



DB2 for z/OS Distributed Data Facility Questions – and Answers

TRIDUG

*Robert Catterall, IBM
December 9, 2015*



Agenda

- DDF monitoring and tuning
- DDF application architecture
- DDF workload management and control
- DDF and data security



DDF monitoring and tuning





Q: Why is CPU time so high for the DDF address space?

- The question is prompted by information like this, from a DB2 monitor statistics long report (this was for a 2-hour interval):

CPU TIMES	TCB TIME	PREEMPT SRB	NONPREEMPT SRB	PREEMPT IIP SRB
SYSTEM SVCS	12.646	0.000	2:03.883	N/A
DB SVCS	5:33.773	0.004	50:05.190	0.394
IRLM	0.003	0.000	21.245	N/A
DDF	1:02.659	3:13:21.699	2:10.283	2:55:43.179

More than 6 hours of DDF CPU time

- People are used to seeing relatively small CPU times for DB2 address spaces – what gives here?
 - Is something wrong?
 - Is DDF a pig?





DDF CPU time – it's an SQL thing

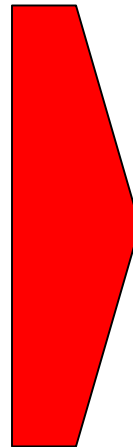
- Typically, the vast majority of DDF's CPU time simply reflects *the cost of executing SQL statements* that come through DDF
 - Referring to DB2 monitor statistics long report snippet on the preceding slide, the CPU times in the two “preemptible SRB” columns of the DDF row indicate the cost of executing DDF-related SQL statements
 - SQL statements from DRDA requesters run under DDF preemptible SRBs
 - Figures in the two preemptible SRB columns for DDF show the general-purpose and zIIP engine CPU time consumed in executing DDF-related SQL statements
 - In this example, 6 hours and 9 minutes (about 99%) of DDF CPU time is SQL statement execution time – figures in “TCB time” and “non-preemptible SRB time” columns show CPU consumed by DDF “system” tasks
 - *Coming through DDF does not make SQL statements more expensive* – same SQL statements from a CICS region would have increased CPU consumption of that address space by about the same amount



Q: Why am I not seeing the DDF zIIP offload I expected?



- Lots of folks know that SQL statements coming through DDF are zIIP-offloadable to the tune of 55-60%
- Sometimes people see information like this in a DB2 monitor accounting long report, and they see that zIIP CPU time is considerably less than 60% of the total – why?



In this report, data was aggregated at the DB2 connection type level, and this is from the DRDA section of the report

AVERAGE	APPL (CL. 1)
-----	-----
CP CPU TIME	(A) 0.003680
AGENT	0.003680
NONNESTED	0.003614
STORED PRC	(C) 0.000052
PAR. TASKS	0.000000
SECP CPU	(E) 0.000410
SE CPU TIME	(B) 0.003397
NONNESTED	0.003383
STORED PROC	(D) 0.000000



"B / (A+B) is only 48% here, not 55-60%. How come?"



Explaining less-than-expected DRDA zIIP offload (1)



- Check stored procedure CPU time (two fields, labeled “C” and “D” on the report snippet on the preceding slide)
 - If C is much larger than D (and here, D is zero), that indicates that stored procedures are mostly (maybe all) of the external variety (versus native SQL procedures)
 - External stored procedures always run under TCBs (in a stored procedure address space) and so are not zIIP-eligible, even when called by DRDA requesters (more on this to come)





Less-than-expected DRDA zIIP offload (2)

- Check “SECP CPU” time (translation: zIIP-eligible work done by a general-purpose engine – labeled with an “E” on slide 5)
 - If zIIP-eligible work is ready for dispatch and zIIP engines are busy, z/OS waits a few milliseconds and then dispatches the work to a general-purpose engine
 - Check what I call the “zIIP spill-over percentage” ($E/(B+E)$ on slide 5)
 - A small but non-zero value is OK, but if it’s over 5% (figure from slide 5 is almost 11%), I’d be concerned about a zIIP engine contention situation
 - Get with z/OS systems programmer to check on zIIP engine utilization – zIIP contention can become a concern when zIIP utilization gets to about 60% (or less, depending on the number of zIIPs in the LPAR)
 - zIIP over-utilization is of greater consequence in DB2 10 or later system, because prefetch is 100% zIIP-eligible – you don’t want to delay that
 - You can have up to 2 zIIPs per general-purpose engine with a zEC12 or z13, and adding zIIP capacity does not increase software costs



Q: How can I increase zIIP offload for my DDF workload?



- If you have external stored procedures called through DDF, start replacing them with native SQL procedures (see slide 6)
- While an external stored procedure always runs under a TCB in a stored procedure address space, *a native SQL procedure always runs under the task of the calling application process*
 - If process is a DRDA requester, z/OS task will be an enclave SRB in the DDF address space – that makes native SQL procedure zIIP-eligible
 - Ditto for compiled SQL scalar user-defined functions (DB2 10 NFM)
- The low hanging fruit are external stored procedures that:
 1. Access only DB2 data (vs., for example, VSAM data)
 2. Are executed frequently (maximize shift of CPU usage to zIIP engines)
 3. Are relatively simple (coding replacement native SQL procedures will be relatively quick and easy)



Q: How do I know if I have enough DBATs?



- One means of finding out is this DB2 command:

`-DISPLAY DDF DETAIL`

- In the output of this command you'll see a line that looks like this:

```
DSNL092I  ADBAT=      1  QUEDBAT=      0
```

- Value in the QUEDBAT field shows the cumulative number of times (since DDF was last started – which is probably when DB2 was last started) that DDF requests had to wait for a DBAT to become available
 - If the value is greater than zero, you should probably increase the number of DBATs for the DB2 subsystem (set MAXDBAT in ZPARM to a higher value)
 - Remember, DB2 10 increased by 10X the number of threads (DBATs and local threads) that can be concurrently active for a subsystem – *assuming that packages were bound or rebound in a DB2 10 or later environment*



Another way of checking on DBATs (and connections)



- Look at a DB2 monitor statistics long report (or an online display of subsystem statistics), find the section on DDF activity, and check on these two fields:

GLOBAL DDF ACTIVITY	QUANTITY
-----	-----
DBAT/CONN QUEUED-MAX ACTIVE	0.00 (A)
CONN REJECTED-MAX CONNECTED	0.00 (B)

- If field “A” is non-zero, you hit the MAXDBAT limit, and you should probably increase the value of this parameter in ZPARM
- If field “B” is non-zero, you hit the CONDBAT limit (maximum number of connections to this DB2 subsystem) – increase it
- CONDBAT value should be at least 2X the MAXDBAT value



Q: How can I boost the CPU efficiency of my DDF workload?



- Leverage high-performance DBAT functionality
 - Starting with DB2 10 (conversion mode), when a package bound with `RELEASE(DEALLOCATE)` is executed by way of a “regular” DBAT, that DBAT becomes a high-performance DBAT
 - Rather than going back into the DBAT pool at transaction completion, a high-performance DBAT will stay dedicated to connection through which it was instantiated, until it has been reused by 200 transactions
 - CPU savings come from the combination of a persistent thread (a thread that persists through commits) and `RELEASE(DEALLOCATE)` packages
 - Conceptually like `RELEASE(DEALLOCATE)` + CICS-DB2 protected entry threads
 - In-DB2 CPU time can be reduced by 10% or more for simple transactions



Q: Can you use high-performance DBATs selectively?



- Yes – remember that instantiation of high-performance DBATs depends on execution of `RELEASE(DEALLOCATE)` packages
 - So, selective binding of packages executed by DRDA requesters is one control mechanism
 - Example: bind the package of a stored procedure with `RELEASE(DEALLOCATE)`, and the DRDA requester that calls the stored procedure will use a high-performance DBAT
 - Another option: bind IBM Data Server Driver packages (or DB2 Connect packages) into default NULLID collection with `RELEASE(COMMIT)`, and into another collection with `RELEASE(DEALLOCATE)`
 - Then, by way of a client-side data source property, point DDF-using applications to the one collection or the other, depending on whether or not you want an application to use high-performance DBATs



Q: Should I adjust MAXDBAT for high-performance DBATs?



- YES – when a transaction using a high-performance DBAT completes, the high-performance DBAT does NOT go back into the DBAT pool
 - Instead, it stays dedicated to the client connection through which it was instantiated, until it has been reused by 200 transactions
 - THEREFORE, the more high-performance DBATs you have at a given time, the fewer “regular” DBATs you have in the DBAT pool
 - Given a high-enough number of connections from client applications executing `RELEASE(DEALLOCATE)` packages, and a small enough MAXDBAT value, ALL of a DB2 system’s DBATs could go high-performance
 - That would mean zero DBATs in the pool – a situation you’d want to avoid
 - SO, before starting to use high-performance DBATs, consider increasing MAXDBAT value, and monitor high-performance DBAT activity (next slide)





Keeping an eye on high-performance DBATs

- In a DB2 monitor statistics long report, look for these fields in the “DDF activity” section of the report:

GLOBAL DDF ACTIVITY	QUANTITY
-----	-----
CUR ACTIVE DBATS-BND DEALLC	0.00
HWM ACTIVE DBATS-BND DEALLC	0.00

- If the number in the field that I have circled in red (high-water mark for high-performance DBATs) gets anywhere near your MAXDBAT value, make MAXDBAT bigger
 - Do this to ensure that you’ll have a decent number of “regular” DBATs in the DBAT pool



Q: For monitoring, how do I distinguish applications?



- Often, DDF applications are identified in DB2 monitor accounting reports and online displays by authorization ID
- What if different applications connect to DB2 using the same ID?
 - In that case, an option for application differentiation is workstation name
 - Easily set for an application, by several means:
 - WebSphere Application Server: use the administration console GUI to set workstation name as an extended property of an application's data source
 - Via application code, using Java API setClientInfo (for JDBC 4.0 and later)
- Once workstation name is set, you can have your DB2 monitor generate accounting reports with data ordered by that identifier
- More information: sections 5.5, 8.2 of IBM redbook, *DB2 for z/OS and WebSphere Integration for Enterprise Java Applications*
 - <http://www.redbooks.ibm.com/abstracts/sg248074.html?Open>



DDF application architecture



Q: We use DB2 Connect “gateway” servers – should we?



- No – going directly from application server to DB2 for z/OS, using the IBM Data Server Driver Package, is what you should be doing
- Benefits: simplified IT infrastructure, better performance (owing to the fact that “hop” to DB2 Connect gateway server is eliminated)
- Another benefit: quicker problem source identification
 - Suppose that you have an authentication error caused by use of the incorrect password by an application requesting a connection to DB2
 - DB2 error message (DSNL030I) will provide, in THREAD-INFO part of the message text, the IP address* of the “adjacent” (to DB2) server
 - That will be the address of a DB2 Connect gateway server, if that’s what you use, and you won’t know which of the application servers “upstream” from the DB2 Connect gateway server got the authentication error

* See IBM “Technote” at <http://www-01.ibm.com/support/docview.wss?uid=swg21055269> for information on how to interpret the IP address value in the DSNL030I message



More on the IBM Data Server Driver vs. DB2 Connect



- Entitlement to use the IBM Data Server Driver Package is by way of your DB2 Connect license, so if you are licensed for DB2 Connect then you can use the Data Server Driver Package
 - Example: DB2 Connect Unlimited Edition license enables unlimited use of IBM Data Server Driver Package for associated DB2 system(s)
 - Exception to the rule: a “concurrent user” DB2 Connect license requires use of the gateway server configuration
- The Data Server Driver has virtually all the functionality of DB2 Connect (Sysplex workload balancing, connection pooling, transaction pooling, etc.)
 - That said, if you have an application that needs two-phase commit capability AND the client transaction manager uses a dual-transport processing model, you need to use DB2 Connect
 - WebSphere Application Server uses a single-transport processing model



Q: Why are my DDF applications getting thread timeouts?



- Could be that client-side programs are keeping connections from going inactive, and the threads end up timing out instead
 - Quite possible that some client-side programmers are not familiar with the DB2 concept of an inactive connection
 - When DB2 for z/OS is the server, you WANT an application connection to go into an inactive state when a transaction completes
 - Not knowing this, a client-side programmer might have something like `SELECT 1 FROM SYSIBM.SYSDUMMY1` issued periodically from a DDF-using application, just to keep connection to DB2 “alive” – DON'T DO THAT
 - Another factor: a connection won't go inactive unless there is a “clean” commit at end-of-transaction – no WITH HOLD cursors left open, no un-dropped declared global temp tables
 - ALSO: a developer who codes a read-only transaction may think that a commit is not necessary – it is necessary if you want to release locks, and a connection won't go inactive if the DBAT holds locks



Q: Are there considerations for nested stored procedures?



- Yes – for one thing, you want to return result sets from nested stored procedures the right way
- Suppose program A calls stored procedure X, which calls stored procedure Y, which generates a result set that is to be returned to program A – how would you get those rows to program A?
 - Old way (before DB2 10): stored procedure Y puts result set rows in a temporary table, and program A retrieves the rows from that table
 - Result set of a `WITH RETURN TO CALLER` cursor is directly accessible only “one level up” in a chain of nested stored procedure calls
 - Better way, starting with DB2 10: stored procedure Y declares cursor `WITH RETURN TO CLIENT`, and program A (program that initiated the chain of nested stored procedure calls) then fetches rows directly
 - Simplified coding, better performance versus the old temporary table approach



Another nested stored procedure consideration: zIIP offload



- As previously mentioned, a native SQL procedure will always run under the task of its caller
 - If the caller is a DRDA requester, its z/OS task will be a preemptible SRB in the DDF address space, and a native SQL procedure running under such a task is zIIP-eligible (up to 60%)
 - If a DRDA requester calls an external stored procedure, and that stored procedure calls a native SQL procedure, the native SQL procedure will run under the external stored procedure's task
 - That task will be a TCB in a stored procedure address space, and because of that the native SQL procedure will NOT be zIIP-eligible
 - If the external stored procedure is written in COBOL, you might consider having that procedure issue a COBOL CALL to a COBOL subroutine, vs. issuing a call to a native SQL procedure – could be good for CPU efficiency



Q: WebSphere Application Server for z/OS: which JDBC driver should you use?



- Referring to the situation in which WebSphere Application Server (WAS) for z/OS is running in same LPAR as a target DB2 subsystem
- In that case, you can use either the type 2 JDBC driver (local connection to DB2) or the type 4 driver (access to DB2 through DDF) – which should you choose?
- Considerations:
 - Type 2 driver: fewer instructions to get to DB2 (and back) for each SQL statement issued by a Java application, but not much zIIP offload for SQL (SQL executes under the Java application's TCB)
 - Type 4: more path length to get to DB2 (go into LPAR's TCP/IP stack, then through DDF), but SQL statements are zIIP-eligible (up to 60%) because they execute under preemptible SRBs in DDF address space



WAS for z/OS: type 2 vs. type 4 JDBC driver



- Keep in mind that the goal is minimization of general-purpose CPU time, not maximization of zIIP engine CPU time
 - I've seen situations in which the type 4 JDBC driver does best by this measure, but I understand that there are situation in which the type 2 driver would likely deliver lower general-purpose CPU time
 - Type 4 driver might be best choice for applications that issue longer-running SQL statements, and type 2 driver might be better for applications that issue a larger number of quick-running SQL statements
 - It's pretty easy to try them both, so that could be your best bet – use a DB2 monitor accounting report to see which one minimizes class 1 CPU time
- Another consideration: with type 4 driver, application authentication (e.g., with ID and password) can be more of an issue – I'll cover this in the data security part of this session





DDF workload management and control

Q: How should I prioritize DDF and my DDF workload?



- First, get the DDF address space priority right
 - DDF should have same priority as DB2 MSTR, DBM1 address spaces
 - DDF's priority applies only to work done by DDF "system" tasks, and that work consumes VERY little CPU time (see slides 3 and 4)
- At what priority will SQL statements coming through DDF execute?
 - That depends on the priority of the service class(es) to which you assign DDF-using application processes in your WLM policy
 - If you don't classify this work, it will default to "discretionary"
 - Several identifiers for mapping DDF-using applications to service classes
 - DB2 auth ID, collection name, client-provided accounting information...
 - Good write-up on WLM policy set-up for a DDF workload: section 3.3.2 of IBM "redbook" titled DB2 9 for z/OS: Distributed Functions
 - <http://www.redbooks.ibm.com/abstracts/sg246952.html?Open>



Q: Can connections, threads be managed in a granular way?



- YES, by way of the DB2 profile tables (DB2 10 and later)
- Before DB2 10, connections, concurrently active threads, and idle thread timeout could only be managed at DB2 subsystem level
- DB2 10 (and above): manage these limits in a more granular way via:
SYSIBM.DSN_PROFILE_TABLE, SYSIBM.DSN_PROFILE_ATTRIBUTES
 - Insert a row into DSN_PROFILE_TABLE to define scope of a profile
 - Some of the scope-defining identifiers: client IP address or domain name, role, authorization ID, collection name, package name, workstation name
 - Insert row(s) into DSN_PROFILE_ATTRIBUTES to indicate what will be controlled for profile, and how DB2 will respond when limit being reached
 - What: connections, concurrently active threads, idle thread timeout
 - How: issue a warning, or take action (e.g., start failing connection requests)
- More information: [DB2 for z/OS Managing Performance manual](#)





DDF and data security

Q: Can I use stored procedures to bolster data security?



- YES, and for many organizations this is the most important benefit associated with DB2 stored procedures
- The key here is static SQL
 - Client-side programmers often like to use database interfaces, such as JDBC and ODBC, that mean dynamic SQL on the DB2 server side
 - For dynamic SQL DML statements to work, auth ID of application process needs table access privileges (e.g., SELECT, DELETE)
 - Concern: what if someone uses application's ID in an unauthorized way?
 - DB2 stored procedures enable *dynamic invocation of static SQL*
 - Client-side programmers can use their database interface of choice (e.g., JDBC or ODBC) to call stored procedures – and LOTS of programmers are familiar with using stored procedures
 - Data security is strengthened, because application's authorization ID needs only EXECUTE privilege on stored procedures – not table access privileges



Q: Can I limit the use of DDF application passwords and IDs?




- YES – this is what roles and trusted contexts are for
- DDF-using applications typically connect to DB2 using ID and password
 - Likely that multiple people know these authentication credentials – how can an organization protect against them being misused?
 - Solution: do NOT grant privileges to DDF-using application's authorization ID – instead, create a role and grant needed privileges to the role
 - Then, create trusted context so that privileges granted to role will ONLY be usable by application that connects to DB2 using a particular ID *and from a particular application server or servers* (identified by IP address)
 - Now you've reduced the threat scope – you need to secure that application server (or servers), but you don't need to worry about someone using the application's ID and password from another IP address



Q: Can I map users' "network" IDs to RACF IDs?



- YES – individual end-user IDs (in addition to application's ID) can be passed to DB2 from DDF-connected applications, and users' "network" IDs can be mapped to RACF IDs
 - That can be done through something called enterprise identity mapping
 - Involves three steps:
 - The RACF RACMAP command defines a "distributed identity filter" that maps one or more end-user IDs to a RACF ID 
 - A DB2 trusted context is defined, with the mapped-to RACF ID specified in the WITH USE FOR part of the CREATE TRUSTED CONTEXT statement
 - Application gets trusted connection to DB2, requests auth ID switch (to pass to DB2 the distributed user identity and name of associated registry) – do-able via WebSphere Application Server or APIs: Java, CLI/ODBC, .NET
 - See section 7.3 of redbook, *Security Functions of IBM DB2 10 for z/OS*
 - <http://www.redbooks.ibm.com/abstracts/sg247959.html?Open>



Q: How do I manage passwords for DDF-using applications?



- Updating password for a DDF application can cause authentication failures if password is changed in RACF, not on application server
- A mitigating technique: have two IDs for each application, vs. one
 - Say application X is using ID A and its associated password
 - Before the ID A password expires, create a new password for ID B
 - Change each instance of application X to use ID B (and password) instead of A
 - IDs A and B (and passwords) are valid during switch-over, so no need for all-at-once change
 - If application X is running on at least 2 servers, recycling each server in turn does not cause an application outage
 - After switch-over to ID B, change password for ID A

(Item 14 in Web document by IBM WebSphere developer Bill O'Donnell:

http://www.ibm.com/developerworks/websphere/techjournal/1003_botzum/1003_botzum.html)



Or, instead of managing passwords: use certificates, instead



- Organizations can obtain digital certificates, and associated public and private keys, using RACF and/or other tools (and, in some cases, 3rd-party certificate authorities such as Symantec and GlobalSign)
 - These certificates and keys can be used for both server and client authentication purposes
- Client applications using a direct-to-DB2 connection (e.g., via the IBM Data Server Driver Package vs. a DB2 Connect gateway server) can use certificates to authenticate to a DB2 for z/OS server
- z/OS and DB2 can be set up so that a client can authenticate with an ID and a certificate, *without a password being required*



More on certificate-based client authentication



- Certificate-based client authentication typically implies:
 1. Certificate-based server authentication, as well (i.e., server provides its certificate to client, as well as vice-versa) – that’s why you sometimes see client-based authentication described as “mutual authentication”
 2. SSL encryption between client and DB2 (which DB2 for z/OS supports)
- Certificate-based client authentication requires DB2 10 for z/OS or later



And just a bit more...



- A detailed description of client- and server-side configuration and programming for certificate-based authentication can be found in the IBM “redpaper” available at this URL:
 - <http://www.redbooks.ibm.com/abstracts/redp4799.html?Open>
- **Note:** figure 11 on page 57 of this redpaper indicates that a DB2 for z/OS trusted context must be defined for the ID associated with a client’s certificate
 - NOT TRUE – see text for DB2 APAR PM64332 (and while you’re at it, see the text for DB2 APAR PM53450 for useful related information)





Thank You

