#SHAREorg

Making the Most Out of Native SQL Procedures

Maryela Weihrauch, Distinguished Engineer
IBM Corporation

Meg Bernal, Senior Software Engineer
IBM Corporation

Wednesday, February 6, 2013

Session Number 12802

SHARE in San Francisco 2013

Please note

Agenda

- Introduction to Native SQL Procedures
- Enhancements
  - XML
  - Support for Scalar Functions
  - Performance
- Operational Challenges
  - Monitoring
  - Change Management
- Future Outlook
  - Array Data Type
  - Autonomous Transactions

Introduction to Native SQL Procedures

General Stored Procedure Benefits

- Provides modularity in application development
- Data will always be processed in a consistent way, according to the rules defined in the procedure
- Enforcement of business rules
  - i.e. use procedures to define set of business rules common to many applications
  - Can be an alternative to constraints and triggers
- Improved application security
  - i.e. sensitive business logic runs on the DB2 server, end-users are authorized to execute the procedure
- Reduces network traffic for distributed applications i.e. many SQL statements can be encapsulated in a single procedure

Introduction to SQL Procedures

- What is an SQL Stored Procedure?
  - A stored procedure that contains only SQL statements
  - May use SQL *control* statements to write the logic part of the program (WHILE, IF, etc.)
  - **SQL Procedural Language** or **SQL PL**

- Two types of SQL Stored Procedures
  - **External** SQL Procedures (from V5 on) - Generated C program which runs in a WLM environment
  - **Native** SQL Procedures (from DB2 9) - The SQL procedure logic runs in the DBM1 address space

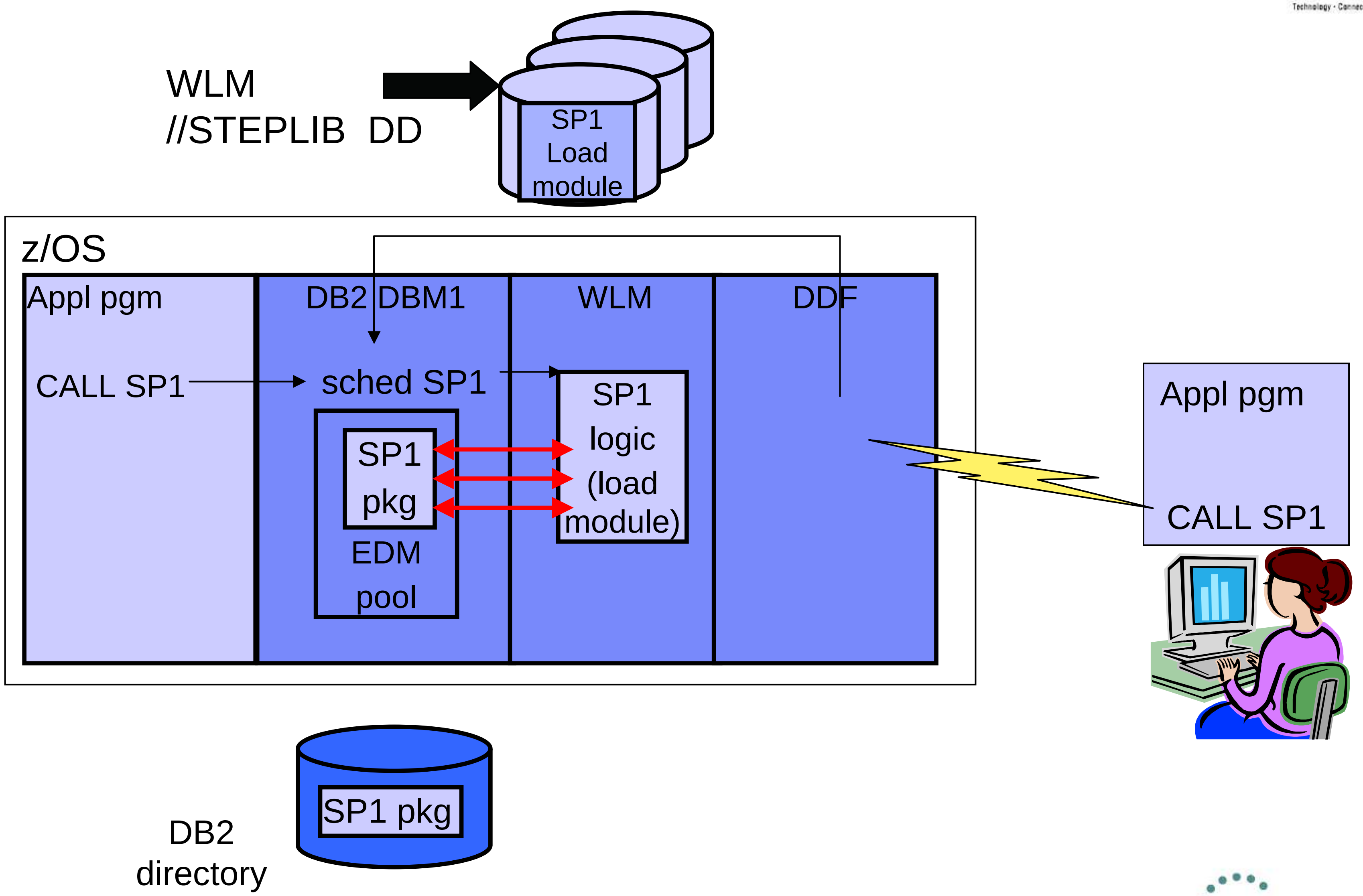
History

	Pre-V9	V9	V10
Stored Procedure	External for Host Languages (C, PLI, JAVA, etc) External SQL	Native SQL	Native SQL Enhancements
Scalar Function	External Inline		Non-Inline SQL
Table Function	External		SQL

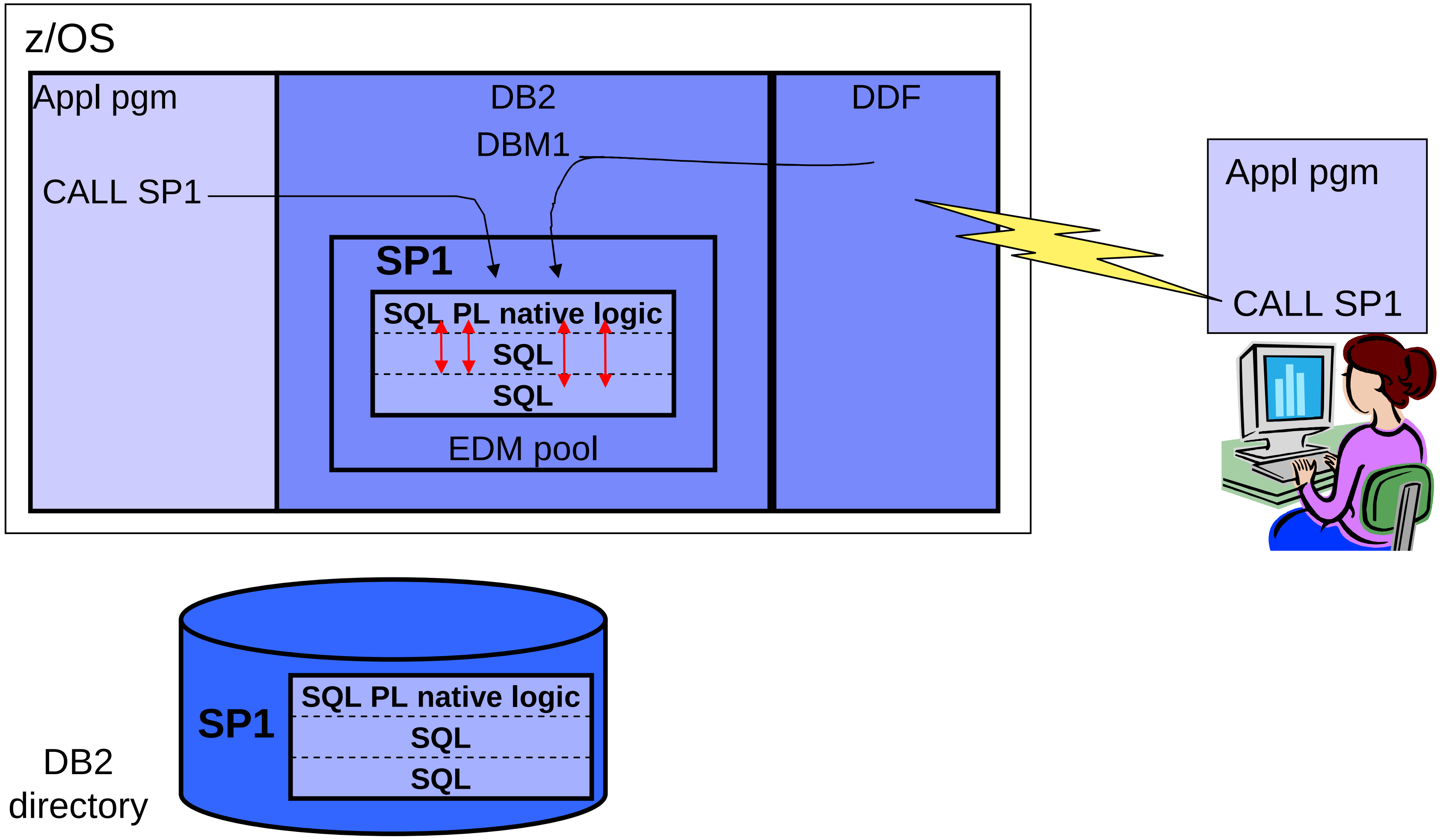
External and Native SQL Procedure Comparison

	External	Native
Preparation	• Multi-step (Precompile, compile, link-edit, BIND, DDL) • Requires C compiler	• Single step • DDL
Execution	Requires WLM environment, load module	Runs entirely within the DB2 engine

External Stored Procedure Processing



Native SQL Procedure Processing



*native SQL procedures do not run IN the WLM address space but are still running UNDER the WLM

When to Use Native SQL Procedures

- Go To Option When ….
  - SQL intensive
  - Contains minimal application logic
  - Lowest billable cost (for remote) and productivity are the most important priorities i.e. the stored procedure execution itself is zIIP off-loadable
  - Classic Example is TPC-C:
    - An OLTP application for order-entry consisting of entering and delivering orders, recording payments, checking the status of orders, and monitoring the level of stock at the warehouses
- Consider Alternatives When ….
  - Contains significant amount of application logic
    - Many IF/WHILE/CASE/REPEAT statements
  - Executes math, string, file manipulation functions

Native SQL Procedure Enhancements in DB2 10

XML Parameters and Variables

- XML is available as a procedure parameter or as an SQL variable inside a Native SQL Procedure

```
CREATE PROCEDURE XMLPROC(IN XMLPARM XML, IN VCHPARM VARCHAR(32000))
LANGUAGE SQL
BEGIN
  DECLARE myXMLVar XML;

  IF (XMLEXISTS('$x/ITEM[value < 200]' passing by ref XMLPARM as "x")) THEN
    INSERT INTO T1 VALUES(XMLPARM);
  END IF;

SET myXMLVar =
  XMLDOCUMENT(XMLELEMENT(NAME "ORDER",
                                                       XMLCONCAT(PARM1, XMLPARM)));

INSERT INTO T1 VALUES(myXMLVar);

END #
```

*Consider the following set of XML APARs: PM66042, PM65046, PM65366, PM66040, PM66142

Enhanced Support for SQL Scalar Functions

- SQL Scalar Functions are enhanced in NFM
  - May contain logic using SQL PL control statements
  - Non-inline, package
  - Parser determines type of scalar function
  - Example – Reverse a string

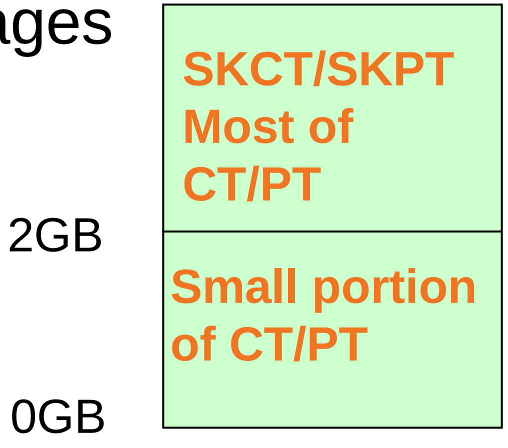
```
CREATE FUNCTION REVERSE(INSTR VARCHAR(20))
RETURNS VARCHAR(20)
DETERMINISTIC NO EXTERNAL ACTION CONTAINS SQL
BEGIN
    DECLARE REVSTR, RESTSTR VARCHAR(20) DEFAULT '';
    DECLARE LEN INT;
    IF INSTR IS NULL THEN
        RETURN NULL;
    END IF;
    SET (RESTSTR, LEN) = (INSTR, LENGTH(INSTR));
    WHILE LEN > 0 DO
      SET (REVSTR, RESTSTR, LEN) =
             (SUBSTR(RESTSTR, 1, 1) CONCAT REVSTR,
              SUBSTR(RESTSTR, 2, LEN - 1),
              LEN - 1);
     END WHILE;
RETURN REVSTR;
END
```

Performance – Virtual Storage and CPU

- Virtual storage improvement
  - Most multiple instances storage moved to agent local ATB pool
- CPU reduction
  - General CPU reduction
  - Specific CPU reduction in commonly used areas in SQL PL
    - SET statement optimizations
      - *With BIFs moved to Section 1 (CPU for TPC-E reduced by 8.3%)*
      - *Chained SET statement support (multiple values can be set in a single statement)*
      - *CONCAT(S1,S2)*
    - Optimization in SELECT x from SYSDUMMY1
  - SQLPL workloads at the lab show 10-20% CPU reduction
    - A workload using SET statements, IF statements, and SYSDUMMY1 in native SQL procedures has shown up to 20% CPU reduction.

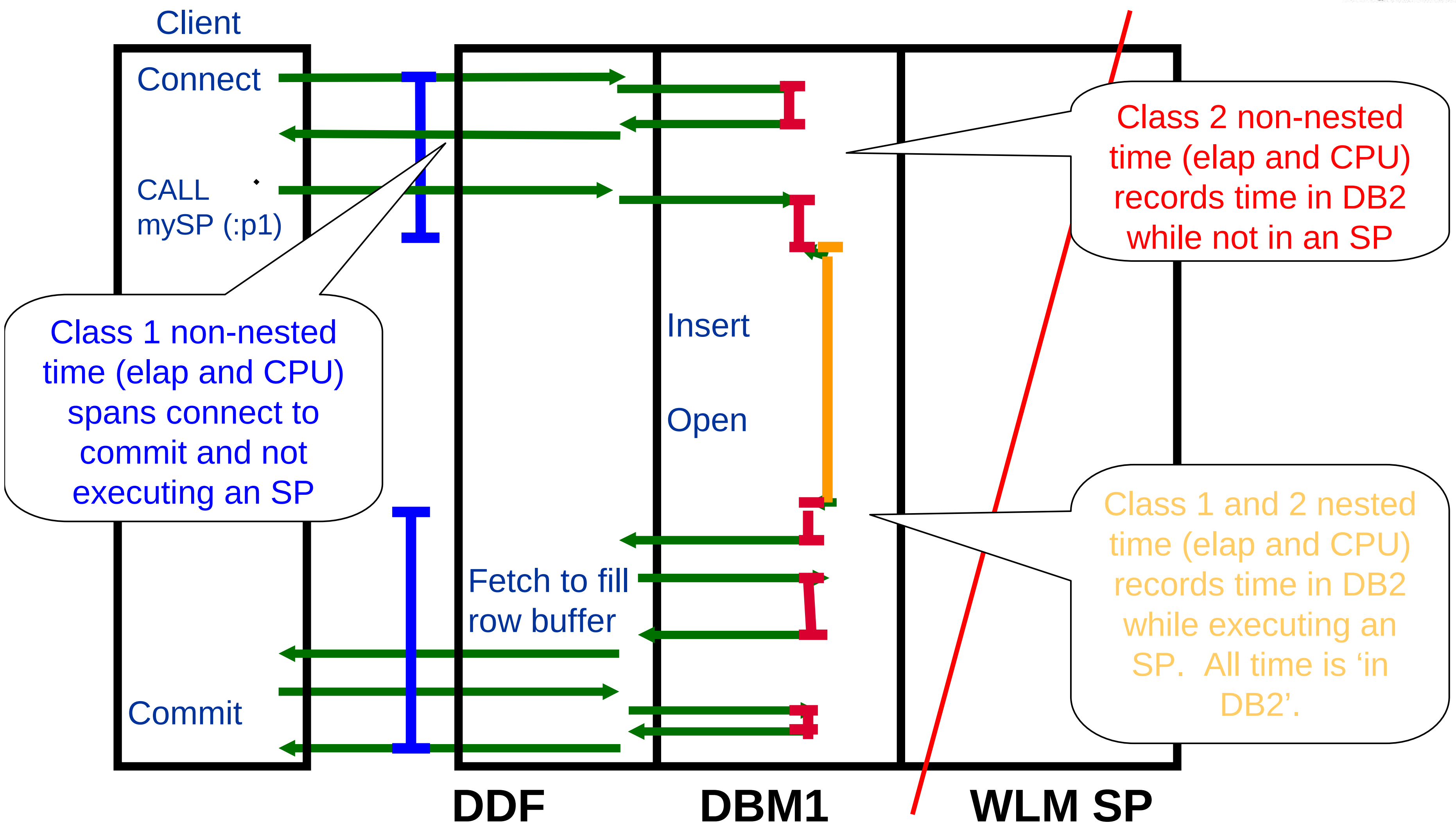
Performance - EDM Pool Storage Impact

- One additional section per package (named Section 1) which has
  - Control logic (IF/THEN/ELSE, CASE)
  - Assignment statements if no scalar full selects and no UDFs
- Size of Section 1 depends on size of control logic
- When stored procedure invoked, most of Section 1 loaded as part of above the bar storage
- For all other statements in the procedure, the EDM pool would go up just like other packages



Operational Challenges

Performance Reporting – Native SQL Stored Procedure

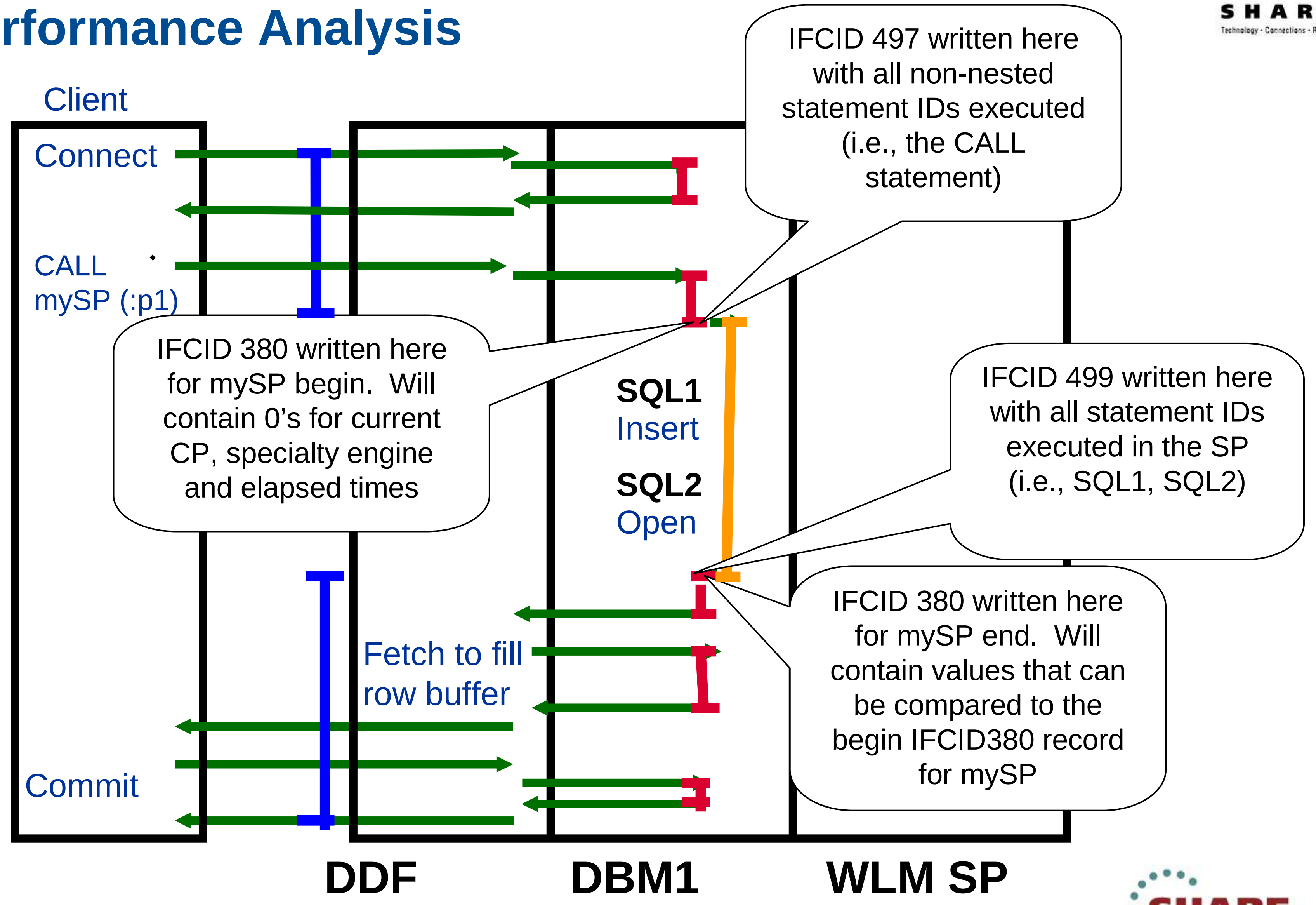


Client
Connect
CALL mySP (:p1)
Commit
Insert
Open
Fetch to fill row buffer
DDF
DBM1
WLM SP
Class 1 non-nested time (elap and CPU) spans connect to commit and not executing an SP
Class 2 non-nested time (elap and CPU) records time in DB2 while not in an SP
Class 1 and 2 nested time (elap and CPU) records time in DB2 while executing an SP. All time is 'in DB2'.

Enhanced Instrumentation for Stored Procedure Performance Analysis

- PM53243 (DB2 10) New IFCIDs 380 and 381 are created for Stored Procedure and User-Defined Function detailed information, respectively.
- These new trace records:
  - Identify the Stored Pprocedure or UDF beginning or ending
  - Include the current CP, specialty engine, and elapsed time details for nested activity
- The IFCID 380 and 381 trace records can be used to determine the CP, specialty engine, and elapsed time for a given SP or UDF invocation
- Additionally PM53243 (DB2 10) added IFCID 497, 498, 499 for SQL drill down analysis. These records contain the dynamic or static statement IDs for non-nested SQL, UDF, and SP work, respectively.
- The statement IDs can be correlated to IFCID 316 dynamic statement or IFCID 401 static statement cache data.

Enhanced Instrumentation for Stored Procedure Performance Analysis

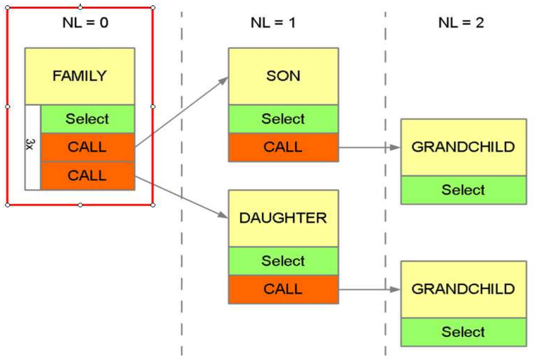


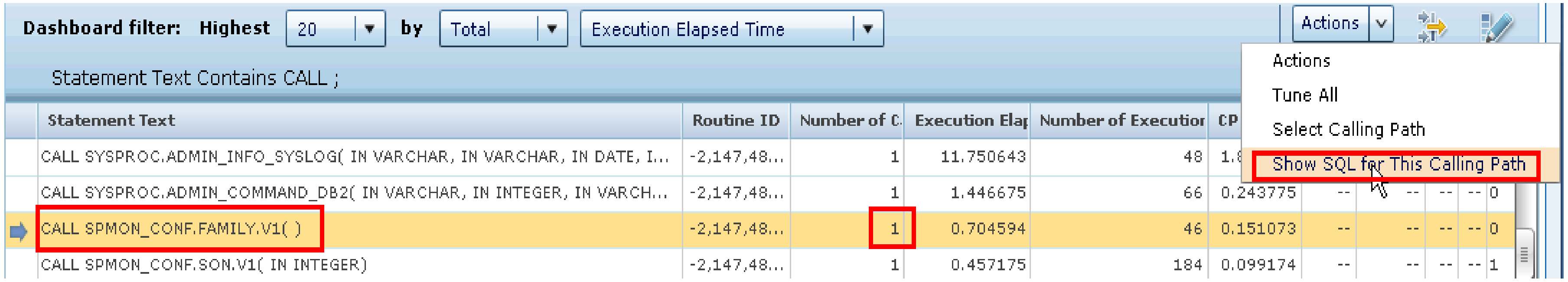
Client
Connect
CALL mySP (:p1)
Fetch to fill row buffer
Commit
SQL1 Insert
SQL2 Open
DDF
DBM1
WLM SP
IFCID 497 written here with all non-nested statement IDs executed (i.e., the CALL statement)
IFCID 380 written here for mySP begin. Will contain 0's for current CP, specialty engine and elapsed times
IFCID 499 written here with all statement IDs executed in the SP (i.e., SQL1, SQL2)
IFCID 380 written here for mySP end. Will contain values that can be compared to the begin IFCID380 record for mySP

Monitoring Stored Procedures with OMPE

- The new DB2 instrumentation records for **Stored Procedures** are ingested by the OMPE Collector, aggregated on a system level, and returned to the OPM Repository Server.
- The OMPE Collector processing includes the sequencing logic and the calculation of elapsed times for the different accounting class times written in the IFI records as timestamps.
- In parallel, the IFCID 316/401 data for the Statement Caches is collected and a correlation to the executed Stored Procedure statements is made.
- Full RECTRACE support for **all new IFCIDs** is provided

Show SQL executed by a SP



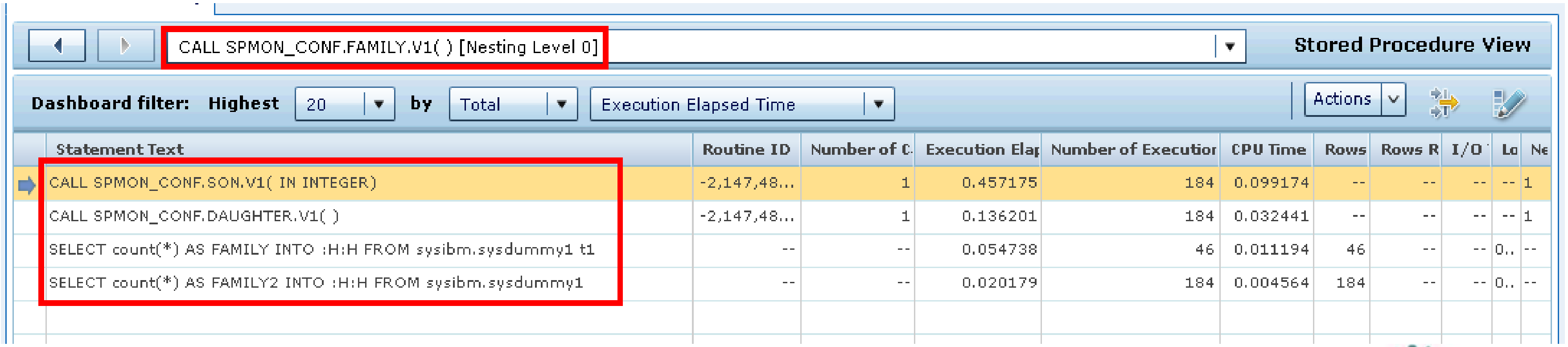


Dashboard filter: Highest 20 by Total Execution Elapsed Time

Statement Text Contains CALL ;

Actions / Tune All / Select Calling Path / Show SQL for This Calling Path

Statement Text	Routine ID	Number of C	Execution Elap	Number of Executior	CP
CALL SYSPROC.ADMIN_INFO_SYSLOG(IN VARCHAR, IN VARCHAR, IN DATE, I...	-2,147,48...	1	11.750643	48	1.8
CALL SYSPROC.ADMIN_COMMAND_DB2(IN VARCHAR, IN INTEGER, IN VARCH...	-2,147,48...	1	1.446675	66	0.243775
CALL SPMON_CONF.FAMILY.V1( )	-2,147,48...	1	0.704594	46	0.151073
CALL SPMON_CONF.SON.V1(IN INTEGER)	-2,147,48...	1	0.457175	184	0.099174



CALL SPMON_CONF.FAMILY.V1( ) [Nesting Level 0]

Stored Procedure View

Dashboard filter: Highest 20 by Total Execution Elapsed Time

Statement Text	Routine ID	Number of C	Execution Elap	Number of Executior	CPU Time	Rows	Rows R	I/O	Lo	Ne
CALL SPMON_CONF.SON.V1(IN INTEGER)	-2,147,48...	1	0.457175	184	0.099174	--	--	--	--	1
CALL SPMON_CONF.DAUGHTER.V1( )	-2,147,48...	1	0.136201	184	0.032441	--	--	--	--	1
SELECT count(*) AS FAMILY INTO :H:H FROM sysibm.sysdummy1 t1	--	--	0.054738	46	0.011194	46	--	--	0..	--
SELECT count(*) AS FAMILY2 INTO :H:H FROM sysibm.sysdummy1	--	--	0.020179	184	0.004564	184	--	--	0..	--

Change Management - Problem

- After widespread adoption of SQL PL, customers running into operational issues in managing SQL PL applications
  - Source code management
    - No good way to hold source code outside of DB2
  - Deployment
    - BIND PACKAGE DEPLOY
      - *Needs DRDA*
      - *Can change only few properties at target server (QUALIFIER, SCHEMA)*
      - *Properties like PATH, VALIDATE cannot be changed on target server*
    - Many forms of DDL, difficult to know which one needed
      - *CREATE PROC*
      - *ALTER PROC ADD VERSION/REPLACE VERSION*

Change Management - Solution

- Provides SQLPL Source Deployment capability
- Introduces a set of sample REXX services that can be combined to perform these basic SQLPL change management processes:
  - SQLPL source extraction (to a file, to a string)
  - SQLPL source transformation and modification (Change the DDL verb form, schema, version, and options)
  - SQLPL source deployment (from a file, from a string)
  - SQLPL file transforms (V-format, F-format, HFS)
  - SQLPL listing services (precompiler)
  - SQLPL source description service (TOC used for editing)
- Replaces BIND DEPLOY usage or complements it

Change Management – Solution (cont'd)

- V9 APAR PM29226 distributes the set of SQLPL source code management services
  - Upgrades the DB2 sample job DSNTEJ67 to exploit the new services
  - Demonstrates the External to Native SQLPL migration process
  - Extract an external SQL proc to file, source deploys a native SQL SP that generates native options, modify the SQL proc source file, produce a native SQLPL listing.

Future Outlook

Future Outlook – V11

- ARRAY Data Type Support
  - Ability to Create a Collection of Elements
  - Provides a Mechanism to Convert Arrays to Tables With The collection-derived-table
  - Functionality Allows User to Convert Tables into Arrays With ARRAY_AGG Aggregate Function
  - New SQL to Manipulate & Ascertain Information About Arrays With A New Set of Array Scalar Functions
- Autonomous Transactions
  - Native SQL Procedures that run in their own unit of work
    - May perform SQL, COMMITs, ROLLBACK it's own SQL
  - No uncommitted changes from it's caller are seen
  - Locks are not shared between the caller and Autonomous procedure
  - Upon completion of the Autonomous procedure, a COMMIT is driven for the Autonomous procedure work
    - The caller's work is untouched

Summary

Summary

- Native SQL PL Routines (procedures/functions) are the way of the future for SQL intensive applications
- Native SQL PL Routines offer benefits in most scenarios (reduced cost, increased performance, easier development and maintenance)
- If you are already using External SQL procedures, migration to Native SQL procedures is worthwhile
- Consider Native SQL PL Routines for SQL intensive new application development
- If your parameter lists are long and/or you use temp tables to return data from your procedures, Array Data Types can simplify your procedures
- Autonomous Native SQL PL Procedures provide a capability to separate units of work

#SHAREorg

SHARE
Technology · Connections · Results

Making the Most Out of Native SQL Procedures

Maryela Weihrauch, Distinguished Engineer
IBM Corporation

Meg Bernal, Senior Software Engineer
IBM Corporation

Wednesday, February 6, 2013

Session Number 12802

SHARE in San Francisco 2013