GRS

Its Scalability, Performance and CPU Utilization

Bryan Childs

GRS Development

bchilds@us.ibm.com

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml

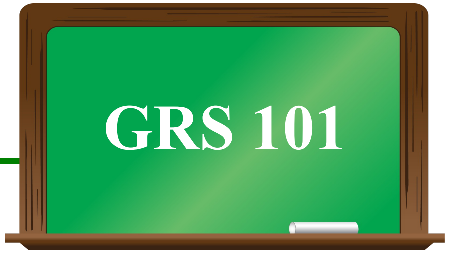
Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

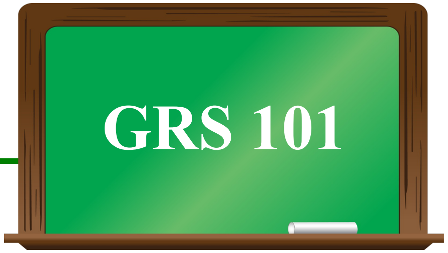
Quick Review

What does GRS do?



- Global Resource Serialization provides:
  - ENQ/DEQ services
    - Abstract, granular resources
    - Scope of step, system, or systems
  - Latch callable services
    - Only scope of system
    - Only available to authorized requesters
    - Fast!

Continued Review



What else does GRS do?

- ENQ/DEQ supporting functionality includes:
  - Corresponding Query service
  - GRSCNFxx & GRSRNLxx parmlib members
  - Display and SetGRS system commands
  - Many CSVDYNEX exits
  - ENF 51 signals for configuration & contention
  - "EQDQ" Monitor

Ring, Star, or None?

Modes for handling global resources

- None mode
  - Single-system GRS Complex
- Ring mode
  - Utilizes base sysplex communication
- Star mode
  - Utilizes parallel sysplex / coupling facility
  - Better scalability and performance

Continued Review

GRS 101

What else does GRS do?

- Latch Callable Services include:
  - Latch Create (ISGLCRT)
  - Latch Obtain (ISGLOBT)
  - Latch Release (ISGLREL)
  - Latch Purge (ISGLPRG)
  - Latch Purge by Space (ISGLPBA)
  - Latch Identity (ISGLID)
- Display and SetGRS system commands apply too

Review Complete

After glossing over a few details

The presentation now continues with a true story...

GRS Constraint Story

One day, a very large system image ran out of ENQs...

- Background:
  - DB2 supports tens of thousands of datasets
  - GRS handles multiple ENQs per dataset open
- Plot:
  - Customer had several large DB2 ramp-ups
  - GRS' virtual memory for ENQs was exhausted
- Conclusion:
  - The system waitstated!

Short-Term Fix

The service stream solution

- GRS maintains a set amount of storage for ENQs
  - This is a large pool of 31-bit private blocks
  - Amount set at IPL-time to aid performance
  - Balanced against other needs of GRS-private
- The pool of storage was increased dramatically
  - DB2 ramp-ups were alleviated
  - However 31-bit virtual has its limits!

GRS Direction

Enhancements can't always be measured in a single release

- This customer scenario significantly shaped GRS
  - Reprioritized many items over many releases
  - Renewed emphasis on scalability **(S)**
  - Coupled with emphasis on performance **(P)**
- Subsequent pages show this progression
  - Not all GRS items are listed
  - Miscellaneous but related items denoted **(*)**

GRS Timeline

A set of stories with happy endings

- z/OS R6
  - ISGENQ & ISGQUERY **(S)**
  - CMSEQDQ relief **(P)**
  - CMSLATCH relief **(P)**
- z/OS R7
  - GRS Health Checks **(*)**

GRS Timeline

The progression of stories continues

- z/OS R8
  - VSCR part 1 **(S)**
  - ENQMAX **(*)**
  - Dynamic CNS **(P)**
- z/OS R9
  - VSCR part 2 **(S)**
  - Latch performance **(P)**

GRS Timeline

Progression of stories continues

- z/OS R10
  - GRS "hot spots" (P)
  - IPCS ENQ Filtering (*)
  - CMSEQDQ relief 2 (P)
  - Monitor enhancement (P)
- z/OS R11
  - VSCR part 3 (S)
  - ISV API Extension (*)

GRS Timeline

External "end" of the progression

- z/OS R12
  - Fast GRSQ **(P)**
  - GRS Ctrace improvements **(S)**

Rather than review these items chronologically,

We will review these categorically…

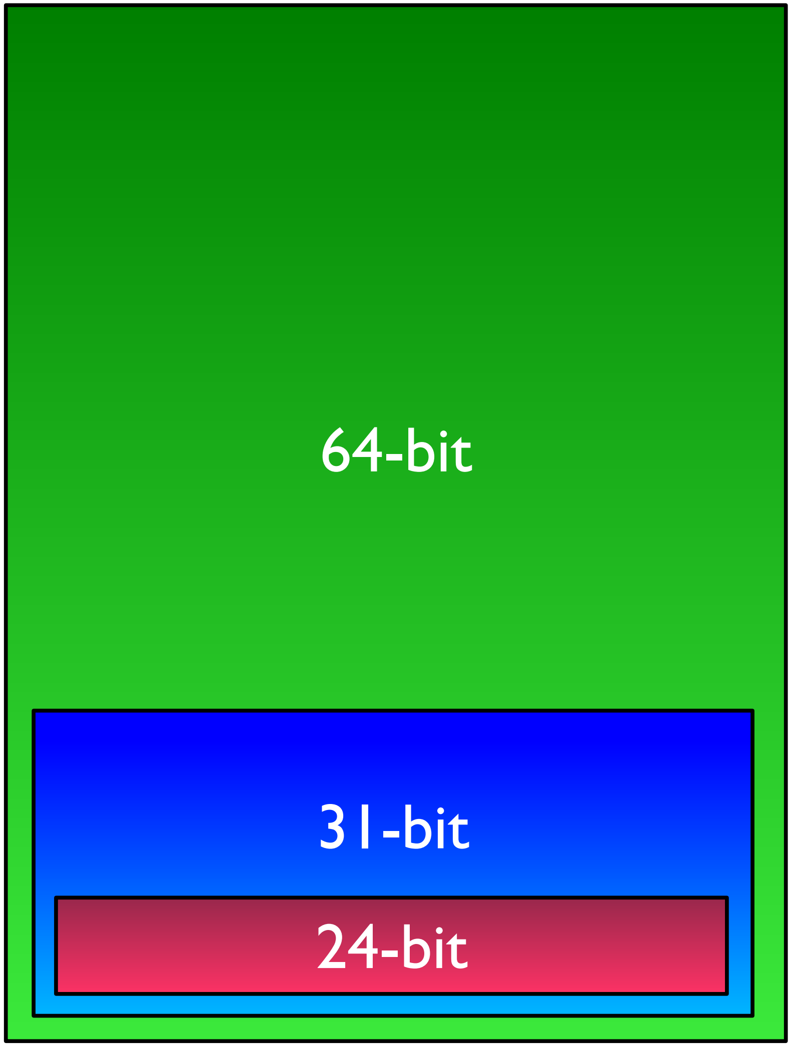
Virtual Storage Constraint

Virtually the biggest scalability issue

- zSeries provided 64-bit addressability
  - Storage Management provided services
    - GRS needed to use them
      - Several challenges to any z/OS component
      - Some challenges specific to GRS

Mapping the Space

Although it's problematic mapping 16E with 16M to scale



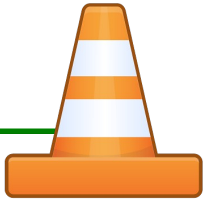
16 Exabytes ...over 8 billion times the size!

2 Gigabytes “The Bar” ...128 times the size!

16 Megabytes “The Line”

0

VSCR Challenges



It's not as easy as changing the Amode

- Control blocks contain addresses
  - 64-bit address is twice the size of 31-bit
    - Changes offset of every subsequent field
    - Lots of incompatibility and recompilation
- Linkage conventions save General Registers
  - Those registers doubled in size
  - Linkage stack can alleviate the complexity
    - Although with poor performance

GRS VSCR Challenges

Nitty gritty details

- We needed to support 64-bit parameter lists
  - In support of ENQ *users*' VSCR
  - ENQ/DEQ/GQSCAN parms “maxed out”
    - Problematic to expand compatibly
- ENQ SVC code-path separate from PC
  - Linkage=System path was considerably slower
  - Had its own recovery model as well

ISV VSCR Challenges

Internal changes can still create incompatibility

- Alternate serialization product not using “PI”
  - Accessing GRS blocks moving above the bar
  - Told IBM that they would be broken
    - New APIs would need to be created
    - Coordination required for GA

Phases of VSC Relief

Starting with services

- ISGENQ & ISGQUERY delivered in z/OS R6
  - 64-bit, expandable parameter lists
  - New function could make VSCR “no worse”
  - New GRS control blocks went above the bar
  - Coincided with z/OS architectural level-set
- 64-bit versions of Latch services delivered as well

Phases of VSC Relief

Moving Star blocks

- True VSCR began in z/OS R8
  - Star-mode specific blocks did not affect ISVs
  - Infrastructure laid out for above-the-bar work
- ENQ dual-pathing still existed, SVC vs. PC
  - Incompatible recovery models persisted too

Phases of VSC Relief

2

Moving nearly everything else

- Bulk of VSCR delivered in z/OS R9
  - Remaining ENQ-related blocks moved up
  - ENQ paths converged
  - Recovery models converged
  - New interfaces for alternate serialization
    - CSVDYNEX exits to intercept ENQs
    - ISGADMIN service to manipulate queues

ISV Segue

What exactly did they get?

- CSVDYNEX exits:
  - ISGNQXITPREBATCH
  - ISGNQXITBATCHCND
  - ISGENDOFLQCB
  - ISGNQXITQUEUED1
- ISGADMIN REQUEST=MOVEWAITER

These are official externals. But don't use them.

ISV Segue Continued

Alternate serialization support needed more work

- APIs delivered in z/OS R9 aided ISV serialization
- They didn't address breakage in ISV monitoring
  - More APIs delivered in z/OS R11 for this
    - ISGNQXITQUEUED2 for global results
    - Global resource token externalized
    - ISGCNFXITxxx got more contention data
      - These exits originally for skipping ENF 51

Phases of VSC Relief

Follow-up on Qscan

- VSCR follow-up delivered in z/OS R11
    - GQScan/ISGQUERY code used a dataspace
    - Constraint had never been seen in the field
    - Blocks moved above the bar

Service Segue

A GRS CTRACE dilemma

Q: What's more frustrating than debugging a GRS issue with insufficient CTRACE options?

Service Segue

A GRS CTRACE dilemma

Q: What's more frustrating than debugging a GRS issue with insufficient CTRACE options?

A: Debugging a GRS problem with sufficient CTRACE options, but having that information wrap off the buffer!

This too, was a VSCR item of sorts...

Phases of VSC Relief

Enhancing GRS CTRACE

- Another VSCR follow-up delivered in z/OS R12
  - GRS CTRACE blocks moved above the bar
  - New maximum buffer size: 2047M
    - Coincided with new CTRACE maximum
  - New default size: 16M (the old maximum)
- New CTRACE FLOWA also added
  - CTRACE data for latches

Understanding ENQMAX

An earlier DB2 ramp-up story

- Background:
  - GRS limits an address space to 250,000 ENQs
    - Safeguards against excessive ENQing
- Plot:
  - Customer had a large DB2 ramp-up
  - GRS still had plenty of virtual storage
- Conclusion:
  - Customer given GVT zap to avoid ABEND538

Clarifying ENQMAXA/U

Authorized & unauthorized maximums

- ENQMAX support provided in z/OS R8
  - Real externals for adjusting ENQ maximums
  - GRSCNFxx and SETGRS support
    - ENQMAXA for authorized requesters
    - ENQMAXU for unauthorized requesters
- Ironically, these externals were an afterthought
  - Cleaned up the GVT zap hack
  - So what was the “main” solution to the issue?

ISGADMIN SETENQMAX

Space-specific maximums

- Solution: space-specific maximum for DB2
  - ISGADMIN REQUEST=SETENQMAX
  - System-wide maximums maintained
    - Safeguard intact for other address spaces
- Problem: DB2's ISGADMIN usage was delayed
  - ENQMAXA became the interim solution
  - Confused as a GRS scalability problem

It's not just Scalability

There were performance challenges too

- Considerable investment into GRS performance
  - More CPU utilization can affect TCO
  - Certain performance issues affect scalability
- Several types of performance factors considered...

Performance Factor #1

Length of the path in front of you

- Pathlength
  - New functionality takes more lines of code
    - Simplified view of instruction speed
    - Simplified view of instruction sequencing
    - Reduction of pathlength still significant

“Fast Paths”

Streamlined ENQ paths

- Authorized requesters are trusted
  - Less parameter checking
  - Parmlists not copied in caller's key
- Fastpath added for ISGENQ in z/OS R9
  - Some other qualifications:
    - Singleton request
    - Local scope
    - No OwningTtoken specified

“Hot Spots”

Miscellaneous updates

- z/OS R10 delivered several performance updates
  - Latch
  - ENQ/ISGENQ
  - GQScan/ISGQUERY
- Examples
  - Expanded hash table sizes
  - Shortened Qscan pathlength in some areas
  - Stopped clearing dynamic areas

"Hot Queues"

Readily available ENQ blocks

- ENQ blocks pre-allocated in separate pools
  - Various internal control blocks used for ENQ
  - Significantly reduced pathlength to obtain
  - Applicable to any ENQ request
  - Usage expanded usage as blocks moved above the bar
    - Began in z/OS R6
    - Expanded in z/OS R8 and again in z/OS R9

Slow GRSQ

Faster mainline meant slower dumps

- GRSQ is an optional keyword for SVC Dumps
  - Provides additional GRS diagnostics
- GRSQ already a topic of dump times in the past
  - Its internally-driven GQScan took too long
    - GRSQ(CONTENTION) in GRSCNFxx
    - GQScan driven after system dispatchable
- However “hot queues” created a new problem...

Fast GRSQ

Hot queue compression compromise

- Spikes of ENQ activity drove hot queue usage
  - e.g. DB2 ramp-up
  - Transient cells were subsequently paged out
  - SVC Dumps with GRSQ hit page-in delays
- Fast GRSQ delivered in z/OS R12
  - RSM enhanced for paged-out dumping
  - GRS used improved SDump interface
  - Hot queues compressed to alleviate paging

Disabling Performance?

Hot queue disablement

- "Fast GRSQ" item also added some disablement
- GRS now listens for ENF signals from RSM
  - Hot queues disabled for Real & Aux shortage
  - Message ISG376I issued when this occurs
  - Queues re-enabled when shortage relieved
  - Originally ISG376I issued for compression too
    - Caused many PMRs against this self-tuning
    - Message removed per customer request

GRS IPCS Filtering

It's not just about taking the dump

- Scalability to GRS means many more ENQs
  - It's not just about the speed to take the dump
  - It's also about the user interface for diagnosis
- In z/OS R10 we introduced GRS IPCS Filtering
  - Applicable to both GRSTRACE & GRSDATA

Q: What's the difference between the two?

GRSTRACE & GRSDATA

Fraternal diagnostic twins

Q: What's the difference between the two?

A: Both provide information on ENQ requests, but...

- GRSTRACE is from in-storage control blocks
  - Star global data only from local requests
- GRSDATA is from the GQScan
  - Star typically has GRSQ(CONTENTION)

GRS IPCS Filtering

What does it filter?

- Filtering includes:
  - SYSNAME
  - JOBNAME
  - ASID
  - TCB
  - CONTENTION
  - QNAME
  - RNAME
  - SCOPE
  - RESERVE
  - START TIME
  - STOP TIME

All of the selected filters, if chosen, must be true

GRS IPCS Data

What interesting data was added?

- Event TODs showing ENQ history
  - Request
  - Contention
  - Ownership
- Alteration flags
  - RNLs
  - Exits
  - Alternate Serialization

Performance Factor #2

When multiple paths converge

- Bottlenecks
  - Shared resource or service slows activity
  - Limits amount of concurrent work
    - Independent of pathlength
    - Adversely affects overall throughput
    - Typically worsens with more concurrency
      - Therefore a scalability issue as well

CMSEQDQ Constraint Relief

Limiting the footprint

- CMSEQDQ is a suspend lock used for ENQs
  - Most Step ENQs skip it as of z/OS R6
  - RNL processing skips it as of z/OS R10
  - Star-mode globals usage reduced in z/OS R10
- Serialization still intact
  - Alternate, more granular mechanisms used

CMSLATCH Constraint Relief

Limiting this footprint too

- CMSLATCH is a suspend lock used for Latches
  - Alternative “fast locks” developed in z/OS R6
  - Four locks per latch set
  - CMSLATCH usage now minimal

Multi-header Cpool

CPU-specific cells

- Latches provide high-performance serialization
- CPU-specific memory provided an opportunity
  - Multi-header Cpool usage began in z/OS R9
  - Used further in z/OS R10, part of "Hot Spots"
    - Cells are chained by specific processors
    - Used for various blocks and dynamic areas
    - Implicit benefit to all latch users

Performance Factor #3

A matter of priority

- Priority problems
  - Units of work can be “starved out”
  - “Server priority” lower than “Client priority”
  - Problems can exist at various levels, e.g.
    - System weight
    - Priority of an address space and its tasks
  - Essentially a special-case bottleneck
    - Insufficient processor time

Dynamic CNS

Understanding what it is to understand the problem

Q: What is the CNS?

CNS Explained

What it is

Q: What is the CNS?

A: The Contention Notification System

- Star system for global contention signals
- Generates GQScan each time for the data
- Broadcasts the ENF 51 signal

Why is this part of a performance discussion?

ENF 51

Informative data, but a lot of data

- GRS is the largest producer of ENF signals by far
  - ENF 51 signals are input to RMF reports
- Global GQScans drive GRS CPU and signalling
- Low-weighted systems can encounter difficulty
  - CNS task will fail if it can't handle the load
  - Sympathy sickness is also possible

Dynamic CNS

Important tuning tip

- SetGRS supports dynamic CNS in z/OS R8
- This is still a common problem reported by L2
  - Installations need to choose wisely!
    - May not be “apples to apples” across CECs
    - Don't choose the lowest weighted LPAR!

EQDQ Monitor

An informative tool, albeit with overhead

- EQDQ Monitor tracks ENQ requests
  - Not limited to contention events
  - Lots of data for Scope=System and Systems
    - Does not include Scope=Step
  - Various filters provided
- Input fed from ISGNQXITBATCH exit
  - Adds overhead to ENQ mainline
  - But there was a worse problem...

ENQ & Monitor Priorities

Priority mismatch

- Priority of the GRS address space is very high
- Priority of the Monitor's space was very low
  - GRS PCs into the space for each ENQ request
  - Local lock priority is that of the primary space
    - Not the priority of the unit of work's home
    - The EQDQ Monitor slowed ENQs further
    - So installations turned the Monitor off

EQDQ Monitor Update

Problem not solved, but much improved

- EQDQ Monitor updated in z/OS R10
  - Priority of its address space much higher
    - Pathlength overhead still remains
    - Overall throughput much better
  - New NCRESERVE filter added
    - Shows unconverted Reserve requests
    - Useful for GDPS migrations
    - Useful for GRS best practices

GRS Health Checks

Formalized best practices

- GRS Health Checks began in z/OS R7
- More added in z/OS R8
  - GRS Star mode
  - Synchres(Yes) for Reserves
  - RNLs converting all Reserves
  - No RNL exclusions preventing conversion
  - GRSQ set to Contention
  - Higher-performance exit usage

GRS CPU Utilization

How your CPU is spent

- There are three main tasks in GRS
  - Global ENQ processor
    - Global ENQs, Ring and Star versions
  - Qscan task
    - Handles GQScan/ISGQUERY requests
  - Contention Notification task
    - Each system has one for local contention
    - The CNS also handles the global contention

GRS Tuning Checklist

Making the most efficient use of the CPU

- Ensure the parallel sysplex is properly tuned
  - Qscans heavily dependent on XCF Signalling
  - Minimize false contention to ISGLOCK in CF
- Consider EQDQ Monitor usage
- Convert all Reserves to global ENQs
- Ensure the CNS is on a high-weighted system
- Limit global Qscan activity where possible

GRS Troubleshooting

White paper from GRS Level 2

- GRS Level 2 recently published this
  - Further detail on GRS CPU consumption
  - Best practices and case studies

http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101725

Summary

Advances in Serialization

- Significant enhancements made to GRS in areas of:
  - Scalability
  - Performance
- With a handful of best practices...
  - Maximize efficiency of GRS' CPU utilization