A Whole New World of Db2 13 for z/OS

Anna McKee
IBM Worldwide Systems Center (WSC)
anna.mckee@ibm.com

Mark Rader
IBM Worldwide Systems Center (WSC)
mrader@us.ibm.com

Agenda

- Profile support for local threads
- More active and archive logs
- Work file specification FOR SORT and FOR DGTT
- Rebind phase-in for native SQL routines and trigger packages
- Generation of Universally Unique Identifiers (UUIDs)
- Distributed tracing and OpenTelemetry (OTel) support

Profile support for local threads

Increased profile support for local applications

FL 508

Previous behavior

- Profile tables only supported setting special registers and global variables for distributed applications
- Db2 13 GA-level release added support for *local threads*:
  - CURRENT LOCK TIMEOUT special register
  - DEADLOCK_RESOLUTION_PRIORITY global variable
    - Schema: SYSIBMADM

New behavior

- Additional *local* thread support with profile tables:
  - 23 existing special registers
  - 6 existing global variables
- *ATTRIBUTE2* column value has new options:
  - *1* (local threads only) or *2* (local and remote threads)
  - Previously *ATTRIBUTE2* had to be *NULL*
    - Remote threads only

Previous behavior

DSN_PROFILE_TABLE

PROFILEID	LOCATION	ROLE	AUTHID	PRDID	COLLID	PKGNAME
101	null	null	USER01	null	null	null
102	null	null	null	null	null	DSNTEP4

DSN_PROFILE_ATTRIBUTES

PROFILEID	KEYWORDS	ATTRIBUTE1	ATTRIBUTE2
101	SPECIAL_REGISTER	SET CURRENT DEGREE = ‘ANY‘	NULL
102	GLOBAL_VARIABLE	SET SYSIBMADM.MAX_LOCKS_PER_TABLESPACE = 500	NULL

- ATTRIBUTE 2: Previously, the required ATTRIBUTE2 value of NULL indicates that the special register or global variable is for remote threads only

Use case example

FL 508

DSN_PROFILE_TABLE

PROFILEID	LOCATION	ROLE	AUTHID	PRDID	COLLID	PKGNAME
101	null	null	USER01	null	null	null
102	null	null	null	null	null	DSNTEP4

DSN_PROFILE_ATTRIBUTES

PROFILEID	KEYWORDS	ATTRIBUTE1	ATTRIBUTE2
101	SPECIAL_REGISTER	SET CURRENT DEGREE = ‘ANY‘	2
102	GLOBAL_VARIABLE	SET SYSIBMADM.MAX_LOCKS_PER_TABLESPACE = 500	1

- USER01 matches PROFILEID 101; USER01 *local AND remote threads* will use CURRENT DEGREE = ‘ANY’
- PROFILEID 102 sets SYSIBMADM.MAX_LOCKS_PER_TABLESPACE to 500 for jobs run under DSNTEP4 for *local threads only*
- ATTRIBUTE 2: setting the column value to 1 applies for local threads only; value of 2 applies for local and remote threads

Increased number of active and archive log data sets

Increased number of active and archive log data sets

FL 508

Challenges

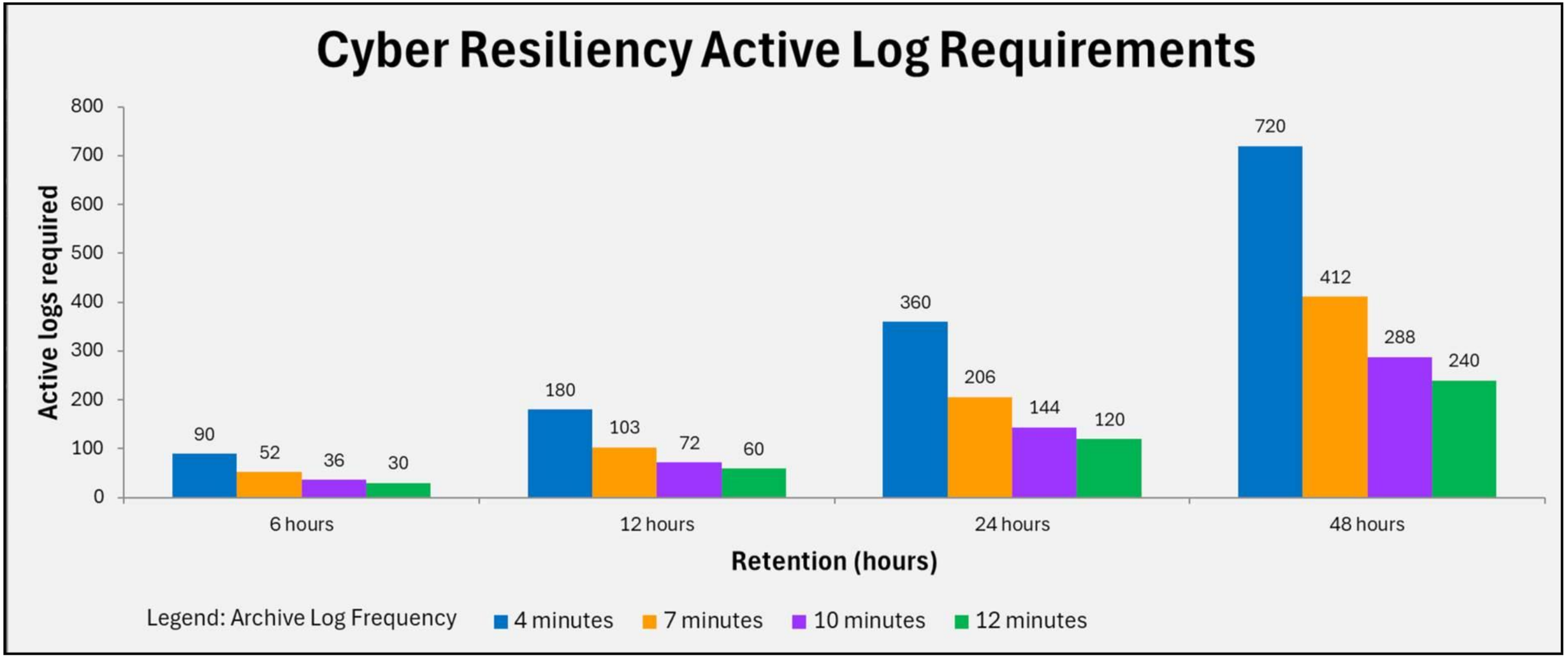
- Db2 13 limit (< FL508):
  - 93 active logs per copy
  - 768 GB maximum active log size
  - 10,000 maximum archive logs per copy
- Cyber Vault and Safeguarded Copy (SGC) with low recovery point objective (RPO) requires more than 93 active log pairs
- Increasing logging rates and desire to keep 24-48 hours of active log data requires more than 93 active log pairs
- Some customers have active log data sets larger than desired due to limit of 93 pairs

New behavior

- Increase possible number of active log pairs to 1,000
- Increase maximum archive logs to 14,500 per log copy

Active log data sets required for SafeGuarded copy (SGC)

FL 508



Increased number of active and archive log data sets

FL 508

Conversion of BSDS (optional*)

- DSNJCNVU to increase formatted BSDS records
  - DSNJU004 displays if DSNJCNVU has been run
  - Db2 must be stopped to convert BSDS
- BSDS should have SMF data class attributes as follows:
  - Extended addressability (EA)
  - Extended format (EF)
  - Extent constraint relief
  - CA reclaim [VSAM Control Area]

Activation of V13R1M508

- If BSDS contains enough formatted records and FL 508 or higher
  - Optional: add more active log pairs
    - Use DSNJU003 to add up to 1,000 active logs
    - -SET LOG NEWLOG to add up to 1,000 active logs
    - Optional but recommended: Run DSNJLOGF to preformat active logs
    - If FL = V13R1M507*, can also add up to 1,000 active logs
- DSNJU004 displays the max value available in the Db2 environment

Increased number of active and archive log data sets

FL 508

Changes to support increased number of data sets

- DSNJ384I -DIS LOG DETAIL reformatted
  - ‘STOPPED=…’ moved to line 2
- Install/migrate CLIST updated several panels to support up to 1,000 active log data sets
- z/OSMF workflows updated to support up to 1,000 active logs
- Log manager data set ID changed
- IFCID 104 changed

Performance

- Db2 for z/OS opens all active logs during start of Db2 [–START DB2]
  - 2,000 log copies may increase start time
- Small active log data sets with frequent archiving will impact write performance as well as log read performance
- Great number of active log pairs should improve recovery performance
  - Avoid archive log reads

Workfiles for SORT/DGTT

Specifying work file purpose with DDL

Challenge

- Work file database handles:
  - Working space for sort work, internal temp space, scrollable cursors, …
  - Storage for created global temporary tables (CGTT) and declared global temporary tables (DGTT)
- Db2 categorizes table spaces into:
  - Used for DGTTs and scrollable cursors
  - Used for sort and other work
- Support for segmented (non-UTS)* and UTS PBG
  - No clear SQL to identify purpose and ensure desired space

Inputs

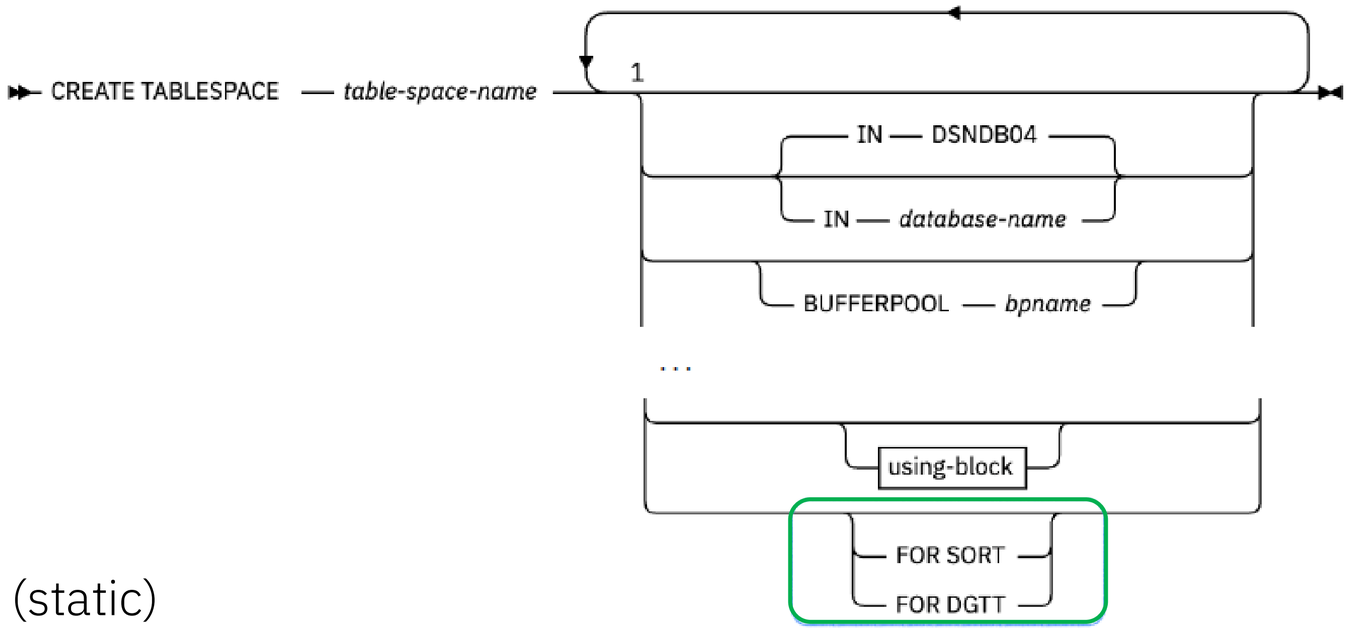
- SECQTY of table space
- Type of table space (UTS or not)
- WFDBSEP (DSNZPARM) NO | YES

Specifying work file purpose with DDL

FL 508

New behavior

- CREATE TABLESPACE option for UTS PBG work file table spaces
  - FOR DGTT
  - FOR SORT
  - If neither specified, default is DGTT
- -DIS DATABASE updated
  - DSNT397I “TYPE”
    - WS: FOR SORT ; WD: FOR DGTT
- Existing SQL codes updated to include FOR SORT and FOR DGTT
  - -4700: APPLCOMPAT <= V13R1M507 (static)
  - -4743: APPLCOMPAT <= V13R1M507 (dynamic)
  - -620: FOR SORT or FOR DGTT specified but not work file database



Rebind phase-in for native SQL routines and trigger packages

Rebind phase-in for native SQL routines and trigger packages

FL 508

Challenges

- Db2 12 FL 505 included rebind phase-in
  - Rebind new copy of package while package in use
  - Phase-in new package copy
  - Existing threads continue with package copy in use
  - New threads use new copy
  - After existing threads deallocate, old package copy is deleted
- Rebind phase-in for new package attributes or to switch to prior access path
- Rebind phase-in did not support native SQL routines or trigger packages in V12R1M505

New behavior

- Rebind phase-in support extended to native SQL routines and trigger packages in V13R1M508
  - Same behavior as packages in V12R1M505
- Rebind native SQL routines or trigger packages for new attributes or to switch to prior access path even if the packages are in use
- Function level-dependent only
  - ACTIVATE V13R1M508 starts the new behavior
  - <= V13R1M507* stops the behavior
  - No APPLCOMPAT dependency

Generation of Universally Unique Identifiers (UUIDs)

Generation of Universally Unique Identifiers (UUIDs)

FL 508

Traditional identifiers

- May only be unique within a single system, but not universally unique
  - Unsuitable for distributed environments
- Often easy to guess or predict
  - Ineffective as a defense against malicious referencing
  - Unsuitable as public IDs

UUIDs

- Industry standard and already used in other databases or systems to generate unique identifiers or keys
- A 128-bit label used to uniquely identify rows in a Db2 table. Example:
  - AA97B177-9383-4934-8543-0F91A7A02836
  - The UUID algorithm (version 4) used in Db2 provides very high randomness, ensuring a high degree of uniqueness
- UUIDs are not 100% duplicate-free
  - The probability that a UUID will be duplicated is close to 0, but not always 0
  - The probability to find a duplicate within 103 trillion version 4 UUIDs is one in a billion

Generation of Universally Unique Identifiers (UUIDs)

FL 508

Db2 introduces the following built-in functions:

- `GENERATE_UUID()` – returns the formatted string representation of a UUID by using the version 4 algorithm as a CHAR(36). Format: 'XXXXXXXX-XXXX-4XXX-XXXX-XXXXXXXXXXXX'
- `GENERATE_UUID_BINARY()` – returns the binary string representation of a UUID by using the version 4 algorithm as a BINARY(16). Format: BX'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
- `VARBINARY_FORMAT (<UUID>, 'xxxxxxxx-xxxx-4xxx-xxxx-xxxxxxxxxxxx')` – returns a binary string representation of a character string that has been formatted using a *format-string*
  - Result: 'XXXXXXXX-XXXX-4XXX-XXXX-XXXXXXXXXXXX'
- `VARCHAR_FORMAT_BINARY(<UUID>, 'xxxxxxxx-xxxx-4xxx-xxxx-xxxxxxxxxxxx')` – returns a character string representation of a bit string that has been formatted using a fo*rmat-string*
  - Result: BX'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'

Example (EMPID is the unique employee identifier, a.k.a. employee key):

- `INSERT INTO EMP (EMPID, EMPNO, NAME) VALUES (GENERATE_UUID(), '0020', 'Max');`

Generation of Universally Unique Identifiers (UUIDs)

FL 508

- The GENERATE_UUID() and GENERATE_UUID_BINARY() functions require the true random number generation (TRNG) provided in Central Processor Assist for Cryptographic Functions (CPACF) on z14 and later

Recommendations:

- UUID by design is only very close to unique (not guaranteed unique)
  - Within a table enforce absolute uniqueness by creating a unique index on the column containing the UUID
  - In case of SQL code -803 (duplicate), the process must redo the insert to generate another UUID value
- To save storage and memory space (table, index, bufferpool, etc.) – especially in case of a huge number of expected table rows – users may prefer to use GENERATE_UUID_BINARY(), which generates a BINARY(16) key
  - Users can convert a binary UUID back to the readable CHAR(36) format using the new scalar function VARCHAR_FORMAT_BINARY()

Distributed tracing and OpenTelemetry support

Distributed tracing and OpenTelemetry support

PH67971 & PH68073

OpenTelemetry is a vendor-agnostic observability framework which consists of:

- A collection of APIs, SDKs, and tools, which can be used to instrument, generate, collect, and export telemetry data to help site reliability engineers (SRE) analyze their software's performance and behavior
- Semantic conventions that define a standard naming scheme for common telemetry data types
- A specification for all components

Understanding OTel Span records enables you to:

- Determine where a problem is in a distributed application
- Identify which team needs to be called in to resolve it
- Reduce the time to identify "in which area" when a problem arises in a complex distributed application

Distributed tracing and OpenTelemetry support

PH67971 & PH68073

Only for inbound workloads with a traceparent from:

- Db2 for z/OS native RESTful services
- JDBC Type-4/Db2 JCC type-4 Client driver [1]
- CICS-Db2 Attachment Facility
- CICS Liberty-JDBC type-2 (RRS AF) [2]
- IMS-Db2 Attachment (ESAF)
- IMS-Db2 Java Adapter-JDBC type-2 (RRS AF) [2]
- z/OS Connect-Db2 Rest service (HTTP header)

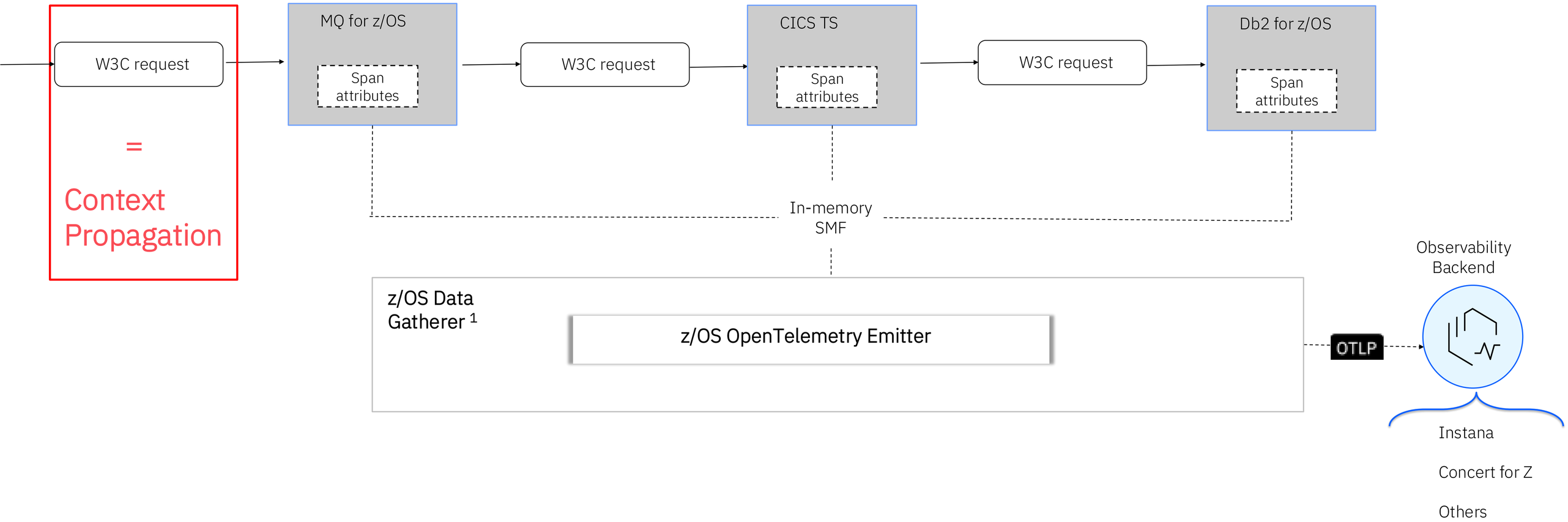
[1] JDBC Type-4 support GA with Db2 Connect 12.1.3 or a special build for Db2 Connect 12.1.2.
[2] Attachment controlled by CICS or IMS and not JDBC type-2.

Db2 will emit an OTel span record with basic performance metrics (CPU time and elapsed time) for each unit of work

- Environments must have IFCID 3 on to report “in Db2” timings
- Environments must be configured to enable the new SMF record 1161 subtype 1

Distributed tracing and OpenTelemetry support

PH67971 & PH68073



[1] APAR OA66345 (UJ98068 for z/OS 3.1 and UJ98067 for z/OS 3.2)

Questions?

Summary

- Profiles for local threads
  - Special registers and global variables changed without changing application
- More active (and archive) logs
  - Flexibility for high volume logging, Safe Guarded Copy, Case information
- Workfile specification FOR SORT and FOR DGTT
  - Ease of managing work file allocations
- Rebind phase-in for native SQL routines and trigger packages
  - Less application disruption
- Generation of Universally Unique Identifiers (UUIDs)
  - Increased application portability and interoperability with distributed applications
- Distributed tracing and OpenTelemetry support
  - Greater observability of performance behavior with end-to-end monitoring tools

FL 508 is available! https://www.ibm.com/docs/en/db2-for-zos/13.0.0?topic=levels-function-level-508

Experience more with IBM

Visit us at the IBM Booth #113

After a full day of technical sessions, take a break with us!

Connect with our experts, snap a photo with the z17 Plexi or the latest Telum II, and get an up-close look at our Spyre Accelerator.

Come back each day for fresh topics and demos at our expert stations.

Think 2026

Join 5000+ senior business and technology leaders who are seizing the AI revolution to unlock unprecedented growth and productivity at **Think 2026**.

Find out more information using the QR code below.

IBM Digital Asset Haven

IBM Digital Asset Haven is the operational backbone for financial institutions and regulated enterprises entering the digital asset economy.

Find out more information using the QR code below.

Your feedback is important!

SHARE
EDUCATE ▸ NETWORK ▸ INFLUENCE

Submit a session evaluation for each session you attend:

www.share.org/evaluation