Exploring Locking and Locking Measurements on z/OS

Bob, Scott, and Peter

Email: Peter.Enrico@EPStrategies.com
Email: Scott.Chapman@EPStrategies.com
Email: tetterrogers@gmail.com

Enterprise Performance Strategies, Inc.
3457-53rd Avenue North, #145
Bradenton, FL 34210
http://www.epstrategies.com
http://www.pivotor.com

Voice: 813-435-2297
Mobile: 941-685-6789

z/OS Performance Education, Software, and Managed Service Providers

Creators of Pivotor®

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:

Enterprise Performance Strategies, Inc. presentation materials contain trademarks and registered trademarks of several companies.

The following are trademarks of Enterprise Performance Strategies, Inc.: **Health Check®, Reductions®, Pivotor®**

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries: IBM®, z/OS®, zSeries®, WebSphere®, CICS®, DB2®, S390®, WebSphere Application Server®, and many others.

Other trademarks and registered trademarks may exist in this presentation

Abstract

Exploring Locking and Locking Measurements on z/OS

Serialization of resources on z/OS is such a fundamental need, that locking is one of the foundational concepts of the MVS platform. There are CMS locks, local locks, CML locks, GRS latches, suspend locks, spin locks, and more.

During this webinar, ***Peter Enrico, Scott Chapman, and Bob Rogers*** will explore the different lock types and any measurements available to help us understand the locking activity of our z/OS environments.

EPS: We do z/OS performance…

- We are z/OS performance!

- Pivotor
  - Performance reporting and analysis of your z/OS measurements
  - Example: SMF, DCOLLECT, other, etc.
  - Not just reporting, but cost-effective analysis-based reporting based on our expertise

- Performance Educational Workshops (while analyzing your own data)
  - Essential z/OS Performance Tuning
  - Parallel Sysplex and z/OS Performance Tuning
  - WLM Performance and Re-evaluating Goals

- Performance War Rooms
  - Concentrated, highly productive group discussions and analysis

- MSU reductions
  - Application and MSU reduction

Like what you see?

- Free z/OS Performance Educational webinars!
  - The titles for our Winter 2022 webinars are as follows:
    - ✓ *SMF Recording Options to Improve Your Performance Analysis*
    - ✓ *SMF 98 and 99: Pinpointing Transient Performance Problems*
    - ✓ *Exploring z/OS Processor Storage Measurements*
    - ✓ *Exploring PR/SM Physical and Logical CPU Utilization Measurements*
    - *Exploring Locking and Locking Measurements on z/OS (with Bob Rogers)*
    - *Exploring z/OS SMF 30 Address Space CPU Measurements*
    - *Exploring z/OS XCF Message Traffic Measurements*
    - *Exploring z/OS SMF 14 / 15 Records for Tape and DASD File Activity*
    - *Exploring z/OS WLM CPU Measurements: SUs vs CPU Secs vs APPL% vs Workload%*
    - *Exploring the Coupling Facility Lock Structure Measurements*
  - Dozens of past webinars are available at our website.

- If you want a free cursory 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

z/OS Performance workshops available

During these workshops you will be analyzing your own data!

- Essential z/OS Performance Tuning
  - October 3-7, 2022
- WLM Performance and Re-evaluating Goals
  - September 12-16, 2022
- Parallel Sysplex and z/OS Performance Tuning
  - August 8-12, 2022
- Also… please make sure you are signed up for our free monthly z/OS educational webinars! (email contact@epstrategies.com)

Serialization Delay Measurements

Performance and Locking

- Why is locking of interest to the performance analyst

- Locking with no lock contention is not very interesting
  - The combined acts of getting a lock, holding a lock, and then releasing the lock typically has little performance impact other than the execution of some instructions

- However, locking with contention can result in performance issues
  - For example, when one unit of work holds a lock exclusively, and the other unit of work needs that lock, then lock contention can exist
  - Depending on locking mechanism used, could result in
    - CPU spin time consumed ‘waiting’
    - or response time elongation due to ‘waiting’

Serialization Delay Measurements

- On z/OS there are several SMF records that provide insight into locking
  - SMF 72, subtype 5
    - S;96)4(25'19
      - CMS lock
      - Local lock
      - CML lock
        - Owner data
        - Requestor data
    - GRS latch
    - GRS enqueue
    - GRS QScan statistics
  - SMF 98
    - Spin lock
    - Suspend lock
      - Local lock
      - CML lock
- The purpose of this presentation is to provide overview of these lock types
  - Most locking measurements are used to show patterns to provide 'insights' to help debug
  - These measurements usually do not need to regularly be monitored

So… Let's learn about the types of locks from Bob Rogers!

Why locking Serialization?

- At the root of it all - Multitasking
  - Multitasking means sharing the CPU resources of a system among many different units of work (jobs, tasks, threads).
  - The tasks are typically run asynchronous to each other. That is, events as perceived by one task have no time order relationship to events as perceived by any other task.
  - Synchronization is the term for causing some event in one task to occur in a predictable order with respect to an event in another task.
  - Serialization is allowing only one program at a time do some particular thing. Synchronization requires serialization. (it is also true that serialization requires synchronization.)

Why locking Serialization?

- A Simple Example
  - A customer has two jobs that update the same master file. The two jobs might run "simultaneously".
  - Each update involves multiple access to different records in the file. If the updates are not performed atomically, the data will lose integrity.
  - Therefore, updates must be serialized (that is, done one at a time) to prevent corruption of the data.
- However, most of the serialization that occurs in the z/OS environments is not done on ‘big’ resources like files or records
  - Most serialization is done at much lower levels on areas of data, queues, bits and bytes, etc.

Locking Categories and Types

- On z/OS, there are a huge number and assortment of locks
  - Some are z/OS operating system locks
  - Some locks are created and used by vendor software product
  - Some locks are created and used by applications

- But at a high-level *(and described in more detail on the next slides)*
  - There are two types of locks that differ in degree of lock enforcement
    - Spin
    - Suspend
  - There are two different categories of locks that influence scope
    - Global
    - Local

Spin Locks

- Spin locks are used to prevent a requesting function on one processor from doing any work until the held lock is freed on another processor.
  - A spin lock causes disablement (and, thus, 'spin')
  - Spin locks are mechanisms typically used to serialize data areas or queues which require multiple updates to transition from one coherent state to the next

- There are two kinds of spin locks
  - Exclusive Spin lock – When a resource is needed exclusively
    - While lock is held exclusively, all other requests are blocked and spin
  - Shared Exclusive Spin Lock – Some requesters can get lock shared
    - Thus, resource is shared on multiple CPUs since only being read.
    - But if an update is needed, then an exclusive is needed.
      - When an exclusive request is made, requester spins until the shared folks are done.
    - Usage is so a process can read structure while it is stable, but still able to get as exclusive if needed

Spin Locks from a Performance POV

- Spin locks from a performance point-of-view
  - Primary interest in spin locks is how much CPU time is spent spinning while waiting
    - Not an issue of no one else is interested since no contention (thus no spinning)
  - CPU overhead only there when someone is waiting.
    - Exclusive guy needs to wait to all the shared guys to get done with what they want.

- Example: if lock is shared exclusive but say once a day something happens that needs to update the structure.
  - All day long things are getting spin lock shared, but no CPU time
    - Thus, of little interest
    - Number of lock requests of little interest
  - But say once a day then someone wants the resource exclusive
    - Then exclusive requester spins (using CPU), until shared guys give up the lock
  - And it works the other way around
    - Exclusive guy can have the lock, and now shared guys will spin waiting for exclusive guy to give up.

Suspend Locks

- Suspend locks are used to prevent the requesting program from doing work until the lock is available, but still allow the processor to continue doing other work.
  - The requestor is 'suspended' while other work may be dispatched on that processor
  - Upon release of the lock, the suspended requestor is given control with the lock or is re-dispatched to retry the lock obtain

- Example: Someone is running along as a TSO user, but then requires a suspend lock (like a local lock) in its own address space.
  - Depending on the type of suspend lock involved, the thing that is suspended is the specific task (TCB) or SRB involved in the request.
  - The waiter is suspended if contention exists

Suspend Locks from a Performance POV

- Suspend locks from a performance point-of-view have 2 primary issues
  - As a reminder, the user / CPU is not spinning waiting
  - The act of suspending and resume are comparatively expensive (to spin locking)
    - But not a big deal if there is no contention
- Primary issue 1: Possible disruption of the processor caches
  - If contention exists, while suspended the requester may have its processor caches disrupted
  - Thus, a waste of cycles to resolve the cache misses until the requester is re-dispatched
- Primary issue 2: Possible response time issues.
  - If the unit of work that is suspended has the possibility of impacting response times
  - Afterall, it is 'suspended'
  - Maybe little impact, may be great impact…

Summary of Locking Categories

- There are two categories of locks:
  - Global locks : used to protect serially reusable resources related to more than one address space.
  - Local locks : used to protect the resources assigned to a particular address space.
    - When the local lock is held for an address space, the owner of the lock has the right to manipulate the queues and control blocks associated with that address space.

Summary of types of SMF measured locks

- There are lot of system locks, user locks, vendor software locks in the z/OS operating system
  - Locking mechanisms is determined by the code developer
    - The type of lock chosen determines what happens when a function on one processor in an MP system makes an unconditional request for a lock that is held by another unit of work on another processor
    - From an SMF measurement point-of-view, the following table is a summary of the locking types

lock	global	local	spin	suspend	single	multiple (class)
CPU	X		X			X
CMS	X			X	X	
CML		X		X		X
LOCAL		X		X		X

Locking hierarchy

- The locks are arranged in a hierarchy to prevent a deadlock between functions on the processor(s)

- The locks provided by the system in hierarchical order are:
  (described on following slides)
  - CPU (processor lock)
  - CMS (general cross memory services lock)
  - CML (cross memory local lock)
  - Local storage lock (LOCAL)

CPU Locks

- CPU (processor lock)
  - Used to serialize on the processor level. Basically provides system-recognized (valid) disablement
- CPU locking is a 'formality'
  - The 'formality' is that the CPU lock is a mechanism for code that is running disabled to become legally disabled
  - When RTM, or whatever, detects code that is disabled, but not legally disabled, then an ABEND occurs
- Why would a developer want to use a CPU lock?
  - It is the operating system programmers, or a vendor or customer that writes code that will be part of the operating system (for example… someone wants to write a user SVC and needs to be disabled.
- Example:
  - Supposed developer writes code for a CPU save area, but the developer want no other incarnation on this code on *this* CPU while the program is using that save area
  - Needs to run disabled in order to serialize the save area, but also tell RTM that it is legally disabled
  - So uses the CPU lock (which could be considered a pseudo spin lock)

CMS Locking Overview

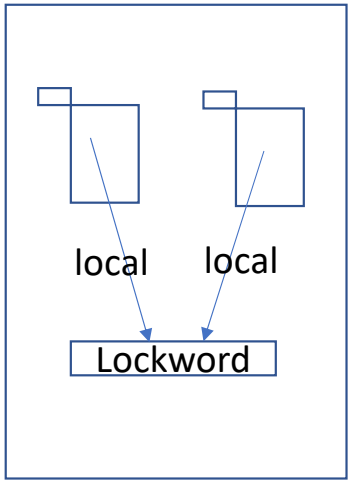
- CMS (general cross memory services lock)
  - Used to serialize on more than one address space where this serialization is not provided by one or more of the other global locks.
    - The CMS lock provides global suspend serialization when enablement is required
    - CMS locks are owned by the system, so the lock instrumentation data is kept at the system level.
  - Examples of CMS locks in SMF records include:
    - CMS lock
    - CMS Enqueue/Dequeue lock
    - CMS Latch lock
    - CMS SMF lock
- Example : Doing a getmain for pageable storage in CSA

Local Locking Overview

- Local storage lock (LOCAL)
  - Used to serialize functions and storage used by the local supervisor within an address space.
  - There is one ‘lockword’ for each address space.
  - Note: probably too much detail, but…
    - You must hold a local lock, either CML or LOCAL, when requesting the CMS lock.
    - You cannot release the local lock while holding the cross-memory services lock.

- Local storage lock - serializes functions and storage within a local address space.
  - One LOCAL lock exists per address space.

- Each address space's local lock can be requested from another address space as a CML lock
  - See next slide on CML lock for more detail



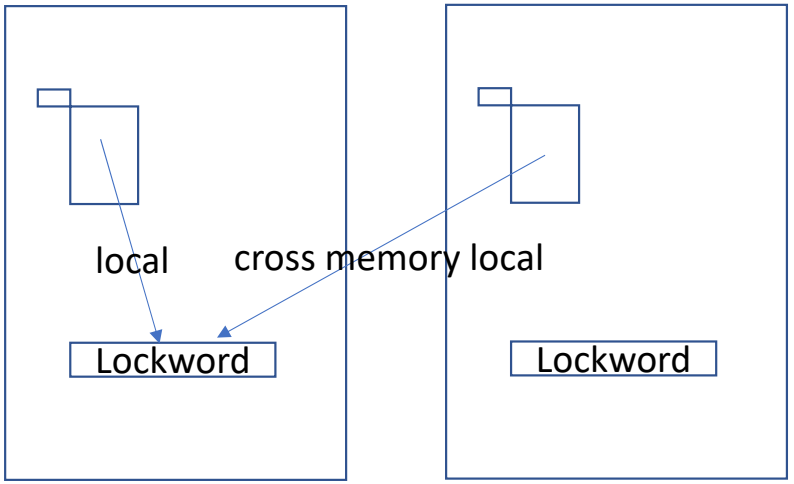
Local Lock Obtain

CML Locking Overview

- CML (cross memory local lock)
  - Used to serialize resources in an address space other than the home address space
  - CML locks have the same attributes as the LOCAL lock.
    - The owner of a CML lock can be suspended for the same reasons as the owner of the LOCAL lock, such as CMS lock suspension or page fault suspension.

- CML and Local lock are the same lockword
  - If an address space was holding its local lock, then someone else wanting a CML lock will get suspended since both lockwords are the same

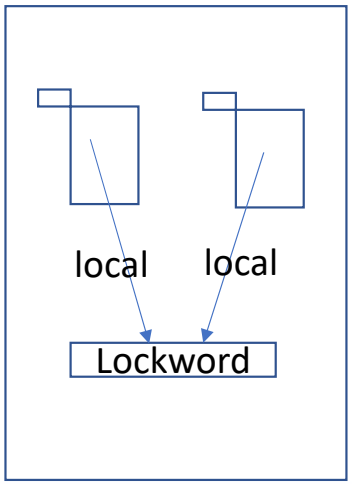
- Example: A cross memory getmain



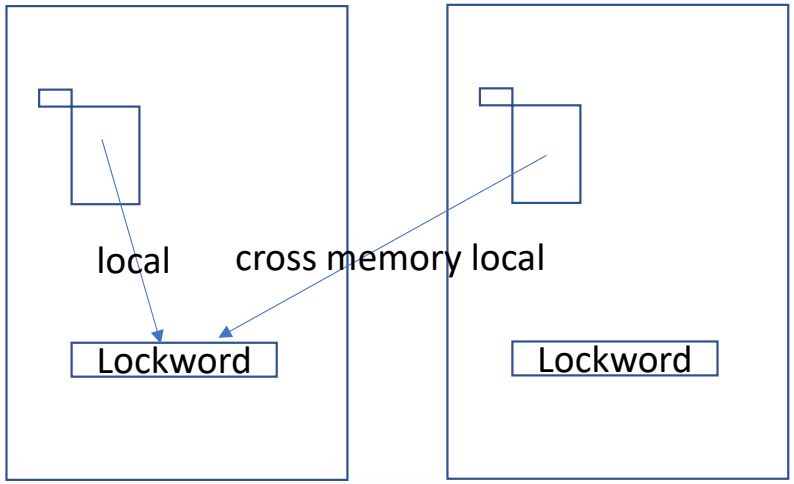
CML Lock Obtain

Summary Local Lock vs (CML) Cross Memory Local Lock

- Local lock requests and CML lock requests use the same lockword
  - Local lock is when and address space has code that wants to serialize resources within itself
  - CML lock when something else outside the address space wants to serialize resources in the local address space



Local Lock Obtain



CML Lock Obtain

Questions?