



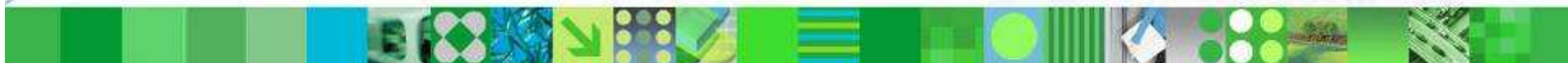
Data Management

DB2 9.7: Monitoring Enhancements

David Kalmuk
Senior Software Engineer, DB2 for LUW
dckalmuk@ca.ibm.com

Agenda

- **Background**
- **What Is New In DB2 9.7 GA?**
- **What Is New In DB2 9.7 FP1?**
- **Questions?**



Data Management

Background

Release Objectives for Monitoring

- **Substantially reduce the time and effort needed to detect and resolve common problems in**
 - System Performance
 - Query Performance
 - Locking
- **Provide a solid framework from which to address other domains in the future**
- **Provide essential metrics to our tools in a fast, efficient, and complete manner**

Tooling

- **The tooling provided for DB2 9.5 has been merged into the Optim tooling strategy**
 - DB2 Performance Expert product is now part of that organization
 - For more information: www.ibm.com/software/data/optim
- **Optim's focus is on providing a consistent, integrated solution to the data lifecycle**
 - From "requirements to retirement": Design, Deploy, Manage, and Govern
 - Includes end-to end monitoring capabilities
 - Heterogeneous support across IBM and competitor databases
 - Aligning with Tivoli and other tooling in the enterprise "eco-system"
- **A lot of our work in DB2 9.7 was driven by tooling requirements**
- **No specific tooling enhancements discussed as part of this presentation**



Data Management

What Is New In DB2 9.7 GA?

What Is New In DB2 9.7 GA?

- **In-memory Metrics**
 - A New Monitoring Infrastructure
 - New Table Functions / Reporting Levels
 - “Time Spent” Metrics: Wait Times

- **Historical Data Capture**
 - New Event Monitor Infrastructure
 - New Event Monitor Target Type: UE Table
 - New Event Monitors
 - Enhancements to Event Monitors



Data Management

In-Memory Metrics

A New In-memory Monitoring Infrastructure

- **As of DB2 9.7, we begin to move away from system monitor and snapshot technology for database monitoring**
 - Moving towards constant “in-memory” aggregation and accumulation of metrics within DB2 at different levels
- **In DB2 9.7, we are introducing a low-impact, efficient alternative to the traditional system monitor infrastructure**
 - Independent from system monitor (i.e. not connected to existing infrastructure or monitor switches)

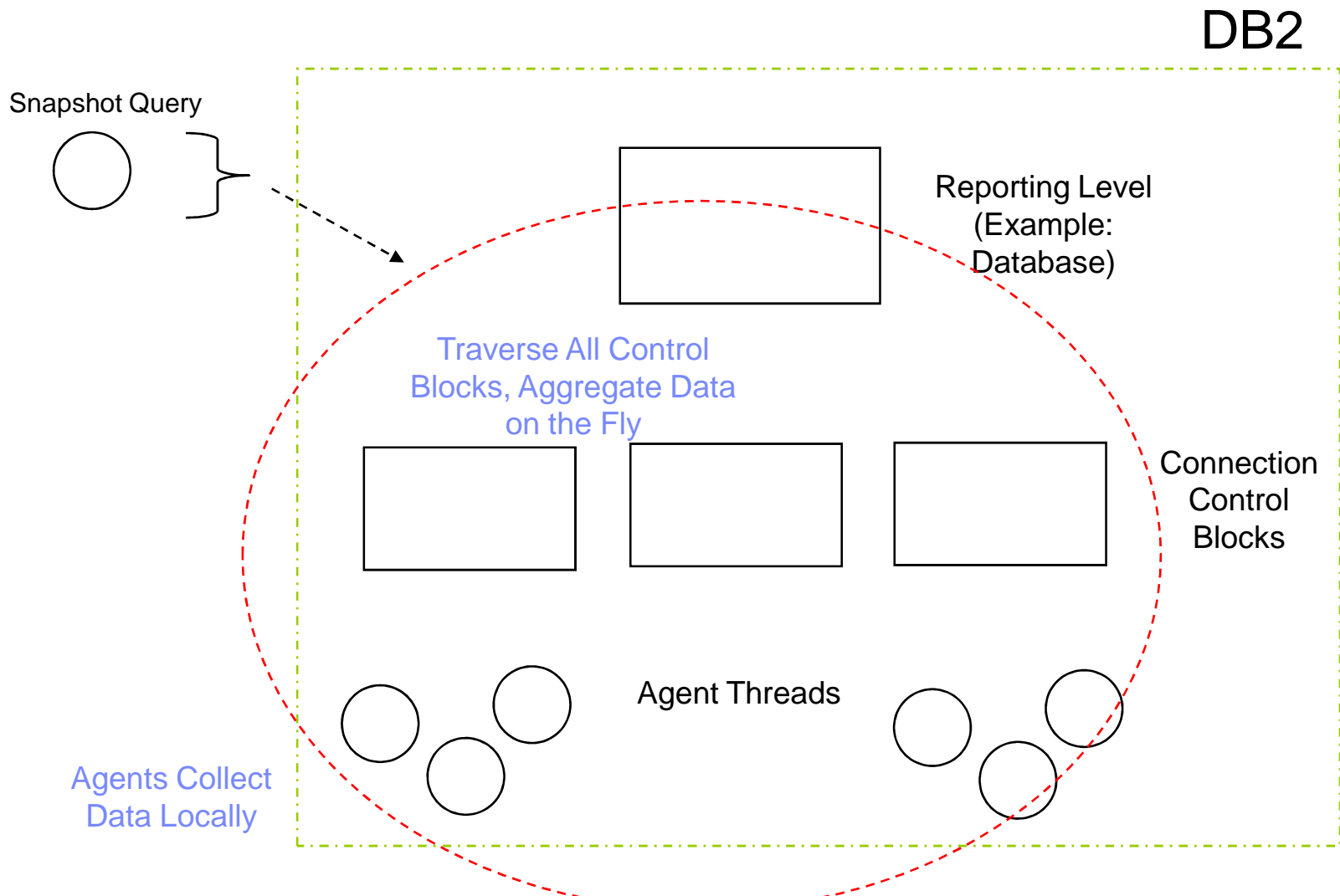
Goals / Motivation

- **Reduce overhead of collecting active monitoring data**
- **Reduce overhead of monitoring queries**
- **Enable more monitoring capabilities by default**

Background: Snapshot Infrastructure

- **Monitor Data Collection**
 - Agent threads collect operational metrics in thread local storage
 - Data kept with the agent is rolled up to higher accumulation levels as late as possible (generally only when an agent terminates)
- **Querying Monitor Data**
 - Snapshot query will drill-down and chase agent chains and add up agent data on the fly for the given accumulation level (application, database, etc).
- **Very good model for keeping runtime costs low when monitoring queries are relatively infrequent**
- **Consequence is that frequent queries / certain types of snapshots may introduce contention that can impact the runtime workload**

Snapshot Infrastructure



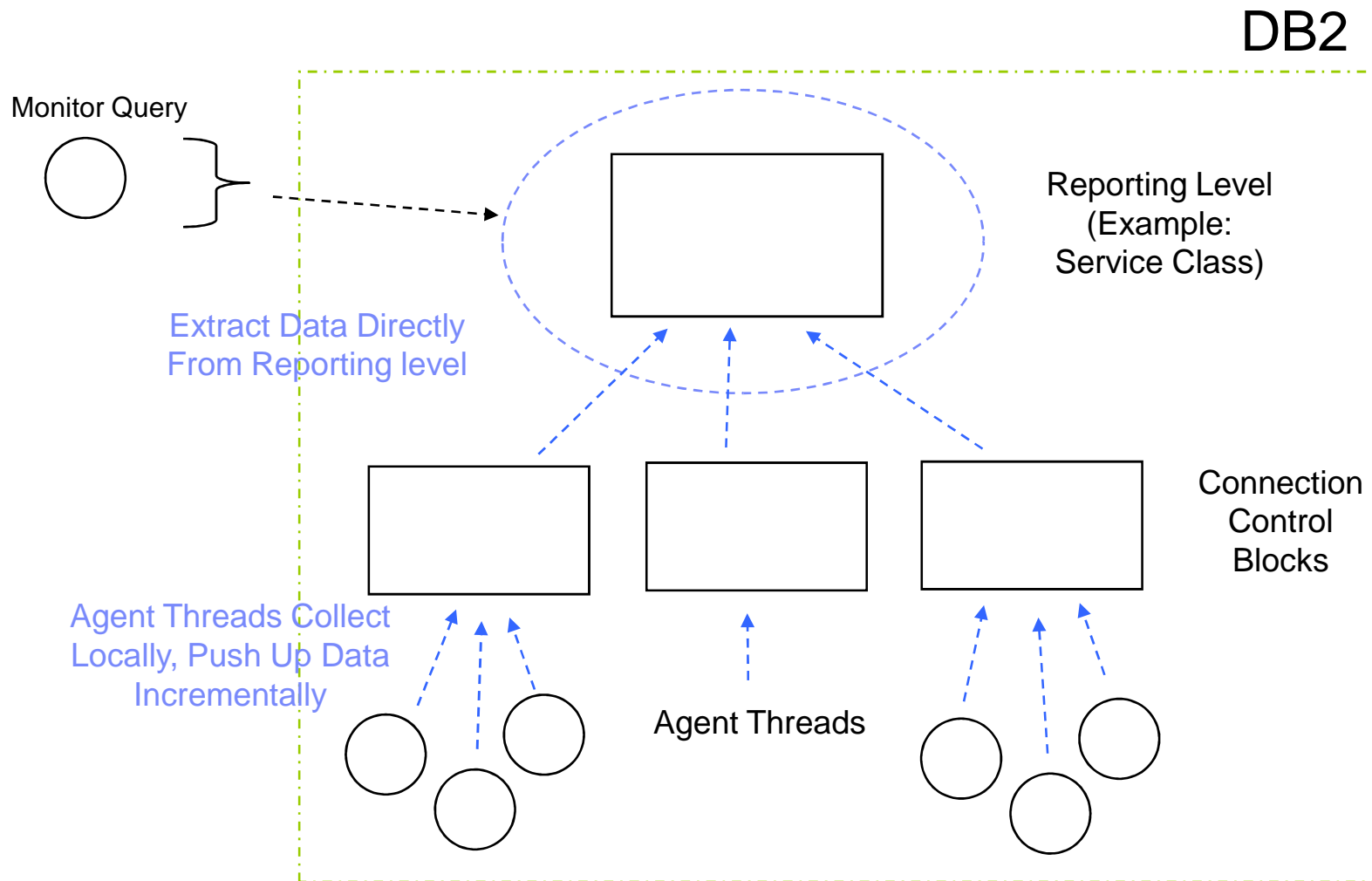
Background: Snapshot Infrastructure

- **C API based interfaces**
- **SQL wrappers provided on top of native C APIs**
 - Formatted binary data stream is mapped to relational output
 - Routines are fenced
 - Not as efficient as 'native' trusted table functions
- **Snapshots return unfiltered data**
 - No filtering at source; full data stream is returned (size depends on snapshot type) and can be filtered by client
- **Some data available by default**
 - Significant amount of other data controlled by switches that are off by default

New Monitoring Infrastructure

- **Agent threads collect operational metrics in thread local storage**
- **Data is rolled up to higher accumulation levels at transaction boundaries or every 10 seconds**
- **Queries return data from higher level accumulation points only; no drilldown performed**
 - Reduces contention of frequent queries
- **Specific effort devoted to performance optimization and tuning of the infrastructure**

New Monitoring Infrastructure



New Monitoring Infrastructure

- **'Native' SQL Table Function Interfaces**
 - Approach used in V9.5 for WLM table functions, and some admin functions.
 - Scrape engine in-memory data directly
 - Input arguments allow filtering of data at source to reduce volumes and overhead of queries
- **In-memory data collected by default on new databases**
 - Global controls available via db config parameters
 - More granular controls available at workload object level

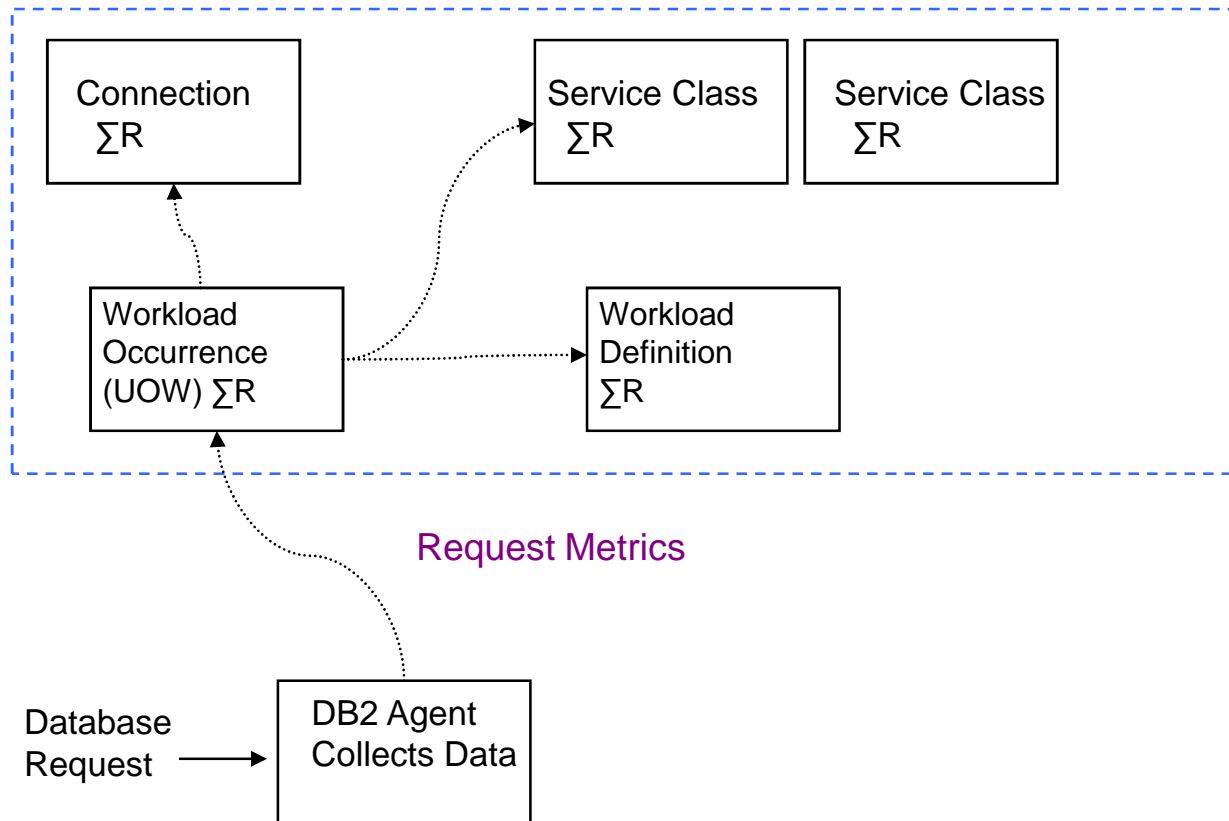
Comparing the Overhead (OLTP)

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

New Reporting Levels

- **System**
 - Provide total perspective of application work being done by database system
 - Aggregated through the WLM infrastructure
- **Data objects**
 - Provide perspective of impact of application work on data objects
 - Aggregated through data storage infrastructure
- **Activity**
 - Provide perspective of work being done by specific SQL statements
 - Aggregated through the package cache infrastructure

In-Memory Metrics: System Perspective



Legend

ΣR = Accumulation of request metrics collected by agent

Access Points: System Perspective

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

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

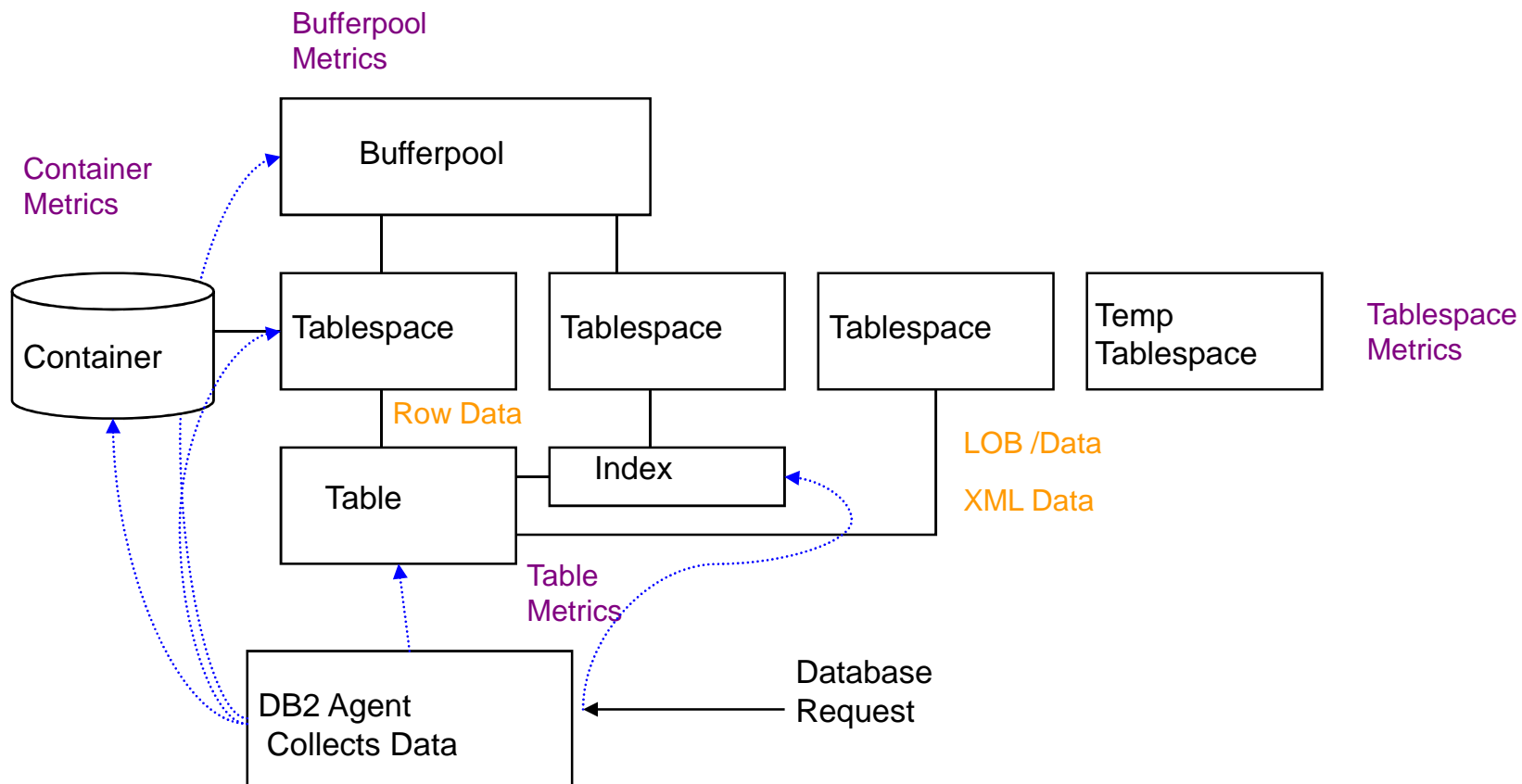
Example

```
$ db2 "select workload_name, application_handle, total_rqst_time from
      table(mon_get_unit_of_work(NULL,-2)) as t order by total_rqst_time desc
```

WORKLOAD_NAME	APPLICATION_HANDLE	TOTAL_RQST_TIME
-----	-----	-----
SYSDEFAULTUSERWORKLOAD	80	20512
SYSDEFAULTUSERWORKLOAD	81	19300
SYSDEFAULTUSERWORKLOAD	83	5210
SYSDEFAULTUSERWORKLOAD	82	810

```
4 record(s) selected.
```

In-Memory Metrics: Data Object Perspective



Access Points: Data Object Perspective

- **MON_GET_TABLE**
- **MON_GET_INDEX**
- **MON_GET_BUFFERPOOL**
- **MON_GET_TABLESPACE**
- **MON_GET_CONTAINER**

Example

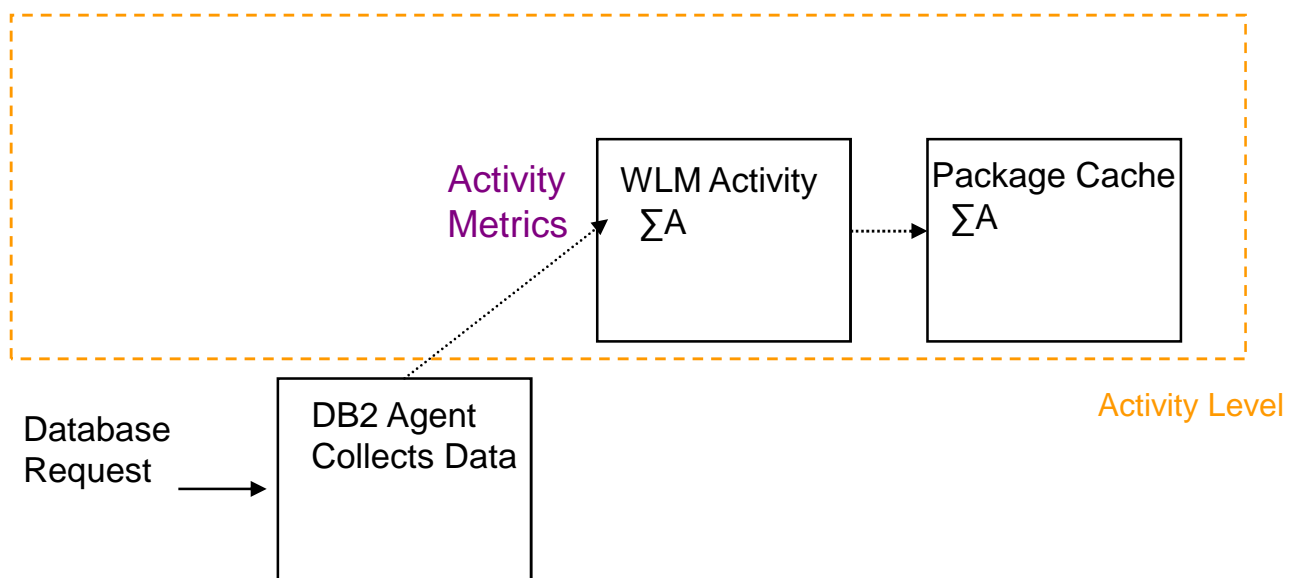
```
db2 "select substr(tabschema,1,10) as schema, substr(tabname,1,10) as tab, iid,
      nlevels from table (mon_get_index('','",-2)) as t order by nlevels desc"
```

SCHEMA	TAB	IID	NLEVELS
-----	-----	-----	-----
SYSIBM	SYSPLAN	3	2
SYSIBM	SYSPLAN	1	2
SYSIBM	SYSROUTINE	10	2
SYSIBM	SYSROUTINE	9	2
SYSIBM	SYSROUTINE	8	2

...

82 record(s) selected.

In-Memory Metrics: Activity Perspective



Legend

ΣA = Accumulation of metrics from activity execution portion of request

Access Points: Activity Perspective

- **MON_GET_PKG_CACHE_STMT**
 - Provides metrics for both static + dynamic statement entries in the cache
- **MON_GET_ACTIVITY_DETAILS (XML)**

Example

```
db2 "select substr(stmt_text,1,40) as stmt, total_cpu_time from
     table(mon_get_pkg_cache_stmt('',NULL,NULL,-2)) as t order by total_cpu_time
     desc fetch first 10 rows only
```

STMT	TOTAL_CPU_TIME
select application_id, application_handl	50140
create table t1(a int)	21966
select substr(stmt_text,1,20) as stmt, t	3308
select substr(tabschema,1,10), substr(ta	2294
select substr(tabschema,1,10) as schema,	1335
select substr(tabschema,1,10) as schema,	1269
select substr(tabschema,1,10), substr(ta	1262
select workload_name, application_handle	622
select * from t1	184
insert into t1 values 0	114

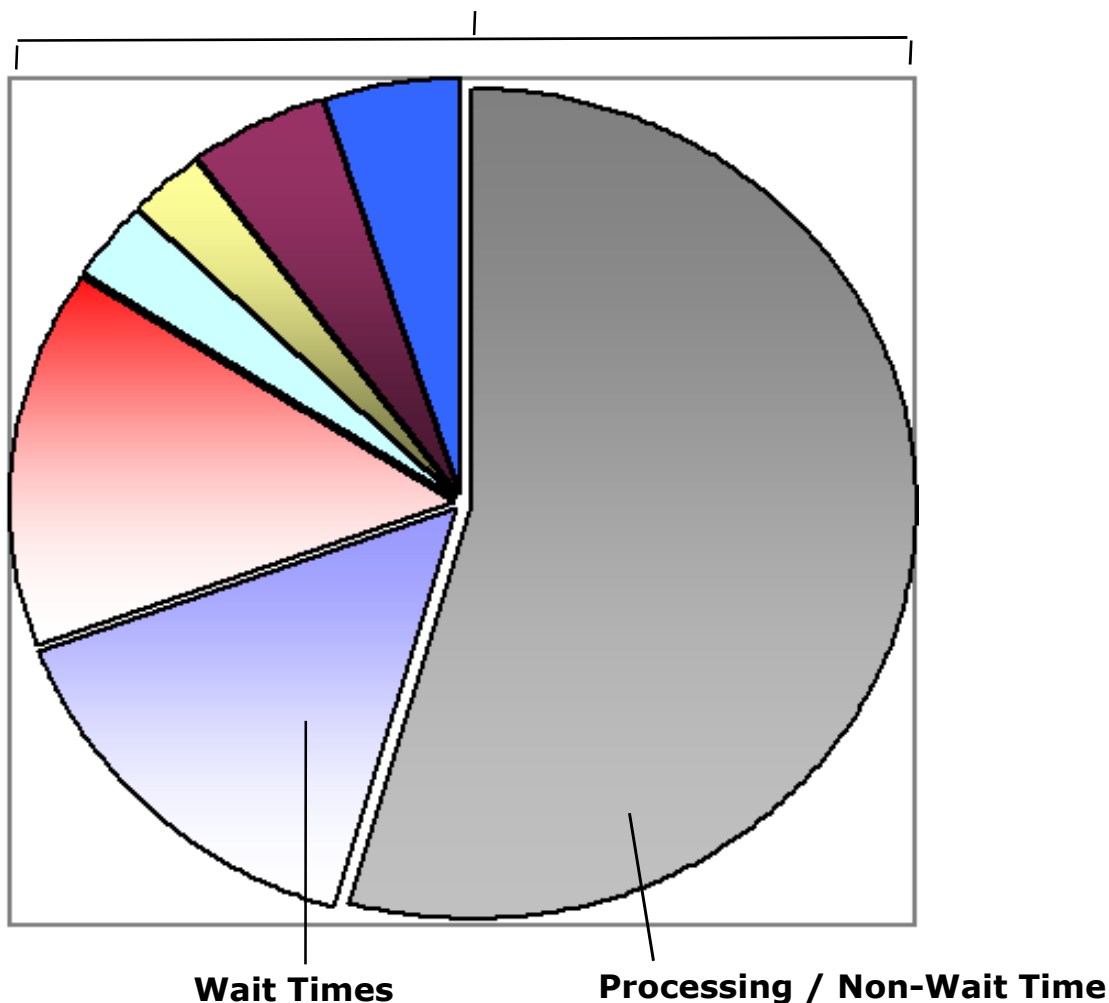
10 record(s) selected.

Introducing “Time Spent” Metrics

- **A new set of metrics are being introduced into DB2 that represent a breakdown of where time is spent within DB2**
 - Represents sum of time spent by each agent thread in the system (foreground processing)
 - Available in both system and activity perspectives
 - Can be used for rapid understanding of performance issues and problem areas
- **In 9.7 GA, introducing “wait times”**
 - These measure the time spent within DB2 waiting for a particular entity or resource

“Time Spent” Metrics: Wait Times

Total Time



Default Time Metrics

- Bufferpool Read Wait
- Bufferpool Write Wait
- Direct I/O Read Wait
- Direct I/O Write Wait
- Lock Wait
- Agent Wait
- WLM Queue Wait
- FCM Send Wait
- FCM Receive Wait
- Network Send Wait
- Network Receive Wait
- Log Write Wait
- Log Buffer Insert Wait

WLM Table Function Enhancements

- **A number of the WLM table functions introduced in DB2 9.5 have been enhanced in DB2 9.7 to contain more information**
 - See documentation for more information
 - Some fields are marked as reserved and will be filled in with FP1
- **Changed functions:**
 - WLM_GET_WORKLOAD_STATS_V97
 - WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES_V97
 - WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES_V97
 - WLM_GET_SERVICE_CLASS_AGENTS_V97
 - WLM_GET_SERVICE_SUBCLASS_STATS



Data Management

Historical Data Capture

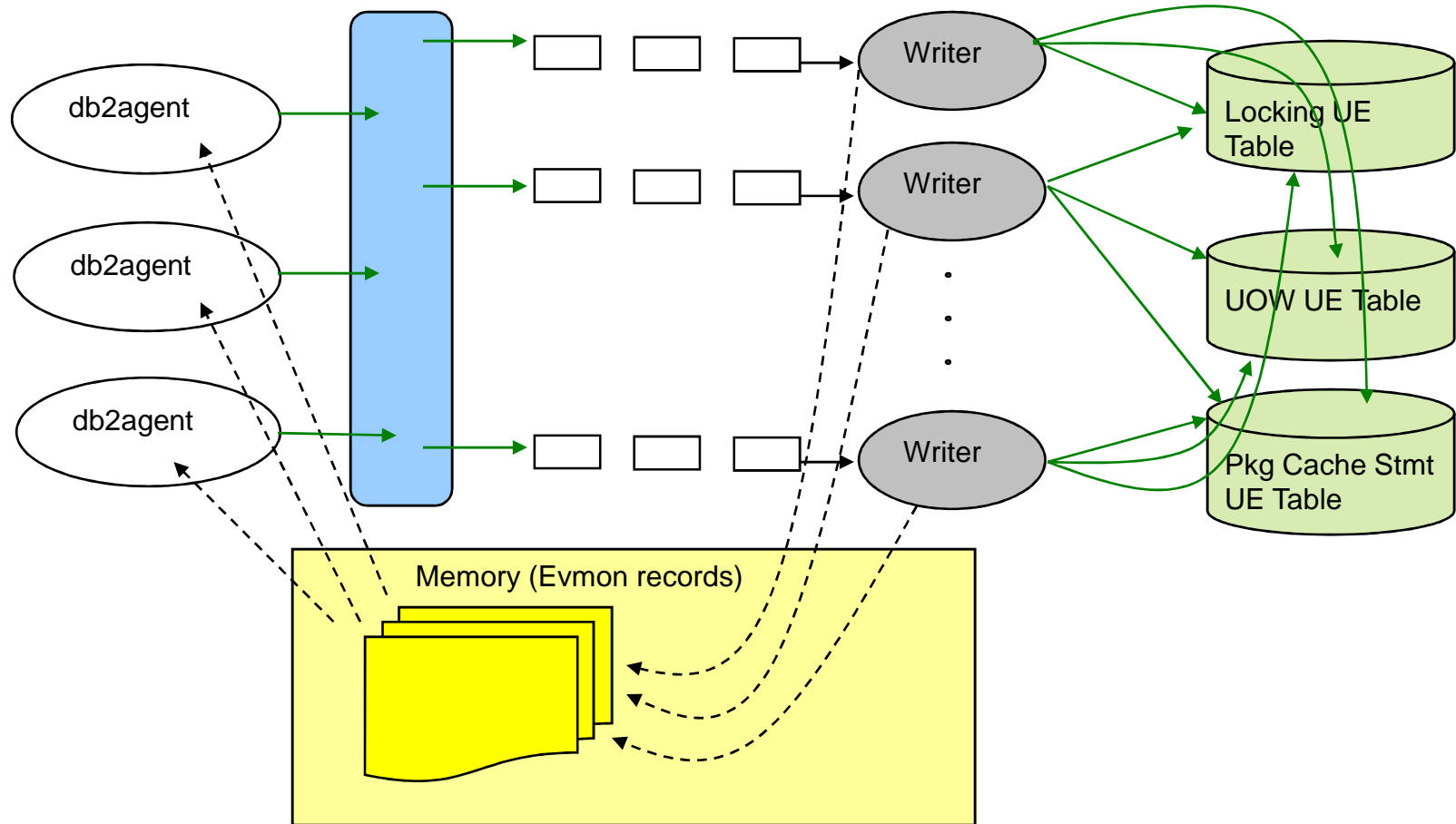
New Event Monitor Infrastructure

- **Trend towards higher volume event monitors**
 - Activity Event Monitor in V9.5 (SQL Trace)
 - Capturing unit of work data continuously (OPM)
- **Current event monitors utilize single dedicated writer thread**
 - Bottleneck for high volume data capture
 - Significant overhead for capturing detailed activity data today
- **Goal is to introduce next generation event monitor technology**
 - Improve scalability of high volume event monitors
 - Handle trend towards much greater data volumes, continuous data capture

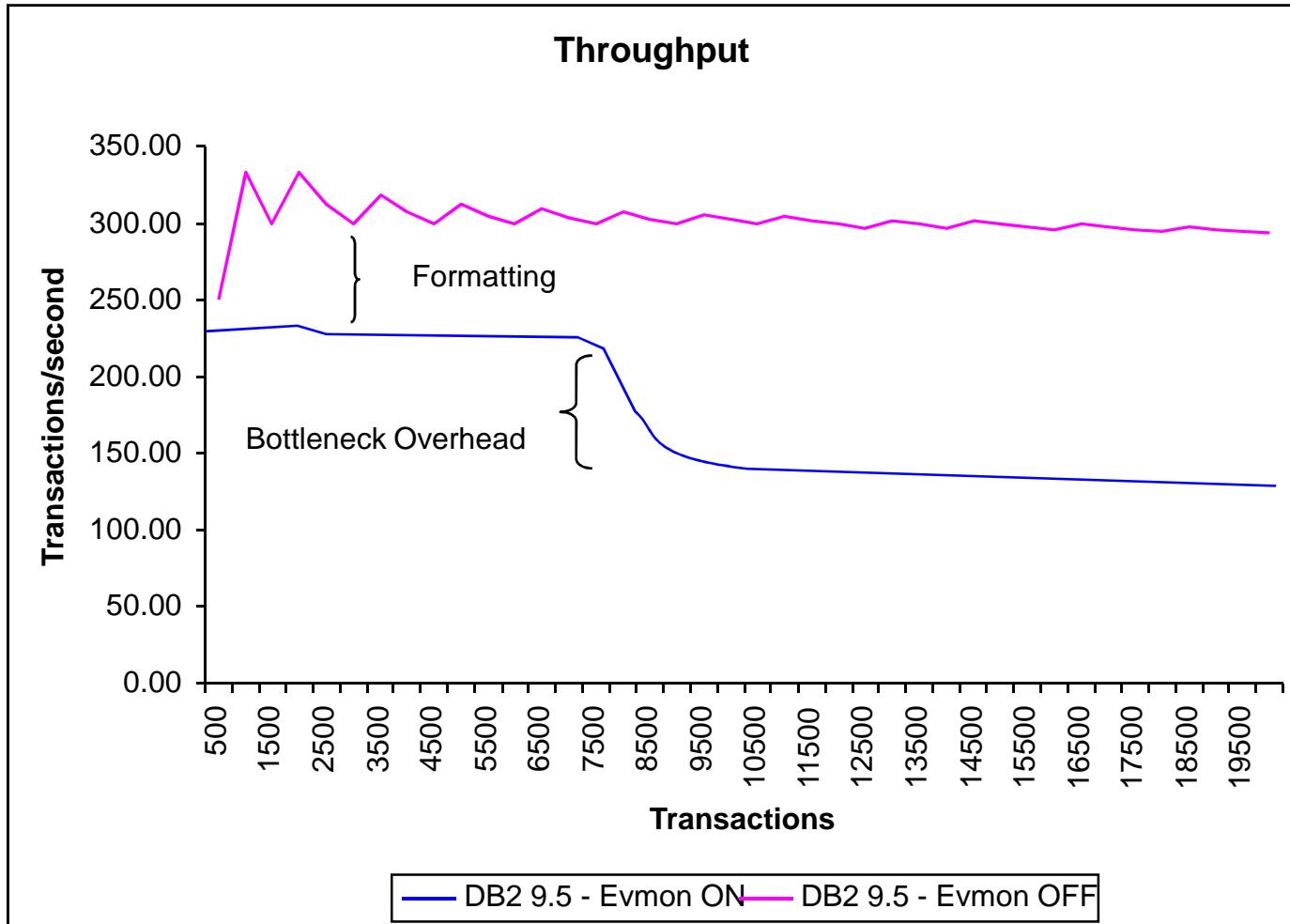
'Fast Writer' Architecture

- **New high volume event monitor infrastructure**
- **Set of parallel 'fast-writer' threads on the database system as opposed to a single dedicated thread**
- **'Fast-writer' threads are common and are shared by all event monitors using the architecture**
- **High volume event monitors can exploit multiple threads to drive out data in parallel**
 - One thread per CPU core on the system to allow full leveraging of available parallelism
- **Lower volume event monitors don't hold onto a thread for longer than they need it**
- **Removing the single thread bottleneck allows us to push formatting costs off the agent and onto the asynchronous backend**

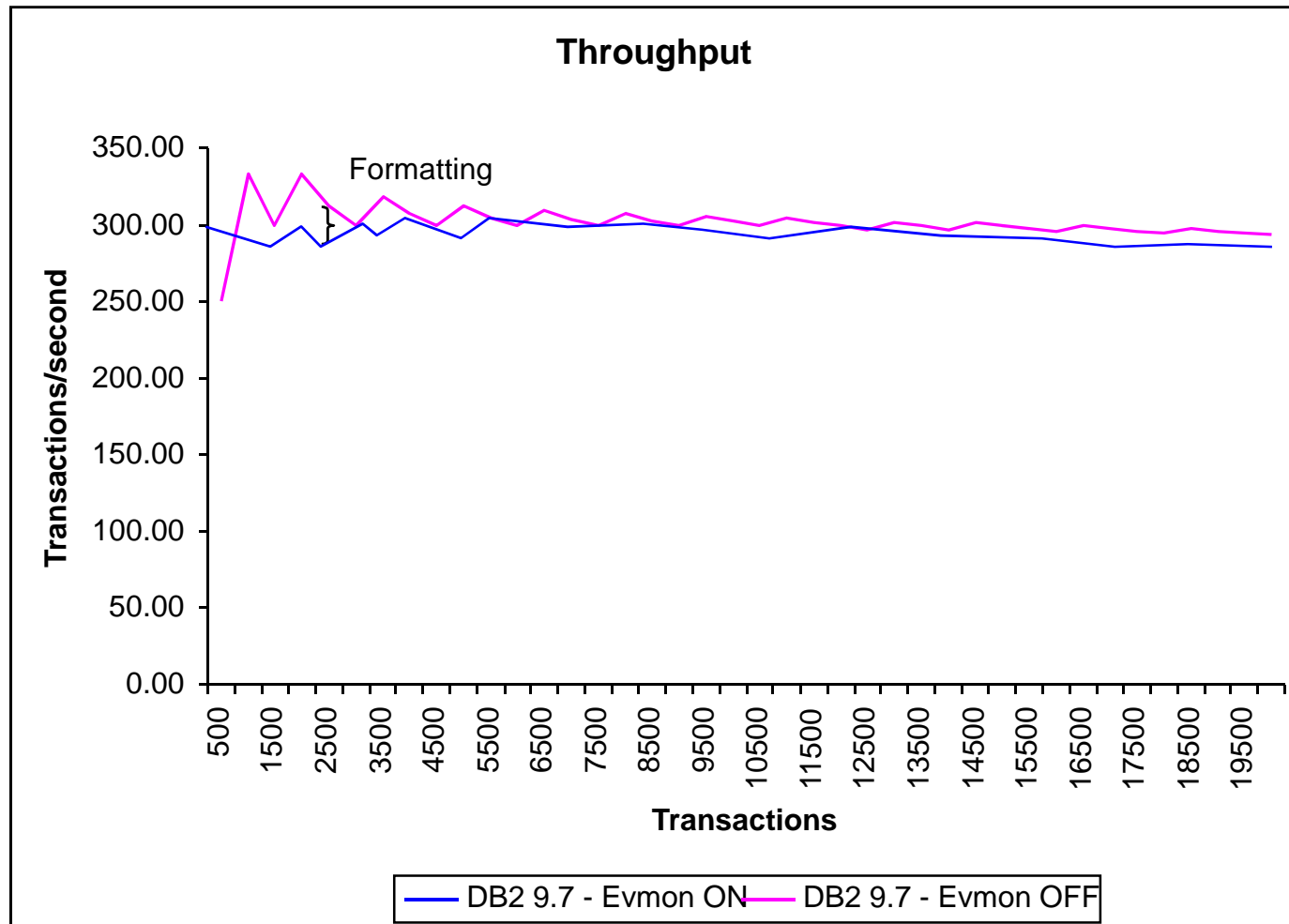
DB2 9.7 Fast Writer Architecture



Potential Issues...



New Event Monitor Architecture...



Comparing the Overhead

- **Agents do not need to pay formatting costs; these are handled asynchronously by the backend writers which reduces overhead on application queries**
- **Multiple fast-writer threads make it much more difficult for agents to generate enough event data to backlog the event monitor queues and force the agents to wait**
- **Example scenario:**
 - During development we experimented with event monitor scenarios that generate huge volumes of data in an attempt to backlog the event monitor queues and force agents to wait
 - In DB2 9.5 with sufficient volume we are able to produce a backlog
 - In DB2 9.7 these same scenarios no longer cause any backlogging when using the new architecture

New Target Type: Unformatted Event Table

- **There is still an underlying cost to the system that needs to be paid for generating event monitor data (formatting and writing events is still 'work' done by the system).**
- **With the new event monitor architecture we have taken this overhead off of the foreground agents, but it's still desirable to reduce this impact on the system for scalability reasons**
- **DB2 9.7 also introduces a new 'unformatted event' table event monitor target type which is designed to address this need**

New Target Type: Unformatted Event Table

- **Event monitor data is written to table, but as a binary lob instead of formatted relational output**
 - BLOB data is both inlined and compressed for performance purposes
- **Goal is to move formatting costs outside of the engine to provide further scalability improvements; impact on the system is reduced for generating the data**
- **Data can be consumed by using one of several formatting methods provided**
 - XML formatting via `EVMON_FORMAT_UE_TO_XML`
 - Relational table via `EVMON_FORMAT_UE_TO_TABLE`
- **Strategic direction for new event monitors**

New Event Monitors

- **Locking Event Monitor**

- Consolidated mechanism for capturing and performing in-depth analysis of locking data
- Support for capturing:
 - Deadlocks
 - Lock Timeouts
 - Lock waits
- Optional statement history
- Replaces existing deadlock event monitor and lock timeout report
- Supports 'Unformatted Event Table' target type

New Event Monitors:

- **Unit of Work Event Monitor**
 - Replaces existing transaction event monitor
 - Capture includes new Unit of Work Metrics
 - Supports 'Unformatted Event Table' target type

Event Monitor Enhancements

- **Can now have more than one WLM event monitor active at any one time**
 - Affects activity, statistics, and threshold event monitors
 - Allows for easier maintenance by establishing new event monitor and disabling old one for maintenance
- **Activity event monitor migrated to new 'fast writer' architecture**
 - Greatly reduces impact on processing agent overhead
 - Improves system throughput to reduce system impact under high volumes of capture
 - Still uses relational table output

Event Monitor Enhancements

- **Metrics data captured for both activity and statistics write-to-table event monitors**
 - Metrics data corresponds to that returned via in-memory interfaces
 - Metrics are reported as an XML document and can be processed via SQL using the XMLTABLE function, or the row-based formatters being provided in FP1
- **Ability to capture section data via activity event monitor**
 - Previously available via registry setting, now supported via 'with section' clause at workload level



Data Management

What Is New In DB2 9.7 FP1?

What's New In DB2 9.7 FP1?

- **“Time Spent” Metrics: Component Times**
- **Row-based Access for “Time Spent” Metrics**
- **New Locking Table Functions**
- **Package cache filtering**
- **Explain From Section**
- **Section “Actuals”**
- **Event Monitor Enhancements**

What's New In DB2 9.7 FP1?

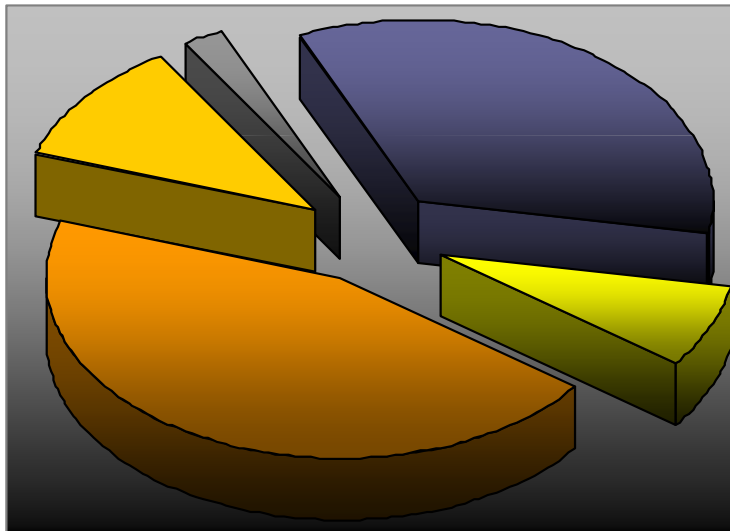
- **New Package Cache Event Monitor**
- **New Administration Views**
- **Text Reports**
- **Last Used information**
- **Some other miscellaneous items**

“Time Spent” Metrics: Component Times

- Recall that in DB2 9.7 we introduced a set of wait times designed to give a comprehensive view of where agent threads spend their time blocking on resources
- Component times represent different logical areas in DB2 where agent threads spend their time *processing* (i.e. consuming CPU)
- These times show major processing stages in DB2 such as query compilation, section execution, and commit / rollback time
- Times were specifically designed with no overlap to allow a comprehensive breakdown of time in DB2
- Like wait times these are available in both system and activity perspectives
- This completes our comprehensive “time spent” model and provides an overall breakdown of where time is spent in DB2 for both processing and waits allowing the user to quickly pinpoint key areas of interest

“Time Spent” Metrics: Component Times

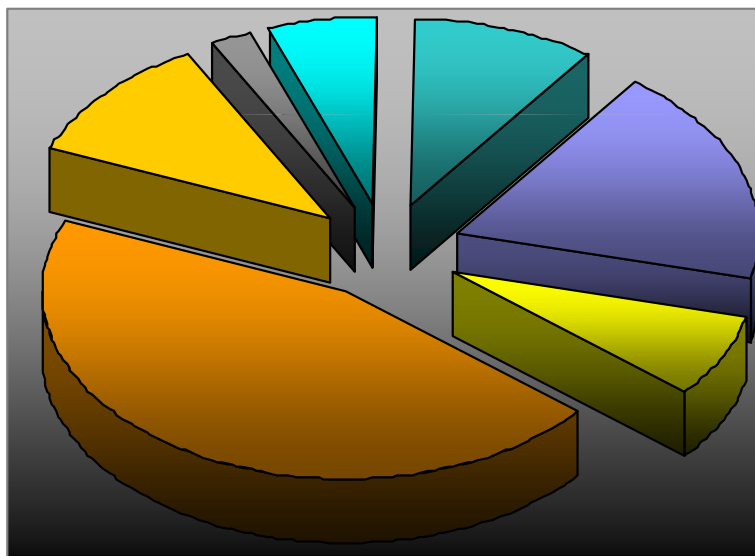
Total Request Time in DB2



- Compile Proc Time
- Section Proc Time
- Commit / Rollback Proc Time
- Other Proc Time
- Non-processing (wait time)

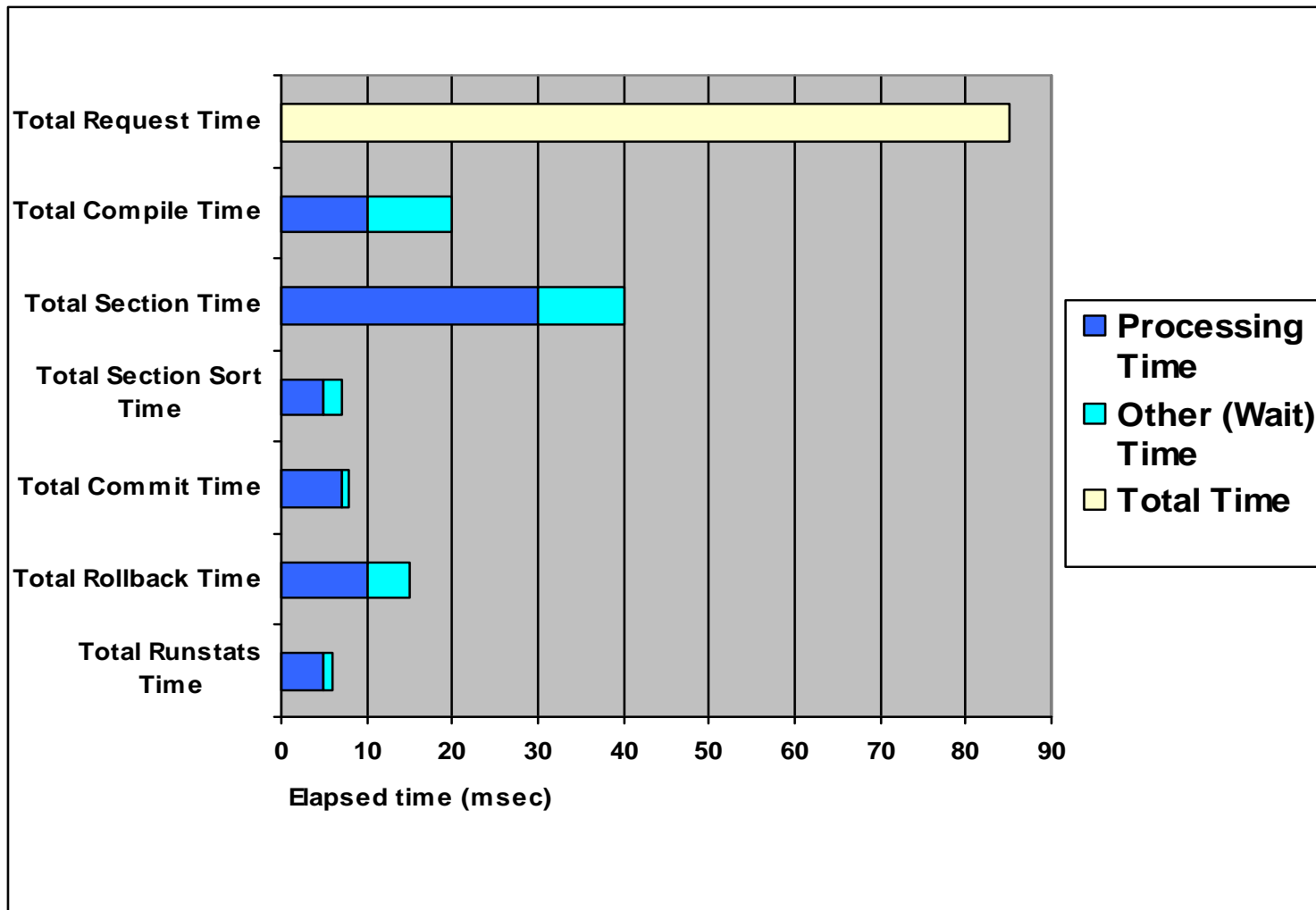
“Time Spent” Metrics: Combined Wait + Component Times

Total Request Time in DB2



- Direct I/O
- Bufferpool I/O
- Lock Wait Time
- Compile Proc Time
- Section Proc Time
- Commit / Rollback Proc Time
- Other Proc Time

Alternate Component Time Perspective



Row-based Access for “Time Spent” Metrics

- **Two new formatting functions will allow generic row-based output of “time spent” metrics**
 - MON_FORMAT_XML_WAIT_TIMES_BY_ROW
 - MON_FORMAT_XML_COMPONENT_TIMES_BY_ROW
- **Allows generic ranking of time spent metrics to determine top areas where time is spent**
- **Functions format metrics from any of the XML output documents as produced from the *_DETAILS table functions or event monitors**
- **Leverages generic interchange capabilities of the XML data**

Example

```

SELECT SUBSTR(TFXML.WORKLOAD_NAME, 1, 25) AS WORKLOAD_NAME,
       SUBSTR(WAITS.METRIC_NAME, 1, 25) AS METRIC_NAME,
       WAITS.TOTAL_TIME_VALUE_TIME,
       WAITS.COUNT
FROM
  TABLE( MON_GET_WORKLOAD_DETAILS( NULL, -2 ) ) AS TFXML,
  TABLE( MON_FORMAT_XML_WAIT_TIMES_BY_ROW(
                                     TFXML.DETAILS
                                     ) ) AS WAITS
ORDER BY WAITS.TOTAL_TIME_VALUE DESC

```

WORKLOAD_NAME	METRIC_NAME	TOTAL_TIME_VALUE	COUNT
PAYROLL	CLIENT_IDLE_WAIT_TIME	2193672	174
FINANCE	CLIENT_IDLE_WAIT_TIME	738290	16
PAYROLL	DIRECT_READ_TIME	67892	81
FINANCE	DIRECT_READ_TIME	32343	8
FINANCE	LOCK_WAIT_TIME	8463	3
PAYROLL	LOCK_WAIT_TIME	55	1
...			

New Locking Table Functions

- **Allows adhoc analysis of locking problems on the system**
- **Completes DB2 9.7 locking diagnostic capabilities**
- **MON_GET_APPL_LOCKWAIT**
 - Displays all applications in lock wait state, and which locks and applications they are waiting for
- **MON_GET_LOCKS**
 - Displays information on locks held on the database system
- **MON_FORMAT_LOCK_NAME**
 - Row based formatter for the binary lock name that shows applicable lock attributes (such as table name, rowid, etc)

Example

```

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

```

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

5 record(s) selected.

Package Cache Filtering

- **MON_GET_PKG_CACHE_STMT** has been enhanced to provide some additional filtering options via its generic **search_args** field
 - **modified_within** allows filtering of only package cache entries that were updated within a certain delta from the current time
 - **update_boundary_time** is used in conjunction with the new package cache event monitor to limit the scope of entries captured (more details on this later)

Explain From Section

- **Three new explain procedures allow explain tables to be populated using actual runtime section data**
 - EXPLAIN_FROM_SECTION
 - EXPLAIN_FROM_ACTIVITY
 - EXPLAIN_FROM_CATALOG
- **Explain output can be generated from the following sources:**
 - Statement entry in the SQL cache
 - SQL cache entry captured by the new package cache event monitor
 - Static statement from the catalog tables
 - Statement execution captured with section by the activity event monitor

Explain From Section

- **Provides a major subset of SQL Compiler explain information but available from any section at any time**
- **Enhances and formalizes undocumented capability available in 9.5 and 9.7 GA**
- **Output can be processed using any existing explain tools (eg. db2exfmt)**

Section “Actuals”

- **Cardinality counts now kept per major section operators during execution (also referred to as runtime statistics)**
- **Collected as part of the section data captured by the activity event monitor**
- **EXPLAIN_FROM_ACTIVITY procedure will produce a new explain table containing actuals data**
- **db2exfmt output will be updated to include cardinality data when available**

Event Monitor Enhancements

- **New FP1 metrics will automatically be available through event monitors where metrics were introduced in GA**
 - WLM Statistics
 - Activity Event Monitor
 - Unit of Work Event Monitor
- **Unit of Work Event Monitor is being enhanced to include the list of packages involved in the unit of work**
 - Will provide an indication of which static statements and routines were involved in a given unit of work

Package Cache Event Monitor

- **Allows capture of SQL cache entries (and associated metrics) when they are evicted from the package cache**
- **Both dynamic and static cache entries will be captured**
- **Filtering available via event monitor 'where' clause to control volume on a per event monitor basis**
 - Based on number of executions
 - Based on overall aggregate execution time
 - Based on a boundary time from when the event monitor was activated, and optionally updated for a specific event monitor via `MON_GET_PKG_CACHE_STMT` (only entries updated since the boundary will be captured when evicted from the SQL cache)

New Administration Views

- **We are providing a set of views built on top of new monitoring functions in FP1**
- **Goal of these views is to provide useful computed metrics and tie together key information from the table functions**
- **List of views not finalized yet but will likely comprise:**
 - Locking information
 - Currently executing SQL or Units of Work
 - System and/or Statement level summary information

New Text Reports

- **Table functions are primarily tooling interfaces**
 - Provide raw metrics, easy analysis via SQL
 - Expected to be used by tools or scripts
 - More difficult to consume from command line
- **In FP1 we are introducing a series of text based reports for command-line users**
- **Module based procedures that provide formatted text output readable by human beings**
- **Reports will take deltas based on a user specified interval**
- **Our current answer to the 'get snapshot' functionality available through CLP in DB2 9.7**

New Text Reports

- **Proposed text reports in FP1**
 - db2monreport.summary
 - db2monreport.connection
 - db2monreport.pkgcachesummary
 - db2monreport.activitysummary
 - db2monreport.pitapplication
 - db2monreport.lock

Example

```
db2 "call db2monreport.connection()"
```

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

```
-----  
Monitoring report by connection  
-----
```

```
Database:                SAMPLE  
Generated:                06/26/2009 00:28:23  
Interval monitored:      10  
-- Command options --  
APPLICATION_HANDLE:      All
```

```
...
```

```
-----  
#      APPLICATION_  TOTAL_   TOTAL_   ACT_COMPLETED  TOTAL_WAIT_  CLIENT_IDLE  
      HANDLE        CPU_TIME ACT_TIME  _TOTAL         TIME          _WAIT_TIME  
-----  
1      1207          0        0         0              0            0
```

```
...
```

Last Used Information

- **We are introducing a LASTUSED data column for various objects to provide an indication on when the object was last utilized**
 - Indexes
 - SYSCAT.INDEXES.LASTUSED
 - Packages
 - SYSCAT.PACKAGE.LASTUSED
 - Tables
 - SYSCAT.TABLES.LASTUSED
 - Table Data Partitions
 - SYSCAT.DATAPARTITIONS.LASTUSED
- **Allows users to determine which objects are being actively used, and might be candidates for removal**

Synchronized WLM Statistics Collection

- **Synchronize WLM stats daemon collection interval against wall-clock to ensure consistent intervals; even across outages**
- **For example if 5 minute collection intervals are specified, collection will occur at 12:00, 12:05, 12:10, as opposed to every 5 minutes starting from the arbitrary activation time**

(Proposed) Change to WLM License Coverage

- **CREATE WORKLOAD will no longer require the WLM license**
 - Will enable more granular application identification and monitoring
- **Still under WLM License:**
 - CREATE SERVICE CLASS
 - CREATE THRESHOLD
 - CREATE WORK ACTION SET



Data Management

Summary

A wide variety of new facilities available in DB2 9.7! 😊

- **'Next generation' infrastructure for both in-memory metrics and event monitors**
- **A variety of new monitoring functions and metrics for analyzing and diagnosing several classes of problems:**
 - System Performance
 - SQL Performance
 - Locking
- **A variety of new event monitors and historical capture capabilities to complement the in-memory functions**
- **DB2 9.7 GA introduces the new infrastructure and a core set of function**
- **DB2 9.7 FP1 completes the picture with further function and enhancements**



Data Management

Questions ?