IDUG

2026

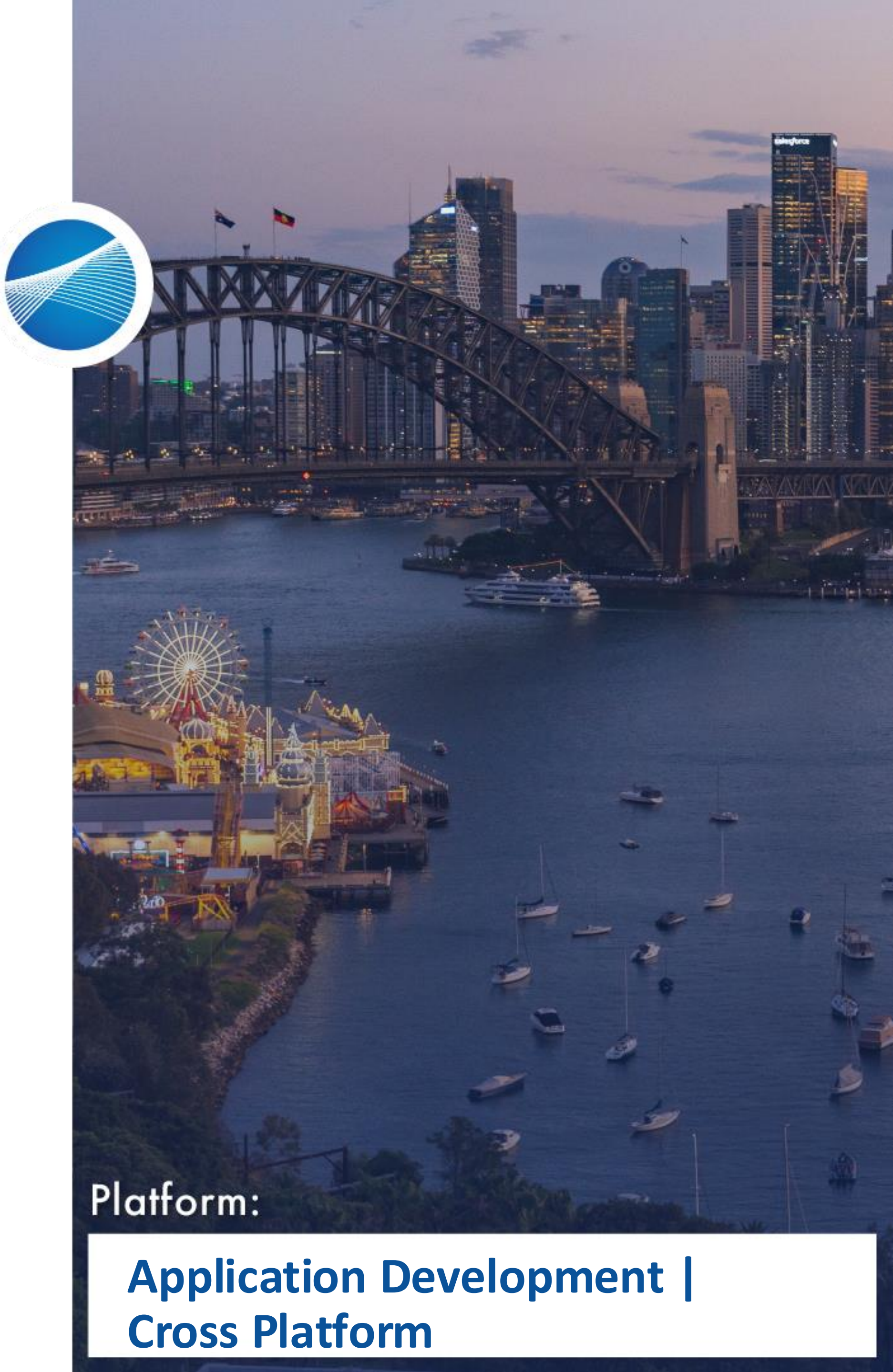
Sydney | March 16 - 18

AU Db2 TECH CONFERENCE

Low Risk Mainframe Application Modernization Using Java Accessing Db2 for z/OS

Anthony Ciabattoni, IBM

Session Code: C04

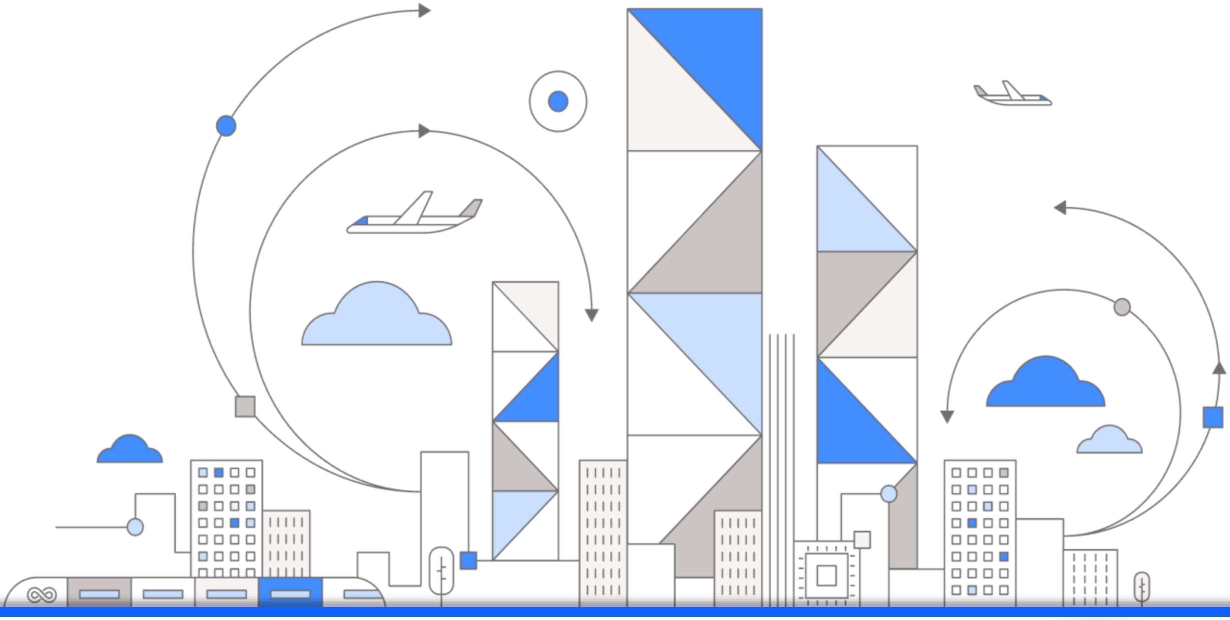


Platform:

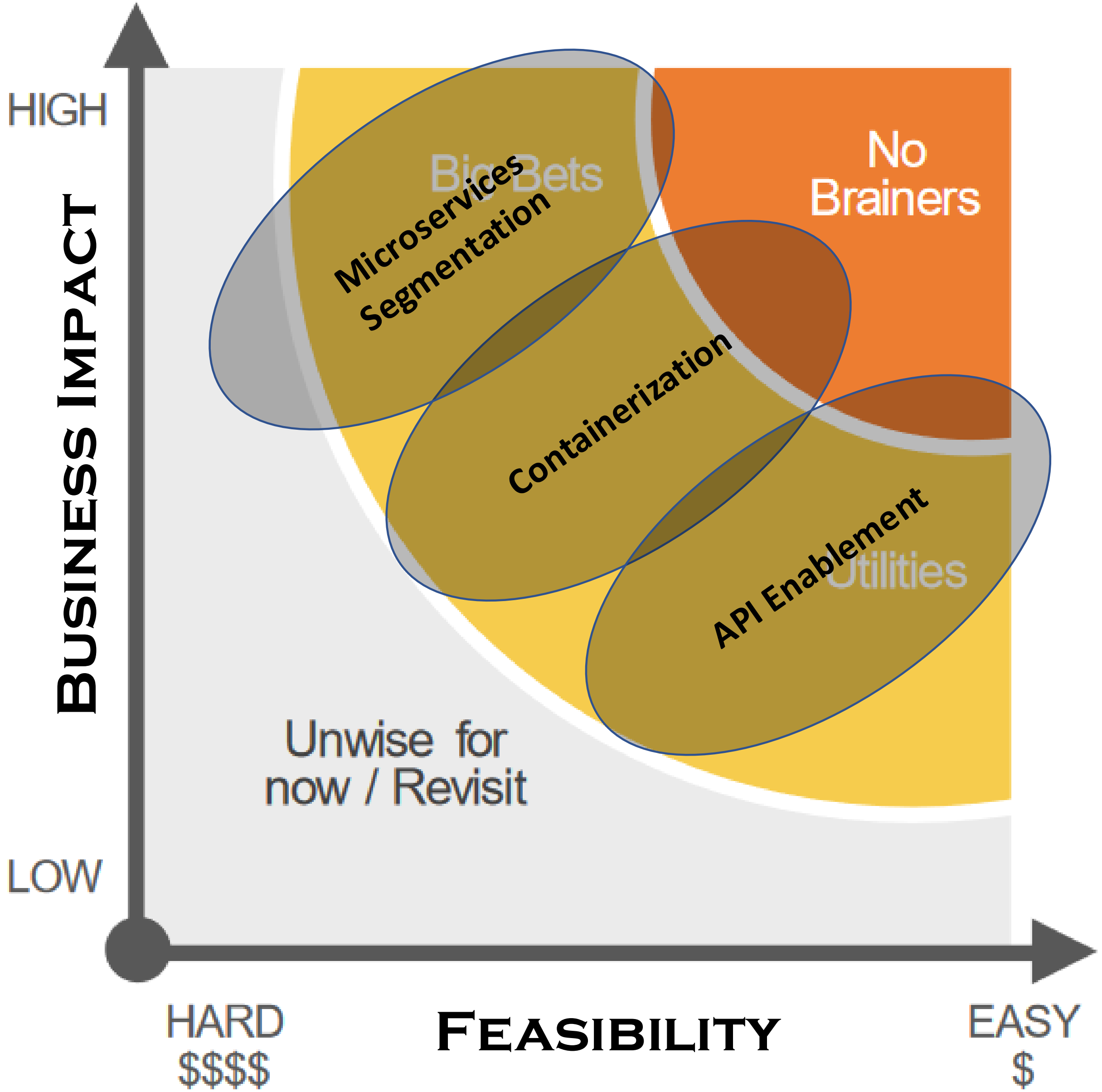
Application Development | Cross Platform

Agenda

- Approaches for modernizing Cobol applications using Java
- Cobol-Java inter-language capabilities
- Optimizations and best practices for accessing data stored in Db2 for z/OS
- Questions



Operational application transformation approaches



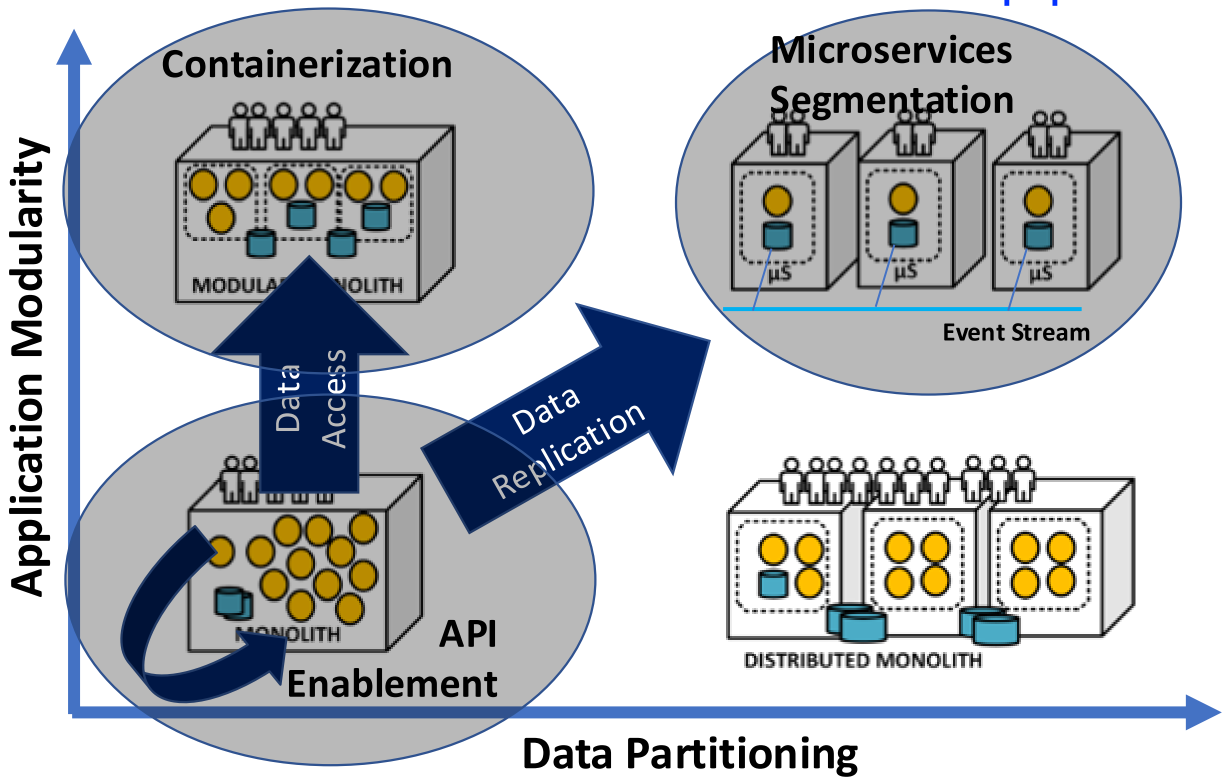
There are three major approaches to Core System Transformations. Each client's environment, end state requirements, risk tolerance, financial justification, etc will define the business impact of each of these approaches – clearly hybrid approaches and variants are viable

- ***API Enablement***: Simplest approach is for these core systems to provide a set of consumable APIs for the Systems of Engagement. The reverse is also viable where the Core System consumes APIs provided by external systems
- ***Containerization***: The systematic movement of application logic to containers that alternately can run anywhere. Preserves data and batch processes
- ***Microservices Segmentation***: Total restructuring of the Core System into a cloud native systems. This is extremely risky and is best achieved by starting with Containerization and developing a data model that can be partitioned. Eventual consistency of data is a fundamental change whose impact needs to be clearly understood

Protecting/leveraging data will drive transformation approach

Data Access vs Data Replication Considerations

- Establishing and maintaining data replication pipeline is expensive and time consuming
- Maintaining a data replication pipeline typically creates data quality and data latency challenges for consuming applications
- Both Microservices and containerization enable
  - Cloud native
  - Agile processes
- Accessing data in place accelerates the transformation, also improves opportunity for success
- Data access preserves the existing data management and recovery processes



Data Consistency versus Eventual Consistency Considerations

- Data consistency reduces the amount of compensating application logic and/or business processes that need to be developed for systems that are out of sync
- Data Consistency addresses the double spend problem
- Eventual consistency enables systems to be developed independent of each other however can result in data quality issues

Cobol-Java Inter-Language Capabilities - Background

- As part of many organizations' enterprise modernization strategy, Java is the preferred language to implement new or changed business functions

- Because many are not starting from scratch, the new business function written in Java need to seamlessly integrate and interoperate with existing business logic written in COBOL or C

- **220+ billion lines of COBOL**
  - COBOL accounts for more than **70%** of the business transactions that take place in the world today[1]

- An application developer needs to be able to focus on writing business logic regardless of the chosen language

- A system programmer should be able to integrate the new application (existing application with Java extensions) into the existing and proven systems management framework (e.g. scheduling, workload management, security, monitoring, problem determination)

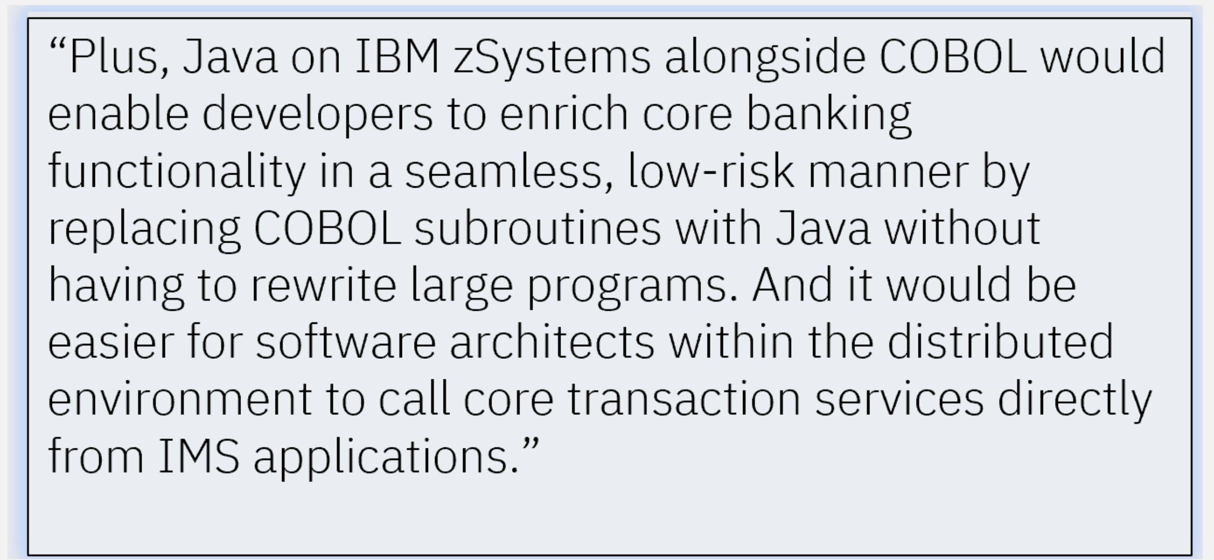
- 1 - http://cobolpros.com/the-need-for-cobol/

COBOL / Java Interoperability Support

- The ability to run AMODE 31 COBOL programs with AMODE 64 Java programs as part of a single application in the same address space
- **Benefits include:**
  - Condition handling between COBOL and Java portions of application
  - Improved serviceability - both COBOL and Java program diagnostics provided in a single dump
  - Same security context for entire application including COBOL and Java programs.
- **Goal:**
  - Provides full application transparency to allow customers to perform enterprise modernization
    - COBOL runtime batch
    - Java Native Interface (JNI)
    - Other language runtimes
    - IMS/CICS
    - WAS

COBOL / Java Interoperability Support

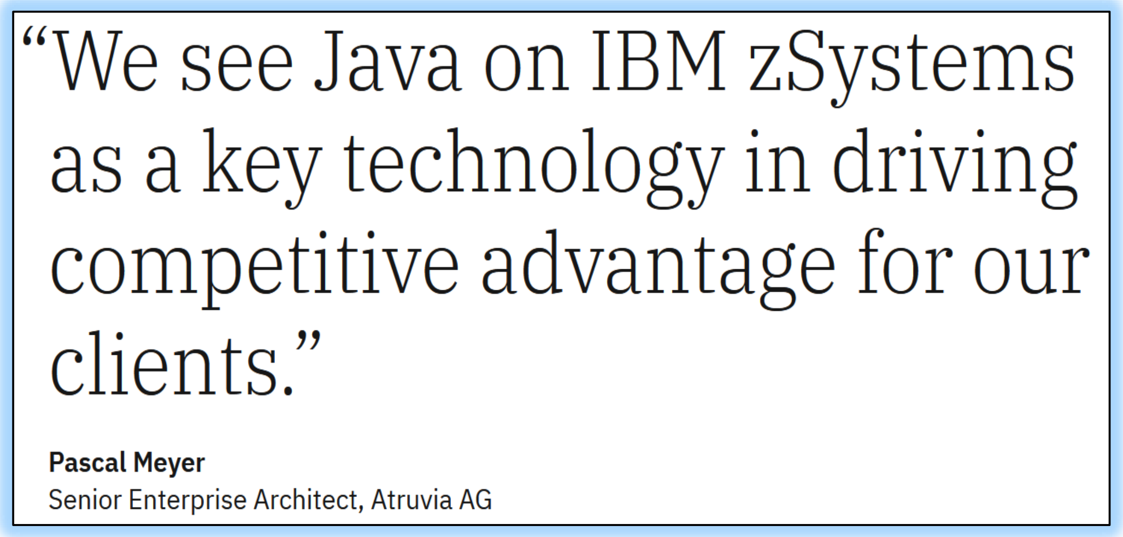
ATRUVIA



“Plus, Java on IBM zSystems alongside COBOL would enable developers to enrich core banking functionality in a seamless, low-risk manner by replacing COBOL subroutines with Java without having to rewrite large programs. And it would be easier for software architects within the distributed environment to call core transaction services directly from IMS applications.”

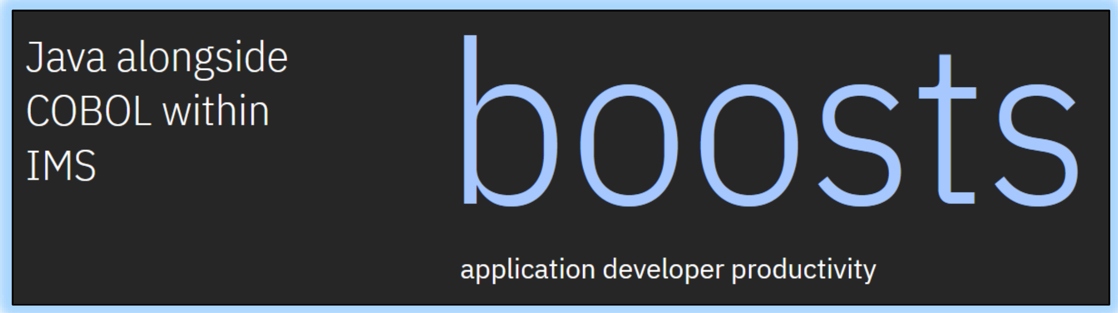
https://www.ibm.com/case-studies/atruvia-ag/

(7-minute read)



“We see Java on IBM zSystems as a key technology in driving competitive advantage for our clients.”

Pascal Meyer
Senior Enterprise Architect, Atruvia AG



Java alongside COBOL within IMS

boosts

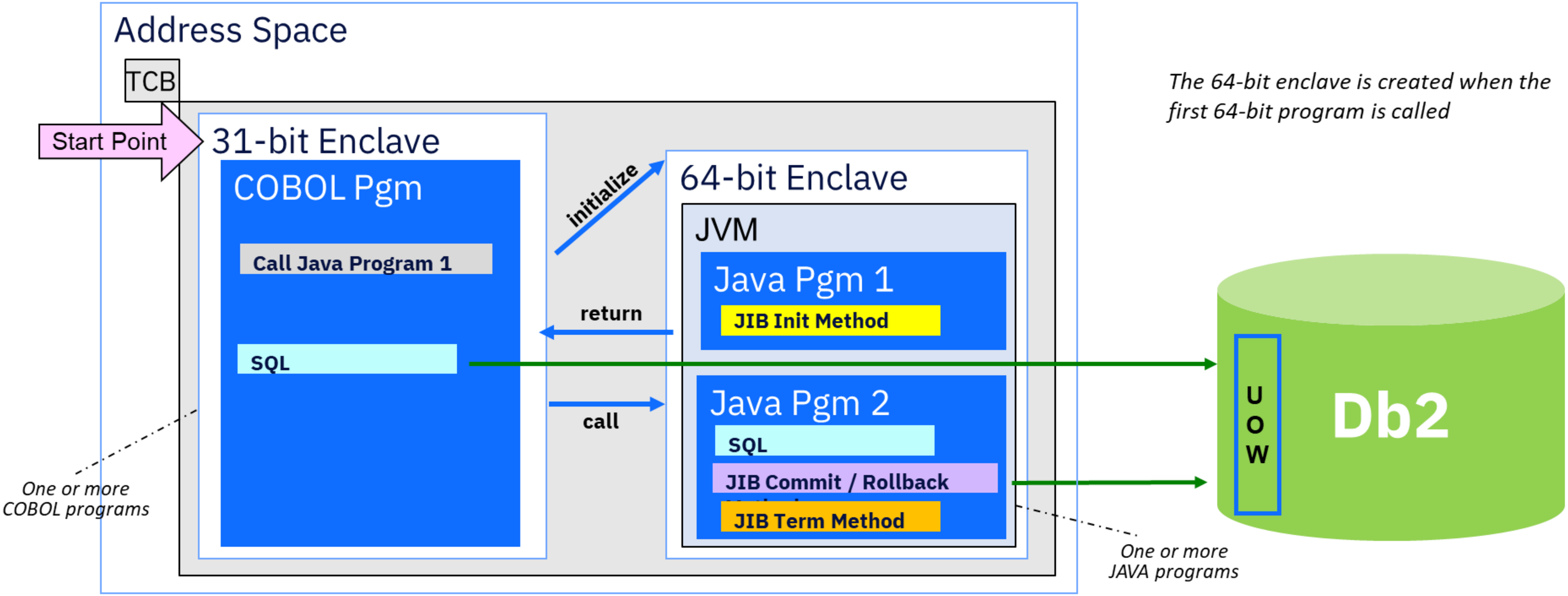
application developer productivity

COBOL / Java Interoperability – Db2 Connection Sharing Support

- Db2 Connection Sharing
  - Requires JCC Type 2 Driver support class in Db2 for environment
  - The sharing of a Db2 connection in a 31/64-bit (COBOL/Java) interoperability environment for batch with transactional integrity as part of a single Unit of Work (UOW)
    - This capability is available under existing transaction managers (IMS, CICS, WAS) but is not available in pure batch
  - Provide a framework called the Java Interlanguage Batch (JIB) environment that provides transactional support in batch using RRS (Resource Recovery Services)
  - Java Interlanguage Batch will provide four new Java methods:
    - JIB Initialization method - must be called before any Db2 SQL calls are made.
      - ✓ Will register with RRS to establish an RRS Global Transaction and an RRSAF connection for this TCB
    - JIB Commit / Rollback - All commits and rollbacks wil need to be done by calling these methods (uses RRS Commit/Rollback under the covers)
    - JIB Termination - deregister from RRS, disconnect from Db2

COBOL / Java Interoperability – Db2 Connection Sharing Support

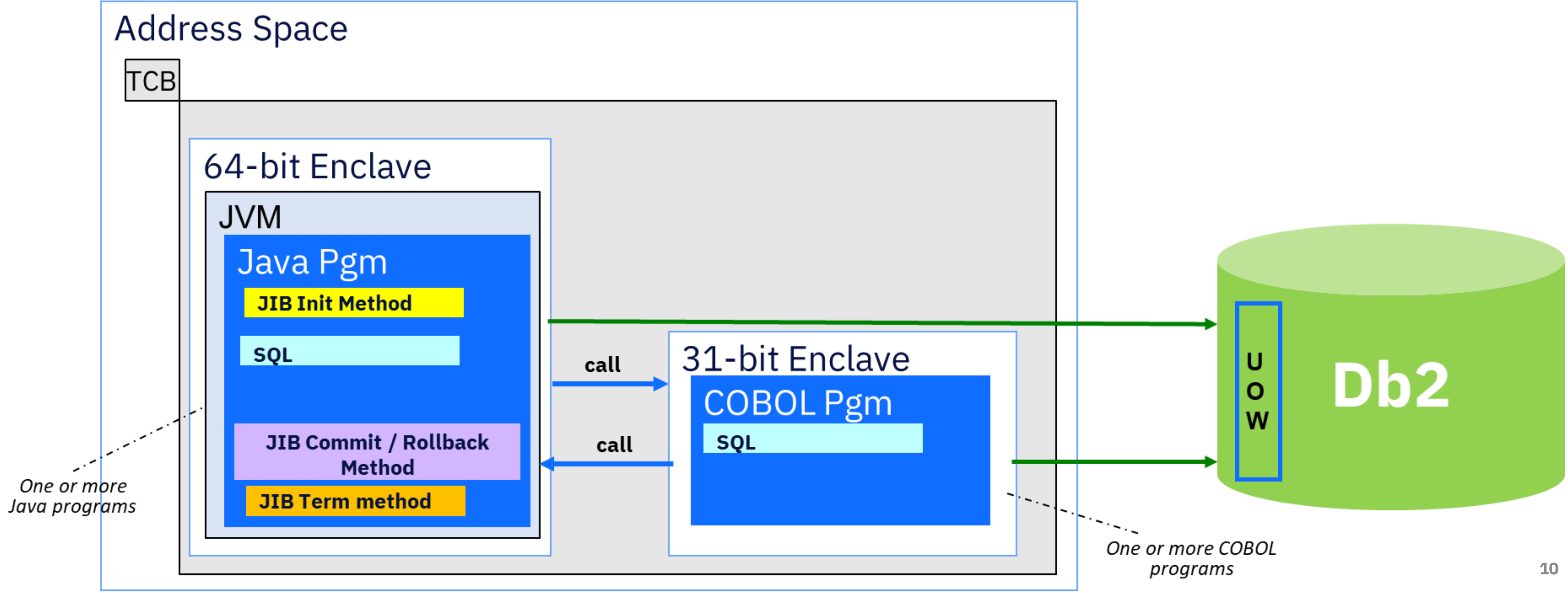
- **Scenario A:** 31-bit COBOL program → 64-bit Java program
  - COBOL program must call JIB Initialization method before making any Db2 calls
  - All Commits/Rollbacks need to be done by calling JIB Commit/Rollback methods



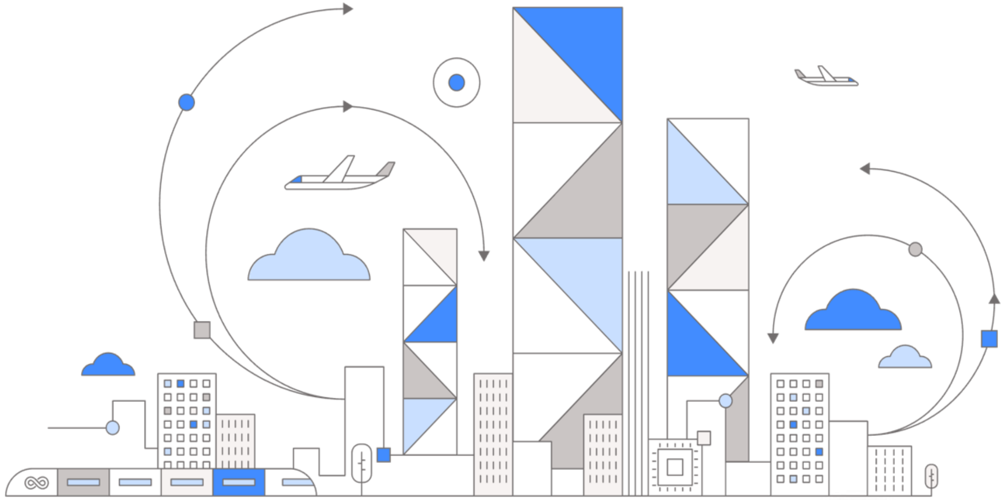
The 64-bit enclave is created when the first 64-bit program is called

COBOL / Java Interoperability – Db2 Connection Sharing Support

- Scenario B: 64-bit Java program → 31-bit COBOL program
  - Java program must call the JIB initialization method before making any Db2 calls
  - All Commits/Rollbacks need to be done by calling JIB Commit/Rollback methods



Optimizations and best practices for accessing data stored in Db2 for z/OS



Use Client Accounting Information Fields

- Can be used in WLM, RLF and profile definition and performance monitoring
- WebSphere Application Server supports explicit and implicit setting of client information
  - Example how to call explicitly

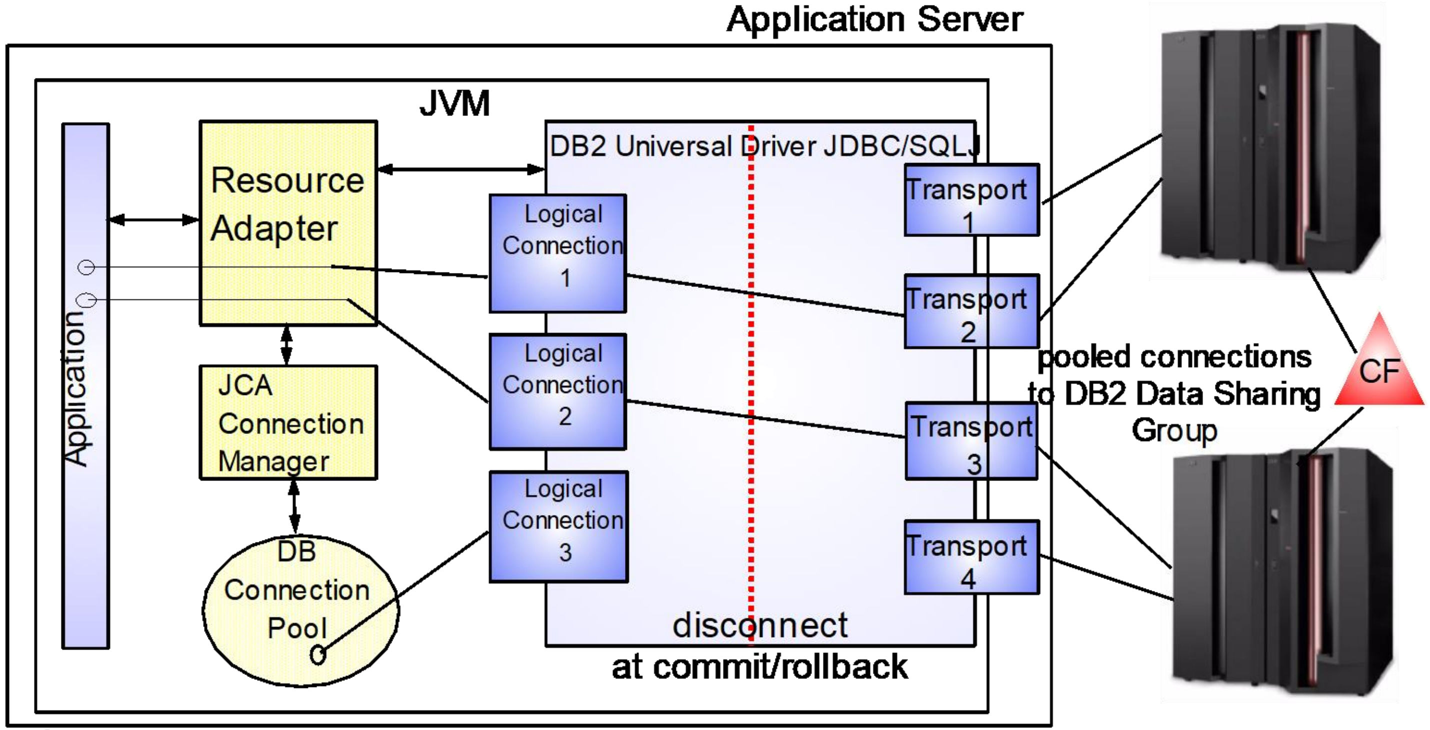
```
WSConnection conn = (WSConnection) ds.getConnection();
props.setProperty(WSConnection.CLIENT_ID, "user123");
conn.setClientInformation(props);
```

  - Example how to call implicitly by turning on WebSphere Trace Group

```
WAS.clientinfo=all=enabled or
WAS.clientinfopluslogging=all=enabled
```

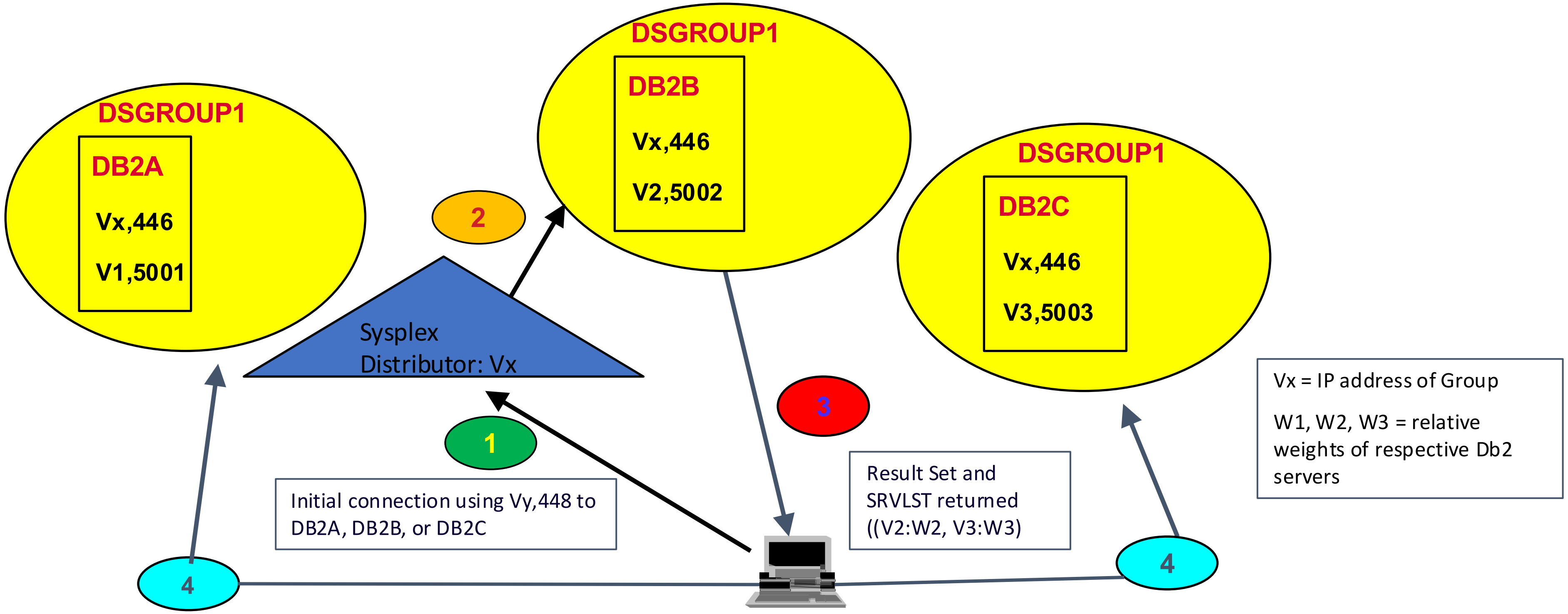
Remote Java Access to a Db2 Data Sharing Group

- Enable Sysplex Workload Balancing
  - Enables transport pooling (connection concentration)
    - Separates the logical connection from the application to the driver and the physical connection from the driver to Db2 for z/OS
    - Global pool of Transport objects per JVM
    - Initial connection goes thru Sysplex Distributor, then drive relies on its WLM server list to route to the member directly
    - Connection reuses/changes transport at transaction boundary (commit)



1. Provides Automatic Client Reroute (ACR)
   - Masks connectivity issue with Db2 including Db2 failure/shutdown
   - See next slides....

High Availability – Sysplex Distributor



Initial connection using Vy,448 to DB2A, DB2B, or DB2C

Result Set and SRVLST returned ((V2:W2, V3:W3)

Vx = IP address of Group

W1, W2, W3 = relative weights of respective Db2 servers

1. First connection goes through Sysplex Distributor (DVIPA) connecting to the group DVIPA
2. Sysplex Distributor routes based on DISTMETHOD
3. Result set (and if Sysplex WLB is enabled) a list of available Db2 subsystems and their respective weightings are returned to the driver
4. Based on the respective weighting subsequent units of work are routed to the data sharing members (if Sysplex WLB is enabled) otherwise Sysplex Distributor decides

Dynamic Location Aliasing

- Similar to subgroup attach you can use a dynamic location alias for member specific access or to point connection requests to a subset of the data sharing members accepting DDF work

  -MODIFY DDF ALIAS(SUBGROUP1) ADD
  - Alias1 is created and is stopped by default

  -MODIFY DDF ALIAS(SUBGROUP1) PORT(448)
  - Alias1 is associated with port 448

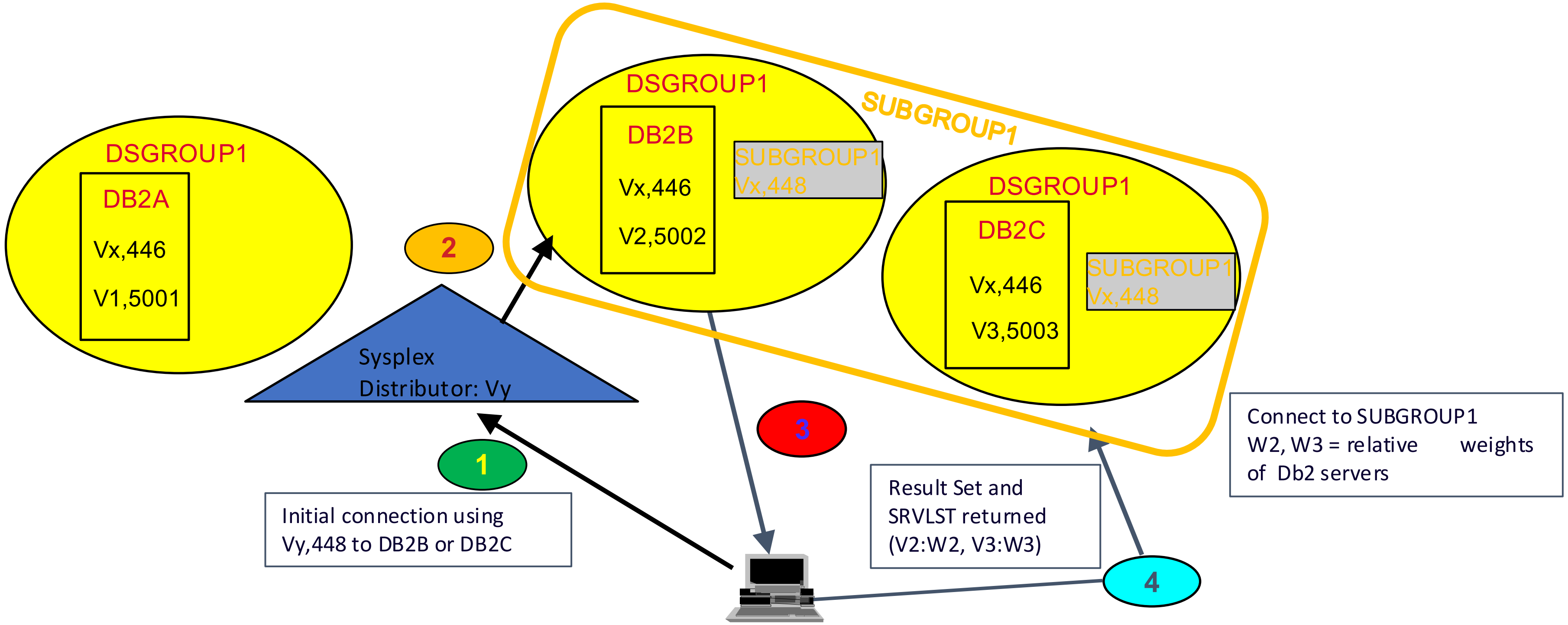
- Start command must be issued each time Db2 member is restarted (can be issued prior to DDF being started)

- Stopping the alias only stops work coming from applications using that alias without the need to stop the DIST address space

  -MODIFY DDF ALIAS(SUBGROUP1) STOP
  - Behaves similar to a quiesce where currently executing work completes

  – DISPLAY DDF DETAIL (DSNL080I) of alias will show all associated DBATs, connections, as well as weighted order of other Db2 members with the same alias
  - DSNL096I ADBAT= 100 CONQUED= 1000 TCONS= 1000
  - DSNL101I WT IPADDR

High Availability – Sysplex Distributor and Subgroup



Initial connection using Vy,448 to DB2B or DB2C

Result Set and SRVLST returned (V2:W2, V3:W3)

Connect to SUBGROUP1 W2, W3 = relative weights of Db2 servers

1 First connection goes through Sysplex Distributor (DVIPA) connecting to a subset of the group

2 Sysplex Distributor routes based on DISTMETHOD

3 Result set (and if Sysplex WLB is enabled) a list of available Db2 subsystems in that subset and their respective weightings are returned to the driver

4 Based on the respective weighting subsequent units of work are routed to the data sharing members (if Sysplex WLB is enabled) otherwise Sysplex Distributor decides

High Availability – Sysplex WLB …

- With ACR, application connectivity is *NOT* lost if DB2A is brought down or crashes
  - The intent is to route around connection failures and mask them from the applications
    - This also avoids stale connection exceptions in app server connection pools
  - If connectivity to DB2A is lost i.e., taken down for maintenance or crashes
    - If the first SQL statement of a transaction encounters this situation:

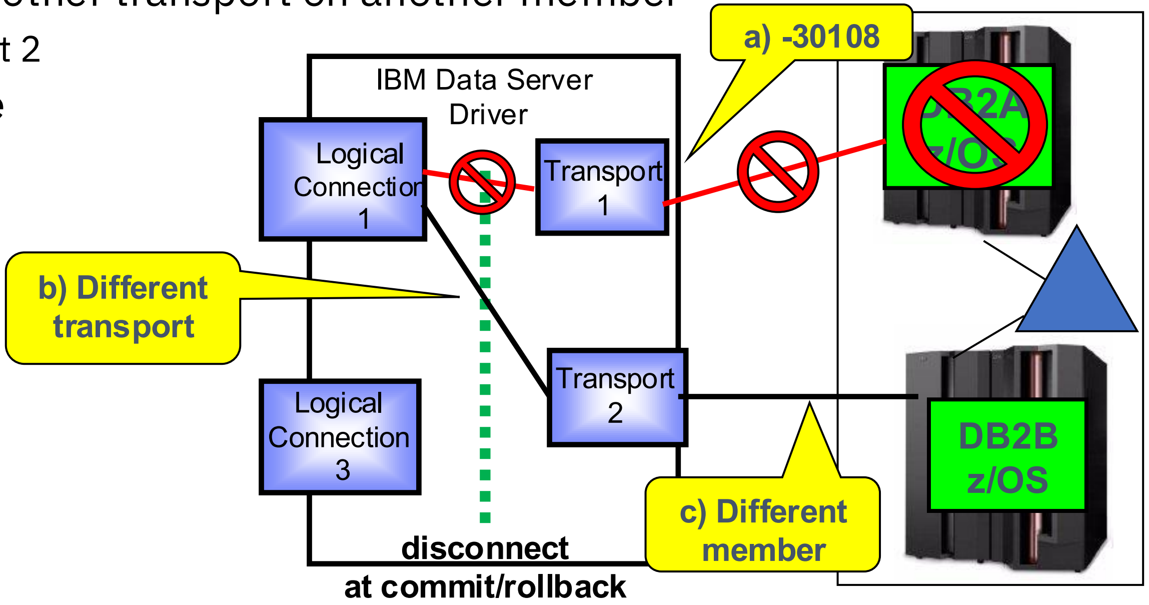
a) Driver receives network failure (-30108)

- That transport would then be recycled to avoid being used by other requests

b) Driver seamlessly routes transaction to another transport on another member

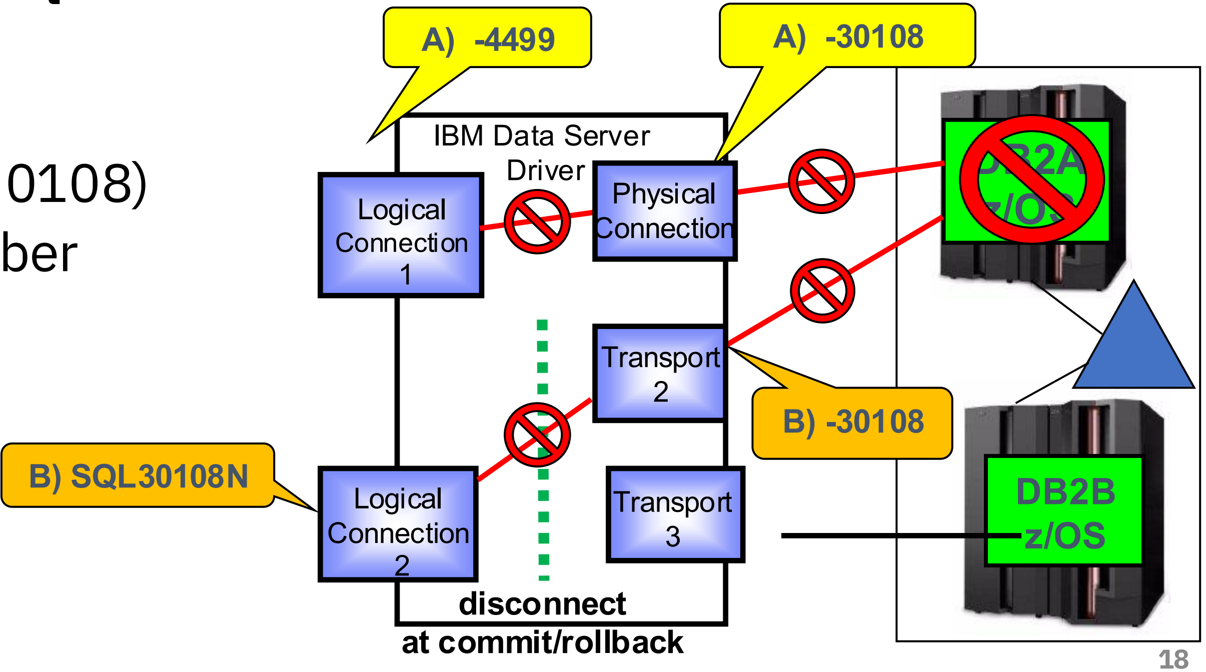
- Logical connection 1 moves to transport 2

c) Application receives no negative SQL code and continues processing



High Availability – Sysplex WLB …

- Without ACR, application connectivity is lost if DB2A is brought down or crashes (A)
  - Without Sysplex WLB, physical connection gets a stale connection exception
    - Driver receives network failure (-30108)
    - Application receives a stale connection exception
      - SQLCODE -4499 (JAVA) or SQL30108N (non-JAVA)
- **With** ACR, if the failure occurs **after** the first SQL statement of a transaction (B)
  - Transaction rolled back in Db2
  - Driver (transport) receives network failure (-30108)
  - Connection is re-established to another member
    - Set statements replayed on connection
  - Application receives SQL code (SQL30108N)
    - Must re-drive SQL statement itself



High Availability – Sysplex WLB …

- Provides intelligent workload distribution at transaction level
  - Every 10 seconds driver polls WLM to refresh the weighted list of data sharing members
  - Driver does the routing, Sysplex Distributor only provides the first connection to the group
  - Enabling Sysplex WLB means the driver should distribute the work to the appropriate Db2 based on (IWMSRSRS WLM interface) 3 main factors
    - Displaceable capacity
      - Sysplex WLB favors LPARs with more CPs/zIIPs over smaller LPARs
        - Also favors single member per LPAR because system weight is divided by # of servers on it
      - After the initial connection using Sysplex distributor the driver takes over
        - Changing the Sysplex Distributor DISTMETHOD will not have any effect on DDF workload distribution
      - **Recommendation**:
        - Try to define and maintain a symmetric configuration

High Availability – Sysplex WLB …

- Enabling Sysplex WLB means the driver should distribute the work to the appropriate Db2 based on (IWMSRSRS WLM interface) 3 main factors **…**
  - Performance index of enclaves as well as queue time
    - If threads delayed on LPAR 1 miss their WLM goal, they may ‘slosh’ to a Db2 on another LPAR
    - If the WLM goal is too loose the driver will not re-route work in a timely manner and DBAT requests will queue, possibly encroaching on CONDBAT
    - **Recommendation**:
      - WLM goals must be realistic and ACHIEVABLE i.e., may mean moving from Execution Velocity to Response time goal
      - Monitor -DIS THREAD(*) DETAIL DSNV482I during peak times as well as RMF 72-3 Workload activity reports to determine effectiveness of the WLM policy and adjust if necessary

High performance DBATs

- High-Performance DBATs have the potential to provide a significant opportunity for CPU reduction by
  - Avoiding connection going inactive and switching back to active later
  - Avoid DBAT being pooled at transaction end i.e., going back into DDF thread pool for reuse by probably a different connection
  - Supporting true RELEASE(DEALLOCATE) execution for static SQL packages to avoid repeated package and statement block allocation/deallocation
- Very powerful performance option, but can be dangerous … if not used intelligently
- High Performance DBAT behavior
  - Must be using CMTSTAT=INACTIVE so that DBATs can be pooled and reused
  - Db2 MODIFY DDF PKGREL(BNDOPT|BINDPOOL) command must also be in effect
  - When a DBAT is about to be pooled, if there is at least one package bound with RELEASE(DEALLOCATE) allocated against the thread, then the DBAT will stay active with connection
    - All it takes is one package to be bound with RELEASE(DEALLOCATE) - it could be a driver package (SYS######), stored procedure package, UDF/trigger package - one is enough to create a High-Performance DBAT!!!

High performance DBATs …

- High-Performance DBAT behavior …
  - Connections will turn inactive after 200 times in V12, 500 times in V13 (no user control) to free up DBAT
    - BNDOPT = DBAT is immediately terminated
    - BNDPOOL = DBAT is returned to the pool for reuse (see APAR PI31597)
  - HP-DBAT will not go into the DBAT pool when the thread completes
  - Number of DBATs will increase, potentially substantially
    - ZPARM MAXDBAT will probably need to increase
  - Sysplex WLB will continue to provides intelligent workload distribution at transaction level
  - Identify application workloads with long-lived connections from a combination of accounting trace summary and Db2 statistic trace data
    - (COMMIT RECEIVED+ABORTS RECEIVED)/INTIATED FROM REMOTE SITE > 200|500
  - ZPARM POOLINAC controls the duration of inactivity i.e., a High-Performance DBAT that is idling will be killed off after a period of time (default is 120 seconds)
  - BIND/REBIND, SQL DDL and online REORG cannot break in High-Performance DBAT
    - Must issue command -MODIFY DDF PKGREL (COMMIT) to overlay BNDOPT|BNDPOOL|RQSTWLB|DFLTWLB option
      - Switch to PKGREL(COMMIT) will occur gradually
      - Once completely switched, allows BIND/REBIND, SQL DDL and online REORG to break in

High performance DBATs …

- Important operational considerations
  - Do not use RELEASE(DEALLOCATE) on common packages that are widely shared across distributed workloads as it will considerably drive up the requirement for MAXDBAT
    - Risk of not having enough DBATs available for workloads not using High-Performance DBATs
    - Worst case running out of DBATs completely or escalating thread management costs
  - Check that all the ODBC/JDBC driver packages in collection NULLID are bound as RELEASE(COMMIT) – otherwise they must be rebound with RELEASE(COMMIT)
    - WARNING: RELEASE(DEALLOCATE) is the default BIND option for Db2 client driver packages
  - As a starting point all packages such as external stored procedures and native SQL procedures should be bound with RELEASE(COMMIT)
    - Be very careful about DDF workloads re-using common static SQL packages used by CICS/IMS and/or batch workloads bound with RELEASE(DEALLOCATE)
  - Do not over-inflate the application server connection pool definitions, otherwise it will considerably drive up the demand for High-Performance DBATs
  - Be prepared to use -MODIFY DDF PKGREL(COMMIT) to switch off High-Performance DBATs at first signs of DBAT congestion i.e., overuse of DBATs

High performance DBATs ...

- High Performance DBATs should be used <u>selectively</u> ...

1. BIND all the driver packages with RELEASE(DEALLOCATE) into an alternate package collection e.g., NULLID2
2. For the candidate applications, use the alternate package collection e.g., NULLID2
   - ✓ For JDBC applications, set the currentPackageSet property in the respective datasource
   - ✓ For .NET and ODBC / CLI applications, set CurrentPackageSet parameter in the db2dsdriver.cfg configuration
   - ✓ Exploit Db2 System Monitor Profiles to route transactions to the alternative collection e.g., NULLID2
4. To activate High-Performance DBATs, issue command -MODIFY DDF PKGREL(BNDOPT|BNDPOOL)
   - ✓ BNDOPT = purge DBAT
   - ✓ BNDPOOL = return DBAT to pool for reuse (see APAR PI31597)
5. Monitor and adjust MAXDBAT as the demand for DBATs is likely to increase
6. To disable, issue command -MODIFY DDF PKGREL (COMMIT) to overlay BNDOPT|BNDPOOL option
   - ✓ Switch to PKGREL(COMMIT) will occur gradually over several minutes

Db2 System Monitor Profile table support

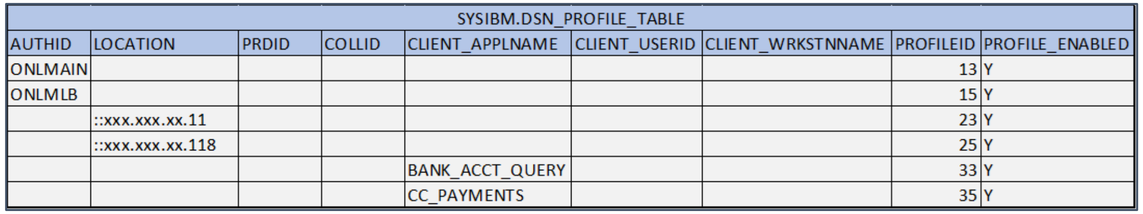
- System profiling monitoring
  - Introduced as long ago as Db2 9 and is continually enhanced in each release and in the service stream since from customer feedback and usage
  - Provides a central control point to set various ZPARMs at a more granular level than what is set at the subsystem level
    - Use Cases
      - Reserve connection and threads for critical applications
        - Prioritize and assign resources to critical workload
        - Place limits/throttles on less important workload
      - Protect the system from any unexpected rogue access
      - Denial-of-service attack characteristics
      - Ability to supply appropriate settings to support a variety of applications
      - Manage migration to a new driver
    - Resources to manage
      - Connections
      - Threads
      - Special Registers
      - Db2 13 support of local threads
    - Modes
      - Warnings
      - Exceptions

Db2 System Monitor Profile table support …

- System profiling monitoring …
  - Resources to manage
    - Connections
      - MONITOR CONNECTIONS – CONDBAT
      - Total number of remote connections from a TCP/IP requestor, includes active and inactive connections
        - ** If set at 100 then the 101st connection trips the exception
      - Filtering criteria - LOCATION value, limited to IP Address or Domain Name
    - Threads
      - MONITOR THREADS – more granular MAXDBAT
      - ATTRIBUTE2 - Total number of concurrent active remote threads that use TCP/IP on the Db2 subsystem
      - ATTRIBUTE3 – Maximum number of queued threads prior to exception triggering
        - If blank, queued threads = 2 x ATTRIBUTE2
      - Examples
        - ATTRIBUTE2=100, ATTRIBUTE3 (blank)
          - 100 active threads, 100 queued threads, exception fires on 201st thread
    - Filtering criteria: LOCATION, or PRDID, or ROLE and/or AUTHID, or COLLID and/or PKGNAME, or One of (CLIENT_APPLNAME, CLIENT_USERID, or CLIENT_WORKSTNNAME)

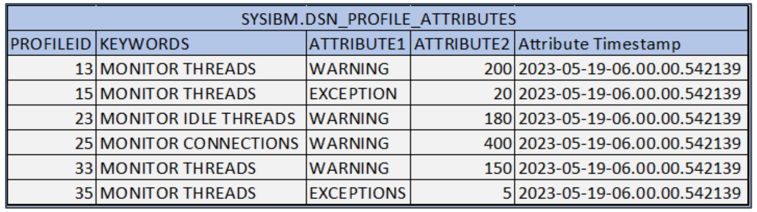
Db2 System Monitor Profile table support …

- Profile Tables
  - SYSIBM.DSN_PROFILE
    - Defines the profile and filtering combinations



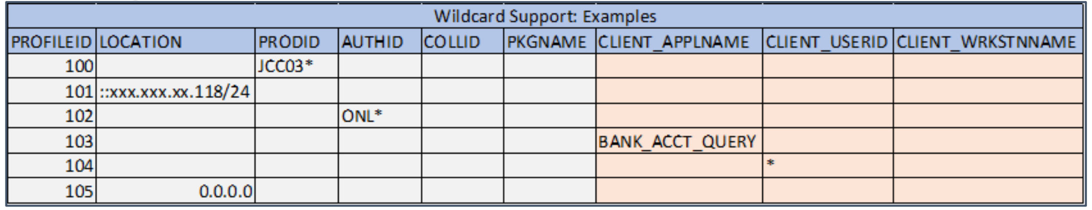
SYSIBM.DSN_PROFILE_TABLE								
AUTHID	LOCATION	PRDID	COLLID	CLIENT_APPLNAME	CLIENT_USERID	CLIENT_WRKSTNNAME	PROFILEID	PROFILE_ENABLED
ONLMAIN							13	Y
ONLMLB							15	Y
	::xxx.xxx.xx.11						23	Y
	::xxx.xxx.xx.118						25	Y
				BANK_ACCT_QUERY			33	Y
				CC_PAYMENTS			35	Y

  - SYSIBM.DSN_ATTRIBUTES
    - Defines actions and settings



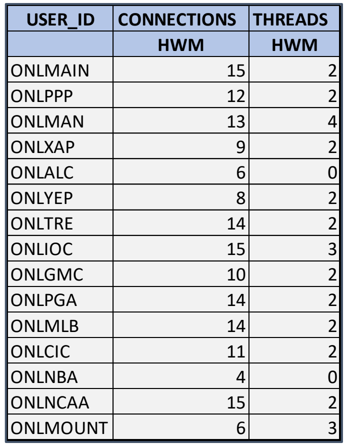
SYSIBM.DSN_PROFILE_ATTRIBUTES				
PROFILEID	KEYWORDS	ATTRIBUTE1	ATTRIBUTE2	Attribute Timestamp
13	MONITOR THREADS	WARNING	200	2023-05-19-06.00.00.542139
15	MONITOR THREADS	EXCEPTION	20	2023-05-19-06.00.00.542139
23	MONITOR IDLE THREADS	WARNING	180	2023-05-19-06.00.00.542139
25	MONITOR CONNECTIONS	WARNING	400	2023-05-19-06.00.00.542139
33	MONITOR THREADS	WARNING	150	2023-05-19-06.00.00.542139
35	MONITOR THREADS	EXCEPTIONS	5	2023-05-19-06.00.00.542139

  - Wildcarding examples

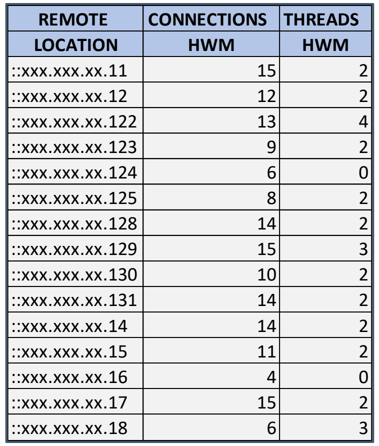


Wildcard Support: Examples								
PROFILEID	LOCATION	PRODID	AUTHID	COLLID	PKGNAME	CLIENT_APPLNAME	CLIENT_USERID	CLIENT_WRKSTNNAME
100		JCC03*						
101	::xxx.xxx.xx.118/24							
102			ONL*					
103						BANK_ACCT_QUERY		
104							*	
105	0.0.0.0							

  - How do I determine the values
    - Intellectual knowledge
    - IFCID 365, 411, 412
    - Use Db2 System Monitoring Profiles in warning mode



USER_ID	CONNECTIONS HWM	THREADS HWM
ONLMAIN	15	2
ONLPPP	12	2
ONLMAN	13	4
ONLXAP	9	2
ONLALC	6	0
ONLYEP	8	2
ONLTRE	14	2
ONLIOC	15	3
ONLGMC	10	2
ONLPGA	14	2
ONLMLB	14	2
ONLCIC	11	2
ONLNBA	4	0
ONLNCAA	15	2
ONLMOUNT	6	3



REMOTE LOCATION	CONNECTIONS HWM	THREADS HWM
::xxx.xxx.xx.11	15	2
::xxx.xxx.xx.12	12	2
::xxx.xxx.xx.122	13	4
::xxx.xxx.xx.123	9	2
::xxx.xxx.xx.124	6	0
::xxx.xxx.xx.125	8	2
::xxx.xxx.xx.128	14	2
::xxx.xxx.xx.129	15	3
::xxx.xxx.xx.130	10	2
::xxx.xxx.xx.131	14	2
::xxx.xxx.xx.14	14	2
::xxx.xxx.xx.15	11	2
::xxx.xxx.xx.16	4	0
::xxx.xxx.xx.17	15	2
::xxx.xxx.xx.18	6	3

- Order of precedence in evaluation of profiles
  - https://www.ibm.com/docs/en/db2-for-zos/13?topic=mcdbupt-how-db2-applies-multiple-matching-profiles-threads-connections

Prepared Statement Objects - Benefits

- Prepared Statement objects vs. Statement objects
  - 2 DB calls needed for fetch (describe, data)
  - Prepared Statement makes description calls at construction time
  - Statement makes them on every execution
  - Prepared Statement enables Statement Pooling
  - Use Statement when SQL is not executed often

- Prepared Statement object pool
  - Client-side optimization
  - Pool of Prepared Statement and Callable Statement objects, not active in a Connection
  - Reduced overhead of Java object creation and garbage collection
  - Pool exists for the life of an open connection, effectiveness depends on connection pooling
  - No impact to application

- **No concern for SQLJ**

Statement or Prepared Statement

- Statement example

```
Statement stmt = con.createStatement();
stmt.executeUpdate("INSERT INTO EMPLOYEE VALUES( 'John', 123 )");
stmt.executeUpdate("INSERT INTO EMPLOYEE VALUES( 'Mary', 425 )");
```

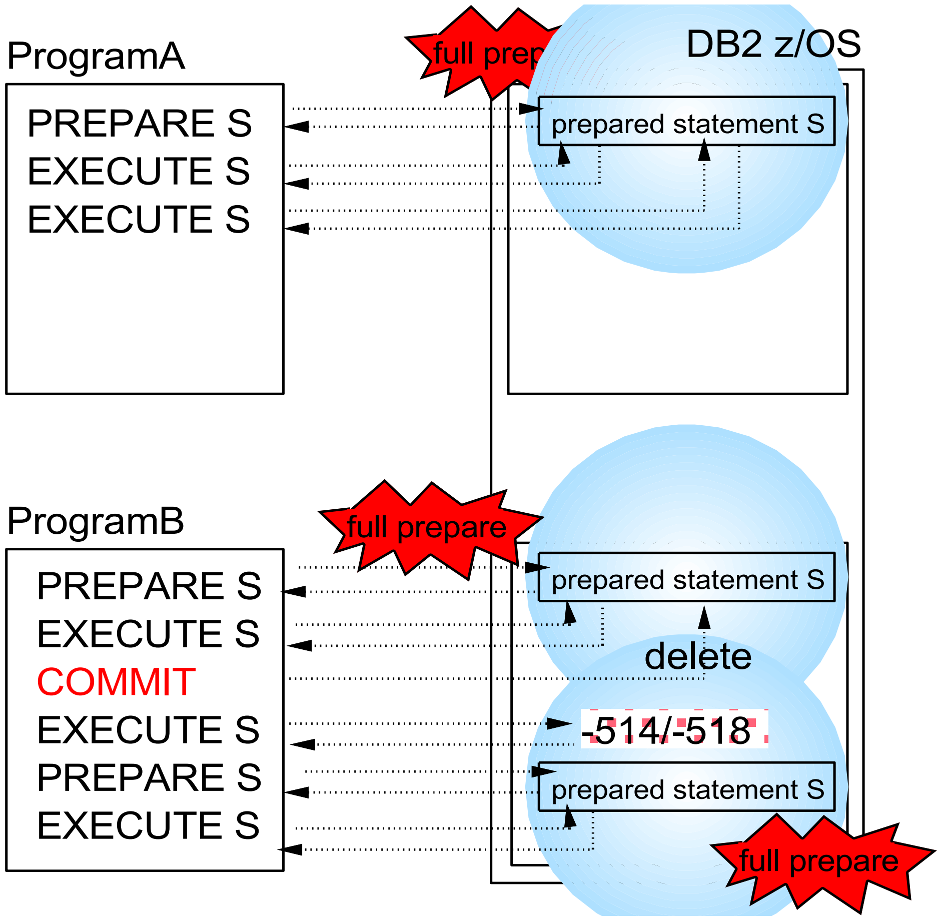
- Prepared Statement example

```
PreparedStatement ps = con.prepareStatement(“INSERT INTO EMPLOYEE VALUES(?,
?)”);
ps.setString(1,”John”);
ps.setInt(2, 123);
ps.executeUpdate();
ps.setString(1,”Mary”);
ps.setInt(2, 425);
ps.executeUpdate();
```

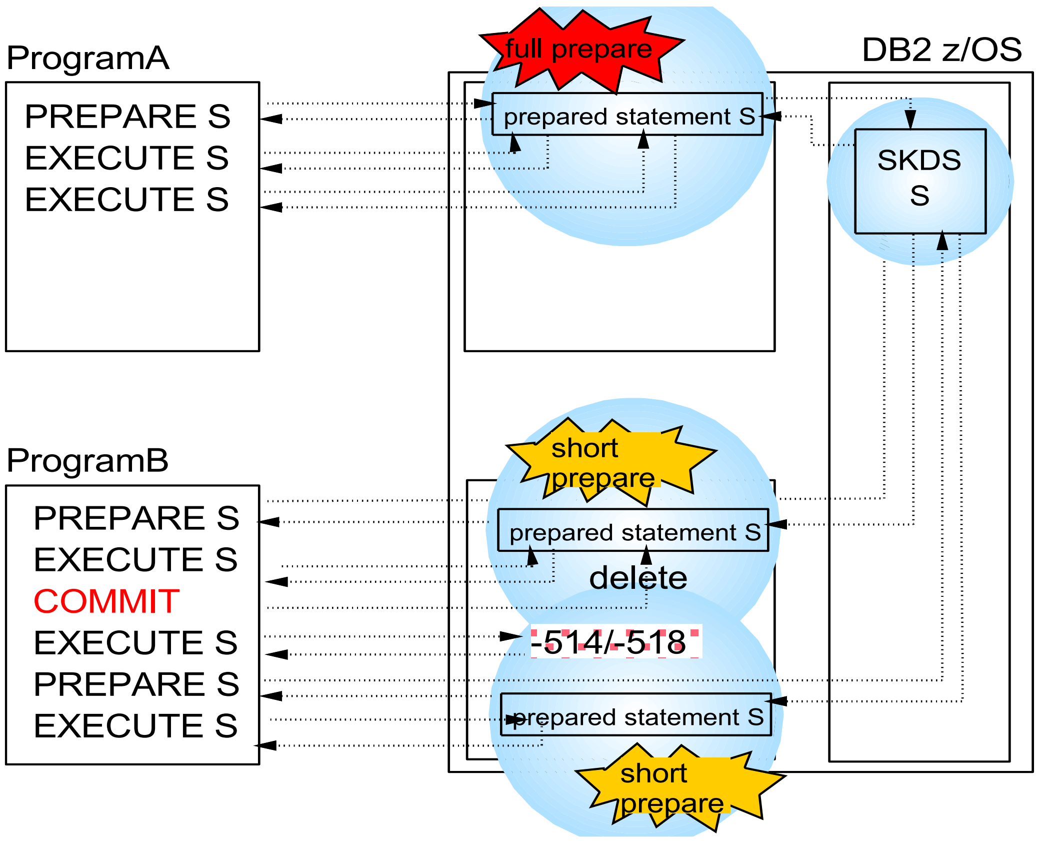
Db2 Dynamic Statement Cache

- Dynamic statement is prepared at run time
- Dynamic Statement Cache (DSC)
  - Improves performance of dynamic SQL
  - Enabled by ZPARM CACHEDYN = YES
  - Allows applications (multiple threads) to reuse and share a prepared statement
- Prepared statement is saved in an in-memory cache
- Subsequent prepares of same statement loads from cache if cache match criteria is met
  - SQL, AUTHID, special registers, bind options etc.
- Cache pool shared by different threads, plans and packages ("global cache")
- Good cache hit rate produces significant performance benefits
- Full Prepare can consume 10-100x more CPU than a Short Prepare!

Db2 Dynamic Statement Caching ...



NO CACHING

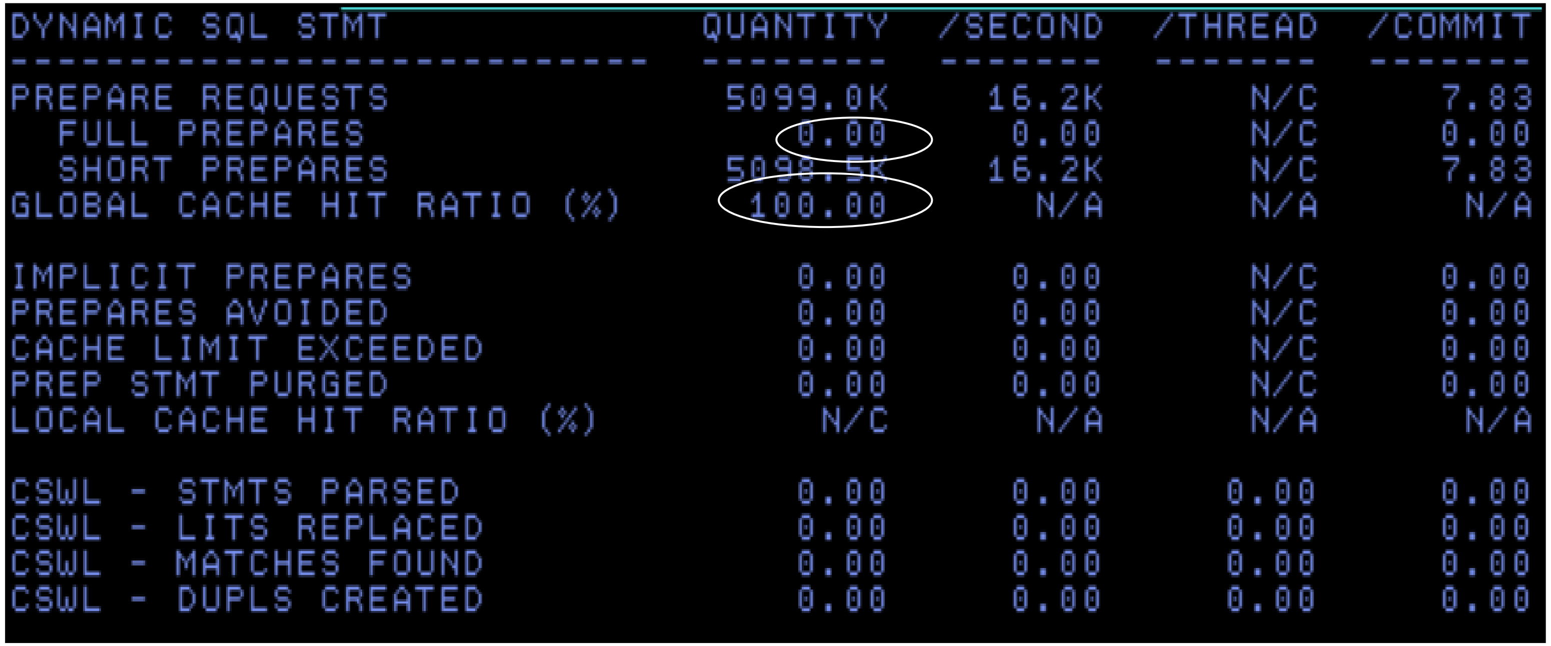


CACHEDYN=YES

Dynamic Statement Cache

- SQL can be EXPLAINed using the ‘EXPLAIN STMTCACHE’ feature
  - Populates various explain tables with details on statements in the dynamic statement cache including access path information

- Use Dynamic SQL Stmt section of statistics to monitor the Global Cache Hit Ratio % to determine if the cache size needs to be increased.



```
DYNAMIC SQL STMT             QUANTITY  /SECOND  /THREAD  /COMMIT
-------------------------    --------  -------  -------  -------
PREPARE REQUESTS              5099.0K    16.2K      N/C     7.83
  FULL PREPARES                  0.00     0.00      N/C     0.00
  SHORT PREPARES              5098.5K    16.2K      N/C     7.83
GLOBAL CACHE HIT RATIO (%)     100.00      N/A      N/A      N/A

IMPLICIT PREPARES                0.00     0.00      N/C     0.00
PREPARES AVOIDED                 0.00     0.00      N/C     0.00
CACHE LIMIT EXCEEDED             0.00     0.00      N/C     0.00
PREP STMT PURGED                 0.00     0.00      N/C     0.00
LOCAL CACHE HIT RATIO (%)         N/C      N/A      N/A      N/A

CSWL - STMTS PARSED              0.00     0.00     0.00     0.00
CSWL - LITS REPLACED             0.00     0.00     0.00     0.00
CSWL - MATCHES FOUND             0.00     0.00     0.00     0.00
CSWL - DUPLS CREATED             0.00     0.00     0.00     0.00
```

Literal Replacement for Global Dynamic Statement Cache

- Dynamic SQL with literals can be re-used in the cache
  - Literals replaced with &
    (similar to parameter markers but not the same)
- To enable set the property enableLiteralReplacement='YES' in the JCC Driver
- Lookup Sequence
  - Original SQL with literals is looked up in the cache
  - If not found, literals are replaced and new SQL is looked up in the cache
    - Additional match on literal usability
    - Can only match with SQL stored with same attribute, not parameter marker
  - If not found, new SQL is prepared and stored in the cache
- **Db2 12 support as BIND option CONCENTRATESTMT on package**

Why SQLJ?

- Static SQL performance for Java applications
- Static SQL authorization model
- Monitoring/Manageability
  - Static SQL packages for accounting/monitoring
  - Static SQL locks in access path, so that access path changes do not occur without a conscious choice
- Measurements with the IRWW workload comparing JDBC vs SQLJ with the T2 driver

	Throughput (ETR)	**Normalized Throughput (ITR)**	**z/OS CPU Utilization**	**CL.1 CPU time**
JDBC T2	2636.83	3773.37	69.88	0.000672
SQLJ T2	2694.80 (+2.20%)	5174.35 (+37.13%)	52.08 (-25.47%)	0.000457 (-32.00%)

Java Performance Problem Areas

- Java Application
  - Autocommit(on) - default
  - Mismatch of Java and Db2 data types
  - Usage of String for numbers
  - Retrieval of unused columns (select *)
  - Transaction isolation REPEATABLE READ (default in WAS) or SERIALIZABLE
  - Open cursor SELECT … FOR UPDATE for locking semantics
    - Consider using WITH RS USE AND KEEP UPDATE LOCKS
- JDBC
  - JDBC resources not closed (cursor, statements, connections)
  - No usage of Parameter Markers
    - E.g. select c1, c2 FROM t1 WHERE c3=?
-> use literal replacement option
  - Cursor are defined as hold by default
  - Usage of Statement() instead of preparedStatement() objects
    - No object caching in WAS

Summary

- Business critical Java applications with Db2 for z/OS as enterprise database server have been implemented commonly and successfully for many years now
- Going through installation checklist is highly recommended prior to each implementation to ensure success
  - Communication among WAS Administrator, Db2 System Programmer, and Application Architect
- No shortcuts in respect to setup for availability
  - User sees application availability and not Db2 system availability
- Monitor and react proactively and do not wait until user complains
  - Workload behavior changes over time

IDUG

2026

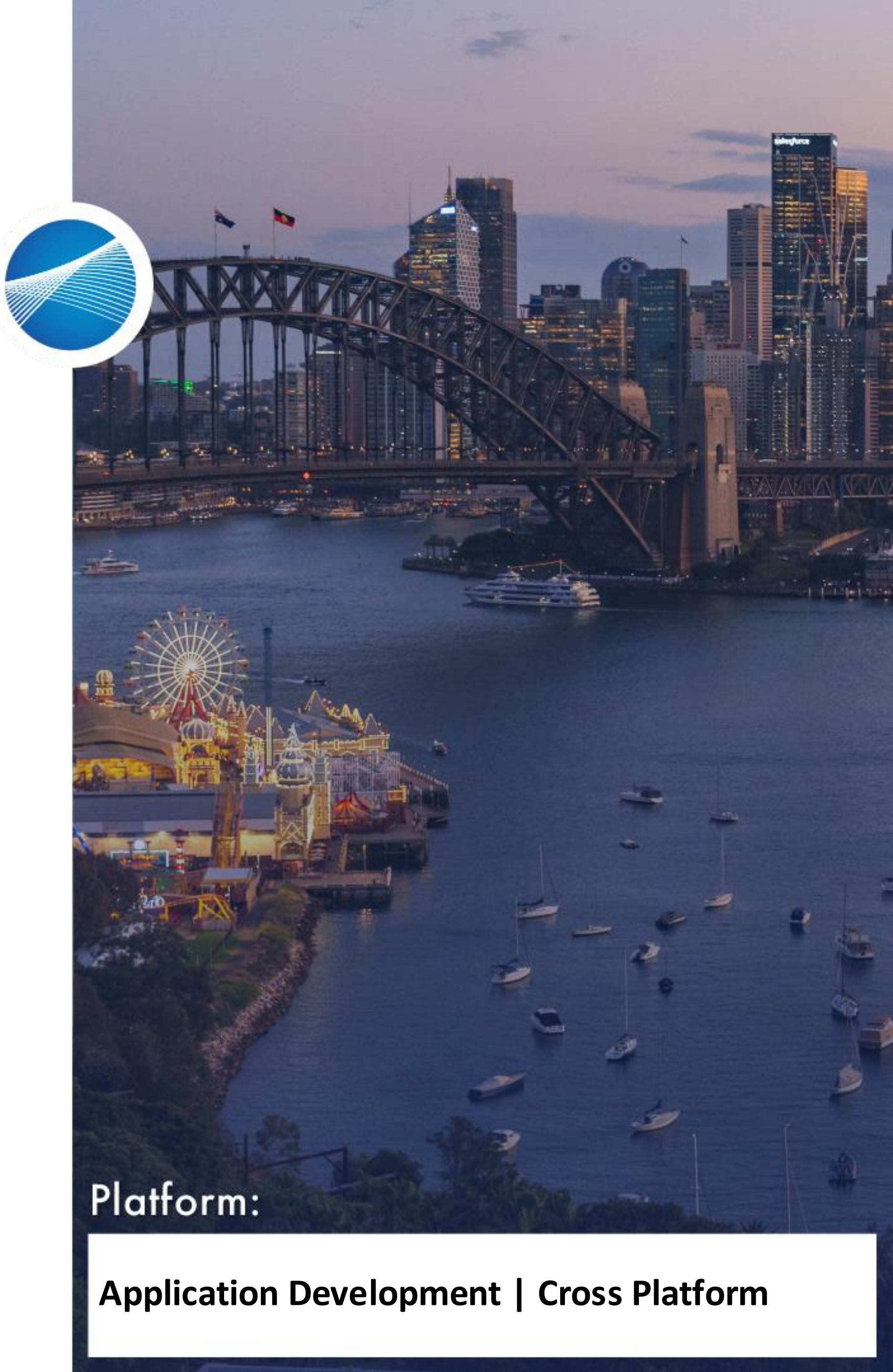
Sydney | March 16 - 18

AU Db2 TECH CONFERENCE

Low Risk Mainframe Application Modernization Using Java Accessing Db2 for z/OS

Anthony Ciabattoni, ACiabattoni@ibm.com

Session Code: C04



Platform:

Application Development | Cross Platform

Questions

Thank You

IDUG

2026 Australia **Db2** Tech Conference