DB2 Performance Tuning: Tips & Techniques

Sweta Singh
IBM

Session 6
Platform: DB2 LUW
Agenda

• Performance Basics
• Diagnosing Performance Problems
  • Sanity Check
  • First Aid toolkit
    • OS tools
    • New Monitoring in DB2 9.7
    • OS & DB2 Monitoring Combo
  • RCA Decision Tree
• Interesting Case Study
Basics: Orchestration matters

Hardware: CPU, Memory, Storage and Network

- CPU cores and RAM available continue to grow
  - 12 cores/chip for both Intel Ivybridge EP and IBM POWER8
  - 8GB RAM/core is a good Rule of Thumb
  - Think of IOPS and MB/sec read/write for sizing storage
- Include SSDs in your storage strategy— they can make an astonishing difference!
  - Random I/O, especially reads, benefit significantly
  - Internal SSDs have best and most cost-effective performance but
    - They often don't have a write cache
    - Consider HA and DR

Software: OS, DB2 and Application

- Use best practice recommendations
- Stay current on maintenance
Basics: Understand the physical layout of your database

- **Disk spindles still matter**
  - With sophisticated storage subsystems and storage virtualization it just requires more sleuthing than ever to find them
  - Drives keep getting bigger, 146GB-300GB now the norm

- “Spread everything everywhere” strategy works reasonably well
  - For OLTP, make sure that the transaction logs reside on a separate array

- **Be leery of Storage Administrators that tell you**
  - “Don’t worry, it doesn’t matter”
  - “The cache will take care of it”
Basics: Storage best practices

Typical OLTP DB2 Storage Layout

Typical Data Warehouse DB2 Storage Layout

File System Usage
- Permanent and temp data
- Archival logs
- Active logs

File System Usage
- Permanent and temp data
- Archival logs
- Active logs and database directory
Basics: Best Practices (DB2 configuration)

- **Use DB2 automatic storage**
  - A modest number of storage paths that map sensibly to the SAN storage

- **Minimize the use of distinct page sizes**
  - 8K is a good default (no more than 2)
  - Specify the smaller one at create database time for the catalog

- **Minimize the number of bufferpools**
  - 1 for data/index, 1 for temp is usually sufficient

- **Split tables for logical groups**
  - Often helpful to split data/index/LOB (especially LOB) into separate tablespaces
  - No need to put EACH table in its own tablespace

- **Use the Autonomics**
  - Autoconfigure & STMM
    - But give DB2 a budget via instance_memory or database_memory

- **Compression**
  - Turn on compression at create table time for top ~50 large tables
Agenda

- Performance Basics
- Diagnosing Performance Problems
  - Sanity Check
  - First Aid toolkit
    - OS tools
    - New Monitoring in DB2 9.7
  - How to identify Resource Bottleneck
  - RCA Decision Tree
- Interesting Case Studies
Performance problem has occurred: Sanity Check

- Did something change since when performance met expectations?
  - New database applications?
  - Other non-database loads on the system?
  - More users?
  - More data?
  - Software configuration changes?
    - DB, DBM configuration, registry variables, schema changes, etc.
    - Hardware configuration changes?

- If you can identify the change at this stage, can it / should it be undone? Or is this something you have to live with?

- If this is a real problem, a methodical approach is key to finding the root cause
First Aid toolkit: OS tools

- Make `vmstat` your best friend
  - It will give you the first hint at where your bottleneck is

- `iostat` gives valuable insights into I/O activity
  - Use “`iostat –D`” on AIX, “`iostat –x`” on Linux to get I/O response time
First Aid toolkit: New Monitoring in DB2 9.7

- DB2 9.7 brings many big improvements in monitoring
  - Component & wait times
  - Static SQL / SQL procedure monitoring
  - Improvements in low-overhead activity monitoring via event monitors
  - db2pd sorting & currently committed statistics
  - Section actuals

- SQL Access to performance data
  - A good set of monitor queries makes diagnosing problems much easier
  - Choosing & tracking the most important metrics
  - Calculating derived values (hit ratio, time per IO, etc.)
  - Comparing against baseline values

- Transitioning from snapshots to SQL monitoring made easy
  - MONREPORT module
Glimpse of New Monitoring: monreport.dbsummary

Work volume and throughput
-------------------------------

<table>
<thead>
<tr>
<th></th>
<th>Per second</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTAL_APP_COMMITS</td>
<td>137</td>
<td>1377</td>
</tr>
<tr>
<td>ACT_COMPLETED_TOTAL</td>
<td>3696</td>
<td>36963</td>
</tr>
<tr>
<td>APP_RQSTS_COMPLETED_TOTAL</td>
<td>275</td>
<td>2754</td>
</tr>
</tbody>
</table>

TOTAL_CPU_TIME = 23694526
TOTAL_CPU_TIME per request = 8603

Row processing
ROWS_READ/ROWS_RETURNED = 1 (25061/20767)
ROWS_MODIFed            = 22597

Component times
-------------------

-- Detailed breakdown of processing time --

<table>
<thead>
<tr>
<th></th>
<th>%</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total processing</td>
<td>100</td>
<td>119880</td>
</tr>
</tbody>
</table>

Section execution

TOTAL_SECTION_PROC_TIME = 11 13892
TOTAL_SECTION_SORT_PROC_TIME = 0 47

Compile

TOTAL_COMPILE_PROC_TIME = 22 27565
TOTAL_IMPLICIT_COMPILE_PROC_TIME = 2 3141

Transaction end processing

TOTAL_COMMIT_PROC_TIME = 0 230
TOTAL_ROLLBACK_PROC_TIME = 0 0

Wait times
-----------

-- Wait time as a percentage of elapsed time

<table>
<thead>
<tr>
<th></th>
<th>%</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>For requests</td>
<td>70</td>
<td>286976/406856</td>
</tr>
<tr>
<td>For activities</td>
<td>70</td>
<td>281716/401015</td>
</tr>
<tr>
<td>-- Time waiting for next client request --</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLIENT_IDLE_WAIT_TIME</td>
<td>= 13069</td>
<td></td>
</tr>
<tr>
<td>CLIENT_IDLE_WAIT_TIME per second</td>
<td>= 1306</td>
<td></td>
</tr>
</tbody>
</table>

-- Detailed breakdown of TOTAL_WAIT_TIME

<table>
<thead>
<tr>
<th></th>
<th>%</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTAL_WAIT_TIME</td>
<td>100</td>
<td>286976</td>
</tr>
</tbody>
</table>

I/O wait time

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>POOL_READ_TIME</td>
<td>88</td>
<td>253042</td>
</tr>
<tr>
<td>POOL_WRITE_TIME</td>
<td>6</td>
<td>18114</td>
</tr>
<tr>
<td>DIRECT_READ_TIME</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>DIRECT_WRITE_TIME</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LOG_DISK_WAIT_TIME</td>
<td>1</td>
<td>4258</td>
</tr>
<tr>
<td>LOCK_WAIT_TIME</td>
<td>3</td>
<td>11248</td>
</tr>
<tr>
<td>AGENT_WAIT_TIME</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Buffer pool
-----------

Buffer pool hit ratios

<table>
<thead>
<tr>
<th></th>
<th>Ratio</th>
<th>Reads (Logical/Physical)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>72</td>
<td>54568/14951</td>
</tr>
<tr>
<td>Index</td>
<td>79</td>
<td>223203/45875</td>
</tr>
<tr>
<td>XDA</td>
<td>0</td>
<td>0/0</td>
</tr>
<tr>
<td>Temp data</td>
<td>0</td>
<td>0/0</td>
</tr>
<tr>
<td>Temp index</td>
<td>0</td>
<td>0/0</td>
</tr>
<tr>
<td>Temp XDA</td>
<td>0</td>
<td>0/0</td>
</tr>
</tbody>
</table>
First Aid toolkit Queries: Core Activity

- Transactions, statements, rows

```
select
current timestamp as "Timestamp",
substr(workload_name,1,32) as "Workload",
sum(TOTAL_APP_COMMITS) as "Total app. commits",
sum(ACT_COMPLETED_TOTAL) as "Total activities",
case when sum(TOTAL_APP_COMMITS) < 100 then null else
  cast( sum(ACT_COMPLETED_TOTAL) / sum(TOTAL_APP_COMMITS) as decimal(6,1)) end
  as "Activities / UOW",
case when sum(TOTAL_APP_COMMITS) = 0 then null else
  cast( 1000.0 * sum(DEADLOCKS)/ sum(TOTAL_APP_COMMITS) as decimal(8,3)) end
  as "Deadlocks / 1000 UOW",
case when sum(ROWS_RETURNED) < 1000 then null else
  sum(ROWS_READ)/sum(ROWS_RETURNED) end as "Rows read/Rows ret",
case when sum(ROWS_READ+ROWS_MODIFIED) < 1000 then null else
  cast(100.0 * sum(ROWS_READ)/sum(ROWS_READ+ROWS_MODIFIED) as decimal(4,1)) end
  as "Pct read act. by rows"
from table(mon_get_workload(null,-2)) as t
group by rollup ( substr(workload_name,1,32) );
```
First Aid toolkit: Time-spent Query #1

- TOTAL_RQST_TIME is “Time Spent in DB2”
  - Wait Time vs Processing Time

```sql
select
current timestamp as "Timestamp",
TOTAL_RQST_TIME,
TOTAL_RQST_TIME / TOTAL_APP_COMMITS "Time Spent per txn",
(TOTAL_CPU_TIME+500) / 1000 as TOTAL_CPU_TIME,
TOTAL_WAIT_TIME,
cast(100.0 * TOTAL_WAIT_TIME/TOTAL_RQST_TIME as decimal(4,1)) end
as Wait_time_pct,
cast(100.0 * ((TOTAL_CPU_TIME+500) / 1000) / TOTAL_RQST_TIME as decimal(4,1))
as CPU_time_pct
from  table(mon_get_workload(null,-2)) as t
```
select current timestamp as "Timestamp",
        substr(workload_name,1,32) as "Workload",

        case when sum(TOTAL_SECTION_SORTS) < 1000 then null else cast(100.0 * sum(SORT_OVERFLOWS)/sum(TOTAL_SECTION_SORTS) as decimal(4,1)) end as "Pct spilled sorts",

        case when sum(TOTAL_SECTION_TIME) < 100 then null else cast(100.0 * sum(TOTAL_SECTION_SORT_TIME)/sum(TOTAL_SECTION_TIME) as decimal(4,1)) end as "Pct section time sorting",

        case when sum(TOTAL_SECTION_SORTS) < 100 then null else cast(float(sum(TOTAL_SECTION_SORT_TIME))/sum(TOTAL_SECTION_SORTS) as decimal(6,1)) end as "Avg sort time",

        case when sum(TOTAL_RQST_TIME) < 100 then null else cast(100.0 * sum(TOTAL_COMPILE_TIME)/sum(TOTAL_RQST_TIME) as decimal(4,1)) end as "Pct request time compiling",

        case when sum(PKG_CACHE_LOOKUPS) < 1000 then null else cast(100.0 * sum(PKG_CACHE_LOOKUPS-PKG_CACHE_INSERTS)/sum(PKG_CACHE_LOOKUPS) as decimal(4,1)) end as "Pkg cache h/r",

        case when sum(CAT_CACHE_LOOKUPS) < 1000 then null else cast(100.0 * sum(CAT_CACHE_LOOKUPS-CAT_CACHE_INSERTS)/sum(CAT_CACHE_LOOKUPS) as decimal(4,1)) end as "Cat cache h/r"

        from

        ...

• Sorting, Compilation Time
select current timestamp as "Timestamp", substr(workload_name,1,32) as "Workload",
sum(TOTAL_RQST_TIME) as "Total request time",
sum(CLIENT_IDLE_WAIT_TIME) as "Client idle wait time",
case when sum(TOTAL_RQST_TIME) < 100 then null else
  cast(float(sum(CLIENT_IDLE_WAIT_TIME))/sum(TOTAL_RQST_TIME) as decimal(10,2)) end
as "Ratio of client wt to request time",
case when sum(TOTAL_RQST_TIME) < 100 then null else
  cast(100.0 * sum(TOTAL_WAIT_TIME)/sum(TOTAL_RQST_TIME) as decimal(4,1)) end
as "Wait time pct of request time",
case when sum(TOTAL_WAIT_TIME) < 100 then null else
  cast(100.0*sum(LOCK_WAIT_TIME)/sum(TOTAL_WAIT_TIME) as decimal(4,1)) end
as "Lock wait time pct of Total Wait",
... sum(POOL_READ_TIME+POOL_WRITE_TIME)/ ... as "Pool I/O pct of Total Wait",
... sum(DIRECT_READ_TIME+DIRECT_WRITE_TIME)/ ... as "Direct I/O pct of Total Wait",
... sum(LOG_DISK_WAIT_TIME)/ ... as "Log disk wait pct of Total Wait",
... sum(TCPIP_RECV_WAIT_TIME+TCPIP_SEND_WAIT_TIME)/ ... as "TCP/IP wait pct ...",
... sum(IPC_RECV_WAIT_TIME+IPC_SEND_WAIT_TIME)/ ... as "IPC wait pct of Total Wait"
... sum(FCM_RECV_WAIT_TIME+FCM_SEND_WAIT_TIME)/ ... as "FCM wait pct of Total Wait"
... sum(WLM_QUEUE_TIME_TOTAL)/ ... as "WLM queue time pct of Total Wait",
... sum(XML_DIAGLOG_WRITE_WAIT_TIME)/ ... as "diaglog write pct of Total Wait"
First Aid toolkit: Hot queries

- Sorted by CPU, ROWS_READ, NUM_EXECUTIONS

```sql
select MEMBER, TOTAL_ACT_TIME, TOTAL_CPU_TIME,
    (TOTAL_CPU_TIME+500)/1000 as "TOTAL_CPU_TIME (ms)"
    , TOTAL_SECTION_SORT_PROC_TIME,
    NUM_EXEC_WITH_METRICS, substr(STMT_TEXT,1,40) as stmt_text
from table(mon_get_pkg_cache_stmt(null,null,null,-2)) as t
order by TOTAL_CPU_TIME desc fetch first 20 rows only;

select ROWS_READ, ROWS_RETURNED,
    case when ROWS_RETURNED = 0 then null
          else ROWS_READ / ROWS_RETURNED end as "Read / Returned",
    TOTAL_SECTION_SORTS, SORT_OVERFLOWS, TOTAL_SECTION_SORT_TIME,
    case when TOTAL_SECTION_SORTS = 0 then null
          else TOTAL_SECTION_SORT_TIME / TOTAL_SECTION_SORTS end as "Time / sort",
    NUM_EXECUTIONS, substr(STMT_TEXT,1,40) as stmt_text ...

select TOTAL_ACT_TIME, TOTAL_ACT_WAIT_TIME, LOCK_WAIT_TIME,
    FCM_SEND_WAIT_TIME+FCM_RECV_WAIT_TIME as "FCM wait time",
    LOCK_TIMEOUTS, LOG_BUFFER_WAIT_TIME, LOG_DISK_WAIT_TIME,
    TOTAL_SECTION_SORT_TIME-TOTAL_SECTION_SORT_PROC_TIME as "Sort wait time",
    NUM_EXECUTIONS, substr(STMT_TEXT,1,40) as stmt_text ...
```
## Finding your resource bottleneck

### “OS & DB2 Monitoring” Combo

<table>
<thead>
<tr>
<th>Bottleneck</th>
<th>OS System Monitoring</th>
<th>DB2 Monitoring</th>
</tr>
</thead>
</table>
| Disk       | • High I/O wait seen in vmstat / iostat / perfmon  
             • Following symptoms in iostat  
             • Disk > 80% busy  
             • Average I/O Response time > 10 ms  
             • Long disk queues | • Disk I/O Wait time pct of wait time > 10% - 30%  
                          • ms per Bufferpool Read > 10 ms  
                          • ms per Bufferpool Write > 8 ms  
                          • ms per Log Write > 6 ms |
| CPU        | • Total CPU utilization near 100% in vmstat / perfmon | • Majority of time spent in processing (TOTAL_RQST_TIME – TOTAL_WAIT_TIME)  
                          • Percent of time spent in sorting  
                          • > 5% for OLTP  
                          • > 25% for DSS  
                          • Percent of time spent in compilation  
                          • > 1% for OLTP  
                          • > 5-10% for DSS |
# Finding your resource bottleneck

<table>
<thead>
<tr>
<th>Bottleneck</th>
<th>OS System Monitoring</th>
<th>DB2 Monitoring</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>▪ Low free memory seen in vmstat / perfmon&lt;br&gt;▪ Swapping reported in vmstat / perfmon&lt;br&gt;▪ Activity on swap disks seen in iostat&lt;br&gt;▪ Higher-than-normal system CPU time seen in vmstat</td>
<td>• Db2pd –memsets&lt;br&gt;• Db2pd -mempools</td>
</tr>
<tr>
<td>Lazy System</td>
<td>▪ Lower than expected CPU utilization in vmstat</td>
<td>• Lock Wait time pct of wait time &gt; 10%&lt;br&gt;• Latch Wait time pct of wait time &gt; 10%</td>
</tr>
</tbody>
</table>
RCA Decision Tree(*)

Inadequate disk configuration / subsystem?

- SQL w/o parameter markers?
  - Too small dyn SQL cache?
  - Apps connecting/disconnecting?
  - Non-parallelize application?

- Old device drivers
  - Creating/destroying agents
  - Too many connections

- High User Time
  - CPU Bottleneck

- High System Time
  - Network Bottleneck

Data Tablespace

- Bad plan giving tablesan?
  - Old statistics?
  - Need more indexes?
  - Insufficient prefetchers?
  - Over-aggressive cleaning?
  - LOB reads/writes?

- Insufficient sortheap?
  - Missing indexes?

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

Log Devices

- Lock escalation?
  - Lock contention?
  - Deadlocks?
  - Too few prefetchers?
  - Too few cleaners?
  - Append contention?
  - Mincommit too high?

Disk Bottleneck

- Bad plan(s) giving excessive index scanning?
  - Need more/different indexes?

Index Tablespace

- Excessive data flow?
  - LOBs
  - Intermediate data
  - Shared n/w conflict

Temp Tablespace

CPU Bottleneck

Network Bottleneck

(*) Steve Rees
Agenda

• Performance Basics
• Diagnosing Performance Problems
  • Sanity Check
  • RCA Decision Tree
  • First Aid toolkit
    • OS tools
    • New Monitoring in DB2 9.7
    • How to identify Resource Bottleneck
• Interesting Case Study
Case study

Problem Statement: During month end processing of a financial application, end users complain of very poor response time. Sometimes it gets so bad that DBAs have to ask end users to log off to bring performance under control 😞

Time Spent Query #1: Where is DB2 spending time?
### Time Spent Query: Wait time breakup & I/O response time

| BTIMESTAMP       | TOTAL_RQST_TIME | CLIENT_IDLE_WAIT_TIME | TOTAL_WAIT_TIME | Wait time pct of request time Lock wait time pct of Total Wait Pool I/O pct of Total Wait Direct I/O pct of Total Wait Log disk wait pct of Total Wait TCP/IP wait pct of Total Wait IPC wait pct of Total Wait FCM wait Lock waits Lock timeouts Deadlocks Lock escalations |
|------------------|------------------|-----------------------|-----------------|-----------------|----------------|-----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| 2013-12-2714.33.06.331026 | 182307546         | 751154075             | 105314269       | 57.7            | 11.1           | 83.9            | 0.0            | 0.0            | 0.0            | 0.0            | 0.0            | 0.0            | 0.0            |
| 2013-12-2715.07.57.484561 | 213516694         | 681158134             | 130611517       | 61.1            | 6.8            | 86.6            | 0.0            | 0.0            | 0.0            | 0.0            | 0.0            | 0.0            | 0.0            |
| 2013-12-2716.16.16.403637 | 133897425         | 802722190             | 87745314        | 65.5            | 7.8            | 86.3            | 3.5            | 0.0            | 0.0            | 0.0            | 0.0            | 0.0            | 0.0            |

**Page cleaning ratio**

<table>
<thead>
<tr>
<th>BTIMESTAMP</th>
<th>BPName</th>
<th>Total Reads</th>
<th>Total Writes</th>
<th>Async writes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Avg read time</td>
<td>Avg write time</td>
<td></td>
</tr>
<tr>
<td>2013-12-2714.02.19.485416</td>
<td>IBMDEFAULTBWP</td>
<td>3.18</td>
<td>24.50</td>
<td>663812</td>
</tr>
<tr>
<td>2013-12-2714.33.06.293863</td>
<td>IBMDEFAULTBWP</td>
<td>9.24</td>
<td>44.02</td>
<td>738442</td>
</tr>
<tr>
<td>2013-12-2715.07.57.393521</td>
<td>IBMDEFAULTBWP</td>
<td>17.88</td>
<td>45.06</td>
<td>384557</td>
</tr>
<tr>
<td>2013-12-2716.16.16.374577</td>
<td>IBMDEFAULTBWP</td>
<td>14.44</td>
<td>29.56</td>
<td>596890</td>
</tr>
</tbody>
</table>
### OS tools

2 disks at ~100% disk utilization

**Disk %Busy 12/27/2013**

<table>
<thead>
<tr>
<th>Disk</th>
<th>%Busy</th>
<th>I/O Bandwidth</th>
<th>Throughput</th>
<th>Error Rate</th>
<th>Avg Serv</th>
<th>Min Serv</th>
<th>Max Serv</th>
<th>Timeouts</th>
<th>Fails</th>
<th>Avg Time</th>
<th>Min Time</th>
<th>Max Time</th>
<th>Avg WQ Size</th>
<th>Avg SQ Size</th>
<th>SQ Full</th>
</tr>
</thead>
<tbody>
<tr>
<td>hddk434</td>
<td>95.1</td>
<td>2.4M</td>
<td>242.5</td>
<td>74.5K</td>
<td>2.3M</td>
<td>14.7</td>
<td>10.4</td>
<td>0.1</td>
<td>2.5S</td>
<td>0</td>
<td>0</td>
<td>227.8</td>
<td>12.3</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>5.0S</td>
<td>0</td>
<td>0</td>
<td>5.6</td>
<td>0.0</td>
<td>250.3</td>
<td>1.0</td>
<td>2.0</td>
<td>27.6</td>
<td>0</td>
<td>0</td>
<td>5.0</td>
<td>0</td>
<td>250.3</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>hddk434</td>
<td>94.5</td>
<td>1.8M</td>
<td>258.3</td>
<td>110.5K</td>
<td>1.7M</td>
<td>21.8</td>
<td>11.7</td>
<td>0.1</td>
<td>2.5S</td>
<td>0</td>
<td>0</td>
<td>236.5</td>
<td>12.3</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>5.0S</td>
<td>0</td>
<td>0</td>
<td>7.2</td>
<td>0.0</td>
<td>250.3</td>
<td>1.0</td>
<td>3.0</td>
<td>41.0</td>
<td>0</td>
<td>0</td>
<td>7.2</td>
<td>0</td>
<td>250.3</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>hddk434</td>
<td>94.3</td>
<td>1.8M</td>
<td>258.9</td>
<td>108.5K</td>
<td>1.7M</td>
<td>21.3</td>
<td>10.2</td>
<td>0.1</td>
<td>2.5S</td>
<td>0</td>
<td>0</td>
<td>237.7</td>
<td>12.5</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>5.0S</td>
<td>0</td>
<td>0</td>
<td>7.6</td>
<td>0.0</td>
<td>250.3</td>
<td>2.0</td>
<td>3.0</td>
<td>44.3</td>
<td>0</td>
<td>0</td>
<td>7.6</td>
<td>0</td>
<td>250.3</td>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td>hddk434</td>
<td>93.3</td>
<td>1.8M</td>
<td>251.9</td>
<td>103.4K</td>
<td>1.7M</td>
<td>20.2</td>
<td>10.2</td>
<td>0.1</td>
<td>2.5S</td>
<td>0</td>
<td>0</td>
<td>231.7</td>
<td>11.8</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>5.0S</td>
<td>0</td>
<td>0</td>
<td>6.6</td>
<td>0.0</td>
<td>250.3</td>
<td>1.0</td>
<td>2.0</td>
<td>35.7</td>
<td>0</td>
<td>0</td>
<td>6.6</td>
<td>0</td>
<td>250.3</td>
<td>1.0</td>
<td></td>
</tr>
</tbody>
</table>
### Hot Queries

<table>
<thead>
<tr>
<th>TIME</th>
<th>ROWNUM</th>
<th>RR/RS</th>
<th>Rows Read</th>
<th>Rows Returned</th>
<th>TOTAL_ACT_WAIT_TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>2012-12-27-14.32.45.822962</td>
<td>1</td>
<td>112</td>
<td>32599</td>
<td>291</td>
<td>1600321</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>294</td>
</tr>
<tr>
<td>2012-12-27-14.32.45.822962</td>
<td>2</td>
<td></td>
<td>1309</td>
<td></td>
<td>871785</td>
</tr>
<tr>
<td>UPDATE PSIBQUEUEINST SET</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2012-12-27-14.32.45.822962</td>
<td>3</td>
<td>1</td>
<td>679</td>
<td>679</td>
<td>819480</td>
</tr>
<tr>
<td>SELECT POL_TOTAL FROM YL WHERE PERIOD_END_DT=? AND EMPLID = ?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2012-12-27-14.32.45.822962</td>
<td>4</td>
<td></td>
<td>1177</td>
<td>1177</td>
<td>563161</td>
</tr>
<tr>
<td>SELECT SHEET_ID, SEQ_NUM, ATTACHSYSFILENAME, ATTACHUSERFILE, LAST_UPDATE_DTTM,</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2012-12-27-14.32.45.822962</td>
<td>5</td>
<td>1</td>
<td>537</td>
<td>536</td>
<td>510173</td>
</tr>
<tr>
<td>SELECT CUST_ID, PROJ_ROLE, DESCRIPT FROM XXX WHERE CUST_ID=?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2012-12-27-14.32.45.822962</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>949</td>
<td>352744</td>
</tr>
<tr>
<td>SELECT 'Y' FROM YYY WHERE EMPLID = ? FETCH FIRST 1 ROWS ONLY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2012-12-27-14.32.45.822962</td>
<td>7</td>
<td>85</td>
<td>8391</td>
<td>98</td>
<td>340375</td>
</tr>
<tr>
<td>SELECT SHEET_ID, SEQ_nbr_3, EXPENSE_TYPE, DESCRIPT, TXN_AMOUNT,</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2012-12-27-14.32.45.822962</td>
<td>8</td>
<td>1</td>
<td>268</td>
<td>268</td>
<td>333748</td>
</tr>
<tr>
<td>SELECT CUST_ID, PROJ_ROLE, DESCRIPT FROM...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2012-12-27-14.32.45.822962</td>
<td>9</td>
<td>1</td>
<td>3964</td>
<td>3859</td>
<td>325703</td>
</tr>
<tr>
<td>select b.emplid from x a, y b where a.oprid = ?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2012-12-27-14.32.45.822962</td>
<td>10</td>
<td>3</td>
<td>925</td>
<td>293</td>
<td>288198</td>
</tr>
<tr>
<td>SELECT BUSINESS_UNIT, PROJECT_ID, EFFDT, RESOURCE_CATEGORY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2012-12-27-14.32.45.822962</td>
<td>11</td>
<td>1</td>
<td>181</td>
<td>181</td>
<td>285040</td>
</tr>
</tbody>
</table>

- Explain plans for the hot queries shows lot of I/O happening due to large index-scans and table-scans
Performance Tuning

- Two pronged approach
  - Query Tuning: Create indexes to reduce large index scans and table scans
  - DB level Tuning:
    - Compressing the top 50 large tables
    - Increasing the Bufferpool size
  - Storage team advised to add I/O capacity
Sweta Singh
IBM
sweta_singh@in.ibm.com

DB2 Performance Tuning: Tips & Techniques
Session 6

Please fill out your session evaluation before leaving!