Expert Stored Procedure Monitoring, Analysis and Tuning on System z

Steve Fafard, Product Manager, IBM
OMEGAMON XE for DB2 Performance Expert on z/OS

August 16, 2013
13824
Agenda

- What are stored procedures?
  - Benefits of stored procedures
  - Stored procedure analysis – Issues and solutions
- Monitoring stored procedures using OMEGAMON DB2 Performance Expert
- Isolating and tuning poorly performing stored procedures
What are stored procedures?

• A stored procedure is a user-written program that can be called by an application with an SQL CALL statement.
• It is a compiled program that is stored at a DB2 server
• It can execute business logic and SQL statements
• Stored procedure types
  • External high level language procedures COBOL, PL/I, C, C++, Assembler, REXX, and Java
  • External SQL procedures
  • Native SQL procedures introduced by DB2 9 for z/OS
DB2 z/OS stored procedure processing (External)
Native SQL procedure processing (Internal)
Programming benefits of stored procedures

• Modularity in application development
  • Data will be processed always in a consistent way according to the rules defined in the stored procedure

• Enforcement of business rules
  • Use stored procedures to define business rules that are common to several applications.
  • Can be an alternative to using constraints and triggers.

• Improved application security
  • Sensitive business logic runs on the DB2 server
  • End users are authorized to execute a stored procedure, and do not need table privileges. (Similar to static authorization model.)

• Application integration solutions
  • Can access non-DB2 resources, e.g. VSAM files, MQ queues, IMS or CICS transactions
  • Stored procedures can have access to commands that run only on the server.
Total cost of ownership benefits of stored procedures

• Reduced network traffic for distributed applications
  • Grouping SQL statements into a stored procedure results in two trips across the network for each group of statement, resulting in better performance for applications

• Cost of ownership reduction
  • If stored procedure is called from distributed client via DRDA, a portion is eligible for zIIP redirect.
    • Including: Call statement processing; Result set processing; Commit processing
  • Stored procedures written in Java can take advantage of zAAP engines
  • Native SQL procedures run as enclave SRB in DBM1 address space and the stored procedure execution itself is zIIP off-loadable with DB2 9 for z/OS.
  • For WLM managed stored procedures:
    • SQL processing runs under a TCB hence not eligible for zIIP redirect

• As of now, there is NO performance benefit for calling a stored procedure from a local application
## Stored procedure language / API CPU Cost comparison - Update

- IRWW workload (OLTP workload consisting of 7 transactions)
- Called from distributed JCC type 4 client

<table>
<thead>
<tr>
<th>Language / API</th>
<th>Base CPU / Tran Cost</th>
<th>Billable CPU / Tran Cost after zIIP and / or zAAP redirect</th>
</tr>
</thead>
<tbody>
<tr>
<td>COBOL stored procedure</td>
<td>1X (BASE)</td>
<td>0.80x (Some zIIP)</td>
</tr>
<tr>
<td>C stored procedure</td>
<td>1.02x</td>
<td>0.82x (Some zIIP)</td>
</tr>
<tr>
<td>SQLJ stored procedure</td>
<td>2.01x</td>
<td>1.11x (zAAP+ some zIIP)</td>
</tr>
<tr>
<td>JDBC stored procedure</td>
<td>2.97x</td>
<td>1.84x (zAAP+ some zIIP)</td>
</tr>
<tr>
<td>Native SQL stored procedure</td>
<td>1.09x</td>
<td>0.59x (Significant zIIP)</td>
</tr>
</tbody>
</table>
Performance reporting – External stored procedure

Client

Connect

CALL mySP (:p1)

Class 1 non-nested time (elap and CPU) spans connect to commit

Commit

Class 2 non-nested time (elap and CPU) records time in DB2 while not in an SP

DDF

CALL mySP (:p1)

Return

Class 1 nested time (elap and CPU) spans SP start to exit

Insert

Open

SQL1

Fetch to fill row buffer

Class 2 nested time (elap and CPU) records time in DB2 while executing an SP

DDF

DBM1

WLM
External stored procedure performance summary – plan level

- DB2 Accounting class 1 and 2 needed (3 is recommended)

<table>
<thead>
<tr>
<th></th>
<th>APPL (CL.1)</th>
<th>DB2 (CL.2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELAPSED TIME</td>
<td>0.003212</td>
<td>0.002575</td>
</tr>
<tr>
<td>NONNESTED</td>
<td>0.000714</td>
<td>0.000694</td>
</tr>
<tr>
<td>STORED PROC</td>
<td>0.002498</td>
<td>0.001881</td>
</tr>
<tr>
<td>UDF</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>TRIGGER</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>CP CPU TIME</td>
<td>0.000715</td>
<td>0.000654</td>
</tr>
<tr>
<td>AGENT</td>
<td>0.000715</td>
<td>0.000654</td>
</tr>
<tr>
<td>NONNESTED</td>
<td>0.000149</td>
<td>0.000129</td>
</tr>
<tr>
<td>STORED PRC</td>
<td>0.000567</td>
<td>0.000525</td>
</tr>
<tr>
<td>UDF</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>TRIGGER</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>PAR. TASKS</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
</tbody>
</table>

Class 1 non-nested time (ET & CPU)

Class 2 non-nested time (ET & CPU)

Class 1 nested time (ET & CPU)

Class 2 nested time (ET & CPU)
Performance reporting – Native SQL stored procedure

Class 1 non-nested time (elap and CPU) spans connect to commit and not executing an SP

Class 2 non-nested time (elap and CPU) records time in DB2 while not in an SP

Class 1 and 2 nested time (elap and CPU) records time in DB2 while executing an SP. All time is 'in DB2'.
Native SQL stored procedure performance summary - plan-level

- DB2 Accounting class 1 and 2 needed (3 is recommended)

<table>
<thead>
<tr>
<th></th>
<th>APPL (CL.1)</th>
<th>DB2 (CL.2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELAPSED TIME</td>
<td>0.004834</td>
<td>0.002789</td>
</tr>
<tr>
<td>NONNESTED</td>
<td>0.002819</td>
<td>0.000774</td>
</tr>
<tr>
<td>STORED PROC</td>
<td>0.002015</td>
<td>0.002015</td>
</tr>
<tr>
<td>UDF</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>TRIGGER</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>CP CPU TIME</td>
<td>0.000963</td>
<td>0.000909</td>
</tr>
<tr>
<td>AGENT</td>
<td>0.000963</td>
<td>0.000909</td>
</tr>
<tr>
<td>NONNESTED</td>
<td>0.000198</td>
<td>0.000143</td>
</tr>
<tr>
<td>STORED PROC</td>
<td>0.000765</td>
<td>0.000765</td>
</tr>
<tr>
<td>UDF</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>TRIGGER</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>PAR. TASKS</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
</tbody>
</table>

CL1 and CL2 will always be equal!
Stored procedure detail reporting - package level reporting

- Accounting class 7 and/or 8 needed

- SYSSTAT package contains time for CALL statement, result set processing, SET special registers, and VALUES statements for LOB handling

<table>
<thead>
<tr>
<th>SYSSTAT</th>
<th>VALUE</th>
<th>SYSSTAT</th>
<th>TIMES</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE</td>
<td>PACKAGE</td>
<td>ELAP-CL7 TIME-AVG</td>
<td>0.000387</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CP CPU TIME</td>
<td>0.000072</td>
</tr>
<tr>
<td>LOCATION</td>
<td>DSND91B</td>
<td>AGENT</td>
<td>0.000072</td>
</tr>
<tr>
<td>COLLECTION ID</td>
<td>NULLID</td>
<td>PAR.TASKS</td>
<td>0.000000</td>
</tr>
<tr>
<td>PROGRAM NAME</td>
<td>SYSSTAT</td>
<td>SE CPU TIME</td>
<td>0.000000</td>
</tr>
<tr>
<td>NSQLNEW</td>
<td>VALUE</td>
<td>NSQLNEW</td>
<td>TIMES</td>
</tr>
<tr>
<td>TYPE</td>
<td>PACKAGE</td>
<td>ELAP-CL7 TIME-AVG</td>
<td>0.004751</td>
</tr>
<tr>
<td>LOCATION</td>
<td>DSND91B</td>
<td>CP CPU TIME</td>
<td>0.001667</td>
</tr>
<tr>
<td>COLLECTION ID</td>
<td>USRT001</td>
<td>AGENT</td>
<td>0.001667</td>
</tr>
<tr>
<td>PROGRAM NAME</td>
<td>NSQLNEW</td>
<td>PAR.TASKS</td>
<td>0.000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SE CPU TIME</td>
<td>0.000000</td>
</tr>
</tbody>
</table>
Issues with plan and package level stored procedure analysis

- Multiple stored procedures called in a transaction are summed at the plan level. By definition this affects the analysis of nested SPs.

- Package level analysis can be difficult if a stored procedure executes different paths and SQL based on parameters. How do you differentiate between the invocations?

- Package level analysis does not apply to stored procedures that do not execute SQL.
Enhanced instrumentation for stored procedure performance analysis

- PM53243 (DB2 10) New IFCIDs 380 and 381 are created for stored procedure and User-Defined Function detail respectively. These records:
  - Identify the stored procedure or UDF beginning or end
  - Include the current CP, specialty engine, and elapsed time details for nested activity

- These record can be used to determine the CP, specialty engine, and elapsed time for a given stored procedure or UDF invocation
Enhanced instrumentation for stored procedure performance analysis

IFCID 380 written here for mySP begin. Will contain 0’s for current CP time, specialty engine time and elapsed times

IFCID 380 written here for mySP end. Will contain values that can be compared to the begin 380 record for mySP

Client

Connect

CALL mySP (:p1)

Commit

Fetch to fill row buffer

SQL1

Insert

SQL2

Open

Return

DDF

DBM1

WLM
Enhanced instrumentation for stored procedure performance analysis

• Additionally PM53243 (DB2 10) added IFCID 497, 498, 499 for SQL drill down analysis.
  • These records contain the dynamic or static statement IDs for non-nested, UDF, and SP work respectively.

• The statement IDs can be correlated to IFCID 316 dynamic statement or IFCID 401 static statement cache data.
Enhanced instrumentation for stored procedure performance analysis

**Client**
- Connect
- CALL mySP (:p1)
- Commit

**DDF**
- Fetch to fill row buffer

**DBM1**
- SQL1 Insert
- SQL2 Open
- Return

**WLM**

- IFCID 380 written here for mySP begin. Will contain 0’s for current CP, specialty engine and elapsed times
- IFCID 497 written here with all non-nested statement IDs executed (i.e., the CALL statement)
- IFCID 499 written here with all statement IDs executed in the SP (i.e., SQL1, SQL2)
- IFCID 380 written here for mySP end. Will contain values that can be compared to the begin 380 record for mySP
Monitoring stored procedures with OMEGAMON XE for DB2 Performance Expert

• These DB2 instrumentation records for stored procedures are evaluated by OMEGAMON, aggregated on a system level and returned to the repository server engine.
• OMEGAMON processing includes the sequencing logic and the calculation of elapsed times for the different accounting class times written in the IFI records as timestamps, considering nesting as well.
• In parallel the IFCID 316/401 data for the Statement Caches is collected and a correlation to the executed stored procedure statements via IFCID 499 is made.
Using the OMEGAMON DB2 Performance Expert web console to analyze stored procedures

- Workload:
SQL Dashboard – aggregation by ROUTINEID

- Workload at SQL dashboard (“All statements” view) executed in the selected time period (time slider), valid for all subsequent views

### Table: Statement Text and Execution Details

<table>
<thead>
<tr>
<th>Statement Text</th>
<th>Routine ID</th>
<th>Number of Calling Paths</th>
<th>Execution Elapsed Time</th>
<th>Number of Execs</th>
<th>CPU Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALL SYSIBM.SQLPROCEDURECOLS( IN VARCHAR, IN VARCHAR, IN ...</td>
<td>-2,147,483,102</td>
<td>1</td>
<td>1.160051</td>
<td>168</td>
<td>0.148540</td>
</tr>
<tr>
<td>CALL SPMON_CONF_IDD.FAMILY.V1( )</td>
<td>-2,147,482,976</td>
<td>1</td>
<td>0.403588</td>
<td>40</td>
<td>0.018785</td>
</tr>
<tr>
<td>CALL SYSPROC.ADMIN_COMMAND_DB2( IN VARCHAR, IN INTEGER, I...</td>
<td>-2,147,483,148</td>
<td>2</td>
<td>0.372614</td>
<td>13</td>
<td>0.060311</td>
</tr>
<tr>
<td>CALL SYSPROC.ADMIN_INFO_SYSPARM( IN VARCHAR, OUT INTEGER,...</td>
<td>-2,147,483,134</td>
<td>1</td>
<td>0.360020</td>
<td>2</td>
<td>0.033512</td>
</tr>
<tr>
<td>CALL SYSIBM.SQLPROCEDURES( IN VARCHAR, IN VARCHAR, IN VAR...</td>
<td>-2,147,483,101</td>
<td>1</td>
<td>0.268017</td>
<td>84</td>
<td>0.051966</td>
</tr>
<tr>
<td>CALL SPMON_CONF_IDD.DAUGHTER.V1( )</td>
<td>-2,147,482,977</td>
<td>2</td>
<td>0.142537</td>
<td>60</td>
<td>0.006785</td>
</tr>
<tr>
<td>CALL SPMON_CONF_IDD.GRANDCHILD.V1( )</td>
<td>-2,147,482,979</td>
<td>6</td>
<td>0.108440</td>
<td>164</td>
<td>0.003870</td>
</tr>
<tr>
<td>CALL SPMON_CONF_IDD.SON.V1( )</td>
<td>-2,147,482,978</td>
<td>2</td>
<td>0.083759</td>
<td>52</td>
<td>0.006528</td>
</tr>
<tr>
<td>CALL DB2MON_LOC.V1( OUT VARCHAR)</td>
<td>-2,147,482,972</td>
<td>1</td>
<td>0.023140</td>
<td>1</td>
<td>0.004644</td>
</tr>
</tbody>
</table>
Showing stored procedure details

CALL SPMON_CONF.FAMILY.V1()
Showing the calling paths of SPs (1/2)

• Select Calling Path for Daughter

∑ of Daughter(1) called by Family(0)

∑ of Daughter(0)

Select a calling path from the list to show the SQL statements that are executed in the context of the calling path.

Calling paths for:

CALL SPMON_CONF_IOD.DAUGHTER.V1()
Showing the calling paths of SPs (2/2)

- Select Calling Path for Grandchild

\[ \sum \text{ of } \text{Grandchild}(2) \text{ called by Family}(0)+\text{Son}(1) \]

\[ \sum \text{ of } \text{Grandchild}(2) \text{ called by Family}(0)+\text{Daughter}(1) \]

\[ \sum \text{ of } \text{Grandchild}(1) \text{ called by Family}(0) \]

Stored Procedure Calling Paths

Select a calling path from the list to show the SQL statements that are executed in the context of the calling path.

Calling paths for: CALL SPMON_CONF_IOD.GRANDCHILD.V1()

<table>
<thead>
<tr>
<th>Calling Path</th>
<th>Nesting Level</th>
<th>Number of Execut</th>
<th>Nested Elapsed T</th>
<th>Nested CPU T</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALL SPMON_CONF_IOD.FAMILY.V1() \ CALL SPMON_CONF_IOD.GRANDCHILD.V1()</td>
<td>1</td>
<td>40</td>
<td>0.059105</td>
<td>0.001066</td>
</tr>
<tr>
<td>CALL SPMON_CONF_IOD.FAMILY.V1() \ CALL SPMON_CONF_IOD.SON.V1() \ CALL SPMON_CONF_IOD.GRAND...</td>
<td>2</td>
<td>40</td>
<td>0.024042</td>
<td>0.002116</td>
</tr>
<tr>
<td>CALL SPMON_CONF_IOD.GRANDCHILD.V1()</td>
<td>0</td>
<td>12</td>
<td>0.020322</td>
<td>0.000746</td>
</tr>
<tr>
<td>CALL SPMON_CONF_IOD.DAUGHTER.V1() \ CALL SPMON_CONF_IOD.GRANDCHILD.V1()</td>
<td>1</td>
<td>20</td>
<td>0.001797</td>
<td>0.000783</td>
</tr>
<tr>
<td>CALL SPMON_CONF_IOD.FAMILY.V1() \ CALL SPMON_CONF_IOD.DAUGHTER.V1() \ CALL SPMON_CONF_IOD....</td>
<td>2</td>
<td>40</td>
<td>0.001689</td>
<td>0.000641</td>
</tr>
<tr>
<td>CALL SPMON_CONF_IOD.SON.V1() \ CALL SPMON_CONF_IOD.GRANDCHILD.V1()</td>
<td>1</td>
<td>12</td>
<td>0.001203</td>
<td>0.000524</td>
</tr>
</tbody>
</table>

Complete your sessions evaluation online at SHARE.org/BostonEval
Show SQL executed by a SP (1/2)

- Action: Show SQL for This Calling Path

Select of Family (0) shows:

\[
\sum \text{ of Call Son(1) called by Family(0)}
\]

\[
\sum \text{ of Call Daughter(1) called by Family(0)}
\]

\[
\sum \text{ of Call Grandchild(1) called by Family(0)}
\]

Select of Daughter(1) shows:

\[
\sum \text{ of Call Grandchild(2) called by Daughter(1)}
\]

Select of Grandchild(1) shows:

\[
\sum \text{ of Call Grandchild called by Grandchild(1)}
\]

- Action: Show SQL for All Calling Paths

Select of Grandchild() shows:

\[
\sum \text{ of Call Grandchild called by Grandchild()}
\]

\[
\sum \text{ of Call Grandchild called by Grandchild(1)}
\]

\[
\sum \text{ of Call Grandchild called by Grandchild(2)}
\]

* see next slide
Show SQL executed by a SP (2/2)

- Show SQL for This Calling Path for Family(0)
SQL cache correlation

- For a nested statement correlation to the cache is shown in “SQL Statements Details” area:

```
[Nesting Level 1] CALL SPMON_CONF.SON.V1( IN INTEGER)
```

![SQL Statement Details](image)
History navigator

- The History Navigator shows the drill down history for stored procedures and can be used similar to a Browser History.
Isolating and tuning stored procedures

- The poorly performing stored procedure has been identified and its performance analyzed using OMEGAMON XE for DB2 Performance Expert

- The next step is to isolate the stored procedure so that other workloads will not be adversely affected

- With the stored procedure isolated, then it can be tuned
Launch Optim Configuration Manager for z/OS
Navigate to the “Rule Set Manager” tab
Add Rule Set To Isolate Application

- Rule set: Isolate Application
- Name: Isolate Application - Rule
- Action: Isolate Application Transactions
- Client type: JCC

OK
Cancel
Define A Condition

Rules that isolate application transactions: Create client rules that isolate application transactions when problematic applications negatively affect performance, or new or modified applications require testing.

Properties

Conditions

Field: JndiWASDataSource Name
Value: Family Workload

Action

Review the conditions that specify the clients that are affected by this rule. You can edit the conditions by adding and removing clients or by filtering for clients. The managed connection identifier is always the first condition and cannot be removed.

WHEN

*serverName IS LABEC416.VMEC.SVL.IEM.COM AND portNumber IS 8000 AND databaseName IS STLECO1ALIAS*
Define An Action

IBM InfoSphere Optim Configuration Manager

<table>
<thead>
<tr>
<th>Rule Set Manager</th>
<th>Clients and Servers</th>
<th>Manage Aliases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RS - Isboc416 NONALIAS</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Isolate Application</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

Rules that isolate application transactions: Create client rules that isolate application transactions when problematic applications negatively affect performance, or new or modified applications require testing.

- **Save Rule**

**Properties**
- Specify the alias name, host, and port to which the applications or transactions will be routed. You may optionally choose an existing alias already identified on the server.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias Name</td>
<td>PENALTYBOX</td>
</tr>
<tr>
<td>Hostname/IP</td>
<td>LABBC418.vmecsdl.ibm.com</td>
</tr>
<tr>
<td>Port</td>
<td>8090</td>
</tr>
</tbody>
</table>
## Activate The Rule Set

### IBM InfoSphere Optim Configuration Manager

![IBM InfoSphere Optim Configuration Manager](image)

**Connection:** All managed connections

<table>
<thead>
<tr>
<th>Name</th>
<th>Active</th>
<th>Last Activated</th>
<th>Managed Connection</th>
<th>Managed Connection Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS - uitec730</td>
<td>No</td>
<td>CASEYM @ 2013-04-02 15:50:46.494</td>
<td>uitec730</td>
<td>v10 cm STLEC1</td>
</tr>
<tr>
<td>Isolate Application</td>
<td>No</td>
<td>CASEYM @ 2013-04-04 02:20:21.592</td>
<td>v10 cm STLEC1</td>
<td>DTEC297, VMIC, SVL, IBM.COM/446/STLEC1</td>
</tr>
<tr>
<td>RS - labec non alias</td>
<td>No</td>
<td>System @ 2013-04-02 12:23:45.131</td>
<td>labec non alias</td>
<td>LABC416, VMIC, SVL, IBM.COM/446/STLEC1</td>
</tr>
<tr>
<td>RS - abbc416 STLEC1</td>
<td>No</td>
<td>System @ 2013-04-02 12:23:43.921</td>
<td>labbc416 STLEC1</td>
<td>labbc416, VMIC, IBM.COM/446/STLEC1</td>
</tr>
<tr>
<td>RS - m10ec5</td>
<td>No</td>
<td>System @ 2013-04-03 16:35:19.592</td>
<td>m10ec5</td>
<td>m10ec5, VMIC, IBM.COM/446/STLEC1</td>
</tr>
<tr>
<td>RS - abbc416</td>
<td>No</td>
<td>CASEYM @ 2013-04-02 14:56:36.996</td>
<td>labbc416 STLEC1</td>
<td>LABC416, VMIC, SVL, IBM.COM/446/STLEC1</td>
</tr>
<tr>
<td>RS - fitec783</td>
<td>No</td>
<td>System @ 2013-04-02 12:23:42.229</td>
<td>fitec783</td>
<td>fabtec783, VMIC, IBM.COM/446/STLEC1</td>
</tr>
<tr>
<td>RS - testing</td>
<td>No</td>
<td>CASEYM @ 2013-04-03 10:32:57.036</td>
<td>testing</td>
<td>bono@79, tonsil, IBM.COM/446/TESTIN</td>
</tr>
</tbody>
</table>

- **Rules that isolate application transactions:** Create client rules that isolate application transactions when problematic applications negatively affect performance, or new or modified applications require testing.

**Rule:** Isolate Application

- **Managed connection:**

**Properties:** Specify the alias name, host, and port to which the applications or transactions will be routed. You may optionally choose an existing alias already identified on the server.

**Complete your sessions evaluation online at SHARE.org/BostonEval**
Tuning stored procedures at the SQL workload level

### Execution Summary

<table>
<thead>
<tr>
<th>Statement Text</th>
<th>Routine ID</th>
<th>Number of Calling Path</th>
<th>Number of Execution</th>
<th>Execution Elapsed Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT COUNT(*) AS GRANDCHILD1_EQ INTO :H1M FROM systems...</td>
<td>--</td>
<td>--</td>
<td>32</td>
<td>0.0061488</td>
</tr>
<tr>
<td>SELECT COUNT(*) AS GRANDCHILD2_LT INTO :H1M FROM system sys...</td>
<td>--</td>
<td>--</td>
<td>32</td>
<td>0.001917</td>
</tr>
<tr>
<td>SELECT COUNT(*) AS GRANDCHILD2_GT INTO :H1M FROM system sys...</td>
<td>--</td>
<td>--</td>
<td>32</td>
<td>0.001653</td>
</tr>
</tbody>
</table>

### SQL Statement Details

**Overview**

- Server Execution Times
- Row Activity
- I/O
- Locking and Communication

**Statement**

- Most Recent Identification
- Most Recent Compilation
SQL Workload is loaded into Optim Query Workload Tuner
Invoke advisors to generate expert tuning recommendations
Review advisor recommendations summary
Review specific advisor recommendations (Stats)
Further analysis such as plan comparison
Conclusion

- Stored procedures are a good application architecture and programming technique
- Certain types of stored procedures can save MIPs
- Can have many-to-many relationships between applications
- Have the potential to be reused millions of times daily

➤ A key monitoring and tuning opportunity
Thank you!
More information

• Websites
  • DB2 for z/OS home page
  • DB2 Tools for z/OS home page
  • Tivoli OMEGAMON XE for DB2 PE on z/OS home page
  • Optim Query Workload Tuner for z/OS home page
  • Optim Configuration Manager for z/OS home page
  • DB2 for z/OS: Information Roadmap

• Other resources
  • Online demo: stored procedure monitoring and analysis
  • eBook: Optimizing database performance through an integrated solution for DB2
Backup slides
Navigating to OMPE Extended Insight and the SQL Dashboard

Navigates to Extended Insight

Launches SQL Dashboard

Once launched, tabs are available to navigate between them.
Finally: Link to ‘Extended Insight’ functionality