

Modernizing the Architecture of Your DB2 for z/OS-Accessing Applications

Robert Catterall
IBM

Session Code: A16

May 4, 2017 10:30 - 11:30 AM | Platform: DB2 for z/OS

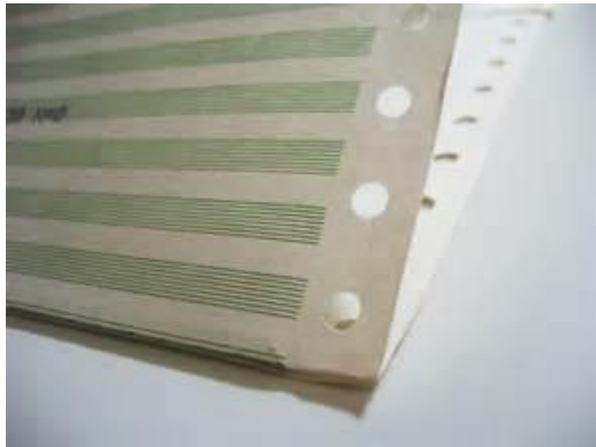
Agenda

- New imperatives driving architectural change
- Important concept: the “invisible” DB2 for z/OS system
- Enhancing system resiliency and flexibility with IBM MQ
- Embracing modern programming languages
- Leveraging application-enabling features of DB2 for z/OS

New imperatives driving architectural change

DB2 for z/OS* was introduced over 30 years ago...

* originally, DB2 for MVS



...and a lot has changed since then

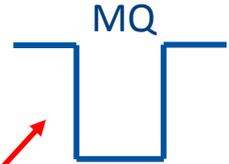
1) The balance of power has shifted (in a good way)

- Then: the “glass house” people (i.e., “systems” people) were on top of the IT heap
 - These folks often looked down on application developers
- Now: it’s all about the apps (and the people who develop them)
 - Why that is as it should be: the value of IT systems (and the value of the data housed in those systems) is largely dependent on the value of the applications that use those systems (and access that data)
 - Applications bring in revenue
 - Applications serve customers
 - Applications schedule deliveries, determine product mix, pay suppliers, manage inventory, etc., etc.
- Application architecture has to support application development
 - Agility, ease of use, flexibility – all are of primary importance from a development perspective



2) It's a heterogeneous IT world

"Green screen"



RESTful services



Batch



Perl, Python, PHP, Ruby, ...



Java, JDBC



.NET, ODBC

The choice: support these interfaces, or become increasingly irrelevant

3) Availability is more important than ever

- These days, when applications go down....
 - ...assembly lines shut down
 - ...trucks don't leave the loading dock
 - ...orders can't be placed
 - ...reservations can't be taken
 - ...organizations are liable for financial losses
- And, perhaps most importantly, customers say, “bye-bye” – maybe permanently

*Application architecture has to support high –
sometimes ultra-high – availability requirements*



At least one thing hasn't changed

- Cost efficiency is still a really big deal
 - IT spend has to be justified by value delivered
 - Application architecture has to support “right-placement” of functional components to help deliver cost-effective computing
 - Data, business logic, user interface
 - Across various platforms
 - Across various locations – on-prem, off-prem, cloud (public, private, hybrid)



30 years later, money still
doesn't grow on trees...

Important concept: the “invisible”
DB2 for z/OS system

What is the key indicator of application architecture success?

(from a DB2 for z/OS perspective)

- New DB2 for z/OS-accessing applications (not just growth of existing DB2 workload)
- If you want modern application developers to be favorably inclined towards writing code that accesses DB2 for z/OS, don't make them do anything different from what they do in writing code to access other data-serving platforms
 - In other words, make the particulars of z/OS and DB2 invisible to developers



"What mainframe?
I only see data."



Why is mainframe transparency important?

- Two reasons:
 1. It removes a reason why developers might be opposed to writing DB2 for z/OS-accessing code
 2. When a DB2 for z/OS system looks the same (to a developer) as other data-serving platforms, what becomes more noticeable is that it doesn't act the same as other platforms
 - Talking about quality of service here
 - The DB2 for z/OS system is the one that is always up, is never hacked, and performs well no matter how much work is thrown at it
 - Robert Goodman, a longtime DB2 for z/OS DBA and a frequent speaker at conferences, put it well: **“Just call it the super-server”**



“Um, could our application's data go on the super-server?”

Abstraction: DB2's cloak of invisibility

- One very important abstraction mechanism: non-DBMS-specific data access interfaces, such as:
 - JDBC (Java database connectivity)
 - ODBC (open database connectivity)
 - ADO.NET
- All are provided via the IBM Data Server Driver (which you should be using) or DB2 Connect, for remote applications accessing DB2 through TCP/IP connections
- IBM also provides JDBC and ODBC drivers for local-to-DB2 applications (i.e., applications running in the same z/OS LPAR as DB2)



Taking abstraction to the next level: REST access to DB2

- REST is short for **RE**presentational **S**tate **T**ransfer – a lightweight protocol that uses HTTP verbs (e.g., GET, PUT) to invoke services (these services, in turn, are called “RESTful services”)
- Why the interest in accessing DB2 data via REST calls?
 - REST is very popular with developers of cloud-based and mobile applications, and lots of data in DB2 for z/OS databases is being accessed, and could be accessed, by these applications
 - But it’s not just about cloud and mobile apps – many developers who work for organizations that use DB2 for z/OS want a higher level of abstraction than is offered by JDBC and ODBC
 - The issue: when using JDBC or ODBC, you know that the target data server is a relational database management system, and lots of developers see that as “plumbing” – something of which they should not have to be aware
 - What they want: data as a service (DaaS)



REST access to DB2 for z/OS

- DB2 12 delivered a native REST interface, and that capability was retrofitted to DB2 11 via the fix for APAR PI66828
- What it does: enables exposure of a single SQL statement (which could be a call to a stored procedure) as a REST API (Application Program Interface)
 - Input and output data is sent in JSON format (JavaScript Object Notation) – a very easy to use, easy to interpret data format →
- How it can be used: either with a direct call from a client, or through IBM's z/OS Connect Enterprise Edition
 - The native REST interface is the basis of DB2's support for z/OS Connect EE

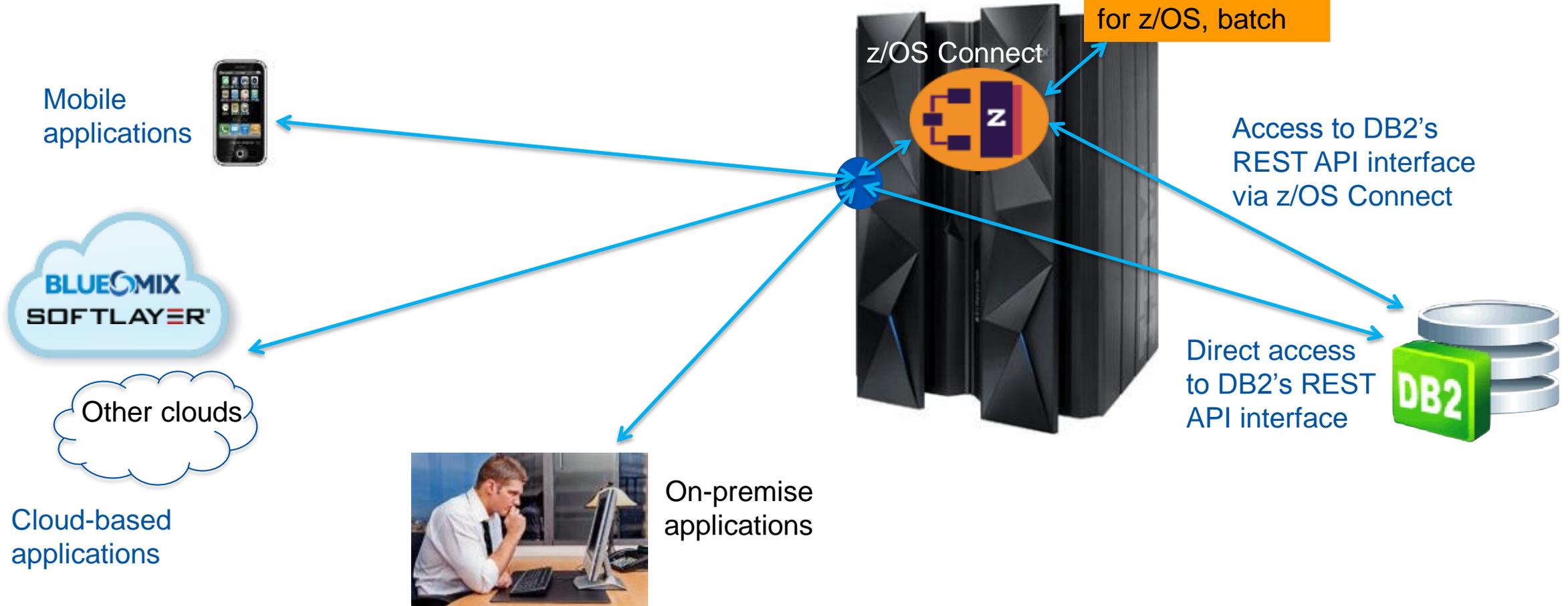
```
'{"Customer":{"@cid": 777,  
              "name": "George",  
              "age": 29,  
              "telephone": "566-898-1111",  
              "country": "USA"  
            }}'
```

DB2 data as a service (DaaS)

Accessing DB2 data with RESTful APIs and JSON

z/OS Connect also makes other z/OS-based data processing services available through REST APIs

CICS, IMS, WAS
for z/OS, batch



Spark on z/OS: abstraction for analytics

- Apache Spark is an open-source framework for data analytics
- Spark can run natively under z/OS, because it is Java byte code
- Spark can natively access Hadoop-managed data, and any data from a source that has a JDBC interface (including DB2 and IMS)
 - IBM's [z/OS Platform for Apache Spark](#) offering provides connectors that enable Spark access to VSAM and other z/OS-based data sources that do not have a JDBC interface

R is a language that is very popular with data scientists, and there is an R interface to Spark, so data from a wide variety of sources can be ingested by, and analyzed through, Spark – with users not having to know anything about z/OS



One more thing about mainframe “invisibility”



“That application team insists on row-level locking for their tables!”

- Don’t be so reluctant to use row-level locking
- I’m calling this out because it’s something I’ve been hearing for 20 years
- Keep this in mind: with other relational DBMSs that use locks to protect data integrity (including DB2 for LUW), row-level locking is the default
- Keep this in mind, too: in a non-data sharing environment, row-level locking can be just about as CPU-efficient as page-level locking
 - In a data sharing environment, row-level locking adds some CPU overhead (because it drives increased page P-lock activity), but should be manageable if used selectively
 - In any case, use it where you need it; otherwise, use page-level locking
- Key point: row-level locking is a DB2 feature – using it does not mean that a DB2 for z/OS DBA is “selling out” or “surrendering”
 - Remind yourself: “It’s all about enabling new DB2 for z/OS workloads”

Enhancing system resiliency and flexibility with IBM MQ

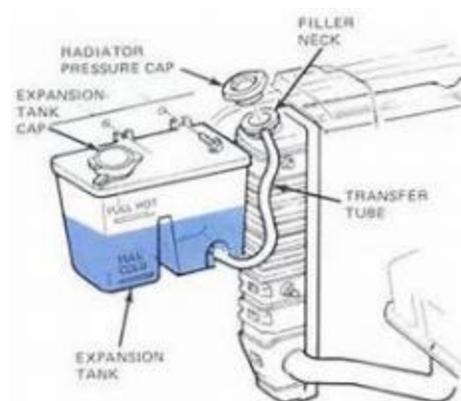
MQ as a means of boosting application availability

- Consider this scenario: an application user inputs some data that will lead to a DB2 for z/OS data change, and clicks “Submit”
 - If the DB2 database operation will be synchronous with respect to the user clicking “Submit,” and a to-be-updated table is unavailable for some reason, one of two things is likely to happen:
 - **Bad:** the user stares for a while at his locked screen, and gets frustrated
 - **Worse:** the user’s transaction times out, and he gets really frustrated
- Now consider another scenario: when the aforementioned user clicks “Submit,” his information is captured in a message on an MQ queue
 - User gets a quick “transaction complete” indicator (performance benefit from user’s perspective)
 - Application availability is enhanced: if a to-be-updated DB2 table is unavailable, messages simply accumulate on the queue, and are processed when the table is available again



MQ as a means of managing IT costs

- Suppose a large number of DB2 data-changing transactions enter the system over a short period of time
- If those transactions are processed synchronously, and if good response time has to be maintained, the DB2 server has to have the capacity for that peak load
- Alternatively, if the transactions land first on an MQ queue, and the DB2 server does not have the capacity to keep up with the inflow, the queue depth just builds up for a while
- In that way, MQ acts conceptually like an automobile's coolant expansion tank – it enables the system to “flex”



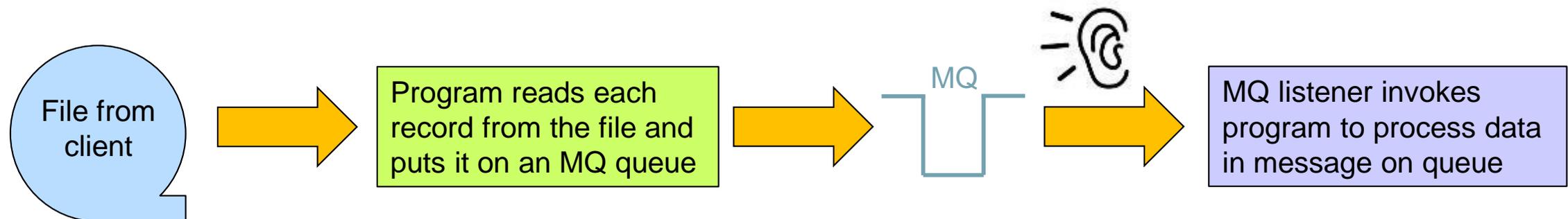
But how do MQ messages become DB2 updates?

- Typically, by way of an MQ listener
 - There are CICS and IMS MQ listeners (a message arriving on a queue drives execution of a CICS or IMS transaction – data in the MQ message is input to the transaction)
 - There is also a DB2 MQ listener – it ships with DB2 and enables you to associate an MQ queue with a DB2 stored procedure
 - When message arrives on a queue, the associated stored procedure is automatically invoked – the message is passed as input
 - You can set up several queues for different message types, and each queue can be associated with a particular stored procedure



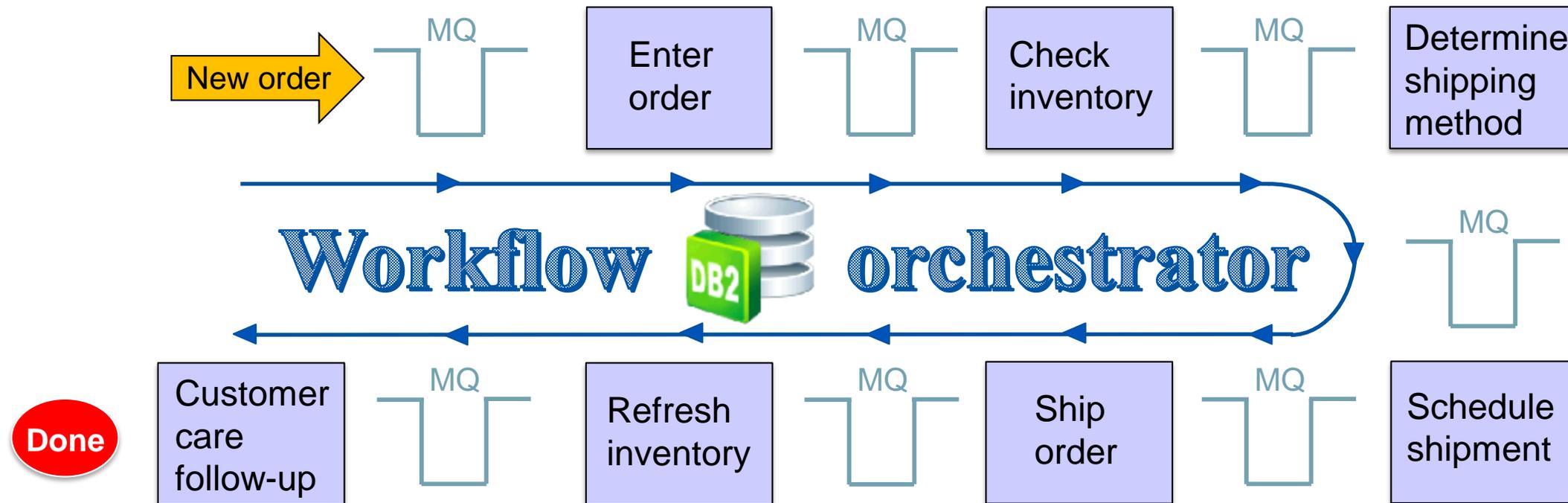
More advanced MQ: batch transactionalization

- If clients now send you files of records that drive database update jobs, would it be beneficial for clients to be able to send – and you to process – a record at a time?
 - That can be done with MQ – queues can be securely exposed as Web services
 - Your system then functions more like a refinery – continuous flow – versus a “job shop”
- A way to move in this direction: take the input files you receive today from clients, and “pre-process” them with a step that takes individual records and puts them on an MQ queue – subsequently picked up by an MQ listener (see preceding slide)
 - The transaction kicked off by the MQ listener processes the information in the record as the existing file-ingesting batch job would



More advanced MQ: workflow orchestration

- With workflow orchestration software (available from IBM and other vendors), MQ can be the foundation of an intelligent system that ensures that incoming data items (e.g., orders) are acted on by the right processes, in the right order



Embracing modern programming languages

Languages for network-attached DB2 programs

- Some people may think your choices here are pretty limited: Java, C, .NET C#
- In fact, the IBM Data Server Driver provide drivers that enable access to DB2 for z/OS from a variety of other, modern languages
 - PHP } Drivers available in the Data Server Driver Package
 - Ruby }
 - Python } Drivers can be downloaded and built using application
 - PERL } header files available in the Data Server Driver Package
- So, when you hear that one of these languages is desired for an application, don't think that DB2 for z/OS can't be the data server for that application
- Note: entitlement to use the Data Server Driver is through your DB2 Connect license
- And remember, via DB2's new REST interface, a program that can issue a REST call – whatever its language – can access DB2 for z/OS-based data

Java for z/OS-based applications

- Don't presume that z/OS-based applications have to be written in COBOL
- Java is a VERY viable choice for such applications
 - As CPU- and memory-efficient as COBOL? No, but Java in z/OS uses less-costly zIIP MIPS, and z Systems memory is much cheaper than it used to be (and z/OS LPARs often have lots of it)
 - z Systems and z/OS can deliver outstanding performance for Java programs
 - 1200% performance improvement from Java 5 on a z9 mainframe to Java 7 on an EC12 (even better performance with Java 8 on a z13)
- One execution environment: WebSphere Application Server (WAS) for z/OS
 - Paired with DB2, provides a complete Java application infrastructure in one z/OS system
 - Your choice of type 2 and type 4 JDBC drivers (with latter, go into TCP/IP stack, use DDF)
 - Like DB2, can use 1 MB page frames (for Java heap)
- But WAS isn't your only Java option in a z/OS system...



Other options for Java in z/OS

- Java DB2 stored procedures
 - Here, DB2 11 delivered an important enhancement: one 64-bit, multi-threaded JVM per Java stored procedure address space (vs. a 31-bit JVM per stored procedure before)
 - Can run more Java stored procedures concurrently in an address space – more-efficient resource usage
 - One reason some organizations use Java DB2 stored procedures: sometimes a team of Java developers wants to be able to write all of an application's code, including data-layer code
- Another option: the IBM JZOS tool kit
 - A set of classes, distributed with IBM Java SDKs for z/OS, that can be used to launch Java applications as batch jobs (or started tasks)
 - As with WAS for z/OS, can use type 2 or type 4 JDBC driver
 - More information:

<http://www-03.ibm.com/systems/z/os/zos/tools/java/products/jzos/overview.html>

SQL Procedure Language – important, and getting more so

- Introduced for stored procedures (“native SQL procedures”) with DB2 9 for z/OS, and enhanced since then:
 - DB2 10: user-defined functions (UDFs) can be written in SQL PL
 - DB2 11: Autonomous stored procedures (separate unit of work from caller) and support for arrays as input and output parameters (for Java and .NET callers)
 - DB2 12: SQL PL in body of trigger
- SQL PL is based on a category of SQL statements called **control statements** (referring to logic flow control)
 - Examples are IF, ITERATE, GOTO, WHILE, LOOP, REPEAT, SIGNAL
- Key difference between SQL PL routines and other DB2-accessing programs: a SQL PL routine’s only executable is its package – there’s nothing outside of DB2

Reasons organizations use SQL PL routines

- Cost savings: a SQL routine runs under the invoker's task, so if the invoker is a DRDA requester then execution of the routine will be up to 60% zIIP-offload-able
- Throughput improvement: again because a SQL PL routine runs under invoker's task – no need to switch DB2 thread from invoker's task to routine's task
 - This can be particularly beneficial for a UDF referenced in a correlated subquery
- Functionality exclusives (aforementioned autonomous transactions, array support)
- Could align well with the concept of “SQL center of excellence” in IT organization
 - And, a team of SQL experts who can write SQL PL routines can do so for DB2 for z/OS and LUW



SQL PL

Leveraging application-enabling features of DB2 for z/OS

Major recent DB2 advances – do your developers know?

- Every release of DB2 for z/OS delivers multiple features pertaining to application enablement – should developers have to keep up with that on their own?
- Consider DB2's XML data management functionality – there is a good chance that your organization is storing XML data somewhere
 - In DB2 for z/OS? If not, why not?
 - In DB2, but not in an XML column (i.e., in a VARCHAR or CLOB column)?
 - Either way, is it because developers don't know about DB2's XML capabilities?
 - Schema validation
 - Document integrity checking via CHECK DATA utility
 - Ability to return data from XML documents in relational form (XMLTABLE function)
 - High-volume XML data update via multi-versioning
 - Full XQuery support (with DB2 11)
 - and more...



People won't know these things unless someone tells them!

How about temporal data support?



- Does your organization have any applications for which it would be useful to know what data looked like at a time in the past (system time)?
- How about applications that would benefit from being able to put future data updates (e.g., price changes) into the database, without affecting programs accessing “currently in effect” data (business time)?
- These are capabilities DB2 provides – they don’t have to be implemented with application code
- And remember, with DB2 11’s temporal special registers, existing programs can have a “time travel” view of data (forward or backward) *without having to be modified*

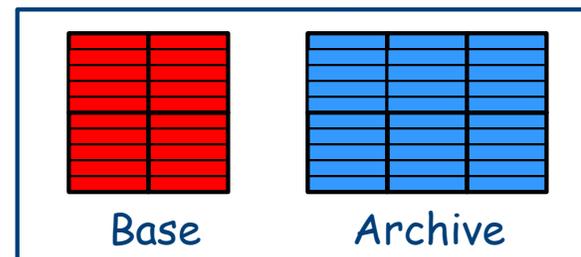
How about DB2-managed archiving (DB2 11 feature)?

- Do you have any situations in which it would be beneficial to concentrate newer (and more frequently accessed) rows in one table, with “older and colder” rows going to a separate table, while allowing programs to see the two tables as one?
- You may already have “base” and “archive” tables (organizations have been doing this for years), but with the burden of dealing with this mode of data management placed on developers – let DB2 do it!

Before DB2-managed
data archiving



After DB2-managed
data archiving



Programs see one table

- Newer, more “popular” rows
- Older rows, less frequently retrieved

And more...

- Are you leveraging DB2's LOB capabilities (LOB in-lining, efficient and easy LOB load and retrieval, and – with DB2 12 – LOB compression with zEDC)?
- Have you used DB2's result set data grouping options (GROUPING SETS, ROLLUP, CUBE) and OLAP specifications (RANK, DENSE_RANK, ROW_NUMBER, moving aggregates) to simplify reporting and analytics?
- Have you considered DB2's ability to store data in JSON format, or to retrieve data from an IBM BigInsights Hadoop-managed data store in relational form?
- **If you'd like some help** in telling your developers about recent DB2 enhancements related to application enablement, contact me or one of my IBM DB2 specialist colleagues – we can do it!



A closing thought

- Your key partners in modernizing the architecture of your DB2 for z/OS-accessing applications are your organization's developers and application architects
 - **They** want architecture that enables them to be productive, agile, and responsive to the needs of the business
 - **You** can help them to achieve those objectives for DB2-accessing applications, but you may need to be the one who *initiates the partnership*





IDUG DB2 North American Tech Conference
Anaheim, California | April 30 - May 4, 2017



Robert Catterall

IBM

rfcatter@us.ibm.com

Modernizing the Architecture of Your DB2 for z/OS-Accessing Applications

*Please fill out your session
evaluation before leaving!*