#SHAREorg

SHARE

Wooing Java Developers to SQL/PL Native Procedures

Carol Elliott
Shelter Insurance Companies
February 7, 2013
Session Number 12518

http://www.linkedin.com/profile/view?id=25501181&trk=tab_pro
@comocello

SHARE in San Francisco 2013

Level Set

- What is an SQL/PL Native Procedure?
  - SQL (Structured Query Language)
    +
  - PL (Procedural Language)
    = SQL/PL
  - Native = Runs Inside DB2 Engine
    +
  - [Stored] Procedure = Precompiled Code Routine Stored Within DBMS
    = Native Procedure

Level Set

SQL: special-purpose programming language designed for managing data in a relational database management system (RDBMS)

SQL/PL: A set of SQL statements introduced in DB2 UDB Version 7. Provides procedural constructs necessary for implementing control flow logic around traditional SQL queries and operations.

Supports comprehensive high-level programming in SQL.

SQL PL is a subset of the SQL Persistent Stored Modules (SQL/PSM) language standard. The specification of the current SQL/PSM standard can be found in ANSI/ISO/IEC 9075-4:1999 Information Technology, Database Language SQL, Part 4: Persistent Stored Modules (SQL/PSM).

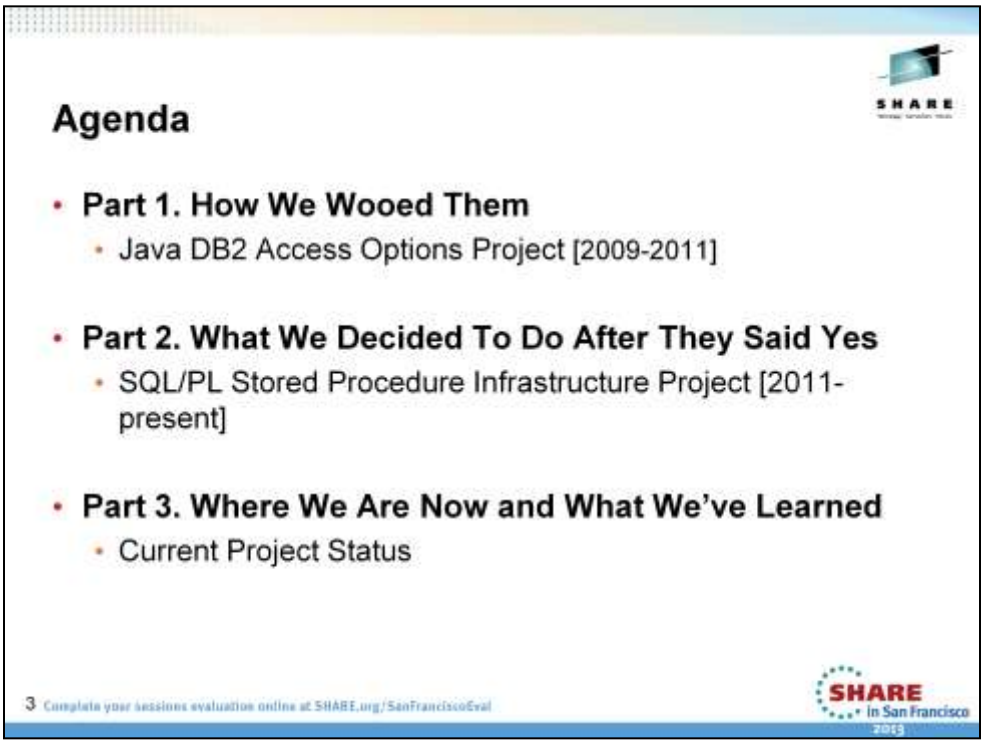
Native vs. External:

Native: Definition + Executable are stored in the DBMS

External: Definition in DBMS, Executable in File System

- zOS Library (COBOL)

- Unix file system (Java)



Today we'll talk about 3 major topics:

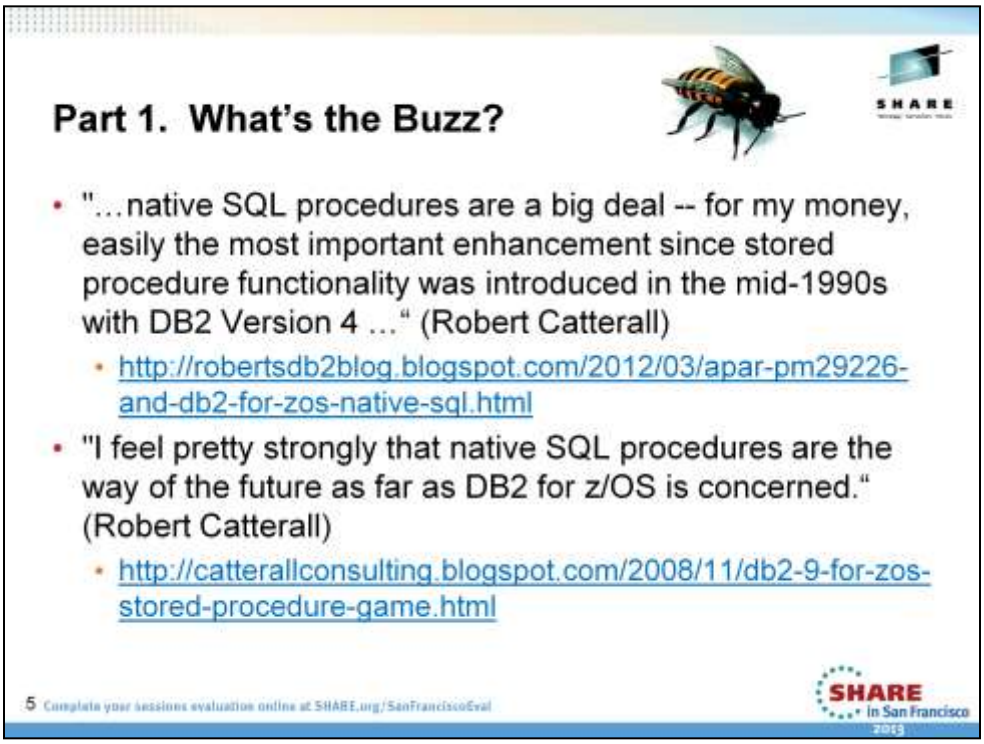
1. How we convinced Java developers to give SQL/PL native procedures a try.

2. What were the next steps after our DB2 access technologies throw-down, aka the Java DB2 Access Options Project

3. What is the current status of Shelter's adoption of SQL/PL Native Procedures?

Part 1. How We Wooed Them

- Java DB2 Access Options Project [2009-2011]



1. With this much buzz in the DB2 community about SQL/PL stored procedures, how could Java developers were so skeptical?

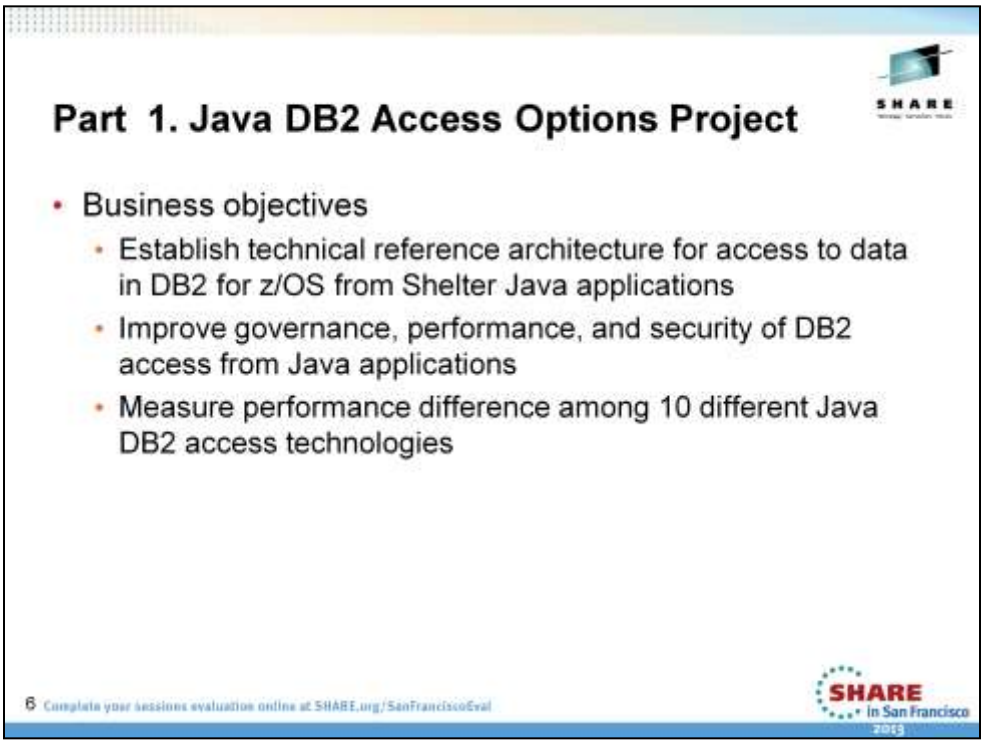
2. And how did we convince them to give SQL/PL stored procedures a try?

3. This presentation is really about two things:

- Our "discovery" of SQL/PL native stored procedures at Shelter when we were looking for alternatives to JDBC that would provide more predictable, governable performance then dynamic SQL

  and

- The story of taking a technology much touted by IBM and DB2 experts and meeting Shelter's requirements for enterprise-quality technology adoption.

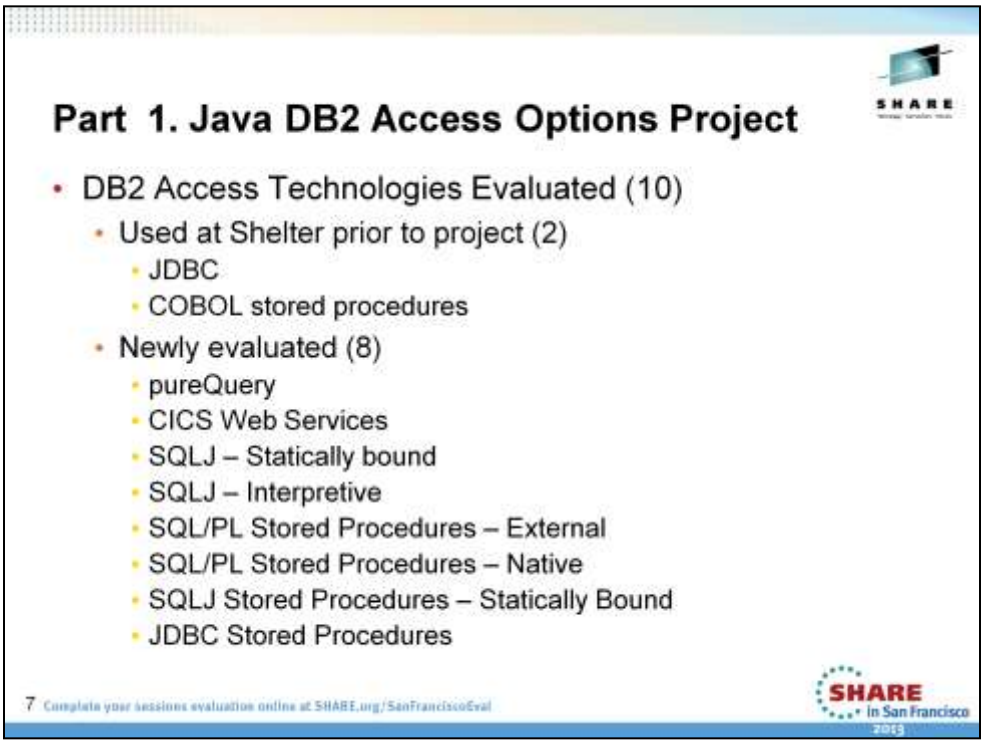


1. We began the Java DB2 Access Options Project in 2009.
2. The landscape at Shelter at the time included these factors:

- An interest in services and service-oriented architecture, including answering the question, What criteria should be used to judge the effectiveness of a particular technology used for service enablement?
- A few highly visible production issues where the performance of dynamic SQL coming from distributed Java applications surprised everyone (in a bad way) and people were put under great pressure to solve those issues in live fire .

  (Raise your hand if you've been a witness to something similar!)

3. These two factors influenced the business objectives for the Java DB2 Access Options Project



1. Why did we choose these technologies to evaluate?

- JDBC had been in use at Shelter since Shelter developed its first browser-based web application server application, around 1999
- COBOL stored procedures were adopted in the mid-2000s
    - We began exploring COBOL stored procedures because the approach chosen for the early JDBC applications was to replicate IMS/DB or denormalized DB2 structures into more normalized DB2 tables
    - The process of replication was becoming prohibitive and we wanted an alternative
- We wanted to check out the WSDL-based interface for CICS Web Services. Shelter did not have a large inventory of CICS transactions that it was interested in reusing.
- We heard a good deal about SQLJ in the mid- to late 2000s. We were interested in it because it allowed Java developers to stay within the Java API for SQL development (like JDBC) but with the potential advantage of the static SQL model.
- For the same reason we wanted to check out pureQuery (and of course IBM encouraged us to evaluate it).

Part 1. Java DB2 Access Options Project

- Java Developer Urban Legends
  - Java is cheaper
  - Java is faster
  - Java is better
  - Whatever it is, it is better done using Java
- JDBC accessing DB2 tables directly is the prevalent Java DB2 Access Option at Shelter
  - There is generally some bias toward the status quo

1. [At this point perhaps half the room is cheering or at least laughing and the other half is hissing.]
2. Does it sound like I have a chip on my shoulder about Java?!
3. Quite the contrary! I am a fan of Java and am even paying my own money to take online classes to learn Java! I want to be respected and look people straight in the eye and say words like “interface” “encapsulation” “inheritance” and “iterate.” Oh, and don’t forget “method signature,” “primitive,” and “enumeration.” Anyway ….

Part 1. Java DB2 Access Options Project

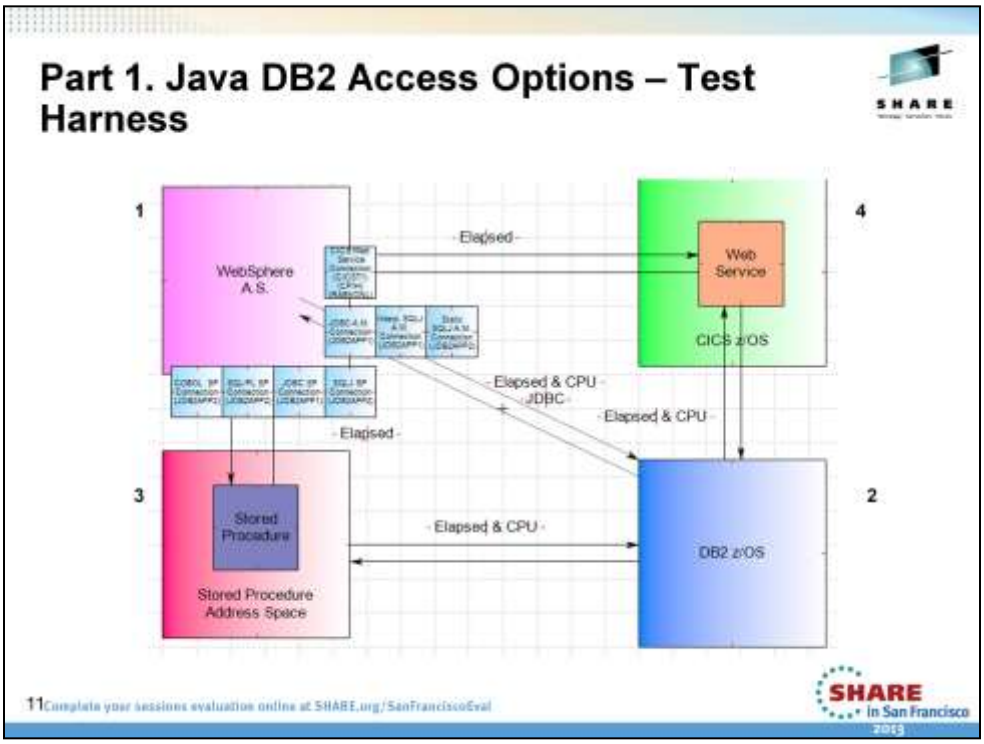
- Java Developer Objections to Stored Procedures
  - Frameworks don't support Object-Relational Mapping (ORM) of stored procedure result sets
  - Distrust of [COBOL] stored procedures
    - Java UNIT testing methods are perceived as more effective and thorough than COBOL UNIT testing methods
    - "I didn't build it, so I don't completely trust it"
  - Belief that JDBC code accessing DB2 tables directly is more efficient than JDBC code accessing stored procedures

1. Frameworks (such as Spring and Hibernate) are extremely important to Java developers. One of the advantages Java developers see in frameworks is that they can "introspect" relational tables and map tables and columns to objects. This is called object-relationship mapping ("ORM"). Apparently, frameworks do not support ORM of stored procedure result sets. My observation is that Java developers place high value on frameworks because frameworks are perceived to enable speed of development; and that, at least for some Java developers, speed of development can be valued above efficiency of execution.
2. Java developers have also objected to COBOL stored procedures because they are too "black box" – "I don't know what's going on inside it, so I don't trust it." Java developers place a high value on the effectiveness of Java methods of unit testing. Apparently JDBC works well with Java unit testing, and stored procedures are perceived to not work as easily.
3. With SQL/PL stored procedures, we wanted to be able to provide a stored procedure vehicle that Java developers would trust as much as they trust their own Java code.

Part 1. Java DB2 Access Options Project

- Proving or Disproving the Urban Legend
  - Customized "Test Harness" application
    - Systematically drive a workload consisting of all the Java DB2 Access Options to be compared
    - Formalized test scripts
  - Extreme scrutiny of SQL
    - Code SQL as identically as possible across different access options
  - Multiple measurement techniques

1. Perhaps we cannot address all the philosophical questions, but we can measure performance. To do this, we built a Test Harness application to drive our DB2 Access Options and enable fair and equal comparisons.
2. Test scripts
   - Run this combination of DB2 Access Options this many times.
   - Record results, start / stop times. That enabled us to correlate to SMF data for performance reporting.

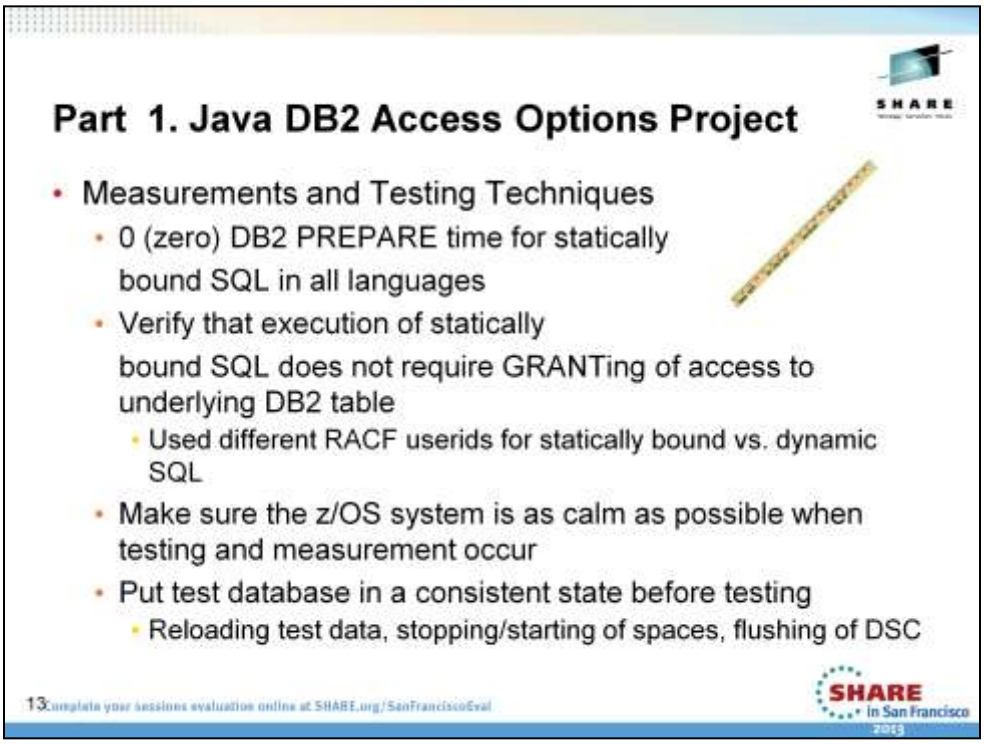


1. This diagram shows a conceptual diagram of the application we built to measure the 10 different access technologies: The Test Harness
2. Notice from the diagram:
    1. Test Harness on Web Sphere Application Server – distributed platform – in our case, Windows
    2. DB2 for z/OS
        1. JDBC talks directly to DB2 for z/OS tables
        2. JDBC talks directly to SQL/PL native stored procedures, which run within DB2
    3. External stored procedure address space (COBOL, JDBC, SQLJ, SQL/PL external stored procedures)
        - WLM-managed
        - Test Harness calls SP via JDBC. Stored Procedures talk to DB2 for z/OS tables
    4. CICS Web Services
        1. Test Harness speaks WSDL to CICS Web Services
        2. CICS region talks to DB2 for z/OS tables
    - We used different DB2 AUTHIDs depending on whether the DB2 Access Method used static or dynamic SQL
    - For dynamic SQL, the BINDer of the SQL statement requires SELECT, INSERT, UPDATE, DELETE authority to the DB2 object based on the statement it is BINDing

Part 1. Java DB2 Access Options Project

- Measurement Approach
  - Measure in multiple ways
    - To gain stakeholder trust
    - To see if results can be corroborated with different ways of measuring
  - Measurement techniques
    - DB2 Accounting Reports
    - CICS Performance Analyzer (CICS Web Services only)
    - ITCAM for WebSphere
      - *Method profiling*
    - "Self-measurement" by Test Harness

ITCAM = IBM Tivoli Composite Application Manager



Excerpt from Test Script to put test database in a consistent state:

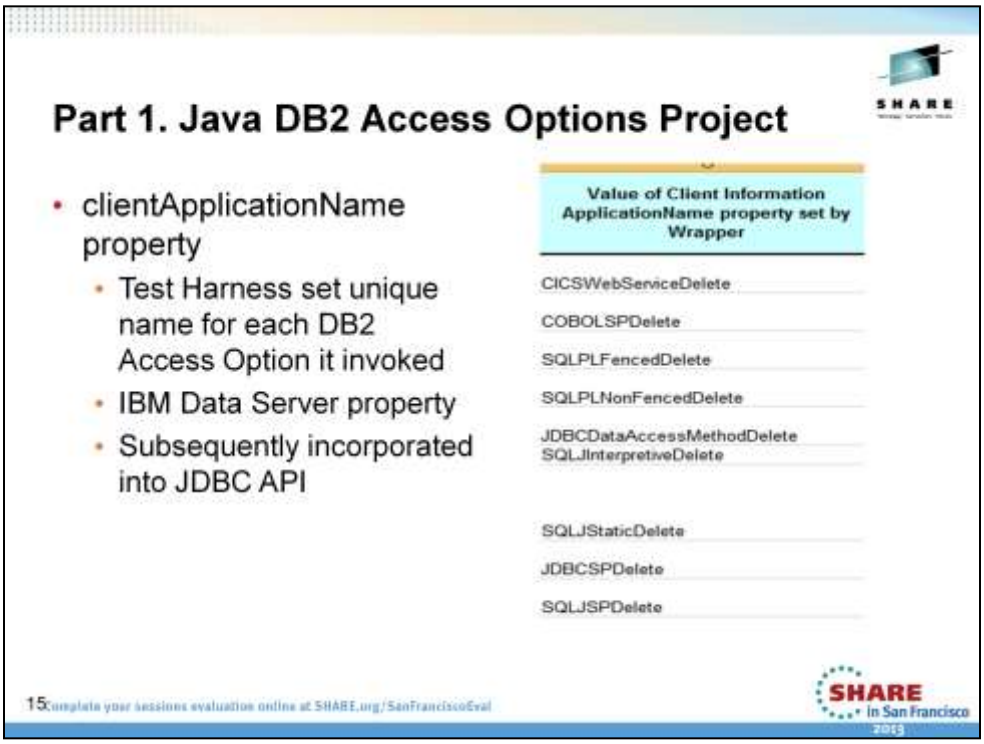
Instructions

1. For each of the 9 DB2 Access Methods, execute Retrieve Test 1 followed by Retrieve Test 2.
2. Before each pair of tests:
    a. Stop spaces
    b. Start spaces
    c. Run Access Database command to open spaces
    d. Run RUNSTATS UPDATE NONE REPORT NO to flush DSC

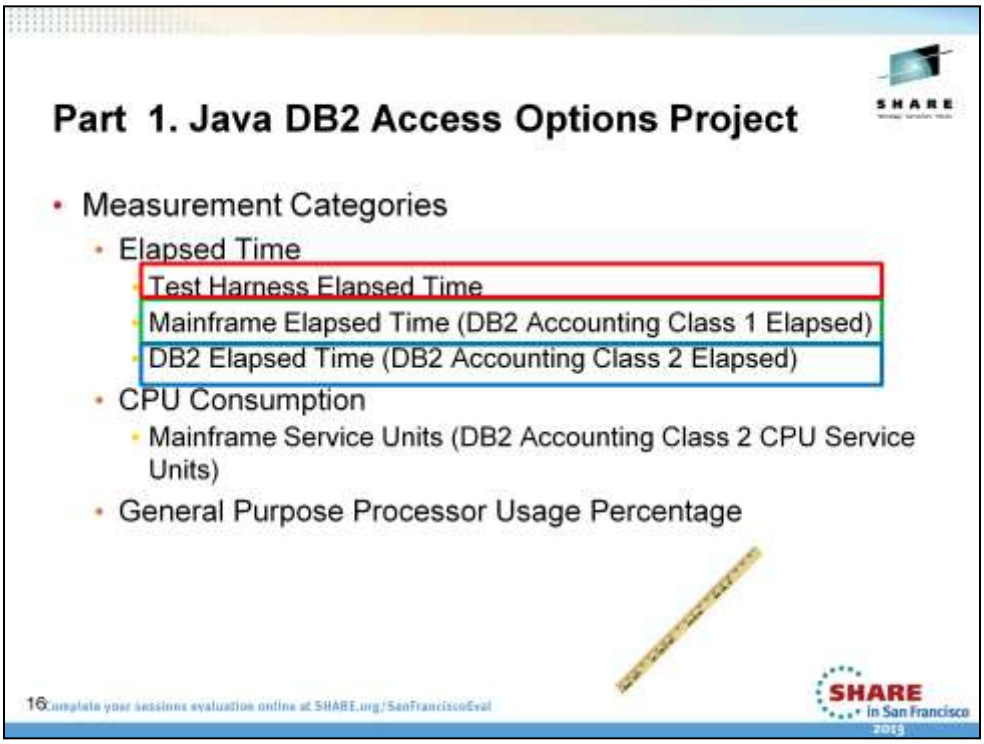
Part 1. Java DB2 Access Options Project

- Measurement Techniques
  - DB2 Accounting Reports
    - ORDER by TRANSACT
    - Omegamon XE for DB2 Performance Expert
  - Test Harness
    - Logging of elapsed time and identifying metrics to "Timer" csv file
  - ITCAM for WebSphere method profiling
    - We abandoned use of this product for measurement

1. We wanted to be able to compare DB2's version of Application Elapsed Time to the application's measurement.
2. There was an urban legend floating around that asserted that the application elapsed time did not necessarily correlate to the "mainframe" or DB2 elapsed time. We found that these two measurements were generally correlated although there were a couple of exceptions.



1. At our shop, the RACF administrators tend to prefer having fewer rather than more RACF userids. RACF userids tend to be at the application level rather than a more granular level.
2. This makes it difficult to measure, monitor, and isolate a particular function within a large application.
3. What we were trying to accomplish in this project was what we get easily with statically bound applications, where it's very easy to measure/monitor at the DB2 package level.
4. We discovered the clientApplicationName property of the IBM data server driver. A Java application can set this value which is stored in DB2's accounting data. In turn, DB2 Accounting Reports can then be used to group reporting by this property. The information how to set this property is documented in the DB2 Application Programming Guide for Java.
5. The ability to set this property was subsequently incorporated into many of Shelter's Java DB2 applications since it was so useful in allowing us to report on more granular pieces of our applications.
6. In a subsequent project, one of our developers discovered that this (formerly IBM proprietary) feature was moved into JDBC API.

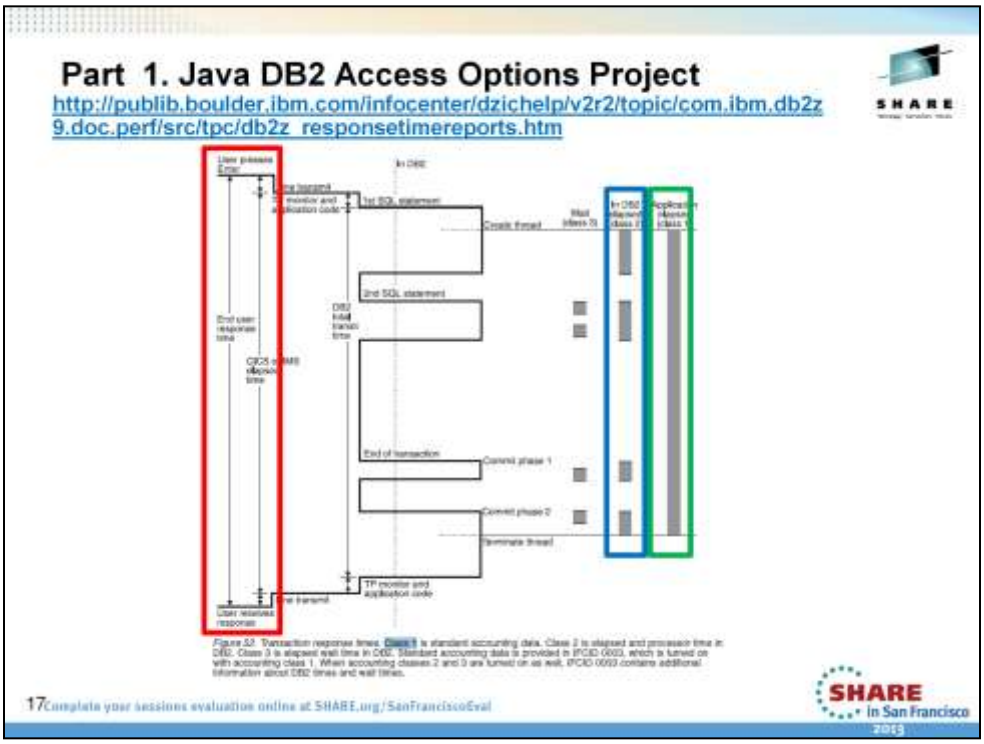


1. We measured 3 components of Elapsed Time
    1. Test Harness, DB2 Class 1, and DB2 Class 2
    2. See DB2 Elapsed Time chart on next slide
2. CPU time
    1. We wanted to see how DB2 CPU time compared among the three options
    2. We were not interested in comparing the different application languages

    or transaction managers.

    3. We were interested in seeing which of our applications used the General Purpose Processor the least,

    so we computed the percentage use of general processor for each DB2 Access Option

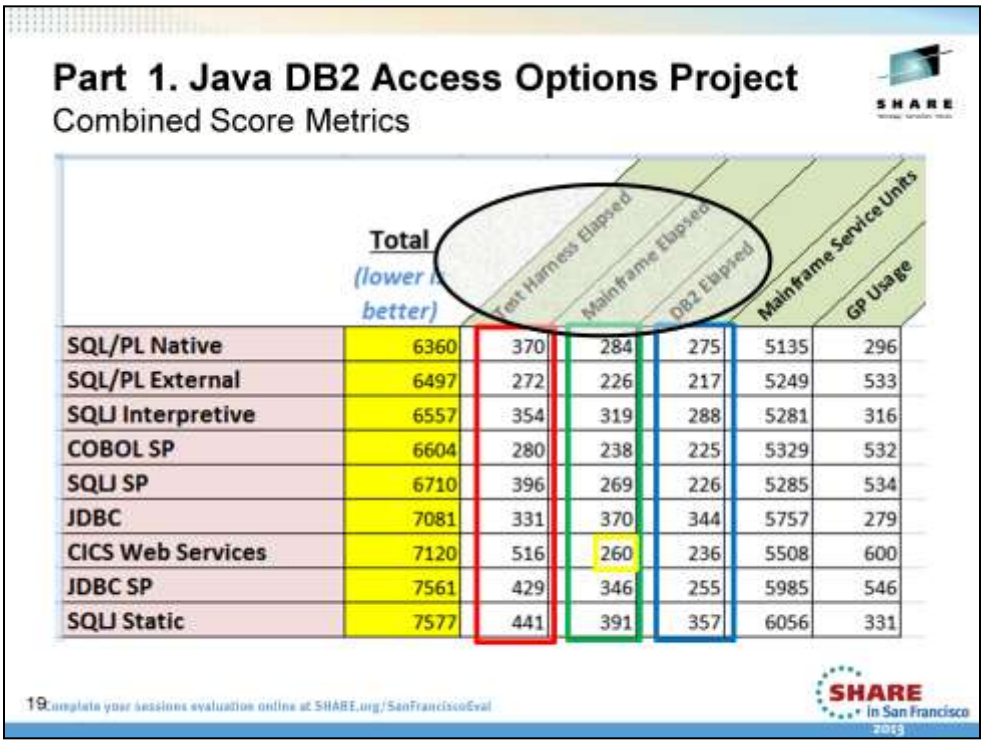


1. This is the definitive IBM diagram explaining the components to user response time for DB2 applications
2. Class 1 (Application Elapsed Time) is shown in green
    1. This includes the time from when the application connects to DB2 until it disconnects.
    2. Class 1 time includes Class 2 time
3. Class 2 (DB2 Elapsed Time) is show in blue
    1. This is the time spent executing SQL statements

Part 1. Java DB2 Access Options Project

Explanation of Metrics (Part 1 of 3)

- Elapsed Time Metrics
  - Test Harness Elapsed Time
    - Average elapsed time in milliseconds (1/1,000 of a second) for the execution of a transaction
    - Calculated by the Test Harness application using the Java API and averaged using Microsoft Excel functions
  - Mainframe Elapsed Time (DB2 Accounting Class 1 Elapsed)
    - Average elapsed time in milliseconds
    - Time application is connected to DB2
  - DB2 Elapsed Time (DB2 Accounting Class 2 Elapsed)
    - Average elapsed time in milliseconds
    - Time spent performing SQL statements



Elapsed Time Metrics

Test Harness Elapsed Time

Average elapsed time in milliseconds (1/1,000 of a second) for the execution of a transaction

Calculated by the Test Harness application using the Java API and averaged using Microsoft Excel functions

Mainframe Elapsed Time (DB2 Accounting Class 1 Elapsed)

Average elapsed time in milliseconds

Time application is connected to DB2

DB2 Elapsed Time (DB2 Accounting Class 2 Elapsed)

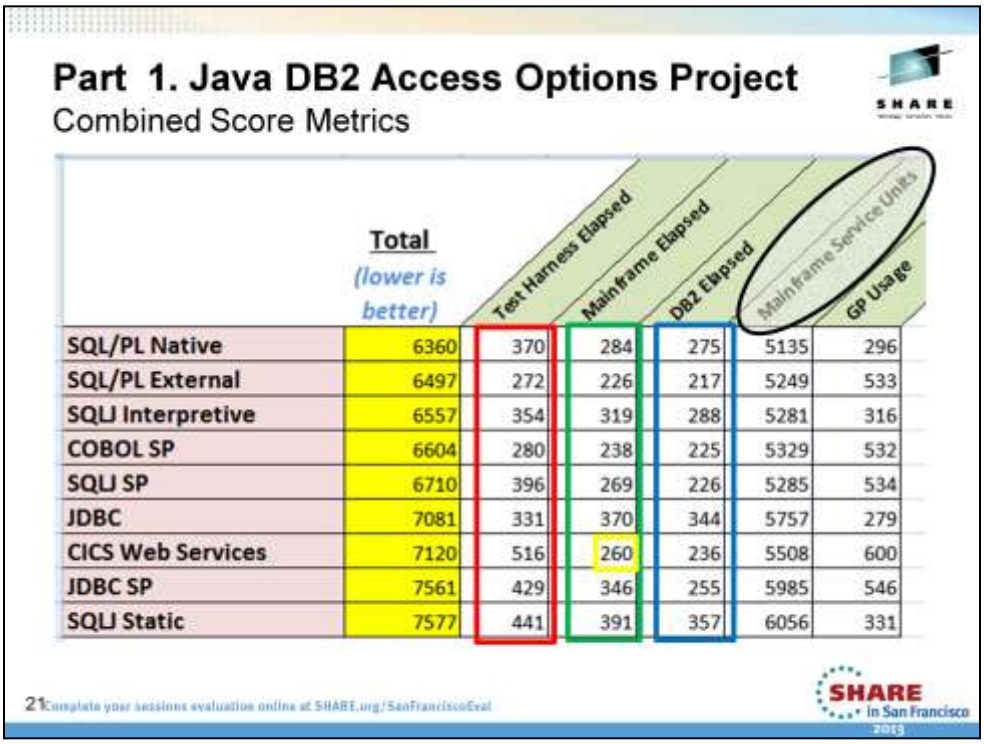
Average elapsed time in milliseconds

Time spent performing SQL statements

Part 1. Java DB2 Access Options Project

Explanation of Metrics (Part 2 of 3)

- CPU Consumption Metric
  - Mainframe Service Units
    - A measure of the CPU used performing SQL statements in DB2
    - Service units are a way of measuring CPU usage in z/OS in a consistent way that is independent of processor model or CPU speed.
    - CPU time varies across processor model, while number of service units should remain constant across process models for a given workload
    - DB2 Accounting Long Report, DB2 Class 2 Service Units
    - The lower the Mainframe Service Units, the less CPU the application uses
    - Therefore, Mainframe Services Units are a measure of *efficiency* (CPU resource usage)



Part 1. Java DB2 Access Options Project

Combined Score Metrics

	Total (lower is better)	Test Harness Elapsed	Mainframe Elapsed	DB2 Elapsed	Mainframe Service Units	GP Usage
SQL/PL Native	6360	370	284	275	5135	296
SQL/PL External	6497	272	226	217	5249	533
SQLJ Interpretive	6557	354	319	288	5281	316
COBOL SP	6604	280	238	225	5329	532
SQLJ SP	6710	396	269	226	5285	534
JDBC	7081	331	370	344	5757	279
CICS Web Services	7120	516	260	236	5508	600
JDBC SP	7561	429	346	255	5985	546
SQLJ Static	7577	441	391	357	6056	331

CPU Consumption Metric

Mainframe Service Units

A measure of the CPU used performing SQL statements in DB2

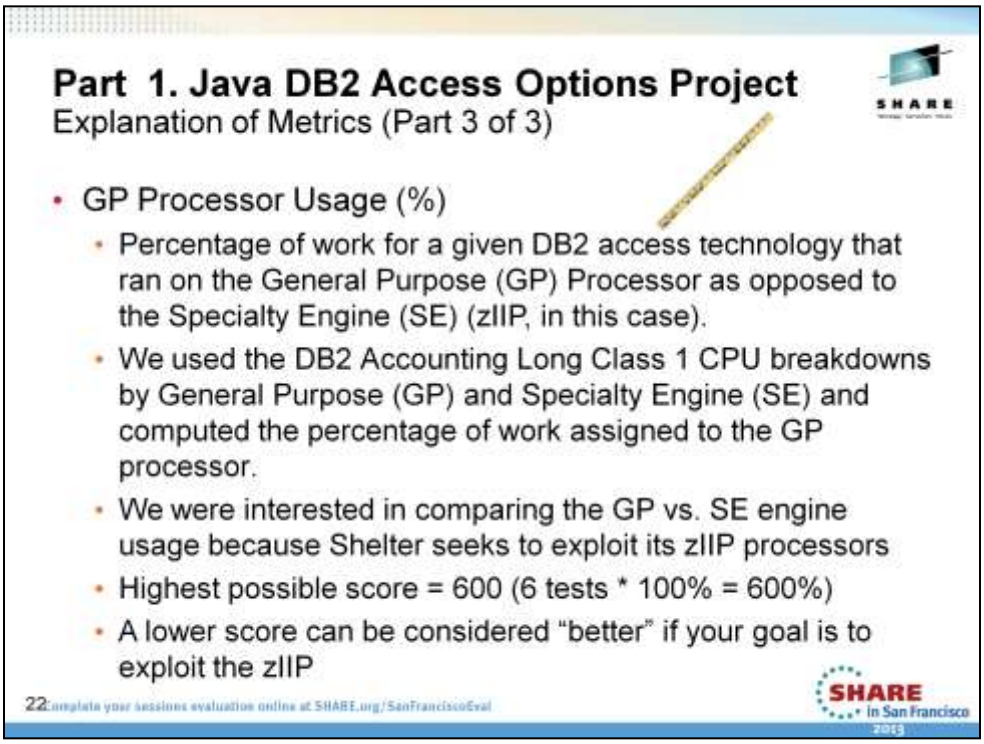
Service units are a way of measuring CPU usage in z/OS in a consistent way that is independent of processor model or CPU speed.

CPU time varies across processor model, while number of service units should remain constant across process models for a given workload

DB2 Accounting Long Report, DB2 Class 2 Service Units

The lower the Mainframe Service Units, the less CPU the application uses

Therefore, Mainframe Services Units are a measure of *efficiency* (CPU resource usage) or *performance*



1. Use of specialty engines can be important to enterprises using z/OS because of their cost model.

2. An enterprise pays for a specialty engine once. There is no capacity-based licensing model for specialty engines as there is for general purpose processes.

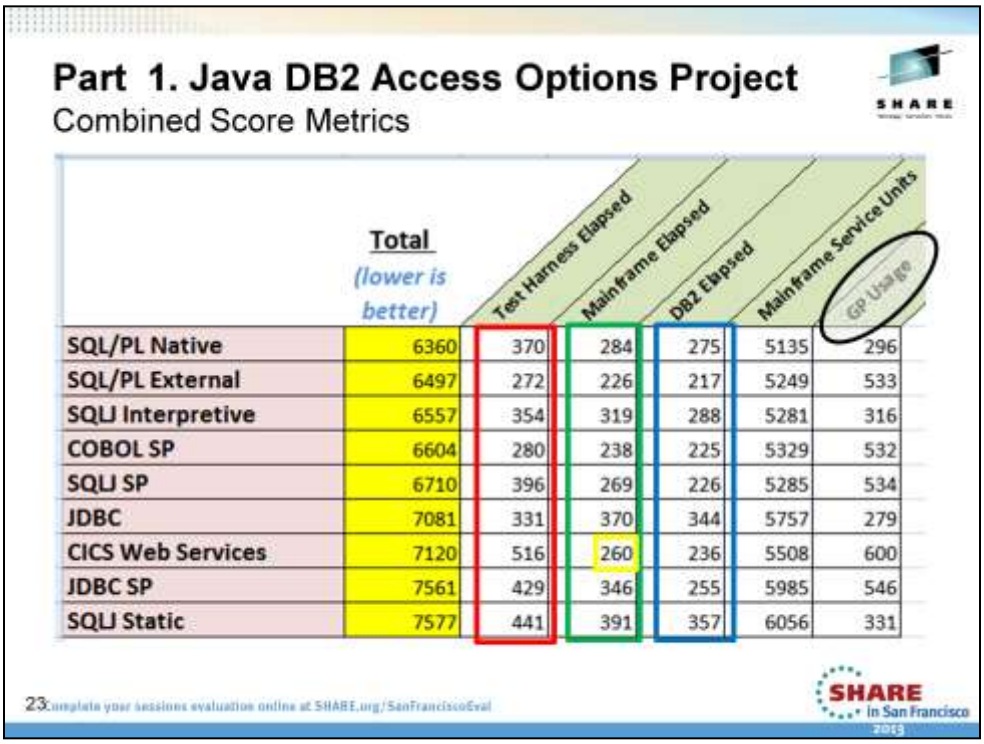
3. A capacity-based licensing model can be thought of as a variable cost because as the enterprise increases (or decreases)

its available mainframe computing capacity, software licensing charges will also increase (or decrease).

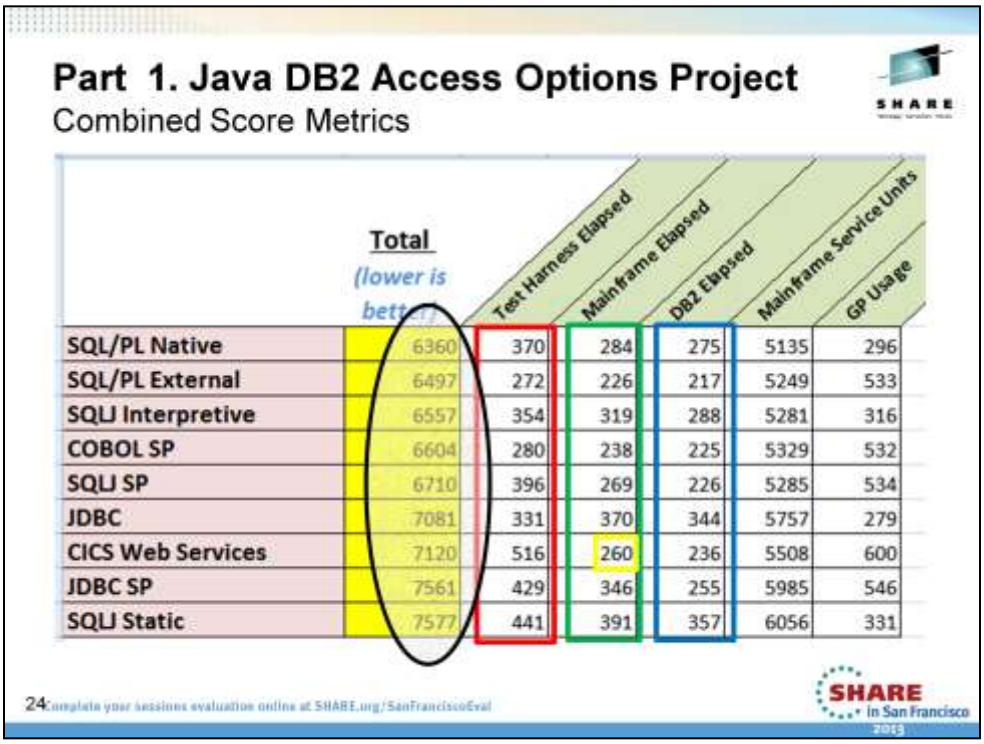
4. On the other hand, work running on the specialty engine incurs a fixed cost, the one-time cost of procuring the specialty engine processor.

Note: I have found there is a tendency to confuse *efficiency of execution* with *cost of computing*.

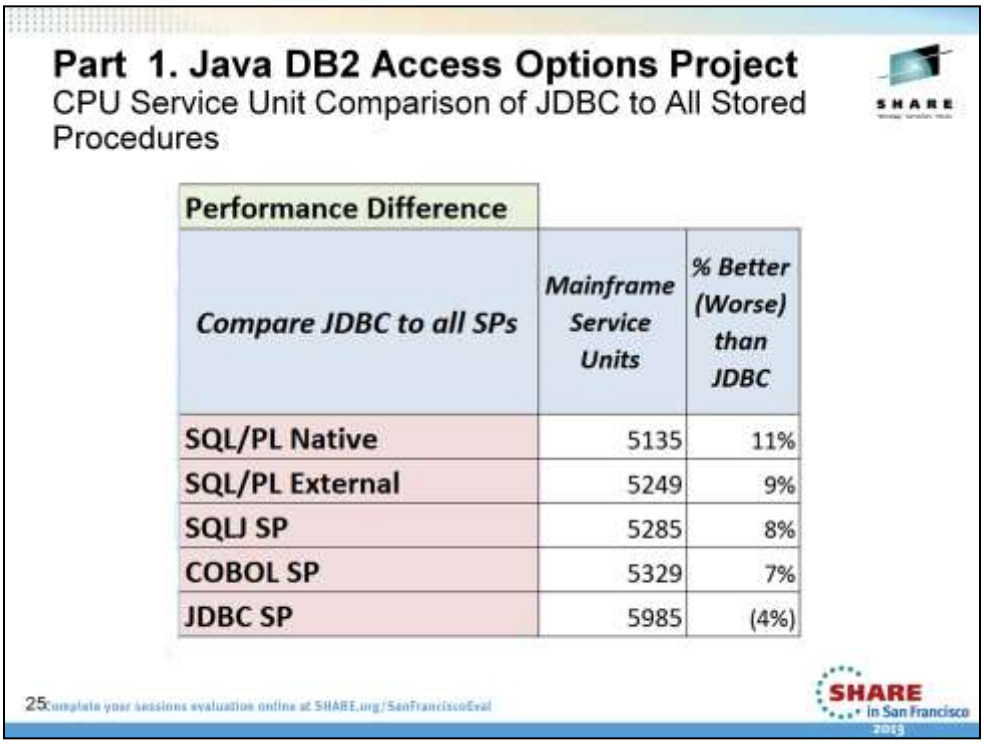
• Remember, regardless of whether work runs on the specialty engine or the general purpose processor, the lower the total Mainframe Service Units, the less CPU the work uses, and the more efficient it is.



1. Highest possible score = 600 (6 tests * 100% = 600%)
2. A lower score can be considered “better” if your goal is to exploit the zIIP
3. Notice:
    a. CICS had the highest GP usage – all its workload ran on the GP Processor.
    b. JDBC had the lowest GP usage followed by SQL/PL native procedures



1. The idea for a “Combined Score Metric” came from one of my co-workers who had worked for the public schools in our town.
2. She told me that teachers would use a similar approach as an alternative to assigning percentages for assignments. The idea is that every assignment receives a certain number of points which are summed to provide an overall student score for a set of assignments for the grading period.
3. Here we are combining dissimilar measurements to provide a combined score.
4. The overall goal is to be able to provide a measure of “overall goodness”
5. Lower is better:
   - Elapsed time
   - Service Units
   - GP Usage

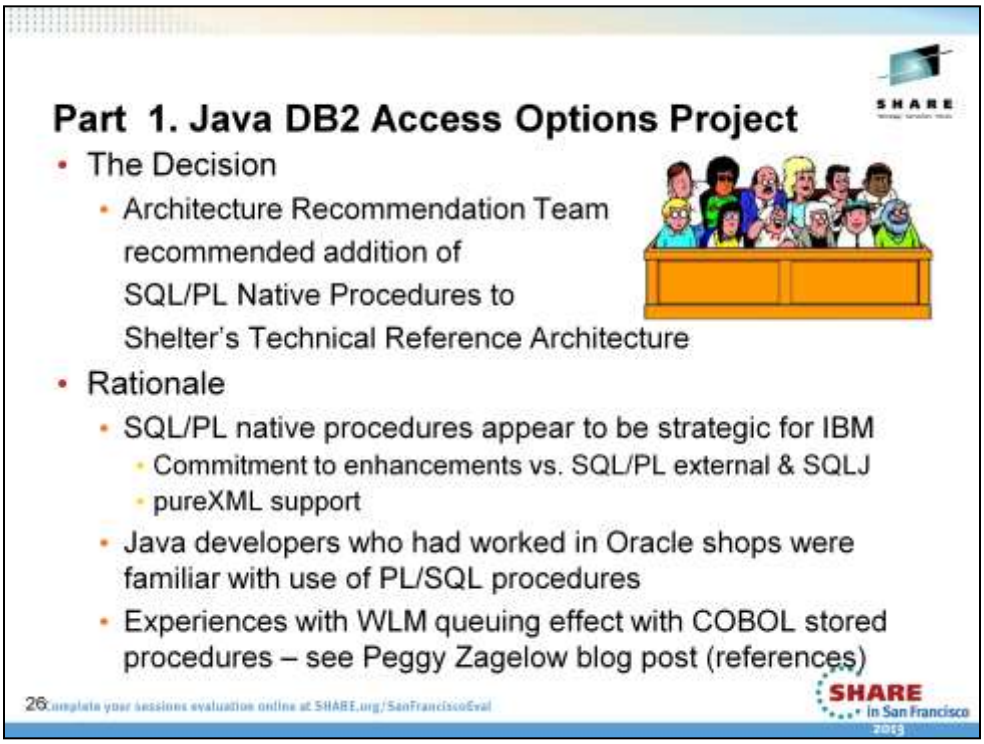


Part 1. Java DB2 Access Options Project

CPU Service Unit Comparison of JDBC to All Stored Procedures

Performance Difference		
Compare JDBC to all SPs	*Mainframe Service Units*	*% Better (Worse) than JDBC*
SQL/PL Native	5135	11%
SQL/PL External	5249	9%
SQLJ SP	5285	8%
COBOL SP	5329	7%
JDBC SP	5985	(4%)

1. The Service Units reported here are the DB2 Class 2 Service Units.
2. This chart reflects DB2 CPU Usage for JDBC compared to stored procedures.
3. The points I want to emphasize are:
   - SQL/PL Native Procedures in our test workload used 11% less CPU than JDBC.
   - COBOL stored procedures in our test workload used 7% less CPU than JDBC.
   - For our test workload where we made very sure to control for all the factors that could sway the results, this is very convincing evidence of the performance advantages of both SQL/PL and COBOL stored procedures over JDBC direct access to tables.
   - JDBC using direct access to tables is highly efficient, but the evidence from our project is that SQL/PL Native Procedures offer a considerable opportunity for CPU savings.



1. Here are some factors over and above metrics that influenced our decision:
    - Our research indicated that pureXML would be supported in SQL/PL native procedures.
2. SQL/PL external vs native – WLM effect
    - There is a WLM queuing effect for external stored procedures.
    - This is discussed in the Peggy Zagelow blog post cited in References.
3. Some Java developers were familiar with use of PL/SQL procedures from working in Oracle shops and were receptive to the idea of using a similar technology in DB2.
4. Recognition that IBM was not pushing SQLJ as they had 3-5 years earlier
    - Perception that SQLJ was a dead-end technology
5. Consideration of pureQuery
    - Shelter does not invest lightly in a new technology
    - Too many hands in the deployment pie
        - DBAs and WebSphere Administrators would need to be involved in code deployment as well as Java developers
        - Specialized skill required – not a typical DBA skill
        - "Recording" on WAS; XML file imported into Data Studio, then put back into WAS; custom properties in WAS

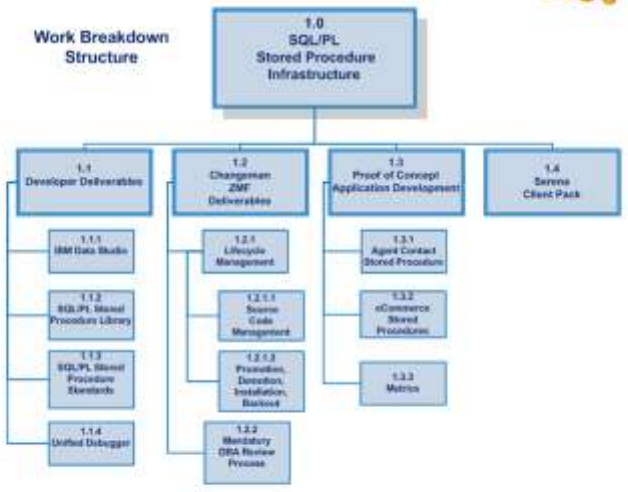
Part 2. What We Decided To Do After They Said Yes

- SQL/PL Stored Procedure Infrastructure Project [2011-present]

Part 2. SQL/PL Stored Procedure Infrastructure Project

- Project Objectives and Scope
  - Provide a development, build, and deployment infrastructure
    - Compatible with Java development and deployment tools and processes
    - Governed and secured
  - Proof of concept
    - Develop and deploy two SQL/PL stored procedures into a production environment in Shelter's eCommerce application
    - Replace current function coded as JDBC calls to tables
  - Measure "before" and "after" effect
    - Of converting from JDBC calls directly to tables to calls to SQL/PL stored procedures
    - Elapsed and CPU times

Part 2. SQL/PL Stored Procedure Infrastructure Project



Part 2. SQL/PL Stored Procedure Infrastructure Project

- Developer Deliverables
  - Use of Eclipse-based IBM Data Studio V3.1 for IDE
  - SQL/PL Stored Procedure Library
    - A library of 10-20 stored procedures demonstrating SQL/PL language syntax and features identified in requirements gathering
  - SQL/PL Stored Procedure Standards
    - Align standards with existing DB2 standards as well as Java standards
      - *Java-like: (Upper) Camel Case for procedure variable names*
      - *DB2: Use existing stored procedure naming conventions*
  - SQL/PL Stored Procedure Debugging
    - The use of the Unified Debugger is mandatory because there is no way to log to a console or SYSOUT.

Part 2. SQL/PL Stored Procedure Infrastructure Project

- Changeman ZMF Deliverables
  - Changeman ZMF 7.1.2 supports SQL/PL native procedures
    - New Changeman Application "SQLP" for SQL/PL Stored Procedures

Part 2. SQL/PL Stored Procedure Infrastructure Project

- Changeman ZMF Deliverables
  - Mandatory DBA review points
    - The DBA team at Shelter is a great advocate of SQL/PL native procedures
    - They are also especially protective of the DBM1 address space
  - Therefore
    - We are establishing mandatory SQL/PL DBA package approval before code is allowed to be installed into production
    - We are implementing a structured review process
      - *Checklist for SQL/PL Standards Compliance*
      - *Performance Review using Query Tuning feature of Data Studio*

Part 2. SQL/PL Stored Procedure Infrastructure Project

- Changeman ZMF Deliverables
  - Lifecycle Management
    - Secure SQL/PL source code
    - Promotion path defined to Changeman ZMF
      - Promote to/demote from: UNIT → DEVL → TEST → FUNC → MODL
      - Install into/backout from: PROD
    - Same promotion paths as COBOL applications (including COBOL stored procedures) using Serena Changeman
    - Conform to Shelter's Implementation Management Process
      - Follow the rules about how to put something into production
      - Shelter is gradually and systematically adopting ITIL practices

Part 2. SQL/PL Stored Procedure Infrastructure Project

- Lifecycle Management
  - Source Code Management
    - Initial assumption: Serena Dimensions
      - Shelter uses Eclipse IDE for Java development
    - Upon further reflection: Serena Client Pack for Changeman ZMF
      - Plug-in to Eclipse IDE that interacts with Changeman ZMF
      - Source code and deployment managed by one product
      - No explicit TSO login required to use Changeman ZMF
  - Financial considerations
    - Serena Dimensions seat license cost
    - Serena Client Pack license cost
  - Share the knowledge
    - We want both COBOL and Java developers to able to develop in SQL/PL

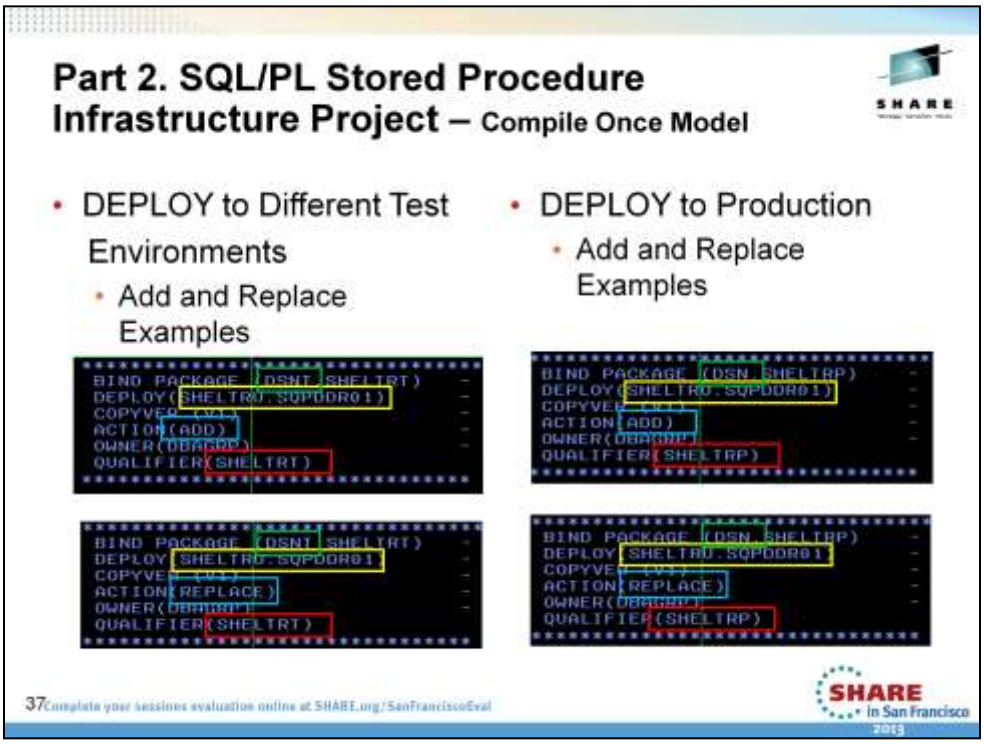
1. We initially assumed we would manage SQL/PL source code with Serena Dimensions like we do for Java source code, and that we would use separate products and processes to manage source code and deployment for SQL/PL like we do for Java. In that scenario, we would manage SQL/PL source code with Serena Dimensions, initially deploy SQL/PL stored procedures into UNIT using Data Studio, then DEPLOY into higher testing levels and production using Serena Changeman ZMF.
2. In September 2012 Serena came onsite at Shelter and demonstrated the Serena Client Pack. Following the demo, the project team reconsidered the initial assumptions.
3. We looked at integration considerations and the licensing costs for Serena Dimensions seat licenses vs. Client Pack licenses. We looked at our target developer audience. Although our initial audience is Java developers, we are also targeting COBOL developers as SQL/PL procedure developers. It made sense to have management of SQL/PL stored procedures entirely under Serena Changeman. The price of a Serena Client Pack seat license is approximately ¼ the cost of a Dimensions seat license, so it makes sense to consider the Serena Client Pack if our target developers are COBOL as well as Java developers. (All Java Developers have Serena Dimensions seat licenses, but no COBOL developers have Serena Dimensions Seat licenses).
4. Note, we plan to completely implement Serena Changeman ZMF SQL/PL support before embarking on the Client Pack.

Part 2. SQL/PL Stored Procedure Infrastructure Project

- Lifecycle Management
  - Compile Once vs. Compile Many Model
    - Which model would we use for SQL/PL Native Procedures?
  - Compile Once Model
    - Aligns with mainframe development methodology
      - *Source code changes occur only at initial testing level (UNIT)*
      - *Source code is compiled into an executable module residing in a library*
      - *Executable (not source code) is moved up ("promoted") into a series of testing levels ("promotion path") and eventually installed into production*
      - *Better ability to trace and verify*
    - Compile Many Model
      - *Source code is re-compiled when promoted into each testing level and when installed into production*

Part 2. SQL/PL Stored Procedure Infrastructure Project

- Lifecycle Management
  - Compile Once Model for SQL/PL Native Procedures
    - CREATE PROCEDURE … at initial testing level (UNIT)
    - BIND PACKAGE with DEPLOY option for subsequent promotion levels (DEVL, TEST, FUNC, MODL) and installation into PROD
    - "DEPLOY" is a new DB2 BIND PACKAGE option
      - *It allows the stored procedure definition and executable to be copied from one environment to another*
      - *Ability to change these items in the target environment:*
        - *DB2 LOCATION*
        - *Stored Procedure SCHEMA*
        - *QUALIFIER*



1. This example illustrates the use of BIND PACKAGE … DEPLOY for a stored procedure called SQPDDR01.
2. It is compiled / initially created in the UNIT testing level

using DB2 subsystem DSNT.

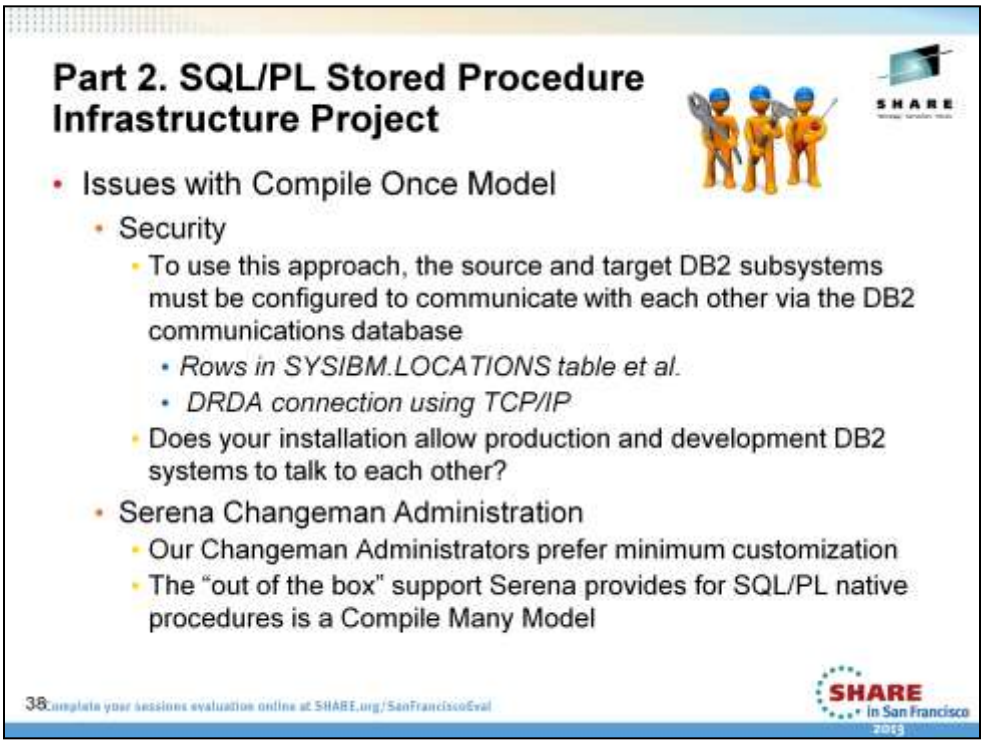
3. DB2 object qualifiers correspond to the testing level or

Production.

- UNIT uses a qualifier of SHELTRU
- TEST uses a qualifier of SHELTRT
- PROD uses a qualifier of SHELTRP

4. DSNT is Shelter's main development DB2 subsystem, and

DSN is Shelter's main production DB2 subsystem.



1. Despite our Changeman administrator's initial preference for the Compile Once Model, there are issues with the use of this model for SQL/PL stored procedures.
2. The Security team does not really like production and DB2 subsystems to be able to communicate with each other.

- This is still possible at Shelter because we use some DB2 replication techniques between production and test that require this communication (the DB2 Crossloader for one).
- So we were able to successfully test BIND PACKAGE … DEPLOY from the test DB2 subsystem to production.

3. We found out that Changeman ZMF 7.1.2 (the first release Shelter has used that supports SQL/PL native procedures) uses the Compile Many model. It performs a DROP PROCEDURE followed by a CREATE PROCEDURE when promoting or installing.

- Since our administrators also prefer as little customization as possible, that sealed the deal for the Compile Many Model.

Part 3. Where We Are Now and What We've Learned

- Current Project Status

1. I had hoped to have the project completely wrapped by the time I needed to have the presentation ready for SHARE in December 2012!
2. Oh well!

Part 3. Where We Are Now and What We've Learned

- Developer Deliverables
  - August-December 2012
- Changeman ZMF Deliverables
  - December 2012-January 2013
- Proof of Concept Application Development
  - Agent Contact Information stored procedure with State Liability Reporting

  (January-February 2013)
  - eCommerce stored procedures (planned)
    - 1H 2013
  - Serena Client Pack Implementation (planned)
    - 1H 2013

Part 3. Where We Are Now and What We've Learned

- Developer Deliverables
  - Creating a Safe Sandbox (1 of 2)
    - Shelter's DB2 subsystems are highly locked down
    - We do not normally allow developers at Shelter to create statically bound packages in DB2 collections using their own RACF authority
      - *DB2 authority is GRANTed to Serena Changeman ZMF, and all statically bound DB2 code is bound using Changeman's authority*

Part 3. Where We Are Now and What We've Learned

- Developer Deliverables
  - Creating a Safe Sandbox (2 of 2)
    - We wanted to give developers the ability to exercise the Data Studio IDE and make progress with development of SQL/PL Stored Procedure Library before the SQL/PL application was available in Changeman ZMF
    - So we GRANTed:
      - *CREATE IN SHELTRU SCHEMA to SQLPLGRP*
      - *CREATE IN COLLECTION SHELTRU to SQLPLGRP*
      - *BINDADD to SQLPLGRP*
    - Once our Changeman infrastructure is complete, we should be able to revoke these privileges

1. SQLPLGRP is the RACF group we designated for SQL/PL stored procedure development.

Part 3. Where We Are Now and What We've Learned

- Changeman ZMF Deliverables
  - Changeman required more customization than we anticipated
  - Templating feature for QUALIFIER did not work
    - We coded REXX scripting to overcome this
    - Opened case with Serena Changeman support
  - Templating feature for SCHEMA does not trim trailing blanks
    - Causes Data Studio not to be able to retrieve code that is syntactically valid at database server
    - Opened case with Serena Changeman support

Part 3. Where We Are Now and What We've Learned

- Changeman ZMF Deliverables
  - Use of AUTODROP feature in Changeman
    - SQL/PL stored procedure source code specifies CREATE PROCEDURE
    - We chose Changeman AUTODROP feature
    - EXECUTE privileges lost every time code is changed or promoted!
  - We created our own copy of SYSIBM.SYSROUTINEAUTH
    - REXX scripting to MERGE rows from (IBM) SYSROUTINEAUTH into (Shelter) SYSROUTINEAUTH before AUTODROP
    - Re-issue GRANTs within Changeman process after successful CREATE PROCEDURE
  - Prevent production outages and DBA workload thrashing

Part 3. Where We Are Now and What We've Learned

- Changeman ZMF Deliverables
  - Use of AUTODROP feature in Changeman
    - SQL/PL stored procedure source code specifies CREATE PROCEDURE
    - We chose Changeman AUTODROP feature
    - EXECUTE privileges lost every time code is changed or promoted!
  - We created our own copy of SYSIBM.SYSROUTINEAUTH
    - REXX scripting to MERGE rows from (IBM) SYSROUTINEAUTH into (Shelter) SYSROUTINEAUTH before AUTODROP
    - Re-issue GRANTs within Changeman process after successful CREATE PROCEDURE
  - Prevent production outages and DBA workload thrashing

Part 3. Where We Are Now and What We've Learned

- Proof of Concept Application Development
  - Agent Contact Information Stored Procedure to validate Changeman infrastructure
  - SQL/PL Stored Procedure calling COBOL Stored Procedure
    - *COBOL Stored Procedure had previously been used only with COBOL applications*
    - *De-normalized VARCHAR output parameter*
      - *Concatenated character and integer data elements*
      - *SQL/PL Stored Procedure enforces data typing and wouldn't allow SUBSTRing and CASTING of VARCHAR data to integer*
      - *Developer wrote a third COBOL stored procedure to translate VARCHAR to INTEGER using COBOL language rather than SQL*

Part 3. Where We Are Now and What We've Learned

- Serendipity
  - No more -805 SQLCODEs!
  - Because the procedure definition and executable can never be separated, -805 SQLCODEs should become a thing of the past with SQL/PL native stored procedures

What's Your Opinion?

- Will SQL/PL ever succeed in winning over Queen Duke?!

References (Part 1)

- DB2 9 for z/OS Stored Procedures: Through the Call and Beyond
  - http://www.redbooks.ibm.com/abstracts/sg247604.html
- Native SQL Procedures in DB2 9 for z/OS *by Florence Dubois*
  - http://www.sqladria.net/en/wp-content/uploads/2010/03/DB2-9-Native-SQL-Procedures.pdf

References (Part 2)

- DB2 V9.1 for z/OS Application Programming Guide and Reference for Java (SC18-9842)
  - http://publib.boulder.ibm.com/epubs/pdf/dsnjvk19.pdf
- DB2 V9.1 for z/OS Performance Monitoring and Tuning Guide (SC18-9851)
  - http://publib.boulder.ibm.com/epubs/pdf/dsnpfk1c.pdf
- Helpful articles on configuring and using the Unified Debugger (Parts 1 and 2)
  - http://www.ibm.com/developerworks/data/library/techarticle/dm-0811zhang/
  - http://www.ibm.com/developerworks/data/library/techarticle/dm-0903debugdatastudio/index.html

References (Part 3)

- Robert Catterall blog posts
  - http://robertsdb2blog.blogspot.com/2011/08/important-db2-10-for-zos-stored.html
  - http://robertsdb2blog.blogspot.com/2012/03/apar-pm29226-and-db2-for-zos-native-sql.html
  - http://catterallconsulting.blogspot.com/2008/11/db2-9-for-zos-stored-procedure-game.html
  - http://robertsdb2blog.blogspot.com/2012/07/db2-for-zos-whats-with-ddf-cpu-time.html
- Peggy Zagelow blog post
  - https://www.ibm.com/developerworks/mydeveloperworks/blogs/peggggggy/entry/native_sql_procedures_in_dbm1?lang=en


#SHAREorg
SHARE
Wooing Java Developers to SQL/PL Native Procedures
Carol Elliott
Shelter Insurance Companies
February 7, 2013
Session Number 12518
http://www.linkedin.com/profile/view?id=25501181&trk=tab
@comocello
SHARE in San Francisco 2013