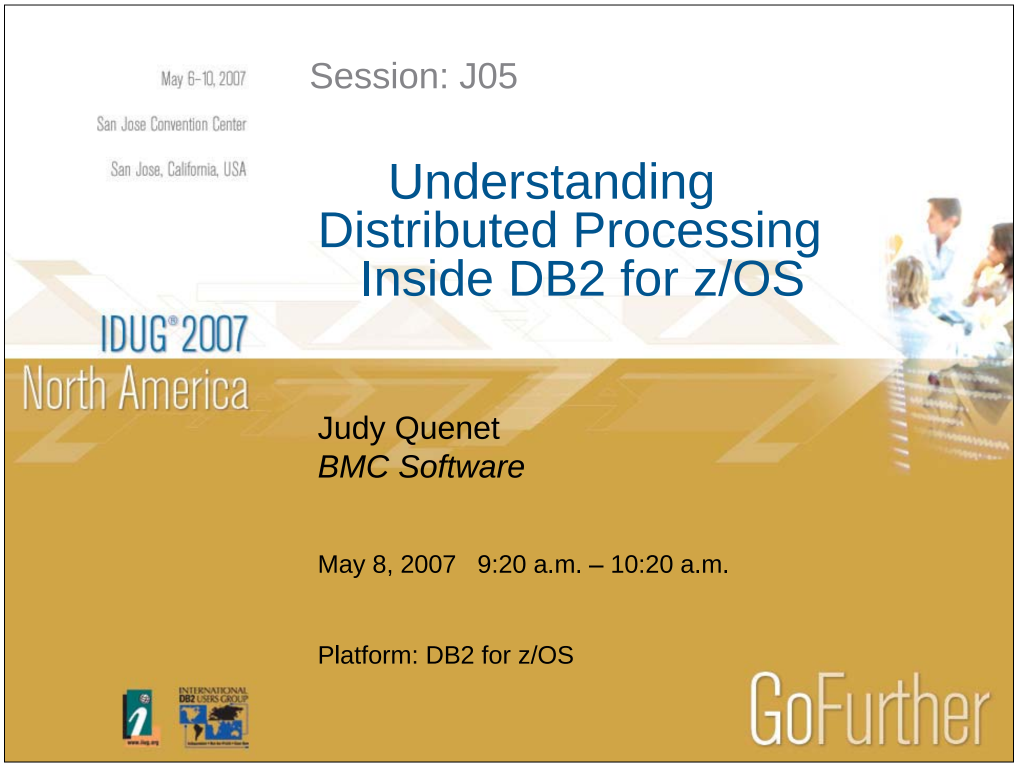


May 6–10, 2007

San Jose Convention Center

San Jose, California, USA

IDUG® 2007 North America

Session: J05

Understanding Distributed Processing Inside DB2 for z/OS

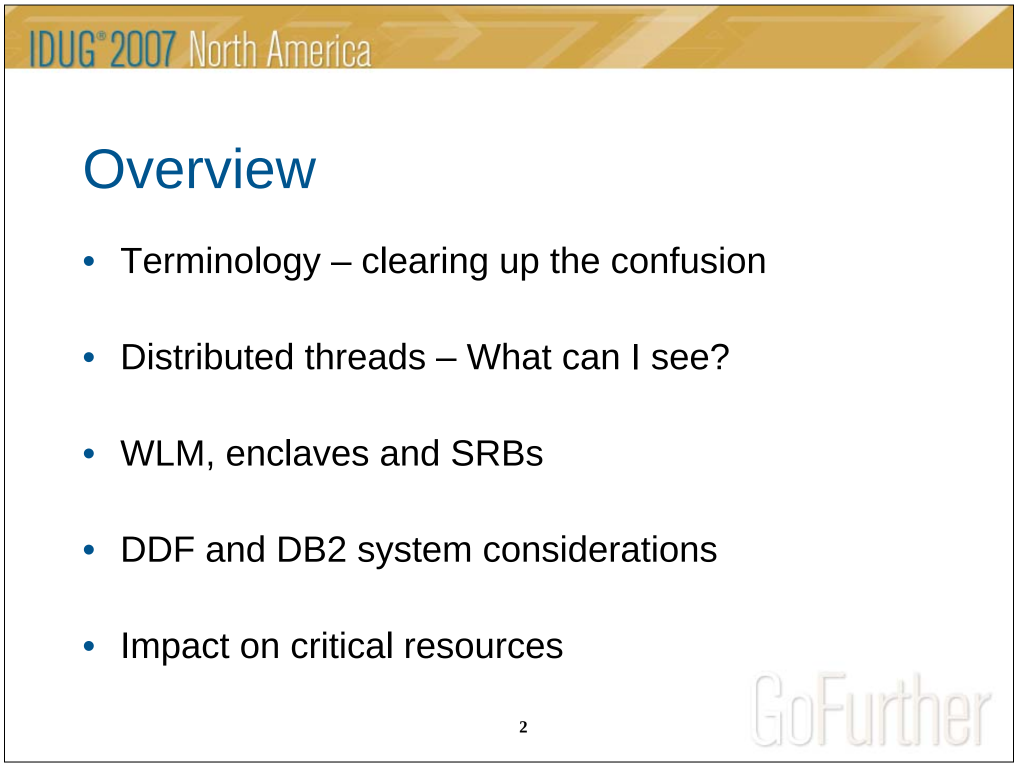
Judy Quenet
BMC Software

May 8, 2007 9:20 a.m. – 10:20 a.m.

Platform: DB2 for z/OS

GoFurther

DB2 users are seeing more and more of their workload originating off the mainframe. This shift to more distributed processing presents new challenges for the DB2 performance specialist and brings up new questions about how to manage performance for distributed threads. This session will use BMC’s MAINVIEW for DB2 product to answer many questions about distributed processing, including where these threads originate, what statistics and accounting data is available about distributed threads, and why WLM definitions and z/OS enclaves are important in this area. This presentation provides practical knowledge identifying performance metrics associated with distributed processing.



IDUG® 2007 North America

Overview

- Terminology – clearing up the confusion
- Distributed threads – What can I see?
- WLM, enclaves and SRBs
- DDF and DB2 system considerations
- Impact on critical resources

2

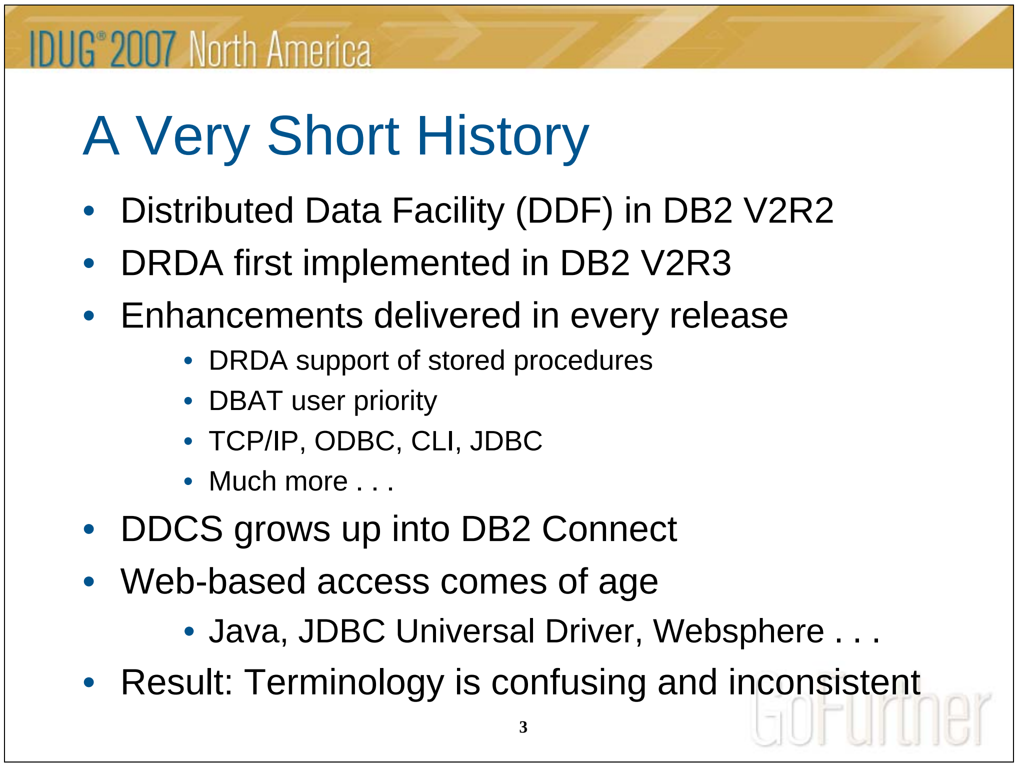
GoFurther

Today I will be discussing distributed processing from <u>inside</u> DB2, looking primarily at Distributed Data Facility (DDF) processing and its impact on thread activity and DB2 resources, as well as the data provided by DB2 to monitor and analyze it.

Although there are many other ways that distributed clients can access DB2, they are entering through other attach types, such as CICS, IMS, MQ and RRS. These are beyond the scope of this presentation, since they are subject to most of the same constraints and monitoring methods as other allied threads. Indoubt situations are also not addressed, since that again is a broad subject.

I will begin by covering some of the basic terminology and attempt to clear up some of the confusing and contradictory usage that is still present in various documents – and also in the instrumentation data itself.

The focus then turns to distributed thread processing modes, the available options and their impact on what you see in thread accounting and DB2 statistics or display output. Interweaved with this discussion is an overview of WLM and distributed enclaves and SRBs. Then we will look more closely at the instrumentation data available to you to monitor and tune your distributed workloads, focusing on what measures are most important – and why. This covers both DDF and DB2 considerations. At the end, we will wrap up the topic by reviewing what impact distributed workloads can have on critical system and DB2 resources.


IDUG® 2007 North America

A Very Short History

- Distributed Data Facility (DDF) in DB2 V2R2
- DRDA first implemented in DB2 V2R3
- Enhancements delivered in every release
  - DRDA support of stored procedures
  - DBAT user priority
  - TCP/IP, ODBC, CLI, JDBC
  - Much more . . .
- DDCS grows up into DB2 Connect
- Web-based access comes of age
  - Java, JDBC Universal Driver, Websphere . . .
- Result: Terminology is confusing and inconsistent

3

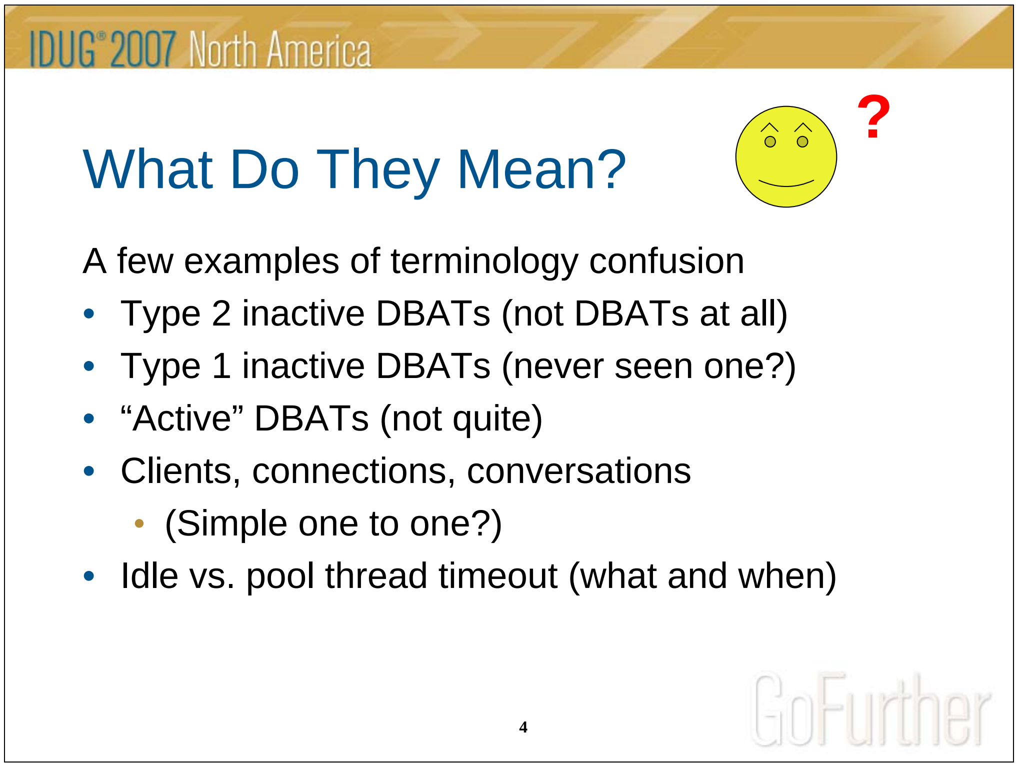
GoFurther

A very short history!

The first DDF (Distributed Data Facility) only provided access from one mainframe DB2 to another, and was first delivered in DB2 V2R2.
Note: Any reference to DB2 is shorthand for DB2 for z/OS.
MAINVIEW for DB2 is sometimes shortened to MVDB2.

The DRDA protocol (level 1) was first implemented in DB2 V2R3, providing the first client / server access. DB2 V4 was a significant step forward, with DRDA supported stored procedures to reduce network traffic, an increased number of supported connections (25,000), and DBATs processing with different priorities (not just as standard DIST SRBs). DB2 V5 (with DRDA level 3) supported TCP/IP, ODBC, CLI and JDBC, and DDF enhancements have continued to be delivered in every DB2 release. Concurrently, the original DDCS has been growing up into DB2 Connect.
Now we are seeing web-based access coming of age and changing rapidly, with Java applications, the DB2 SQLJ and JDBC Universal Driver, Websphere, and other AR/AS servers between the remote clients and DB2 for z/OS (like SAP).

One of the problems with understanding DDF is that the terminology hasn't always kept up with the rapid changes, and can often be confusing and inconsistent. Part of this is historical, with the swift migration from simple DB2 to DB2, to the current wealth of connectivity options and frequent enhancements in DB2 for z/OS. Another issue arises from the culture clash between the mainframe and distributed worlds, compounded by the many new distributed clients and access methods.



IDUG® 2007 North America

What Do They Mean?

A few examples of terminology confusion

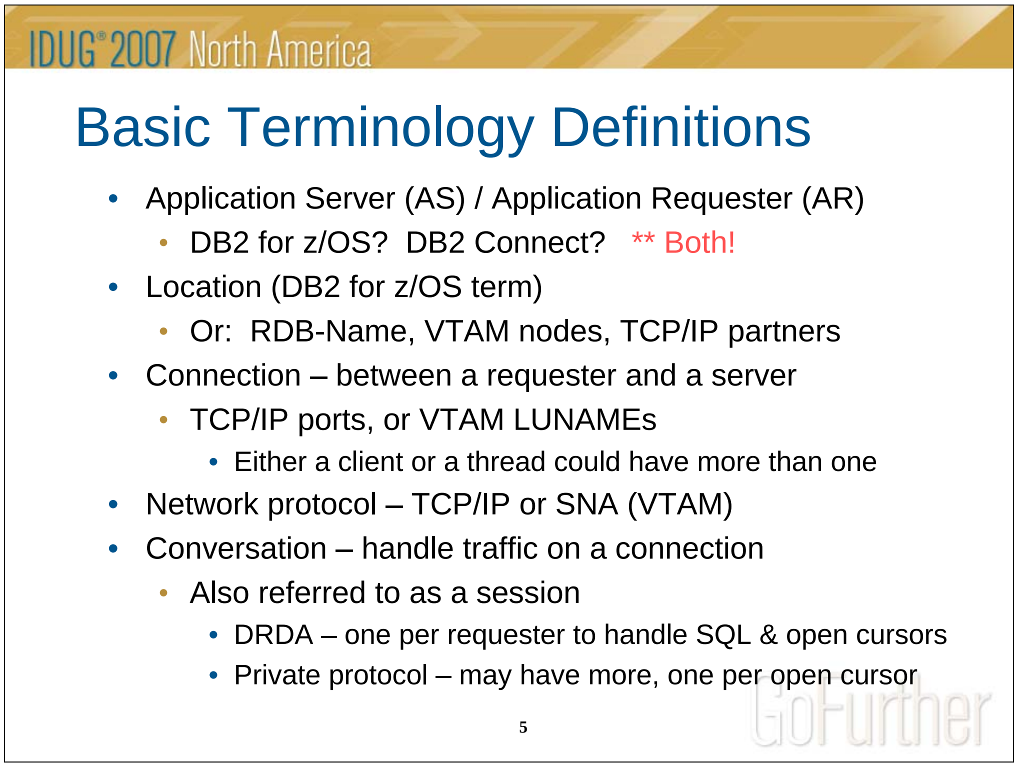
- Type 2 inactive DBATs (not DBATs at all)
- Type 1 inactive DBATs (never seen one?)
- "Active" DBATs (not quite)
- Clients, connections, conversations
  - (Simple one to one?)
- Idle vs. pool thread timeout (what and when)

4

GoFurther

These are a sampling of some of the terms that can be confusing.

I am going to start back at the basics of thread identification, just to get us all on the same page. This will also provide a cross-reference of DB2 identifiers for distributed workloads and the WLM classification attributes. However, I will move fairly quickly through these first slides.


IDUG® 2007 North America

Basic Terminology Definitions

- Application Server (AS) / Application Requester (AR)
  - DB2 for z/OS? DB2 Connect? ** Both!
- Location (DB2 for z/OS term)
  - Or: RDB-Name, VTAM nodes, TCP/IP partners
- Connection – between a requester and a server
  - TCP/IP ports, or VTAM LUNAMEs
    - Either a client or a thread could have more than one
- Network protocol – TCP/IP or SNA (VTAM)
- Conversation – handle traffic on a connection
  - Also referred to as a session
    - DRDA – one per requester to handle SQL & open cursors
    - Private protocol – may have more, one per open cursor

5

GoFurther

Let's start by reviewing some of the basic terminology.

Application Server (AS) / Application Requester (AR)
- DB2 for z/OS is primarily a server
  - A real application server from a JDBC Type 4 driver
  - A database server from DB2 Connect
- It can also be a requester to obtain data from a remote data base (DB2 for z/OS or LUW)
- DB2 Connect has morphed from AR to both AR / AS.

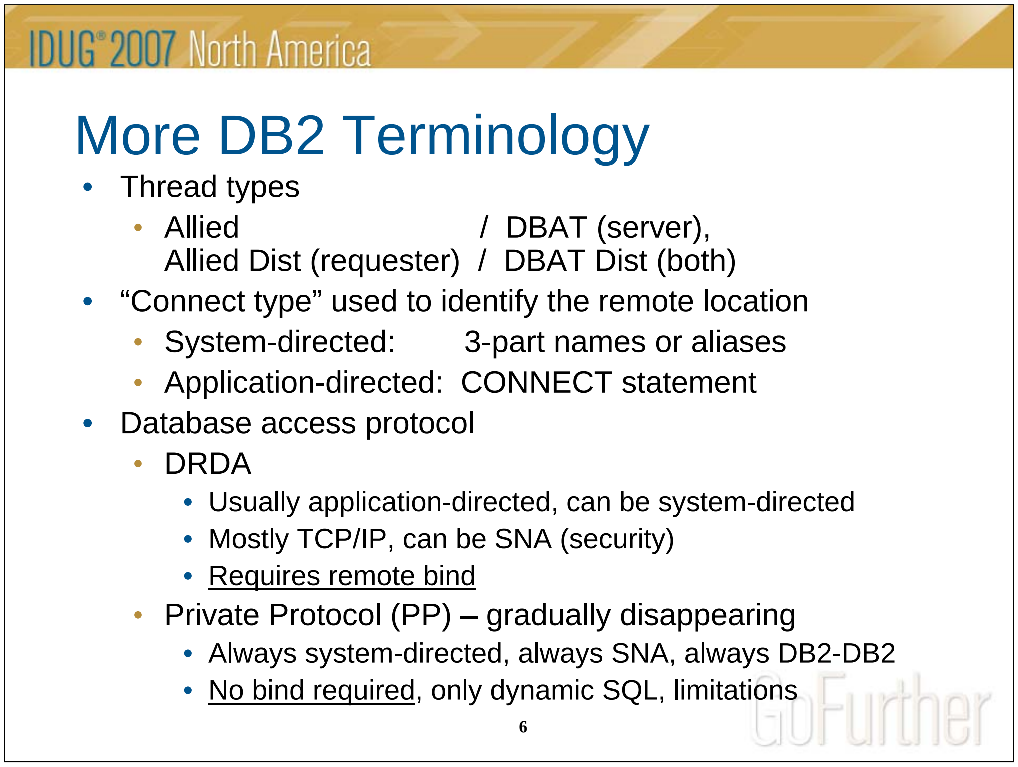
Location - a DB2 for z/OS term
- Distributed term is RDB_NAME or database
- VTAM – nodes
- TCP/IP – partner, or address:port

Connection – between a requester and a server
- For TCP/IP connections, it is defined by the IP addresses and the two ports
- For SNA, it is defined by the LUNAMEs of the nodes

(Note that either a client or a thread may have multiple connections.)

Network protocol – TCP/IP or SNA (VTAM)

Conversation
- Also sometimes referred to as a session
- It handles traffic on a connection
  - DRDA - one per connection
    - Must share for different SQL / open cursors
    - Affects blocking and network traffic
  - Private Protocol
    - Two system conversations allocated
      - Used by DB2 to manage communications – send / receive
    - May have multiple conversations per active thread
      - Can manage one conversation per open cursor if resources allow


IDUG® 2007 North America

More DB2 Terminology

- Thread types
  - Allied / DBAT (server),
    Allied Dist (requester) / DBAT Dist (both)
- "Connect type" used to identify the remote location
  - System-directed: 3-part names or aliases
  - Application-directed: CONNECT statement
- Database access protocol
  - DRDA
    - Usually application-directed, can be system-directed
    - Mostly TCP/IP, can be SNA (security)
    - Requires remote bind
  - Private Protocol (PP) – gradually disappearing
    - Always system-directed, always SNA, always DB2-DB2
    - No bind required, only dynamic SQL, limitations

GoFurther

6

More DB2 Terminology:

Thread types

- Allied – local thread with no distributed work
- DBAT – server (AS)
- Allied Distributed – local thread plus distributed requester (AR)
- DBAT Distributed – DBAT plus distributed requester (AS / AR)

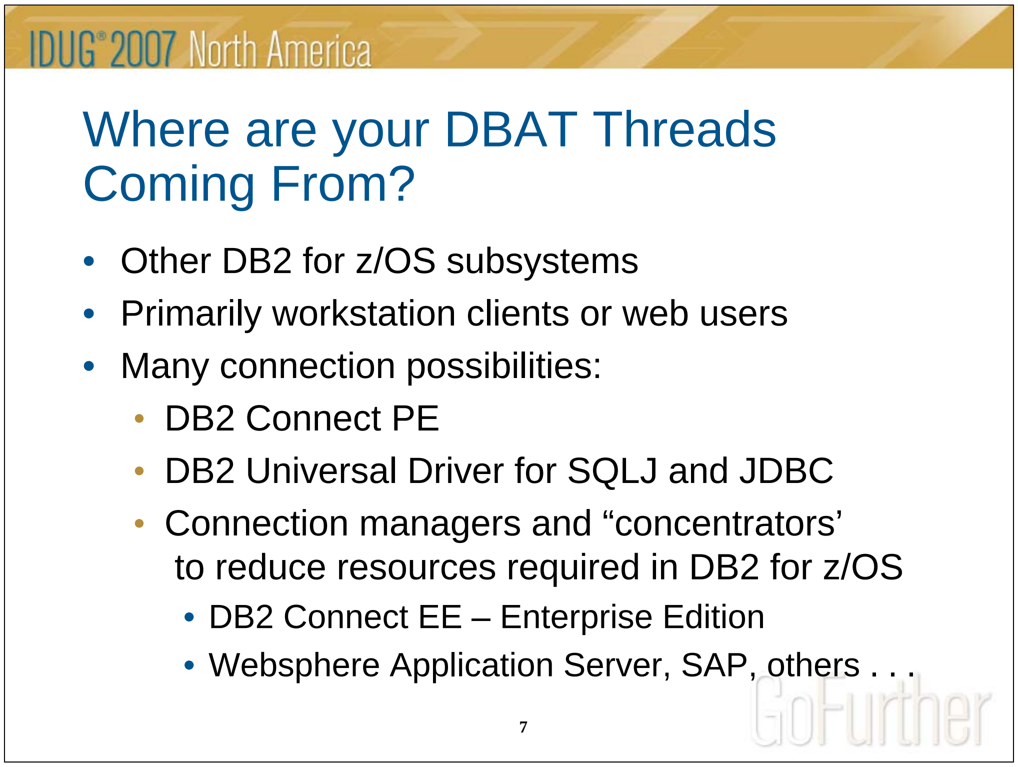
"Connect type" - used in the application to identify the remote location

- System-directed uses 3-part names or aliases
- Application-directed uses an SQL CONNECT statement

  (Note: This terminology was actually used for Connect Type in the DB2 accounting records in some of the earlier releases, but now the data base access protocol name is used.)

Database access protocol

- DRDA
  - Usually application-directed, can be system-directed
  - Mostly TCP/IP, can be SNA (to provide an extra level of security)
  - Requires a remote bind of a package to execute SQL
- Private Protocol - or "PP" for short, is gradually disappearing.
  - Always system-directed, always SNA, only DB2 to DB2
  - No bind required, supports only dynamic SQL, and has other limitations, especially regarding SQL features
- The use of a remote package (or not) is the key differentiator between the two protocols.



IDUG® 2007 North America

Where are your DBAT Threads Coming From?

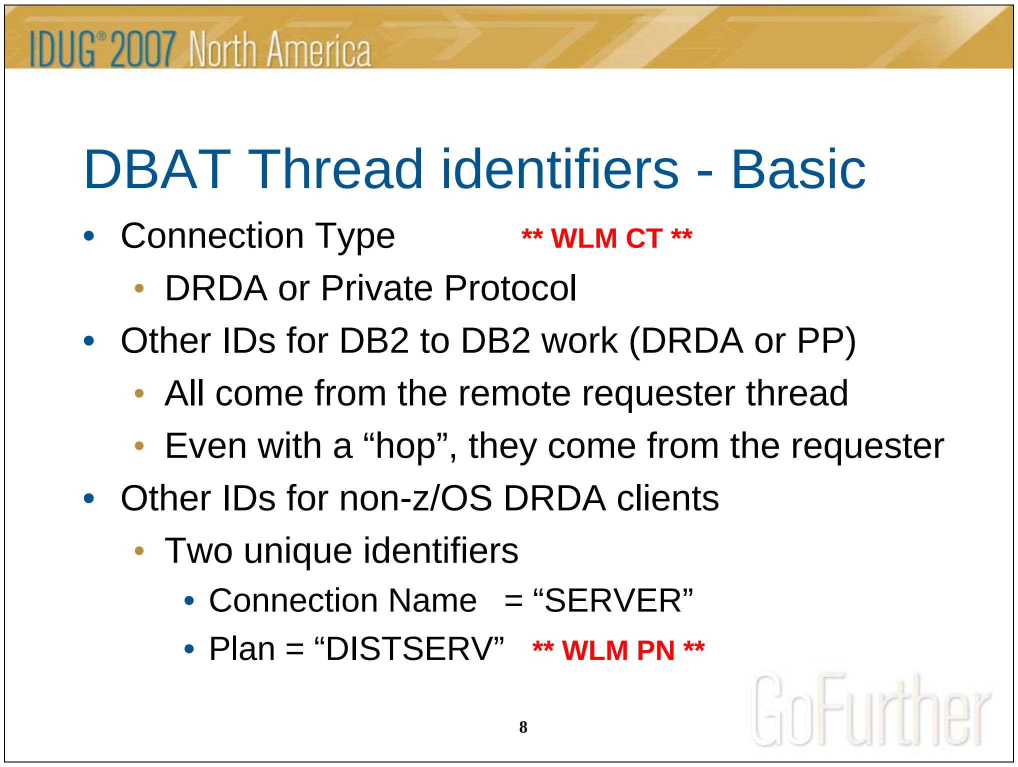
- Other DB2 for z/OS subsystems
- Primarily workstation clients or web users
- Many connection possibilities:
  - DB2 Connect PE
  - DB2 Universal Driver for SQLJ and JDBC
  - Connection managers and "concentrators' to reduce resources required in DB2 for z/OS
    - DB2 Connect EE – Enterprise Edition
    - Websphere Application Server, SAP, others . . .

7

GoFurther

Where are the DBAT threads coming from? Sometimes from other DB2 for z/OS subsystems, but primarily from workstation clients or web users nowadays. There are many connection possibilities. Some of the most common:

- DB2 connect PE – Personal Edition (Windows / Linux)
- DB2 Universal Driver for SQLJ (static SQL) and JDBC (dynamic SQL)
- Connection managers and "concentrators"
    - Primarily to reduce the number of client connections kept inside DB2
  - DB2 Connect EE – Enterprise Edition
    - For example (since this is one of the most common)
      - With connection pooling, all connected agents also have connections inside DB2
      - With connection concentrator, only <u>active</u> agents in DB2 Connect have connections inside DB2
        - Active "coordinator agents" are released at commit
        - "Logical agents" are released at termination
  - Websphere Application Server
  - SAP and other application-specific implementations


IDUG® 2007 North America

DBAT Thread identifiers - Basic

- Connection Type **** WLM CT ****
  - DRDA or Private Protocol
- Other IDs for DB2 to DB2 work (DRDA or PP)
  - All come from the remote requester thread
  - Even with a “hop”, they come from the requester
- Other IDs for non-z/OS DRDA clients
  - Two unique identifiers
    - Connection Name = “SERVER”
    - Plan = “DISTSERV” **** WLM PN ****

8

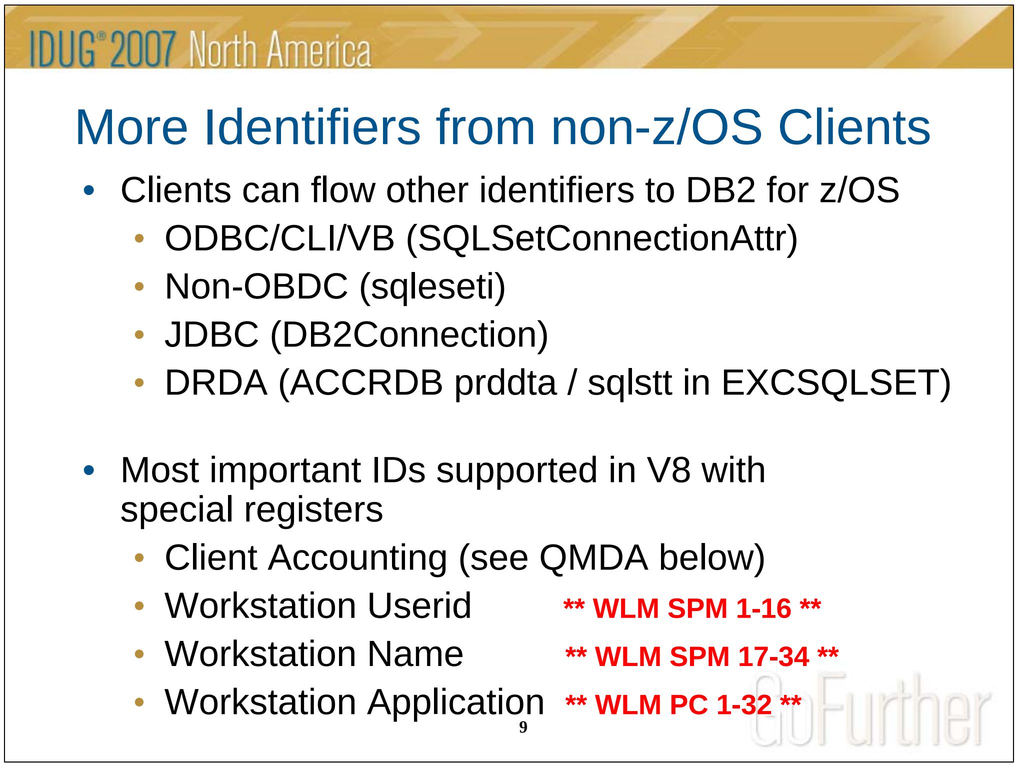
GoFurther

Understanding the various thread identifiers is very important because they are used for WLM workload classification to set the priority at which each request is processed. In the following slides, the WLM classification attributes are noted for reference as **WLM xx”, where “xx” is the classification type.

Most of the important identifiers are found in the accounting correlation header, QWHC. This header not only prefixes the accounting records IFCID 03 / 239, but also other thread-related IFCIDs, like SQL or timeouts. The IFCID field names are included in the notes.

- Connection Type (QWHCATYP)
  - DRDA (flag = 7) / Private Protocol (flag = 8)
  - The value passed to WLM is always “DIST “ **(** WLM CT **)**
- Other IDs
  - For DB2 for z/OS to DB2 for z/OS only (PP or DRDA)
  - All other IDs are those of the remote DB2 requester thread
  - Even with a “hop” to another DB2 for z/OS, the original requester IDs are passed on

- For non-z/OS DRDA clients, there are two unique identifiers
  - Connection Name (QWHCCN) = “SERVER”
  - Plan (QWHCPLAN) = “DISTSERV” **** WLM PN ****



IDUG® 2007 North America

More Identifiers from non-z/OS Clients

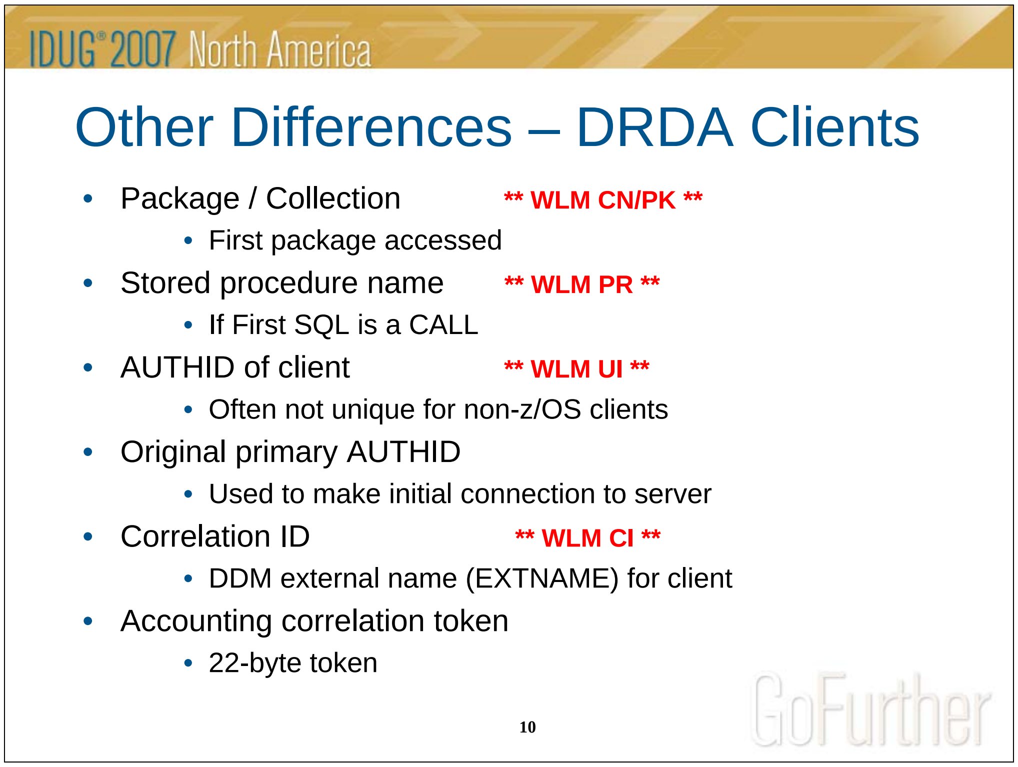
- Clients can flow other identifiers to DB2 for z/OS
  - ODBC/CLI/VB (SQLSetConnectionAttr)
  - Non-OBDC (sqleseti)
  - JDBC (DB2Connection)
  - DRDA (ACCRDB prddta / sqlstt in EXCSQLSET)
- Most important IDs supported in V8 with special registers
  - Client Accounting (see QMDA below)
  - Workstation Userid **** WLM SPM 1-16 ****
  - Workstation Name **** WLM SPM 17-34 ****
  - Workstation Application **** WLM PC 1-32 ****

9

GoFurther

Distributed clients can flow other identifiers to DB2
- ODBC/CLI/VB (SQLSetConnectionAttr)
  Example: SQL_ATTR_INFO_WRKSTNAME
- Non-OBDC (sqleseti Administrative API function)
- JDBC (DB2Connection)
  Example: DB2Connection.setDB2ClientWorkstation
- DRDA (ACCRDB prddta / sqlstt in EXCSQLSET command)

Most important IDs are supported in V8 with special registers
- Client Accounting (covered in the next slides)
- Workstation Userid (QWHCEUID) **** WLM SPM 1-16 ****
- Workstation Name (QWHCEUWN) ****WLM SPM 17-34 ****
- Workstation Application / Transaction (QWHCEUTX) **** WLM PC 1-32 ****


IDUG® 2007 North America

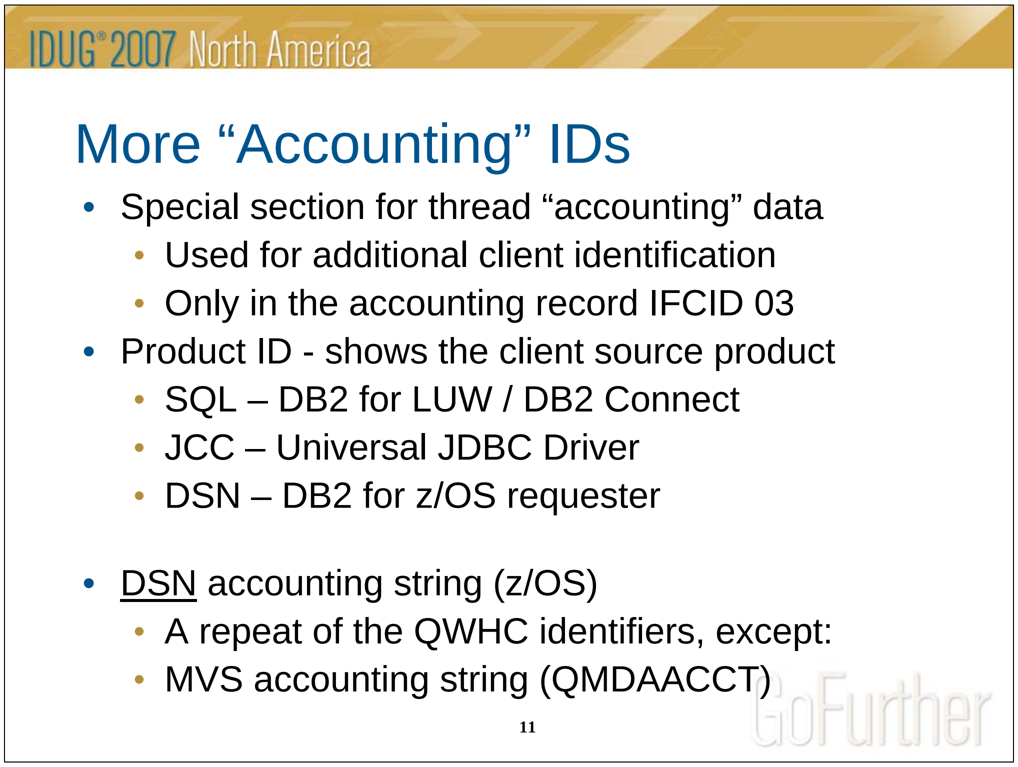
Other Differences – DRDA Clients

- Package / Collection **** WLM CN/PK ****
  - First package accessed
- Stored procedure name **** WLM PR ****
  - If First SQL is a CALL
- AUTHID of client **** WLM UI ****
  - Often not unique for non-z/OS clients
- Original primary AUTHID
  - Used to make initial connection to server
- Correlation ID **** WLM CI ****
  - DDM external name (EXTNAME) for client
- Accounting correlation token
  - 22-byte token

10

GoFurther

Other differences for DRDA clients
- <u>Package / Collection</u> (QPACPKNM / PKID) **** WLM CN/PK ****
  - First package accessed (DRDA requires a local package bind)
- <u>Stored procedure name</u> **** WLM PR ****
  - Only if the first SQL is a CALL statement
- <u>AUTHID</u> (QWHCAID) of client **** WLM UI ****
  - Often not unique from distributed clients
  - Can be changed from OPID with a distributed authorization exit
- <u>Original Primary AUTHID</u> (QWHCOPID)
  - User ID used to make the initial connection to the server
    - Passed by requester in Allocate conversation flow (FMH5)
    - Passed in the DRDA security flow
    - Derived by the server from the RACF passticket
- <u>Correlation ID</u> (QWHCCV) ****WLM CI****
  - For non-z/OS DRDA clients
    - First 12 characters of the DDM external name
      (EXTNAM parameter of the DDM EXCSAT command received as part of the SQL CONNECT statement)
    - DB2 Connect assigns the application program name by default
- <u>Accounting correlation token</u> (QWHCTOKN) (22 bytes)
  - CCRTKN (correlation token) of ACCRDB (access RDB) command
    (received from requester during connect processing) (Also used by RRS)



IDUG® 2007 North America

More “Accounting” IDs

- Special section for thread “accounting” data
  - Used for additional client identification
  - Only in the accounting record IFCID 03
- Product ID - shows the client source product
  - SQL – DB2 for LUW / DB2 Connect
  - JCC – Universal JDBC Driver
  - DSN – DB2 for z/OS requester

- <u>DSN</u> accounting string (z/OS)
  - A repeat of the QWHC identifiers, except:
  - MVS accounting string (QMDAACCT)

11

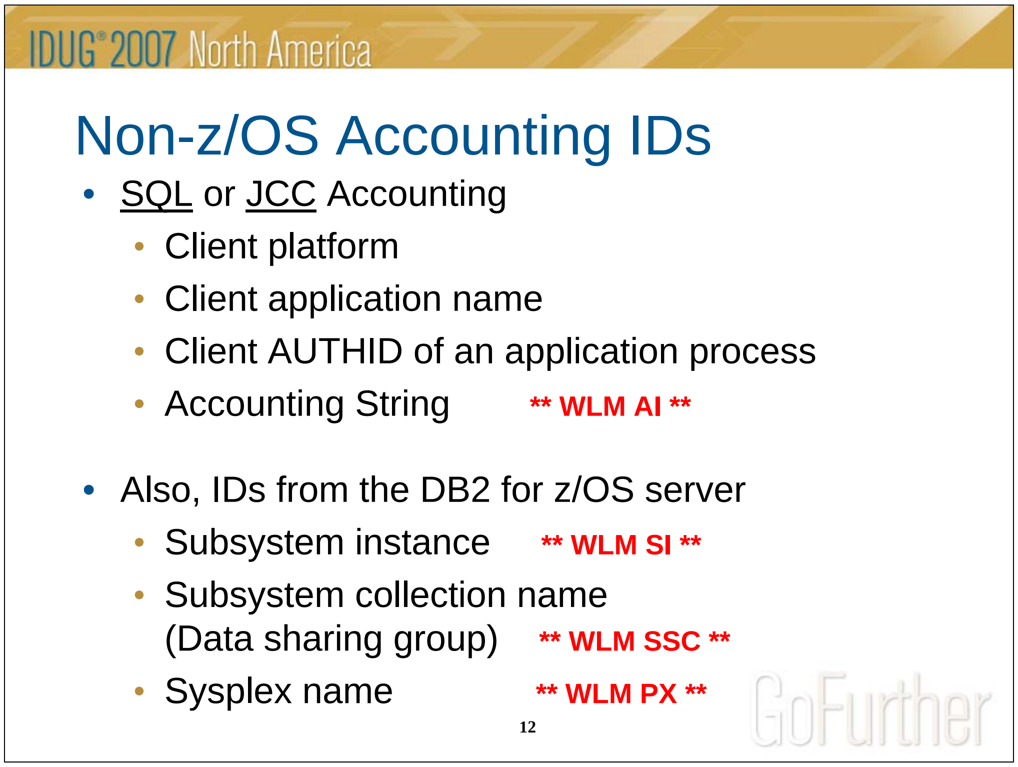
There is an extra section called the QMDA that only appears in IFCID 03 (and the related IFCID 148 for active threads), and not in the other thread-related IFCIDs. It includes other accounting data that can be quite valuable for some distributed workloads.

The Accounting string (QMDAAIF) includes several sections

- Product ID (QMDAPRID) <u>shows source</u> and version of the client product

  - **SQL** (DB2 LUW), **DSN** (DB2 for z/OS), **JCC** (Universal JDBC Driver)
  - (less important) ARI (DB2 for VM&VSE), QSQ (DB2 for i/Series)

For a DB2 for z/OS requester, the rest of the data is mapped by QMDAINFO. It is mostly a repeat of the requester IDs already in the QWHC. However, it does also include an accounting string (QMDAACCT) that maps the value from the ACCT= parameter on the job statement of the address space in which the SQL application is running.


IDUG® 2007 North America

Non-z/OS Accounting IDs

- SQL or JCC Accounting
  - Client platform
  - Client application name
  - Client AUTHID of an application process
  - Accounting String **** WLM AI ****
- Also, IDs from the DB2 for z/OS server
  - Subsystem instance **** WLM SI ****
  - Subsystem collection name (Data sharing group) **** WLM SSC ****
  - Sysplex name **** WLM PX ****

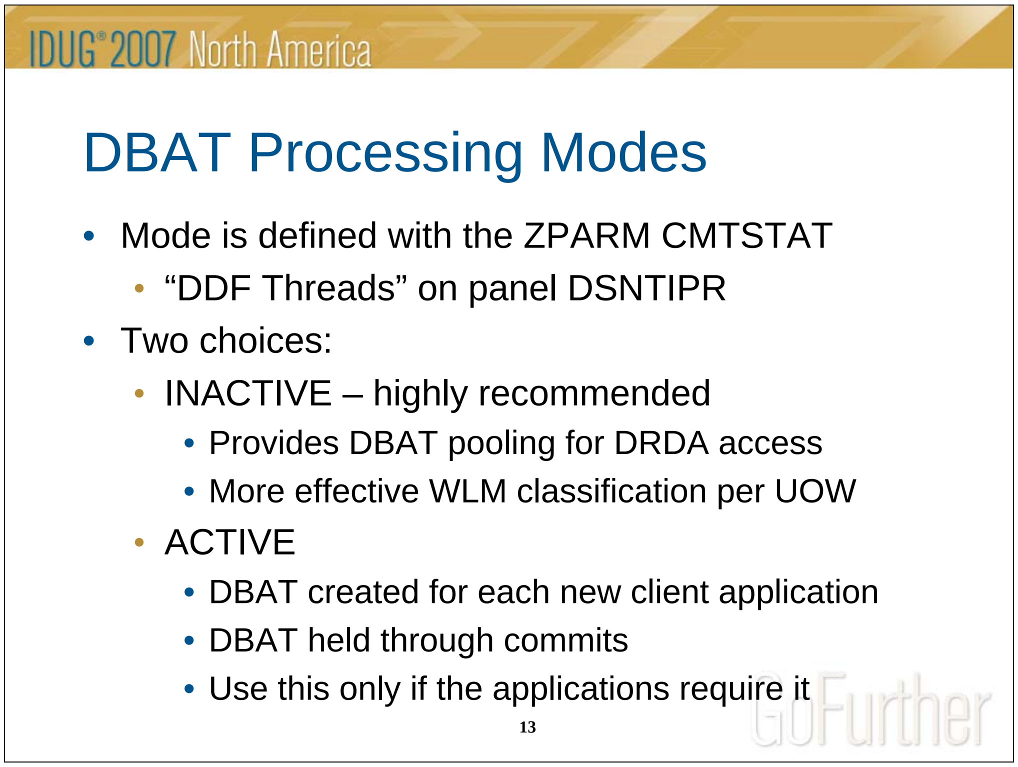
12

GoFurther

For the non-z/OS clients, identified by product IDs SQL or JCC, the rest of the data is mapped by QMDASQLI. It includes:
- Client Platform (PLAT)
- Accounting String (SUFX) ** **WLM AI ****
  - It is 200 bytes long, but the length processed by WLM is less

Just for completeness, there are also three types related to the DB2 for z/OS server processing the work:

- Subsystem instance	**** WLM SI ****
- Subsystem collection name (data sharing group)	**** WLM SSC ****
- Sysplex name	**** WLM PX ****



IDUG® 2007 North America

DBAT Processing Modes

- Mode is defined with the ZPARM CMTSTAT
  - "DDF Threads" on panel DSNTIPR
- Two choices:
  - INACTIVE – highly recommended
    - Provides DBAT pooling for DRDA access
    - More effective WLM classification per UOW
  - ACTIVE
    - DBAT created for each new client application
    - DBAT held through commits
    - Use this only if the applications require it

13

GoFurther

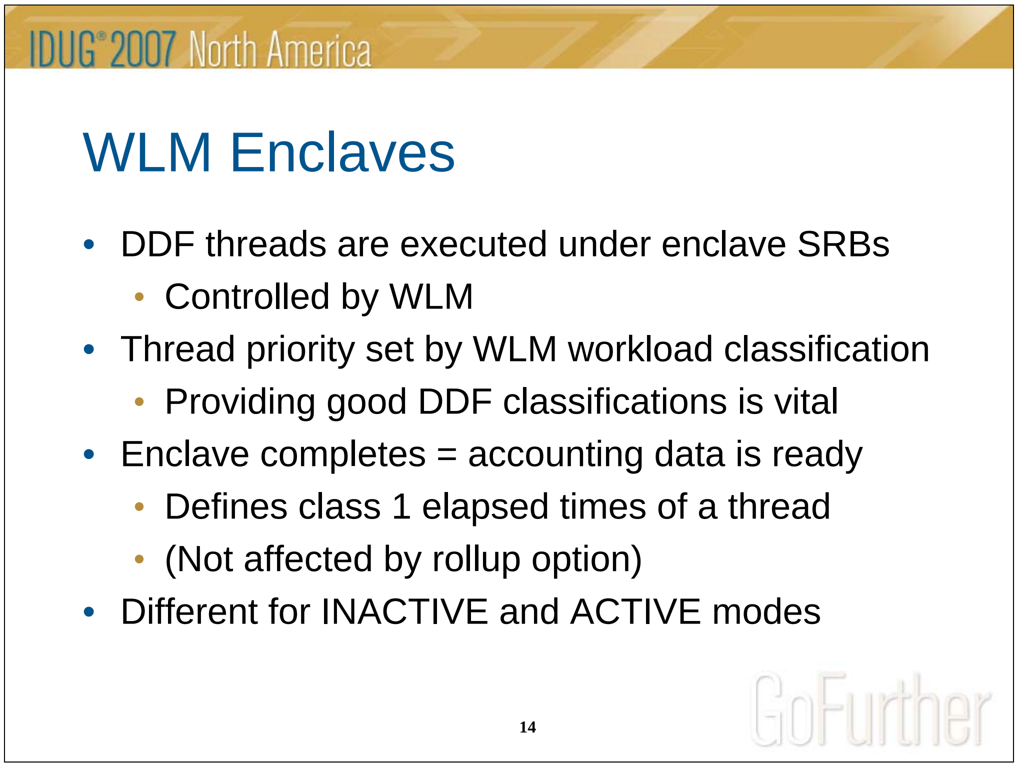
DBAT processing mode is one of the most critical options in DDF, since it defines the flow of DBAT processing. Although there are two options, one is definitely preferred and the other much less in use. However, it is very important to understand both, because much of the DDF documentation sources refer to both – and tend to combine the discussion of other choices and also the processing behavior affected by these modes. This can make it difficult to understand what really applies to a specific situation. It is included here to help clarify the terminology and processing flow.

Mode is defined with the ZPARM CMTSTAT, specified as "DDF THREADS" on panel DSNTIPR. It can be either ACTIVE or INACTIVE. The default has been changed from ACTIVE to INACTIVE, since it is highly recommended.

<u>INACTIVE</u>
- Provides DBAT pooling for DRDA access
- Supports a large number of connections (clients)
- Reduces resource usage (DIST CPU, DBM1 storage, real storage)
- Provides more effective WLM workload classification per unit of work (UOW)

<u>ACTIVE</u>
- DBAT created for each new client connection
- DBAT held through commits until deallocation / termination
- All resources held
- More likely to queue waiting for a DBAT

** use only if the applications require it


IDUG® 2007 North America

WLM Enclaves

- DDF threads are executed under enclave SRBs
  - Controlled by WLM
- Thread priority set by WLM workload classification
  - Providing good DDF classifications is vital
- Enclave completes = accounting data is ready
  - Defines class 1 elapsed times of a thread
  - (Not affected by rollup option)
- Different for INACTIVE and ACTIVE modes

14

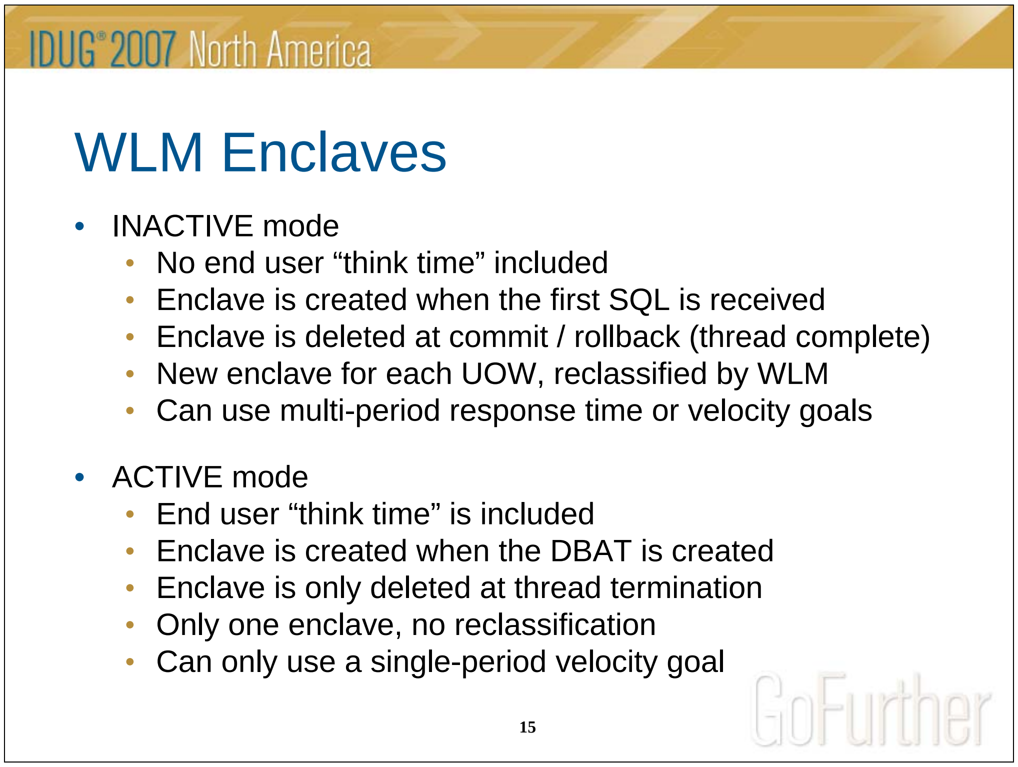
GoFurther

DDF threads are executed under enclave SRBs controlled by WLM. These are preemptible SRBs, designed to process at the priority of the end user. Although "owned" by the DDF address space DIST, DDF enclaves are considered "independent".
Note that DIST SRBs used for queue management, etc., are regular SRBs that are not preemptible. The DDF "DIST" address space workload is classified in WLM as a started task (STC), like the other DB2 address spaces.

The priority of a DBAT is set by its WLM workload classification rule in the active WLM policy. Note that classification is required for DDF to function well. The default classification is at the bottom of the heap - SYSOTHER, or "Discretionary" work.

You first decide how many different service classes you need, and how to define their performance goals. Then you write classification rules, defined for SUBSYS=DDF, to classify transactions coming in through DDF to be assigned to one of these service classes.

What is the duration of the enclave? It depends! It is different for INACTIVE and ACTIVE modes. Note that this also determines the thread class 1 times and accounting.


IDUG® 2007 North America

WLM Enclaves

- INACTIVE mode
  - No end user “think time” included
  - Enclave is created when the first SQL is received
  - Enclave is deleted at commit / rollback (thread complete)
  - New enclave for each UOW, reclassified by WLM
  - Can use multi-period response time or velocity goals
- ACTIVE mode
  - End user “think time” is included
  - Enclave is created when the DBAT is created
  - Enclave is only deleted at thread termination
  - Only one enclave, no reclassification
  - Can only use a single-period velocity goal

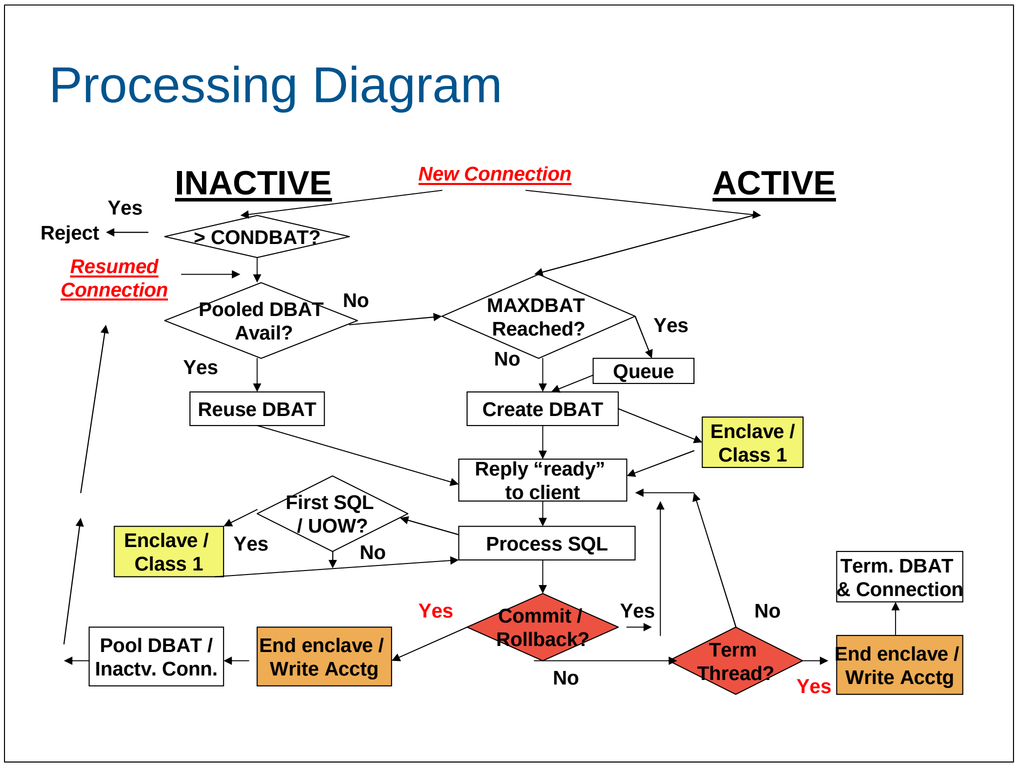
15

GoFurther

INACTIVE mode
- No end user “think time” included
  - DDF creates the DBAT, and responds to the client that it is ready to accept work.
- The enclave is created only when the first SQL of a UOW is received
- The enclave is deleted at thread completion (and accounting is ready to write)
- A new enclave is created and reclassified for each new UOW
  - May have different workstation IDs and be classified differently
  - True also for KEEPDYNAMIC(YES)
- Class 1 times run from first SQL to commit/rollback
  - Can use multi-period response time or velocity goals

Note that if the identifiers are changed before the commit, the accounting record will contain those values, not those used for the classification.

ACTIVE mode
- End user “think time” is included
- Enclave is created once the DBAT is created
  - Before replying to the client and receiving the first SQL
- Enclave is only deleted at thread termination (not at commit)
  - Class 1 times run from thread creation to termination
- Classification remains unchanged for the life of the thread
- Can only use a single-period velocity goal



Processing Diagram

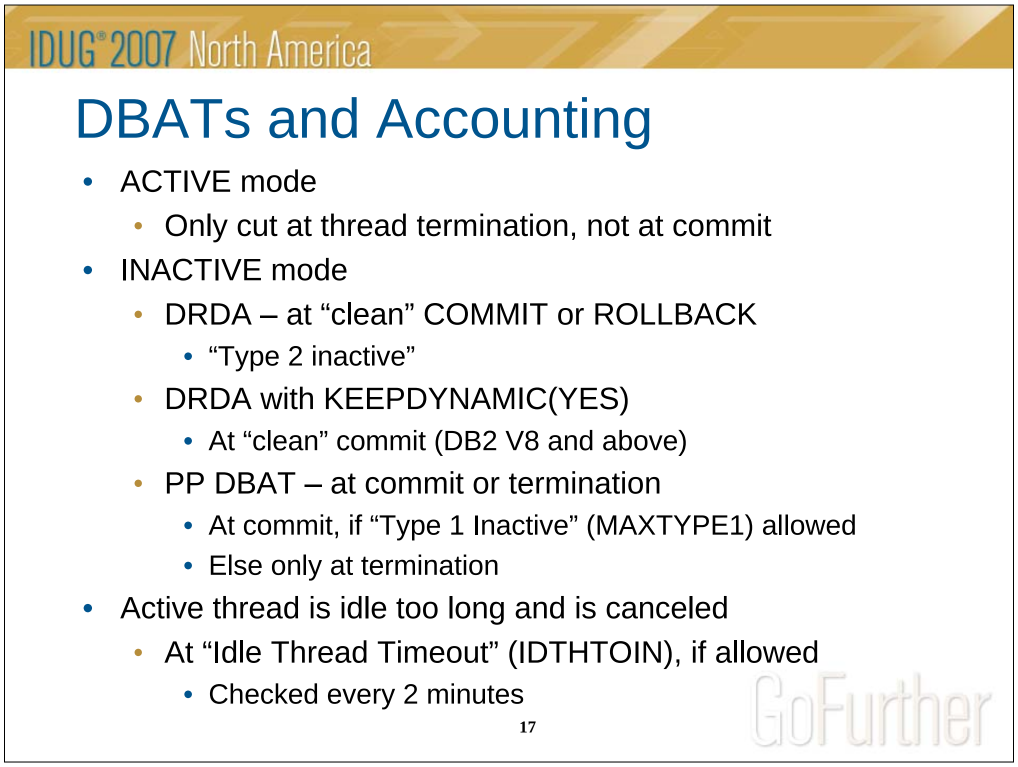
This processing diagram attempts to clarify the different processing flows for INACTIVE and ACTIVE modes. Here you can see the basic differences caused by the use of pooled threads and inactive connections (INACTIVE), vs. the creation and termination of a DBAT per connection request (ACTIVE). The following slides expand on the accounting considerations.

The diagram does not include the different paths for KEEPDYNAMIC(YES) or Private Protocol, although they are described in some of the later slides. In short:
- KEEPDYNAMIC ends the enclave at commit, but keeps the DBAT
- Private Protocol supports type 1 inactive processing

Although inactive is recommended and you may only be using that option, this can help in understanding other documentation references.

Here is a short summary of ZPARM differences.

CMTSTAT	INACTIVE / ACTIVE
POOLINAC	Used / not applicable
IDTHTOIN	Used / not recommended
CONDBAT:MAXDBAT	> / =


IDUG® 2007 North America

DBATs and Accounting

- ACTIVE mode
  - Only cut at thread termination, not at commit
- INACTIVE mode
  - DRDA – at “clean” COMMIT or ROLLBACK
    - “Type 2 inactive”
  - DRDA with KEEPDYNAMIC(YES)
    - At “clean” commit (DB2 V8 and above)
  - PP DBAT – at commit or termination
    - At commit, if “Type 1 Inactive” (MAXTYPE1) allowed
    - Else only at termination
- Active thread is idle too long and is canceled
  - At “Idle Thread Timeout” (IDTHTOIN), if allowed
    - Checked every 2 minutes

17

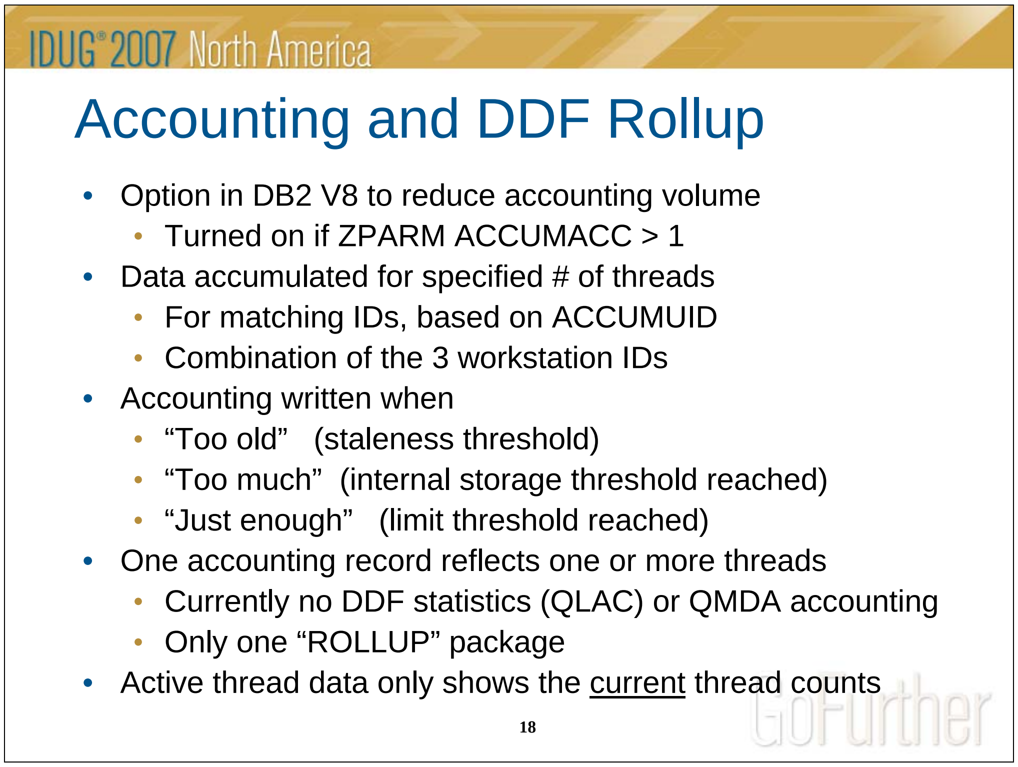
GoFurther

Accounting considerations are somewhat different for DBATs. The first difference is how the decision is made of when an accounting record is cut. And that depends. It is concurrent with the deletion of the enclave.

ACTIVE mode
- Only at thread termination

INACTIVE mode
- DRDA DBAT at “clean” COMMIT or ROLLBACK
- DRDA DBAT at “clean” COMMIT plus KEEPDYNAMIC(YES)
  - Only DB2 V8 and above
- PP DBAT at “clean” COMMIT or ROLLBACK
  - But only at thread termination if it can’t become type 1 inactive (MAXTYPE1 = 0 or limit reached)

<u>Active thread</u> is idle too long and is canceled to free resources
- “Idle Thread Timeout” (IDTHTOIN) limit is checked every two minutes
- Not recommended for ACTIVE mode


IDUG® 2007 North America

Accounting and DDF Rollup

- Option in DB2 V8 to reduce accounting volume
  - Turned on if ZPARM ACCUMACC > 1
- Data accumulated for specified # of threads
  - For matching IDs, based on ACCUMUID
  - Combination of the 3 workstation IDs
- Accounting written when
  - “Too old”   (staleness threshold)
  - “Too much” (internal storage threshold reached)
  - “Just enough”   (limit threshold reached)
- One accounting record reflects one or more threads
  - Currently no DDF statistics (QLAC) or QMDA accounting
  - Only one “ROLLUP” package
- Active thread data only shows the current thread counts

18

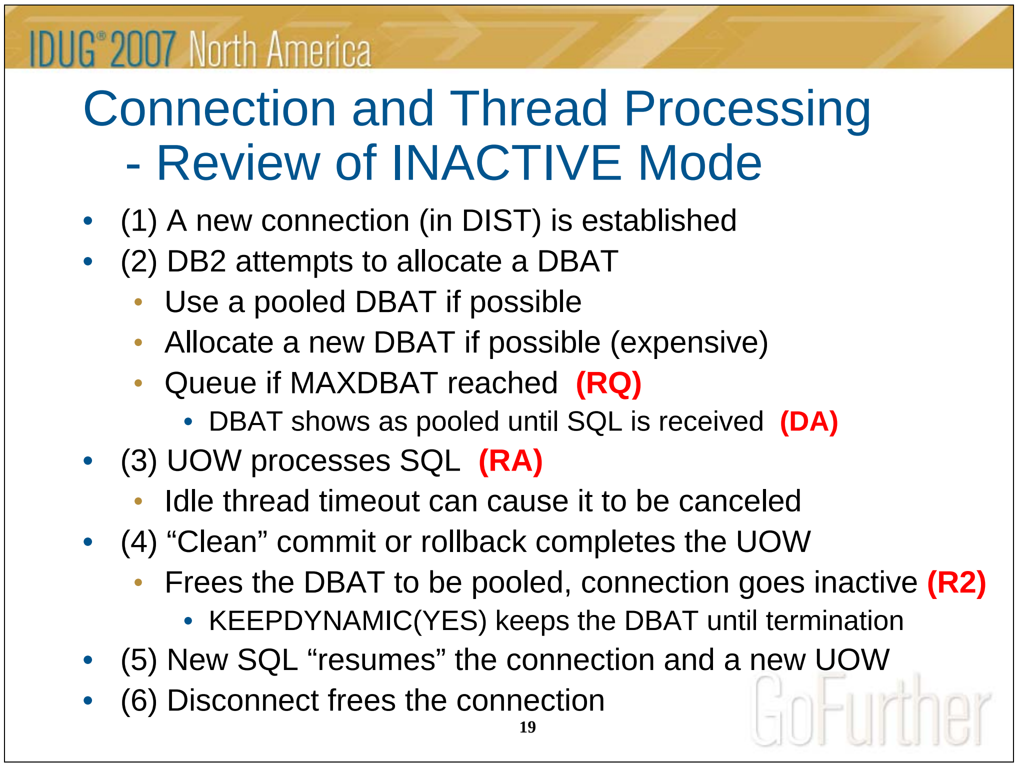
GoFurther

Often too many accounting records are cut for DDF workloads. DB2 V8 offers a way to reduce the volume. Accounting rollup is turned on if ZPARM ACCUMACC > 1. Accounting data is accumulated for a specified number of threads, only for matching threads based on ACCUMUID, which defines some combination of the 3 workstation IDs. An accounting record is written:
- When an internal staleness threshold is reached
  - TIME (too old)
- When an internal storage threshold is reached
  - STORAGE (too much)
- When specified number of threads is reached
  - LIMIT (just enough)

The accounting data should reflect that multiple threads are included. Also, averages per thread should be calculated by including the counts of the rolled up threads.

Currently there are limitations in the rollup accounting records. No QLAC data (DDF statistics) or QMDA information (platform and the client accounting string) are included! Also, only one package section is used for rollup data. Since data from multiple packages may be summarized, it is usually identified as “ROLLUP”, instead of a package name.

➔ When viewing active threads (in INACTIVE mode) with a performance monitor like MAINVIEW for DB2, only the data for the active UOW commit scope is shown.


IDUG® 2007 North America

Connection and Thread Processing - Review of INACTIVE Mode

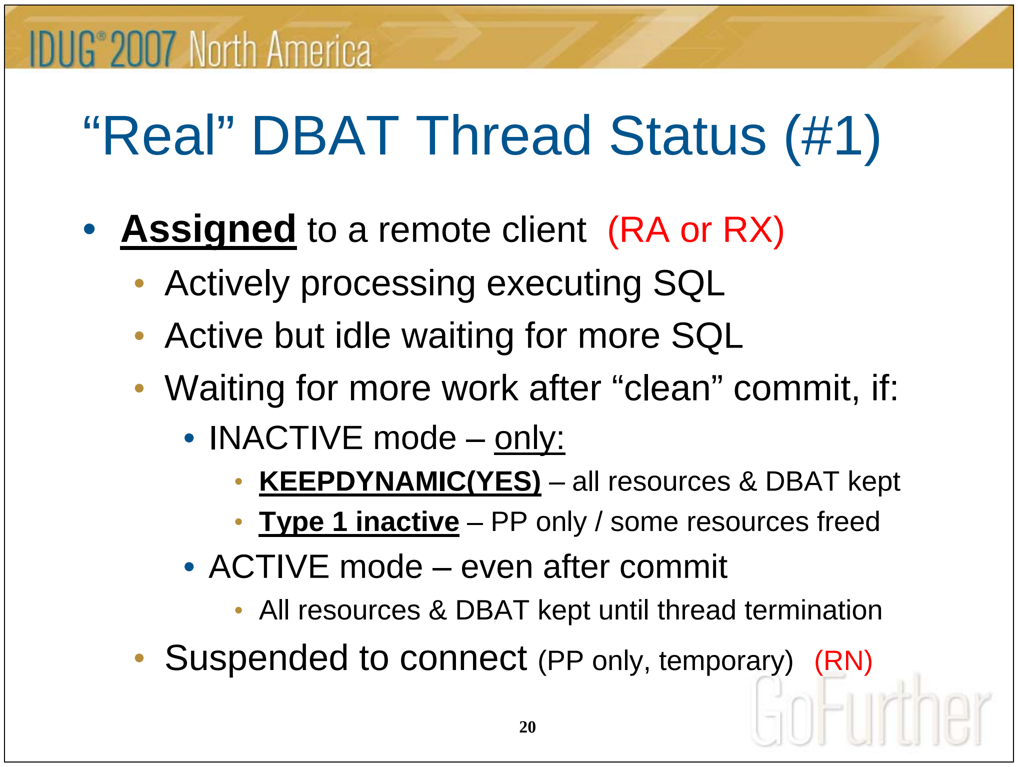
- (1) A new connection (in DIST) is established
- (2) DB2 attempts to allocate a DBAT
  - Use a pooled DBAT if possible
  - Allocate a new DBAT if possible (expensive)
  - Queue if MAXDBAT reached **(RQ)**
    - DBAT shows as pooled until SQL is received **(DA)**
- (3) UOW processes SQL **(RA)**
  - Idle thread timeout can cause it to be canceled
- (4) “Clean” commit or rollback completes the UOW
  - Frees the DBAT to be pooled, connection goes inactive **(R2)**
    - KEEPDYNAMIC(YES) keeps the DBAT until termination
- (5) New SQL “resumes” the connection and a new UOW
- (6) Disconnect frees the connection

19

GoFurther

(1) The connection is established when a CONNECT statement is executed by a client, or dynamic SQL is issued from a DB2 requester.
(2) Usually, a pooled DBAT is assigned, but sometimes a new one must be created. In this case, the connection may be queued for a DBAT because MAXDBAT was reached **(RQ).**
Note that when new DBAT is created, it is a pooled DBAT until the first SQL is received from the client.
(3) During the active UOW SQL processing phase, a thread can be canceled if it is idle too long waiting for additional SQL. The threshold is defined with the ZPARM IDTHTOIN.
(4) A commit without restrictions, or a rollback, signals the completion of the UOW. Normally the DBAT is pooled and the connection goes inactive. However, with KEEPDYNAMIC(YES) in DB2 V8, the UOW completion can still process (and delete the enclave and cut accounting data) without freeing the DBAT and the prepared SQL in the local cache.

(5) Once established, the connection is kept for reuse while needed. New SQL from that client “resumes” the connection and starts a new UOW. Often, the same (pooled) DBAT can be reused.
(6) The connection usually disappears when the client issues a DISCONNECT, but can be freed by DB2 Connect at commit. Use of the TCPKPALV ZPARM is recommended to avoid hangs caused by a network failure.


IDUG® 2007 North America

"Real" DBAT Thread Status (#1)

- **<u>Assigned</u>** to a remote client (RA or RX)
  - Actively processing executing SQL
  - Active but idle waiting for more SQL
  - Waiting for more work after "clean" commit, if:
    - INACTIVE mode – <u>only:</u>
      - **<u>KEEPDYNAMIC(YES)</u>** – all resources & DBAT kept
      - **<u>Type 1 inactive</u>** – PP only / some resources freed
    - ACTIVE mode – even after commit
      - All resources & DBAT kept until thread termination
  - Suspended to connect (PP only, temporary) (RN)

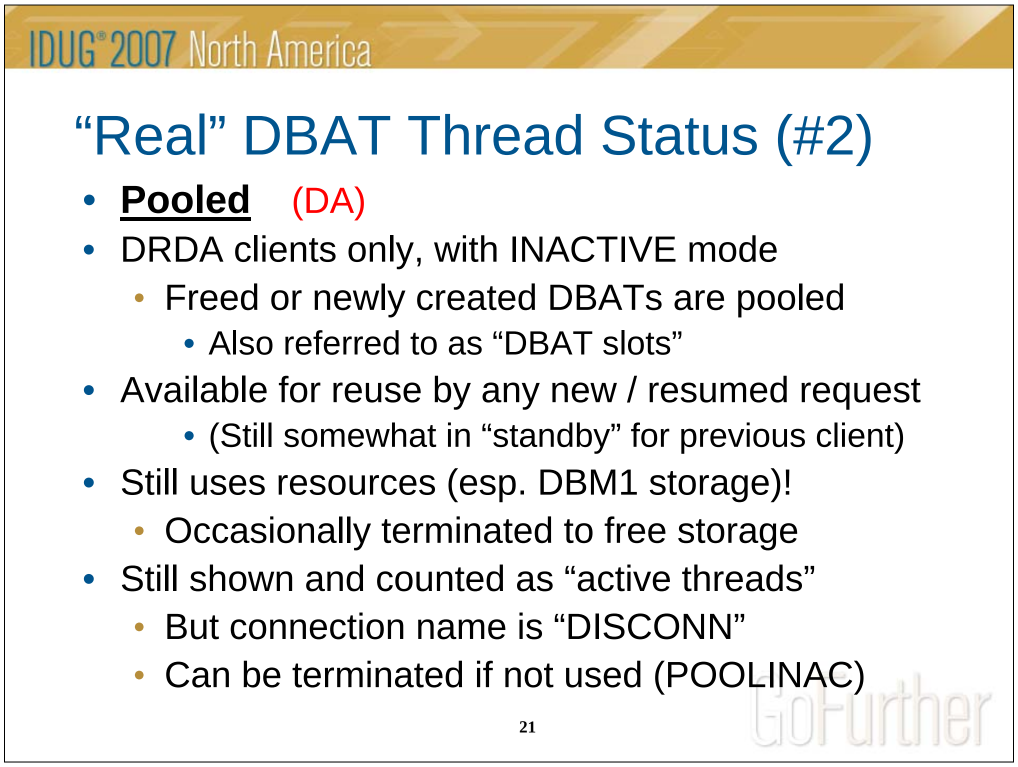
20

GoFurther

You have likely often read about active and inactive DBATs, and seen various status values in DISPLAY THREAD output or monitor displays. But it may help to consider that there are really only two conditions for a DBAT – assigned to a requester, or pooled for reuse.
Note: The highlighted codes in parentheses are the status values in DISPLAY THREAD TYPE(ACTIVE)

<u>Assigned</u> to a remote client connection

- Thread is actively processing to retrieve DB2 data **(RA)**
- Thread is active but idle waiting for more SQL
  (Perhaps a commit was issued with held cursors or global temp tables)
- Thread is waiting for more work after a clean commit. This can only occur if:
  - INACTIVE mode – only two conditions
    - <u>(KEEPDYNAMIC(YES)</u> - All resources and DBAT are still kept
    - <u>Type 1 Inactive</u> (PP only) - some resources freed
  - ACTIVE mode – even after commit
    - All resources and DBAT are still kept until thread termination
- Suspended to connect ("hop") to another DB2 **(RN)**
  (This is a rare and temporary condition for Private Protocol only.
  It can occur only on the first connection to a remote location
  and is to establish DB2 to DB2 system conversations)


IDUG® 2007 North America

"Real" DBAT Thread Status (#2)

- **<u>Pooled</u>** (DA)
- DRDA clients only, with INACTIVE mode
  - Freed or newly created DBATs are pooled
    - Also referred to as "DBAT slots"
- Available for reuse by any new / resumed request
    - (Still somewhat in "standby" for previous client)
- Still uses resources (esp. DBM1 storage)!
  - Occasionally terminated to free storage
- Still shown and counted as "active threads"
  - But connection name is "DISCONN"
  - Can be terminated if not used (POOLINAC)

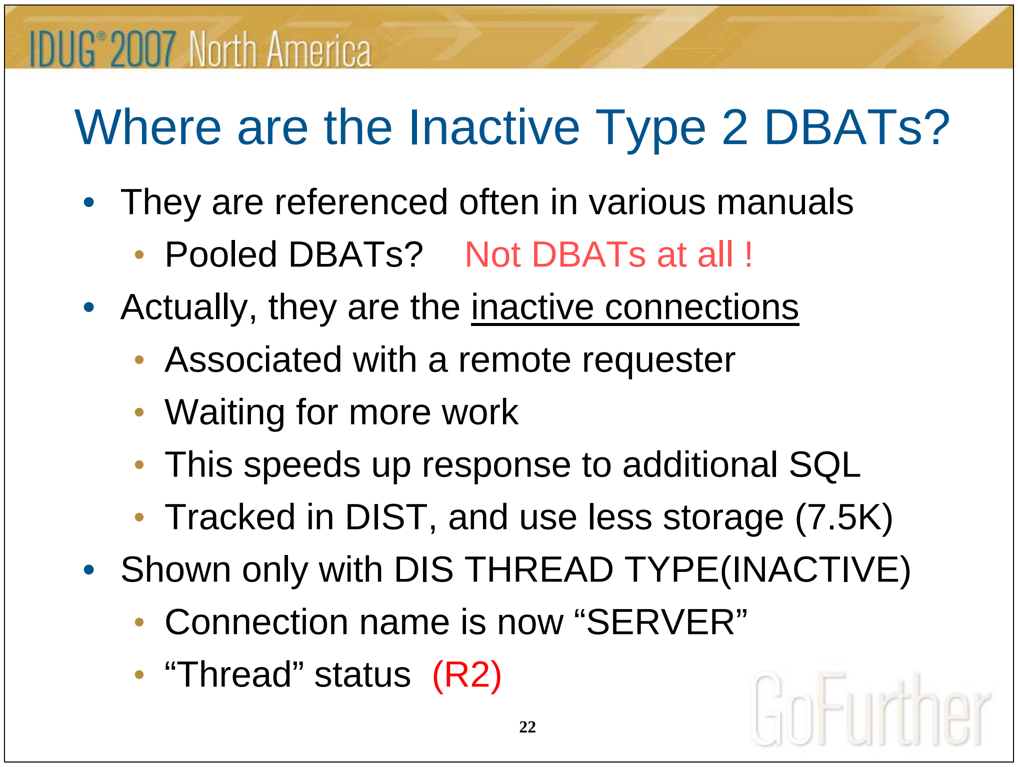
21

GoFurther

<u>Pooled</u>
With INACTIVE mode, a DRDA thread is pooled for reuse by any other requester when a UOW is completed with a commit without restrictions. A newly created DBAT is pooled until SQL is received. The status of a pooled DBAT is shown as DA. It is also referred to as a "DBAT slot".
<u>Restrictions at commit</u> that stop pooling:
Some are temporary, until the resources are freed and another commit is issued (or idle thread timeout cancels it)
- Open cursors WITH HOLD, or held LOB locators
  (WITH HOLD is the default in ODBC/CLI/JDBC, change in db2cli.ini)
- Declared global temporary tables exist

KEEPDYNAMIC(YES) is a special case (already described)

A pooled DBAT does still hold resources. The most critical is DBM1 storage. Each uses around 200K in DBM1 as a minimum, but it can be much more. DB2 does free up a DBAT to clean up this storage after a certain number of uses (I've heard 50, and read 200). A pooled thread is not officially associated with a remote connection, but is somewhat in "standby" for the original client to submit additional SQL.
Pooled threads are still shown as active threads with - DIS THREAD(*) LOCATION(*) TYPE(ACTIVE). The connection name is "DISCONN". (Note: No AUTHID but most other IDs are kept, no times or counts.) They are also counted in "Current Active DBATs" and used in checks against MAXDBAT. They can be terminated to free storage if inactive too long. This is controlled by the ZPARM POOLINAC and is checked every 3 minutes. This helps to maintain a "steady state" of "useful" DBATs.


IDUG® 2007 North America

Where are the Inactive Type 2 DBATs?

- They are referenced often in various manuals
  - Pooled DBATs?   Not DBATs at all !
- Actually, they are the inactive connections
  - Associated with a remote requester
  - Waiting for more work
  - This speeds up response to additional SQL
  - Tracked in DIST, and use less storage (7.5K)
- Shown only with DIS THREAD TYPE(INACTIVE)
  - Connection name is now "SERVER"
  - "Thread" status (R2)

22

GoFurther

Where are the Inactive Type 2 DBATs? They are still referenced often in documentation and reports, and are sometimes related to pooled DBATs. However, they are not DBATs at all. They have been disassociated from a recently used pooled DBAT (mostly). No real thread exists (no ACE), and no storage is being used in DBM1.

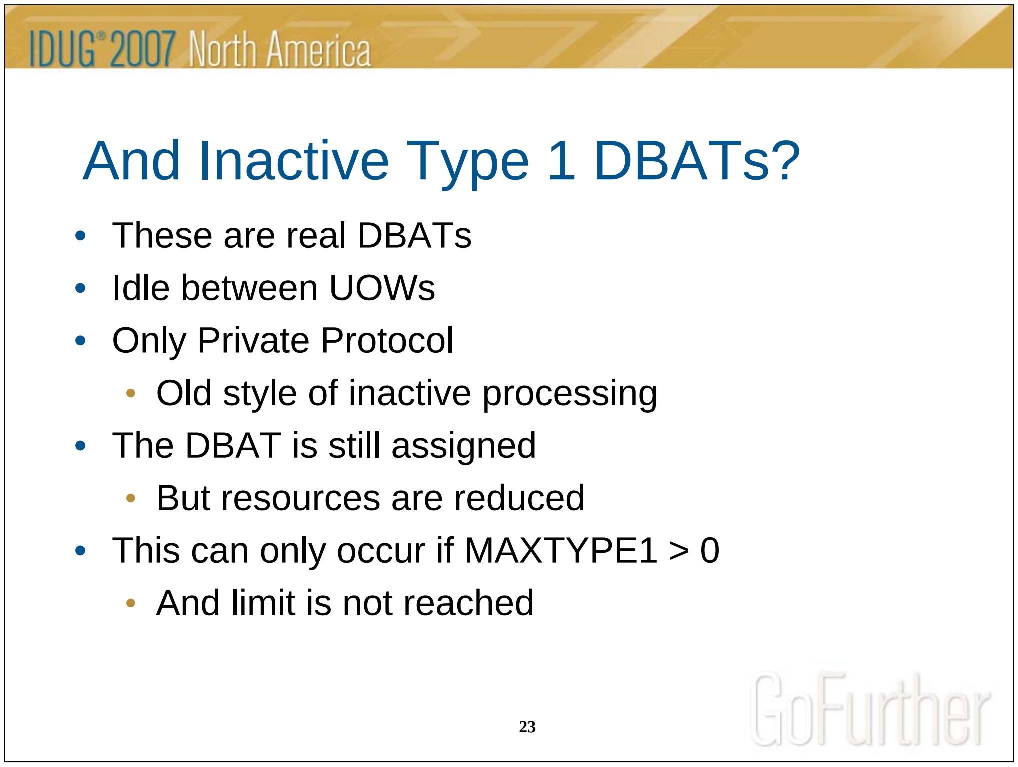
Note: The terms active / inactive, idle / pooled, and active / idle are not used consistently. For example, I enjoyed this sentence: "Once an active DBAT has executed a transaction, it stays active in the pool of idle threads, waiting to be reused."   This described a pooled DBAT – not wrong, but confusing.

Inactive type 2 DBATs are actually inactive connections that are associated with the remote requester, waiting for more SQL. Keeping this information speeds up response to the remote clients because they hold authentication and network controller information related to the connection. They hold no database access information (or "database stateful context").
They are tracked in the DIST address space, and use much less storage than a DBAT (approx. 7.5K). If looked at with a performance monitor, the connection name is "SERVER", all remote IDs are kept, but no times or counts are present.

However, they are displayed with

```
   DISPLAY THREAD(*) LOCATION(*) TYPE(INACTIVE)
      - but  not shown with TYPE(*)!
```

Obviously, this is a hold-over from private protocol and inactive type 1 DBATs.


IDUG® 2007 North America

And Inactive Type 1 DBATs?

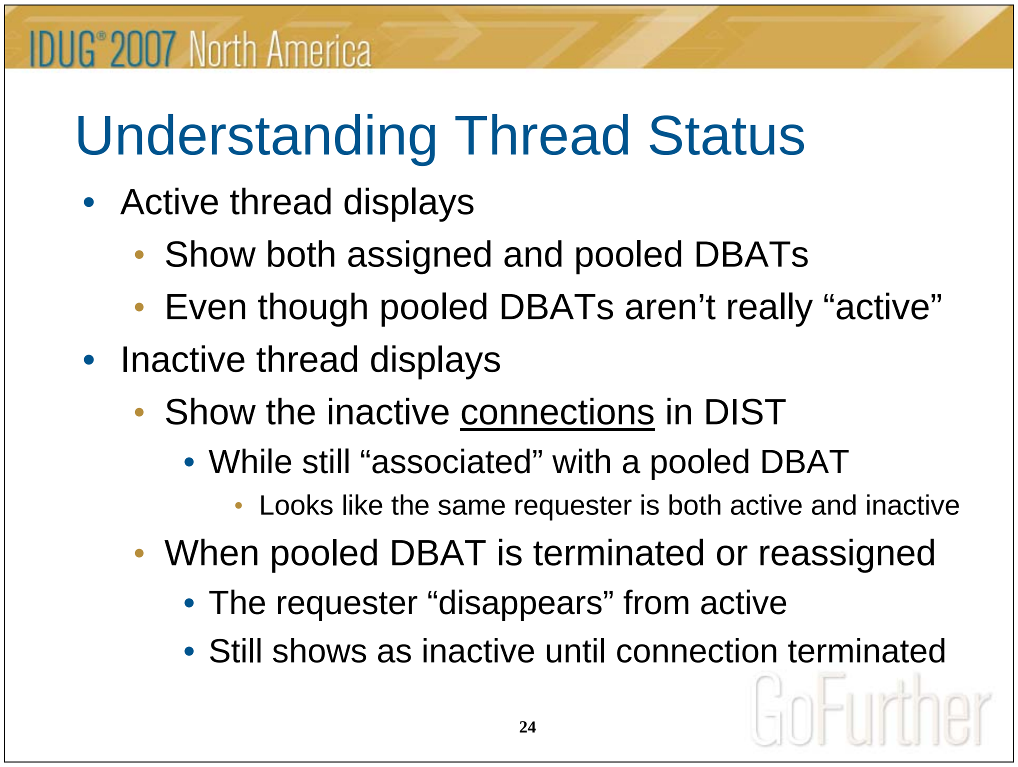
- These are real DBATs
- Idle between UOWs
- Only Private Protocol
  - Old style of inactive processing
- The DBAT is still assigned
  - But resources are reduced
- This can only occur if MAXTYPE1 > 0
  - And limit is not reached

23

GoFurther

Inactive Type 1 DBATs are real DBATs and still hold resources. They are threads that are idle between UOWs – in other words, a "clean" commit has been issued. This type of processing is only valid for Private Protocol, and is essentially the first version of inactive processing before DBAT pooling was implemented. Although the DBAT is still in use, as many resources as possible are released. This saves storage, and reduces conflicts with active threads. This process is sometimes referred to as "cleansing".

This kind of inactive processing only occurs if the ZPARM MAXTYPE1 is defined as greater than zero. Since only the number defined here is allowed to go Type 1 inactive, it is recommended to specify enough threads in MAXTYPE1 to be able to handle the maximum number of concurrent Private Protocol accesses.


IDUG® 2007 North America

Understanding Thread Status

- Active thread displays
  - Show both assigned and pooled DBATs
  - Even though pooled DBATs aren't really "active"
- Inactive thread displays
  - Show the inactive <u>connections</u> in DIST
    - While still "associated" with a pooled DBAT
      - Looks like the same requester is both active and inactive
  - When pooled DBAT is terminated or reassigned
    - The requester "disappears" from active
    - Still shows as inactive until connection terminated

24

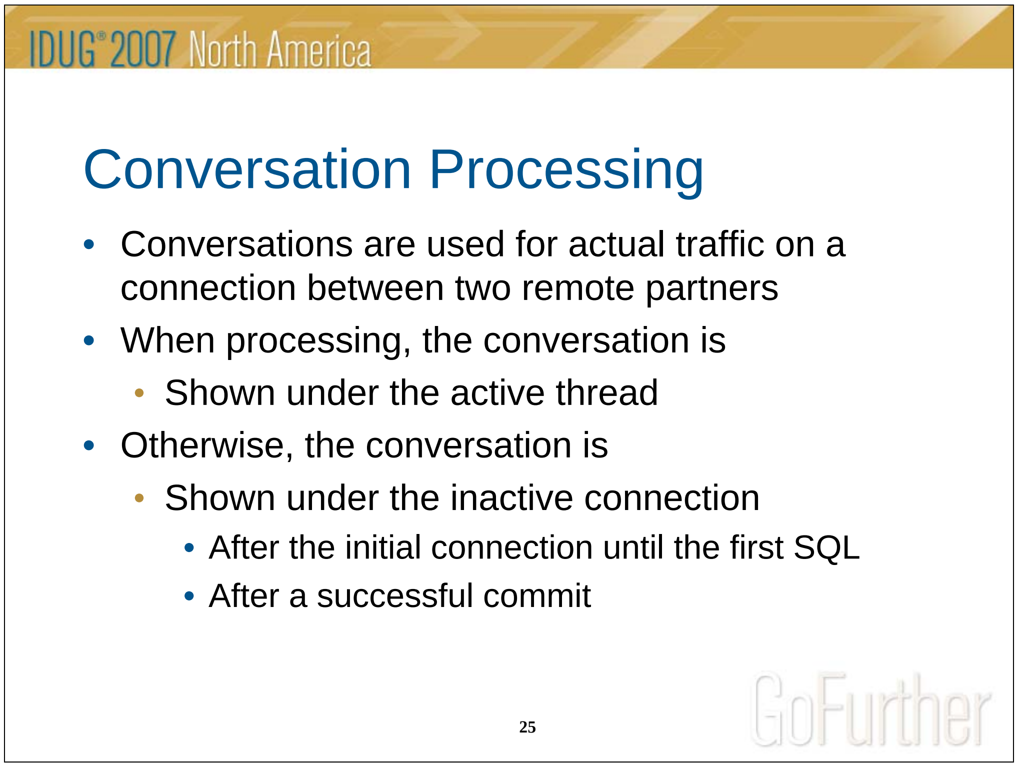
GoFurther

Now lets look at the practical side. What do you see when looking at distributed workloads? Understanding thread status can be somewhat confusing.

- Assigned <u>and</u> pooled threads are shown as <u>active threads</u>
  - Even though pooled threads aren't really active or assigned to a client
  - Also, the pooled thread still has many IDs from the last user

- It looks like pooled threads are also shown as <u>inactive threads</u>
  - But these are actually the inactive connections in DIST

- If a pooled DBAT is terminated or reassigned to a different client
  - The requester disappears from active threads
  - Still shown as inactive until the client terminates the connection.



IDUG® 2007 North America

Conversation Processing

- Conversations are used for actual traffic on a connection between two remote partners
- When processing, the conversation is
  - Shown under the active thread
- Otherwise, the conversation is
  - Shown under the inactive connection
    - After the initial connection until the first SQL
    - After a successful commit

25

GoFurther

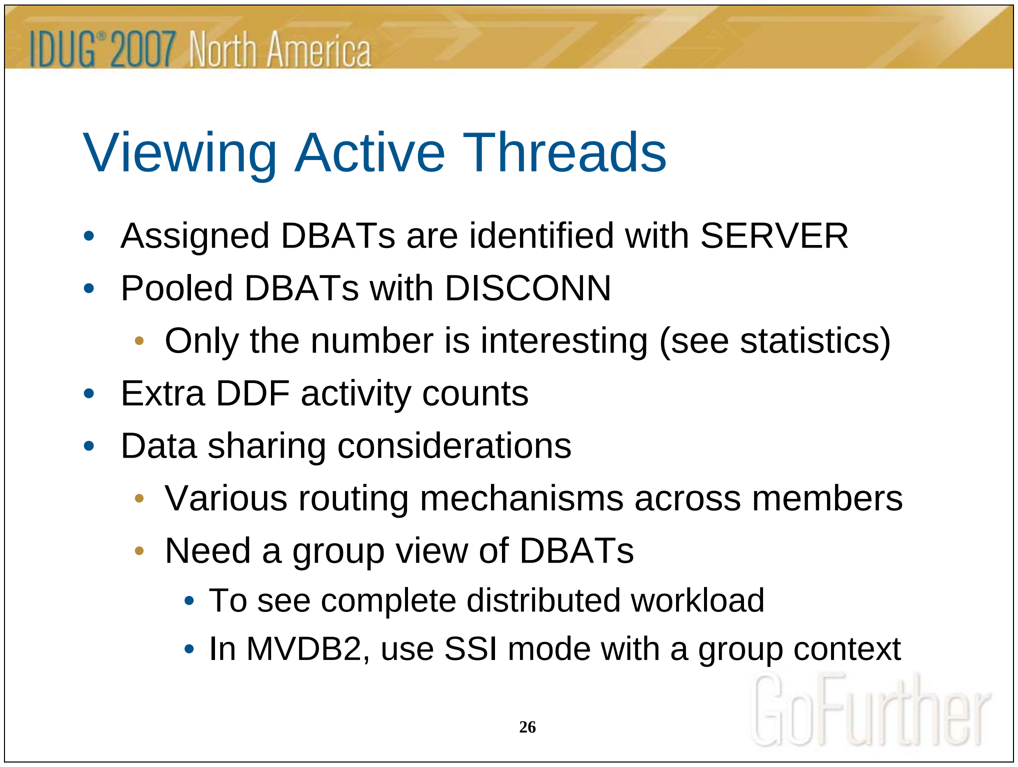
Conversations are used for actual data flows on a connection between remote requester / server partners.

When processing, the conversation is shown under the active thread

When the connection is not active, the conversation is shown under the inactive connection, or "inactive thread". This is true after the initial connection before the first SQL is received, or after a successful commit that has pooled the DBAT.

Here is an example of a DBAT that has just committed its UOW.

```
DISPLAY ACTIVE
DSNV402I  *DECE ACTIVE THREADS -
NAME       ST A   REQ ID             AUTHID   PLAN       ASID  TOKEN
DISCONN  DA *      3  NONE         NONE     DISTSERV 00CA 13491
 V471-USBMCN01.LUDECE.C018343A2C5F=13491

DISPLAY INACTIVE
DSNV424I  *DECE INACTIVE THREADS -
NAME       ST A   REQ ID              AUTHID    PLAN       ASID TOKEN
SERVER   R2       0    db2bp.exe    BOLJXO1  DISTSERV 00CA 13502
 V437-WORKSTATION=joott-SJC-70  USERID=boljxo1
      APPLICATION NAME=db2bp.exe
 V445-AC173AFE.F70D.01F581235657=13502 ACCESSING DATA FOR
      172.23.58.254
 V447--LOCATION         SESSID             A  ST TIME
 V448--172.23.58.254    7623:3575        W R2 0703216095818
```


IDUG® 2007 North America

Viewing Active Threads

- Assigned DBATs are identified with SERVER
- Pooled DBATs with DISCONN
  - Only the number is interesting (see statistics)
- Extra DDF activity counts
- Data sharing considerations
  - Various routing mechanisms across members
  - Need a group view of DBATs
    - To see complete distributed workload
    - In MVDB2, use SSI mode with a group context

26

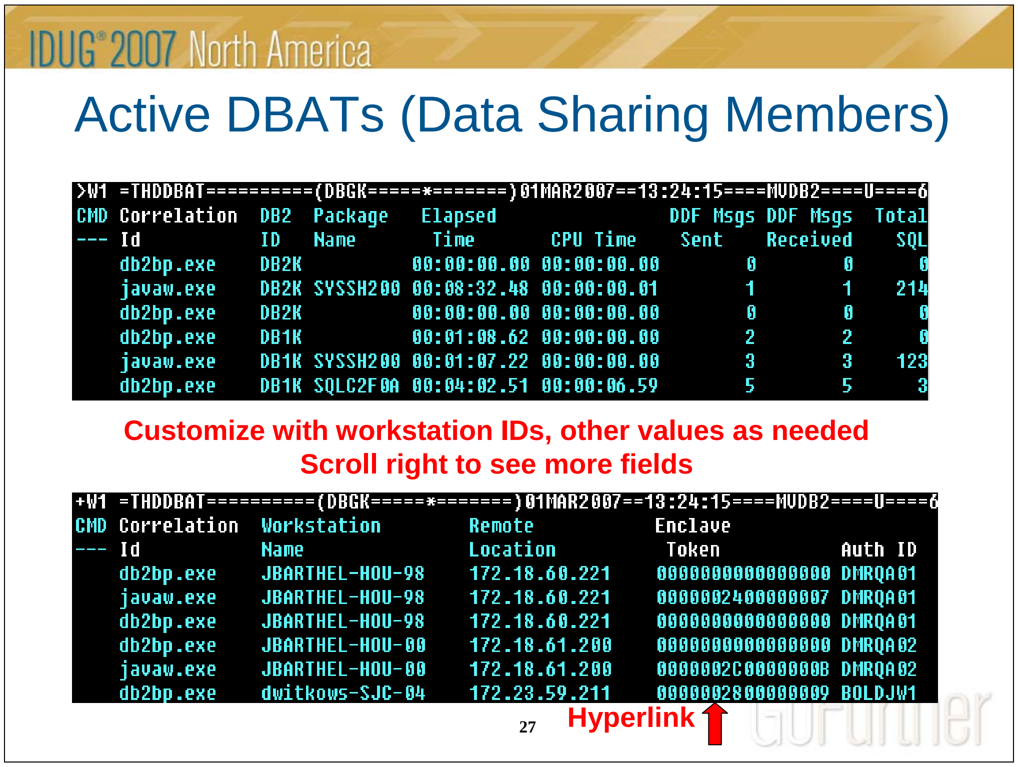
GoFurther

When viewing active threads, the first step is to consider only those that are real threads. In general, you can ignore DISCONN pooled threads. Only the number available for new work is of interest. That is better seen in the global statistics.

Check for server activity. All DBATs will have a DDF section that shows client activity with this DB2 for z/OS server.

Requester activity is only available for allied threads or DBATs that make distributed requests. In this case, there will be a DDF section per location to show activity by this DB2 to get data from other DB2s – z/OS or LUW

There are special data sharing considerations, since various routing mechanisms and WLM can be used to distribute incoming workload across several or all members of a data sharing group. This leads to the first issue – of where to find the active DBATs. And you may also want to compare DDF statistics about resource usage and workload balancing across the group. In MAINVIEW, Single System Image (SSI) contexts allow <u>any</u> tabular view to be invoked to show data from many DB2s. It is a simple setup step to define such a context for a data sharing group, and that provides a simple way to look at distributed activity across the group, or even a subset of the members.


IDUG® 2007 North America

Active DBATs (Data Sharing Members)

```
>W1 =THDDBAT==========(DBGK=====*=======)01MAR2007==13:24:15====MVDB2====U====6
CMD Correlation  DB2  Package   Elapsed                   DDF Msgs DDF Msgs  Total
--- Id           ID   Name       Time        CPU Time      Sent    Received    SQL
    db2bp.exe    DB2K          00:00:00.00 00:00:00.00        0         0        0
    javaw.exe    DB2K SYSSH200 00:08:32.48 00:00:00.01        1         1      214
    db2bp.exe    DB2K          00:00:00.00 00:00:00.00        0         0        0
    db2bp.exe    DB1K          00:01:08.62 00:00:00.00        2         2        0
    javaw.exe    DB1K SYSSH200 00:01:07.22 00:00:00.00        3         3      123
    db2bp.exe    DB1K SQLC2F0A 00:04:02.51 00:00:06.59        5         5        3
```

Customize with workstation IDs, other values as needed
Scroll right to see more fields

```
+W1 =THDDBAT==========(DBGK=====*=======)01MAR2007==13:24:15====MVDB2====U====6
CMD Correlation  Workstation       Remote            Enclave
--- Id           Name              Location           Token            Auth ID
    db2bp.exe    JBARTHEL-HOU-98   172.18.60.221     0000000000000000 DMRQA01
    javaw.exe    JBARTHEL-HOU-98   172.18.60.221     0000002400000007 DMRQA01
    db2bp.exe    JBARTHEL-HOU-98   172.18.60.221     0000000000000000 DMRQA01
    db2bp.exe    JBARTHEL-HOU-00   172.18.61.200     0000000000000000 DMRQA02
    javaw.exe    JBARTHEL-HOU-00   172.18.61.200     0000002C0000000B DMRQA02
    db2bp.exe    dwitkows-SJC-04   172.23.59.211     0000002800000009 BOLDJW1
```

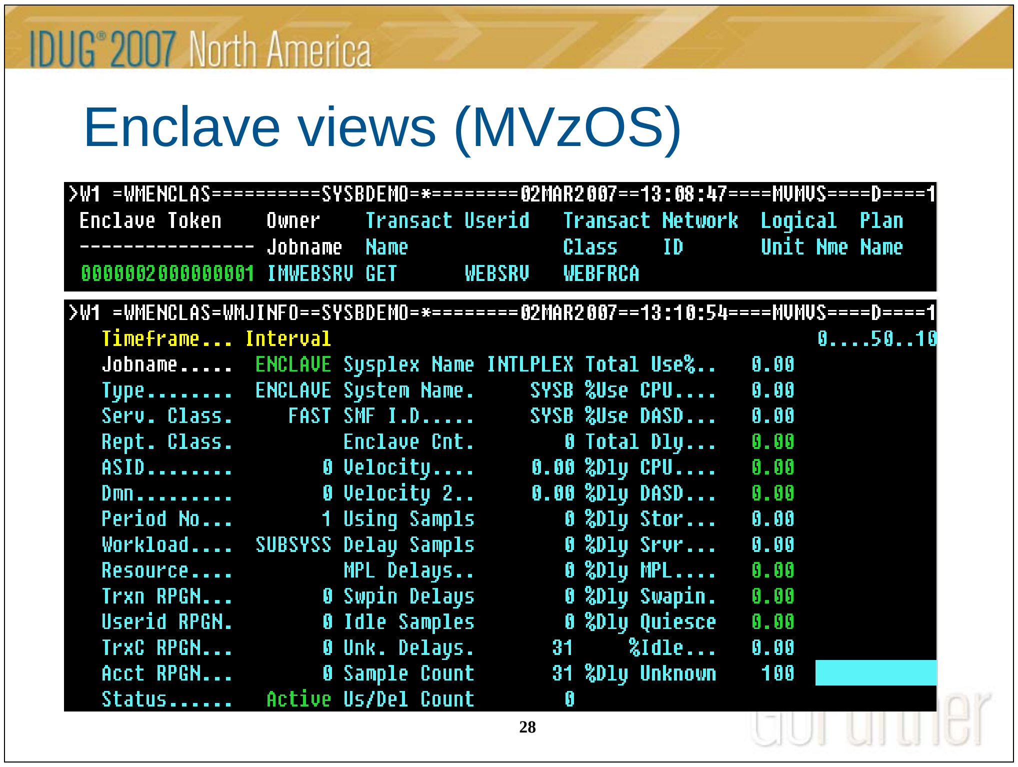
27 **Hyperlink**

GoFurther

Here is a sample view of active and pooled DBATs from a data sharing group (SSI context – DBGK). There are threads from both DB1K and DB2K. There are many fields that you can choose to customize the perfect view for your specific workloads. All the identifiers are available, including the three workstation IDs. Besides the measurement values shown here, there are many others as well.

A few valuable ones for checking the status of a DBAT are
- Last timestamp
- Conversation state (receiving, sending)
- Last activity (send or receive)

The enclave token provides a direct hyperlink to the WMENCLVZ view for detailed information of this specific enclave in MAINVIEW for z/OS.


IDUG® 2007 North America

Enclave views (MVzOS)

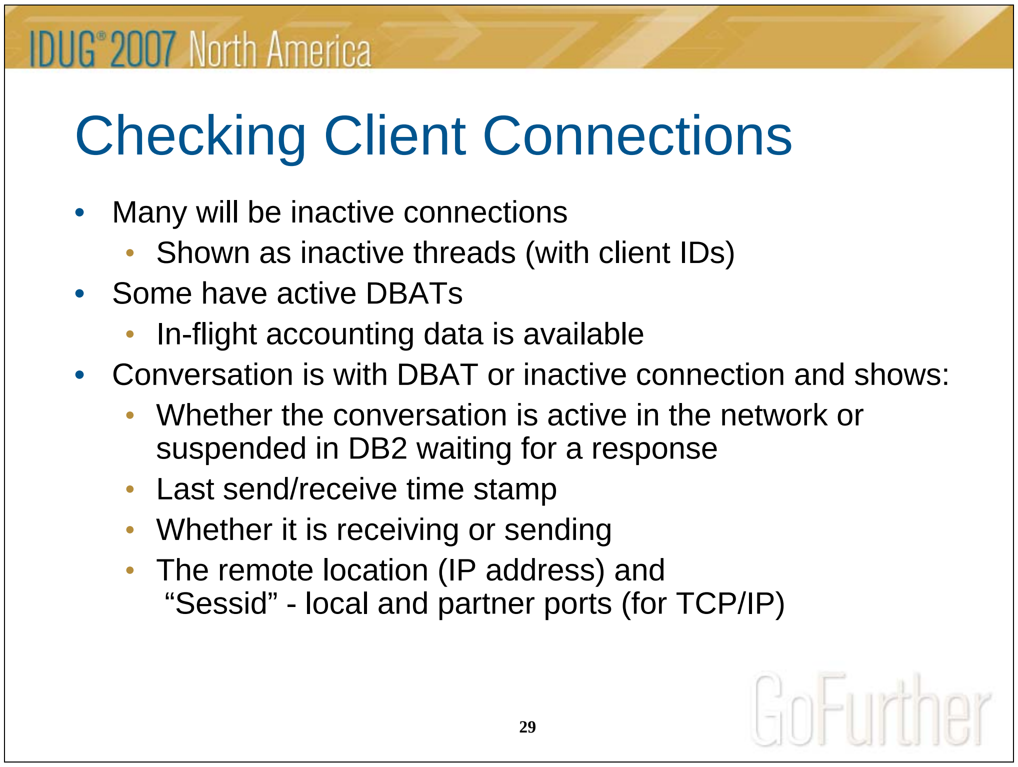
```
>W1 =WMENCLAS==========SYSBDEMO=*========02MAR2007==13:08:47====MVMVS====D====1
 Enclave Token     Owner    Transact Userid   Transact Network  Logical  Plan
 ---------------- Jobname  Name               Class    ID       Unit Nme Name
 0000002000000001 IMWEBSRV GET       WEBSRV   WEBFRCA
```

```
>W1 =WMENCLAS=WMJINFO==SYSBDEMO=*========02MAR2007==13:10:54====MVMVS====D====1
   Timeframe... Interval                                           0....50..10
   Jobname..... ENCLAVE Sysplex Name INTLPLEX Total Use%..   0.00
   Type........ ENCLAVE System Name.     SYSB %Use CPU....   0.00
   Serv. Class.    FAST SMF I.D.....     SYSB %Use DASD...   0.00
   Rept. Class.         Enclave Cnt.        0 Total Dly...   0.00
   ASID........       0 Velocity....     0.00 %Dly CPU....   0.00
   Dmn.........       0 Velocity 2..     0.00 %Dly DASD...   0.00
   Period No...       1 Using Sampls        0 %Dly Stor...   0.00
   Workload.... SUBSYSS Delay Sampls        0 %Dly Srvr...   0.00
   Resource....         MPL Delays..        0 %Dly MPL....   0.00
   Trxn RPGN...       0 Swpin Delays        0 %Dly Swapin.   0.00
   Userid RPGN.       0 Idle Samples        0 %Dly Quiesce   0.00
   TrxC RPGN...       0 Unk. Delays.       31     %Idle...   0.00
   Acct RPGN...       0 Sample Count       31 %Dly Unknown    100
   Status......  Active Us/Del Count        0
```

28

Here are some sample views of the online enclave information (although not for a DDF thread enclave in this case). You can directly access this same information for an active DBAT thread with the hyperlink to MAINVIEW for z/OS. Note especially the delay percents.

You can also use the SDSF ENClave command to see some activity data. Use the “i” line command to see the classification attributes for an enclave.


IDUG® 2007 North America

Checking Client Connections

- Many will be inactive connections
  - Shown as inactive threads (with client IDs)
- Some have active DBATs
  - In-flight accounting data is available
- Conversation is with DBAT or inactive connection and shows:
  - Whether the conversation is active in the network or suspended in DB2 waiting for a response
  - Last send/receive time stamp
  - Whether it is receiving or sending
  - The remote location (IP address) and "Sessid" - local and partner ports (for TCP/IP)

29

GoFurther

Client connection information remains in DB2 after the first request, as long as there is an expectation that additional work may be submitted by the client. How long is very dependent on the connectivity options in use. For example, DB2 Connect with only connection pooling will keep client information available until the client disconnects, and the equivalent connection is also kept inside DB2 for z/OS. DB2 Connect with the connection concentrator (with assistance from DB2 for z/OS) will terminate a "coordinator agent" at commit / rollback, and this also terminates the DB2 inactive connection. DB2 Connect continues to trace the client with a "logical agent" until the disconnect.

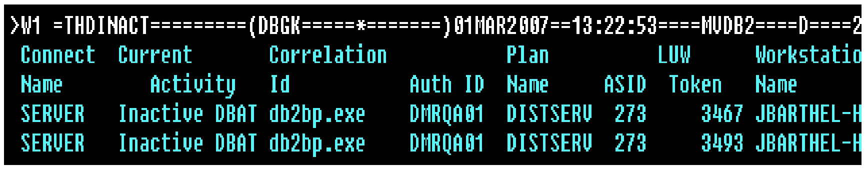
So, many connections are likely inactive, waiting for additional requests (and displayed as inactive threads). Of course, some connections will have an active thread (a DBAT server thread). For these, in-flight accounting data is available to show SQL statements, elapsed and CPU times, and other resource usage. Even with accounting rollup, only the data from the current UOW is shown.

Conversation information is kept either with the active thread, or with the inactive connection. It includes some interesting data on the status of a connection.

The last time stamp can be valuable to determine whether a DBAT is still actively processing or whether it is hung. If the time stamp is not changing, you may need to consider canceling the thread.

IDUG® 2007 North America

Inactive Thread (Connection) View



```
>W1 =THDINACT=========(DBGK=====*=======)01MAR2007==13:22:53====MVDB2====D====2
 Connect  Current       Correlation          Plan          LUW      Workstatio
 Name        Activity   Id           Auth ID  Name     ASID  Token    Name
 SERVER   Inactive DBAT db2bp.exe    DMRQA01  DISTSERV  273      3467 JBARTHEL-H
 SERVER   Inactive DBAT db2bp.exe    DMRQA01  DISTSERV  273      3493 JBARTHEL-H
```

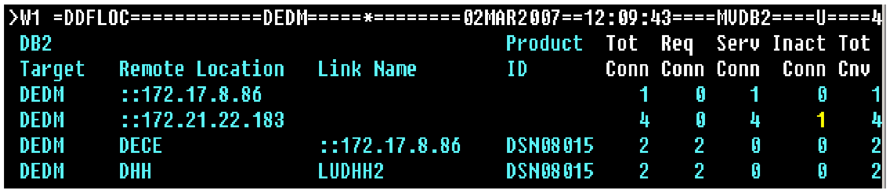
30

GoFurther

Here is a sample THDINACT view of the inactive connections. Not really very interesting at this level, so on to the next views.

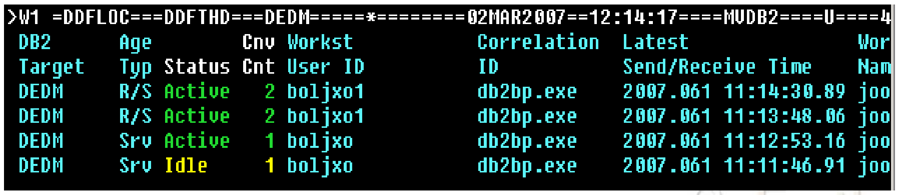
IDUG® 2007 North America

Connection / Conversation Views



```
>W1 =DDFLOC===========DEDM=====*========02MAR2007==12:09:43====MVDB2====U====4
 DB2                                          Product  Tot  Req  Serv Inact Tot
 Target   Remote Location   Link Name         ID       Conn Conn Conn  Conn Cnv
 DEDM     ::172.17.8.86                                   1    0    1     0    1
 DEDM     ::172.21.22.183                                 4    0    4     1    4
 DEDM     DECE              ::172.17.8.86     DSN08015    2    2    0     0    2
 DEDM     DHH               LUDHH2            DSN08015    2    2    0     0    2
```

Hyperlink on "Tot Conn" to see details



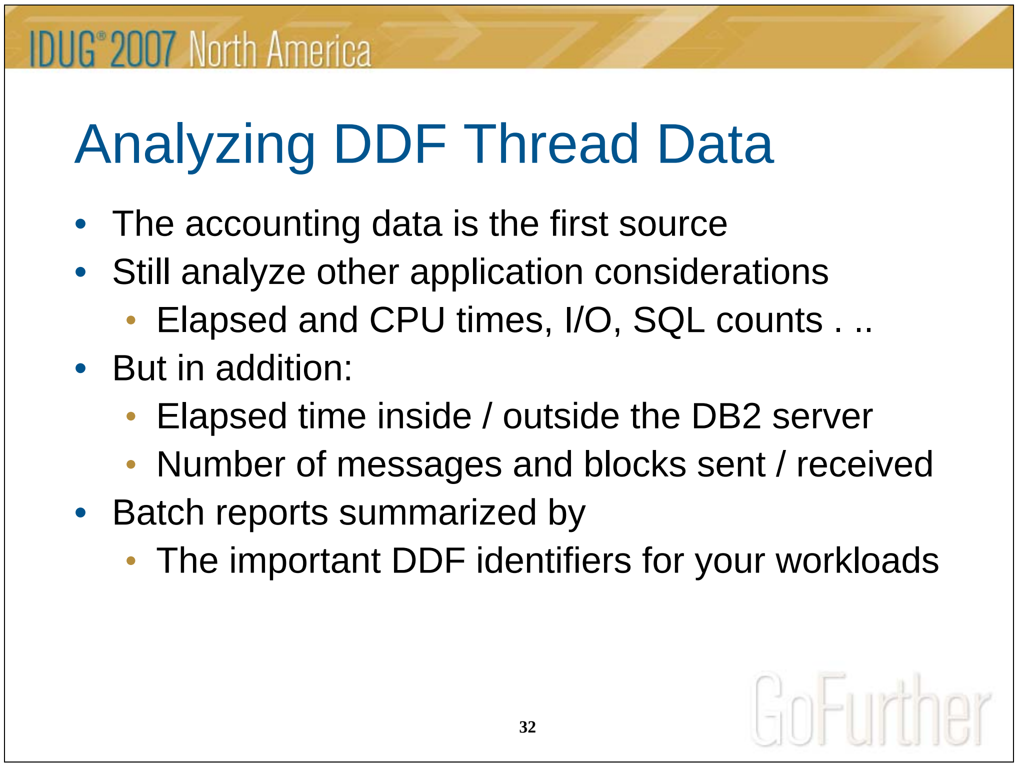
```
>W1 =DDFLOC===DDFTHD===DEDM=====*========02MAR2007==12:14:17====MVDB2====U====4
 DB2      Age        Cnv Workst            Correlation  Latest               Wor
 Target   Typ Status Cnt User ID           ID           Send/Receive Time    Nam
 DEDM     R/S Active   2 boljxo1           db2bp.exe    2007.061 11:14:30.89 joo
 DEDM     R/S Active   2 boljxo1           db2bp.exe    2007.061 11:13:48.06 joo
 DEDM     Srv Active   1 boljxo            db2bp.exe    2007.061 11:12:53.16 joo
 DEDM     Srv Idle     1 boljxo            db2bp.exe    2007.061 11:11:46.91 joo
```

31

GoFurther

This is another way to look at distributed activity, concentrating on the connections and conversations instead of the "threads".

The DDFLOC view shows all the connections to DB2, similar to the DB2 DISPLAY LOCATION command, but with more information on the connection statuses. But here you can sort on the "Total Connections" column to see your high-activity connection concentrators like DB2 Connect EE at the top. When you drill down, you can then see all the active threads and inactive connections – as well as the latest time stamp of each. That makes it much easier to identify whether the network connection itself may be in trouble, or if specific clients have been inactive for a long time.



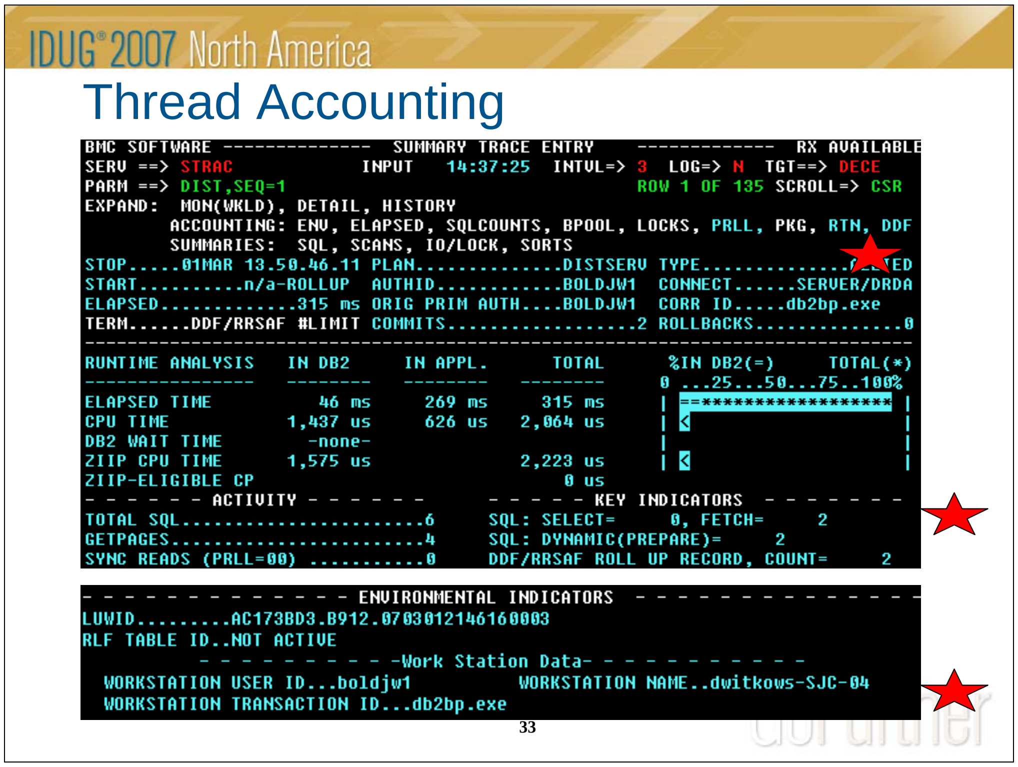
Analyzing DDF Thread Data

- The accounting data is the first source
- Still analyze other application considerations
  - Elapsed and CPU times, I/O, SQL counts . ..
- But in addition:
  - Elapsed time inside / outside the DB2 server
  - Number of messages and blocks sent / received
- Batch reports summarized by
  - The important DDF identifiers for your workloads

When analyzing DDF Threads, all the normal application issues and accounting information still need to be considered. As usual, the primary source of data is the accounting record. You should always first look to see if there are any obvious signs of problems in the basic information, such as elapsed times, waits, possible MVS dispatching issues causing waits for CPU that would show up as "not accounted time", SQL statements executed, I/O, etc.

There is also an extra step you can take when you suspect that a problem may exist outside of DB2, before you start running complicated traces outside DB2. A calculation of "time spent in DB2" subtracted from class 1 elapsed time gives you a "time spent outside DB2". This includes both the application time in the client, and the network time. If high, further investigation may be needed.
Time in DB2 is calculated as
= class 2 non-nested elapsed (time spent by the application in DB2)
+ class 1 nested elapsed for stprocs, triggers, UDFs
(time accounted separately in a separate A.S.)
+ (class 1 CPU – class 2 CPU) (CPU time other than processing SQL)
(mostly related to DIST communications)
Note: With DDF rollup in effect, class 1 elapsed time may be too high, and make this calculation invalid.

(Although CPU scheduling problems would likely show up in other workloads as well, one way to detect this within the DDF enclaves is to look at enclave reported queue time.)



IDUG® 2007 North America

Thread Accounting

```
BMC SOFTWARE --------------  SUMMARY TRACE ENTRY    -------------  RX AVAILABLE
SERV ==> STRAC             INPUT   14:37:25  INTVL=> 3  LOG=> N  TGT==> DECE
PARM ==> DIST,SEQ=1                                  ROW 1 OF 135 SCROLL=> CSR
EXPAND:  MON(WKLD), DETAIL, HISTORY
        ACCOUNTING: ENV, ELAPSED, SQLCOUNTS, BPOOL, LOCKS, PRLL, PKG, RTN, DDF
        SUMMARIES:  SQL, SCANS, IO/LOCK, SORTS
STOP.....01MAR 13.50.46.11 PLAN..............DISTSERV TYPE.............[illegible]ED
START..........n/a-ROLLUP  AUTHID............BOLDJW1  CONNECT......SERVER/DRDA
ELAPSED.............315 ms ORIG PRIM AUTH....BOLDJW1  CORR ID.....db2bp.exe
TERM......DDF/RRSAF #LIMIT COMMITS..................2 ROLLBACKS.............0
-----------------------------------------------------------------------------
RUNTIME ANALYSIS    IN DB2      IN APPL.      TOTAL       %IN DB2(=)     TOTAL(*)
----------------   --------    --------    --------     0 ...25...50...75..100%
ELAPSED TIME          46 ms      269 ms      315 ms     | ==****************** |
CPU TIME           1,437 us      626 us    2,064 us     | <                    |
DB2 WAIT TIME         -none-                            |                      |
ZIIP CPU TIME      1,575 us                2,223 us     | <                    |
ZIIP-ELIGIBLE CP                               0 us
- - - - - - ACTIVITY - - - - - -       - - - - - KEY INDICATORS - - - - - - -
TOTAL SQL.......................6      SQL: SELECT=      0, FETCH=      2
GETPAGES........................4      SQL: DYNAMIC(PREPARE)=     2
SYNC READS (PRLL=00) ...........0      DDF/RRSAF ROLL UP RECORD, COUNT=      2
```

```
- - - - - - - - - - - - - ENVIRONMENTAL INDICATORS - - - - - - - - - - - - - -
LUWID.........AC173BD3.B912.070301214616000З
RLF TABLE ID..NOT ACTIVE
          - - - - - - - - - -Work Station Data- - - - - - - - - - -
  WORKSTATION USER ID...boldjw1          WORKSTATION NAME..dwitkows-SJC-04
  WORKSTATION TRANSACTION ID...db2bp.exe
```

33

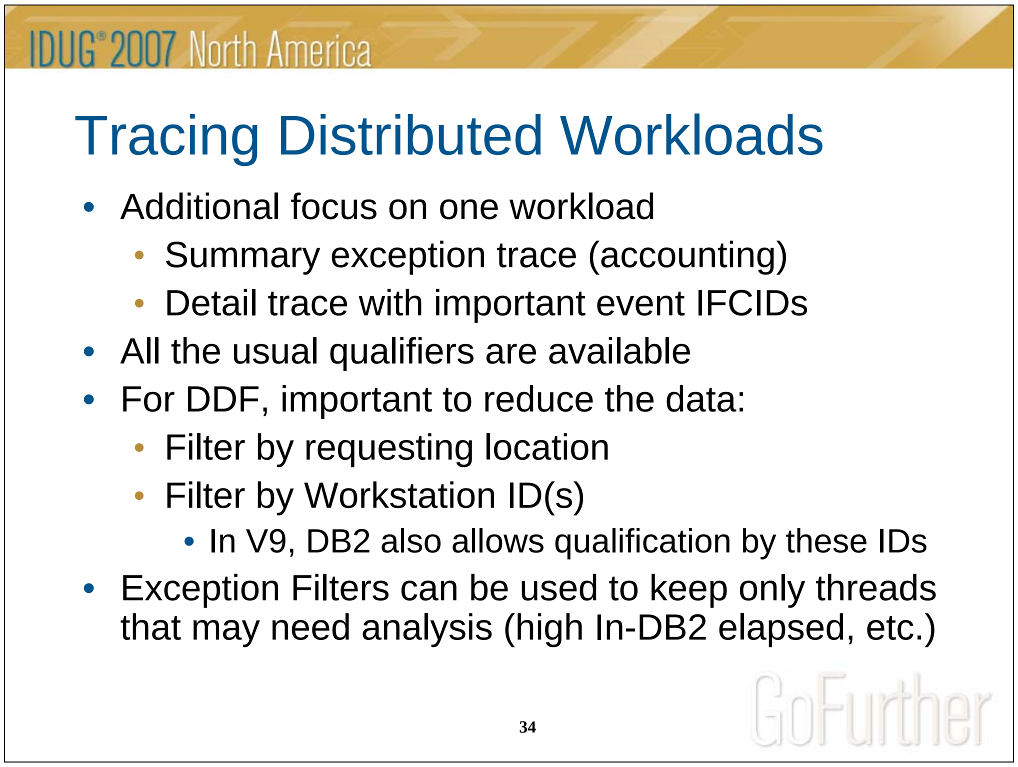
Here is a sample view of one accounting record for a DBAT, to identify the important sections. Note that the DDF section is unavailable for this rollup record.

The client application enclave SRB time is charged to the thread and included in the accounting record CPU times. It can best be analyzed in a summary accounting report of the distributed threads. An important option became available last year for DB2 V8 – the distributed enclave processing is now eligible to run on a zIIP processor, which is much less expensive. However, be aware that the standard class 1 and 2 CPU times no longer include the work on a zIIP, so you may not be able to compare trends of these fields alone. This work is reported separately in specific zIIP fields. These fields are shown in all accounting reporting within MVDB2.

For DBATs, you can look at the In Application CPU as reflecting the time spent in DDF "overhead" (still under the client enclave SRB). Class 2 - class 1 measures time not processing SQL in DB2, and application processing for distributed threads occurs in the client workstation, so this is what remains. It includes part of thread creation and termination CPU, but is mostly the CPU time that is needed to move data in and out of communications buffers.

A special elapsed time consideration for DB2 requester threads is that the time the requesting thread waits for responses from the server to execute remote SQL is reported as Class 3 "service task switch - other". There is also some CPU cost that will show up in the DIST SRB times. It is used for TCP/IP or VTAM message processing that cannot be charged to the thread.

Some of the most important DDF counts help you quantify the amount of traffic required between requester and server, the key differentiator from a local (allied) thread. Messages, rows and blocks sent indicate how well block fetch is being utilized, and this is critical in reducing the network traffic. There are various options that affect this, such as OPTIMIZE for n ROWS, FETCH FIRST n ROWS ONLY, CURRENTDATA(NO) and other BIND options. DB2 V8 now supports multi-row fetch and insert, as well as a larger query block.



IDUG® 2007 North America

Tracing Distributed Workloads

- Additional focus on one workload
  - Summary exception trace (accounting)
  - Detail trace with important event IFCIDs
- All the usual qualifiers are available
- For DDF, important to reduce the data:
  - Filter by requesting location
  - Filter by Workstation ID(s)
    - In V9, DB2 also allows qualification by these IDs
- Exception Filters can be used to keep only threads that may need analysis (high In-DB2 elapsed, etc.)

34

GoFurther

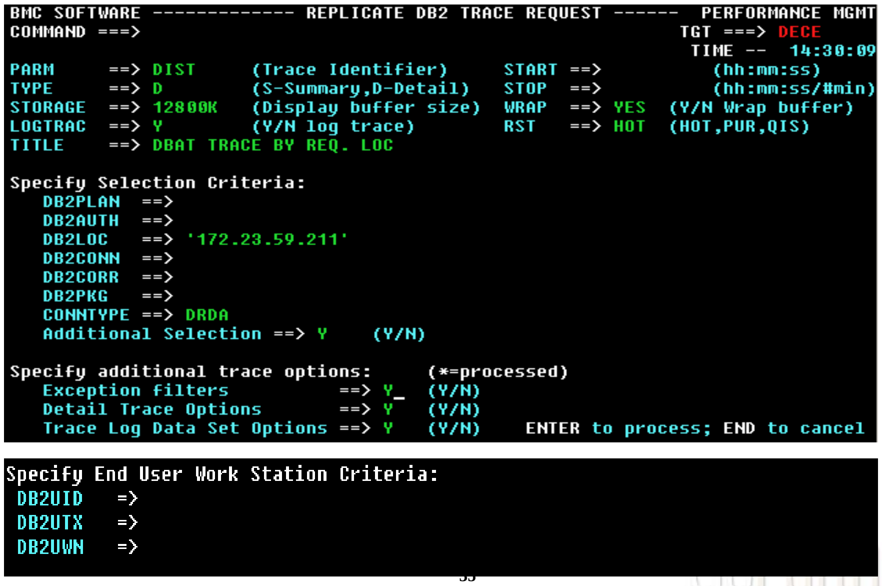
There are two levels of tracing you may need to use in tuning distributed workloads. The first is at the accounting level, and will often provide all the information you need for analysis. The key here is to use some kind of exception tracing to capture only those accounting records that are worth the time to analyze. This may be simply to filter by the identifiers of an application brought to your attention. MVDB2 gives you all the usual qualifiers like AUTHID and plan, but also those that are more important for DBATs, such as requesting location or any of the three workstation IDs. In V7 and V8, only requesting location can actually filter the data returned from DB2, but V9 will also provide filtering by the workstation IDs and Correlation ID as well.

In addition, you may want to qualify the records to be kept by some processing characteristics. One of the most used is a simple one – high in-DB2 elapsed time. Other application-related filters are available as well.

A detail trace can be filtered in the same way, but also collects performance IFCIDs cut during the life of the thread.

IDUG® 2007 North America

Start Trace Options



```
BMC SOFTWARE ------------- REPLICATE DB2 TRACE REQUEST ------  PERFORMANCE MGMT
COMMAND ===>                                                  TGT ===> DECE
                                                               TIME --   14:30:09
PARM     ==> DIST      (Trace Identifier)     START ==>          (hh:mm:ss)
TYPE     ==> D         (S-Summary,D-Detail)   STOP  ==>          (hh:mm:ss/#min)
STORAGE  ==> 12800K    (Display buffer size)  WRAP  ==> YES  (Y/N Wrap buffer)
LOGTRAC  ==> Y         (Y/N log trace)        RST   ==> HOT  (HOT,PUR,QIS)
TITLE    ==> DBAT TRACE BY REQ. LOC

Specify Selection Criteria:
   DB2PLAN   ==>
   DB2AUTH   ==>
   DB2LOC    ==> '172.23.59.211'
   DB2CONN   ==>
   DB2CORR   ==>
   DB2PKG    ==>
   CONNTYPE ==> DRDA
   Additional Selection ==> Y    (Y/N)

Specify additional trace options:     (*=processed)
   Exception filters          ==> Y_  (Y/N)
   Detail Trace Options       ==> Y   (Y/N)
   Trace Log Data Set Options ==> Y   (Y/N)    ENTER to process; END to cancel
```

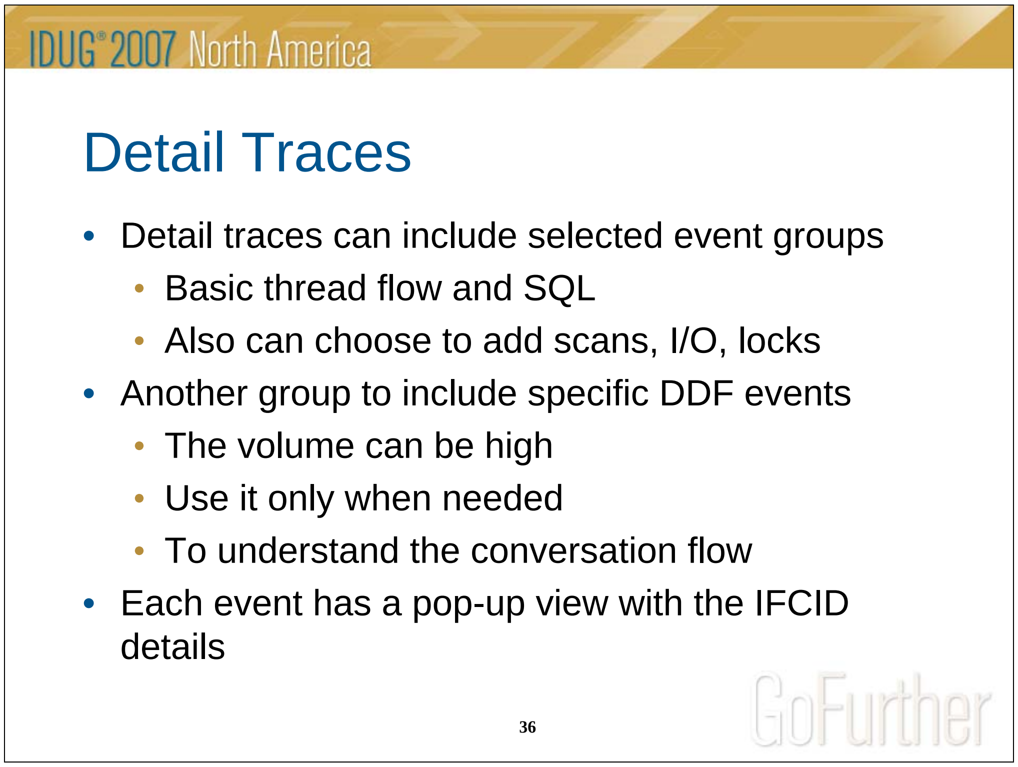
```
Specify End User Work Station Criteria:
 DB2UID   =>
 DB2UTX   =>
 DB2UWN   =>
```

The Start Trace panel allows you to select the type – summary accounting or with detail events, logging to keep the data in its own data set(s), and to specify a variety of workload qualifiers.

Here I have specified one particular requesting location, and will also add the workstation transaction (program name) in DB2UWN.

The Exceptions filters option provides a panel to specify thresholds on measures like elapsed time.

The detail trace option allow you to select different groups of performance IFCIDs to be included in a detail trace.


IDUG® 2007 North America

Detail Traces

- Detail traces can include selected event groups
  - Basic thread flow and SQL
  - Also can choose to add scans, I/O, locks
- Another group to include specific DDF events
  - The volume can be high
  - Use it only when needed
  - To understand the conversation flow
- Each event has a pop-up view with the IFCID details

36

GoFurther

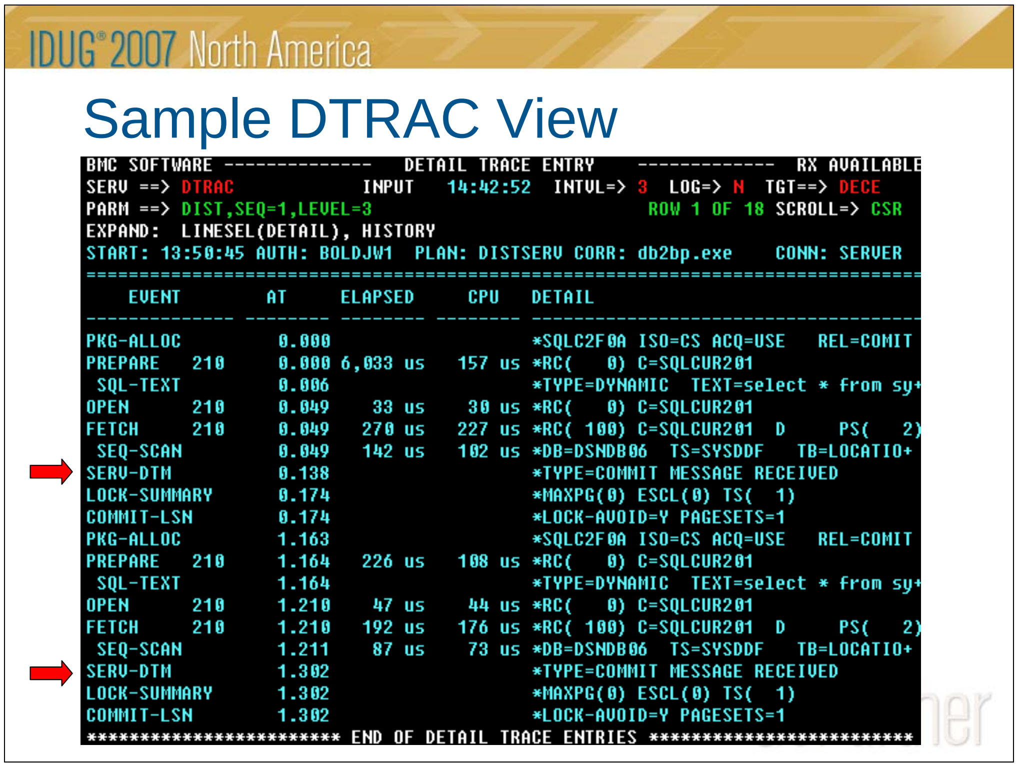
Detail traces are specified to include only selected groups of events. A basic set of events define the life of a thread and major exceptions such as timeouts. The SQL statements are almost always included as well. Note that the Prepare for dynamic SQL will also provide the SQL text and mini-Explain. You can also choose to add scans and/or I/O to capture object access information, or lock events for complicated lock analysis (usually not needed).

There are also two groups of DDF events, but the volume can be high, since they track each step of the communications between the client and DB2. It can be very useful to understand the flow of the application, the conversations and the traffic generated, but it will not directly point out problems in the network itself. However, it is likely easier to capture and understand than a client DB2 CLI trace, or a ddcstrc DRDA trace.

It includes IFCIDs 157 – 163, 167 and 183. For example, IFCID 163 traces events at the DB2 for z/OS server for DBATs, and identifies critical events, such as:

- DBAT creation
- Commit request received from the coordinator
- Backout request received from the coordinator
- DBAT creation queued
- Deallocation initiated
- more . . .

Other IFCIDs are related to the conversation or message traffic.



Sample DTRAC View

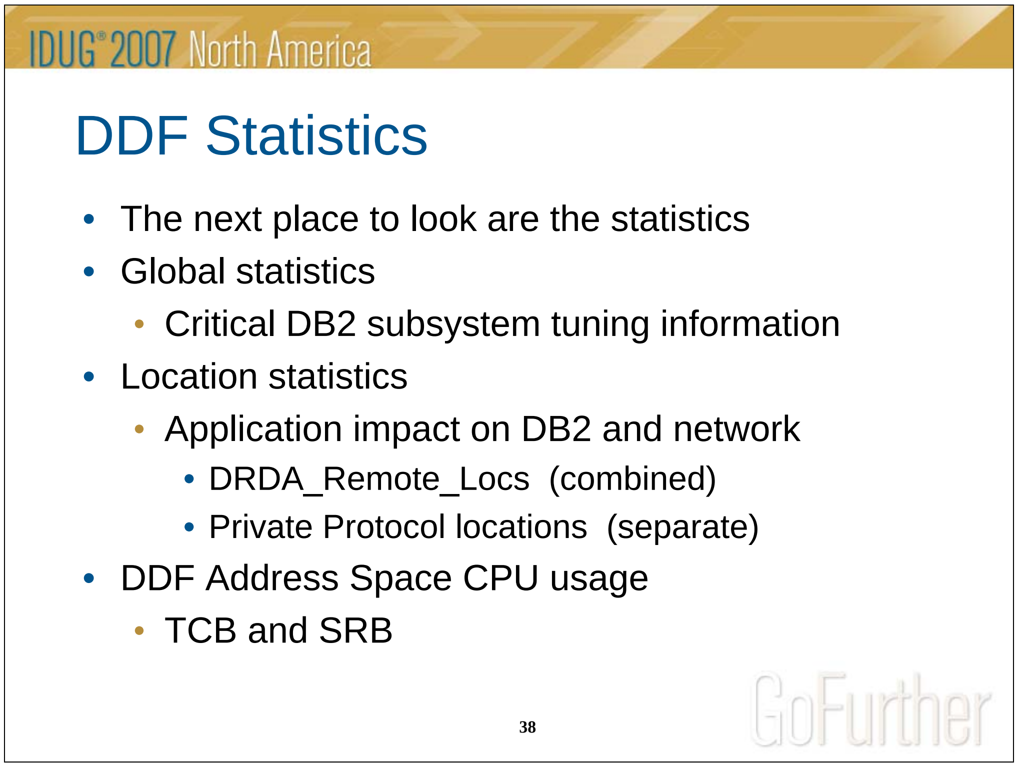
```
BMC SOFTWARE --------------   DETAIL TRACE ENTRY    -------------  RX AVAILABLE
SERV ==> DTRAC             INPUT   14:42:52  INTVL=> 3  LOG=> N  TGT==> DECE
PARM ==> DIST,SEQ=1,LEVEL=3                           ROW 1 OF 18 SCROLL=> CSR
EXPAND:  LINESEL(DETAIL), HISTORY
START: 13:50:45 AUTH: BOLDJW1  PLAN: DISTSERV CORR: db2bp.exe    CONN: SERVER
===============================================================================
    EVENT        AT      ELAPSED     CPU   DETAIL
-------------- -------- -------- -------- -------------------------------------
PKG-ALLOC          0.000                   *SQLC2F0A ISO=CS ACQ=USE   REL=COMIT
PREPARE   210      0.000 6,033 us   157 us *RC(    0) C=SQLCUR201
 SQL-TEXT          0.006                   *TYPE=DYNAMIC  TEXT=select * from sy+
OPEN      210      0.049    33 us    30 us *RC(    0) C=SQLCUR201
FETCH     210      0.049   270 us   227 us *RC( 100) C=SQLCUR201  D     PS(   2)
 SEQ-SCAN          0.049   142 us   102 us *DB=DSNDB06  TS=SYSDDF   TB=LOCATIO+
SERV-DTM           0.138                   *TYPE=COMMIT MESSAGE RECEIVED
LOCK-SUMMARY       0.174                   *MAXPG(0) ESCL(0) TS(  1)
COMMIT-LSN         0.174                   *LOCK-AVOID=Y PAGESETS=1
PKG-ALLOC          1.163                   *SQLC2F0A ISO=CS ACQ=USE   REL=COMIT
PREPARE   210      1.164   226 us   108 us *RC(    0) C=SQLCUR201
 SQL-TEXT          1.164                   *TYPE=DYNAMIC  TEXT=select * from sy+
OPEN      210      1.210    47 us    44 us *RC(    0) C=SQLCUR201
FETCH     210      1.210   192 us   176 us *RC( 100) C=SQLCUR201  D     PS(   2)
 SEQ-SCAN          1.211    87 us    73 us *DB=DSNDB06  TS=SYSDDF   TB=LOCATIO+
SERV-DTM           1.302                   *TYPE=COMMIT MESSAGE RECEIVED
LOCK-SUMMARY       1.302                   *MAXPG(0) ESCL(0) TS(  1)
COMMIT-LSN         1.302                   *LOCK-AVOID=Y PAGESETS=1
************************ END OF DETAIL TRACE ENTRIES *************************
```

Besides this chronological event listing, the full details from each event IFCID are available in pop-up views.

You can change the level of events included in the event listing from LEVEL=2, the default with the most important events, to LEVEL=1 to focus only on the SQL, or LEVEL=3 to show all events traced.

You can also use PF10 / PF11 to easily move from one thread to the next, or even from one event popup to the next.

Note that this is a trace with DDF rollup active, with a limit of 2. So two queries / threads are included.


IDUG® 2007 North America

DDF Statistics

- The next place to look are the statistics
- Global statistics
  - Critical DB2 subsystem tuning information
- Location statistics
  - Application impact on DB2 and network
    - DRDA_Remote_Locs (combined)
    - Private Protocol locations (separate)
- DDF Address Space CPU usage
  - TCB and SRB

38

GoFurther

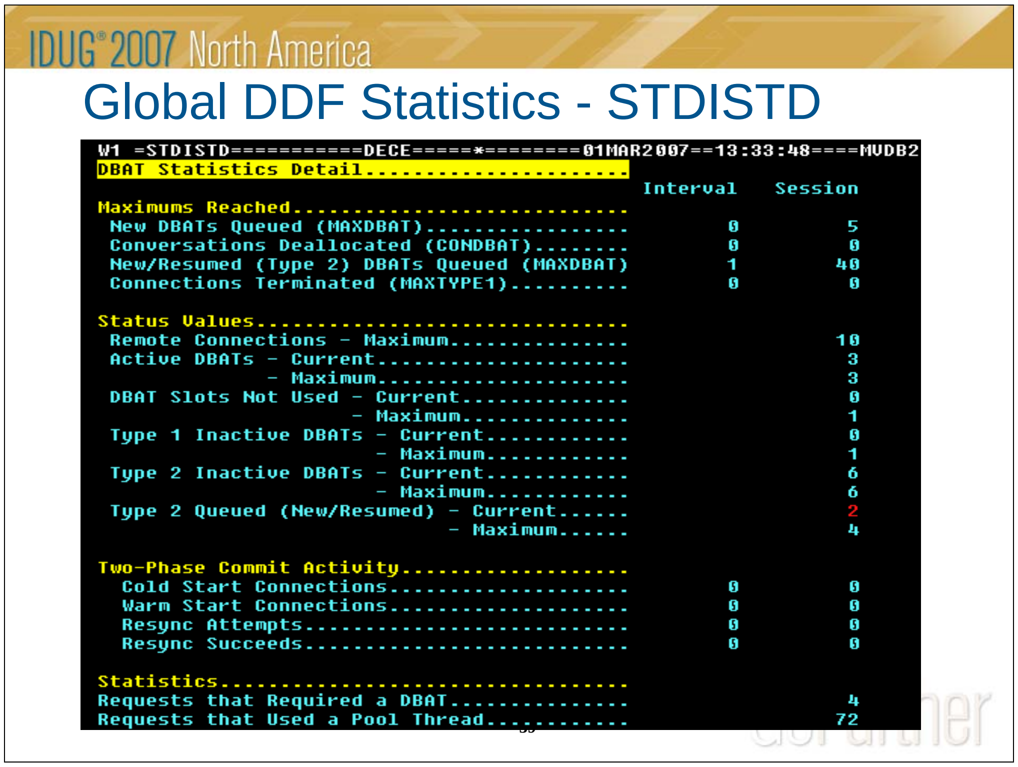
After accounting, the most important information about the impact of DDF processing in DB2 is in the statistics. There are different types of information available.

QDST - global statistics on DBAT and connection usage

QLST – activity measurements per "location":
- DRDA_Remote_Locs – all DRDA work is summarized under this name
- Private Protocol locations have individual statistics (if you still have any)

QWSA - DDF CPU usage – see slide 45

Note: When looking at any statistics reporting, be sure to include the DDF thread and commit counts to get good averages.



IDUG® 2007 North America

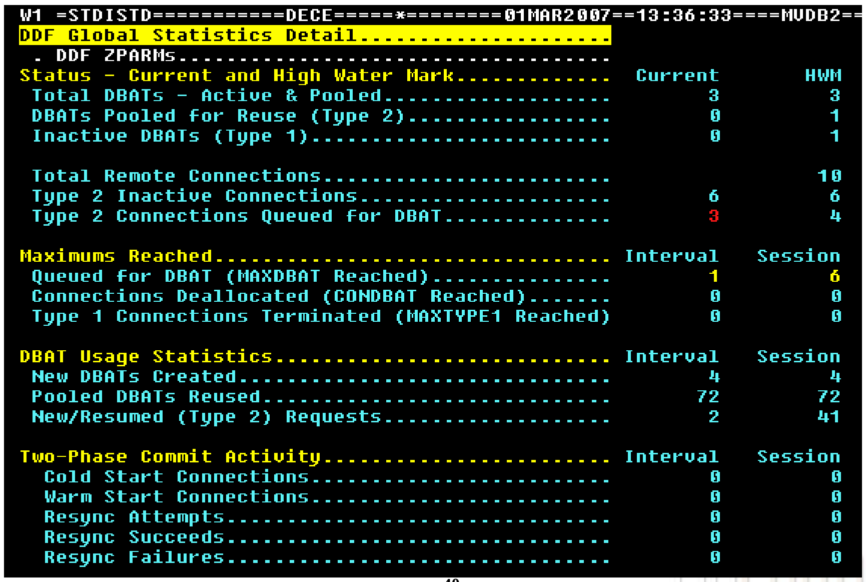
Global DDF Statistics - STDISTD

```
W1 =STDISTD==========DECE=====*========01MAR2007==13:33:48====MVDB2
DBAT Statistics Detail......................
                                              Interval   Session
Maximums Reached.............................
 New DBATs Queued (MAXDBAT).................         0         5
 Conversations Deallocated (CONDBAT)........         0         0
 New/Resumed (Type 2) DBATs Queued (MAXDBAT)         1        40
 Connections Terminated (MAXTYPE1)..........         0         0

Status Values................................
 Remote Connections - Maximum...............                  10
 Active DBATs - Current.....................                   3
              - Maximum.....................                   3
 DBAT Slots Not Used - Current..............                   0
                     - Maximum..............                   1
 Type 1 Inactive DBATs - Current............                   0
                       - Maximum............                   1
 Type 2 Inactive DBATs - Current............                   6
                       - Maximum............                   6
 Type 2 Queued (New/Resumed) - Current......                   2
                             - Maximum......                   4

Two-Phase Commit Activity....................
  Cold Start Connections....................         0         0
  Warm Start Connections....................         0         0
  Resync Attempts...........................         0         0
  Resync Succeeds...........................         0         0

Statistics...................................
Requests that Required a DBAT...............                   4
Requests that Used a Pool Thread............                  72
```

This is the current view of the global DDF statistics, using the standard terminology as reflected in other reports and the field definitions in DSNWMSGS.

Note that there is another view, STDDFD, that shows the DDF activity counts, such as the number of SQL, messages and blocks sent and received. They can be used to evaluate the amount of network traffic being generated. Originally designed to be reported by location (private protocol), all DRDA traffic is included in one set of data named DRDA_REMOTE_LOCS. The similar counts in the accounting records are more useful to evaluate the effectiveness of blocking options used by the applications.

STDISTD View - Revised



```
W1 =STDISTD===========DECE=====*========01MAR2007==13:36:33====MVDB2==
DDF Global Statistics Detail......................
 . DDF ZPARMs.....................................
Status - Current and High Water Mark.............  Current       HWM
 Total DBATs - Active & Pooled...................        3         3
 DBATs Pooled for Reuse (Type 2).................        0         1
 Inactive DBATs (Type 1).........................        0         1

 Total Remote Connections........................                 10
 Type 2 Inactive Connections.....................        6         6
 Type 2 Connections Queued for DBAT..............        3         4

Maximums Reached................................. Interval   Session
 Queued for DBAT (MAXDBAT Reached)...............        1         6
 Connections Deallocated (CONDBAT Reached).......        0         0
 Type 1 Connections Terminated (MAXTYPE1 Reached)        0         0

DBAT Usage Statistics............................ Interval   Session
 New DBATs Created...............................        4         4
 Pooled DBATs Reused.............................       72        72
 New/Resumed (Type 2) Requests...................        2        41

Two-Phase Commit Activity........................ Interval   Session
  Cold Start Connections.........................        0         0
  Warm Start Connections.........................        0         0
  Resync Attempts................................        0         0
  Resync Succeeds................................        0         0
  Resync Failures................................        0         0
```

40

This view of global DDF system activity has been extensively revised to reflect the results of our research into terminology, and testing what the statistics really measure. And we moved the current / high water mark values next to each other – both to make them easier to read, and also to make it easier to see if the current value is close or equal to the high water mark.

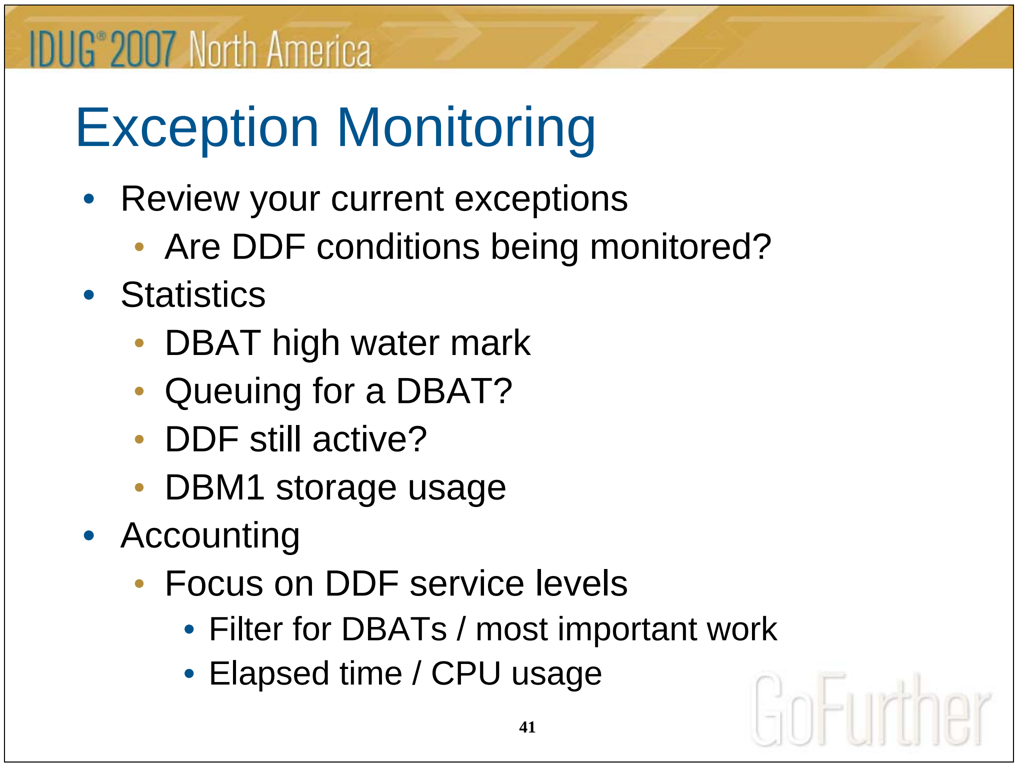
The current status values are on the top, so that you can see how many current threads and connections there are. The total DBAT value, usually called just “active DBATs”, has been renamed to show that the values include both active and pooled DBATs – in other words, all real DBATs. The pooled DBATs are identified as such.

The connection status values now include what was previously called the “inactive type 2 DBATs”. They have been renamed to what they really are: “type 2 inactive connections”, not DBATs at all. If any connections are currently queued because MAXDBAT was reached, the “Type 2 Connections Queued for DBAT” values are highlighted in red.

The maximums reached section shows potential warning conditions. Any number that is consistently non-zero should be analyzed to see if the related ZPARM limit should be revised upward. On the other hand, if the current and high water mark values are consistently below the limits, they can perhaps safely be lowered. Since the total thread count is limited to 2000, MAXDBAT is the most likely to need this kind of tuning.

One more addition is a count of resync failures. Various sources comment that if resynchronization attempts are much greater than the number that succeeded, a communication problem with one of the remote connections is likely.

A hyperlink to the view of DDF-related ZPARMs is provided at the top to simplify a quick review of the values in effect.


IDUG® 2007 North America

Exception Monitoring

- Review your current exceptions
  - Are DDF conditions being monitored?
- Statistics
  - DBAT high water mark
  - Queuing for a DBAT?
  - DDF still active?
  - DBM1 storage usage
- Accounting
  - Focus on DDF service levels
    - Filter for DBATs / most important work
    - Elapsed time / CPU usage

41

GoFurther

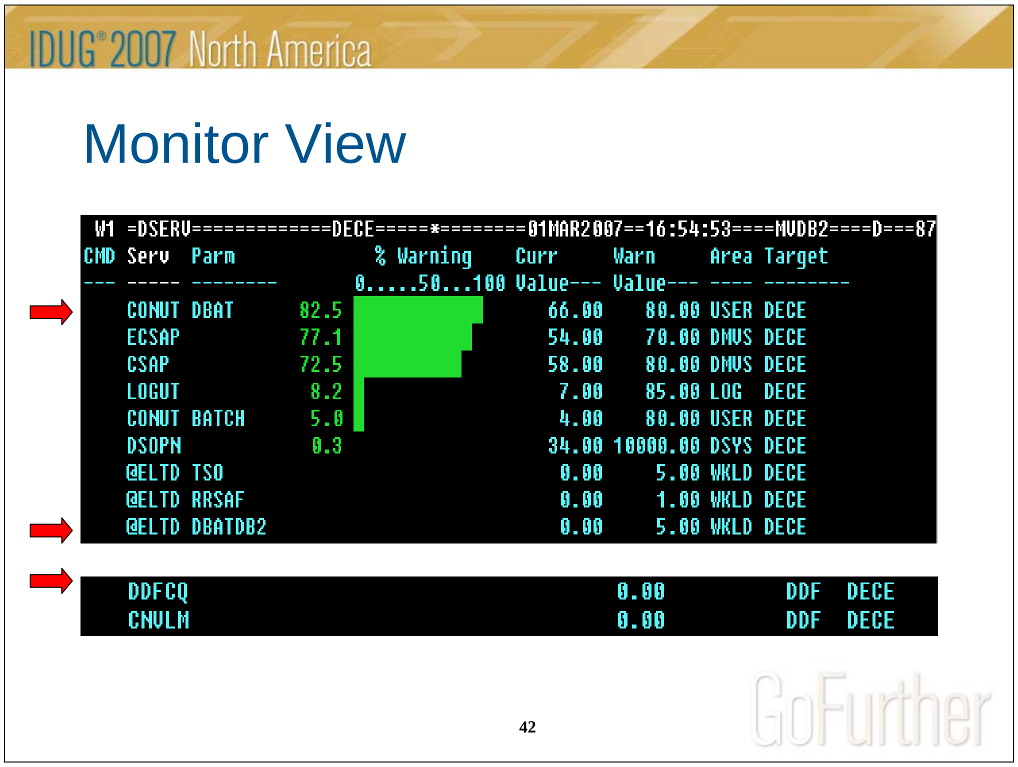
Once you have an understanding of the accounting and statistics measurements, review your current exception monitoring to determine whether additional monitors specifically aimed at DDF are needed. For workloads, filter first on connect type (DRDA or PP), or on a requesting location or workstation IDs.

For subsystem analysis, the global statistics values like the DBAT high water mark and percent of MAXDBAT are the most important, to help catch overuse before DBAT queuing occurs. Simple and efficient time-driven monitors issue exception messages based on user-defined thresholds for
- DBATQ / DBTUT - DBAT queuing or DBAT % utilization
- CNVLM / DDFCQ - Conversations queued or deallocated
- NACTC - The current number of pooled DBATs (type 2) , also type 1
- P2CON - Cold start or warm start of connections (two-phase commit only)
- P2RSY - The number of resync failures (= attempts - successes)
This could indicate network problems

Other important indicators can be set as alarms, such as whether DDF is no longer active, or DBM1 has storage constraints.

For active threads, an indicator of a hung thread is whether the last conversation time stamp is increasing. If not, you may need to consider canceling it.

Also, when MAXDBAT is reached (primarily in ACTIVE mode), the connection request may simply wait for one to become available – forever. This can be difficult to track down unless you have an exception for DBAT queuing.



IDUG® 2007 North America

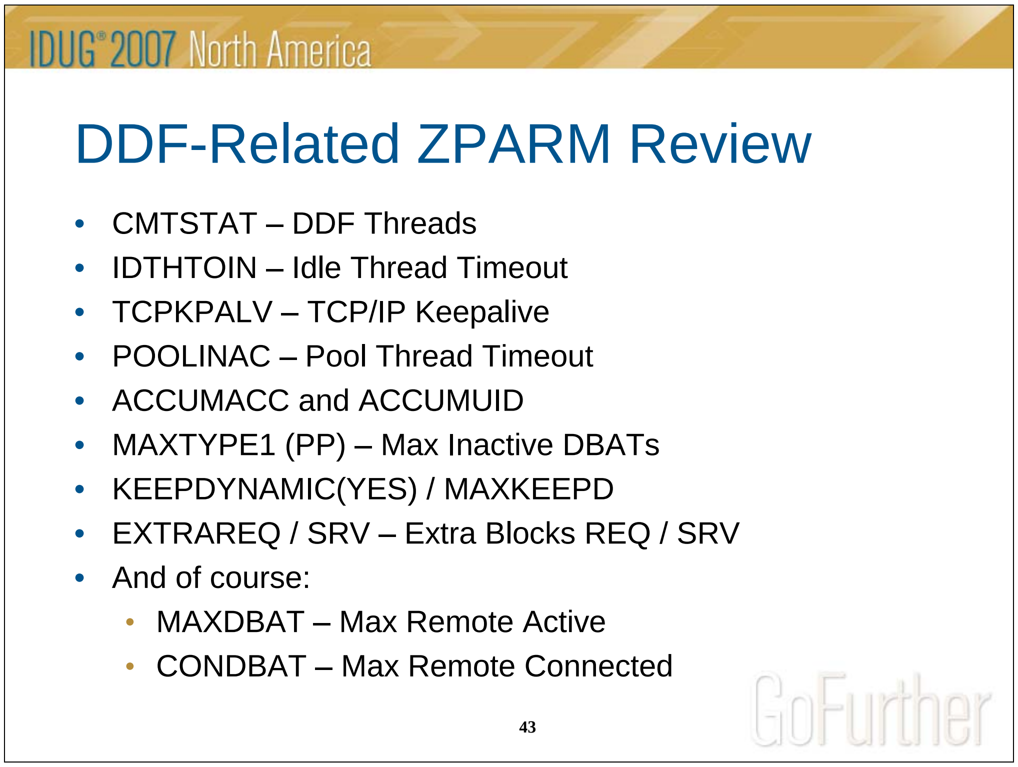
Monitor View

```
 W1 =DSERV============DECE=====*========01MAR2007==16:54:53====MVDB2====D===87
CMD Serv  Parm              % Warning     Curr      Warn      Area Target
--- ----- --------        0.....50...100 Value---  Value---  ---- --------
    CONVT DBAT      82.5                  66.00     80.00 USER DECE
    ECSAP           77.1                  54.00     70.00 DMVS DECE
    CSAP            72.5                  58.00     80.00 DMVS DECE
    LOGVT            8.2                   7.00     85.00 LOG  DECE
    CONVT BATCH      5.0                   4.00     80.00 USER DECE
    DSOPN            0.3                  34.00  10000.00 DSYS DECE
    @ELTD TSO                              0.00      5.00 WKLD DECE
    @ELTD RRSAF                            0.00      1.00 WKLD DECE
    @ELTD DBATDB2                          0.00      5.00 WKLD DECE

    DDFCQ                                       0.00           DDF   DECE
    CNVLM                                       0.00           DDF   DECE
```

42

GoFurther

Monitors are simple to set up in a group request that is automatically started at DB2 startup. You can define the sampling interval (default is one minute), the warning threshold, and when and how often an exception message should be written – and where it should be sent. The messages automatically go into the Journal log, where you have a chronological view of what has been happening in your DB2 subsystems, since this log also includes DB2 messages. The exception messages can also be processed by MAINVIEW AutoOPERATOR rules to take action, or sent to the console for operator notification.


IDUG® 2007 North America

DDF-Related ZPARM Review

- CMTSTAT – DDF Threads
- IDTHTOIN – Idle Thread Timeout
- TCPKPALV – TCP/IP Keepalive
- POOLINAC – Pool Thread Timeout
- ACCUMACC and ACCUMUID
- MAXTYPE1 (PP) – Max Inactive DBATs
- KEEPDYNAMIC(YES) / MAXKEEPD
- EXTRAREQ / SRV – Extra Blocks REQ / SRV
- And of course:
  - MAXDBAT – Max Remote Active
  - CONDBAT – Max Remote Connected

43

GoFurther

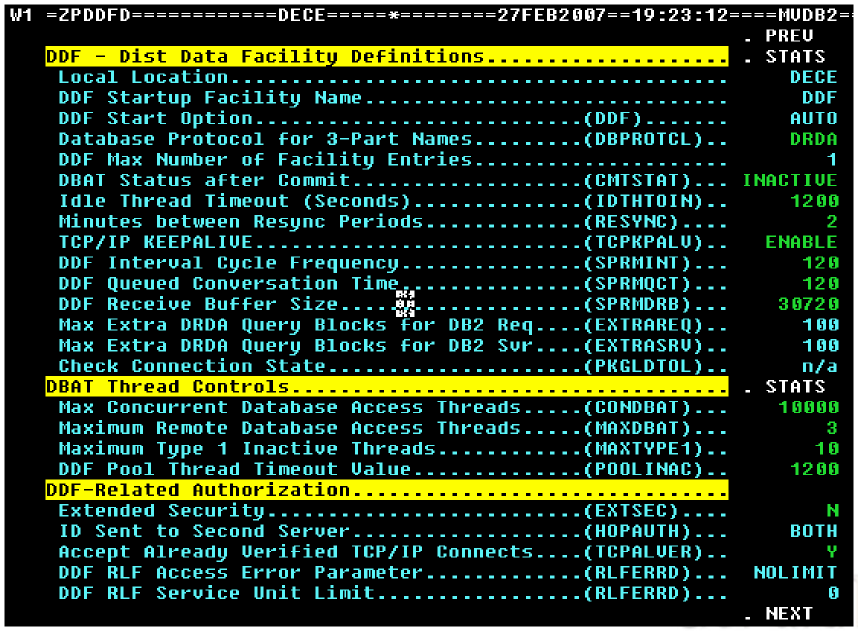
Many of these ZPARMs have already been covered during the processing flow explanations. Here is a short summary of the most critical**:**

- CMTSTAT = INACTIVE / ACTIVE
- IDTHTOIN – Idle thread timeout (DRDA), to clean up "active" DBATs doing no work (not recommended for ACTIVE mode)
- TCPKPALV – Coordinate with IDTHTOIN to help detect network failures
- POOLINAC – Pool thread timeout (DRDA), to reduce the number of pooled DBATs to match the workload (INACTIVE mode)
- ACCUMACC and ACCUMUID – to specify DDF accounting rollup to reduce record volume
- MAXTYPE1 – to allow type 1 inactive processing for private protocol
- KEEPDYNAMIC(YES) – driven by application needs (SAP, for example)
  And related MAXKEEPD – to control storage usage in DBM1
- EXTRAREQ and EXTRASRV to set upper limits on the number of query blocks transmitted at one time
- And of course – to control resource usage:
  - MAXDBAT – maximum number of DBATs
  - CONDBAT – maximum number of remote connections

  In INACTIVE mode CONDBAT can be much higher than MAXDBAT
  In ACTIVE mode they should generally be equal

IDUG® 2007 North America

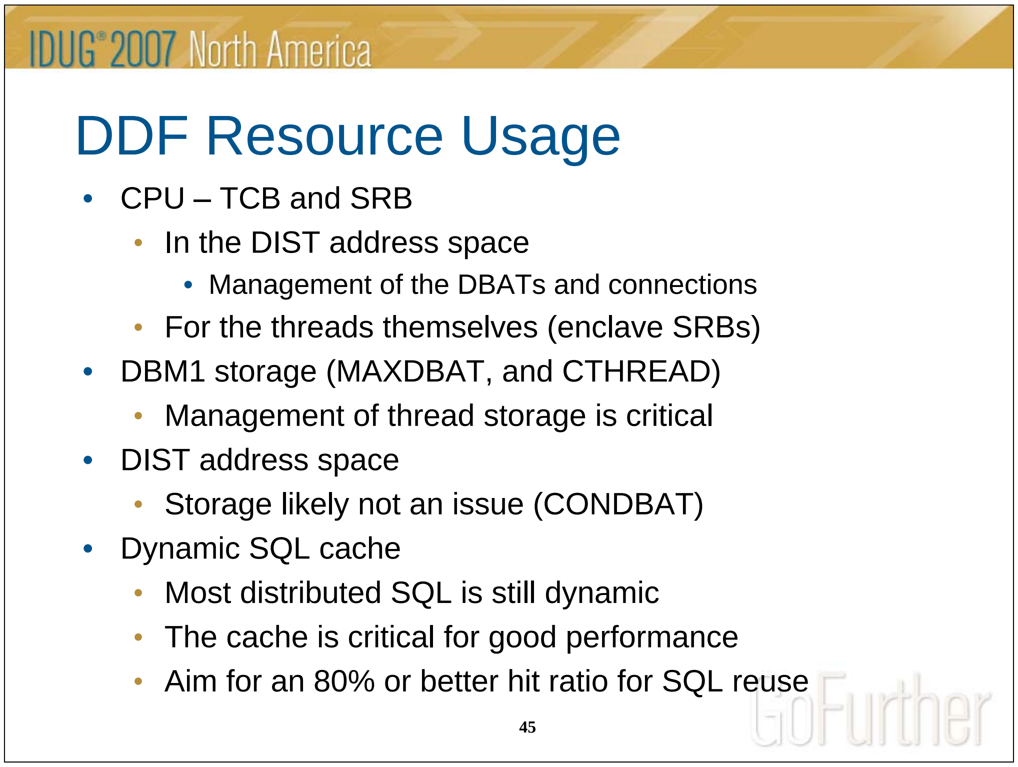
DDF ZPARM View



```
W1 =ZPDDFD===========DECE=====*========27FEB2007==19:23:12====MVDB2=
                                                           . PREV
   DDF - Dist Data Facility Definitions..................  . STATS
    Local Location........................................   DECE
    DDF Startup Facility Name.............................    DDF
    DDF Start Option...........................(DDF).......   AUTO
    Database Protocol for 3-Part Names.........(DBPROTCL)..   DRDA
    DDF Max Number of Facility Entries....................      1
    DBAT Status after Commit...................(CMTSTAT)... INACTIVE
    Idle Thread Timeout (Seconds)..............(IDTHTOIN)..   1200
    Minutes between Resync Periods.............(RESYNC)....      2
    TCP/IP KEEPALIVE...........................(TCPKPALV)..  ENABLE
    DDF Interval Cycle Frequency...............(SPRMINT)...    120
    DDF Queued Conversation Time...............(SPRMQCT)...    120
    DDF Receive Buffer Size....................(SPRMDRB)...  30720
    Max Extra DRDA Query Blocks for DB2 Req....(EXTRAREQ)..    100
    Max Extra DRDA Query Blocks for DB2 Svr....(EXTRASRV)..    100
    Check Connection State.....................(PKGLDTOL)..    n/a
   DBAT Thread Controls..................................  . STATS
    Max Concurrent Database Access Threads.....(CONDBAT)...  10000
    Maximum Remote Database Access Threads.....(MAXDBAT)...      3
    Maximum Type 1 Inactive Threads............(MAXTYPE1)..     10
    DDF Pool Thread Timeout Value..............(POOLINAC)..   1200
   DDF-Related Authorization.............................
    Extended Security..........................(EXTSEC)....      N
    ID Sent to Second Server...................(HOPAUTH)...   BOTH
    Accept Already Verified TCP/IP Connects....(TCPALVER)..      Y
    DDF RLF Access Error Parameter.............(RLFERRD)...  NOLIMIT
    DDF RLF Service Unit Limit.................(RLFERRD)...      0
                                                           . NEXT
```

This view consolidates all the DDF-related ZPARMs together. There is help available for each ZPARM. Also, those highlighted in green can be changed dynamically (and individually) through OPERTUNE, if it is installed.

There are also ZPARM index views (ZPNAMEA, ZPNAMEB, etc.) that provide direct hyperlinks when you want to look up one specific ZPARM, like POOLINAC on ZPNAMEP (that links to this view), or KEEPDYNAMIC on ZPNAMEK (that links you to a different view).


IDUG® 2007 North America

DDF Resource Usage

- CPU – TCB and SRB
  - In the DIST address space
    - Management of the DBATs and connections
  - For the threads themselves (enclave SRBs)
- DBM1 storage (MAXDBAT, and CTHREAD)
  - Management of thread storage is critical
- DIST address space
  - Storage likely not an issue (CONDBAT)
- Dynamic SQL cache
  - Most distributed SQL is still dynamic
  - The cache is critical for good performance
  - Aim for an 80% or better hit ratio for SQL reuse

45

GoFurther

CPU usage
DDF DIST address space TCB and SRB is shown in view STDBSYSD. The DDF CPU usage is an indicator of the amount of management overhead necessary to manage the distributed threads, connections and message traffic in the DB2 subsystem, either when acting as a server or a requester. For example, creating a DBAT is relatively expensive (CPU), and also causes some serialization. That is one reason INACTIVE mode is preferred, along with the retention of connection information. Much of this CPU is accumulated as DIST SRB time. The SDSF ENClave display can also be used to see the current CPU and the classification attributes.

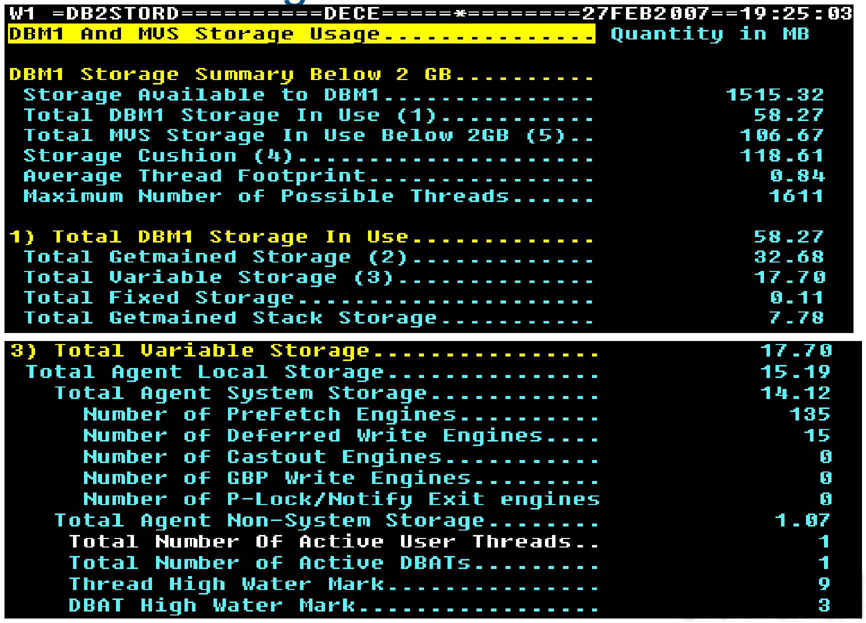
DBM1 storage
DBM1 storage usage is impacted greatly by thread-related storage – including DBATs. That is why methods to limit the number of DBATs like pooling DBATs for reuse, and managing the number are so important. Without these measures, DB2 would not be able to manage such a high number of connections. See next slide.

DIST storage is mainly affected by CONDBAT.

Dynamic SQL cache
Distributed threads are still heavy users of dynamic SQL, although the improved SQLJ (static) support may change this over time. The effectiveness of the cache improves with statement reuse. Check the statistics and aim for a 80% or greater hit ratio.

IDUG® 2007 North America

DBM1 Storage – DB2STORD View



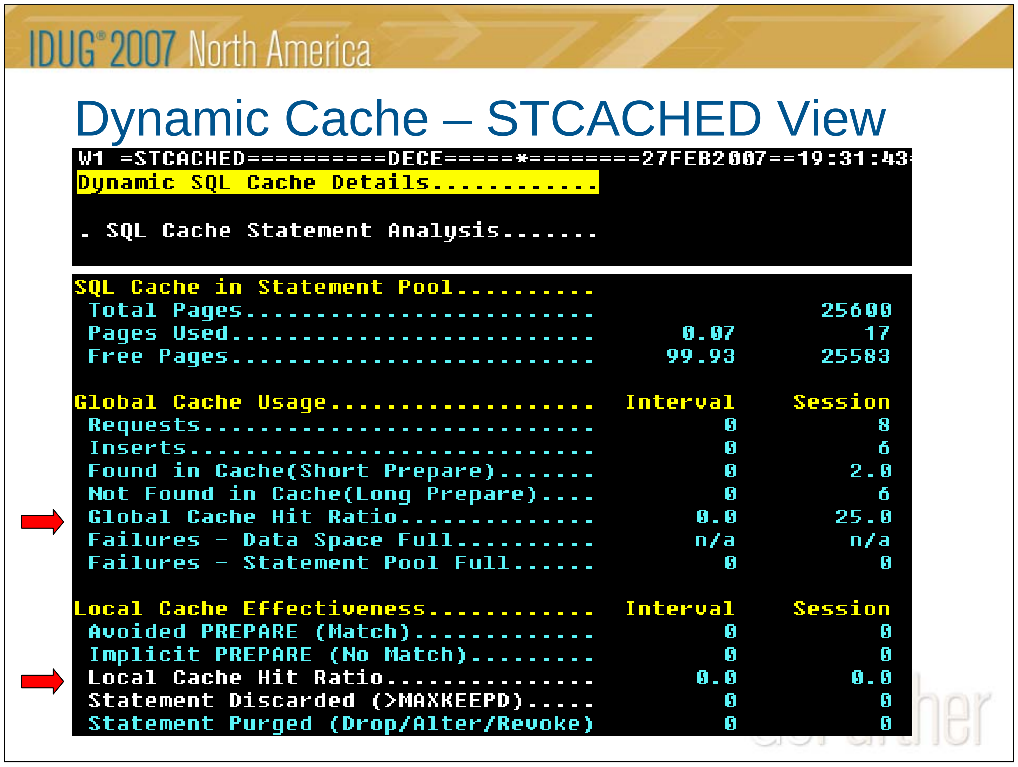
```
W1  =DB2STORD==========DECE=====*========27FEB2007==19:25:03
DBM1 And MVS Storage Usage............... Quantity in MB

DBM1 Storage Summary Below 2 GB..........
 Storage Available to DBM1...............         1515.32
 Total DBM1 Storage In Use (1)...........           58.27
 Total MVS Storage In Use Below 2GB (5)..          106.67
 Storage Cushion (4).....................          118.61
 Average Thread Footprint................            0.84
 Maximum Number of Possible Threads......            1611

1) Total DBM1 Storage In Use.............           58.27
 Total Getmained Storage (2).............           32.68
 Total Variable Storage (3)..............           17.70
 Total Fixed Storage.....................            0.11
 Total Getmained Stack Storage...........            7.78

3) Total Variable Storage................           17.70
 Total Agent Local Storage...............           15.19
   Total Agent System Storage............           14.12
     Number of PreFetch Engines..........             135
     Number of Deferred Write Engines....              15
     Number of Castout Engines...........               0
     Number of GBP Write Engines.........               0
     Number of P-Lock/Notify Exit engines               0
   Total Agent Non-System Storage........            1.07
    Total Number Of Active User Threads..               1
    Total Number of Active DBATs.........               1
    Thread High Water Mark...............               9
    DBAT High Water Mark.................               3
```

DBM1 storage

The view DB2STORD and the batch report BSTATSTM both include the thread footprint calculation, which assists you in understanding how many threads can be supported. Note that the active thread count used in these calculations include both the standard threads (controlled by CTHREAD) and the high water mark of the DBATs. Since pooled DBATs still hold storage in DBM1, it is important to include them in the calculation. The high water mark helps define the range of steady state DBAT usage that includes both active and pooled DBATs. Again, INACTIVE mode is much more efficient in managing this critical resource.

Another user of DBM1 storage is the local statement cache, related to active threads. Normally this cache is cleared when a DBAT is pooled, but with KEEPDYNAMIC(YES), the statements are retained. The amount of local cache used in this manner is controlled by MAXKEEPD. Also note that a KEEPDYNAMIC application can force a rollback to clear the cache.

ZPARMs CONTSTOR and MINSTOR provide other options for DB2 management of thread working storage.



IDUG® 2007 North America

Dynamic Cache – STCACHED View

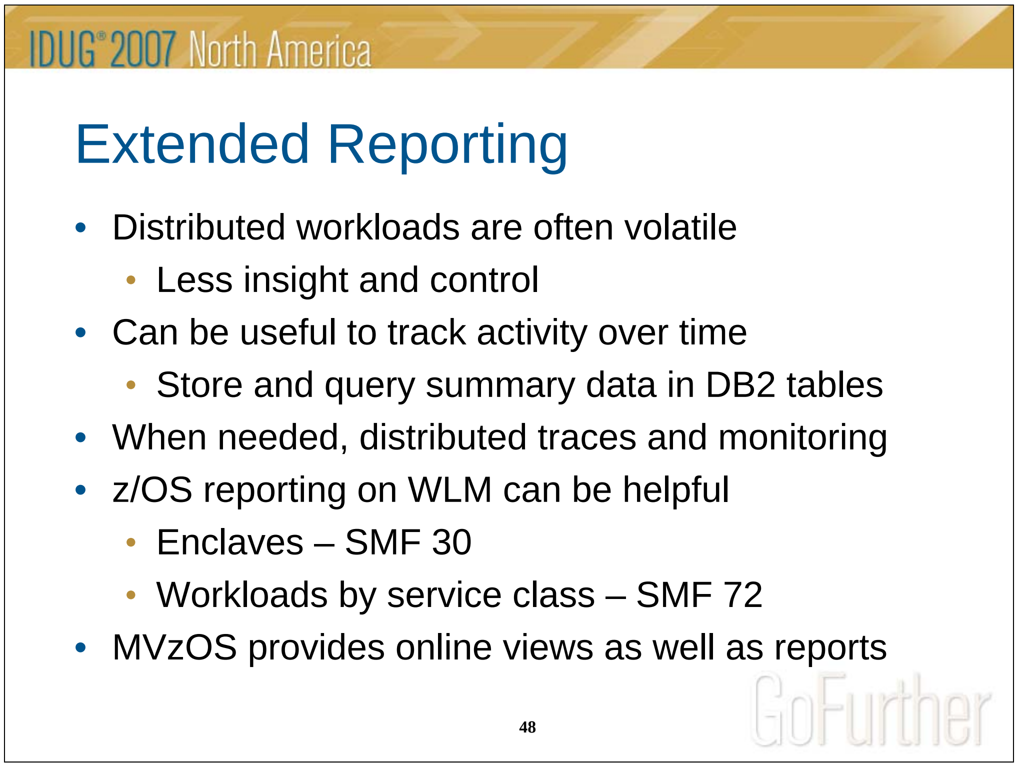
```
W1 =STCACHED=========DECE=====*========27FEB2007==19:31:43
Dynamic SQL Cache Details............

. SQL Cache Statement Analysis.......

SQL Cache in Statement Pool..........
 Total Pages.........................                25600
 Pages Used..........................       0.07        17
 Free Pages..........................      99.93     25583

Global Cache Usage...................   Interval   Session
 Requests............................          0         8
 Inserts.............................          0         6
 Found in Cache(Short Prepare).......          0       2.0
 Not Found in Cache(Long Prepare)....          0         6
 Global Cache Hit Ratio..............        0.0      25.0
 Failures - Data Space Full..........        n/a       n/a
 Failures - Statement Pool Full......          0         0

Local Cache Effectiveness............   Interval   Session
 Avoided PREPARE (Match).............          0         0
 Implicit PREPARE (No Match).........          0         0
 Local Cache Hit Ratio...............        0.0       0.0
 Statement Discarded (>MAXKEEPD).....          0         0
 Statement Purged (Drop/Alter/Revoke)          0         0
```

The most important values on this view of the dynamic SQL cache are the two hit ratios.

The global cache hit ratio monitors the effectiveness of the SQL caching with statement reuse. Check the statistics and aim for a 80% or greater hit ratio.

The local cache hit ratio monitors the effectiveness of the local cache with KEEPDYNAMIC(YES). Since this requires application involvement, a poor ratio suggests that the accounting data should be reviewed. That includes similar counts that show effectiveness at the application / thread level.

Also, you should track are the percent of statement pool used, although this no longer affects DBM1 storage. In DB2 V8, the statement pool is above the bar. Only real storage usage is a possible concern.


IDUG® 2007 North America

Extended Reporting

- Distributed workloads are often volatile
  - Less insight and control
- Can be useful to track activity over time
  - Store and query summary data in DB2 tables
- When needed, distributed traces and monitoring
- z/OS reporting on WLM can be helpful
  - Enclaves – SMF 30
  - Workloads by service class – SMF 72
- MVzOS provides online views as well as reports

48

GoFurther

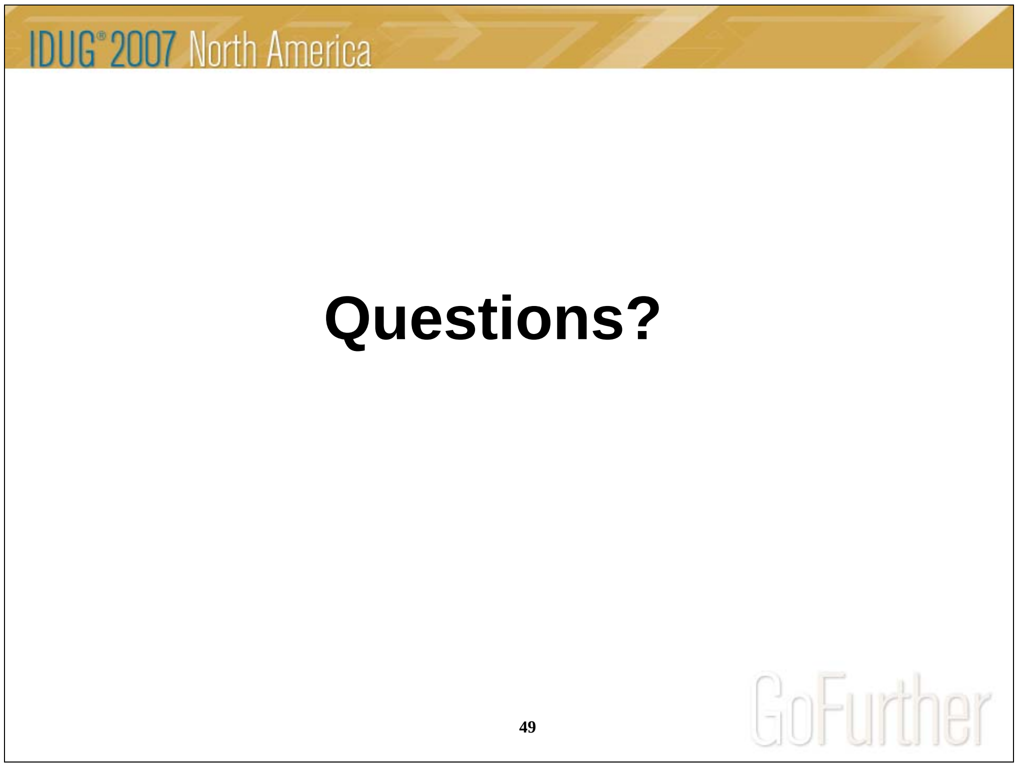
Trending data can be especially useful for distributed workloads, since the DB2 for z/OS personnel have much less control over its characteristics or volume. Summarizing statistics and accounting data in DB2 tables can provide data for queries.

Sometimes distributed monitoring and traces may be needed to diagnose a problem. There have been several good presentations on that topic.

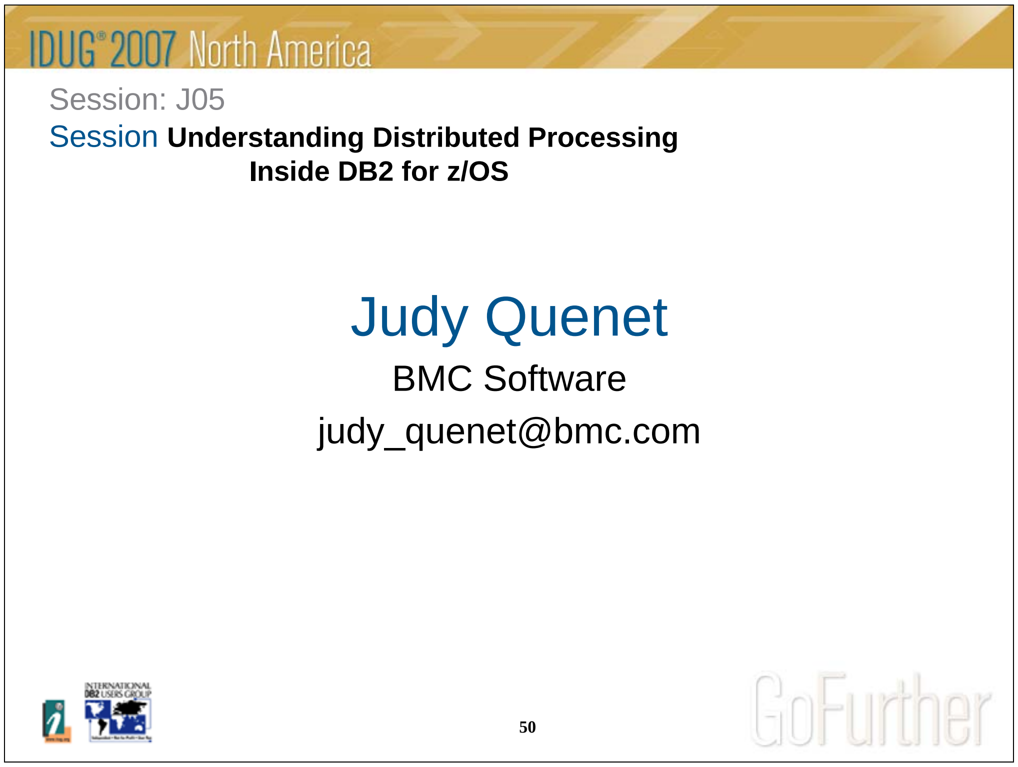
z/OS reporting from SMF records can also be helpful to gain insight into WLM and enclave usage by DDF. Such online views and reports can be found in MAINVIEW for z/OS and CMF, and also in RMF or other z/OS performance monitors.

- The SMF 30 record contains service for all completed or in-flight enclaves since the previous record was cut. It provides resource consumption information at the address space level. It is not granular enough to measure CPU usage by enclave. You should use the DB2 accounting records for that.

- The SMF 72 record is cut for each service class period and report class. It provides information about the goals set for your enclaves and their actual measured values for their particular service class.



Questions?



IDUG® 2007 North America

Session: J05

Session **Understanding Distributed Processing Inside DB2 for z/OS**

Judy Quenet

BMC Software

judy_quenet@bmc.com

50

GoFurther