THINK GLOBAL – ACT LOCAL

Best Practices for Performance Tuning with CICS and DB2 for z/OS

Miao Zheng, zhengm@cn.ibm.com
Agenda

- Customer challenges on performance and availability
- Improve application cutover
- Improve online transactions
- Tools
Customer challenges on performance and availability
Today’s Challenges in Chinese Banks

- Huge data size based on the population
  - Largest table space with 4096 partitions, 8TB data
  - Largest table space page size as 64GB, 3 billion account details
- Quick grow business result a fast grow IT requirement
  - The peak transaction rate grows from ~3000 TPS in 2007 to more than 5800 TPS in 2011, so ~20% increasing per year
- Business innovation & expansion lead to frequent application change
  - iPhone/Android mobile banking, iPad e-Banking
  - Hundreds of packages rebound each quarter
- High demand of performance
  - Focus on elapsed time, any online transaction longer than 1 sec need to be reviewed
Customer System Configuration

- **Configurations**
  - Production system include 4 CICSPlex, more than 100 AORs/TORs; 10 DB2 data sharing groups, nearly 40 DB2 members, the biggest data sharing group is 12 members.
  - 10-way SYSPLEX, 4 way as primary (where all workloads running on), 4 way as standby (activated but nothing is running), and 2 way PPRC K system.

**Primary Active**
- z/OS V1R11, DB2 9, CICS TS 4.1

**Backup System**
- 4 z10, 2 z10 CF
Customer Application Environment

- **DB2 Applications**
  - 50TB db2 data.
  - Core banking for personal: approx. 8100 online packages, 11000 batch packages. 2100 tables.
  - Core banking for corporation: approx. 5900 online packages, 8600 batch packages. 1500 tables.

- **CICS Applications**
  - 5 CICSPlex
  - 50+ TORs & 70+ AORs.

- The highest transaction rate in history is about 5800 TPS.
- 3 hours for daily core batch window and more than 1000 batch jobs running concurrently
Application Availability Challenges

- Application availability
  - The availability excluding planned outage is very close to 100%, as a result of SYSPLEX, Data Sharing, PPRC/HyperSwap, SPOF elimination, etc.
  - The availability including planned outage is much lower, but which is the actual availability.

*Typically, availability of 99.75% means about 22 hours outage a year.*
In 2007, the unplanned outage is 59 min, while the total planned outage is 1494 min. The major causes are,
Application Cutover – 827 min, DB2 Offline REORG – 271 min, PTF Maintenance – 183 min, etc.

In 2010, the unplanned outage is 0 min, while the total planned outage is 1174 min. The major causes are,
Application Cutover – 1023 min, DB2 Offline REORG – 146 min, etc.

So, it’s time to CHANGE it!
Improve application cutover
Application cutover activities – bottleneck

- DDL implementation
  - DROP, CREATE, ALTER, ROTATE, etc.
- Data migration
  - COPY, UNLOAD, Sort, LOAD, REORG, REBUILD, Application Batch, etc.
- Package/Plan update
  - BIND, REBIND, FREE, etc.
DDL processing – challenges

- CREATE/DROP table space
  - It takes long time to delete VSAM data sets serially, to scan SYSLGRNX and SYSCOPY for cleanup.
  - Takes long time to define and open the data sets.

- ALTER ROTATE
  - Elapsed time is spent on package invalidation, log write I/O, notify messaging.
DDL processing – best practice

- Improve DROP table space
  - REORG SYSLGRNX before DROP table spaces before the cutover window.
  - MODIFY RECOVERY on the table spaces to be dropped, to delete related SYSCOPY and SYSLGRNX entries before the cutover window.
  - Delete the VSAM data sets before DROP table spaces using IDCAMS DEELTE jobs, which can be run in parallel

- Improve ALTER ROTATE
  - FREE packages depending on the table to be rotated before ROTATE
  - REORG INDEX on DSNDPX03 (index of SYSTABLEPART) to reserve sufficient free space
  - Keep only one DB2 member where ROTATE is running active and shutdown other members, this would dramatically reduce notify message time.
  - If that is not an option,
    - Remove GBP dependency of DSNDPX03 to reduce log write I/O
    - Remove GBP dependency on the tablespace to be rotated
DDL processing – product enhancement

- z/OS V1R12 shows significant improvement creating a table with 4096 partitions when other 80,000 DB2 objects are opened
- DB2 online schema enhancements
  - DB2 10 supports more online schema changes, e.g. alter tablespace page size, data set size, segment size, tablespace type, index page size, MEMBER CLUSTER, etc.

CREATE TS/TBL/IX (4096 parts) when other 80K open objects

<table>
<thead>
<tr>
<th>Second</th>
<th>V1R11</th>
<th>V1R12</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>500</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1500</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2500</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Others
- Update Commit
- SYSLGRNG
- Define
- Open/Close
Data migration – challenges

- **UNLOAD/LOAD**
  - It took UNLOAD utility long time to unload data for huge table

- **OLR Logapply is not catching up**
  - Heavy update: 1.8 Million changes per hour for 470 partitions
    Through 12 members * 300 user applications
  - LOG apply: 0.6 Million changes per hour
    Apply in parallel in part level, however storage constraint.

- **Slow NPI processing for REORG by partition range**
Data migration – best practice

- UNLOAD/LOAD
  - Increase the parallelism of COPY/UNLOAD by part level to reduce the total elapsed time.
  - Sort data during program unload (DSNTIAUL) or before load data back to tables, by clustering order.
Data migration – product enhancement

- **DB2 Sort 1.2 with DB2 Utilities**
  - Up to 39% reduction of sort CPU usage
  - Up to 41% reduction of utility elapsed time

- **LOAD & UNLOAD**
  - **LOAD/UNLOAD FORMAT INTERNAL**
    - 85% CPU & elapsed time reduction on UNLOAD
    - 77% elapsed time, 56% CPU reduction on LOAD
  - **LOAD PRESORTED**
    - Up to 25% CPU reduction, 33% ET reduction depending on no. of indexes

- **Fast LOAD through index avoidance**
  - New INDEXDEFER option to skip index key insert
    - Leaves indexes or logical partitions in RBDP
  - For LOAD RESUME or partition-level LOAD REPLACE with NPIs
    - Can even skip unique indexes
  - LOAD single part (5% of data) with 5 NPIs: Save 64% ET
Package update – challenges

- **BIND**
  - BIND can’t run concurrent which become the biggest time consume during application cutover.
Package update – best practice

- **BIND**
  - REORG plan_table, SYSPACKAGE, SPT01, etc. beforehand to improve BIND/REBIND performance.
  - VALIDATE(RUN) first
    - VALIDATE(RUN) does NOT check object existence and validity, which means packages can be preliminarily bound before DDL implementation is completed.
    - The elapsed time of REBIND VALIDATE(BIND) after BIND VALIDATE(RUN) is shorter than normal BIND, due to the package entry is created already.
    - Therefore, the first-step BIND can be run concurrently with DDL implementation with no contention, and the second-step BIND can save the total application cutover critical path time.
  - BIND with multiple qualifiers
    - The original objects in use are based on QUALIFIER1, the target objects to be in use after cutover are based on QUALIFIER2.
    - All DDL implementation, data migration, packages update are done under QUALIFIER2, when the original system based on QUALIFIER1 is still running and in service.
    - The cutover is to BIND plan to honor new QUALIFIER2 system.
Package update – product enhancement

- DB2 catalog restructure
  - The DB2 10 NFM Catalog is restructured to dramatically reduce contention, include removing all links, using Row Level Locking
  - As a result, concurrent BIND/REBIND become possible.
Improve online transactions
Protected threads

- **Good**
  - CPU reduction

- **But**
  - Long-running CICS DB2 threads that are constantly being reused build up resources in DB2 that can cause storage problems leading to abends and DB2 subsystem outages.
  - The threads become "fat" in that they build up "baggage" causing the EDM pool to grow causing storage problems in DB2.
  - Other Attaches into DB2 (RRS attach, IMS attach) have a hardcoded limit on the number of times a thread can be reused.
Protected threads – new in CICS TS 4.2

- New parameter to the DB2CONN definition
  - A new parameter REUSELIMIT has been added to the connection attributes section of the DB2CONN definition
  - A value of 0 means that there is no limit on the number of times that a thread can be reused; this was the situation before CICS TS 4.2.
  - Using the default of 1000 provides sufficient protection against over-allocating thread storage and EDM pool storage below the 2 GB bar when you are using a DB2 bind option of RELEASE(DEALLOCATE) without adversely affecting performance.
  - It applies to all threads, i.e., all DB2ENTRY threads and pool threads

- Changes to the DB2CONN definition
  - PURGECYCLE now allows a lower limit of 5 seconds (today it is 30 seconds which is also the default)
  - This controls how long protected threads are allowed to stay dormant before either being reused or terminated. Again it contributes to the "fat thread" problem.
  - The default remains at 30 seconds meaning on average a protected thread will be purged after 45 seconds.
### Protected threads – monitoring and evaluation

- **%Thread reuse** = \((\text{commits+rollbacks-deallocation}) / (\text{commits+rollbacks}) \) \times 100

- **DB2 PM report**

  **CONNTYPE: CICS**

<table>
<thead>
<tr>
<th>HIGHLIGHTS</th>
<th>NORMAL TERM.</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>#OCCURRENCES: 313424</td>
<td>NEW USER</td>
<td>50</td>
</tr>
<tr>
<td>#COMMTS: 575002</td>
<td>DEALLOCATION</td>
<td>293116</td>
</tr>
<tr>
<td>#ROLLBACKS: 3743</td>
<td>APPL. PROGR. END</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>RESIGNON</td>
<td>2025</td>
</tr>
</tbody>
</table>
BIND option

- RELEASE option can impact
  - When table space locks are released
  - Reuse of EDM pool pages occupied by cursor tables (CT) and package tables (PT)
  - Release of thread storage
  - Sequential detection resetting of and index look-aside information
  - Destruction of procedures (SPROC)

Note: DB2 10 deprecated ACQUIRE(ALLOCATE), so should use (USE) instead
DB2 10 below the 2G bar storage relief in DBM1

DB2 9-> DB2 10

Buffer Pools, Sort, RID pools
Global DSC  DBD
SKCT / SKPT
CT/PT
Local DSC
Thread, Stack

70-90% less usage in DBM1 2GB below virtual storage

Distributed IRWW Workload - DBM1 Below

Number of concurrent threads

- TOTAL GETMAINED STORAGE
- TOTAL VARIABLE STORAGE
- TOTAL FIXED STORAGE
- TOTAL GETMAINED STACK STORAGE

12/05/2012
Memory vs CPU strategy in DB2 10

- **DB2 V8 and DB2 9**
  - Need to reduce virtual storage consumption at the expense of CPU
    - RELEASE(COMMIT)
    - Smaller MAXKEEPD value for local stmt cache
    - Conservative max threads, more data sharing members
    - Storage contractions

- **DB2 10 strategy**
  - Reduce CPU at the expense of real storage
    - RELEASE(DEALLOCATE) in local
    - RELEASE(DEALLOCATE) for distributed threads (DBATs)
    - Larger MAXKEEPD values
    - More threads per member, less data sharing members
RELEASE (DEALLOCATE) and DB2 10

- Benefit of RELEASE(DEALLOCATE)
  - CPU reduction of package allocation and parent lock/unlock requests
  - Simpler the SQLs, larger the benefit
  - Keeping index lookaside across commits

- Impact of RELEASE(DEALLOCATE)
  - More memory (real storage) usage
  - Overhead of keeping package and lock information
  - Concurrency : BIND, DDL cannot break-in

DB2 CPU time =

SETUP TIME +

SQL EXECUTION TIME +

CLEAN UP TIME
Benefit of protected threads + release deallocate

- Effect of CICS protect thread and DB2 RELEASE(DEALLOCATE) in real banking application
  - CICS protected thread vs non-protected threads
  - RELEASE DEALLOCATE on both cases

<table>
<thead>
<tr>
<th>Tran</th>
<th>Response</th>
<th>Start</th>
<th>Start</th>
<th>SYNCPT</th>
<th>MIPS</th>
<th>CICS MIPS</th>
<th>DB2 MIPS</th>
<th>DB2 Reqs</th>
<th>TD Total</th>
<th>TS Total</th>
<th>PCLOAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATRAN</td>
<td>0.0827</td>
<td>6/29/04</td>
<td>0:37:02</td>
<td>1</td>
<td>22.3344</td>
<td>13.1472</td>
<td>9.1872</td>
<td>228</td>
<td>0</td>
<td>0</td>
<td>170</td>
</tr>
<tr>
<td>ATRAN</td>
<td>0.0943</td>
<td>6/29/04</td>
<td>0:37:33</td>
<td>1</td>
<td>21.7008</td>
<td>12.9492</td>
<td>8.7516</td>
<td>228</td>
<td>0</td>
<td>0</td>
<td>170</td>
</tr>
<tr>
<td>ATRAN</td>
<td>0.0826</td>
<td>6/29/04</td>
<td>0:37:53</td>
<td>1</td>
<td>22.0176</td>
<td>13.1472</td>
<td>8.8704</td>
<td>228</td>
<td>0</td>
<td>0</td>
<td>170</td>
</tr>
<tr>
<td>ATRAN</td>
<td>0.1552</td>
<td>6/29/04</td>
<td>0:38:12</td>
<td>1</td>
<td>22.8888</td>
<td>13.4244</td>
<td>9.4644</td>
<td>228</td>
<td>0</td>
<td>0</td>
<td>170</td>
</tr>
<tr>
<td>ATRAN</td>
<td>0.106</td>
<td>6/29/04</td>
<td>0:38:29</td>
<td>1</td>
<td>22.7304</td>
<td>13.3056</td>
<td>9.4248</td>
<td>228</td>
<td>0</td>
<td>0</td>
<td>170</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>22.30965</td>
<td>13.1868</td>
<td>9.12285</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BTRAN</td>
<td>0.1436</td>
<td>6/29/04</td>
<td>1:33:02</td>
<td>1</td>
<td>26.928</td>
<td>13.464</td>
<td>13.464</td>
<td>228</td>
<td>0</td>
<td>0</td>
<td>170</td>
</tr>
<tr>
<td>BTRAN</td>
<td>0.1627</td>
<td>6/29/04</td>
<td>1:33:19</td>
<td>1</td>
<td>26.8488</td>
<td>13.5036</td>
<td>13.3056</td>
<td>228</td>
<td>0</td>
<td>0</td>
<td>170</td>
</tr>
<tr>
<td>BTRAN</td>
<td>0.1323</td>
<td>6/29/04</td>
<td>1:33:43</td>
<td>1</td>
<td>26.6508</td>
<td>13.5036</td>
<td>13.1472</td>
<td>228</td>
<td>0</td>
<td>0</td>
<td>170</td>
</tr>
<tr>
<td>BTRAN</td>
<td>0.3901</td>
<td>6/29/04</td>
<td>1:34:10</td>
<td>1</td>
<td>27.72</td>
<td>13.7808</td>
<td>13.9392</td>
<td>228</td>
<td>0</td>
<td>0</td>
<td>170</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>27.0369</td>
<td>13.563</td>
<td>13.464</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
High performance DBAT capabilities

- High Performance DBATs reduce CPU consumption by:
  - Supporting RELEASE(DEALLOCATE) to avoid repeated package allocation/deallocation
  - Avoids processing to go inactive and then back to active
  - Bigger CPU reduction for short transactions
  - Targeted for users that do not use KEEPDYNAMIC(YES)

- High Performance DBAT behavior
  - DBAT will stay active with connection when DBAT is about to go be pooled and there is at least one RELEASE(DEALLOCATE) package allocated against thread
  - Connections will turn inactive after 200 times (not changeable) to free up DBAT
  - Normal idle thread time-out detection will be applied to these DBATs. If DBAT is in completed unit-of-work status, connection will turn inactive instead of being canceled
High performance DBATs

GLOBAL DDF ACTIVITY   QUANTITY
CUR ACTIVE DBATS-BND DEALLC   5.39
HWM ACTIVE DBATS-BND DEALLC   10.00

%CPU reduction from DB2 9

<table>
<thead>
<tr>
<th></th>
<th>COMMIT</th>
<th>High Perf DBAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDBC Type4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SQLJ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Native SQL proc</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Threadsafe

- Non-threadsafe
  - Programs run on the Quasi-reentrant (QR) TCB
  - TCB mode switch to the L8 occurs on each SQL/MQ command
  - Each TCB switch is approximately 2,000 instructions, a round trip (switch there then back) is around 4K.

- Threadsafe
  - Program starts on the QR TCB
  - SQL/MQ commands cause a TCB mode switch to the L8 TCB
  - Stay on the L8 TCB until a non-threadsafe CICS command is encountered
  - Non-threadsafe CICS commands switch back to the QR
  - SQL/MQ command is required to switch back to the L8
  - Can deliver much better CPU performance – IBM internal test show up to 22% for typical workload, in customer environment it varies from 5 to 10%
Threadsafe – new in CICS TS4.2

- CONCURRENCY(REQUIRED) primary to be used with API(CICSAPI)
  - CICS calls and RMI calls can run on a key 8 TCB regardless of the execution key
  - L8 TCB always used
  - Program starts on the L8 TCB
  - Threadsafe CICS calls run on the L8
  - OPENAPI TRUEs run on the L8 (DB2, MQ, sockets)
  - Non threadsafe CICS requests or TRUE calls switch to QR but switch back to L8 afterwards
  - Two TCB switches for each non-threadsafe CICS command, because control always returns to the application on the open TCB, so applications containing lots on non-threadsafe commands should not use REQUIRED
CONCURRENCY (THREADSAFE) vs (REQUIRED)

The program for this transaction is defined
CONCURRENCY(THREADSAFE) API (CICSAPI),
EXECKEY(USER) or EXECKEY(CICS)

The program for this transaction is defined
CONCURRENCY(REQUIRED) API (CICSAPI),
EXECKEY(USER) or EXECKEY(CICS)
Threadsafe – TCB switch tracking by CICS IA
Optimize PKLIST Search

- Within each collection (e.g. "COL_a.*, COL_b.*, COL_c.*"), efficient matching index access to find the package, but DB2 goes serially through the PKLIST entries
- Impact of long PKLIST search
  - Additional CPU resource consumption, catalog accesses, and elapsed time
  - Can aggravate DB2 internal latch (LC32) contention
  - Indicator of long PKLIST search

<table>
<thead>
<tr>
<th>PLAN/PACKAGE PROCESSING</th>
<th>QUANTITY</th>
<th>/SECOND</th>
<th>/THREAD</th>
<th>/COMMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>PACKAGE ALLOCATION ATTEMPT</td>
<td>26248.5K</td>
<td>5141.18</td>
<td>73.96</td>
<td>49.81</td>
</tr>
<tr>
<td>PACKAGE ALLOCATION SUCCESS</td>
<td>2693.1K</td>
<td>527.48</td>
<td>7.59</td>
<td>5.11</td>
</tr>
</tbody>
</table>

- Optimize
  - Reduce the number of collections on the PKLIST
  - Prioritize and reorder the collection sequence on PKLIST based on frequency of access
  - Use SET CURRENT PACKAGESET to direct the search to a specific collection
Lock Avoidance

- Benefits of lock avoidance
  - Increase in concurrency
  - Decrease in lock and unlock activity requests, with an associated decrease in CPU
- Lock avoidance may not be working effectively if Unlock requests/commit is high, e.g. >5

<table>
<thead>
<tr>
<th>LOCKING ACTIVITY</th>
<th>QUANTITY</th>
<th>/SECOND</th>
<th>/THREAD</th>
<th>/COMMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCK REQUESTS</td>
<td>521.0M</td>
<td>24.2K</td>
<td>3134.34</td>
<td>1050.75</td>
</tr>
<tr>
<td>UNLOCK REQUESTS</td>
<td>478.1M</td>
<td>22.2K</td>
<td>2876.06</td>
<td>964.16</td>
</tr>
</tbody>
</table>

- Plans and packages have a better chance for lock avoidance if they are bound with ISOLATION(CS) and CURRENTDATA(NO)
- High Unlock requests/commit could be possible from
  - Long-running URs
  - Large number of relocated rows after update of compressed or VL row
  - Large number of pseudo-deleted entries in unique indexes
  
  Both can be eliminated by REORG
Tools
CICS/DB2 performance optimization

- Optimize CPU usage and improve performance
  - Application optimization
    - Remove redundant or unnecessary logic
    - Separate high volume application
    - Program size
  - CICS / Language Tuning
    - APPL. options (LE & Compile options)
    - Thread reuse
    - Threadsafe
  - DB2 Tuning
    - Subsystem level performance tuning
    - SQL Access Path
CICS PA

Total CICS Response Time

Start

Suspend Time

Dispatch Wait

End

First Dispatch Wait

DSPDELAY

TCLDELAY

MXTDELAY

Other

JCIOWTT
TSIOWTT
TCIOWTT
FCIOWTT
TDIOWTT
IRIOWTT
EXWTTIME
RMISUSP
ENQDELAY
SZWAIT
LU61WTT
LU62WTT

Dispatched Time

USRDISPT

CPU Time

USRCPUT

PC Load

PCLOADTM

Involuntary MVS wait time

12/05/2012
CICS PA (cont.)

Various Report Templates

Statistics Alerts Automation

Graphical Report for Sharing

Most CICS experts are using
### DB2 PM

<table>
<thead>
<tr>
<th>TCB</th>
<th>SRB</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL processing</td>
<td>Asynchronous I/O</td>
</tr>
<tr>
<td>Synchronous I/O</td>
<td>Memory management</td>
</tr>
<tr>
<td>Buffer updates</td>
<td>Real-time Statistics</td>
</tr>
<tr>
<td>Global lock requests*</td>
<td>Castout*</td>
</tr>
<tr>
<td>GBP reads*</td>
<td>Physical logging</td>
</tr>
</tbody>
</table>

*The same as in TCB case, but only in enclave preemptible SRB mode.*

### Accounting
- Open, Close
- Extend
- Preformat

### Statistics
- Archiving
- Allocate archive dsn
- BSDS processing
- Error checking
- Management

!!! The same as in TCB case, but only in enclave preemptible SRB mode. Reported in TCB instrumentation. !!!
DB2PM (cont.)

<table>
<thead>
<tr>
<th>AVERAGE</th>
<th>APPL (CL. 1)</th>
<th>DB2 (CL. 2)</th>
<th>CLASS 3 SUSPENSIONS</th>
<th>AVERAGE TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELAPSED TIME</td>
<td>1:39.01072</td>
<td>0.000396</td>
<td>LOCK/LATCH (DB2+IRLM)</td>
<td>0.000000</td>
</tr>
<tr>
<td>NONNESTED</td>
<td>1:39.01072</td>
<td>0.000396</td>
<td>IRLM LOCK+LATCH</td>
<td>N/A</td>
</tr>
<tr>
<td>STORED PROC</td>
<td>0.000000</td>
<td>0.000000</td>
<td>DB2 LATCH</td>
<td>N/A</td>
</tr>
<tr>
<td>UDF</td>
<td>0.000000</td>
<td>0.000000</td>
<td>SYNCHRON. I/O</td>
<td>0.000017</td>
</tr>
<tr>
<td>TRIGGER</td>
<td>0.000000</td>
<td>0.000000</td>
<td>DATABASE I/O</td>
<td>0.000017</td>
</tr>
<tr>
<td>CP CPU TIME</td>
<td>0.003926</td>
<td>0.0000334</td>
<td>LOG WRITE I/O</td>
<td>0.000000</td>
</tr>
<tr>
<td>AGENT</td>
<td>0.003926</td>
<td>0.0000334</td>
<td>OTHER READ I/O</td>
<td>0.000000</td>
</tr>
<tr>
<td>NONNESTED</td>
<td>0.003926</td>
<td>0.0000334</td>
<td>OTHER WRTE I/O</td>
<td>0.000000</td>
</tr>
<tr>
<td>STORED PRC</td>
<td>0.000000</td>
<td>0.000000</td>
<td>SER. TASK SWTCH</td>
<td>0.000000</td>
</tr>
<tr>
<td>UDF</td>
<td>0.000000</td>
<td>0.000000</td>
<td>UPDATE COMMIT</td>
<td>0.000000</td>
</tr>
<tr>
<td>TRIGGER</td>
<td>0.000000</td>
<td>0.000000</td>
<td>OPEN/CLOSE</td>
<td>0.000000</td>
</tr>
<tr>
<td>PAR. TASKS</td>
<td>0.000000</td>
<td>0.000000</td>
<td>SYSLGRNG REC</td>
<td>0.000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EXT/DEL/DEF</td>
<td>0.000000</td>
</tr>
</tbody>
</table>
Acknowledgement

- Special Thanks to those who had contributed to this presentation
  - Akiko Hoshikawa (IBM SVL)
  - John Tilling (IBM Hursley)
  - SangSoo Han (IBM Korea)