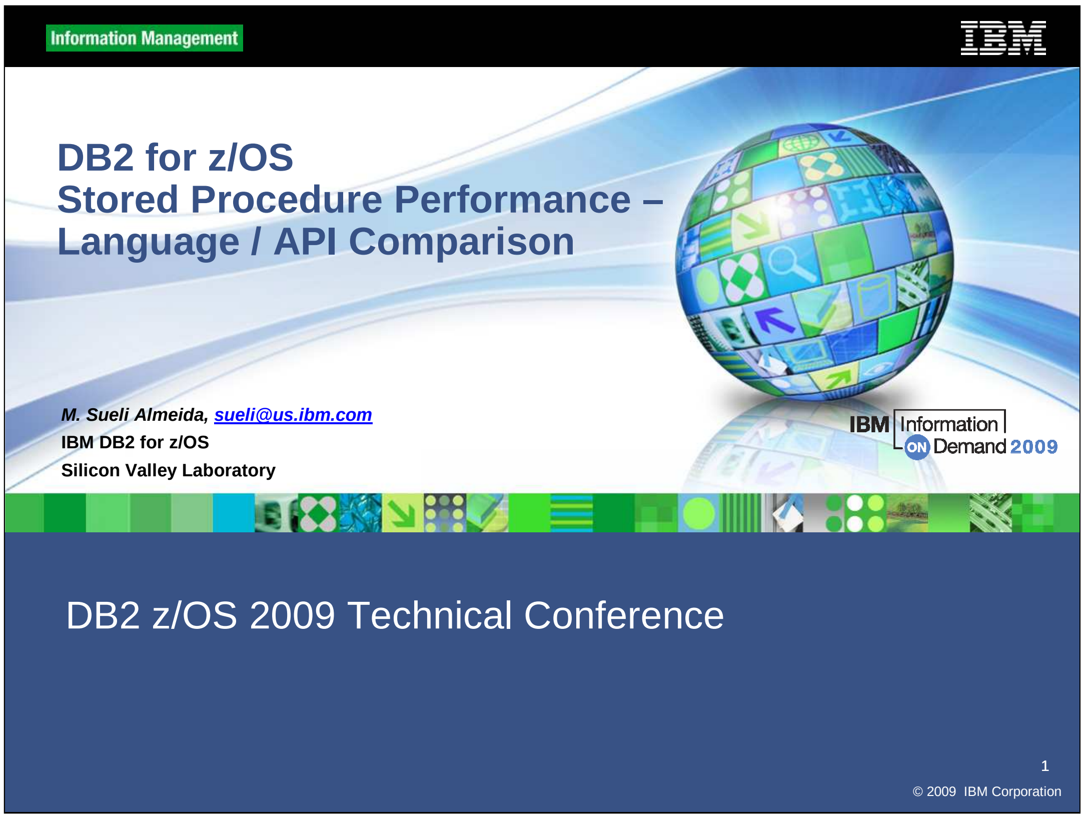


Information Management

DB2 for z/OS Stored Procedure Performance – Language / API Comparison

M. Sueli Almeida, sueli@us.ibm.com
IBM DB2 for z/OS
Silicon Valley Laboratory

IBM Information ON Demand 2009

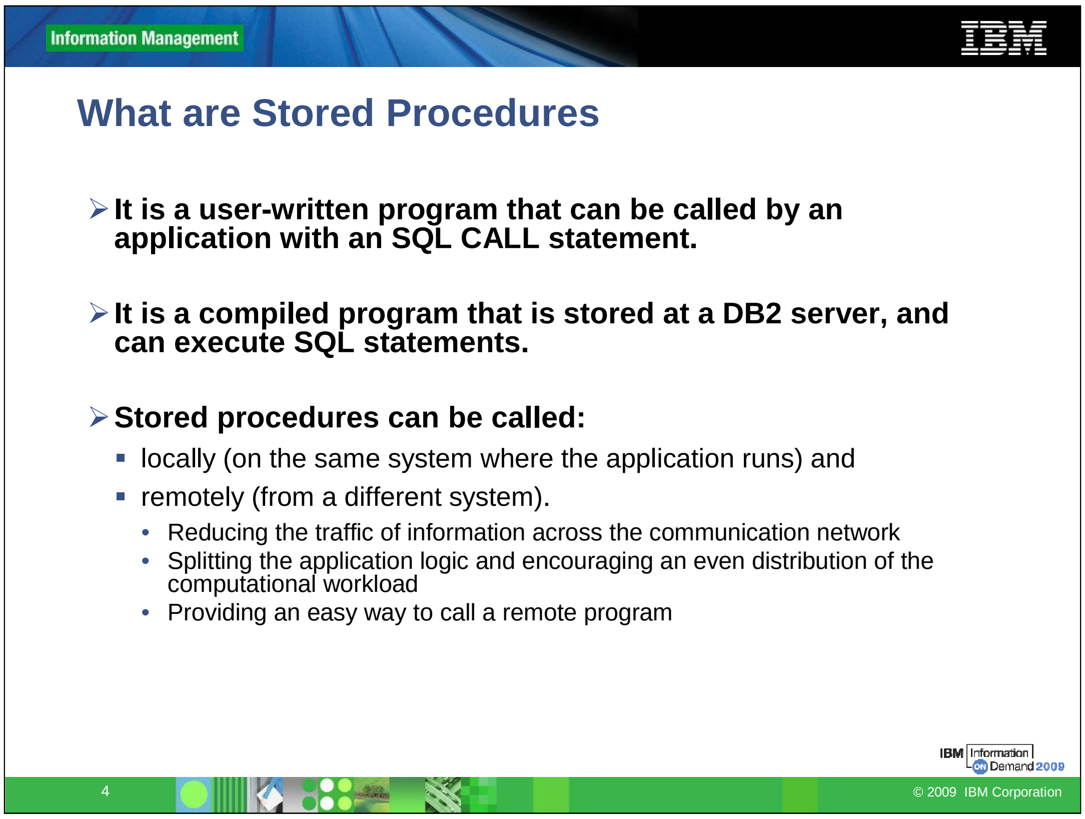
DB2 z/OS 2009 Technical Conference

1

Important Disclaimer

Agenda

- Stored Procedure Overview
- Stored Procedure Performance Checklist
- Customer reasons for looking into Language / API alternatives
- Pros and Cons of each Language / API
- Performance and Cost comparison


Information Management

IBM

What are Stored Procedures

- It 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, and can execute SQL statements.
- Stored procedures can be called:
  - locally (on the same system where the application runs) and
  - remotely (from a different system).
    - Reducing the traffic of information across the communication network
    - Splitting the application logic and encouraging an even distribution of the computational workload
    - Providing an easy way to call a remote program

IBM Information On Demand 2009

Distributed applications require access to databases across a network. Unfortunately, this type of access can result in poor performance when a lot of network interactions and movements of data are involved. A stored procedure runs on the database server with a goal of reducing the network traffic.

Since their introduction in DB2 for MVS/ESA™ Version 4, the roles of stored procedures in an enterprise have continued to grow.

Here we discuss what stored procedures are, why they are important, and how pervasive they are becoming.

Distributed applications to:

Distribute the logic between a client and a server

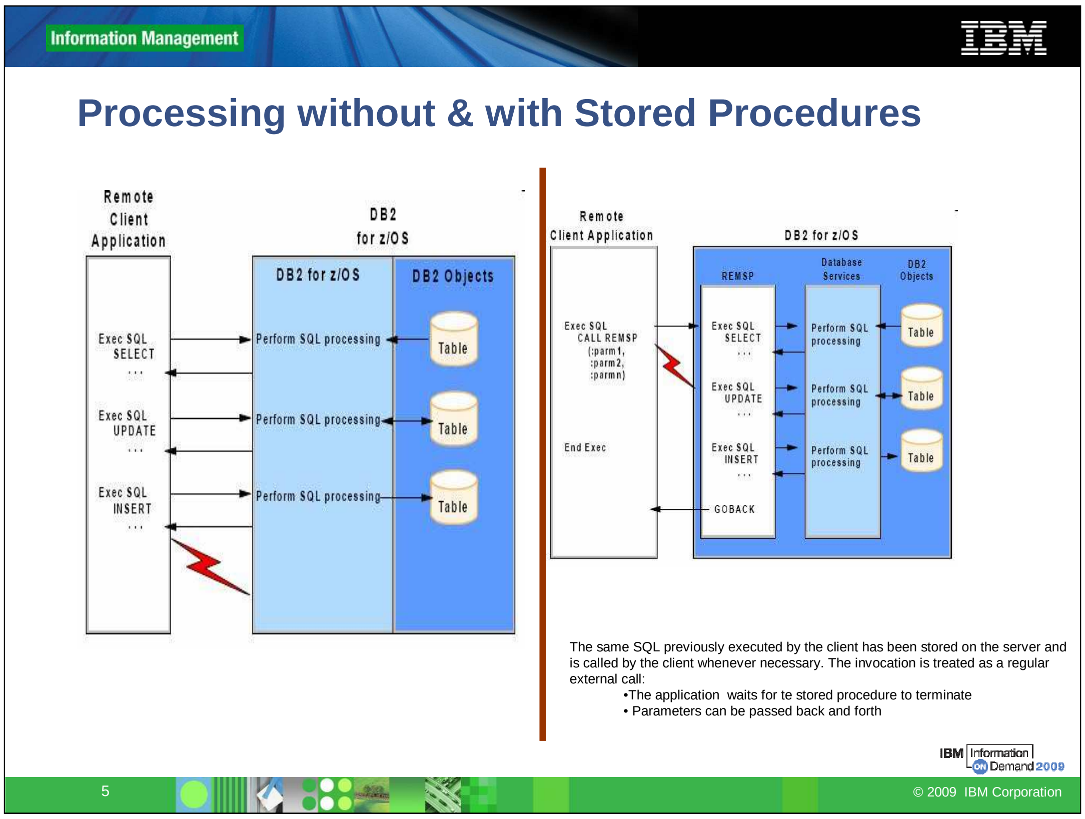
Perform a sequence of operations at a remote site

Combine results of query functions at a remote site

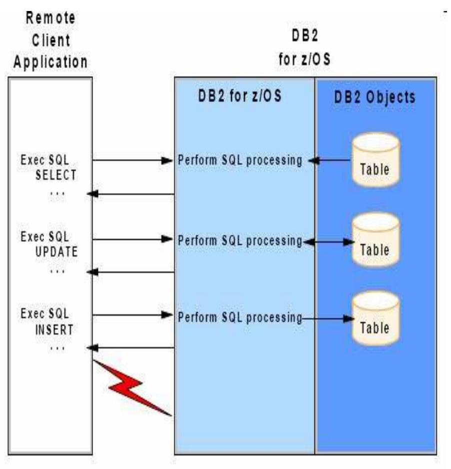
Control access to database objects

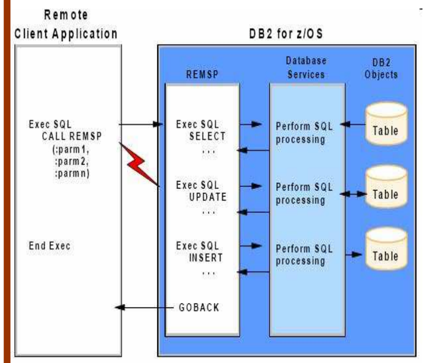
Remove SQL applications from the workstation and prevent workstation users from manipulating the contents of sensitive SQL statements and host variables

Dynamically invoke static SQL rather than use Java Data Base Connectivity (JDBC) dynamic SQL approach



Processing without & with Stored Procedures

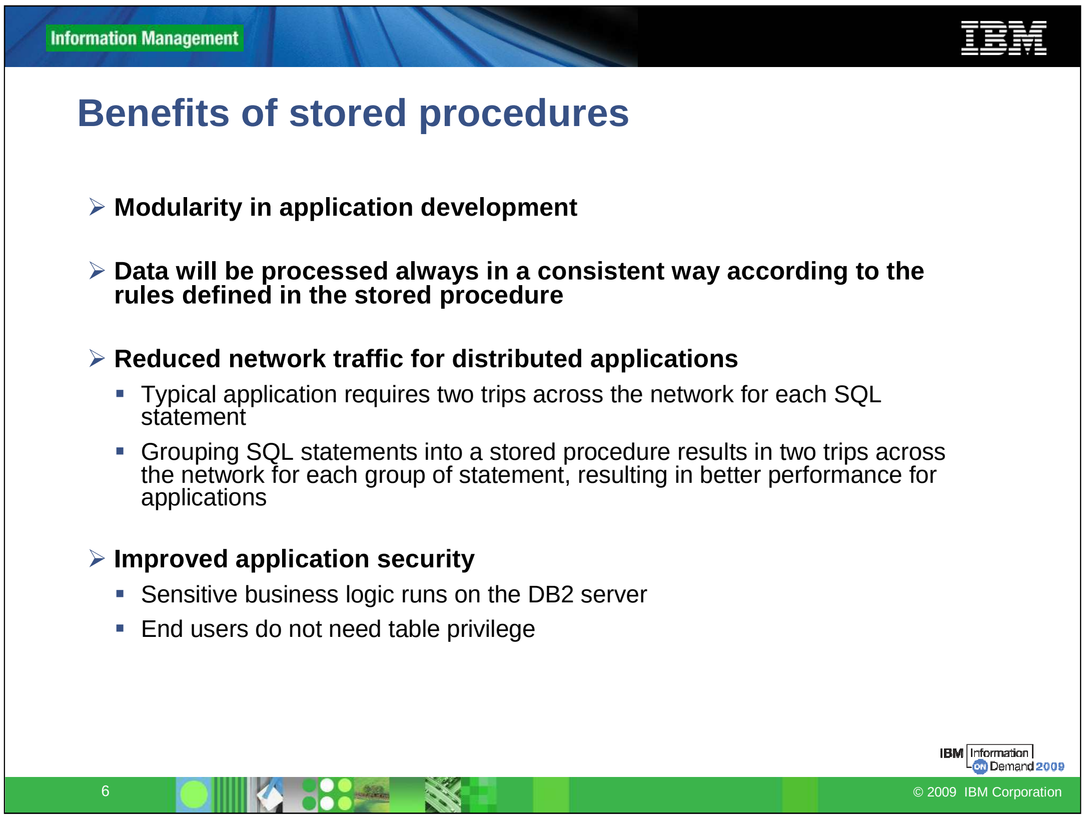




The same SQL previously executed by the client has been stored on the server and is called by the client whenever necessary. The invocation is treated as a regular external call:

- •The application waits for te stored procedure to terminate
- • Parameters can be passed back and forth

The advantages provided by stored procedures are clear when comparing them to a standard distributed application where the client may be a workstation or a Java client. The slide shows that the client communicates with the server separately for each embedded SQL request.



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
- Reduced network traffic for distributed applications
  - Typical application requires two trips across the network for each SQL statement
  - 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
- Improved application security
  - Sensitive business logic runs on the DB2 server
  - End users do not need table privilege

Your **programming productivity** can be **improved** by using stored procedures when you **develop** and **maintain** applications:

Modularity in application development is encouraged with stored procedures.

> **Client developers can focus on their application logic details, while stored procedure programmers develop appropriate DB2 server access**

When any application calls the stored procedure, it will process data in a consistent way according to the rules defined in the stored procedure. If you need to change the rules, you only need to make the change once in the stored procedure, not in every application that calls the stored procedure.

Reduced network traffic for distributed applications:

> A typical application requires two trips across the network for *each* SQL statement.
>
> Grouping SQL statements into a stored procedure results in two trips across the network for each *group* of statements, resulting in better performance for applications.
>
> Improved application security
>
> Sensitive business logic runs on the DB2 server
>
> End users do not need table privileges

Access to features that exist only on the server:

> Stored procedures can have access to commands that run only on the server.
>
> They might have the advantages of increased memory and disk space on server machines.
>
> They can access any additional software installed on the server.

Enforcement of business rules:

> You can use stored procedures to define business rules that are common to several applications.
>
> This is another way to define business rules, in addition to using constraints and triggers.

Application integration solutions:

> You can use stored procedures to easily access non-DB2 resources.
>
> With the use of WebSphere® MQ, you can coordinate accesses to multiple data and platforms.

Cost of ownership reduction

> DRDA® activity is a candidate for zIIP re-routing. A smaller percentage of work is redirected to zIIP for remote non SQL-native-procedures, just the CALL, COMMIT and result set processing.
>
> Stored procedures written in Java can take advantage of zAAP engines
>
> Native SQL procedures have richer SQL functions and remote native SQL procedures, running as enclaves in DBM1 address space, are candidate for zIIP reroute with DB2 V9.

Benefits of stored procedures cont…

➢ **Access to features that exist only on the server:**

- Stored procedures can have access to commands that run only on the server.
- They might have the advantages of increased memory and disk space on server machines.
- They can access any additional software installed on the server.

➢ **Enforcement of business rules:**

- You can use stored procedures to define business rules that are common to several applications.
- This is another way to define business rules, in addition to using *constraints* and *triggers*.

Benefits of stored procedures cont…

➢ Application integration solutions:

- You can use stored procedures to easily access non-DB2 resources.
- With the use of WebSphere® MQ, you can coordinate accesses to multiple data and platforms.

➢ Cost of ownership reduction

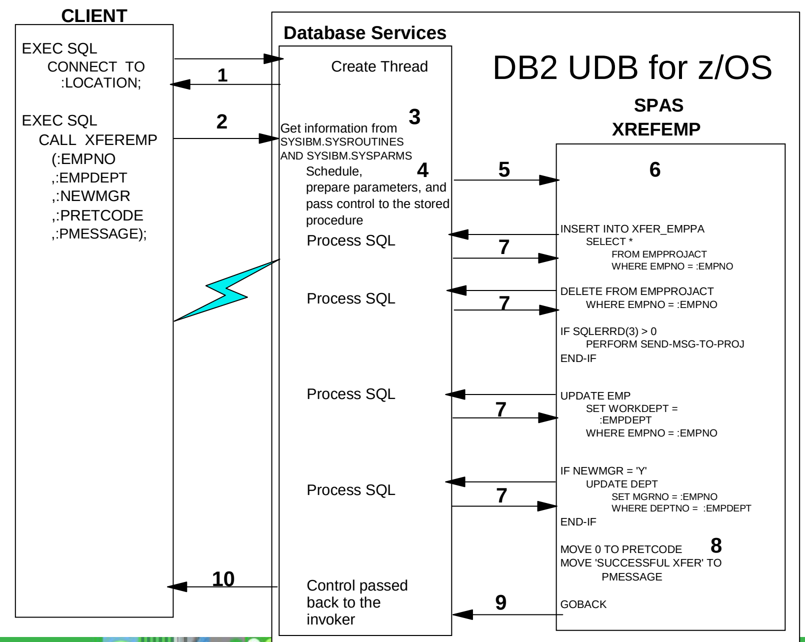
- DRDA® activity is a candidate for zIIP re-routing. A smaller percentage of work is redirected to zIIP for remote *non SQL-native-procedures*, just the CALL, COMMIT and result set processing.
- Stored procedures written in Java can take advantage of zAAP engines
- Native SQL procedures have richer SQL functions and remote native SQL procedures, running as enclaves in DBM1 address space, are candidate for zIIP reroute with DB2 V9.

Use of stored procedures

- **Distributed applications to:**
  - Distribute the logic between a client and a server
  - Perform a sequence of operations at a remote site
  - Combine results of query functions at a remote site
  - Control access to database objects
  - Remove SQL applications from the workstation and prevent workstation users from manipulating the contents of sensitive SQL statements and host variables
  - Dynamically invoke static SQL rather than use Java Data Base Connectivity (JDBC) dynamic SQL approach

- **To access non-DB2 resources:**
  - VSAM files
  - Flat files
  - IMS or CICS transactions
  - DL/I databases
  - MVS/APPC conversations
  - Utilize Recoverable Resource Services (RRS) to coordinate two-phase commit processing of recoverable resources.

Stored procedure flow


CLIENT
Database Services
DB2 UDB for z/OS
Create Thread
1
2
3
Get information from SYSIBM.SYSROUTINES AND SYSIBM.SYSPARMS
4
Schedule, prepare parameters, and pass control to the stored procedure
5
6
SPAS XREFEMP
Process SQL
7
Process SQL
7
Process SQL
7
Process SQL
7
8
9
10
Control passed back to the invoker

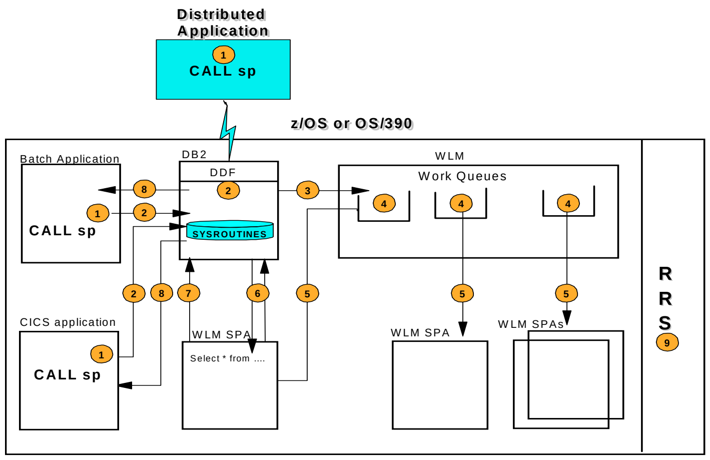
```
EXEC SQL
    CONNECT  TO
      :LOCATION;

EXEC SQL
   CALL  XFEREMP
      (:EMPNO
      ,:EMPDEPT
      ,:NEWMGR
      ,:PRETCODE
      ,:PMESSAGE);
```

```
INSERT INTO XFER_EMPPA
      SELECT *
           FROM EMPPROJACT
           WHERE EMPNO = :EMPNO

DELETE FROM EMPPROJACT
      WHERE EMPNO = :EMPNO

IF SQLERRD(3) > 0
      PERFORM SEND-MSG-TO-PROJ
END-IF

UPDATE EMP
      SET WORKDEPT =
         :EMPDEPT
      WHERE EMPNO = :EMPNO

IF NEWMGR = 'Y'
      UPDATE DEPT
           SET MGRNO = :EMPNO
           WHERE DEPTNO =  :EMPDEPT
END-IF

MOVE 0 TO PRETCODE
MOVE 'SUCCESSFUL XFER' TO
          PMESSAGE

GOBACK
```

1. A thread must be created for each application that needs DB2 services. If the stored procedure is called from a remote client, the thread is created when the client application issues the SQL CONNECT statement. If the application is local, the thread is created when the first SQL statement is executed. After the thread is created, SQL statements can be executed.
2. When a client application issues an SQL CALL statement, the stored procedure name and the I/O parameters are passed to DB2.
3. When DB2 receives the SQL CALL statement, it searches in the SYSIBM.SYSROUTINES catalog table for a row associated with the stored procedure name. From this table, DB2 obtains the load module associated with the stored procedure and the run environment information. It also searches SYSIBM.SYSPARMS to gather the parameter information, such as whether a parameter is input, output or input/output, and the data type of each expected parameter. *Notice that catalog information is cached to avoid I/O*.
4. With the exception of native SQL language procedures, stored procedures are executed in address spaces that run fenced away from the DB2 code. Multiple Workload Manager (WLM) address spaces may be made available for stored procedures. Starting with DB2 for z/OS V8, all newly created stored procedures must use the WLM-established stored procedure address spaces. Those stored procedures that existed prior to Version 8 and were created to run in the DB2-managed address space will continue to run, but all new stored procedures must run in a WLM-managed stored procedure address space. In DB2 9 for z/OS all stored procedures, except for native SQL procedures, are WLM-managed.WLM goal mode is mandatory. You can specify a number of task control blocks (TCBs) in this address space available for stored procedures. Each stored procedure is executed under one TCB. After DB2 has searched the SYSIBM.SYSROUTINES table, an available TCB to be used by the stored procedure is selected, and the stored procedure address space is notified to execute the stored procedure.
5. When the stored procedures address space executes a stored procedure, the thread that was created for the client application is reused for the execution. This has the following implications:
   – CPU cost is low because DB2 does not create a new thread.
   – Accounting is on behalf of the client application.
   – For static SQL, the OWNER of the client program must have execute privilege on the stored procedure package. For dynamic SQL issued by the stored procedure, security is checked against the user of the client program, unless the DYNAMICRULES(BIND) option was specified when binding the package for the stored procedure. No sign-on or connection processing is required.
   – The stored procedures address space uses the LE/370 product libraries to load and execute the stored procedure. Through SYSIBM.SYSROUTINES, you can pass run-time information for LE/370 when the stored procedure is executed.

6. Control is passed to the stored procedure along with the input and output parameters
   – The stored procedure can issue most SQL statements. It also can have access to non-DB2 resources.
   – The stored procedure can either perform all the database access and return the output to the calling program as output parameters (see the next step) or it can open a cursor to build a result set and let the calling program fetch from the result set.
   – Any processing done by the stored procedure is considered a logical continuation of the client application's unit of work. Thus, locks acquired by the stored procedure are released when the unit of work terminates. If DB2 has been instructed, through the definition of the stored procedure, it can commit the logical unit of work upon return to the caller.
7. Before terminating, the stored procedure assigns values to any output parameters and returns control to DB2.
8. DB2 copies the output parameters received from the stored procedure to the client application parameter area and returns control to the client application.
9. The calling program receives the output parameters and continues the same unit of work. If the stored procedure returned a result set, then the client application can fetch rows from the result set until there are no more rows to fetch. The client application implicitly or explicitly issues the COMMIT statement. DB2 can implicitly commit as soon as the stored procedure returns control to the client application based upon the value of the COMMIT ON RETURN option in the CREATE PROCEDURE statement. See 9.1, “CREATE or ALTER PROCEDURE parameters” on page 92 for details of the CREATE statement and the options available. If the client application and the stored procedures used during this execution update at different sites, the two-phase commit protocol is used.
10. DB2 returns control to the invoking program

System management of stored procedures



In order to better understand the execution environment of stored procedures, let us take a look "behind the scenes" of a stored procedure execution. The slide illustrates the relationship between application code, DB2 address spaces, and WLM application environments, etc.

1. The distributed application (or batch, or CICS) issues the SQL CALL statement to invoke a stored procedure.

2. The request is received and handled by the DDF address space, and passed to DB2. For qualified CALLs DB2 uses the three-part name (location, schema, procedure name). For unqualified CALLs (the recommended way), DB2 for z/OS implicitly uses the current server (the location) and SQL path (the schema). If the procedure name is specified as a literal, the SQL path is the value of the PATH bind option that is associated with the calling package or plan. If the procedure name is specified with a host variable, the SQL path is the value of the CURRENT PATH special register. DB2 searches the SYSIBM.SYSROUTINES catalog table using the procedure name and, after verifying authorizations and parameters definitions, retrieves the collection ID (COLLID) and WLM application environment (WLM_ENVIRONMENT) names associated with the stored procedure.

3. DB2 sends a request to WLM to schedule the stored procedure in an application environment.

4. WLM places the request in one of its queues. WLM maintains one queue for each combination of application environment and service class. For example, if you have three service classes DDFWKLD (for DDF workload), ONLNWKLD (for CICS workload) and BATCWKLD (for batch workload) for one application environment WLMAE, WLM maintains three separate queues for the same WLMAE. The requests within a queue are processed in an FIFO manner. 5. Once your request has its turn, WLM checks for the availability of a WLM stored procedure address space (SPAS).

– If no WLM SPAS exists, WLM starts a new one and executes the stored procedure.

– If WLM SPAS exists and free TCBs are available, it executes the stored procedure.

– If WLM SPAS exists and there is no availability of free TCBs:

• If WLM is meeting the performance goal set for service class, it waits for availability of a TCB in one of the active WLM SPAS. The time DB2 waits depends on the TIMEOUT VALUE (on installation panel DSNTIPX). If the wait time exceeds the TIMOUT VALUE, the request will be timed out and the caller receives a -471 SQLCODE with reason code 00E79002.

• If WLM is *not* meeting the performance goal set for the service class, a new WLM SPAS will be started that executes the stored procedure.

– If WLMAE is in a stopped or quiesced state, it sends a return code back to the calling program.

6. Once the stored procedure is scheduled.
DB2 now executes the SQL inside it (if SQL are present).
DB2 first uses the following sequence to determine the collection ID:

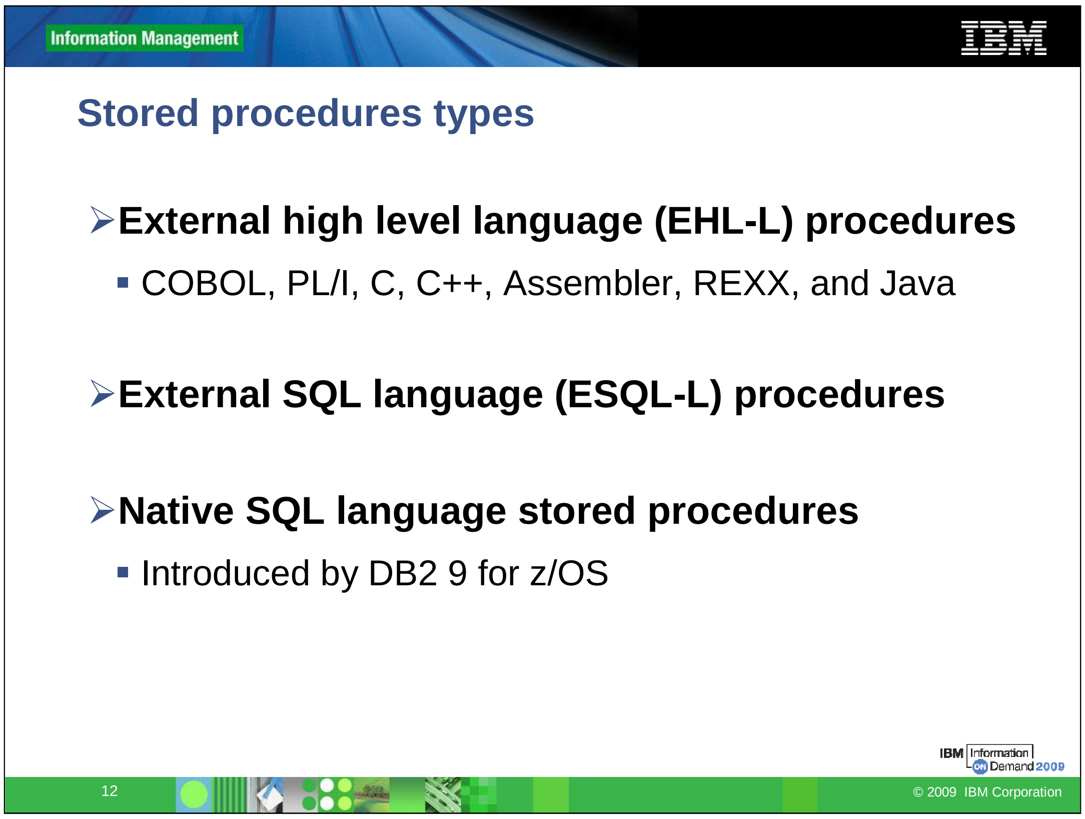
a. CURRENT PACKAGE PATH special register in the storage procedure program or in the calling application or even in the CREATE PROCEDURE statement instead of setting COLLID (APAR PK59752).
b. CURRENT PACKAGESET in the stored procedure program
c. COLLID collection name in CREATE PROCEDURE
d. CURRENT PACKAGESET in the calling application
e. COLLID in the calling application
f. PKLIST in the calling application

7. When the stored procedure reaches return statement, it passes control back to DB2 with or without results depending on the logic.

8. DB2 passes control back to the calling program with or without results depending on the logic.

9. Since the transactions involving stored procedures can span multiple address spaces, RRS plays the role of coordinator for all the resources between all address spaces involved in the transaction.

Steps 1 to 8 are repeated for each execution of a stored procedure. If a stored procedure calls another stored procedure, again all the above steps will happen.



Stored procedures types

- **External high level language (EHL-L) procedures**
  - COBOL, PL/I, C, C++, Assembler, REXX, and Java
- **External SQL language (ESQL-L) procedures**
- **Native SQL language stored procedures**
  - Introduced by DB2 9 for z/OS

There are two common criteria by which stored procedures are often categorized: by the language in which they are written; and by the type of address space in which they run.

External high level language procedures and External SQL language procedures result in an external load module being created and they run in Workload Manager (WLM) address spaces.

Native SQL language procedures do not result in a load module being created and they run in the DBM1 address space.

External SQL language procedures and Native SQL language procedures, the source code is written entirely in SQL, with the program logic being part of the stored procedure definition (within the CREATE PROCEDURE statement itself).

What is the difference between External and Native SQL procedures?

➢ **External SQL**

- Parsed and translated into C.
- Runs as a C stored procedure internally.
- WLM managed.

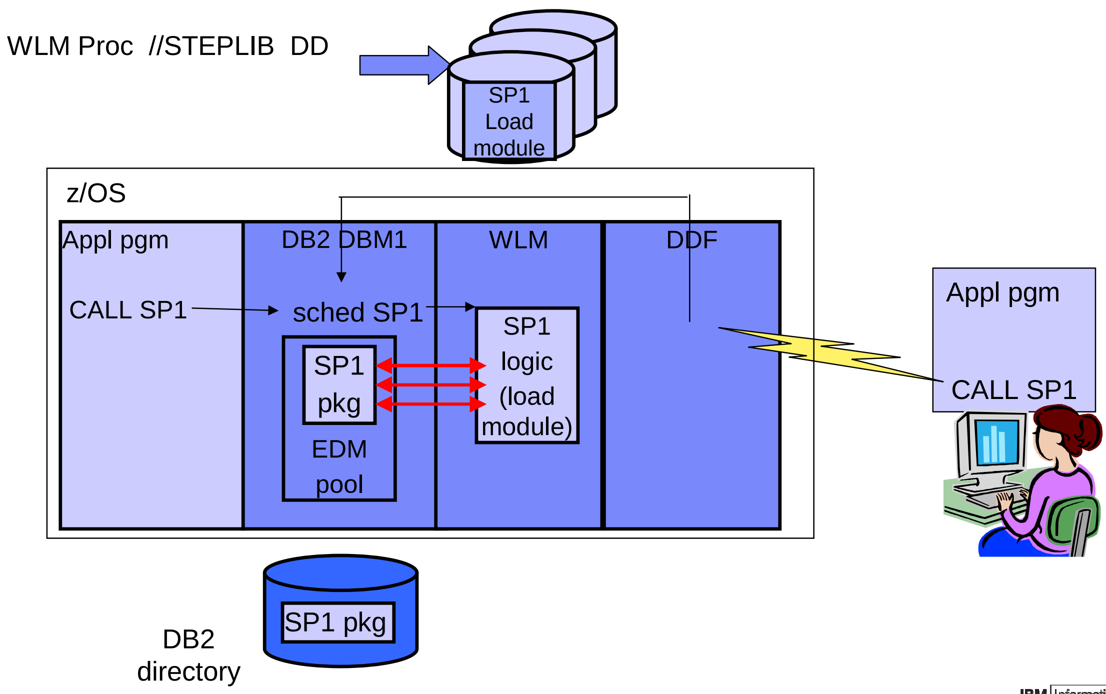
➢ **Native SQL**

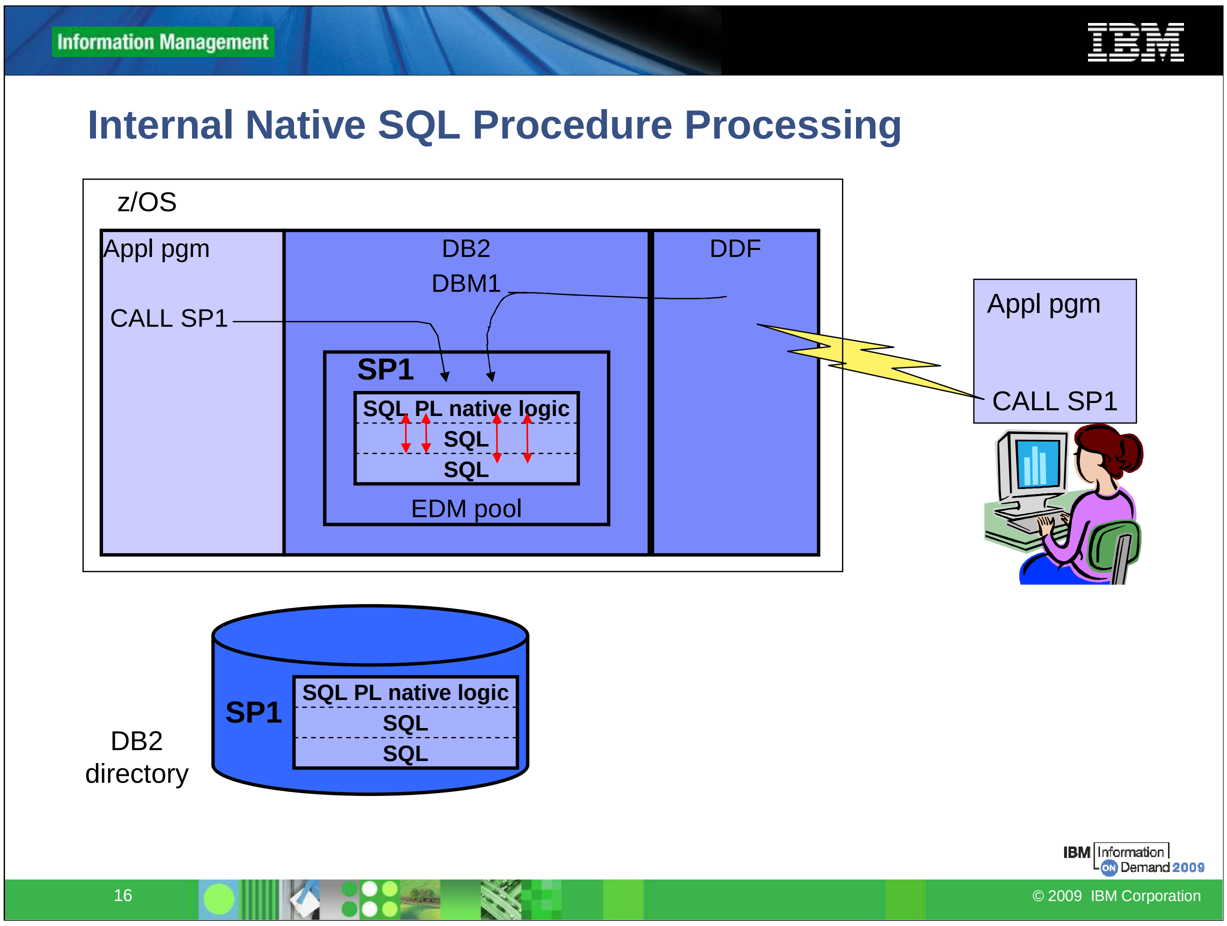
- Broken down into runtime structures like any other SQL statement.
- Does not need a WLM environment and C compiler.
  - Improved throughput and less CPU usage
- Runs under DBM1 enclave SRB, thus zIIP eligible.

When is it good to use SQL language? When is it bad?

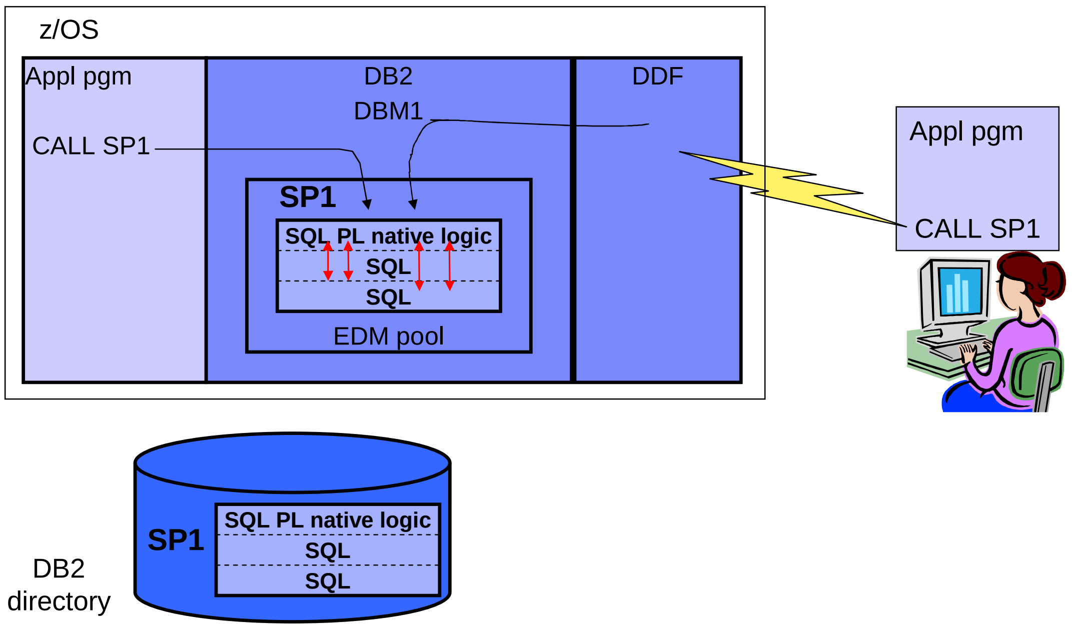
- **Good when …**
  - SQL intensive
  - Contains minimal application logic
  - Lowest billable cost (for remote SP) and productivity are the most important priorities.
- **Bad when …**
  - Contains significant amount of application logic
  - Executes math, string manipulation functions
  - Many IF/WHILE/CASE/REPEAT statements

External Stored Procedure Processing





Internal Native SQL Procedure Processing



A native SQL stored procedure, its procedural statements are now converted to a native representation that is stored in the DB2 catalog and directory, as is done with other SQL statements. The parameter list and procedure options are stored in the database catalog tables as in the prior releases. When you call a native SQL procedure, DB2 loads the native representation from the catalog and the DB2 engine executes the procedure.

This slide shows the DB2 components involved when a native SQL procedure is called either from a remote application, a DB2 attached program, or an allied address space respectively. As illustrated the SQL statements are no longer executed in an external WLM address spaces but natively in the database system services address space. For execution, the procedure packages are loaded into the EDM pool.

When calling a native SQL procedure the resolution is performed with the help of the **schema name, the procedure name, and the number of parameters**.

Once the correct procedure has been resolved, the current active version is determined. Only one active version of a native SQL procedure exists at any point in time. By default the current active version is the version that contains a flag 'Y' in the ACTIVE column of the catalog table SYSIBM.SYSROUTINES. Any version that exists for a native SQL procedure can be "promoted" to be the active version with the ACTIVATE VERSION option of the ALTER statement. To default the execution of the procedure MEDIAN_RESULT_SET to version MEDIAN_V2 the following command can be used:

```
ALTER PROCEDURE MEDIAN_RESULT_SET ACTIVATE VERSION MEDIAN_V2#
```

For ad hoc testing of a specific version, a new special register called `CURRENT ROUTINE VERSION` is provided. If an application sets this register to an existing valid version, it temporarily overrides the setting in column ACTIVE of the SYSIBM.SYSROUTINES catalog table. If the provided value is not valid, the default catalog settings are used. The subsequent CALL to the stored procedure has to be dynamically prepared, otherwise the register is not taken into account. This is mainly because a static CALL to the native SQL procedure already has the version identifier bound to in the respective package.

What is common to all Stored Procedures called via DRDA ?

- **Portion of distributed workload that comes in via DRDA is eligible for zIIP redirect. This includes:**
  - Call statement processing
  - Result set processing
  - Commit processing

- **For WLM managed stored procedures:**
  - SQL processing runs under a TCB hence not eligible for zIIP redirect.

- **For Remote Native SQL procedures:**
  - SQL processing is under an enclave SRB, thus zIIP eligible

Some customer reasons for looking at alternative Languages / APIs

- To leverage potential cost savings of zIIP and zAAP assist processors.
- Productivity and manageability improvements available with new enterprise level tooling.
- Redesigning some applications and want to investigate viability of moving to a different language during redesign.
- Specific language programming skills dwindling and hard to acquire/replace.
- All of the above.

What Languages/API's will be discussed ?

- COBOL
- C/C++
- JDBC
- SQLJ
- External SQL
- Native SQL

Pros and Cons of each Language / API

COBOL Stored Procedure

- Pros
  - Excellent throughput performance
  - Robust development language
  - Low billable cost
- Cons
  - Runs as WLM managed
  - Not eligible for zAAP redirect
  - Future availability of COBOL development resources

C/C++ Stored Procedure

➢ Pros
- Best performance
- Robust development language
- Low billable cost
  - Lower than COBOL
- Future availability of C/C++ development resources
- Portable

➢ Cons
- Runs as WLM managed
- Not eligible for zAAP redirect

SQLJ Stored Procedure

- **Pros**
  - zAAP specialty engine eligible
  - Robust development language
  - Future availability of Java development resources
  - Portable
  - Wide variety of development tools
  - Easy to code
  - Benefits of Static SQL

- **Cons**
  - Runs as WLM managed.
  - Higher billable cost.
    - Above COBOL

JDBC Stored Procedure

- Pros
  - zAAP specialty engine eligible
  - Robust development language
  - Future availability of Java development resources
  - Portable
  - Wide variety of development tools
  - Easy to code
    - More complex than SQLJ

- Cons
  - Runs as WLM managed
  - Highest billable cost
    - Above External SQL
  - Dynamic SQL

External SQL Stored Procedure

- Pros
  - Portable
  - Easy to code
  - Can be migrated to native SQL in DB2 9 for z/OS
- Cons
  - Runs as WLM managed
  - Not a robust development language
  - Not eligible for zAAP redirect
  - Higher billable cost
    - Above SQLJ

Native SQL Stored Procedure

➢ **Pros**

- Portable, no C language dependency
- Better tools (Data Studio) to code, debug, test and deploy
- Simplify migration to DB2
  - Migration tool : http://www-306.ibm.com/software/data/db2/migration/mtk/
- Lowest billable cost with zIIP usage for remote call
- Enhanced SQL support :
  - FOR Loops, nested compound statements
  - More data types (BIGINT,BINARY,VARBINARY,DECFLOAT)
- Extensive support for versioning
  - **VERSION keyword on CREATE PROCEDURE**
  - **CURRENT ROUTINE VERSION special register**
  - **ALTER ADD/REPLACE/ACTIVATE VERSION**
- BIND PACKAGE with new DEPLOY keyword

➢ **Cons**

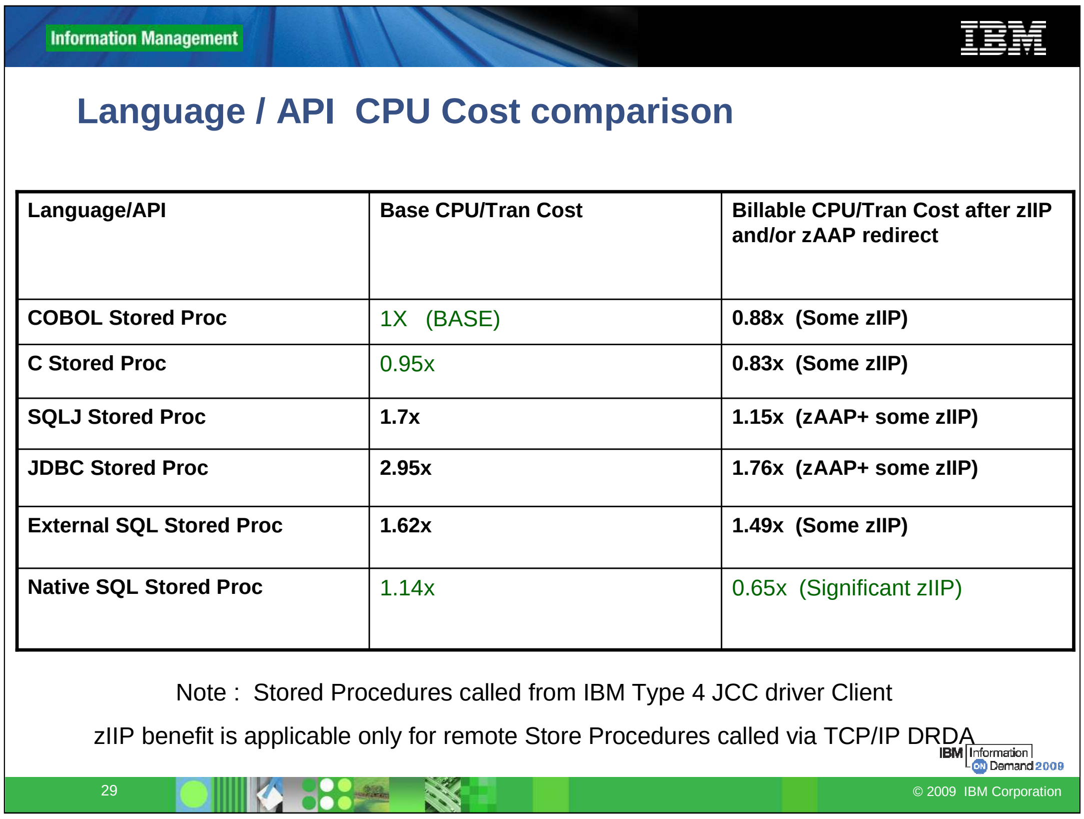
- Not a robust development language
- Requires DB2 9 for z/OS NFM
- Not eligible for zAAP redirect.

Language Pros/Cons Comparison Summary

	Robust Language	Throughput Performance	Billable Cost	Runs in the DB Engine	Specialty Engine	Supported in DB2 Version
COBOL	Yes	High	**Low**	**No**	**No zAAP** **Some zIIP**	**V8,V9**
C/C++	Yes	Above COBOL	**Below COBOL**	**No**	**No zAAP** **Some zIIP**	**V8,V9**
SQLJ	Yes	**Below native SQL**	**Above COBOL**	**No**	zAAP **Some zIIP**	**V8,V9**
JDBC	Yes	**Below External SQL**	**Above External SQL**	**No**	zAAP **Some zIIP**	**V8,V9**
External SQL	**No**	**Below SQLJ**	**Above SQLJ**	**No**	**No zAAP** **Some zIIP**	**V8,V9**
Native SQL	**No**	Below COBOL	Lowest	**Yes**	Significant zIIP	**V9 NFM**

Note : zIIP benefit is applicable only for remote Store Procedures via TCP/IP DRDA

Performance/Cost Comparison



Language / API CPU Cost comparison

Language/API	Base CPU/Tran Cost	Billable CPU/Tran Cost after zIIP and/or zAAP redirect
COBOL Stored Proc	1X (BASE)	0.88x (Some zIIP)
C Stored Proc	0.95x	0.83x (Some zIIP)
SQLJ Stored Proc	1.7x	1.15x (zAAP+ some zIIP)
JDBC Stored Proc	2.95x	1.76x (zAAP+ some zIIP)
External SQL Stored Proc	1.62x	1.49x (Some zIIP)
Native SQL Stored Proc	1.14x	0.65x (Significant zIIP)

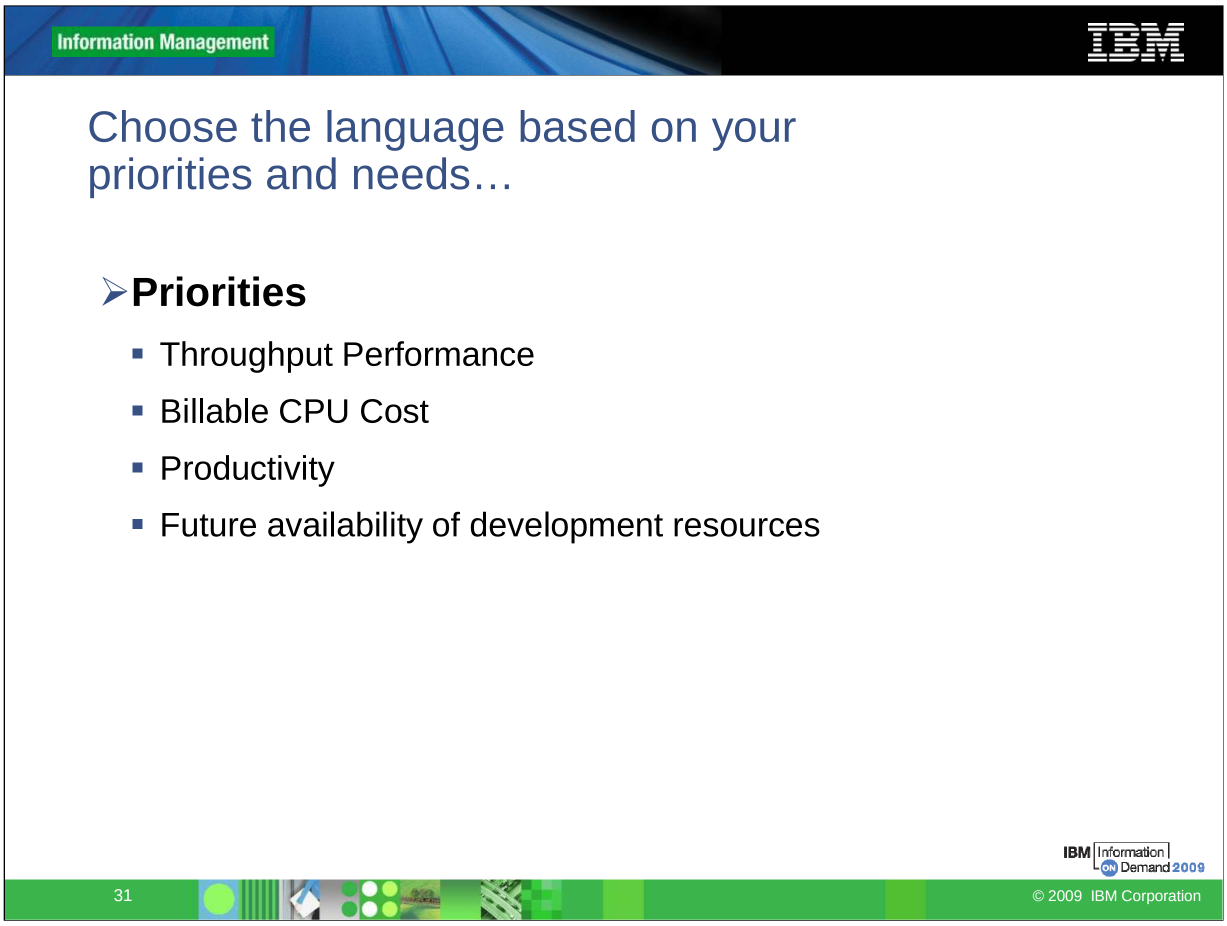
Note : Stored Procedures called from IBM Type 4 JCC driver Client

zIIP benefit is applicable only for remote Store Procedures called via TCP/IP DRDA

zIIP processor

- TCB execution is not eligible for offload
- Stored procedure benefits still apply!
  - Less network trips
  - Better concurrency
  - Static SQL packages invoked from dynamic
- Native SQL procedures in V9 are zIIP eligible *when invoked over DDF*
- Processing on DDF SRB is eligible
  - Commit, result sets
  - We Measured 10-13%

So, what language should I use?



Choose the language based on your priorities and needs…

➢**Priorities**
- Throughput Performance
- Billable CPU Cost
- Productivity
- Future availability of development resources

1) Always evaluate the cost of each invocation vs the benefit in return

2) You should avoid usage of nested procedures, if possible as we have a latch that serializes nested SP calls.

•When Creating:

- •Use STAY RESIDENT YES;
- •Use PROGRAM TYPE SUB;
- •Use SECURITY DB2;
- •Ensure that metadata stored procedures are NOT implicitly invoked due to data type mismatch i.e. SQLPROCEDURECOLS;
- •Specify ASUTIME LIMIT on every DB2 stored procedure definition;
- •Group your stored procedures in WLM AE, but no more than 512 in one WLM AE;
- •Use separate WLM AE for different LE runtime options;

•When Executing

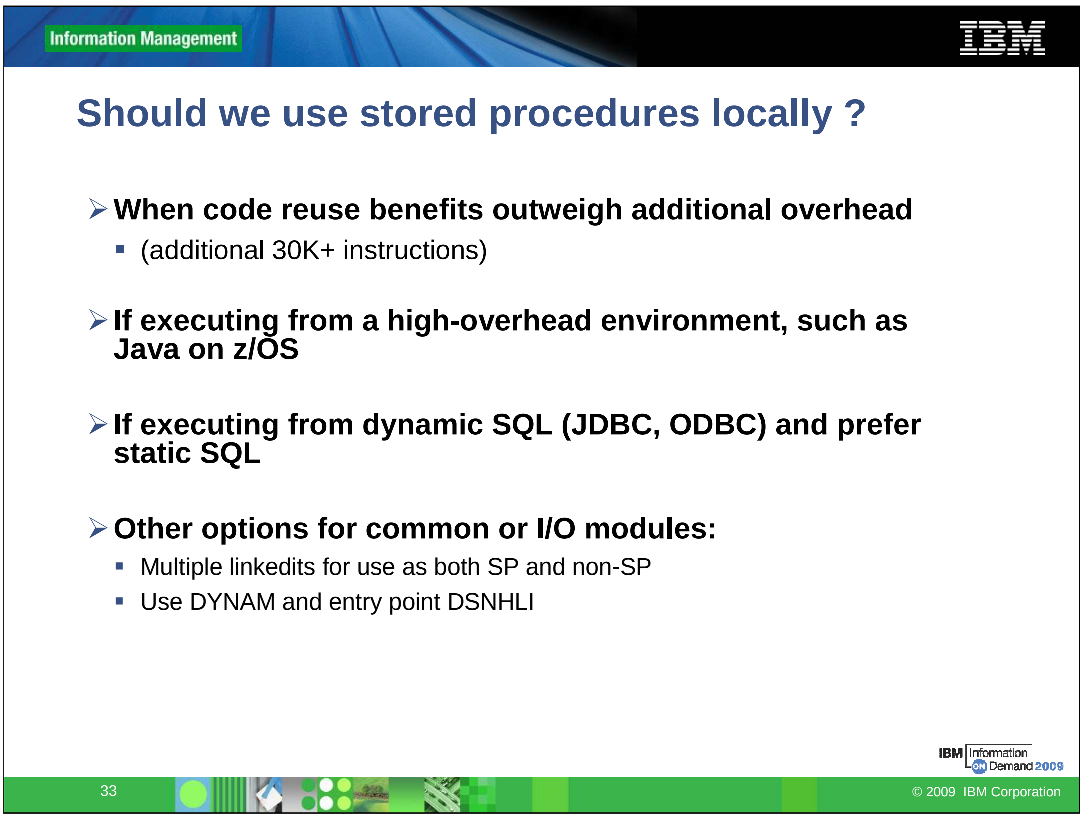
- •Use the stored procedure authorization cache (zPARM CACHERAC);
- •COMMIT ON RETURN NO for stored procedures that are called locally;
- •COMMIT ON RETURN YES for stored procedures that are called from distributed client applications in environments where sysplex workload balancing is not used;
- •Maximize the number of procedures that can run concurrently in a WLM AE;
- •Limit the number of times that a stored procedure can terminate abnormally:
  - •Specify MAX ABEND COUNT field on installation panel DSNTIPX
  - •Specify STOP AFTER FAILURES -- option on the ALTER or CREATE PROCEDURE statement
    Note that STOP AFTER FAILURES overrides MAX ABEND COUNT

Stored Procedure development Tool Links

- DB2 9 for z/OS Stored Procedures: Through the CALL and Beyond **SG24-7083 Through the call and beyond**
  http://www.redbooks.ibm.com/abstracts/sg247604.html?Open

- **Rational Developer for System Z**
  - http://www-01.ibm.com/software/awdtools/rdz/about/?S_CMP=wspace

- **IBM Integrated Data Mangement**
  - http://www-01.ibm.com/software/data/optim/



Should we use stored procedures locally ?

- **When code reuse benefits outweigh additional overhead**
  - (additional 30K+ instructions)
- **If executing from a high-overhead environment, such as Java on z/OS**
- **If executing from dynamic SQL (JDBC, ODBC) and prefer static SQL**
- **Other options for common or I/O modules:**
  - Multiple linkedits for use as both SP and non-SP
  - Use DYNAM and entry point DSNHLI

Application programmers often design to use stored procedures from local z/OS applications, even though the design point was for remote invocation. There are plenty of scenarios where this makes sense.