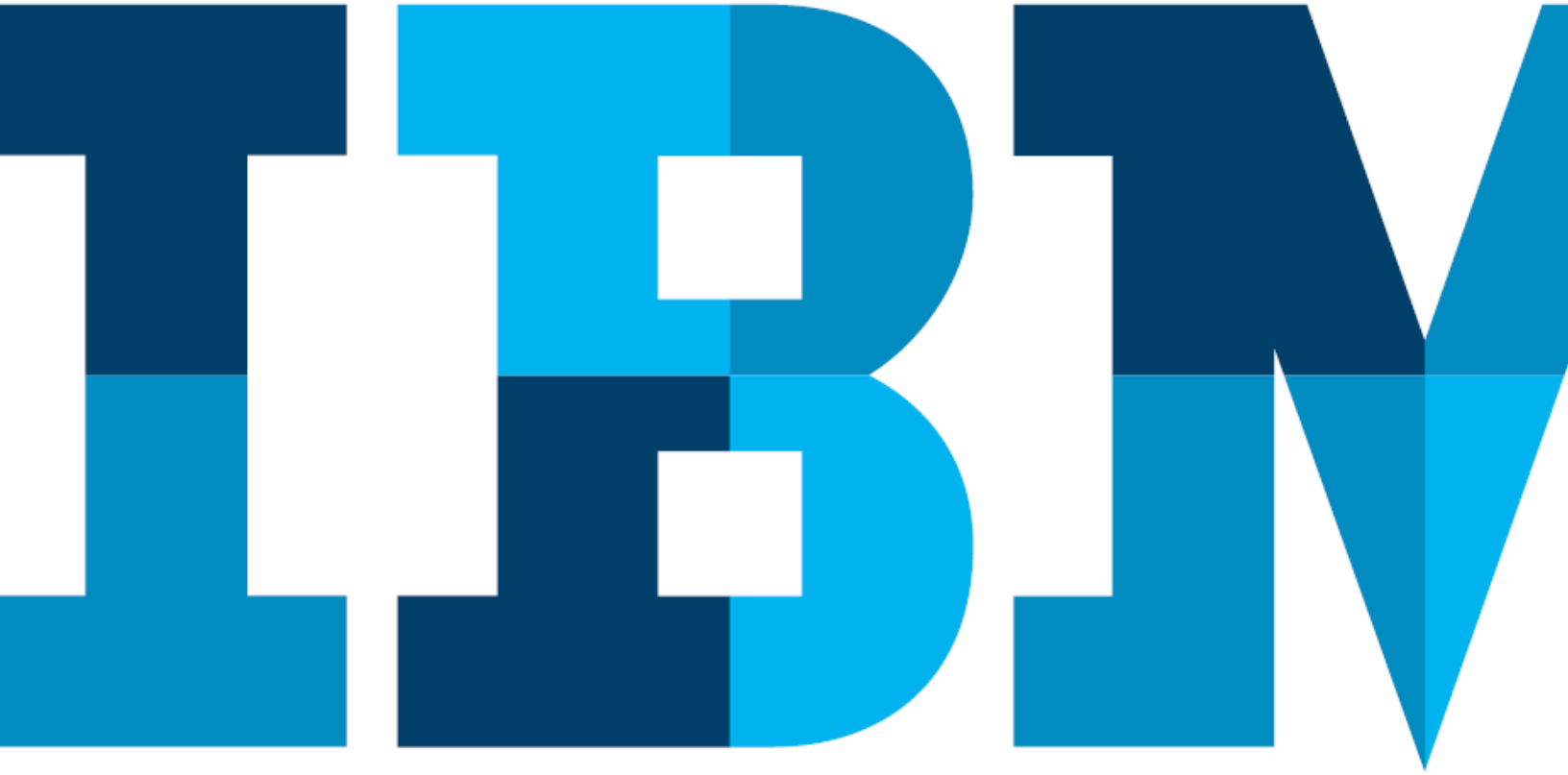
Expert Stored Procedure Monitoring, Analysis and Tuning on System z



IBM

Overview

This white paper addresses DB2® stored procedures, in particular how you can use the IBM® Tivoli OMEGAMON® XE for DB2 Performance Expert on z/OS to more effectively monitor, analyze and tune these procedures on System z®. It contains a brief overview of stored procedures in DB2, including their benefits and limitations. It then discusses how dedicated tools from IBM can help you monitor, analyze and effectively tune the stored procedures in your DB2 environment.

The architecture of stored procedures

Stored procedures are user-written programs either in a high-level language or the SQL procedures language. They may be compiled and stored at a DB2 server and can execute either Business Logic or SQL statements. There are two types of stored procedures in general.

- External stored procedures

  These are implemented using a high level language like COBOL, Java or C++, and typically used to exploit Business Logic in addition to invoking SQL statements. They are called “external” because they involve an external application program and load module.

- Internal stored procedures

  These can also be known as native SQL stored procedures and were introduced in DB2 9.

These two types of stored procedures have different execution architectures within DB2, which have great significance for any efforts to optimize their performance and cost. The execution architecture of an external stored procedure is shown in Figure 1.

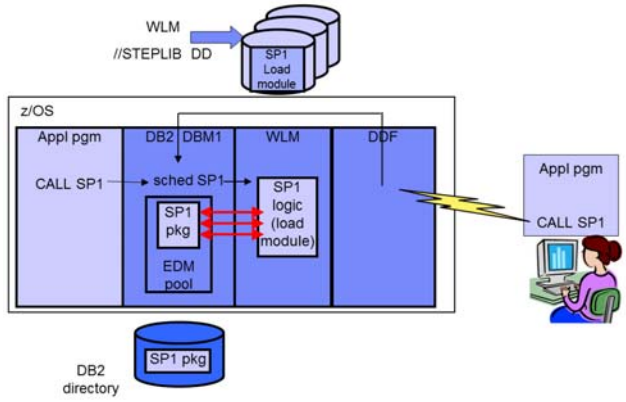


Figure 1: Overview of external stored procedure architecture

This figure shows a user executing a stored procedure from an arbitrary environment, such as via a web application on a remote Linux, UNIX or Windows based application server or via a z/OS native program (such as a CICS transaction). For distributed applications calling a stored procedure, the call arrives in the DB2 DBM1 address space through the distributed data facility (DDF) address space (shown on the right side of the figure). It is then scheduled in the DBM1 address space for execution. External stored procedures are running in the WLM

address space and their program logic/load module is loaded from an external load library that is referenced in the WLM address space STEPLIB.

As a consequence, all of the business logic and SQL statements will execute from within a workload-managed address space. Therefore this architecture features an interaction between the DB2 and workload-managed address space – indicated by the red lines – so there is an overhead associated with address space switch going on during the execution of the stored procedures. Note that, when the SQL is executed, the package is loaded directly from the DB2 directory.

As shown on the left side of the figure, the same stored procedure could have been invoked also from a z/OS local application, for instance one running in a CICS transaction, without changing the other aspects of the operation as described above.

Figure 2 shows the execution architecture of an internal, or "native," stored procedure and differentiates that from the previous example.

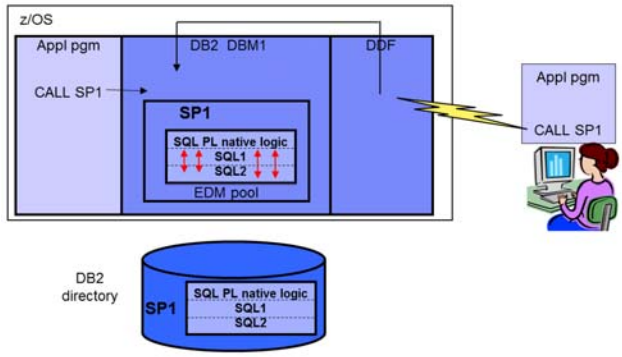


Figure 2: Overview of internal, or native, stored procedure architecture

As was the case with an external procedure, this internal stored procedure could be initiated from a remote client platform or from a local z/OS application. The major difference in the example shown in Figure 2 is that, all of the stored procedure logic is stored at the DB2 server in the DB2 directory and is loaded directly into the DBM1 address space once the execution starts. Therefore, there is no workload-managed address space associated with this execution model and no address space switching performance penalty is paid.

Furthermore, besides the fact that all the SQL activity occurs within the DBM1 address space, the stored procedure code itself is stored in the DB2 catalog as well.

The benefits of stored procedures

DB2 stored procedures can offer benefits to your organization. These can be divided into two main categories, as discussed in the following sections.

Programming benefits

The characteristics of stored procedures can provide your organization with several important programming benefits, such as the following:

- Modular application design

  For instance, a particular logic set can be reused in multiple execution environments.

- Consistent processing of data

  Data will always be processed according to the rules defined in the stored procedure.

- More reliable enforcement of business rules

  Stored procedures can replace ad hoc constraints and triggers.

- Increased application security

  Security clearance can be granted upon execution of the stored procedure, instead of requiring users to have privileges for the underlying DB2 objects.

- Easier integration of a set of data resources

  A stored procedure can encapsulate crucial access requirements such as non-DB2 resources by VSAM files or IMS transactions.

Total cost of ownership benefits

The second category of benefits relates to total cost of ownership. These can include the following:

- Reduced network traffic

  A set of SQL statements can be stored at the server and executed as a set with one network communication across the network. The results of that execution can then be transferred back to the calling client, thereby minimizing the overall network traffic.

- Direct reductions in cost of ownership

  In the case of internal or native stored procedure, the execution costs can be reduced further by redirecting a portion to a specialty engine called a System z9® Integrated Information Processor (zIIP).

This cost reduction potential is significant because everything in the transaction, including the call statement and results that are processed, may be eligible for specialty engine redirection, lowering the billable costs associated with that execution. Furthermore, if the stored procedure is executed or delivered in the Java programming language, DB2 provides the option of redirecting these Java execution costs onto a System z Application Assist Processor (zAAP) specialty engine, which can also lower the costs associated with this

execution. In the example of a native stored procedure, there is no workload managed processing.

To give context for this performance benefit, Figure 3 shows a comparison between several common execution languages and models. (Results may vary)

Language/API	Base CPU/Tran Cost	Billable CPU/Tran Cost after zIIP/zAAP redirect
COBOL Stored Proc	**1X (BASE)**	0.80x (Some zIIP)
C Stored Proc	1.02x	0.82x (Some zIIP)
SQLJ Stored Proc	2.01x	1.11x (zAAP+ some zIIP)
JDBC Stored Proc	2.97x	1.84x (zAAP+ some zIIP)
Native SQL Stored Proc	**1.09x**	**0.59x (Significant zIIP)**

*F*igure 3: Comparison of common execution languages and models, taken on a z10™ with DB2 10 environment

If the COBOL stored procedures form a baseline with execution cost "1," this chart compares it to various programming languages in terms of the base cost and the portion, if any, of that cost that might be redirected to a specialty engine.

A native stored procedure presents a clear opportunity to use the zIIP specialty engine to improve upon the billable execution cost by a factor of about 40 percent.

Figure 3 may be a useful guide to selecting an implementation language and execution model if performance is the key criterion.

Performance reporting tools

For external stored procedures, the DB2 instrumentation facility gives users the ability to trace elapsed and CPU times from an application perspective, and generate a performance reports using the data collected.

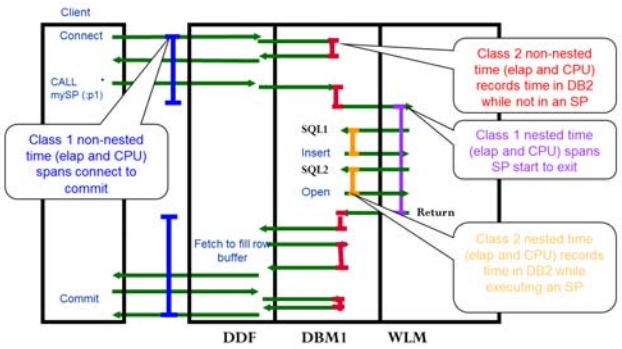


Figure 4: Performance reporting for external stored procedure

Figure 4 shows an overview of application costs in terms of application and DB2 elapsed and CPU times for an external stored procedure.

The so called "Class 1" times (reflected as blue and violet vertical lines) represent the times traced from either connect to commit of the application (Class 1 non-nested times), or the time spent from stored procedure start to end (Class 1 nested time), The term "nested" can be

understood as a “nested activity within DB2” (UDFs and Triggers are also considered nested activities).

For the Class 2 times the exact same categorization into non-nested and nested can be done (reflected as red and orange vertical lines), with the major difference that Class 2 times measure the time spent in DB2 itself.
In addition to the Class 1 and 2 times, by enabling further detail traces, Class 3 time traces, additional data can be collected which mainly details on the wait times that are part of the execution time in DB2 (these trace points are not shown in the figure).

Similar performance reporting is available for internal or native SQL stored procedure, as shown in Figure 5.

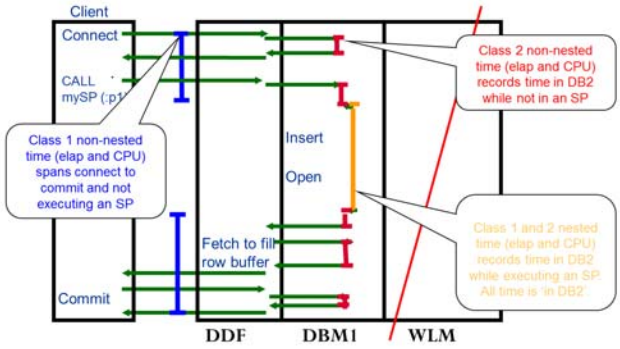


Figure 5: Performance reporting for native SQL stored procedure

The key difference in figure 5 compared to figure 4 is that due to the different execution architecture for native SQL stored procedures (mainly the absence of a WLM address space) the class 1 and class 2 nested times are identical, meaning that all of the native stored procedure execution time is in DB2.

Challenges of stored procedure track and trace

Tracking, tracing and analysis of stored procedures has traditionally included several well-known difficulties, each of which can affect the overall usefulness of the analysis efforts. The three most-relevant challenges are as follows:

- Single transactions can involve more than one stored procedure call.

  It is possible to have multiple stored procedures invoked in a single transaction. If these are summed at the plan level it is very difficult to analyze the details of any individual invocation of that stored procedure or a given stored procedure.

- Package-level analysis can be difficult if a stored procedure can be invoked by different paths.

  Package-level accounting detail makes it difficult to differentiate between scenarios where particular logic or data access is called directly, as opposed to being invoked as part of a stored procedure.

- Package-level analysis does not apply to stored procedures that do not execute SQLs.

For instance, a user trying to integrate VSAM and CICS transactions via a stored procedure in his/her application without the added stored procedure executing any SQL, would not be able to generate any useful information for package-level analysis, as DB2 does not create a package under these circumstances (i.e. no SQL coded in the stored procedure).

In order to address most of the challenges described above, expensive DB2 performance traces can be started, thus incurring high costs be it from a CPU consumption during the trace collection or being it for the time spent to analyze the potentially huge amount of data to analyze a problem. Such an approach might not be doable for the vast majority of users and therefore IBM looked at a smarter alternative to tackle the challenges indicated above.

The IBM solution

DB2 10 has introduced new instrumentation that users can use to trace additional information from a DB2 perspective. This additional information can be exploited in reports that the user can analyze and use as a basis for better tuning and analysis of the DB2 stored procedures.

New tracing records for more granular data

The above mentioned new DB2 10 instrumentation introduced a new IFCIDs 380 that is written at the beginning and the end of a stored procedure invocation. The data contents of this new trace record can be used to determine the nested class 1 and class 2 times of the stored procedure call and a further break down into time spent on a general purpose processor, or redirection to a specialty engine. This more granular information is helpful for more granular tuning of a DB2 stored procedures and can also assist in capacity planning questions for specialty engines.

The new DB2 10 instrumentation also keeps track of the SQL that is associated with the invocation of a stored procedure. This is accomplished by IFCID 499. IFCID 499 keeps a list of statement identifiers along with the statement type (dynamic or static) and the number of executions of the statement in the context of the stored procedure call.

This information and in particular the statement ID can be captured to correlate the SQL statements that are invoked by a DB2 stored procedure to the DB2 dynamic or static statement cache. Performance details that are tracked at the statement cache level can therefore be correlated back to a DB2 stored procedure implementation. The detailed information this provides can be used for detailed analysis of DB2 stored procedures.

Figure 7 shows some additional detail on how the 499 trace record is added to the IFCID 380 record.

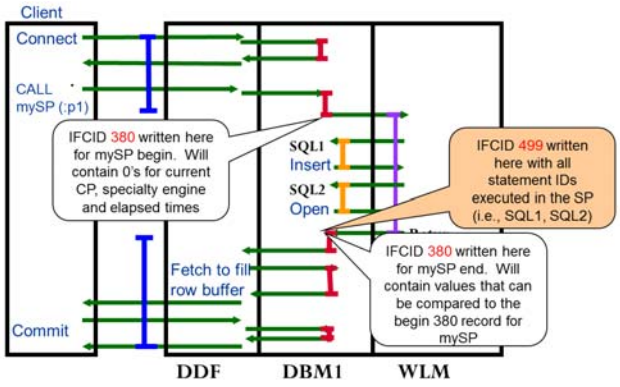


Figure 7: Additional detail provided by the 380 and 499 trace records

Referring to the figure above, when the stored procedure exits, a 499 trace record is written externalizing all of the SQL statements that were executed within the body of that stored procedure. An IFCID 499 record is also written before a nested stored procedure call is made in order to being able to track those SQL statements that have been run before the nested call starts and therefore to allow a "drill down" type of functionality.

After this information is gathered, a functional tool is needed to use it, and this functionality can be found in the Tivoli® OMEGAMON XE for DB2 Performance Expert for z/OS®. The OMEGAMON DB2 collector started task can accept the instrumentation trace records that are written by DB2, aggregates them on a system level and sends that information to the Repository Server component of OMEGAMON DB2. The collector sends also information on the calling path of a stored procedure to the Repository Server In correlating this new information with the information collected from the statement caches, the Web Console SQL Statements Dashboard allows for a detailed analysis for stored procedures.

With the combination of these new records and tools, users now have the ability to generate a full picture of what took place in the stored procedure invocations.

Example scenario

An example scenario can help illuminate these new granular capabilities, as well as the functionality embedded in the OMEGAMON XE for DB2 Performance Expert Web Console to help analyze that information. In our example we will analyze a workload called "family workload" where within the family stored procedure calls can be made to a nested stored procedure for the son, a daughter stored procedure and for a grandchild. Furthermore, from the son and daughter nested stored procedure calls can be made for a grandchild stored procedure. As shown in Figure 8, this results in a top level, a level one and a level two invocation all in this workload.

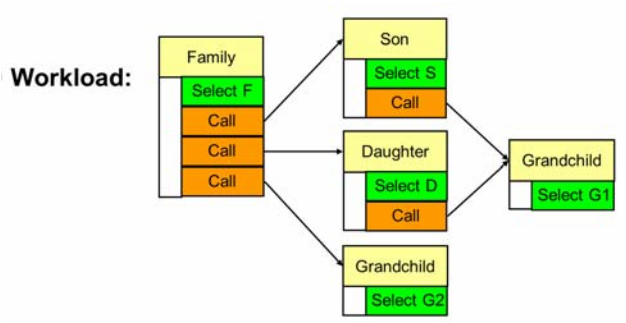


Figure 8: Example scenario of a "family workload"

The basics: The SQL dashboard

The existing SQL Statements dashboard in the Web Console can be used to analyze this family stored procedure workload scenario in greater detail. The initial analysis shows the SQL workload summarized across all of the statements in an "All Statements View" for a specified timeframe. A slider function in this dashboard can be used to specify the range of time within which DB2 stored procedures will be analyzed and thus provides a history functionality. Figure 9 shows an aggregation of all of the family workload stored procedures that were invoked during a specific timeframe selected in the slider function. The aggregation is done on a stored procedure call level (by ROUTINE ID) or statement level (by statement ID). Basically the system wide overall costs of all statement executions of stored procedure executions can be seen in this view.

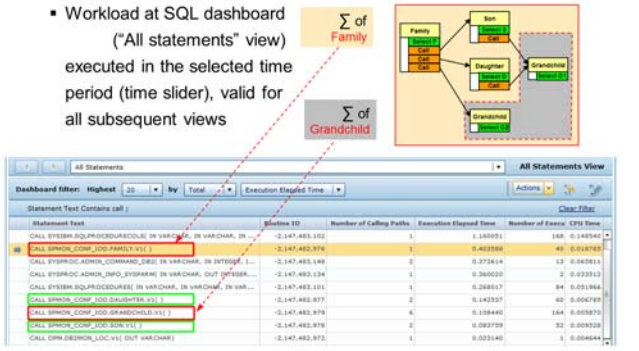


Figure 9: SQL dashboard aggregation by ROUTINE ID

Each particular statement text or stored procedure call can be highlighted and selected for further information, such as the number of times those statements have been invoked and how much CPU time and elapsed time is associated with the invocation. Especially for stored procedures the tool can also graph this data and show the various breakouts of how much of the time was spent on the general purpose processors and how much of it was dispatched on the z/OS specialty engines, thus externalizing the information collected via IFCID 380.

This additional level of detail is also available for a nested stored procedure; for instance if the daughter stored procedure is highlighted the tool can provide a summation of all of the SQL statements and the calling paths associated with the invocation of the daughter path, as shown in Figure 10.

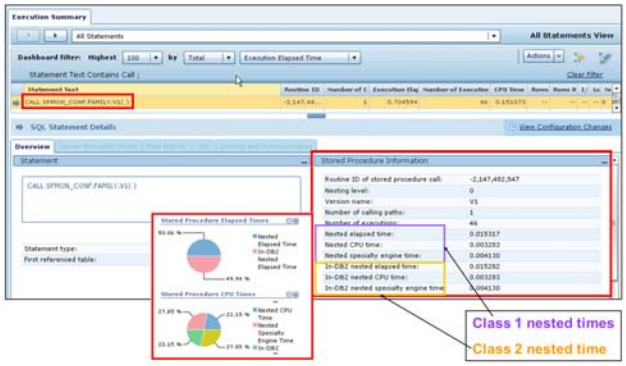


Figure 10: Additional detail about the calling paths to the daughter stored procedure

This particular procedure was called once via a path from the family stored procedure and called once directly. The direct call also included a call to the grandchild stored procedure from within the body of the daughter's stored procedure. This additional detail can show exactly how a stored procedure has been invoked and gives users the ability to differentiate among the various calling paths, thus resolving one of the issues of stored procedure analysis.

Better instrumentation in DB2 10 allows drill down analysis for stored procedures

The enhanced instrumentation in DB2 10 explained earlier makes it possible to correlate the information on the various calling paths to the underlying SQL activity that was executed or the nested stored procedures that were invoked. Therefore, as shown in Figures 11 and 12, users now can have the ability to differentiate among various invocations and see

1. What costs in terms of elapsed, CPU and specialty engine time a stored procedure execution has incurred in a specific calling path execution and
2. Exactly which SQL was invoked for which given invocation ("Show SQL for This Calling Paths"), or to aggregate the information to show the SQL for all of the various calling paths in the aggregate that were invoked ("Show SQL for All Calling Paths").

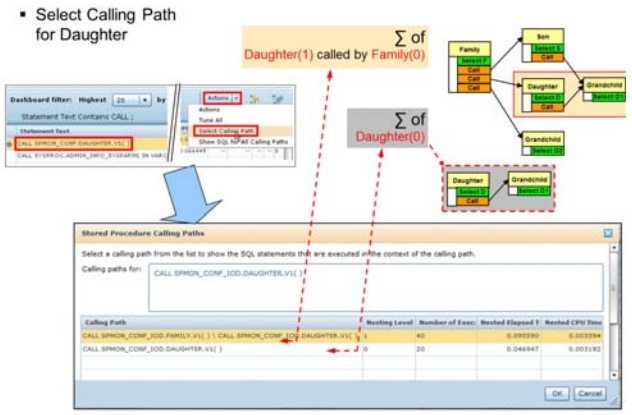


Figure 11: Showing SQLs unique to particular calling paths

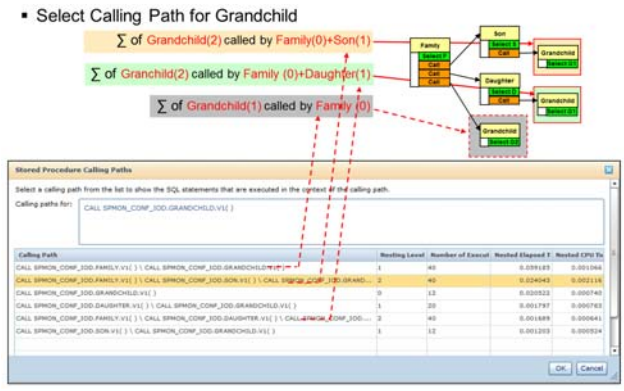


Figure 12: Additional detail of SQL for a family calling path

This additional granularity in SQL call information can then be correlated with the statement cache information for the various statements executed by the stored procedure to reveal details about the statement execution. If package-level accounting traces are enabled and accounting trace 3 or accounting trace 8 is active, further detail can be seen for locking and I/O activity associated with the statements that were invoked.

And finally, this entire analysis process is captured within a history navigator, which functions much like a web browser history, and can reveal the various levels of analysis and provide users the option to go back to one of the sets of analysis. Taken together, these additional levels of functionality provide a very powerful and flexible way to analyze and understand SQL statements of interest.

Using better reporting for better tuning

The enhanced DB2 instrumentation and the OMEGAMON DB2 reporting features described in previous sections of this document can help you identify the performance costs associated with specific stored procedures. After these are identified, the next logical step would be to use this information to perform one or both of the following tasks:

- Isolate any applications that contain poorly performing stored procedures so that their impact on the overall environment is minimized.
- Tune the SQL statements in the poorly performing stored procedures so that their performance is improved.

Additional components of the DB2 performance tools portfolio can be used to help achieve both of these goals.

Optim Configuration Manager for DB2 for z/OS

The IBM InfoSphere® Optim™ Configuration Manager for DB2 for z/OS can be launched directly from within the OMEGAMON XE for DB2 Performance Expert Web Console SQL statements Dashboard. Optim Configuration Manager can give you the flexibility to add rules as needed to isolate one or more workloads and define an action to take when that workload is invoked.

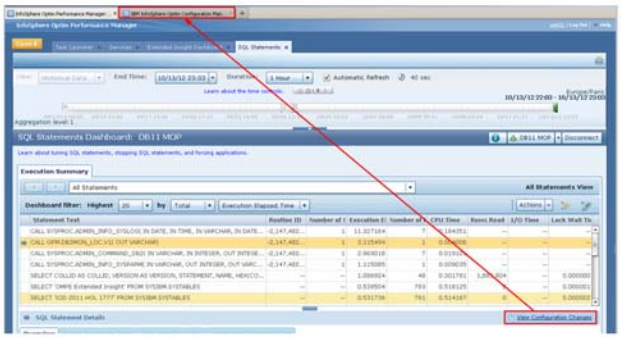


Figure 13: Using Optim Configuration Manager to add a rule set for workload isolation

When activated, this rule set implements the concept of a "penalty box" wherein a workload or application that has some problematic performance characteristics can be isolated away from the remainder of the normal workload that is achieving its throughput objectives. This isolation prevents the problematic workload from consuming an exorbitant amount of resources or using resources that should available for other activity.

However, the functionality of this tool goes beyond simple isolation. After the problematic workload is isolated, other DB2 portfolio capabilities can be used to tune and improve its performance.

SQL tuning

Returning back to the SQL Statements dashboard within OMEGAMON XE for DB2 Performance Expert, users can identify the slow performing SQL statements that are invoked in the body of a stored procedure, and use the "tune all" option in the Actions menu to tune them as a workload via the InfoSphere Optim Query Workload Tuner for DB2 for z/OS. As shown in Figure 14, the Optim Query Workload Tuner gives users the option to invoke a set of advisors that will analyze those SQL statements to find parameters that can be optimized – for instance, the advisors shown here will help ensure that all the necessary statistics and indexes are available, appropriate and up-to-date.

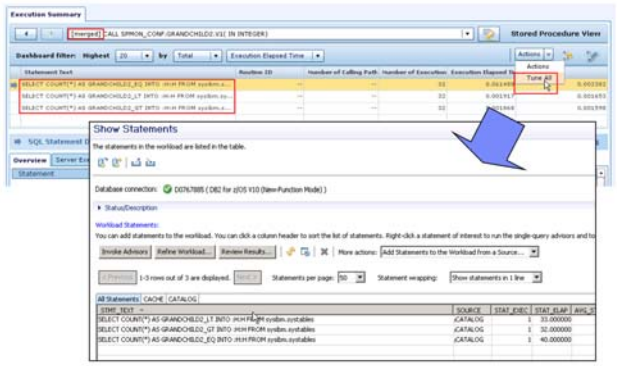


Figure 14: Advisor options within Optim Query Workload Tuner

After any selected advisors are run, the tool will return a set of recommendations for the workload. The user then has the option to view the specific advice that the Optim Query Workload Tuner provided for that workload. If the

user chooses to implement the recommended advice, it is possible to run comparative statistics on the same SQL workload to see how the performance of the workload changes after the tuning advice is implemented.

Taken together, the information and analysis tools discussed in this white paper can provide granular information to help you understand the performance of complicated stored procedure workloads and tune these workloads for better performance.

Conclusion

Complex workloads such as applications calling DB2 on z/OS stored procedures can be difficult to assess and properly tune. Effective tuning requires granular information about the performance of specific subroutines, which is not always available from conventional performance monitors. Intelligent, dedicated tools such as Tivoli OMEGAMON XE for DB2 Performance Expert, the Optim Query Workload Tuner and Optim Configuration Manager are designed to provide detailed insights into your workloads and help you tune them for better performance.

To learn more about useful tools for tuning stored procedures for DB2 on z/OS, please contact your IBM representative or IBM Business Partner, or visit the following website:
ibm.com/software/data/db2imstools/products/db2-zos-tools.html

About the Author

Matthias Tschaffler joined IBM at the Silicon Valley Lab in 2000. At SVL he worked on the “IBM DB2 Text Information Extender” and “IBM DB2 Warehouse Manager Connector for the Web” product deliverables. After transferring to IBM Germany in 2002, Matthias worked on the “EIP Information Mining” and “Information Integration for BPEL on WebSphere Process Server” products. He currently works as an Advisory Software Engineer in the “IBM Tivoli OMEGAMON XE for DB2 Performance Expert/Monitor on z/OS” team, where he acts as the technical development lead.

IBM Corporation
Software Group
Route 100
Somers, NY 10589

Produced in the United Stat es of America
May 2013