HPE POINTNEXT

Using Locking Performance Data to Solve Cluster Problems

OpenVMS Boot Camp 2017, SID 302

Keith Parris, Engineer

Hewlett Packard Enterprise

Introduction

Why look at locking data during performance investigations?

- IT professionals typically investigate performance issues by focusing on 3 areas:
  - CPU
  - Memory
  - I/O
- But in OpenVMS clusters, the Distributed Lock Manager is another important element involved in performance, and while it can sometimes complicate matters, it can often provide valuable clues in identifying performance bottlenecks in other areas

Decoding OpenVMS Resource Names

Decoding OpenVMS Resource Names

- Locks are taken out on symbolic resource names
- Only convention ties symbolic resource names to physical entities
  - With privileges, you could, in theory, directly access a locked resource (physically) without coordinating via the proper locks (but you typically wouldn't want to, of course)
- Prefix indicates subsystem:
  - F11B$ for file system
  - RMS$ for Record Management Services
  - SYS$ for OpenVMS Executive
- See appendix entitled "Lock and Resource Usage by OpenVMS Components" in most of the Internals & Data Structures books by Ruth Goldenberg, et al
- Some data is in clear text (e.g. Volume Names) while other crucial data is encoded in binary (e.g. File IDs)
- Sometimes locks are in a tree, and you may have to look at a parent look to obtain the full context:
  - RMS$ file lock (with File ID and Volume Name) → bucket lock, or RMS$ file lock → record lock
  - F11B$v Volume Allocation Lock (with Volume Name) → F11B$s File Serialization Lock (with File ID abbreviated as Lock Basis, just dropping the Sequence Number)

Lock Activity Counters

Lock Activity Counters

- OpenVMS keeps lock activity counters in Resource Blocks (RSBs) to control dynamic, activity-based lock tree remastering
  - All RSBs have the fields, but OpenVMS only updates counters in the Root RSB at the root of each tree
- OpenVMS scans the Root Resource List every 8 seconds
- RSB$W_NACT field counts lock activity
- RSB$W_OACT field is updated every 8 seconds to form a sort of running average of activity rates
  - So value is in units of operations per 8-second remaster scan interval

Lock Activity Counters

Identifying the Most-Active Lock Trees, and Tuning to Reduce Lock Rates

- You can see the RSB$W_OACT lock activity values in the “Act” column in SDA> LCK SHOW ACTIVE output:

```
SDA> LCK SHOW ACTIVE

Active Resource Tree Information (Node XYZ)
-------------------------------------------
                        Total     Local
RSB Address             Locks     Locks   SubRSB     Act   Node     Resource Name
-----------------     -------   -------   ------   -----   ------   -------------------------------
FFFFFFFF.77F66BC0          18        18        7   65528   XYZ      RMS$.......XYZ_DATA1    ...
                                                             DISK$XYZ_DATA1:[APPL1]CUSTOMERS.DAT;1
FFFFFFFF.7BA5B840        1646      1646     1645   45678   XYZ      F11B$vXYZ_DATA1
```

- Lock trees are listed in descending order by locking activity rate
- SDA provides only per-node data, not cluster-wide, so in a cluster you have to look at this on each node
- Popular tuning strategy is to look at the top few lock trees listed, and try to reduce lock rates, by:
  - Adding RMS Global Buffers to RMS files
    - A significant RMS locking improvement that first shipped in 7.2-1H1 drastically reduces the number of lock requests for data held in RMS Global Buffers compared with data held in RMS Local Buffers
      - For files shared among processes, this can also reduce I/Os and reduce memory requirements
      - GLOBAL_BUFFER_USAGE.COM tool may help identify if more RMS global buffers could help – see http://encompasserve.org/~parris/
  - Volume Allocation Locks: Reducing disk fragmentation, file open rates in applications, etc.

Lock Activity Counters

Identifying Most-Active Lock Trees, Cluster-Wide

- LOCK_ACTV_DC.COM (Data Collector) and LOCK_ACTV_RG.COM (Report Generator) available from http://encompasserve.org/~parris/ provide a cluster-wide picture of this data
- LOCK_ACTV_REPORTS.COM runs LOCK_ACTV_RG.COM for each data file collected by LOCK_ACTV_DC.COM
- COLLECTLA.COM runs LOCK_ACTV_DC.COM every half-hour
- LACSV.COM summarizes LOCK_ACTV_* reports in a .CSV file suitable for importing into Excel or Libre Office
- Rates are adjusted to per-second rates, not per 8 seconds like SDA> LCK SHOW ACTIVE
  - Thus very-active lock trees “peg” at 8191 instead of 65528

Lock Activity Counters

Identifying Most-Active Lock Trees, Cluster-Wide

– Example:

```
0000002020202020203130415441440200006789169924534D52 RMS$...g...DATA01      ...
    RMS lock tree for file [5785,26505,0] on volume DATA01
    File specification: DISK$DATA01:[DATA00.DATA.FILES]CUSTOMERS.DAT;107
        Total:  24849
        NODE02   8191 ← Value pegged at the maximum
       *NODE04   8191 ← Value pegged at the maximum; * indicates lock master node
        NODE01   5164
        NODE03   3302

2020202020203230415441447624423131 46 F11B$vDATA02
    Files-11 Volume Allocation lock for volume DATA02
        Total:  16280
        NODE02   7991
       *NODE01   6835
        NODE04   1436
        NODE03     18
...
       * Lock Master Node for the resource
```

Lock Activity Counters

Counter Anomalies

- RSB$W_NACT and RSB$W_OACT fields are only 16 bits wide
- Rather than overflow, the value “hits the peg” (like an analog automobile speedometer) at 65528
- Implication: If you see 65528 in SDA> LCK SHOW ACTIVE output (or 8191 in LOCK_ACTV_* output), the rate is likely higher, but we can’t tell *how much* higher
  - If you make a tuning change, and the rate is still pegged at 65528, you can’t tell if you’ve made any improvement, unless perhaps you trace locks using the SDA extension LCK

- On a standalone (non-clustered) node, the lock activity counters increment but are never reset, so the loxk activity counters all eventually reach 65528; if this is a problem, consider forming a single-node cluster.

Lock Activity Counters

Counter Anomalies and their effect on Lock Tree Remastering

- OpenVMS fully intends to reset the lock new-activity counter accumulator RSB$W_NACT (and calculate a new RSB$W_OACT running average) each 8-second interval, but:
  - OpenVMS limits the lock-remastering recalculation activity each 8-second interval to about 16 milliseconds, then:
    - Keeps track of where it left off, and attempts to start over again at that point during the next 8-second interval
- This means on nodes with long root resource lists, the activity counter may not be reset as often
  - This tends to increase the average activity values
  - This may tend to keep lock trees on less-active nodes with long root resource lists, or even remaster them to there
  - This seems to happen at roughly a root resource list length size of ~55K to 65K on older Integrity machines
- To determine your root resource list length, use:

```
$ ANALYZE/SYSTEM
SDA> SET FETCH QUAD
SDA> VALIDATE QUEUE/QUAD @LCK$GQ_RRSFL !Count Root Resource List Length
Queue is complete, total of 193659 elements in the queue
```

Lock Activity Counters

Counter Anomalies and their effect on Lock Tree Remastering

- The VMS84I_SYSLOA-V0300 and VMS84A_SYSLOA-V0300 kits added more counters to the SYS$CLUSTER execlet (visible only after issuing SDA> READ /EXEC) to assist in troubleshooting related issues:
  - LCK$Q_ENTRY_COUNT, the number of times the routine ADJ_COUNTERS has been called since boot to scan the root resource list
  - LCK$Q_RESTART_RSB, the number of times we timed out scanning the root resource list and saved the RSB address to start with in the next scan interval. ← Non-zero value here, continuing to increase over time, indicates the problem
  - LCK$Q_RESTART_AT_RSB, the number of times we tried to restart the root resource scan from a saved RSB address.
  - LCK$Q_RESTART_RSB_GONE, the number of times we tried to restart the root resource scan but the RSB we started on disappeared between time intervals and we had to start the scan from the beginning of the root resource list.
  - LCK$Q_COMPLETIONS, the number of times the root resource list scan has been completely scanned.
  - LCK$Q_LAST_COMPLETION, the quadword time of the last time we completed the scan of the entire root resource list. ← If this time is a long time ago (more than 8 seconds ago), this would indicate you have the problem
  - LCK$Q_PREVIOUS_COMPLETION, the quadword time of the previous time we completed scanning the entire root resource list.
  - LCK$Q_TIME_BTWN, the quadword time of the time between complete scans of the root resource list.
  - LCK$Q_MAX_TIME_BTWN, the quadword time of the maximum time between complete scans of the root resource list.
  - LCK$Q_RSBS_WALKED, the approximate number of RSBs scanned in the last time interval (within 1000 RSBs).

Lock Activity Counters

Counter Anomalies and their effect on Lock Tree Remastering

- Remediation:
  - Faster CPUs can scan the root resource list faster
  - Lowering LOCKDIRWT on a node can reduce the length of its root resource list
  - HPE Technology Services may be able to help you examine the counters and perhaps adjust some new kernel tuning knobs available with VMS84{I|A}_SYSLOA-V0300 that can:
    - Raise the maximum amount of time allowed for root resource scanning, or
    - Enable a new feature which moves the already-scanned resources to the end of the root resource list

Lock Queues

Lock Queues
Introduction

- Potential causes of lock queues:
  - Program bug (e.g. not freeing a record lock)
  - I/O or interconnect saturation
  - “Deadman” locks taken out to detect termination of a process
- Since locks serialize access to critical resources, a queue of lock requests building up may indicate a physical resource which is becoming a bottleneck
- Examples:
  - A lock queue on a file serialization lock may indicate:
    - A directory file that has become too large and is heavily used
    - A file that has become very fragmented and has many extension file headers

Lock Queues
Measurement

- Availability Manager
  - Optional Lock Contention Data Collection capability can be turned on
  - Detects locks found waiting during two consecutive scans at 10-second data collection intervals
  - Lock Contention Events appear in Event window; double-click on Event to get details:
    - Process holding the lock
    - Processes waiting for locks (cluster-wide)
    - Can opt to “fix” problem by forcing image exit or ending the offending process
    - Availability Manager uses RMDRIVER at interrupt level, so you can troubleshoot even when you can’t log in
  - Lock Contention events are logged in Lock Contention log file found at AMDS$SYSTEM:AMDS$LOCK_LOG.LOG
- System Dump Analyzer ($ ANALYZE/SYSTEM)
  - SDA> SHOW RESOURCE /CONTENTION
  - SDA> LCK SHOW CONTENTION
    - Starts reporting lock contention events, based on deadlock timeout queue
    - SDA> LCK STOP CONTENTION ends the reporting
  - SDA provides only per-node data, not cluster-wide

Lock Queues
Measurement

– LCKQUE.COM tool from http://encompasserve.org/~parris/
  – Reports lock queues existing cluster-wide at the instant it is run
  – COLLECTLQ.COM runs LCKQUE.COM once a minute to try to catch lock queue events
  – LQCSV.COM summarizes LCKQUE_* reports in a .CSV file suitable for importing into Excel or Libre Office

– Here are a couple of examples:

1) File was CONVERTed without sufficient contiguous free space, resulting in a file with multiple extension file headers:

```
'F11B$s....' 0000000E732442313146
   Files-11 File Serialization lock for file [14,*,0] on volume THDATA
   File specification: DISK$THDATA:[TH]OT.IDX;1
 Convert queue: 0, Wait queue: 28
```

2) A directory file got very large:

```
'F11B$vLOGFILE     ' 20202020454C4946474F4C762442313146
    Files-11 Volume Allocation lock for volume LOGFILE
'F11B$s....' 00000A2E732442313146
    Files-11 File Serialization lock for file [2606,*,0] on volume LOGFILE
    File specification: DISK$LOGFILE:[000000]LOGS.DIR;1
  Convert queue: 0, Wait queue: 3891
```

Process State Clues

RWCLU state

- RWCLU would normally indicate a Cluster State Transition
- Transient appearance would instead indicate Lock Tree Remastering
- All locking on a lock tree is suspended while a tree is remastered to a different node; this can cause a pause perceptible to the user
- Jumbo Frames in the network used as a cluster interconnect can speed lock remastering, by allowing more data to be passed in fewer packets, reducing packet count and reducing interrupt-state overhead
- PE1 can be set to a non-zero value to control lock remstering:
  - Negative value: disable lock remastering entirely (also disables updating of lock activity counters, so you run blind)
  - Positive value: upper limit on lock tree size that is allowed to move away from this node
  - Case study:
    - Customer had one huge Rdb database lock tree which was moving between two nodes of equal horsepower; as soon as one node grabbed the tree, it slowed down, so the other node soon grabbed it, only to slow down in turn, such that the first node grabbed it.
      1. Used SDA> LCK SHOW ACTIVE to observe movement of large lock tree, and then to determine size of large lock tree
      2. Set PE1 to a generous value (smaller than the lock tree) on one node to prevent movement of the tree away from that node
  - Very rarely see lock mastership thrashing between nodes since version 8.3's improved lock remastering algorithm

Process State Clues

RWSCS state

- RWSCS state indicates process is waiting while an SCS operation is going on
- Transient appearance generally means the response to a remote lock request is slow
- Possible reasons:
  - Cluster interconnect might be slow. Multi-site cluster? Network problems?
  - Lock master node might be overloaded
    - Fast Path setting optimization never hurts
    - Reducing lock rates might help
    - Upgrade to faster server
    - Redistribute workload
    - Redesign application for more local locking rather than remote locking

Process State Clues

RWSCS state

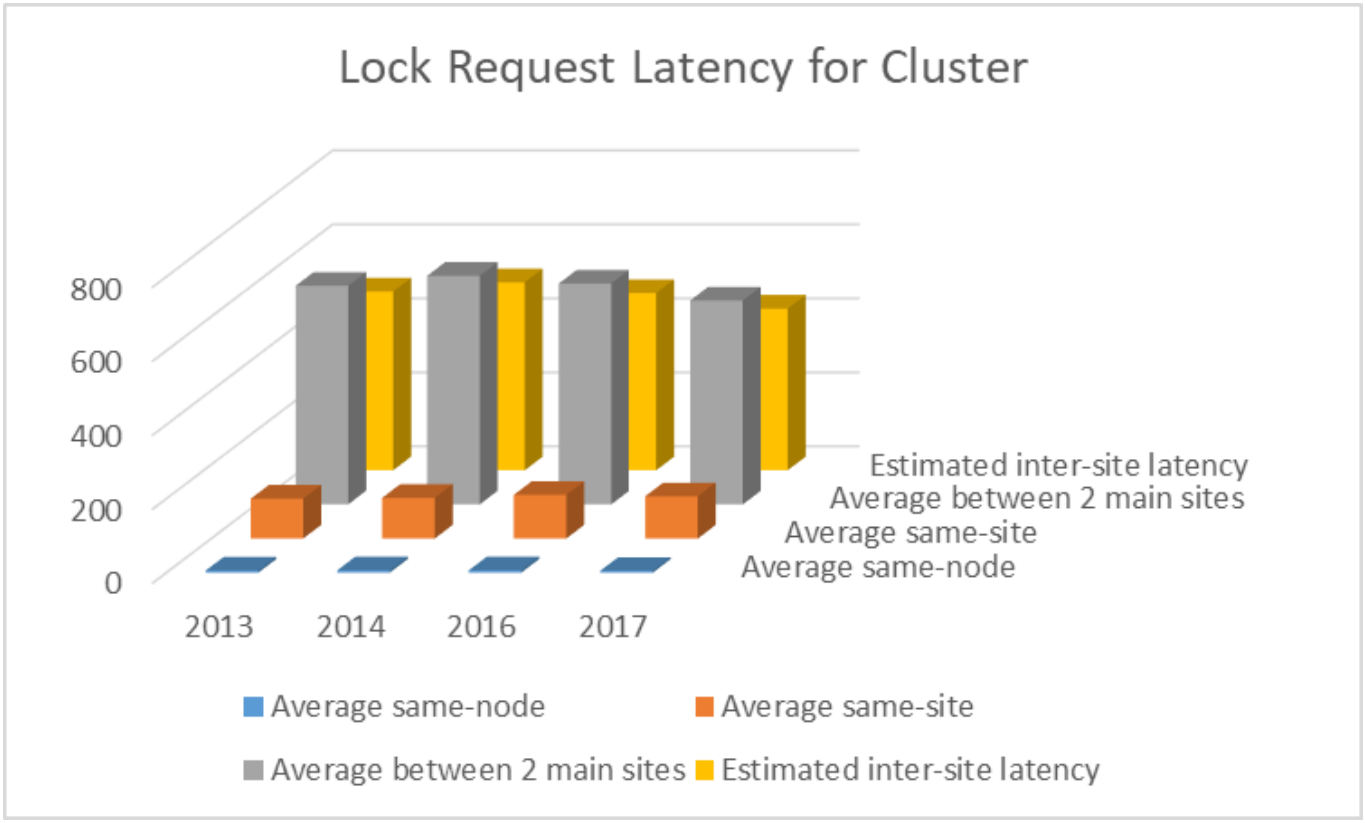
- Case study:
  - Processing of a specific task became slow in a 4-node multi-site disaster-tolerant cluster. Function was distributed by placing a server process on all 4 nodes for greater parallelism. Result: 1 process at a time would be Current; other 3 processes tended to be in RWSCS state.
- Advice:
  - $SHOW FASTPATH showed default OpenVMS assignments, so:
    - Optimize Fast Path settings
  - SDA> LCK SHOW ACTIVE showed RMS indexed files as busiest lock trees, so:
    - Reduce lock rates by adding RMS Global Buffers to busy files
  - RMS_TUNE_CHECK showed 50% of records had RRVs, and $ANALYZE/RMS/FDL showed buckets had 100% fill specified, so:
    - Convert RMS Indexed files, specifying bucket fill of 50%
  - SCACP> SHOW VC showed low PEDRIVER Current transmit window sizes compared with Maximums, and SCACP> SHOW CHANNEL /ALL showed large numbers of retransmissions and resultant duplicates received due to large numbers of delayed-but-not-lost packets, so:
    - Look at packet round-trip times and deviation and then set PE2 to an appropriate value to reduce retransmissions and thus allow window sizes to rise to maximums

Process State Clues

RWSCS state

– Case study advice:

  – LOCKTIME.COM from http://encompasserve.org/~parris/ was used to measure lock request latency. Results were compared with measurements from previous years to make sure the network hadn't suddenly gotten worse. It hadn't:



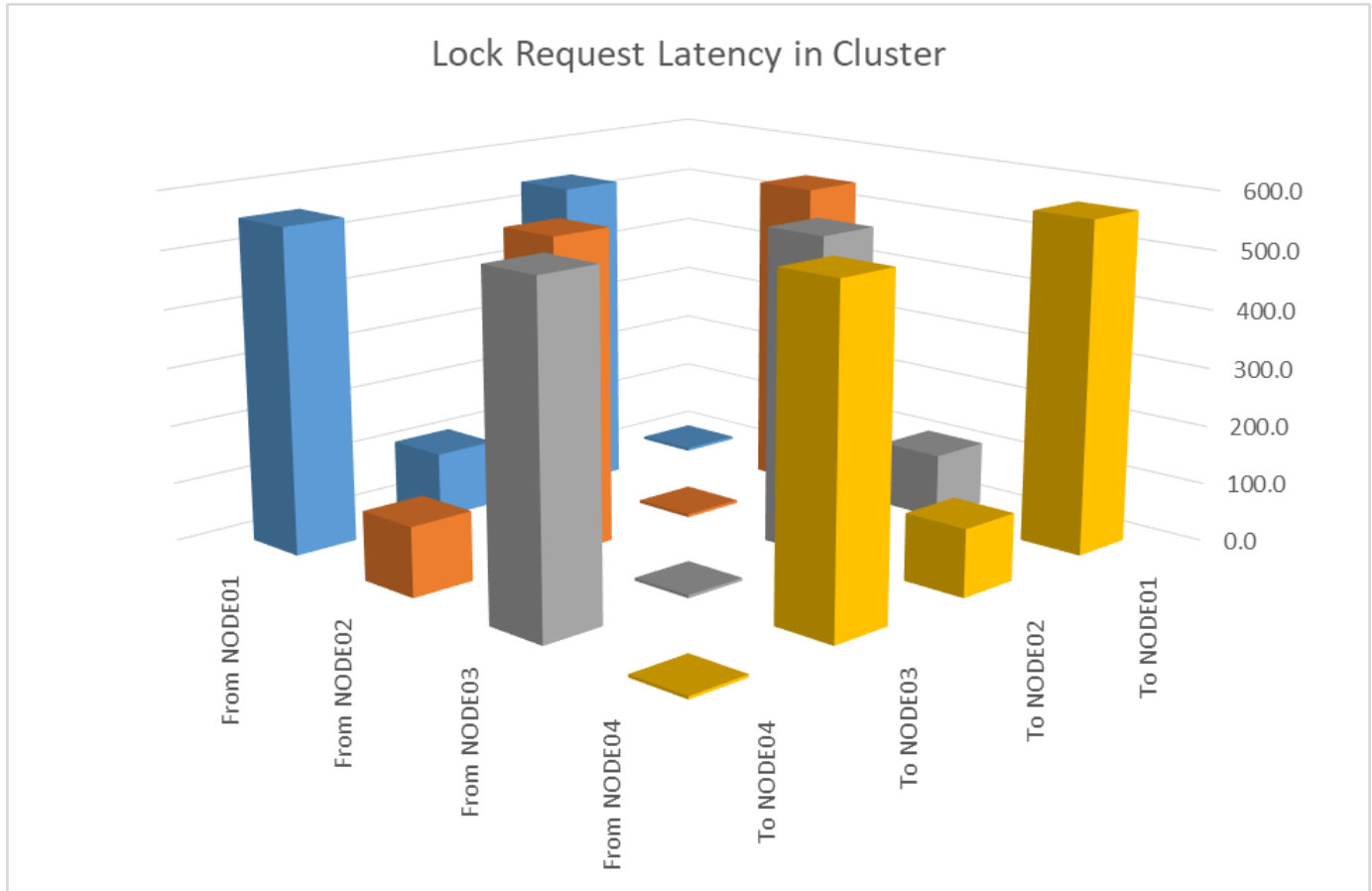
Process State Clues

RWSCS state

– Case study advice:

  – LOCKTIME results showed intra-site remote lock latency significantly better than inter-site, so:

    – For parallelism, consider placing 4 server processes on a single node instead of on 4 nodes, for all-local locking; if that is too much for one node, consider placing 2 server processes each on two nodes within the same site, so you don't suffer the greater inter-site latency.



Questions?

HPE
POINTNEXT

Thank you

keith.parris@hpe.com

Hewlett Packard
Enterprise