Latch onto better contention resolution

CHRIS BROOKER, NICK MATSAKIS, STEVEN PARTLOW

To aid in contention resolution, in z/OS V1R11 Global Resource Serialization (GRS) introduces the latch analyze command and latch identity service.

When the lights are flashing and the phone is ringing off the hook, the last thing you need is to wade through contention reports to figure out what's going on. Wouldn't it be better if the system identified the critical information and distinguished it from the noise? To move toward smarter contention reports, in z/OS V1R11 GRS introduces the latch analyze command and latch identity service.

Background

Since its introduction, the DISPLAY GRS,CONTENTION (D GRS,C) command has become a prominent asset for diagnosing performance and hang problems. The command was adopted because it provides a quick view of both ENQ and latch contention in the GRS complex. For decades, it was the only way for someone to obtain the crucial contention information necessary for debugging system problems. Although D GRS,C provided the right information, its presentation is lacking. The command does not support sorting contention records in a meaningful way or filtering output to eliminate noise. These limitations can hinder the ability to resolve problems.

In OS/390 V2R9, GRS introduced enhanced contention analysis (ECA) using the DISPLAY GRS,ANALYZE (D GRS,AN) command, and then later the ISGECA API to address the customer requirement for a more intelligent breakdown of ENQ contention. With the D GRS,AN command interface you can drill down using system, job, and ASID filter keywords. The command also clearly displays long blocker, long waiter, and dependency relationships including deadly embrace deadlock detection, enabling you to react to a problem quickly and more intelligently.

Put a LID on It

In z/OS V1R11, GRS extends the D GRS,AN command to support a new LATCH keyword for use in analyzing latch contention. The D GRS,AN,LATCH command makes use of the new latch identity (LID for short) information, which aids in the debugging process. The LID API, ISGLID, provides human readable information about the purpose of the latch and additional actions you might take to aid in resolving the contention.

LID also provides attributes about the latch's usage to help guide the analytics. Resource Recovery Services (RRS) and system logger are now providing the LID for their latches. RRS indicates the unit of recovery (UR) for which the latch is being used so you can use RRS commands to get more information about the UR and possibly stop it or back it out. Likewise, system logger provides the log stream name allowing appropriate action against the log stream in order to resolve the contention. The D GRS,AN,LATCH command displays the LID string along with the latch number. We expect more z/OS components to take advantage of the LID function in future releases.

An edge with analyze

The D GRS,AN command is not a replacement for D GRS,C but rather an extension. D GRS,C displays contention on a resource-by-resource basis with a list of holders and waiters for each resource that is in contention. This works well for simple contention cases where you can quickly see deadlock and who is the longest blocker causing all the trouble. Figure 1 shows the easy to read table format:

```
S1  ISG343I 18.37.54 GRS STATUS 538
LATCH SET NAME:  ABC Inc Latchset
CREATOR JOBNAME: SVRASID1  CREATOR ASID: 0029
  LATCH NUMBER:  1
    REQUESTOR  ASID  EXC/SHR    OWN/WAIT  WORKUNIT  TCB  ELAPSED TIME
    SVRASID1   0029  EXCLUSIVE  OWN       004E6B70   Y   00:00:18.695
    SVRASID3   0025  SHARED     WAIT      004E6650   Y   00:00:10.675
LATCH SET NAME:  Latchset C
CREATOR JOBNAME: SVRASID2  CREATOR ASID: 002A
  LATCH NUMBER:  123
    REQUESTOR  ASID  EXC/SHR    OWN/WAIT  WORKUNIT  TCB  ELAPSED TIME
    SVRASID2   002A  EXCLUSIVE  OWN       004E68E0   Y   00:00:20.699
    SVRASID1   0029  SHARED     WAIT      004E6B70   Y   00:00:14.685
LATCH SET NAME:  XYZ Software Lset
CREATOR JOBNAME: SVRASID3  CREATOR ASID: 0025
  LATCH NUMBER:  48
    REQUESTOR  ASID  EXC/SHR    OWN/WAIT  WORKUNIT  TCB  ELAPSED TIME
    SVRASID3   0025  EXCLUSIVE  OWN       004E6650   Y   00:00:16.690
    SVRASID2   002A  EXCLUSIVE  WAIT      004E68E0   Y   00:00:12.680
```

Figure 1. DISPLAY GRS,CONTENTION example output

D GRS,C displays the owners and waiters for each resource with no sorting or analysis of the relationships between the resources. The table format is easy to read, yet lacks the details about latch use and what other commands related to the resource owner that might help determine how to get out of the mess.

Although D GRS,C works well for simple cases in which the most obvious solution is to cancel a job that is holding onto the resource, it can be just the tip of the iceberg when things get complicated. In the example shown in Figure 1, latch number 48 is in contention longer than any of the other latches are, yet latch 48 does not sort to the top. In order to figure out the real problem, some pencil work is probably necessary. Figure 2 shows a possible outcome of such an exercise.

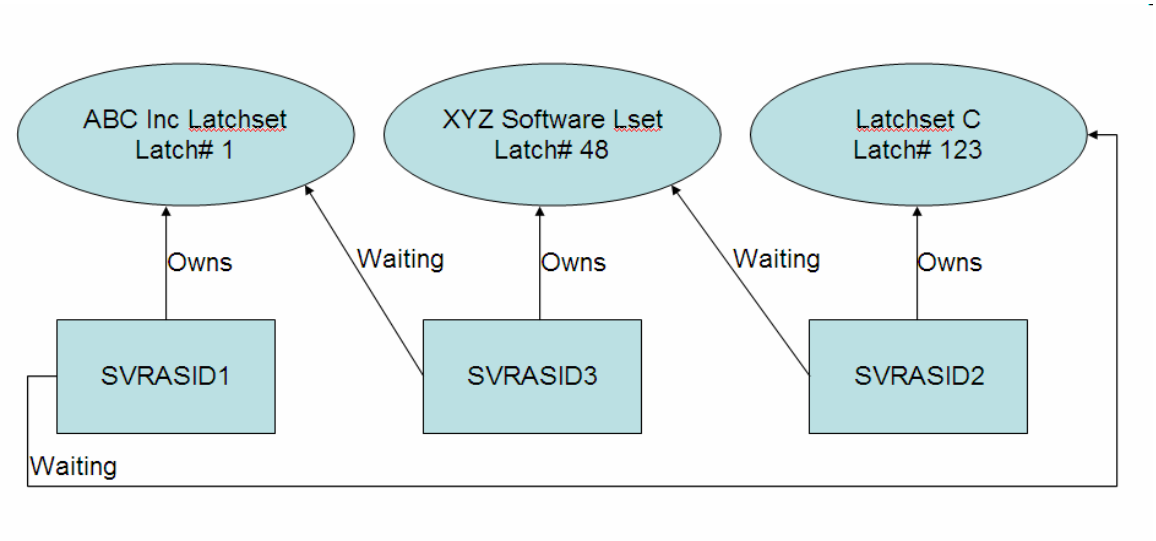

Figure 2 Figuring out the resource relationship with pen and paper

After spending a few minutes to figure out that you have detected a deadlock, you might think to yourself, “If I can do this, why can’t the system figure it out for me?” Well, now it can. D GRS,AN,LATCH,DEPENDENCY provides the following details about the above situation:

```
S1  ISG374I 18.40.20 GRS ANALYSIS 567
DEPENDENCY ANALYSIS: ENTIRE SYSTEM
----- LONG WAITER #1
WAITTIME  JOBNAME  E/S  CASID LSETNAME/LATCHID
00:02:25  SVRASID1 *S*  002A  Latchset C
                              123: (ID NOT SPECIFIED)
BLOCKER   SVRASID2  E
00:02:24  SVRASID2 *E*  0025  XYZ Software Lset
                              48: XYZ Software work process latch. Se T
BLOCKER   SVRASID3  E
00:02:22  SVRASID3 *S*  0029  ABC Inc Latchset
                              1:ABC Log File (F ABC,LOG for details)
BLOCKER   SVRASID1  E
ANALYSIS ENDED: A DEADLOCK WAS DETECTED
```

Figure 3. D GRS,AN,LATCH,DEPENDENCY example output

For each waiter it finds, D GRS,AN,DEPENDENCY looks at the blocking unit of work to see if it is waiting for any latches. If the blocker is also a waiter, the process is repeated until either a deadlock is detected (as shown in Figure 3), or the root blocker is found (a blocker that is not waiting on any latches). Deadlocks sort to the top, and the rest of the dependency chains report in order of the longest waiter.

The LID strings, if present, follow the latch number. As we said earlier, these strings can contain useful information not only about the purpose of the latch but possibly details about how to find out more information. In Figure 3, notice the ‘T’

at the end of the LID string for latch number 48. This indicates that the string is truncated. In order to see the additional text of the string, up to 255 bytes, issue the command again with the DETAIL option. You can also limit the output by specifying CASID=25 and LAT=(XYZ*,48) on the command.

```
S1  ISG374I 21.10.31 GRS ANALYSIS 217
DEPENDENCY ANALYSIS: CASID=0025
LAT=(XYZ*,48)
----- LONG WAITER #1
         JOBNAME: SVRASID2  (ASID=002A, TCB=004E68E0)
         REQUEST: SHARED                          LT:7F50907800000000
WAITING  00:02:48 FOR RESOURCE (CREATOR ASID=002A)
XYZ Software Lset                                LST:7F52E10000000488
48:XYZ Software work process latch. See users guide for information.
…
```

Figure 4. LID string example for latch 48

Another look at contention

Many automation products such as RMF and OMEGAMON XE provide support for ENQ contention monitoring and reporting. Tivoli Systems Automation for z/OS support for Enabling Long Running Enqueues (ENQs) uses the ECA analytics through the ISGECA API to determine which ENQs to consider for possible action.

z/OS UNIX System Services provides its own detailed latch analysis and proactive resolution. This includes functions like:

- F BPXONIT,RECOVER=LATCHES command

mount and file system latch tracking, which allows recycling of system threads that are holding latches

- DISPLAY OMVS,W command with additional information.

Latch on now

z/OS provides many serialization mechanisms for use by programs. More specifically, many critical functions and subsystems use ENQs and GRS latches. Monitoring and speedy analysis of contention on these resources is critical for insuring high availability. The enhanced contention analysis function provided by the D GRS,AN command and the associated API provide additional analysis. Update operational procedures, automation policies, and scripts to take advantage of the new latch support. Customers who are not currently using the Tivoli ENQ support or another equivalent function should look into doing so.

Special thanks

In addition to the authors, special thanks for this function and article go to the rest of the z/OS GRS Development and Test Team: Bryan Childs, Alfred Foster, Tammy Garren, Joseph Gentile, Susan Lamberton, and Thomas Rankin.

Author biographies:

Chris Brooker is a Staff Software Engineer in z/OS global resource serialization (GRS) development and the GRS level 3 team lead. He holds an MS degree in Software Development from Marist College and has been with IBM for 8 years.

Nick Matsakis is an IBM Senior Software Engineer responsible for the design and development of the global resource serialization (GRS) component. He has been with IBM for 25 years.

Steven Partlow is an Advisory Software Engineer in the z/OS global resource serialization (GRS) Development Team. Steven holds a B.S. degree in Computer Science from the University of Illinois at Urbana-Champaign.