Distributed Lock Manager

Hartmut.Becker@compaq.com
OpenVMS Engineering
Topics

◆ Lock Manager Recent Past
  ● OpenVMS V7.1 Changes
◆ Lock Manager Present
  ● OpenVMS V7.2 Changes
◆ Lock Manager Futures
  ● Current Engineering efforts for future releases
Lock Manager Directions

- Improve Availability
- Improve Scaling
- Improve Performance
- Improve Analysis Tools
Recent Updates

- OpenVMS V7.1 (Alpha)
  - Availability
    - Remove need for LOCKIDTBL_MAX
  - Scalability
    - Move the Lock Id Table to S2 space
    - Provide for larger process ENQLM quota
    - Remove sub lock and sub resource limits
  - Performance
    - Allow for larger Resource Hash Table
LOCKIDTBL_MAX

- LOCKIDTBL_MAX controlled the maximum number of lock ids on a system
- Incorrect value would result in a need to reboot and potential system crashes
- Changes made to OpenVMS V7.1
  - Compute a very large maximum during boot
  - Additional cost is 8 bytes per 2048 lock ids
    - 8 bytes is the size of a PTE
    - Each additional page in the lock id table can map 2048 lock ids
  - Computed value can be overridden by the LOCKIDTBL SYSGEN parameter
  - The table is created in S2 space (64-bit address space)
Lock Id Table in S2 space

- LCK$GQ_IDTBL
  - LCK$GL_IDTBL (prior to OpenVMS V7.1)
  - Physical Memory initially is only initially allocated for LCKIDTBL locks
- LCK$GL_MAXID
- LCK$GL_IDTBLMAX
  - Only a single page of physical memory represents the rest of the table

Free lock id
1
2
3
ENQLM quota

- OpenVMS V7.1 Scaling Features
  - The ENQLM quota limits the number of locks a user can create
    - This value is maintained in SYSUAF.DAT by the authorize utility and has a maximum of 32767
  - Due to the amount of software and interfaces which utilized this value as a signed word - allowing values above 32767 would be difficult
  - Customer required larger values
    - Solution: Change the meaning of 32767 in the authorization file to mean an unlimited ENQLM
    - In a mixed architecture cluster, a VAX system would still use this value as 32767
  - The PQL_MENQLM and PQL_DENQLM SYSGEN Parameters have always been longword values on alpha - a value of 32767 has no special meaning
Sub Lock and Sub Resource Limits

- The Sub Lock and Sub Resource Limits have been virtually removed
  - Sub Lock Reference Count is the number of times a lock is used as a parent lock
  - Sub Resource Reference Count is the number of resource under another resource
- Prior limit for the two above was 64k (65535)
- These counters were originally not meant to be limits - their use in the lock manager is for determining interest
V7.1 Performance Work

- Allow for larger Resource Hash Tables
  - The maximum value for the RESHASHTBL SYSGEN parameter was 65535
  - This value can now exceed 4 billion
  - Each resource has an entry in the resource hash table, sparser hash tables means quicker lookups for resources
  - A rule of thumb for the size of the resource hash table is 2-4 times the average number of resources
Current Updates

◆ OpenVMS V7.2 (Alpha)
  ● Scalability
    – Move lock manager data structures from S0S1 space to S2 space
  ● Performance
    – Streamlined code flow
    – Update some PMS$ performance counters per CPU
  ● Analysis Tools
    – Enhance the ability of SDA to look at lock manager information
Lock Manager in S2 Space

- Prior to OpenVMS V7.2, Lock Manager data structures lived in Non-paged pool
- Non-paged pool is in S0S1 space
  - 32 bit addresses
  - 2 gig of address space
- On many systems, the lock manager data structures account for 50%-80% of non-paged pool usage
- We now allocate lock manager data structures from S2 space
Why we had to Move

- We’ve seen systems with 600,000 resources and 800,000 locks
- The size of a resource and a lock are both 192 bytes
- The physical non-paged pool required for these data structures is:
  \[(600,000+800,000)\times 192 = 268,800,000\text{ bytes}\]
- 256mb are 1/8th of the available address space
- Should a system run out of S0S1 address space…Things would not be pretty
Pool Zones

- Pool Zones allow for intelligent management of single size structures
- Features of Pool Zones are:
  - memory allocation/deallocation is provided by the user
  - they can dynamically expand and contract
  - there are no system parameters to dictate the maximum size
  - they do not require contiguous virtual address space
  - synchronization is provided by the caller
Lock Manager Pool Zone

- The Lock Manager Pool Zone manages 256 byte structures
  - Single pages from the free list are allocated when additional structures are needed
  - These pages are mapped in S2 space
  - Each page can hold 31 256 byte structures
  - If none of the 31 structures are in use in a page, the page can be returned to the free page list
    - We try and aggressively return memory to the system
S2 Space Implications

- Approximately 30% more physical memory is required by the lock manager
  - This is due to the larger pointers in the data structures (64bit vs. 32bit) and the header in each pool zone page
- When pages are free, the lock manager is much more aggressive in giving them back to the system for other usage
- Additional ACB structures are needed to deliver ASTs - previously, an ACB was embedded in the LKB
  - However, the ACB must be in 32 bit address space
## Lock Manager Memory Usage

### Output from SHOW MEMORY

<table>
<thead>
<tr>
<th>Dynamic Memory Usage (bytes):</th>
<th>Total</th>
<th>Free</th>
<th>In Use</th>
<th>Largest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nonpaged Dynamic Memory</td>
<td>2088960</td>
<td>468672</td>
<td>1620288</td>
<td>198912</td>
</tr>
<tr>
<td>Paged Dynamic Memory</td>
<td>1351680</td>
<td>586336</td>
<td>765344</td>
<td>582208</td>
</tr>
<tr>
<td>Lock Manager Dynamic Memory</td>
<td>991232</td>
<td>370688</td>
<td>620544</td>
<td></td>
</tr>
</tbody>
</table>

### Output from SHOW MEMORY/FULL

<table>
<thead>
<tr>
<th>Lock Manager Dynamic Memory</th>
<th>Current Size (Mbytes)</th>
<th>Current Size (pages)</th>
<th>Hits</th>
<th>Misses</th>
<th>Number of Empty Pages</th>
<th>Expansions</th>
<th>Packet Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current Size (Mbytes)</td>
<td>26.68</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>256</td>
</tr>
<tr>
<td>Free Space (Mbytes)</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Space in Use (Mbytes)</td>
<td>26.68</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Empty Pages</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Free Packets</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Stream-lined Code Flow

- For V7.2, a pass was made through the lock manager to improve and stream-line the code flow
  - This resulted in about a 20% performance gain for local locking operations
  - This gain will not be seen for distributed locking operations - only local locking operations
Lock Manager PMS$ Cells

- A number of high frequency performance counters are now kept per CPU
  - Avoids cache contention on these data cells
  - Provides additional performance data
  - Once per second, the cells are gathered into the original PMS$ performance cells
  - The per CPU cells can be viewed with the new SDA command SHOW LOCK/SUMMARY
SDA Enhancements

- The SHOW LOCK and SHOW RESOURCE commands in SDA have been greatly enhanced.
- Prior to OpenVMS V7.2, only detailed lock and resource information could be obtained from SDA.
- SDA now supports a new “brief” format along with additional qualifiers to limit the data displayed.
SDA SHOW LOCK

- New SHOW LOCK qualifiers include
  
  /GRANTED    Show granted locks
  /CONVERT    Show converting locks
  /WAITING    Show waiting locks
  /BLOCKING   Show locks blocking others
  /STATUS=    Show locks with a specific status
               (key[,key]...)
  /BRIEF      Single line display per lock
  /ADDRESS    Show a lock at a specific address
SDA SHOW LOCK

- Additional Lock Manager qualifiers
  - /SUMMARY Show Pool Zone and Performance
  - /POOL Shows detailed statistics on the Pool Zone

- Additions to SHOW PROCESS/LOCK
  - /BRIEF Show locks for a process in “brief” format
SDA SHOW RESORUCE

New SHOW RESOURCE qualifiers include:

/CONTENTION [=ALL]  Show resources with contention
/STATUS=(key[,key]…)  Show resources with a specific status
/BRIEF  Single line display per resource
/ADDRESS  Show a resource at a specific address
### SDA> SHOW LOCK/POOL

<table>
<thead>
<tr>
<th>Zone Page</th>
<th>Pkts</th>
<th>Free</th>
<th>LKB</th>
<th>RSB</th>
<th>Hits</th>
<th>Relinks</th>
<th>ACB</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFFFFFFF.7FBB2000</td>
<td>31</td>
<td>29</td>
<td>1</td>
<td>1</td>
<td>34</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>FFFFFFFF.7FBB0000</td>
<td>31</td>
<td>31</td>
<td>0</td>
<td>0</td>
<td>31</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>FFFFFFFF.7FBAE000</td>
<td>31</td>
<td>30</td>
<td>1</td>
<td>0</td>
<td>31</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>FFFFFFFF.7FBAC000</td>
<td>31</td>
<td>29</td>
<td>2</td>
<td>0</td>
<td>31</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>FFFFFFFF.7FBAE000</td>
<td>31</td>
<td>27</td>
<td>0</td>
<td>4</td>
<td>33</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>FFFFFFFF.7FB4000</td>
<td>31</td>
<td>9</td>
<td>11</td>
<td>11</td>
<td>31</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>FFFFFFFF.7FBA8000</td>
<td>31</td>
<td>28</td>
<td>0</td>
<td>3</td>
<td>32</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>FFFFFFFF.7FBA6000</td>
<td>31</td>
<td>31</td>
<td>0</td>
<td>0</td>
<td>31</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

Lock Manager Poolzone:

- **Poolzone Region Address:** FFFFFFFF.80D0A180
- **Packet Size:** 00000100 (256.)
- **Number of Pages:** 00000080 (128.)
- **Maximum Number of Pages:** 000081DA (33242.)
- **Free Page Count:** 0000074A (1866.)
- **Hits:** 00000B9A (2970.)
- **Misses:** 00000000 (0.)
- **Poolzone Expansions:** 00000080 (128.)
- **Allocation Failures:** 00000000 (0.)
- **Allocation not from 1st Page:** 00000000 (0.)
- **Empty Pages:** 0000002F (47.)

---

**Compaq Confidential**

www.compaq.com

Compaq Confidential
```
<table>
<thead>
<tr>
<th>Counters</th>
<th>CPU Id</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>LKB Allocations (cache)</td>
<td></td>
<td>97872</td>
<td>5875</td>
<td>6521</td>
<td>110268</td>
</tr>
<tr>
<td>RSB Allocations (cache)</td>
<td></td>
<td>97680</td>
<td>5879</td>
<td>6555</td>
<td>110114</td>
</tr>
<tr>
<td>New Lock Requests (local)</td>
<td></td>
<td>62208</td>
<td>6454</td>
<td>7049</td>
<td>75711</td>
</tr>
<tr>
<td>New Lock Requests (in)</td>
<td></td>
<td>1242</td>
<td>0</td>
<td>0</td>
<td>1242</td>
</tr>
<tr>
<td>New Lock Requests (out)</td>
<td></td>
<td>35347</td>
<td>0</td>
<td>0</td>
<td>35347</td>
</tr>
<tr>
<td>Conversion Requests (loc)</td>
<td></td>
<td>41231</td>
<td>1864</td>
<td>2023</td>
<td>45118</td>
</tr>
<tr>
<td>Conversion Requests (in)</td>
<td></td>
<td>1065</td>
<td>0</td>
<td>0</td>
<td>1065</td>
</tr>
<tr>
<td>Conversion Requests (out)</td>
<td></td>
<td>3058</td>
<td>1</td>
<td>3</td>
<td>3062</td>
</tr>
<tr>
<td>Dequeue Requests (local)</td>
<td></td>
<td>62173</td>
<td>6180</td>
<td>6337</td>
<td>74690</td>
</tr>
<tr>
<td>Dequeue Requests (in)</td>
<td></td>
<td>539</td>
<td>0</td>
<td>0</td>
<td>539</td>
</tr>
<tr>
<td>Dequeue Requests (out)</td>
<td></td>
<td>35268</td>
<td>15</td>
<td>37</td>
<td>35320</td>
</tr>
<tr>
<td>$ENQ Requests that Wait</td>
<td></td>
<td>28</td>
<td>5</td>
<td>8</td>
<td>41</td>
</tr>
<tr>
<td>$ENQ Requests not Queued</td>
<td></td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>Blocking ASTs (local)</td>
<td></td>
<td>10</td>
<td>3</td>
<td>2</td>
<td>15</td>
</tr>
<tr>
<td>Blocking ASTs (in)</td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Blocking ASTs (out)</td>
<td></td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Directory Functions (in)</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Directory Functions (out)</td>
<td></td>
<td>121563</td>
<td>29</td>
<td>59</td>
<td>121651</td>
</tr>
</tbody>
</table>
```
**SDA> SHOW LOCK/BRIEF**

<table>
<thead>
<tr>
<th>Address</th>
<th>Lockid</th>
<th>ParentId</th>
<th>PID</th>
<th>BLKAST</th>
<th>SubLocks</th>
<th>RQ</th>
<th>GR</th>
<th>Queue</th>
<th>RSB Address</th>
<th>Resource Name</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFFFFFFF.7FE63E50 01000001 00000000 00000000 00000000</td>
<td>01000001</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>0</td>
<td>EX</td>
<td>Granted</td>
<td>FFFFFFFF.7FE63D50</td>
<td>SYS$SYS_ID.O....</td>
<td>Exec</td>
<td></td>
</tr>
<tr>
<td>FFFFFFFF.7FE63C50 04000002 00000000 00000000 80CD8688</td>
<td>04000002</td>
<td>00000000</td>
<td>00000000</td>
<td>80CD8688</td>
<td>0</td>
<td>PR</td>
<td>Granted</td>
<td>FFFFFFFF.7FE64150</td>
<td>CACHE$cmKNOTS_SYS ã7..</td>
<td>Kern</td>
<td></td>
</tr>
<tr>
<td>FFFFFFFF.7FE63A50 01000003 00000000 00000000 00000000</td>
<td>01000003</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>0</td>
<td>CR</td>
<td>Granted</td>
<td>FFFFFFFF.7FE63950</td>
<td>SYS$_DSA0:</td>
<td>Kern</td>
<td></td>
</tr>
<tr>
<td>FFFFFFFF.7FE63850 02000004 00000000 00000000 80CB8100</td>
<td>02000004</td>
<td>00000000</td>
<td>00000000</td>
<td>80CB8100</td>
<td>166</td>
<td>CR</td>
<td>Granted</td>
<td>FFFFFFFF.7FE63750</td>
<td>F11BsvKNOTS_SYS</td>
<td>Kern</td>
<td></td>
</tr>
<tr>
<td>FFFFFFFF.7FE63650 1E000005 00000000 00000000 80CCAD58</td>
<td>1E000005</td>
<td>00000000</td>
<td>00000000</td>
<td>80CCAD58</td>
<td>0</td>
<td>CR</td>
<td>Granted</td>
<td>FFFFFFFF.7FE63550</td>
<td>F11BsbKNOTS_SYS</td>
<td>Kern</td>
<td></td>
</tr>
<tr>
<td>FFFFFFFF.7FE63450 1C000006 00000000 00000000 80CCAD78</td>
<td>1C000006</td>
<td>00000000</td>
<td>00000000</td>
<td>80CCAD78</td>
<td>0</td>
<td>PR</td>
<td>Granted</td>
<td>FFFFFFFF.7FE63350</td>
<td>F11BsaKNOTS_SYS ....</td>
<td>Kern</td>
<td></td>
</tr>
<tr>
<td>FFFFFFFF.7FE63250 01000007 00000000 00000000 00000000</td>
<td>01000007</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>0</td>
<td>CR</td>
<td>Granted</td>
<td>FFFFFFFF.7FE63150</td>
<td>SYS$_$11$DKA0:</td>
<td>Kern</td>
<td></td>
</tr>
<tr>
<td>FFFFFFFF.7FE63050 01000008 00000000 00000000 00000000</td>
<td>01000008</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>0</td>
<td>NL</td>
<td>Granted</td>
<td>FFFFFFFF.7FE62F50</td>
<td>$DSA0000_$SEQPRM</td>
<td>Kern</td>
<td></td>
</tr>
<tr>
<td>FFFFFFFF.7FE62E50 01000009 00000000 00000000 80D07820</td>
<td>01000009</td>
<td>00000000</td>
<td>00000000</td>
<td>80D07820</td>
<td>0</td>
<td>PR</td>
<td>Granted</td>
<td>FFFFFFFF.7FE62D50</td>
<td>$DSA0000_$SEQCMD</td>
<td>Kern</td>
<td></td>
</tr>
<tr>
<td>FFFFFFFF.7FE62C50 0100000A 00000000 00000000 00000000</td>
<td>0100000A</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>0</td>
<td>NL</td>
<td>Granted</td>
<td>FFFFFFFF.7FE62B50</td>
<td>$DSA0000_$MBRPRM</td>
<td>Kern</td>
<td></td>
</tr>
<tr>
<td>FFFFFFFF.7FE62A50 0100000B 00000000 00000000 80D07978</td>
<td>0100000B</td>
<td>00000000</td>
<td>00000000</td>
<td>80D07978</td>
<td>0</td>
<td>PR</td>
<td>Granted</td>
<td>FFFFFFFF.7FE62950</td>
<td>$DSA0000_$MBRSHP</td>
<td>Kern</td>
<td></td>
</tr>
<tr>
<td>FFFFFFFF.7FE62850 0100000C 00000000 00000000 00000000</td>
<td>0100000C</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>0</td>
<td>NL</td>
<td>Granted</td>
<td>FFFFFFFF.7FE62750</td>
<td>$DSA0000_$IN_SET</td>
<td>Kern</td>
<td></td>
</tr>
<tr>
<td>FFFFFFFF.7FE62650 0100000D 00000000 00000000 00000000</td>
<td>0100000D</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>0</td>
<td>NL</td>
<td>Granted</td>
<td>FFFFFFFF.7FE62550</td>
<td>$DSA0000_$THRPRM</td>
<td>Kern</td>
<td></td>
</tr>
<tr>
<td>FFFFFFFF.7FE62450 0100000E 00000000 00000000 80D078E8</td>
<td>0100000E</td>
<td>00000000</td>
<td>00000000</td>
<td>80D078E8</td>
<td>0</td>
<td>CR</td>
<td>Granted</td>
<td>FFFFFFFF.7FE62350</td>
<td>$DSA0000_$THRHL1</td>
<td>Kern</td>
<td></td>
</tr>
<tr>
<td>FFFFFFFF.7FE62250 0100000F 00000000 00000000 00000000</td>
<td>0100000F</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>0</td>
<td>NL</td>
<td>Granted</td>
<td>FFFFFFFF.7FE62150</td>
<td>$DSA0000_$MRGPRM</td>
<td>Kern</td>
<td></td>
</tr>
<tr>
<td>FFFFFFFF.7FE62050 01000010 00000000 00000000 00000000</td>
<td>01000010</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>0</td>
<td>NL</td>
<td>Granted</td>
<td>FFFFFFFF.7FE65E50</td>
<td>$DSA0000_$MRGVAL</td>
<td>Kern</td>
<td></td>
</tr>
<tr>
<td>FFFFFFFF.7FE65D50 01000011 00000000 00000000 80D07BC0</td>
<td>01000011</td>
<td>00000000</td>
<td>00000000</td>
<td>80D07BC0</td>
<td>0</td>
<td>PR</td>
<td>Granted</td>
<td>FFFFFFFF.7FE65C50</td>
<td>$DSA0000_$MRGSIG</td>
<td>Kern</td>
<td></td>
</tr>
</tbody>
</table>
Future Updates

- Performance
  - Fast Re-master - greatly increase the speed for lock tree re-mastering operations
  - De-Couple the LCKMGR/IOLOCK8 spinlocks when running in a Cluster
  - Provide a High Concurrency Lock Manager
Fast Re-Master

- Fast Re-Master
  - Background on distributed Locking
  - Current Re-master methods
  - Fast Re-master plans
Local Locking Operation

- **Process issuing $ENQ**

**Note:**
- Operation occurs completely in process context
Remote Locking Operation

- Process issuing $ENQ
- Local Lock Manager
- Distributed Lock Manager
- Connection Manager
- SCS
- Port Driver

Note:
- Operation requires about 20 times the CPU on the local node
- Communication latency usually accounts for the largest amount of time for the operations

Process goes into RWSCS state
Code Stack on Local Node
Process made computable
Code Stack on Remote Node
Back to Re-mastering

- The best node to master a resource tree is the node performing the most locking operations
  - In a cluster, this can change over time
  - Re-mastering moves lock trees to another node in the cluster
Current Re-Mastering

Legend
- Resource
- Lock
- Message Buffer

Node A
- Old Master
- New Master Node B

Resource
Lock
Message Buffer
Cost of Re-Mastering

- The number of messages to re-master a lock tree is computed by:
  \[2 \times (9 + \text{local locks on old master and other interested nodes})\]

- If we need to move 50,000 locks from Node A to B, how many messages are needed?
  \[2 \times (9+50,000) = 100,018\]

- If a message is 200 bytes, how many megabytes are sent between A and B?
  \[20,003,600 \text{ bytes or about 20 megabytes}\]
Fast Re-Mastering

- Fast Re-Mastering will change the underlying method of moving lock information to the new master.
- The new method will utilize block transfers:
  - Each transfer will send information for many locks.
  - This will reduce CPU overhead and reduce the latency costs of re-mastering.
- We believe this method will be an order of magnitude faster!
Fast Re-Mastering

Resource Information

Lock Data for locks on the above resource
Fast Re-Mastering

- 50,000 blocks with block transfers (64k buffer):
  - 1152 locks per 64k buffer
  - 43 64k block transfers, 43 9k block transfers, 104 sequenced messages
  - \(43 \times 65536 + 43 \times 9230 + 104 \times 132 = 3.2\) Megabytes
- Re-mastering of small trees (fewer than 5 locks) will continue to re-master with the prior method
- Nodes which don’t understand the new block transfer method will still utilize the old method for re-mastering
Performance Ratio’s New:Old Ratio

- SMCI
- NI
- FDDI
- CI

Locks Moved
De-Coupling IOLOCK8/LCKMGR

- OpenVMS synchronizes access to operating system data via a mechanism known as spinlocks
- CPUs own spinlocks and ownership locks out all other CPUs
- Other CPUs attempting to acquire the same spinlock "spin" until the lock is free
  - This spin time has many names:
    - MP Synch
    - busy wait
    - spin time
  - While spinning, the CPU is usually not doing useful work
De-Coupling Background

- Highly used spinlocks result in:
  - Contention
  - More MP Synch time on a system
- For clusters IOLOCK8 was used to synchronize the lock manager, the connection manager, SCS, and Port drivers
- IOLOCK8 is also heavily used by I/O and numerous other OS components
- Large SMP systems with heavy locking and I/O activity can see high MP Synch time
- Standalone systems use a different spinlock for synchronization (LCKMGR) since OpenVMS V6.2
De-Coupled Operations

- De-Coupling intends to allow the local and distributed lock manager to synchronize with the LCKMGR spinlock in a cluster.
- When a remote locking operation occurs, the distributed lock manager will need to lock IOLOCK8 prior to calling the Connection Manager.
Synchronization of Incoming Msgs

- When the distributed lock manager receives a message, only IOLOCK8 is held when the Connection Manager passes the message to the distributed lock manager.

- Since IOLOCK8 is ranked above LCKMGR, it is illegal to lock LCKMGR while holding IOLOCK8.

- To properly synchronize with the DLM, we would need to fork down to LCKMGR.
NoSpin Locking

- Forking is a costly operation
- We are adding a new feature to spinlocks called NoSpin locking
- It allows a lock to be acquired IF available - if not, an error status is returned
- This feature can be used to attempt to acquire out of rank locks
## De-Couple Code Flow

### Code Stack

<table>
<thead>
<tr>
<th>Port Driver</th>
<th>SCS</th>
<th>Connection Manager</th>
<th>Distributed Lock Manager</th>
<th>Local Lock Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>If Lock Manager Message Attempt to acquire LCKMGR NoSpin</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If Lock Manager Message NoSpin LCKMGR fails, Softint Fork Dispatcher
High Concurrency

- With all the performance work shown, the lock manager can still only perform a single locking operation at a time
- We are also working on a High Concurrency Lock Manager
- A High Concurrency Lock Manager will allow multiple operations to occur in parallel on different CPUs
  - These operations would need to be on different resources
  - No other "Exclusive" operation could be in progress
High Concurrency, Sharelocks

- To arbitrate between concurrent and exclusive threads - we are extending spinlocks to support shared owners: Sharelocks
- A CPU owning a spinlock for shared access doesn’t block CPUs wanting shared access
- A CPU owning a spinlock for shared access blocks CPUs wanting exclusive access - they will spinwait
- A CPU owning a spinlock exclusively blocks CPUs wanting shared access - they will spinwait
High Concurrency - New Lock

- New lock on new resource
  - obtain an LKB and RSB
  - take out LCKMGR shared
  - search hash table for resource
  - Since this is a new resource, convert LCKMGR to exclusive
  - Perform the new lock operation - Add Hash Table Entry, update root RSB list, put lock in process lock queue, etc...
  - release LCKMGR
HC - New Lock on Existing Resource

- New lock on existing resource
  - obtain an LKB and RSB from per-CPU cache queue at IPL 8, no spinlock held
  - take out LCKMGR shared
  - search hash table for resource
  - Lock the RSB
  - Perform the new lock operation
  - Unlock the RSB
  - release LCKMGR
High Concurrency DEQueue

- Dequeue operations
  - take out LCKMGR shared
  - lock the RSB
  - perform most of dequeue operation
  - deallocate the LKB
  - if last lock on resource, deallocate the RSB
  - unlock the RSB
  - unlock the LCKMGR spinlock
High Concurrency - Convert

- Convert operations
  - take out LCKMGR shared
  - lock the RSB
  - perform the conversion operation
  - unlock RSB
  - release LCKMGR
High vs. Low Concurrency

- Operating in the Highly Concurrent mode results in longer code paths due to the additional locking on the resource.
- Performance gains are achieved only when operations are occurring concurrently.
- For systems with a small number of CPUs or low locking activity, running in the traditional Low Concurrency mode may be better.
- We plan to allow both methods on a system and potentially allow dynamic switching between models.