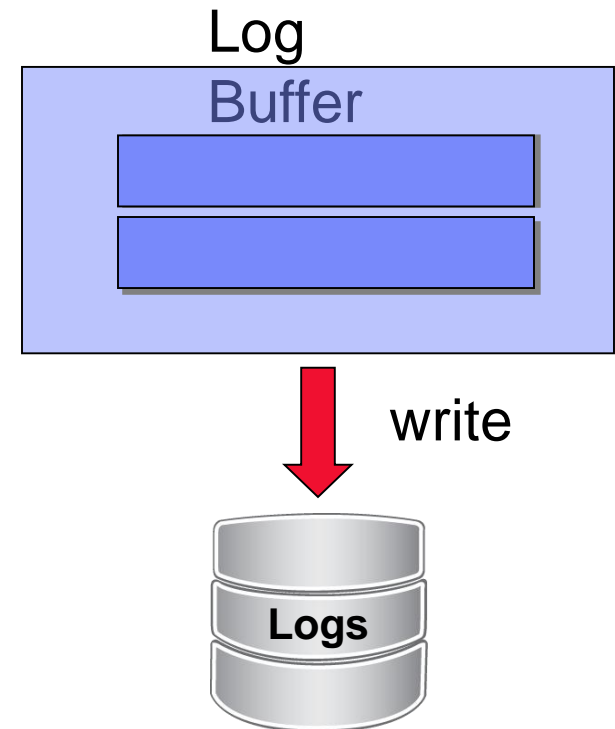
- DB CFG parameter defines the size of each primary and secondary log file in 4K pages

▪ 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

When Are Log Records Written to Disk?

- **Transaction log records are written from log buffer to log files**
- **Transaction committed**
 - application commits
 - 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**



MINCOMMIT This parameter is deprecated in Version 10

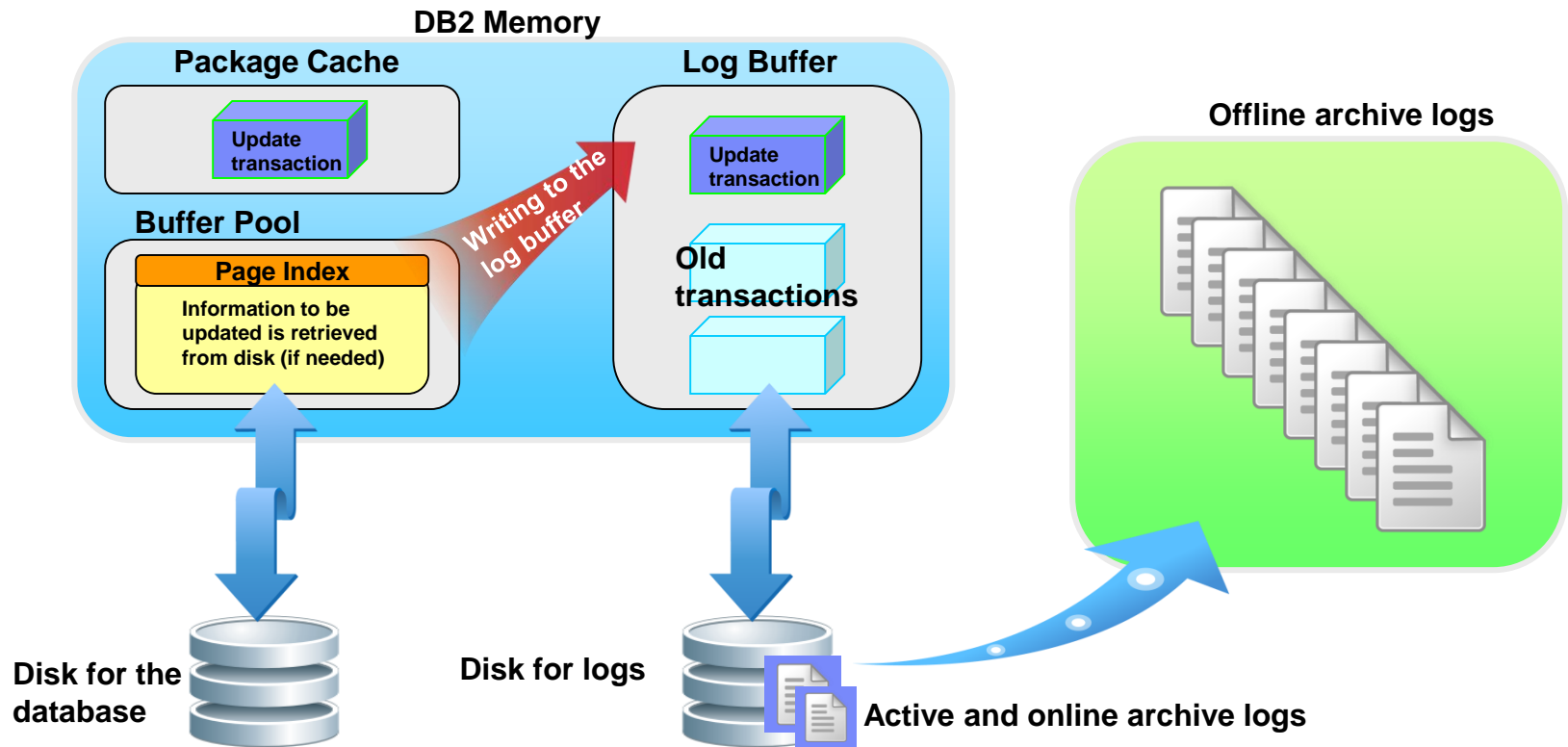
Log Buffer Size (LOGBUFSZ)

- 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)
- Default value usually not large enough for OLTP databases (8 4K pages), 256 (4K pages) is a good starting point:

```
db2 update db cfg for sample using LOGBUFSZ  
256
```

Log File States

- Active logs
 - Contain at least 1 transaction that has not been committed or rolled back
- Online archive logs
 - Contain committed and externalized transactions in the active log directory
- Offline archive logs
 - Contain committed and externalized transactions in a separate repository



DB2CKLOG - DB2 CHECK LOG

- Check the validity of archive log files

 - Determine whether the log files can be used during roll-forward recovery

 - A single archive log file or a range of archive log files can be checked

```
DB2CKLOG log_num ARCHLOGPATH path  
DB2CKLOG log_num to log_num2
```

Validating a range of logs:

```
$ db2cklog 3 to 5  
  
_____ D B 2 C K L O G _____  
DB2 Check Log File tool  
  
...  
"db2cklog": Finished processing log file "S0000003.LOG". Return code: "0".  
...  
"db2cklog": Finished processing log file "S0000004.LOG". Return code: "0".  
...  
"db2cklog": Finished processing log file "S0000005.LOG". Return code: "0".
```

Successful Validation

New Log Path (NEWLOGPATH)

- DB CFG parameter which is used to specify a new location for 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

```
db2 update db cfg for sample using NEWLOGPATH /db2logs
```

General Recommendations

- Location of Database logs
 - Need to be on their own physical disk
 - A fast I/O device for the log is recommended
 - RAID 10 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
- 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?
- 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
- 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.
 - Compressed when using compression which can help reduce I/O traffic
 - If DGTT/CGTTs are being used set NOT LOGGED
 - Larger I/Os can also be due to a poorly performing logging disk

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 Solution**

- Can you reduce commit frequency?

- Database snapshot to verify if 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.

Agenda

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

Identifying The Log Files Location

1. Determine the file systems that reside on the system

```
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/SQLOGDIR/
```

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 database 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      = 9
DDL statements executed               = 0
Inactive stmt history memory usage   = 0
```

How many
Commits

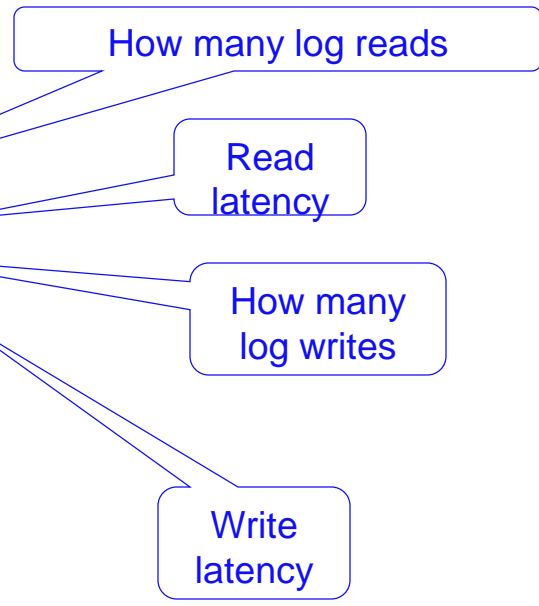
How many
Dynamic
statements

- **GET SNAPSHOT FOR database ON sample**
- **Locate Log section with Commits/Rollback**
- **Reference Commit, Rollback, Dynamic, Static, etc.**
- **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
```



- **Locate log section**
- **GET SNAPSHOT FOR DATABASE ON**
- **Reference log reads and writes**
- **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 and Tuning**
- **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 (DGTs)
 - 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
- 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



Database logging

- Rules of thumb
 - Use archive logging in production environments to be able to perform many recovery operations including, online backup, incremental backup, online restore, point-in-time rollforward, and issuing the **RECOVER DATABASE** command
 - Configure secondary log files to provide additional log space on a temporary basis
 - Compress archive logs
 - Consider the I/O adapter or bus bandwidth requirements for transaction logging



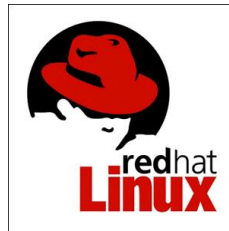
议程

- 数据库性能问题原因
- DB2数据库监控手段
 - 事件监控
 - 监控快照
 - 监控函数与视图
- DB2 SQL监控与调优
- DB2 Lock监控与调控机制
- DB2 Log监控与调控机制
- DB2 监控常用工具

DB2 Monitoring Tools – DB2TOP

- **DB2TOP**

- Provides a unified, single-system view of a multi-partition database or single-partition database on the AIX®, Linux, HP-UX, and Solaris operating systems
- Can be run in interactive mode or in batch mode



DB2 Monitoring Tools – DB2TOP

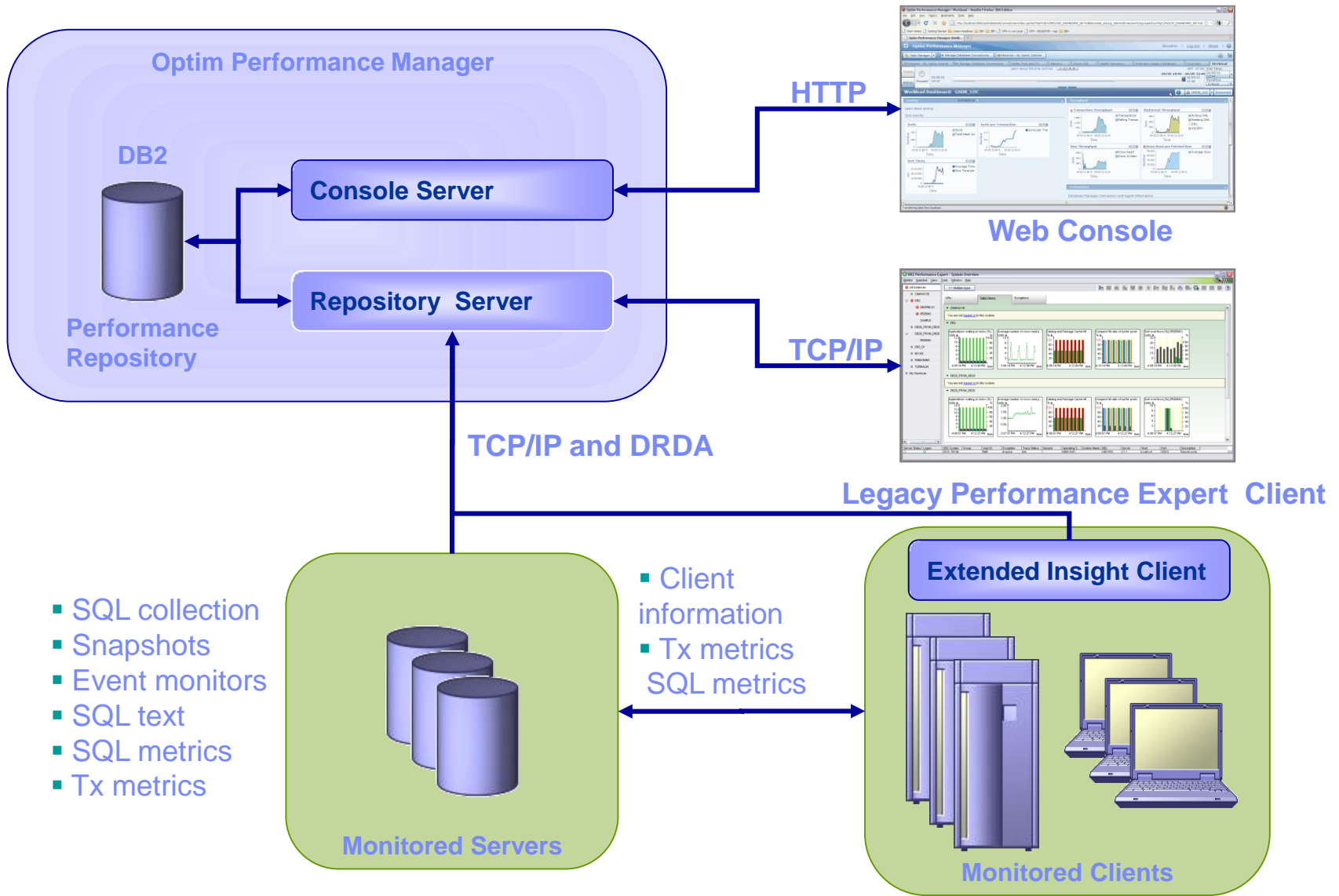
- **View delta or cumulative snapshot counters**
- **Monitor interactively or collect data for analysis on:**
 - Database (d)
 - Tablespace (t)
 - Dynamic SQL (D)
 - Sessions (l)
 - Bufferpool (b)
 - Lock (U)
 - Table (T)
 - Bottlenecks (B)



DB2 Monitoring Tools – IBM Optim Performance Manager

- This tool has a web-based interface to view system health at any time from any location
- It will help you prevent problems by monitoring performance indicators for emergent problems with easy to understand dashboards
- It enables DBA to plan for business growth and also access historical data to generate key reports
- It has out of the box DB2 and application monitoring for:
 - SAP™
 - Cognos™
 - DataStage®
 - Java® (WebSphere®)
 - CLI applications
- With this tool, identify, diagnose and solve problems quickly, pinpointing them in minutes instead of days

Architecture Overview (OPM 5.1)



Monitoring And Optimizing Performance – OPM

Get early warning of potential problems



Diagnose database problems with resource specific dashboard



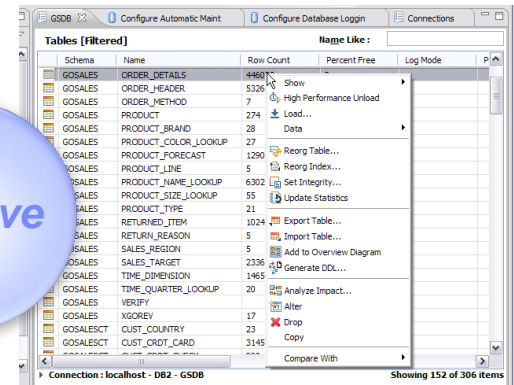
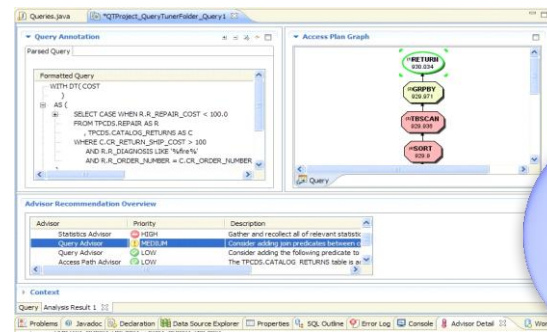
InfoSphere Optim Performance Manager
InfoSphere Optim Configuration Manager



Resolve query problems

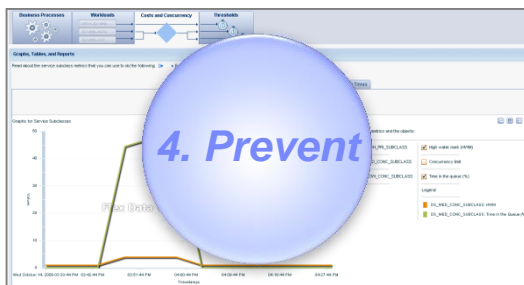
Resolve database problems

Prevent problems



InfoSphere Optim Query Workload
Tuner
InfoSphere Optim pureQuery
Runtime

Data Studio



DB2 Workload Manager

Database Overview Dashboard At A Glance

InfoSphere Optim Performance Manager | bmb | Log Out | Help

Open | Databases | Overview

View: Historical Data | End Time: 12/16/11 17:32 | Duration: 1 Hour | Automate

Learn about the time controls.

Europe/Berlin
12/16/11 16:32 - 12/16/11 17:32

Aggregation level: 1

Overview Dashboard: sample | sample | Disconnect

Workload

- Transactions: 98.367 /min
- Failing transactions: 0.00 %
- Open connections: 8
- Active connections: 1
- Rows read per second: 2,277

Sorting

- Active sorts: 0
- Sorts: 26.550 /min
- Sort overflows: 0.00 %
- Post threshold sorts: 0.00 %
- Sort time per minute: --
- Average sort time: --
- Average sorts per transaction: --
- Sort memory in use: --

Locking

- Currently waiting applications: 0.00 %
- Longest wait time: --
- Average lock wait time per: 0.000000

Caching

- Catalog cache hit ratio: 97.18 %
- Package cache hit ratio: 34.97 %

Utilities

- Active utilities: --

Logging

- Log space used: 11.39 MB
- Log space used: 2.93 %
- Log write rate: 10.247 KB/s
- Log to be read for recovery: --
- Maximum indoubt transactions: --

System

- HADR role: --
- HADR state: --
- HADR connection status: --
- HADR connection time: --
- HADR log gap: --

Select time period

KPI violations identified

Thumbnail overview, click to enlarge

Visualize threshold values in context

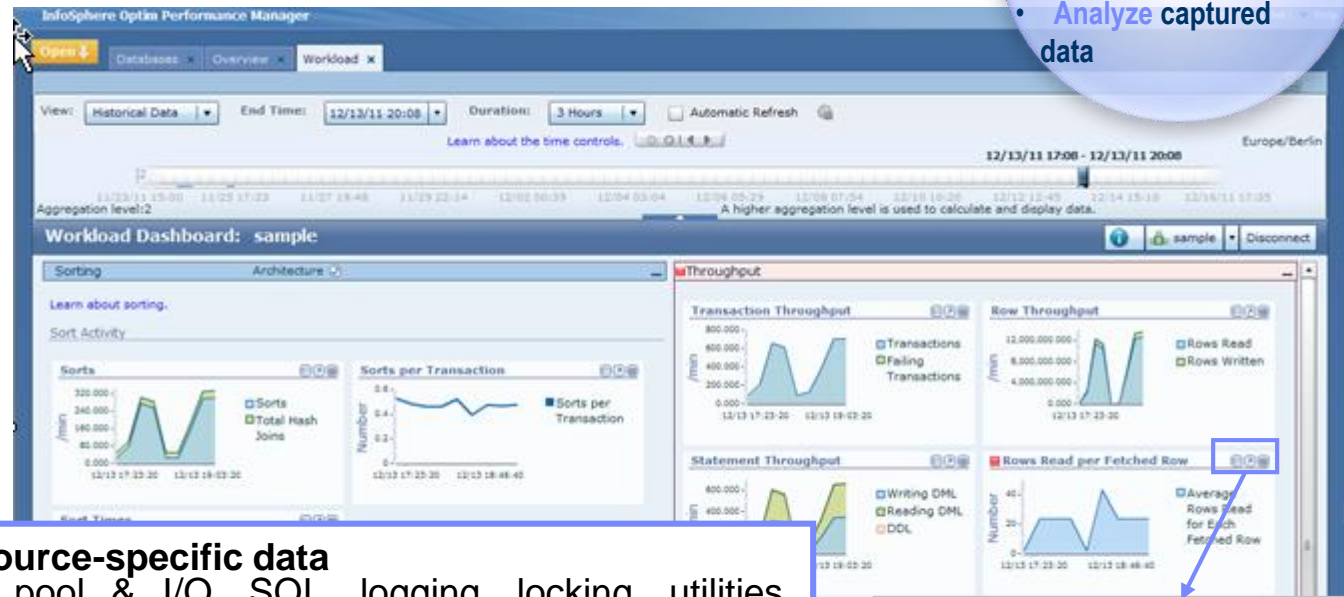
- ### OPM Overview dashboard
- Key-performance-indicators (KPIs) will tell you what's going on
 - Alerts for critical areas will tell you immediately if something is critical and needs further attention
 - Real time or any time views with automated aggregation and retention management
 - Historical information lets you go back in time and see how a problem arose, mitigate problems first and do causal analysis later, compare to prior time periods

Guided Problem Solving Approach

Drilldown to diagnose alerts to find the problem

Diagnose

- Drill down into problem detail and related content
- Analyze captured data



- **Drill down into resource-specific data**
 - Memory, buffer pool & I/O, SQL, logging, locking, utilities, system, workload
 - List, filter, sort, and report within category
 - View by partition or member
- **View real time to any time metrics**
 - What happened while away, mitigate symptoms first, compare to historical values, capture intermittent problems
 - Automatic aggregation and retention management
- **Browser-based any where, any time access**

Rows Read per Fetched Row

Time	Average Rows Read for Each Fetched Row (Number)
12/13 19:15:00	23.777
12/13 19:00:00	23.767
12/13 18:45:00	23.801
12/13 18:30:00	43.415
12/13 18:15:00	2.023
12/13 18:00:00	23.735
12/13 17:45:00	23.797
12/13 17:30:00	23.726
12/13 17:15:00	2.072

Analyze The Lock Conflict

View

- Lock holder
- Lock waiters
- Locked object
- Application details
- SQL statement

Take action

- Force application

Locking Information for Client application names

Locking Event (0) | **Current Waiting Connections (1)** | **Current Blocking Connections (1)**

The grid shows applications that are blocking other applications for the selected workload cluster that the application is running.

Application Name	Application ID	Block Time	Connection Start
db2jcc_application	9.30.249.104.53001....	04:07.896	05/16 17:55:38

Analyze...

Analyze Locking Situations

Each complete set of entries in the tree includes an application that is holding a lock and the applications that are waiting because of that lock. The entries between the main entry and the leaf entry are applications that are blocking and waiting. Each leaf entry is an application that is only waiting.

- db2jcc_application
 - db2jcc_application

Details about the locked object

Table Name: PRODUCT_NAME_LOOKUP
 Table Schema Name: GOSALES
 Table Space Name: GOSALES_TS
 Lock Type: X
 Lock Mode: Exclusive Lock
 Lock Object Type: Table Row Lock
 Lock Wait Time: 0 sec
 Sequence Number: 00001
 Lock Mode Requested: No Lock
 Lock Type Requested: X

Details about the application

Application Mode: Exclusive Lock
 Application Name: db2jcc_application
 Agent ID: 30808
 Application ID: 9.30.249.104.53001.1105170...
 Authentication ID: DB2INST2
 Client User ID: --
 Client Application Name: Sales Order App 1
 Client Workstation Name: asimvsingh-svl
 Application Status: UOW waiting

[Force Application](#)

Details about the current activity

```
SELECT * FROM GOSALES.PRODUCT_NAME_LOOKUP
```

Rows Read: 5432 | [Stop Current Statement...](#) | [Show All Text](#) | [Tune](#)
 Rows Written: 0
 Statement Elapsed Time: 0 sec | [Go to the Active SQL Dashboard](#)
 Statement Start Time: 05/16 18:00:04

Details about all activities

Connection Request Completion	05/16 17:55:38	Rows Written:	2332
Timestamp:		Locks Held:	12
User CPU Time:	17555	Lock Waits:	0
System CPU Time:	0	Time Application Waited on	0 sec
Commits:	0	Locks:	
Rollbacks:	0	Time Waited on Locks per	0 sec
Dynamic SQL Attempted:	67155	Second:	
Static SQL Attempted:	0	Average Wait Time per Lock:	--

Analyze Connections

InfoSphere Optim Performance Manager bmb | Log Out | Help

Open ▾ Databases x Connection x

View: Historical Data ▾ End Time: 11/21/11 18:17 ▾ Duration: 1 Hour ▾ Automatic Refresh

Connection Dashboard: sample Server Status ⓘ sample ▾ Disconnect

Show Highest 100 ▾ by CPU time: ▾ Choose Columns Show Connection by Members Show Executed SQL Force Filter

Application Name	Application Handle	Session User	Latest Connection Status	Connection Established	Connect Closed	Latest Transaction Log Space Used	Lock Wait Time	Activity time:	CPU time:	Other time:
db2jcc_application	23,339	BMB	EXECUTING...	11/21 16:19:09		0	0.000000	02:59.577000	22.078000	
db2jcc_application	23,343	BMB	IDLE UOW...	11/21 16:19:09		0	0.000000	19.461000	4.235000	
db2jcc_application	23,884	BMB	IDLE UOW...	11/21 16:36:26		0	0.000000	6.877000	1.219000	
db2jcc_application	25,309	BMB	CLOSED	11/21 17:20:06	11/21 17:42:07	136	0	3.349000	0.328000	
db2jcc_application	26,193	BMB	CLO			42	0	1.410000	0.140000	
db2jcc_application	26,486	BMB	CLO			41	0	1.469000	0.093000	
db2jcc_application	26,004	BMB	CLO			21	0			
db2jcc_application	26,902	BMB	IDLE			12	0			
db2jcc_application	26,790	BMB	CLO			15	0			
db2jcc_application	25,116	BMB	CLO			14	0			
db2jcc_application	25,961	BMB	CLOSED	11/21 17:43:06	11/21 17:43:07	4	0	0.036000	0.000000	

Filter and action controls

Display details of your top connections

Launch a connection report

Connection Details

Overview Server Times I/O Rows and Transactions Locking and Communication Utilities

Server Times

Server Time Distribution

- CPU Time: 19.16%
- Wait Time: 80.84%

Server Wait Time

- I/O: 0.14%
- Lock Wait: 99.86%
- Log Wait
- Transaction End Wait
- Statement

Server Processing Time

- Transaction End: 83.97%
- Processing: 0.16%
- Sort: 0.03%
- Processing: 15.85%
- Statement

Server Elapsed Time

- Compilation: 26.95%
- WLM Queue: 73.05%
- Utilities
- Statement Activity

- List connections
- Real time and history
- Partition and member views
- Drill down into details of selected
- Generate in context report
- Force connection

Analyze I/O

Buffer Pool and I/O Dashboard: DEMO@local

Buffer Pools Table Spaces Tables

Show Lowest 5 buffer pools by Hit Ratio (%) Show Contained Objects Change Configuration...

Buffer Pool Name	Main Usage	Buffer Pool Size (pages)	Hit Ratio (%)	Logical Reads (/min)	Physical Reads (/min)	Physical Writes (/min)	Updates per Read	Avg Page Read Time (sec)	Avg Page Write Time (sec)	Prefetcher Hit Ratio (%)	Async Read Ratio (%)	Async Write Ratio (%)
Total	N/P	4,919	99.988	452,801.729	52.088	140.123	0	0.008	0.003	100	23.395	13.565
IBMDEFAULTBP	MIXED	4,855	99.988	452,801.729	52.088	140.123	0	0.008	0.003	100	23.395	13.565
IBMSYSTEMBP16K	MIXED	16	N/P	0	0	0	N/P	N/P	N/P	N/P	N/P	N/P
IBMSYSTEMBP8K	MIXED	16	N/P	0	0	0	N/P	N/P	N/P	N/P	N/P	N/P
IBMSYSTEMBP4K	MIXED	16	N/P	0	0	0	N/P	N/P	N/P	N/P	N/P	N/P

Detailed Information for IBMDEFAULTBP

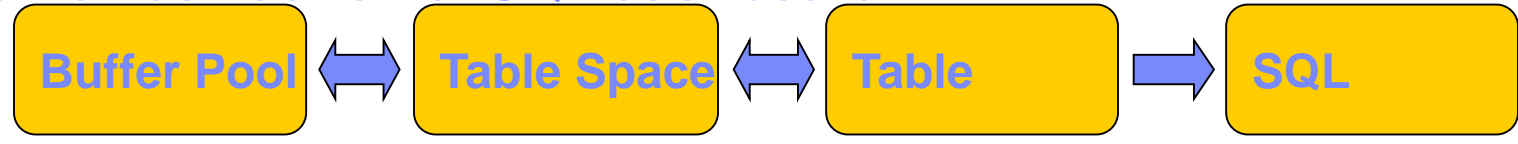
General Prefetchers Utilization and Health

Buffer pool name: IBMDEFAULTBP Page size: 4,096 KB Main usage: MIXED Buffer pool size: 4,855 pages

Number of I/O servers: 3 Number of prefetch requests: 3.252 /min Prefetcher hit ratio: 100 % Asynchronous read ratio: 23.395 %

Buffer Pool Size Buffer Pool Hit Ratio

Double clicking or using the 'Show Contained Objects' button lets you drill down into contained objects. 'Show SQL' for contextual launch of SQL dashboard



- Check buffer pools, table spaces, and tables
 - Identify hot objects that need dedicated buffer pools
 - Check whether buffer pools are appropriately sized
 - Check the disk space and container definition of table spaces
 - See what SQL is accessing a table

From A Table To The SQL Using The Table

Buffer Pool and I/O Dashboard: sample

Buffer Pools | Table Spaces | **Tables**

Show Highest 5 tables by Rows Read in table space: All Show SQL Show Objects by Members Change Configuration...

Table Name	Table Schema	Data Table Space Name	Data Object Physical Size (bytes)	Index Object Logical Size (bytes)	XML Object Physical Size (bytes)	LOB object Physical Size (bytes)	Long Object Physical Size (bytes)	Latest Size (bytes)	Table Scans	Rows Read	Rows Inserted	Rows Updated	Rows Deleted
Total	--	--	--	--	--	--	--	0	5,075	105,799,932	301,337	7,053,718	294,000
OPM_SAMPLE_TABLE	DB2USER1	SYSTOOLS...	794,624	1,163,264	0	0	0	1,957,888	4,997	105,773,630	294,000	7,053,718	294,000
SYSSEQUENCES	SYSIBM	TEMPSPACE1	8,192	32,768	0	262,144	0	303,104	0	14,708	0	0	0
OPMUQZDD5N2	OPM	IBMDB2SA...	8,192	0	0	8,192	0	16,384	6	2,696	2,544	0	0
OPMUQZDD5N1	OPM	IBMDB2SA...	450,560	0	0	8,192	0	458,752	5	2,648	3,134	0	0
SYSPLAN	SYSIBM	TEMPSPACE1	180,224	131,072	0	2,359,296	0	2,670,592	0	2,047	0	0	0

Detailed Information for OPM_SAMPLE_TABLE

SQL Statements Dashboard: sample

Learn about tuning SQL statements, stopping SQL statements, and forcing applications.

Top Individual Executions Execution Summa

Show Highest 20 by Total Execution Elapsed Time Choose Columns Show Statement by Members Tune All Filter

AND Statement text contains OPM_SAMPLE_TABLE ; Clear Filter

Statement Text	Statement Category	Execution Elapsed Time	Number of Executions	CPU Time	Rows Read
INSERT INTO DB2USER1.OPM_SAMPLE_TABLE (SSN,FIRST_NAME, LAST_NAME, JOB_CODE, DEPT, SALARY, DOB) WITH TE...	DML, Insert/Update/Delete	17.712000	116	4.387000	238,612
UPDATE DB2USER1.OPM_SAMPLE_TABLE SET SALARY = 1.1 * SALARY WHERE SALARY < 30000	DML, Insert/Update/Delete	5.392000	43	3.307000	907,647
SELECT OPM_SAMPLE_TABLE_ID, FIRST_NAME ' ' LAST_NAME AS NEWNAME, SALARY, JOB_CODE, sum(SALARY) over(...	DML, Select (blockable)	1.861000	35	1.809000	752,328
SELECT * FROM DB2USER1.OPM_SAMPLE_TABLE ORDER BY FIRST_NAME, LAST_NAME, SSN fetch first 6000 rows only	DML, Select (blockable)	1.318000	36	1.263000	759,890
SELECT * FROM DB2USER1.OPM_SAMPLE_TABLE ORDER BY DOB desc, SALARY ASC, JOB_CODE, FIRST_NAME DESC, L...	DML, Select (blockable)	1.243000	36	1.185000	759,890
UPDATE DB2USER1.OPM_SAMPLE_TABLE SET SALARY = 1.4 * SALARY WHERE SALARY BETWEEN 30001 AND 40000	DML, Insert/Update/Delete	1.204000	36	0.716000	759,891
DELETE FROM DB2USER1.OPM_SAMPLE_TABLE WHERE SALARY > 80000	DML, Insert/Update/Delete	0.912000	36	0.622000	795,890
UPDATE DB2USER1.OPM_SAMPLE_TABLE SET SALARY = 1.7 * SALARY WHERE SALARY BETWEEN 40001 AND 50000	DML, Insert/Update/Delete	0.753000	35	0.544000	738,783
SELECT * FROM DB2USER1.OPM_SAMPLE_TABLE ORDER BY DOB desc, SALARY ASC, JOB_CODE, FIRST_NAME DESC, L...	DML, Select (blockable)	0.677000	36	0.622000	759,891
SELECT * FROM DB2USER1.OPM_SAMPLE_TABLE ORDER BY FIRST_NAME, LAST_NAME, SSN fetch first 1000 rows only	DML, Select (blockable)	0.663000	37	0.654000	780,999

Analyze SQL

The screenshot shows the 'SQL Statements Dashboard' in InfoSphere Optim Performance Manager. The interface includes a navigation bar with 'Open', 'Databases', 'Connection', and 'SQL Statements' tabs. A filter bar shows 'View: Historical Data', 'End Time: 11/21/11 18:17', and 'Duration: 2 Hours'. The main area displays a table of SQL statements with columns for Statement Text, Statement Category, Execution Elapsed Time, Number of Executions, CPU Time, Rows, Lock Wait Time, and Sort overflows. A 'Top Individual Executions' tab is active, showing a list of statements. A 'SQL Statement Details' panel is open for a selected statement, showing its text and execution statistics. Annotations highlight key features: 'Top executions or summary views' points to the table; 'Filter and action controls' points to the 'Show Highest' and 'by' dropdowns; 'View changes in Configuration Manager' points to the 'View Configuration Changes' button; 'Click to tune with Query Tuner' points to the 'Tune' button; and 'In-context switching' points to the 'Show current executions of the selected statement' and 'Show top individual executions of the selected statement' options.

- Top executions or summary
- In-context switching
- Top by criteria and filtering
- Real time or historical
- Partition or member views
- SQL statement details
- In-context WLM reports
- What changed with Configuration Manager
- Tune with Query Tuner

Reporting

General

- Performance Overview
- Configuration
- Connection

SQL analysis

- Top SQL
- Top Package
- SQL comparison

Resource analysis

- Disk space consumption
- Table
- Workload Manager

Top SQL Report

Show Highest 5 by Average Execution Elapsed Time (sec) All Other Statements

Note: Selecting any of the options above will reload the report.
For purpose of readability, the bar chart below will show a max of 20 statements.

Click the chart to drill down to show more detailed periods of time (from years to months, months to days, and days to hours).

Top 5 SQL statements by Average Execution Elapsed Time (sec)

SQL Baseline Comparison

Show Highest 10 by Average Execution Elapsed Time (sec) View Regressions and Improvements

Regressions

Statement Text	Number of Executions		Average Execution Elapsed Time (sec)		Changes	Changes (%)
	Baseline	Report Interval	Baseline	Report Interval		
SELECT POLICY FROM SYSTOOL.POL...	2	4	0.321000	0.638750		98.99%
SELECT TRIGNAME FROM SYSCAT.T...	2	4	0.444000	0.667750		50.39%
SELECT STATS_FLAG FROM SYSTOOL...	2	4	0.241000	0.352500		46.27%
SELECT COLNAME, TYPENAME FROM ...	2	4	0.290000	0.368500		27.07%
				1.173750		19.49%

Response Times

Times

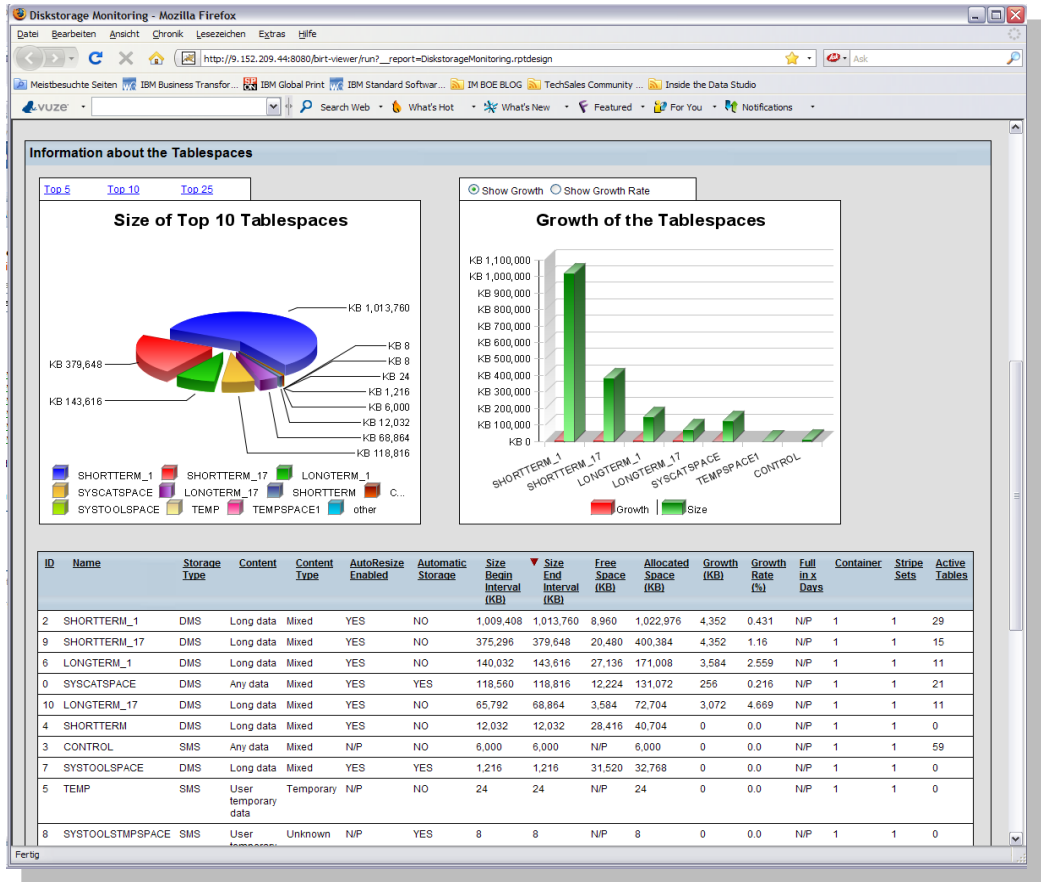
	Average/execution	Total
Elapsed (sec.)	1.128488	505.562502
CPU time (sec)	0.000907	0.406250
System (sec)	0.000140	0.062500
User Time (sec)	0.000767	0.343750
Sort (sec)	0.000027	0.012000

Be Proactive. Focus On Prevention

Prevent problems by leveraging historical information

Prevent

- Monitor and analyze historical trends for planning
- Auto manage workloads



- Tune proactively**
 - Identify heavy hitter SQL
 - Get expert advice
 - Compare results
- Add value to developers**
 - Identify hot spots for developers
 - Give them tuning advice
- Plan for growth**
 - Assess workload and storage growth trends
- Allocate resources by business priority**
 - Ensure critical work has resource preference
 - Improve server utilization
 - Protect data server from overload

**Thank
You**

The text 'Thank You' is rendered in a large, bold, 3D font. Each letter of the word 'Thank' and 'You' is filled with a different photograph of a person. The 'T' shows a man in a suit and tie. The 'h' shows a woman in a green top. The 'a' shows a man with a green face. The 'n' shows a woman in a blue top. The 'k' shows a man with glasses. The 'Y' shows a man in a white lab coat. The 'o' shows a man in an orange shirt. The 'u' shows a woman in a blue top. The letters have a slight shadow and a blueish tint.