WLM Management of DDF Work:
What can you do and what has changed?

Scott Chapman

Enterprise Performance Strategies, Inc.

Scott.Chapman@EPStrategies.com

Contact, Copyright, and Trademarks

Questions?

Send email to performance.questions@EPStrategies.com, or visit our website at https://www.epstrategies.com or http://www.pivotor.com.

Copyright Notice:

Trademarks:

Abstract

Managing DDF workloads can be somewhat tricky because DDF can be quite variable. You may have an application where each row inserted is an extremely short transaction. Or you may have QMF users who submit queries that run for hours or even days. And you probably have a lot of DDF work that falls in-between but which varies in both importance and intensity. Historically, the full range of WLM goal types and controls were available to help manage this challenging workload. But now, in some situations, that’s no longer the case. If you use DDF and want to better understand the WLM options for managing the work, and especially if you’re using (or may use) high performance DBATs, you should attend this presentation to better understand how to take control of DDF.

EPS: We do z/OS performance…

- Pivotor - Reporting and analysis software and services
  - Not just reporting, but analysis-based reporting based on our expertise

- Education and instruction
  - We have taught our z/OS performance workshops all over the world

- Consulting
  - Performance war rooms: concentrated, highly productive group discussions and analysis

- Information
  - We present around the world and participate in online forums

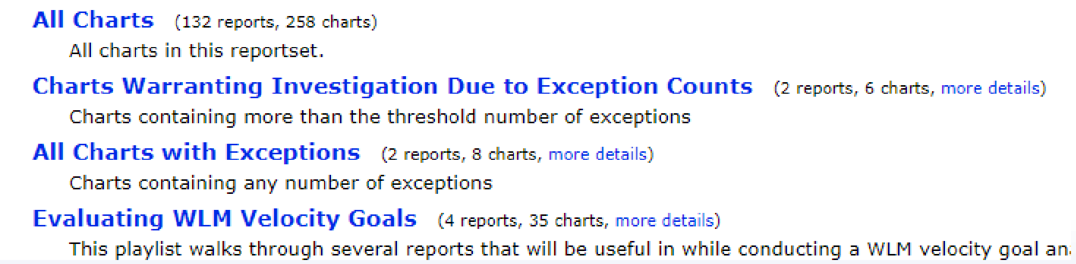
z/OS Performance workshops available

During these workshops you will be analyzing your own data!

- Essential z/OS Performance Tuning
  - March 20-24, 2023
- WLM Performance and Re-evaluating Goals
  - TBD 2023
- Parallel Sysplex and z/OS Performance Tuning
  - February 7-8, 2023
- Also… please make sure you are signed up for our free monthly z/OS educational webinars! (email contact@epstrategies.com)

Like what you see?

- The z/OS Performance Graphs you see here come from Pivotor™
- If you don't see them in your performance reporting tool, or you just want a free cursory performance review of your environment, let us know!
  - We're always happy to process a day's worth of data and show you the results
  - See also: http://pivotor.com/cursoryReview.html
- We also have a free Pivotor offering available as well
  - 1 System, SMF 70-72 only, 7 Day retention
  - That still encompasses over 100 reports!



All Charts (132 reports, 258 charts)
All charts in this reportset.

Charts Warranting Investigation Due to Exception Counts (2 reports, 6 charts, more details)
Charts containing more than the threshold number of exceptions

All Charts with Exceptions (2 reports, 8 charts, more details)
Charts containing any number of exceptions

Evaluating WLM Velocity Goals (4 reports, 35 charts, more details)
This playlist walks through several reports that will be useful in while conducting a WLM velocity goal an

Agenda

- The Story of DB2 and DDF
- Traditional WLM options for managing DDF
- Updated WLM management limitations with DDF High-Perf DBATs

The Story of DB2 and DDF

“Illustrated” by AI (Stable Diffusion Text to Image AI)

In the beginning…

There was DB2 and it was good, and all systems wanted to move across the waters and access the data stored in the walled garden of the mainframe.

And so the Distributed Data Facility arose to allow the plebeians' SQL to run in DB2.

And all rejoiced for they could access the holy data.

Except for the performance analysts who were distraught, for they could see that the DDF SQL ran within the DB2 “DIST” address space at the same priority of the address space and they were much afraid.

And verily it did happen that the plebeians wrote poor SQL which ran at the same priority as the senators' SQL and the senatorial SQL was delayed, thus angering the senators.

So the senators appealed to Emperor Maximus Virtuous Servitus:

"The great empire depends on the timely execution of our most glorious and efficient SQL and it must have a higher priority than the plebeian SQL, some of which bears the stench of automatic generation and runs overlong and consumes valuable resources at the expense of more deserving workloads."

Hearing their complaint, Emperor MVS (the 5.2nd of his name) decreed that there should be enclaves and all distributed SQL would thereafter run in enclaves and in doing so be managed individually and separately from the "DIST" address space.

And so it came to be that the favored senators' SQL ran with great abandon at a priority above the plebeian's SQL and the stinky SQLs that had aged poorly ran behind others, no longer delaying the most glorious and efficient SQL.

The senators’ SQL ran in milliseconds.

The plebians SQL ran in seconds.

While Quericus Maximus Festivus wasn’t completely happy, his stinky SQL eventually finished.

The empire was at peace for over a quarter century.

But was it the end?

TLDR…

- DDF SQL originates from a network connection to DB2
  - Usually a TCP connection from a different OS
- DDF SQL run in enclaves which allows the SQL transactions to be individually managed
- Also: relying on AI to illustrate your story is far from perfect, but fascinating that it works as well as it does

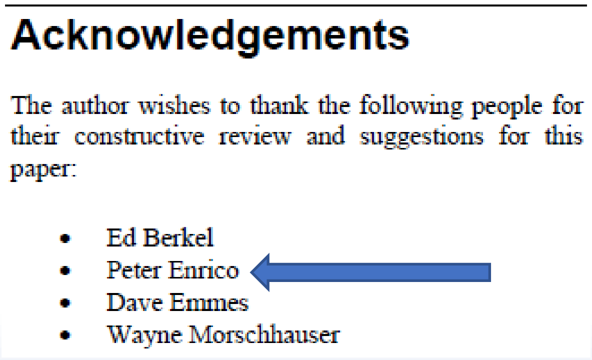
John Awre said it succinctly

Enclaves allow the *management of individual transactions* flowing through address spaces, something that simply has never been possible before. Since MVS is aware of and has access to each transaction, they can be classified individually and most importantly *each transaction is subject to period switch*. This means that you can separate out the long-running CPU killers from the shorter requests in the same manner that most installations already employ to control batch, by period-level controls.

Each enclave is a single transaction, which starts when the enclave is created and ends when the enclave is deleted. DDF creates an enclave for an incoming request when it detects the first SQL statement and deletes the enclave at SQL COMMIT, thus a DDF enclave transaction consists of a single SQL COMMIT scope.

In WLM goal mode, all goal types are valid for enclaves.

From whitepaper “Preemptible SRBs”, John Arwe
Exact date unknown, c. introduction of MVS 5.2 (1995)
Emphasis mine



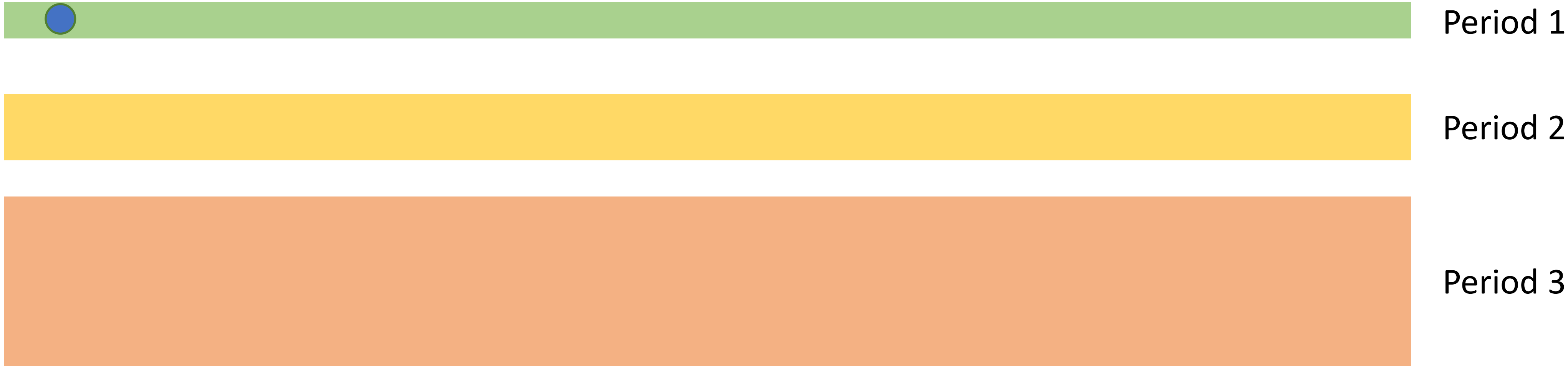
Acknowledgements

The author wishes to thank the following people for their constructive review and suggestions for this paper:

- Ed Berkel
- Peter Enrico
- Dave Emmes
- Wayne Morschhauser

SUs and Period Transitions

- For Service Classes with multiple periods, work transitions between periods as it consumes resources (CPU)



- Each period has its own importance and goal
- So we can automatically adjust the management of long-running work
  - E.G. lower the importance and relax the goal so the hogs don't trample the hummingbirds
  - Very useful when a workload contains a mix of light and heavy work

Managing DDF work with WLM

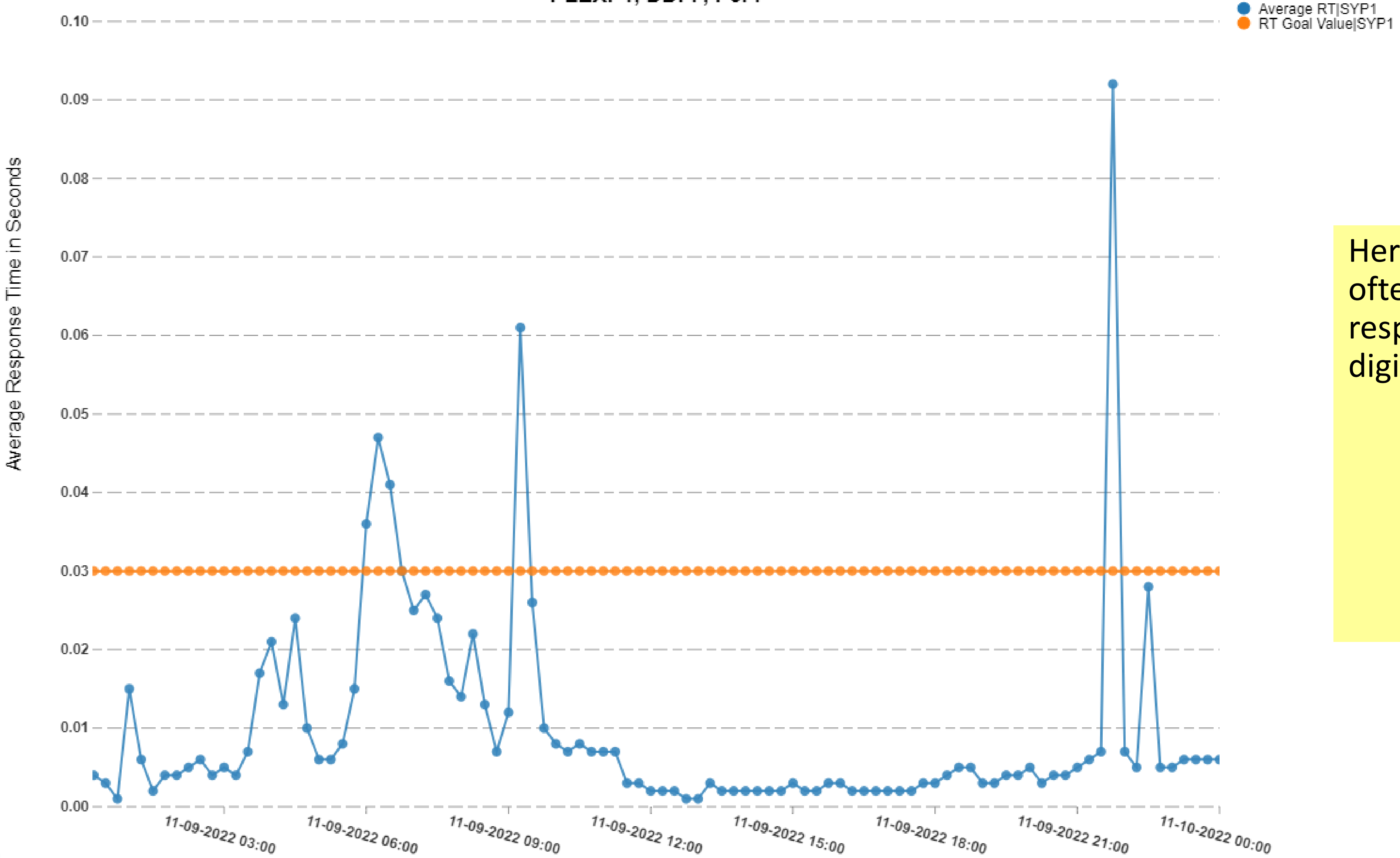
DDF is a bit tricky…

- SQL transactions can be quite variable
  - Can be 1 SQL statement or could be multiple SQL statements
  - Could be well-controlled and well-written SQL or could be random adhoc SQL
  - Could access very little data from memory or could read GBs from disk
  - Could run in milliseconds or seconds or minutes or hours or…
- SQL may come from applications unknown to the performance analyst
  - With an appropriate JDBC or ODBC driver and appropriate authorization, all sorts of things can reach into DB2
    - Basically almost anything that can generically read from a relational data source
- Some DDF SQL may be for "online" users, others may be "batch" work
- Poorly managed DDF can cause problem for other workloads on the system

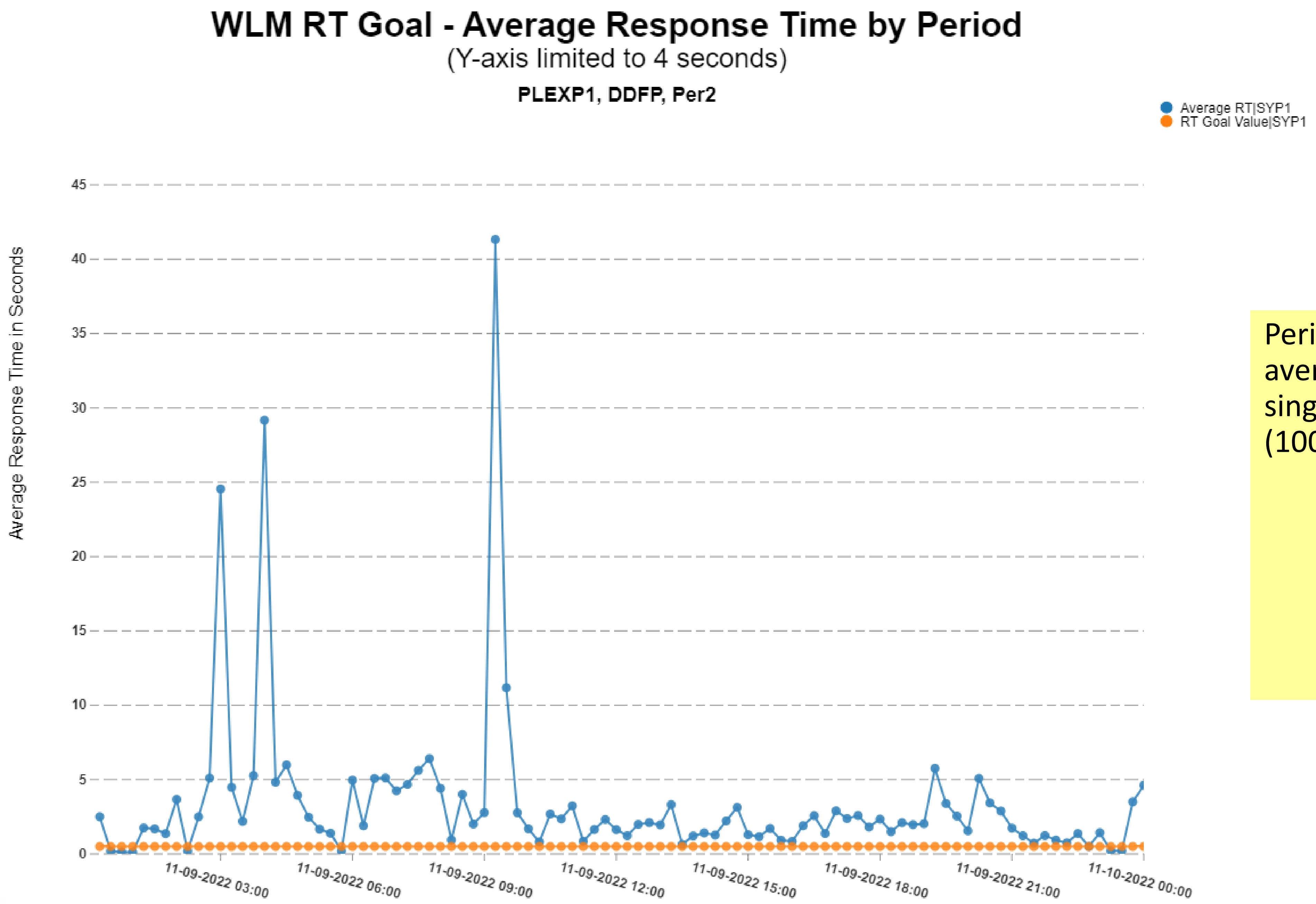
WLM RT Goal - Average Response Time by Period

(Y-axis limited to 4 seconds)

PLEXP1, DDFP, Per1

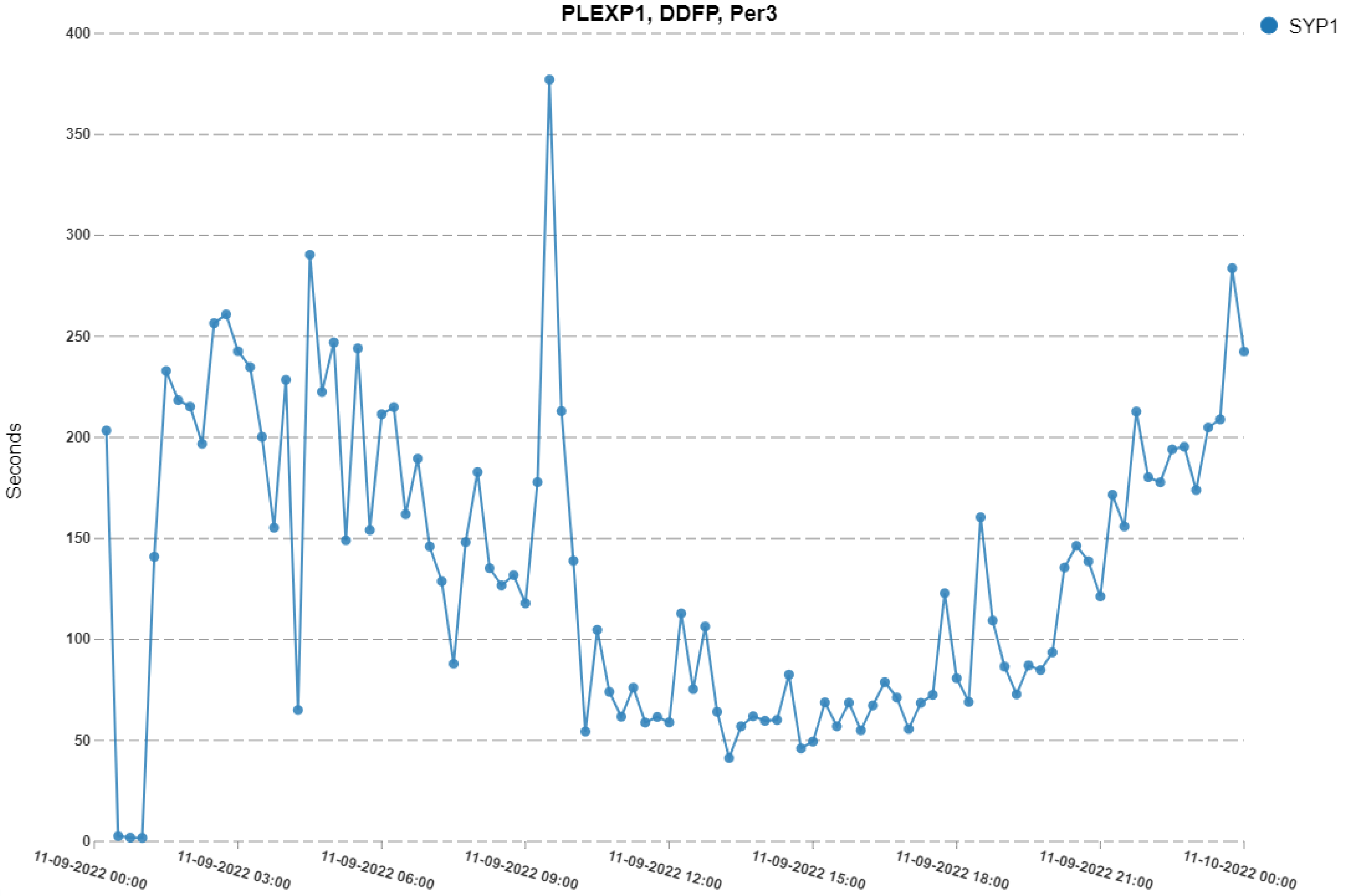


Here period 1 of DDFP is often getting an average response time of single-digit milliseconds.



Period 2 of DDFP is averaging more like single-digit seconds (1000x longer)

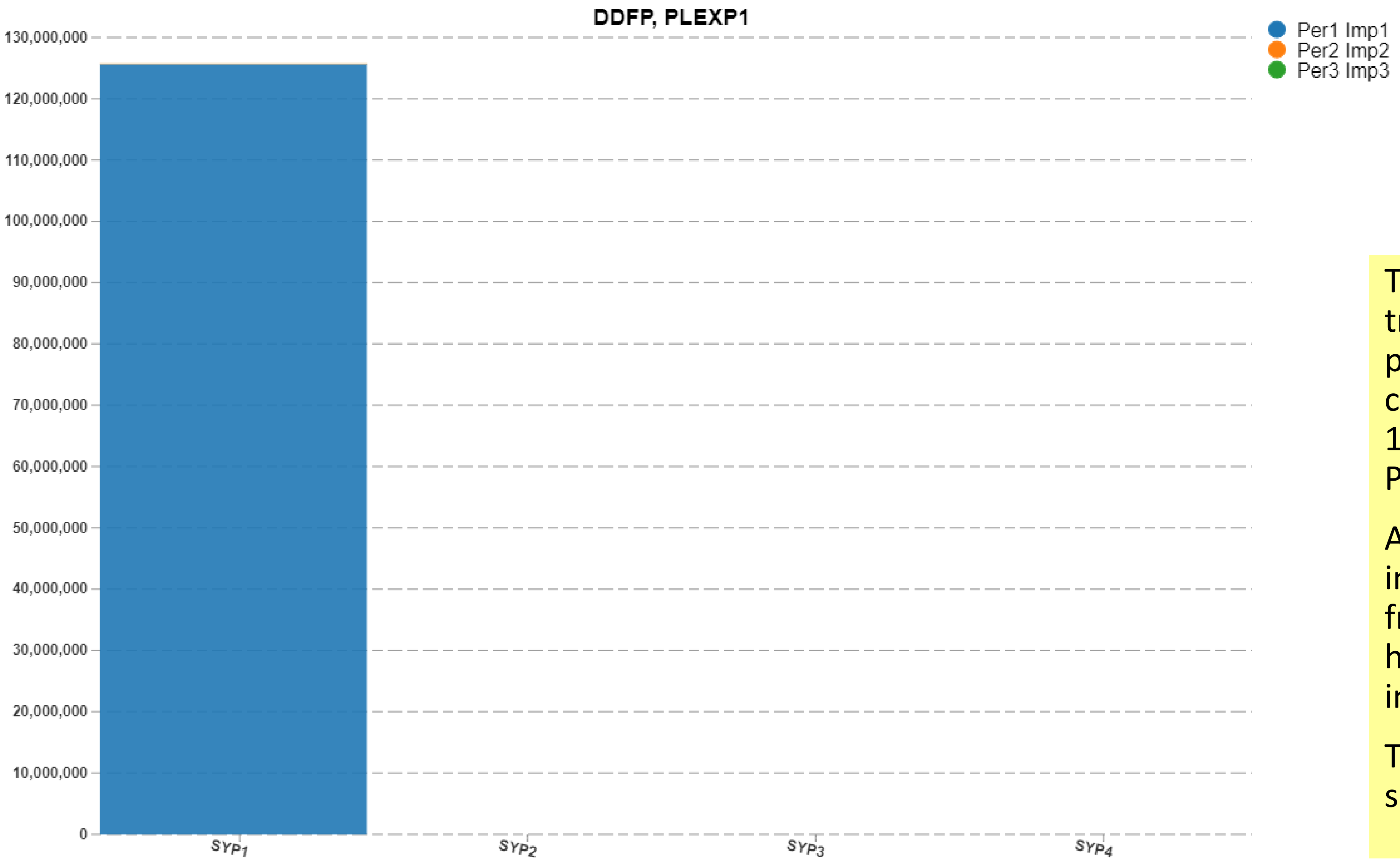
Average Response Times for Non RT Goal Periods



And period 3 for this DDF work is usually above 50 seconds and often above 100 seconds.

Long transactions can be 100,000x longer than short transactions.

WLM Multiple Period - Total Ended Transactions All Intervals

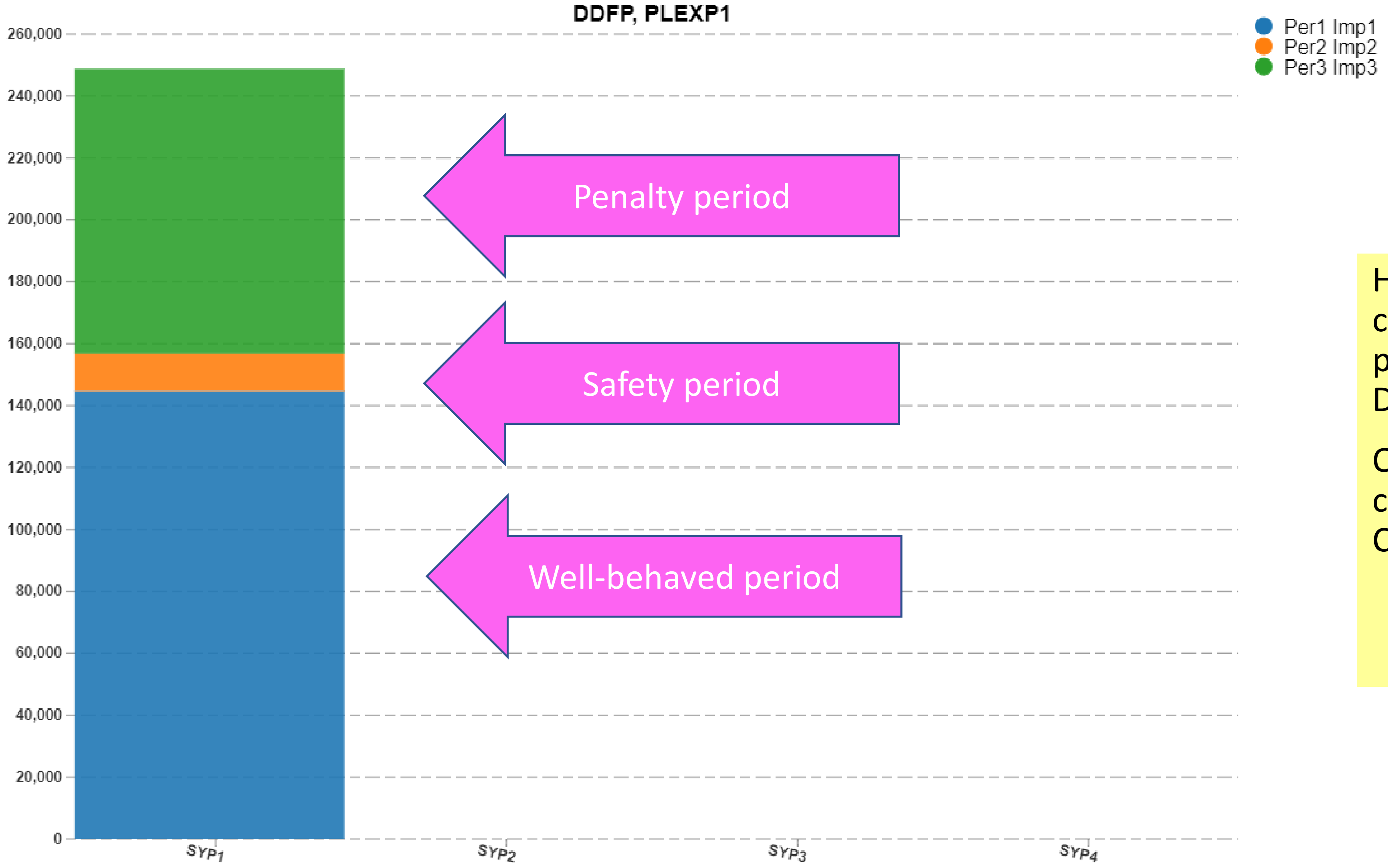


The vast, vast majority of transactions end in period 1. (But in this case there may be 10s-100s of thousands in P2/P3.)

And that 100,000x increase in elapsed time from P1 to P3 probably has a similar correlation in CPU time.

The long tail may have a significant stinger.

WLM Multiple Period - Total CPU Seconds All Intervals



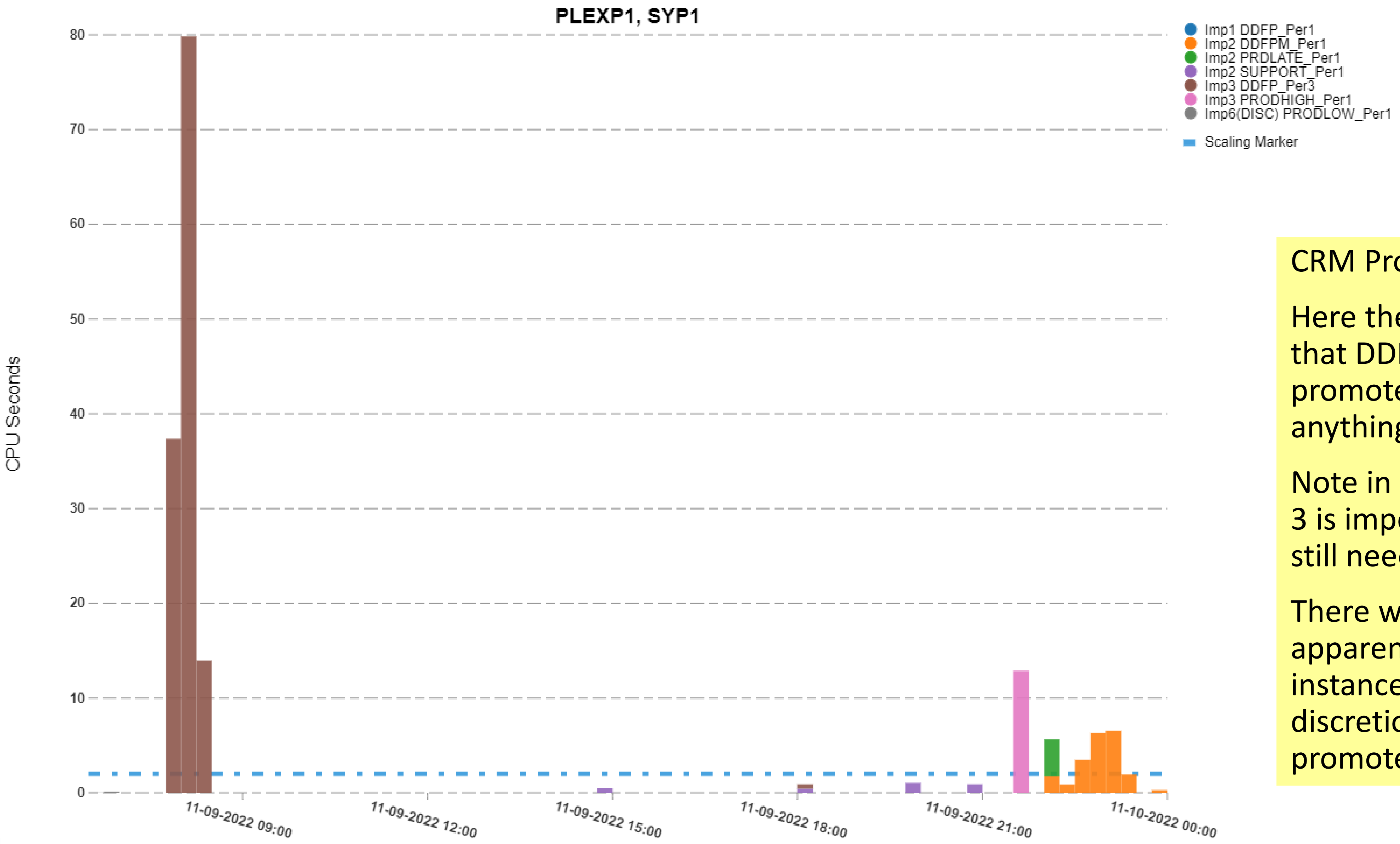
Here we see indeed P3 consumed a significant portion of the overall DDFP CPU.

Often, we'll see P3 consuming even more CPU than P1.

Penalty Period FUD

- Some people may say “Don’t age DDF to low importance!” or “Never run DDF as discretionary!”
  - Those people may love DB2 too much (or possibly remember old problems too well)
- Usually the worry is “that work might hold a lock or latch that might hold up more important work”
  - The same is true for batch jobs using DB2 too! Batch is commonly run at low importance or discretionary.
- Importances are relative: you can’t judge an importance by its number without understanding the work running at all importances
- Modern DB2 and WLM work together to promote work to resolve locking/latches
  - Chronic Resource Contention allows DB2 to signal to WLM that specific work should be promoted to allow a lock to be released
  - Blocked Workload Promotion promotes any work that hasn’t been able to get CPU in a specified time period (usually 1-5 seconds) so that it can release locks/latches

WLM CPU - Promoted CRM CPU Seconds



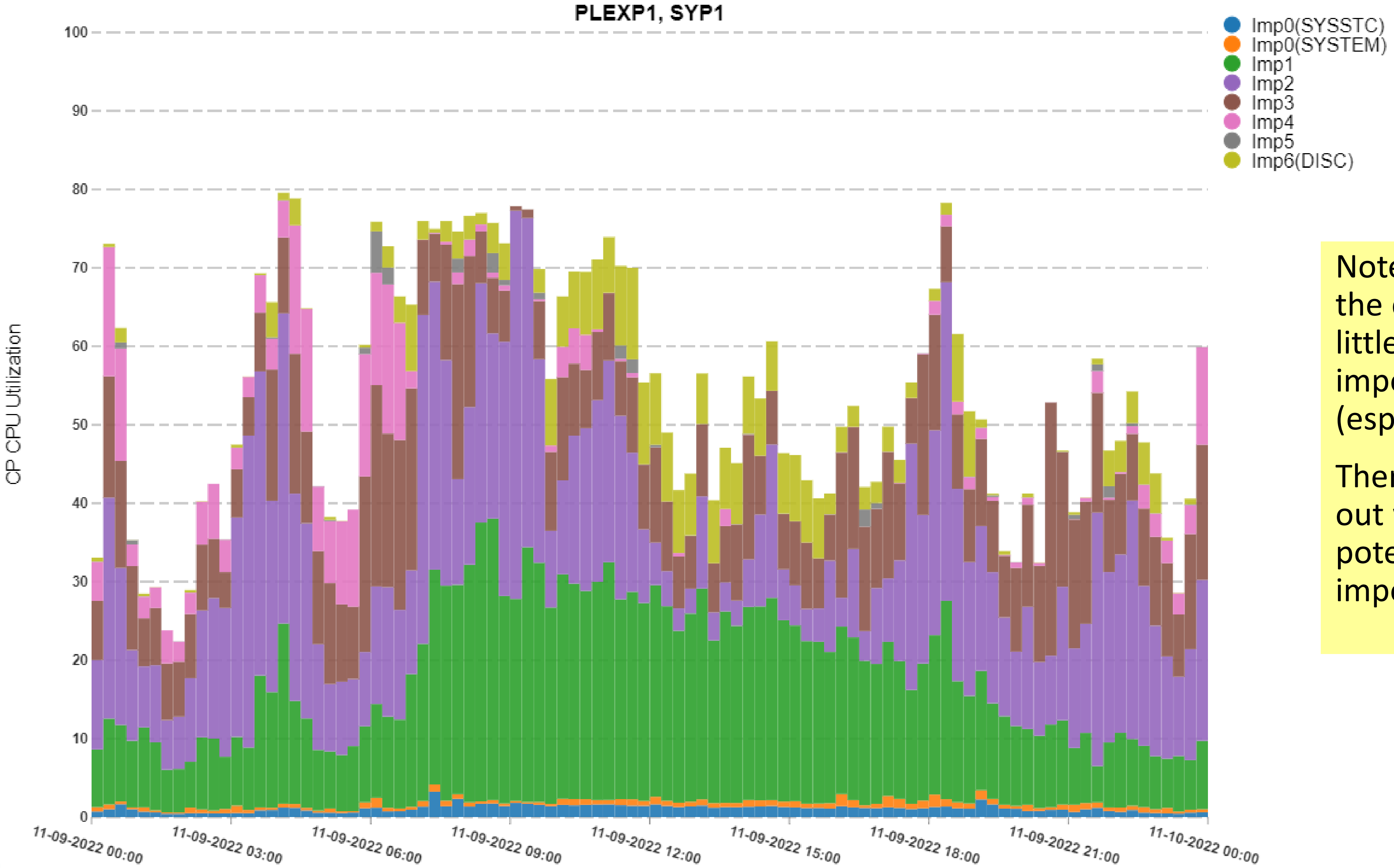
CRM Promotion happens

Here the third period of that DDFP work got promoted more than anything else.

Note in this case period 3 is importance 3, and it still needed help.

There was even apparently some instances of discretionary work being promoted.

WLM Importance - CP CPU Utilization (CP + zAAP on CP + zIIP on CP)



Note that for much of the day there's relatively little work in importances 4 and (especially) 5.

There's room to spread out work to make better potential use of all importance levels.

Understand the application

- What work is more important?
  - Extremely short transactions may really be a batch process doing singleton SQLs
    - E.G. millions of single row inserts, each a separate transaction
  - "Long" transactions may just be because your customers don't call in sorted order
    - First access for a customer call will be random and will likely not be in the buffer (absent big buffers)
    - That first random access may have to do a bunch of I/O to fetch the customer data
    - The time to fetch the customer info may directly impact customer satisfaction
- Managing DDF without understanding the application risks making incorrect decisions about the relative importance of work
  - The duration of a transaction is not necessarily an indication of importance
  - Getting trivial transactions in and out quickly may be good, but may not be optimal
- Use separate SCs for batch-like vs. online-like work

DDF Management Recommendations

- Don’t be afraid of using multiple DDF Service Classes!
- Make liberal use of Report Classes (e.g. by authid)
  - Helps determine what application is doing how many transactions and consuming how much CPU: all from the SMF 72 records without having to look at the voluminous 101s
  - For things like QMF with a large number of adhoc users, may look for correlation IDs
- Treat your default DDF Service Class like batch
  - Default should generally not be “like online users”
  - Also helps catch new DDF exploiters to get them classified to a good RC
- Consider 2-3 period SCs, except for work that is well-known and well-behaved
  - 3rd period may be a “penalty period” for those outliers
- Consider RT goals for first 1 or 2 periods
  - Generally easier to describe, monitor, and relate to application performance

What has changed?

Upending >25 Years of DDF Management

Enclaves allow the *management of individual transactions* flowing through address spaces, something that simply has never been possible before. Since MVS is aware of and has access to each transaction, they can be classified individually and most importantly *each transaction is subject to period switch*. ~~This means that you can separate out the long running CPU killers from the shorter requests in the same manner that most installations already employ to control batch, by period-level controls.~~

Each enclave is a single transaction, which starts when the enclave is created and ends when the enclave is deleted. ~~DDF creates an enclave for an incoming request when it detects the first SQL statement and deletes the enclave at SQL COMMIT, thus a DDF enclave transaction consists of a single SQL COMMIT scope.~~

~~In WLM goal mode, all goal types are valid for enclaves.~~

This is now the situation if your DDF work uses High Performance DBATs!
(With DB2 APAR PH34378)

Technically: multi-period service classes and RT goals still are allowed but they will not work as expected!
This could be a surprise when you apply the DB2 maintenance that made this change.

DB2 Details for non-DB2 People…

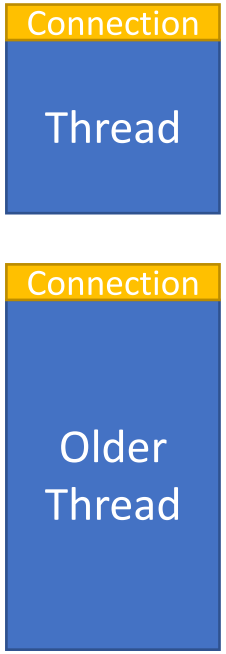
Special thanks to Mark Rader for helping me with this!

But all opinions and conclusions are mine and not reflective of the opinions of Mark or his employer!

What
The
Frak?

WTF is a DBAT?

- Database Access Thread
- How DDF traffic connects to DB2
  - Vs. CICS/Batch/etc. which connect directly from those address spaces
- Consists of the thread (large) and connection (tiny)
- As a thread is used, it tends to grow and become “fat”
  - So can’t keep them around forever
- Threads have to have a place to run: for DDF that’s an enclave
  - DB2 creates the enclave to run the DBAT in (since MVS 5.2, DB2 v4)



Connection

Thread

Connection

Older Thread

DB2 zPARM Options

- CMTSTAT – Commit Status – what happens when the application does a commit?
  - ACTIVE – connection holds the DBAT which remains active and unusable by any other thread
  - INACTIVE – separate the connection from the DBAT and allow the DBAT to be reused (generally, may just shrink)
  - INACTIVE has long been the default and recommendation because it reduces storage usage at the slight cost of re-associating the connection to a DBAT upon the next SQL from the connection
- POOLINAC – Approximate time a DBAT can remain inactive in the pool before it is terminated
  - Defaults to 120 seconds (but thread age is only checked periodically (2 min?))

Q) WTF is zPARM?
A) DB2 config options

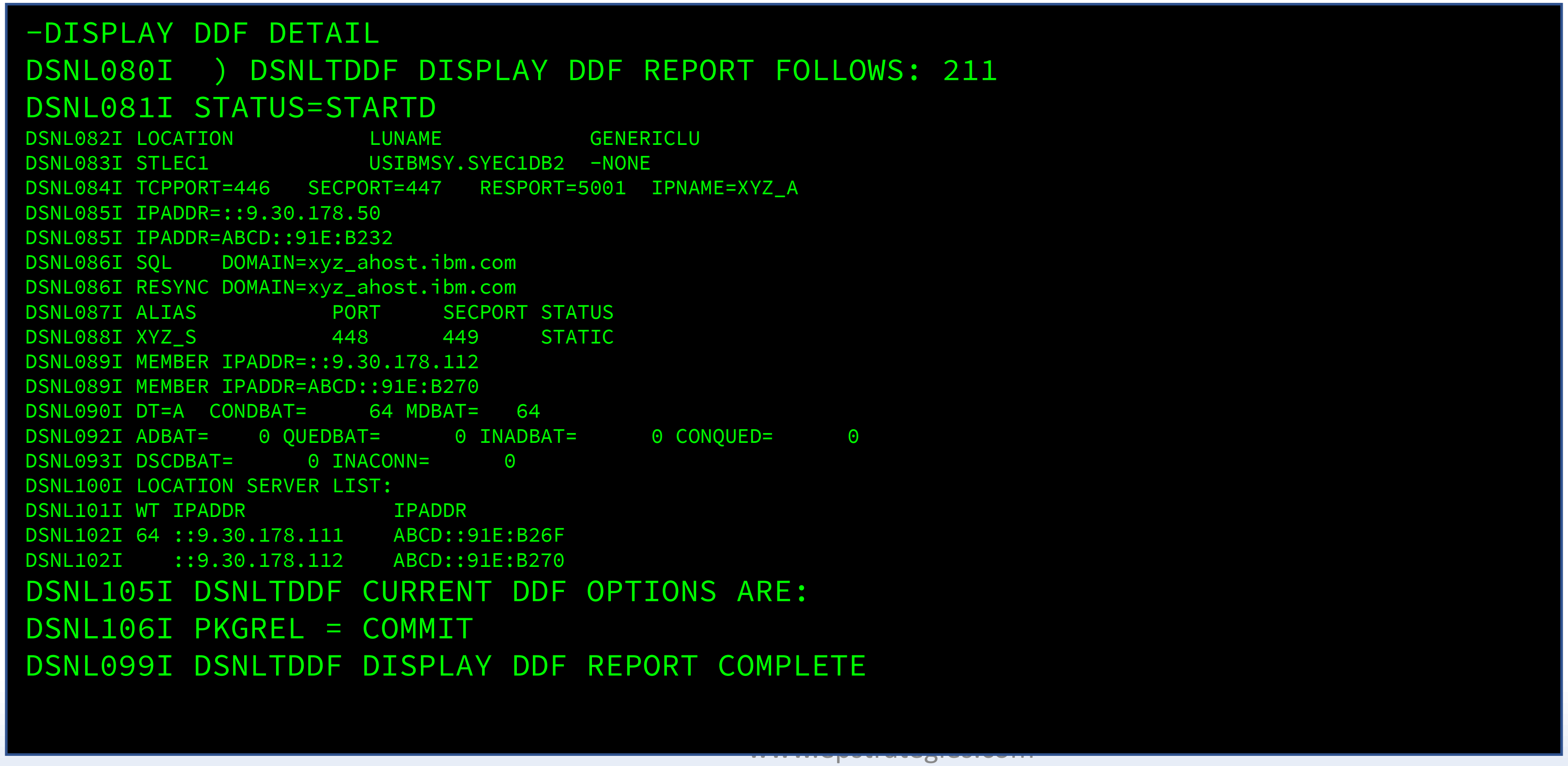
Bind Options

- Release – what happens to resources/locks at commit
  - COMMIT – releases locks and resources when the application issues a commit or rollback in most cases except if there are held cursors, keepdynamic(yes), declared global temp table (DGTT) that is not dropped, or LOB locators
  - DEALLOCATE – only release when the thread terminates
- KEEPDYNAMIC – keep dynamic SQL past commit point
  - YES – do that, so DB2 doesn't have to re-prepare the SQL if it's reused
    - Thread stays associated with the connection, but accounting records cut & enclave deleted
  - NO – don't keep them, reprepare dynamic SQL if it's reused
    - Default and generally recommended

Q) WTF is "Bind"?
A) Very basically: DB2 records the options, description, and (potentially) SQL access paths for the application program in the DB2 system catalog so the program can be used.
Also note: "package" = program, "plan" = group of packages for an application.

DDF Option

- PKGREL (set by MODIFY DDF command)
  - BNDOPT or BNDPOOL – *Allows* for high performance DBATs
  - COMMIT – Disables high performance DBATs
- Can find with Display DDF DB2 command



```
-DISPLAY DDF DETAIL
DSNL080I  ) DSNLTDDF DISPLAY DDF REPORT FOLLOWS: 211
DSNL081I STATUS=STARTD
DSNL082I LOCATION           LUNAME            GENERICLU
DSNL083I STLEC1             USIBMSY.SYEC1DB2  -NONE
DSNL084I TCPPORT=446   SECPORT=447   RESPORT=5001  IPNAME=XYZ_A
DSNL085I IPADDR=::9.30.178.50
DSNL085I IPADDR=ABCD::91E:B232
DSNL086I SQL    DOMAIN=xyz_ahost.ibm.com
DSNL086I RESYNC DOMAIN=xyz_ahost.ibm.com
DSNL087I ALIAS           PORT     SECPORT STATUS
DSNL088I XYZ_S           448      449     STATIC
DSNL089I MEMBER IPADDR=::9.30.178.112
DSNL089I MEMBER IPADDR=ABCD::91E:B270
DSNL090I DT=A  CONDBAT=     64 MDBAT=   64
DSNL092I ADBAT=    0 QUEDBAT=      0 INADBAT=      0 CONQUED=      0
DSNL093I DSCDBAT=      0 INACONN=      0
DSNL100I LOCATION SERVER LIST:
DSNL101I WT IPADDR                IPADDR
DSNL102I 64 ::9.30.178.111     ABCD::91E:B26F
DSNL102I    ::9.30.178.112     ABCD::91E:B270
DSNL105I DSNLTDDF CURRENT DDF OPTIONS ARE:
DSNL106I PKGREL = COMMIT
DSNL099I DSNLTDDF DISPLAY DDF REPORT COMPLETE
```

High Performance DBATs

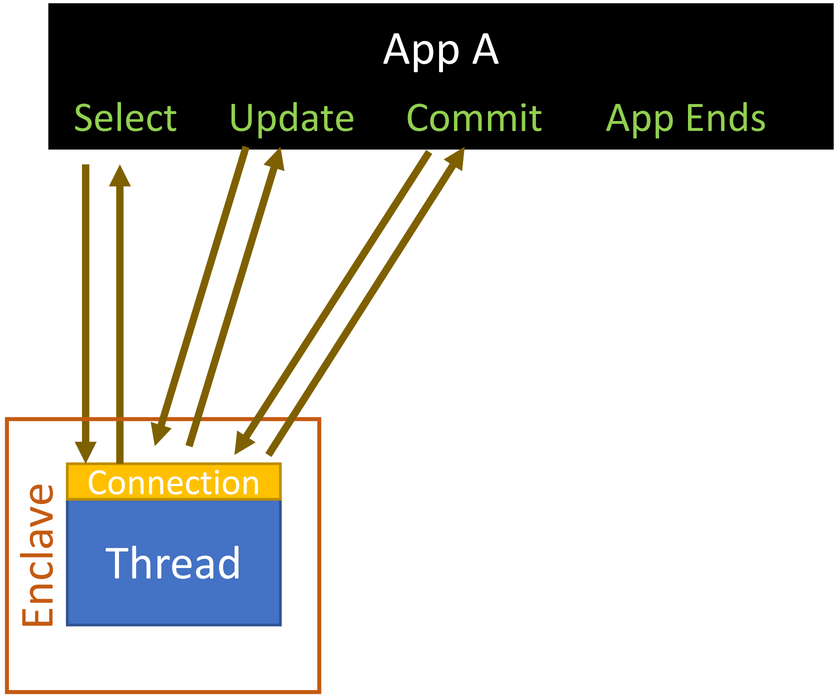
- Allows some resources (such as table space intent locks and EDM pool elements) to be retained in the DBAT across multiple transaction executions, reducing the CPU overhead for subsequent transactions that access those same resources
- Triggered by the thread calling a package bound with RELEASE=DEALLOCATE, KEEPDYNAMIC(NO) and CMTSTAT(INACT) and PKGREL=BNDOPT in effect
- DDF system packages are probably bound with RELEASE=COMMIT
  - But sites can change this
- Possible for thread to end up running a different package due to invoking a stored procedure or user defined function
  - That package could have RELEASE=DEALLOCATE, causing the thread to become a HiPerf DBAT even if the original package was RELEASE=COMMIT

WTF? Say the last part again?

- You think your DDF is not using HiPerf DBATs, but…
- An application programmer adds a call to a SP or UDF bound with RELEASE=DEALLOCATE
- That DBAT may become a HiPerf DBAT (if the other conditions satisfied)
- And will remain so for the remainder of the SQL transactions it handles
  - So some of the WLM transactions are SQL transactions, some are not

Life and times of DBATs, Connections, Enclaves:

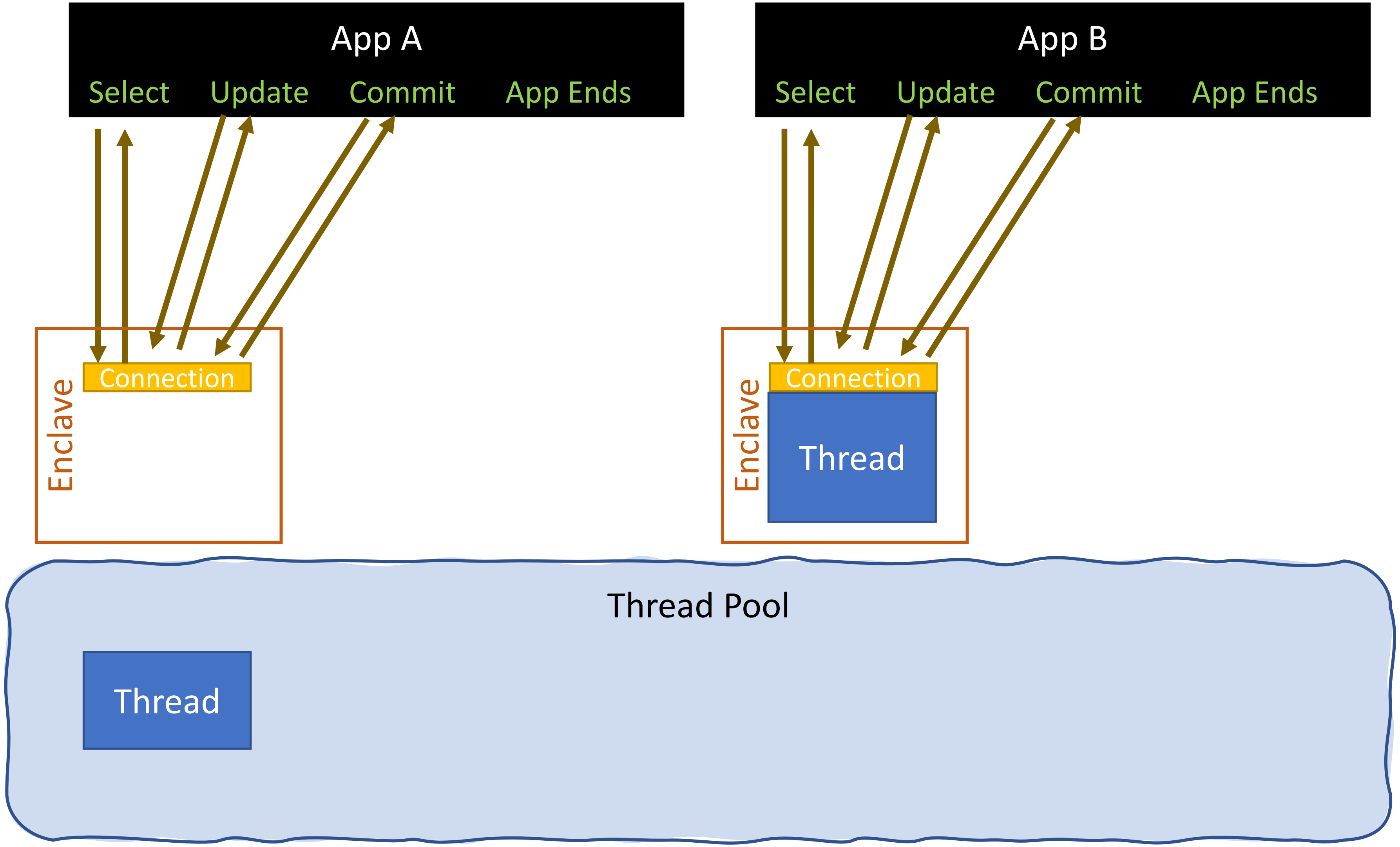
CMTSTAT = ACTIVE (Not default, not recommended)



- DB2 transaction <> WLM Transaction
- WLM Transaction = life of app
- Probably shouldn't use RT goals
- Could potentially use multi-period goals to degrade the app the longer it's consuming resources

Life and times of DBATs, Connections, Enclaves:

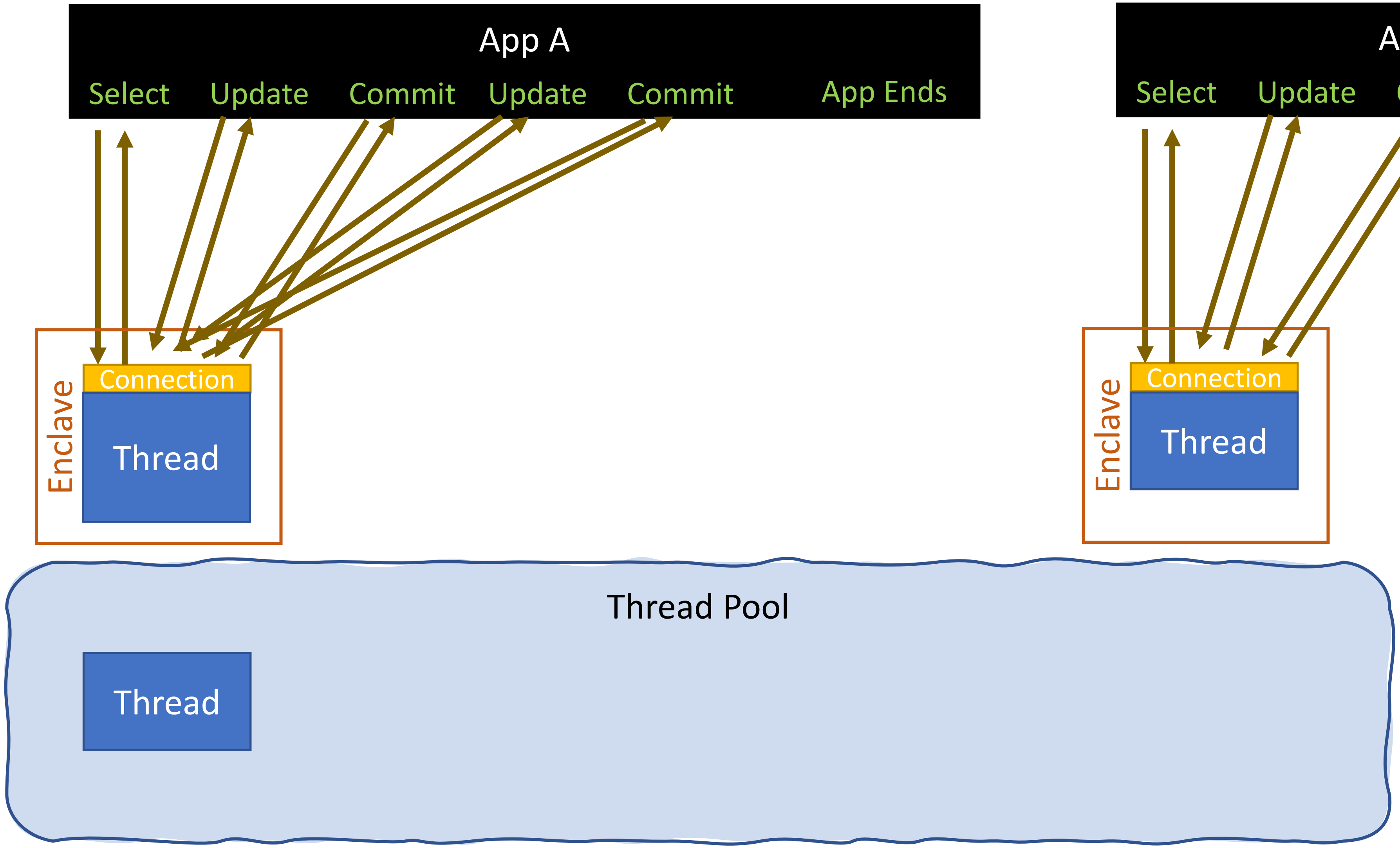
CMTSTAT = INACTIVE (Without HiPerf DBATs)



- WLM Transaction = DB2 transaction
- Can use RT or velocity goals
- Could use multi-period goals to better manage the transaction the longer it's consuming resources
- Small number of DBATs support large number of connections
- Because they fatten up, threads destroyed after 200 uses or 120 seconds (default) (or some other triggers)
  - 500 uses in Db2 13

Life and times of DBATs, Connections, Enclaves:

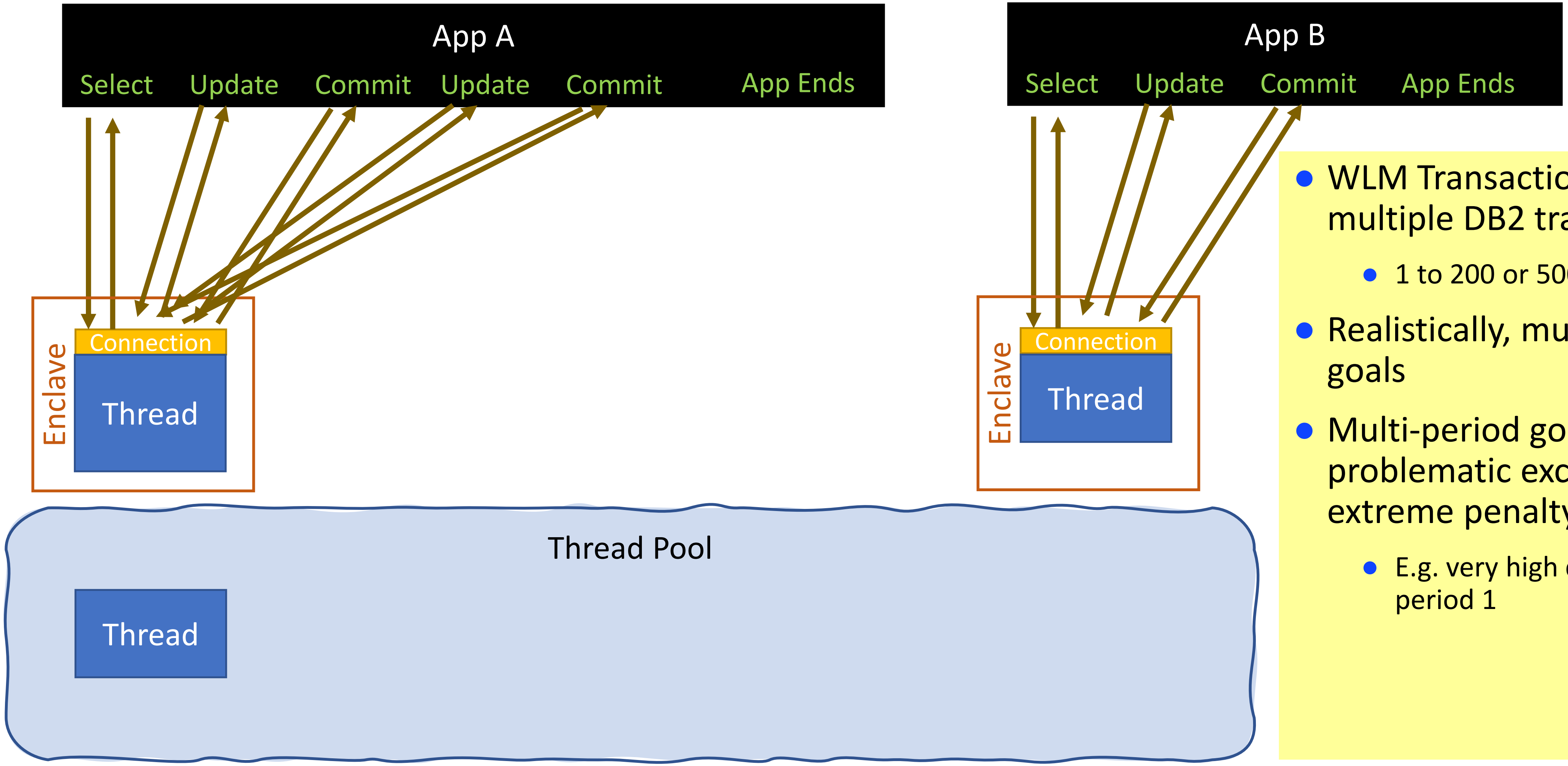
CMTSTAT = INACTIVE (with HiPerf DBATs w/o PH34378)



- WLM Transaction = DB2 transaction
- Can use RT or velocity goals
- Could use multi-period goals to better manage the transaction the longer it's consuming resources
- May need more DBATs since they're not returned to the pool as quickly
- Threads destroyed as previously

Life and times of DBATs, Connections, Enclaves:

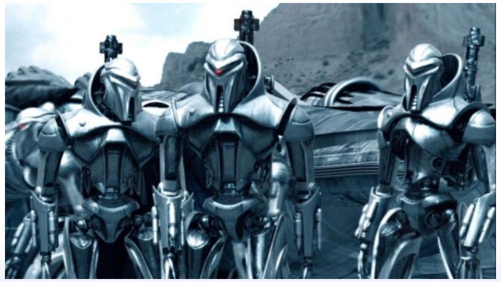
CMTSTAT = INACTIVE (HiPerf DBATs w/ PH34378)



- WLM Transaction = multiple DB2 transactions
  - 1 to 200 or 500 (DB2 v13)
- Realistically, must velocity goals
- Multi-period goals are problematic except as an extreme penalty period
  - E.g. very high duration on period 1

So why the change and why do we care?

- Apparently in some specific high-volume situations, retaining the enclave and not allocating a new one for each transaction provides performance benefit
  - Seems like a small benefit, likely only visible in very high-volume situations
  - One selling point of enclaves in 1995 was that they were light-weight!
- Would have been nice if DB2 would have added a flag/option for this behavior but instead they just changed it for everybody!
- So now WLM administrators need to know what DDF work is potentially using high-perf DBATs and change those to single-period velocity goals
  - Which limits the flexibility of managing that DDF work
  - This may be of limited consequence if that DDF work is well-behaved



How much of your DDF work is "well-behaved" vs "not so well-behaved"??

HighPerf DBATs for some DDF work?

- Note that it is possible that different DDF work could go to different collections and packages
  - How this is done is beyond the scope of this presentation!
- So if you don’t have PKGREL=COMMIT, some DDF work might be HiPerf DBAT and some might not be

Help to report on actual DB2 transactions

- PH41024 Db2 support for WLM OA61811
- In combination, adds fields to SMF 72 records to report on actual DDF transactions from the DB2 perspective instead of number of enclaves
- But does not change how the work can be managed!
- So you’ll still be able to get a count of DB2 transactions for accounting purposes, but … that doesn’t help you manage the work
- At least we should be able to tell that HiPerf DBATs are in use by comparing the DB2 transaction count to the WLM transaction count

Should you be using HiPerf DBATs?

- Not universally
  - There are DB2 considerations (e.g. MAXDBAT, memory, holding resources)
  - There are now WLM management considerations too
- For high-volume, well-behaved applications: maybe
  - But those now need to have single-period velocity goals

Summary

- Understand your applications' use of DDF and set appropriate Service Class goals
- Use multiple Report Classes for capturing DDF consumption by application
- If you're using High Performance DBATs, make sure you're using single-period velocity goals for that work
    - An extreme penalty period may be possible, but your ability to manage DDF short, medium, and long transactions is going to be compromised
- If you're not using HiPerf DBATs, make sure your DB2 folks know to talk to your WLM folks before enabling them
- IBM DB2 Support should talk to WLM Support more often

Questions??

EPS