CheatSheet #43 zTidBits Resource Serialization

* In a multitasking, multiprocessing environment, resource serialization is the technique used to coordinate access to resources that are used by more than one application.
* Programs that change data need exclusive access to the data.
  - If several programs were to update the same data at the same time, the data could be corrupted (also referred to as a loss of data integrity).
  - On the other hand, programs that need only to read data can safely share access to the same data at the same time.
* The most common techniques for serializing the use of resources are enqueuing and locking.
  - These techniques allow for orderly access to system resources needed by more than one user in a multiprogramming or multiprocessing environment.
  - In z/OS, enqueuing is managed by the **global resource serialization** (GRS) component and locking is managed by various lock manager programs in the supervisor component.

Global Resource Serialization: The global resource serialization component processes requests for resources from programs running on z/OS. Its Global Resource Serialization serializes access to resources to protect their integrity.

* An installation can connect two or more z/OS systems with channel-to-channel (CTC) adapters to form a GRS complex to serialize access to resources shared among the systems.
  - When a program requests access to a reusable resource, the access can be requested as exclusive or shared.
  - When global resource serialization grants shared access to a resource as exclusive other users cannot obtain access to the resource.
  - When global resource serialization grants exclusive access to a resource, all other requestors for the resource *wait* until the exclusive requestor frees the resource.

Enqueuing: Enqueuing is the means by which a program running on z/OS requests control of a serially reusable resource.

* Enqueuing is accomplished by means of the ENQ (enqueue) and DEQ (dequeue) macros, which are available to all programs running on the system. For devices that are shared between multiple z/OS systems, enqueuing is accomplished through the RESERVE and DEQ macros.
* On ENQ and RESERVE, a program specifies the names of one or more resources and requests shared or exclusive control of those resources. If the resources are to be modified, the program must request exclusive control**;** if the resources are not to be modified, the program **should** request shared control, which allows the resource to be shared by other programs that do not require exclusive control.
* If the resource is not available, the system suspends the requesting program until the resource becomes available. When the program no longer requires control of a resource, the program uses the DEQ macro to release it.

Locking: Through locking, the system serializes the use of system resources by authorized routines and, in a Parallel Sysplex, by processors.

* A lock is simply a named field in storage that indicates whether a resource is being used and who is using it.
* In z/OS, there are *two* kinds of locks:
  - **Global locks** are resources related to more than one address space.
    Note: Global locks are provided for nonreusable or nonsharable routines and various resources.
  - **Local locks** are resources assigned to a particular address space.
* To use a resource protected by a lock, a routine must first request the lock for that resource.
* If the lock is unavailable (it is already held by another program or processor), the action taken by the program or processor that requested the lock depends whether the it's a **spin lock** or **suspend lock**:
  > A **spin lock** is when the thread simply waits in a loop ("spins") repeatedly checking until the lock becomes available or processor releases it. As soon as the lock is released, the requesting processor can obtain the lock and, thus, control of the protected resource. Most *global locks* are spin locks. The holder of a spin lock should be disabled for most interrupts (if the holder were to be interrupted, it might never be able to gain control to give up the lock).
  > A **suspend lock** is when the processor switches to a different thread while waiting for the lock, placing it on a queue until the lock is available. Other work is dispatched on the requesting processor. (this is used by *local locks)*

IMPORTANT You might wonder what would happen if two users each request a lock that is held by the other? Would they both wait forever for the other to release the lock first, in a kind of stalemate? In z/OS, such an occurrence would be known as a **deadlock**. Fortunately, the z/OS locking methodology prevents deadlocks. To avoid deadlocks, locks are arranged in a hierarchy, and a processor or routine can unconditionally request only locks higher in the hierarchy than locks it currently holds.

For example, a deadlock could occur if processor 1 held lock A and required lock B; and processor 2 held lock B and required lock A. This situation cannot occur because locks must be acquired in hierarchical sequence.

Assume, in this example, that lock A precedes lock B is the hierarchy. Processor 2, then, cannot unconditionally request lock A while holding lock B. It must, instead, release lock B, request lock A, and then request lock B. Because of this hierarchy, a deadlock can never occur.

ADVANTAGES: In a global resource serialization complex, programs can serialize access to data sets on shared DASD volumes at the data set level.

* A program on one system can access one data set on a shared volume while other programs on any system can access other data sets on the same volume.
* Because global resource serialization enables jobs to serialize their use of resources at the data set level, it can reduce contention for these resources and minimize the chance of an interlock occurring between systems.
  - This ends the need to protect resources by job scheduling.
  - Because Global Resource Serialization maintains information about global resources in system storage, it does away with the data integrity exposure that occurs when there is a system reset while a reserve exists. A global resource serialization complex also allows serialization of shared logical resources — resources that might not be directly associated with a data set or DASD volumes.
  - An ENQ with a scope of SYSTEMS might be used to synchronize processing in multisystem the systems that access shared resources into a global resource serialization complex can solve the problems related to using the System's RESERVE macro.
* *The RESERVE macro* serializes access to a resource (a data set on a shared DASD volume) by obtaining control of the volume on which the resource resides to prevent jobs on other systems from using any data set on the entire volume (VTOC !).

The volume is "reserved" to ensure the VTOC reflects current status of the file or dataset attributes

* There is an ENQ associated with the volume RESERVE; this is required because a volume is reserved by the z/OS image and not the requester's unit of work.
  - As such, the ENQ is used to serialize across the requesters from the same system.
* ***IBM recommends*** basic or parallel sysplex technology for resource sharing.
  - Global resource serialization uses parallel sysplex technology to achieve significant scalability and performance benefits in its star mode.
* Even with the basic sysplex technology, global resource serialization provides the capability to dynamically change the resource name list (RNL) and provide superior recoverability to ring disruptions.
* Before looking at the **ring** and **star** complexes in more detail, you need to know what they are and how they are generally implemented.
* The ring consists of one or more systems connected to each other by special links.
  - The links are used to pass information about requests for global resources from one system to another in the complex.
  - Requests are made by passing a message or token, called the ring system authority message (RSA-message), between systems in a round-robin or ring fashion.
    NOTE: There is one RSA token in a global resource serialization ring complex.
  - When a system receives the RSA, it inserts global ENQ and DEQ information, and passes it along to the next system to copy.
  - It also makes a copy of the global ENQ/DEQ info that was inserted by other systems.
  - When the RSA returns to the originating system, it knows that all other systems have seen its request, so the request is removed.
  - The system can now process those requests by adding them to the global resource queues, and can now determine which jobs own resources, and which jobs must wait for resources owned by other jobs.
  - Each system takes this information and updates its own global queues.
  - Systems participating in a ring are 'dependent' on each other to perform global resource serialization processing (**See figure 1**).
* As mentioned earlier, the systems in the complex can use global resource serialization to serialize access to global resources, such as data sets on shared DASD volumes.
  - Global resource serialization in a ring complex uses the links (either XCF paths or the dedicated communication links) to communicate information about global resources from one system in the complex to another (**See figure 2**).
    NOTE: In this document, "link", "path", and "CTC link" can mean a parallel CTC adapter, an ESCON channel operating in basic mode, or a list structure in a coupling facility (CF).
* The **star** method of serializing global resources is built around a coupling facility (CF) in which the global resource serialization lock structure, ISGLOCK, resides.
  - In a star complex, when an ISGENQ, ENQ, DEQ, or RESERVE macro is issued for a global resource, zOS uses information in the ISGLOCK structure to coordinate resource allocation across all systems in the sysplex (**See figure 3**).

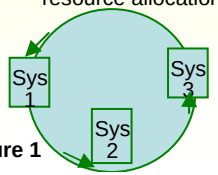

figure 1

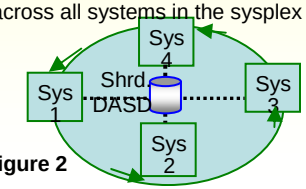

figure 2

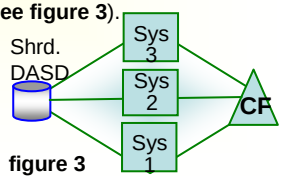

figure 3

Sample Operational Interface to GRS

```
Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------
DNET855 G  CONSOLE DNET855                        LINE 0       COLUMNS 42- 121
COMMAND INPUT ===> _                                          SCROLL ===> PAGE
******************************** TOP OF DATA ********************************
   ISF031I CONSOLE DNET855 ACTIVATED
  -D GRS
   ISG343I 10.27.21 GRS STATUS 555
   SYSTEM    STATE              SYSTEM    STATE
   DEMOMVS   CONNECTED          DEMOMVS2  CONNECTED
   DEMOMVS3  CONNECTED
   GRS STAR MODE INFORMATION
     LOCK STRUCTURE (ISGLOCK) CONTAINS 524288 LOCKS.
     THE CONTENTION NOTIFYING SYSTEM IS DEMOMVS
     SYNCHRES:      YES
     ENQMAXU:     16384
     ENQMAXA:    250000
     GRSQ:   CONTENTION
****************************** BOTTOM OF DATA ******************************
```

A DISPLAY GRS command. This displays information about the systems in the same global resource serialization complex as this system.
NOTE - STAR Configuration

```
DNET855 ERB2BUF    RMF - SENQ System Enqueue Contention          Line 1 of 5

                        CPU=  8/  8 UIC= 65K PR=   0        System= MVSA Total

10:49:41 TSK  TSK  TSK MAJOR NAME
         OWN  WTE  WTS   MINOR NAME

                        KLVGLOCK
     1    1    0   XDEMOPL                                        (SYSS)*
                        SYSZDRK
     1    1    0   TWC8EQQTWSIE(SYS)
     1    1    0   TWC8EQQTWSOE(SYS)

Command ===> _                                              Scroll ===> PAGE
```

RMF II (SNAPSHOT)
SYSTEM ENQ CONTENTION

```
DNET855 ERB2BUF     RMF - SENQR System Enqueue Reserve            Line 1 of 1

                        CPU= 20/ 20 UIC= 65K PR=   0        System= MVSA Total

10:50:31        SYSTEM ENQUEUE RESERVE REPORT
JOBNAME ASID SYSTEM  REQ VOLUME DEV  RSV MAJOR    MINOR

JES2      34 DEMOMVS  EO DMPSPO 0657 CNV SYSZJES2 DMPSPOSYS1.DEMOMVS.CKPT1    *

Command ===> _                                              Scroll ===> PAGE
```

RMF II (SNAPSHOT)
SYSTEM ENQ RESERVE "activity"
(i.e. JES2 Checkpoint with a volume reserve)

```
DNET855 ERB3BUF       RMF V1R9   ENQ Resource Delays              Line 1 of 4

Samples: 100     System: MVSA  Date: 01/18/09  Time: 10.43.20  Range: 100   Sec

----------- Resource Name -----------    ---- Delayed ----    ---- Holding ----
Major/Minor     (Scope)                   %  Name     STAT     %  Name/SYS STAT

SYSZDRK         (SYS)                    27 TWS8E     EW      27 TWC8      EO
 TWC8EQQTWSOE
SYSZDRK         (SYS)                    33 TWC8      EW      33 TWS8E     EO
 TWC8EQQTWSIE

Command ===> _                                              Scroll ===> CSR
```

RMF III (Performance Analysis)
ENQ RESOURCE DELAYS

Slide 2 of 2

See: Performance Analysis of Spin/Suspend Locks in Operating Systems @ www.research.ibm.com/journal/rd/262/ibmrd2602O.pdf