



OpenVMS Distributed Lock Manager Monitoring & Performance

Keith Parris

Systems/Software Engineer
Multivendor Systems Engineering
Global Solutions Engineering
HP Services



Overview of Topics



- Available monitoring tools for the Lock Manager
- How to map OpenVMS' symbolic lock resource names to real physical entities
- Lock request rates and latencies, and how to measure them
- Lock mastership, and why one might care about it
- Dynamic lock tree remastering
- How to detect and prevent lock mastership thrashing
- Lock queues, their causes, and how to detect them
- Examples of real-world problem diagnosis

- OpenVMS system managers have traditionally looked at performance in 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 also provide some valuable clues in identifying performance bottlenecks in other areas

Monitoring tools

- MONITOR utility
 - MONITOR LOCK
 - MONITOR DLOCK
 - MONITOR RLOCK (in VMS 7.3 and above; not 7.2-2)
 - MONITOR CLUSTER
 - MONITOR SCS
- SHOW CLUSTER /CONTINUOUS
- Availability Manager / DECams
- T4
- ECP
- DECps (Computer Associates' Unicenter Performance Management for OpenVMS)

MONITOR LOCK



OpenVMS Monitor Utility
LOCK MANAGEMENT STATISTICS
on node XYZB11
1-JUN-2000 09:16:38.82

	CUR	AVE	MIN	MAX
New ENQ Rate	4362.00	4010.50	3659.00	4362.00
Converted ENQ Rate	690.00	636.66	583.33	690.00
DEQ Rate	4363.33	4011.83	3660.33	4363.33
Blocking AST Rate	6.00	4.83	3.66	6.00
ENQs Forced To Wait Rate	45.00	39.83	34.66	45.00
ENQs Not Queued Rate	0.66	1.16	0.66	1.66
Deadlock Search Rate	0.00	0.00	0.00	0.00
Deadlock Find Rate	0.00	0.00	0.00	0.00
Total Locks	419770.00	419767.50	419765.00	419770.00
Total Resources	180201.00	180194.00	180187.00	180201.00

MONITOR DLOCK



OpenVMS Monitor Utility
DISTRIBUTED LOCK MANAGEMENT STATISTICS
on node XYZB11
1-JUN-2000 09:17:27.77

		CUR	AVE	MIN	MAX
New ENQ Rate	(Local)	902.00	808.00	528.66	993.33
	(Incoming)	2302.00	2149.00	2017.66	2302.00
	(Outgoing)	962.66	950.11	824.33	1063.33
Converted ENQ Rate	(Local)	198.33	199.66	149.33	251.33
	(Incoming)	168.66	222.44	168.66	284.33
	(Outgoing)	181.00	113.22	75.66	181.00
DEQ Rate	(Local)	899.00	807.11	528.00	994.33
	(Incoming)	2300.66	2148.88	2018.66	2300.66
	(Outgoing)	971.00	952.77	824.33	1063.00
Blocking AST Rate	(Local)	0.00	0.33	0.00	0.66
	(Incoming)	1.33	0.88	0.00	1.33
	(Outgoing)	5.66	6.11	3.33	9.33
Dir Functn Rate	(Incoming)	16.33	16.66	14.00	19.66
	(Outgoing)	49.00	24.77	9.66	49.00
Deadlock Message Rate		0.00	0.00	0.00	0.00

MONITOR RLOCK



OpenVMS Monitor Utility
DYNAMIC LOCK REMASTERING STATISTICS
on node KEITH
6-MAY-2002 18:42:35.07

	CUR	AVE	MIN	MAX
Lock Tree Outbound Rate	0.00	0.00	0.00	0.00
(Higher Activity)	0.00	0.00	0.00	0.00
(Higher LCKDIRWT)	0.00	0.00	0.00	0.00
(Sole Interest)	0.00	0.00	0.00	0.00
Remaster Msg Send Rate	0.00	0.00	0.00	0.00
Lock Tree Inbound Rate	0.00	0.00	0.00	0.00
Remaster Msg Receive Rate	0.00	0.00	0.00	0.00

MONITOR CLUSTER



Statistic: CURRENT

OpenVMS Monitor Utility
CLUSTER STATISTICS

4-APR-2001 17:57:18

CPU					MEMORY						
CPU Busy	0	25	50	75	100	%Memory In Use	0	25	50	75	100
XYZB14	57	#####				XYZB15	48	#####			
XYZB13	57	#####				XYZB16	48	#####			
XYZB18	48	#####				XYZB22	48	#####			
XYZB22	42	#####				XYZB21	47	#####			
XYZV	40	#####				XYZB18	46	#####			
XYZB11	40	#####				XYZV	42	#####			
DISK					LOCK						
I/O Operation Rate	0	25	50	75	100	Tot ENQ/DEQ Rate	0	125	250	375	500
\$256\$DPA23:	1312	#####				XYZB14	22297	#####			
\$256\$DPA14:	161	#####				XYZB13	5814	#####			
\$256\$DPA26:	125	#####				XYZB11	1844	#####			
\$256\$DPA122:	78	#####				XYZB20	1452	#####			
\$256\$DPA3500:	77	#####				XYZV	888	#####			
\$DSA6002:	66	#####				XYZB21	857	#####			

MONITOR SCS (example: / ITEM=M_RECEIVED)



OpenVMS Monitor Utility
SCS STATISTICS
on node XYZB11
4-APR-2001 18:04:57.44

Message Receive Rate	CUR	AVE	MIN	MAX
XYZB11	0.00	0.00	0.00	0.00
HS100B	1.33	3.90	1.33	7.33
XYZB21	85.00	120.84	38.66	189.33
HS101B	2.33	2.37	0.66	3.33
HS103B	7.66	8.61	5.33	15.00
HS102B	0.33	0.33	0.00	0.66
XYZB14	169.33	237.17	97.00	400.33
XYZB22	11.00	14.51	2.00	33.00
XYZB16	4.00	5.09	0.00	14.00
XYZB18	46.00	18.46	1.00	46.00
XYZB15	65.00	77.53	30.66	135.33
SYZB13	59.00	93.90	47.66	159.00
XYZV	31.66	56.87	29.00	114.00
XYZB19	5.00	22.70	0.00	49.66
XYZB12	128.66	61.78	33.66	128.66

Monitoring tools

- SHOW CLUSTER/CONTINUOUS
 - ADD CONNECTIONS
 - ADD CR_WAITS



SHOW CLUSTER/CONTINUOUS

View of Cluster from system ID 12345 node: NODE01 6-MAY-2002 14:45:16

SYSTEMS		MEMBERS	CONNECTIONS		COUNTERS
NODE	SOFTWARE	STATUS	LOC_PROC_NAME	CON_STA	CR_WAITS
NODE01	VMS E7.3	MEMBER	SCS\$DIRECTORY	LISTEN	
			MSCP\$DISK	LISTEN	
			MSCP\$TAPE	LISTEN	
			VMS\$SDA_AXP	LISTEN	
			VMS\$VAXcluster	LISTEN	
			SCA\$TRANSPORT	LISTEN	
NODE02	VMS V7.2	MEMBER	SCA\$TRANSPORT	OPEN	0
			VMS\$DISK_CL_DRVR	OPEN	0
			MSCP\$DISK	OPEN	0
			MSCP\$TAPE	OPEN	0
			VMS\$VAXcluster	OPEN	1024
NODE03	VMS V7.3	MEMBER	VMS\$DISK_CL_DRVR	OPEN	0
			SCA\$TRANSPORT	OPEN	0
			VMS\$TAPE_CL_DRVR	OPEN	0
			MSCP\$DISK	OPEN	0
			MSCP\$TAPE	OPEN	0
NODE04	VMS V7.2	MEMBER	VMS\$VAXcluster	OPEN	3
			SCA\$TRANSPORT	OPEN	0

Monitoring tools

- ANALYZE/SYSTEM
 - SDA> SHOW RESOURCE
 - SDA> SHOW LOCK
 - SDA> LCK !New SDA extension, 7.2-2 and above

Monitoring tools

– ANALYZE/SYSTEM

- SHOW LOCK qualifiers (OpenVMS 7.2 and above)

- /WAITING

- Displays only the waiting lock requests (those blocked by other locks)

- /SUMMARY

- Displays summary data and performance counters

- SHOW RESOURCE qualifier (OpenVMS 7.2 and above)

- /CONTENTION

- Displays resources which are under contention

Monitoring tools

– ANALYZE/SYSTEM

- New SDA extension LCK (OpenVMS 7.2-2 and above)
 - SDA> LCK !Shows help text with command summary
- Can display various additional lock manager statistics:
 - SDA> LCK STATISTIC !Shows lock manager statistics
- Can show busiest resource trees by lock activity rate:
 - SDA> LCK SHOW ACTIVE !Shows lock activity
- Can trace lock requests:
 - SDA> LCK LOAD !Load the debug execlt
 - SDA> LCK START TRACE !Start tracing lock requests
 - SDA> LCK STOP TRACE !Stop tracing
 - SDA> LCK SHOW TRACE !Display contents of trace buffer

Availability Manager / DECamds Event Window



Event Log Copyright (C) Digital Equipment Corporation, 1995. All rights reserved.

File Control Customize

```
----Time--- Sev Event - Description -----
17:18:49.78 80 LCKCNT, LARRY possible contention for resource PSCDNS_LARRY
17:18:30.28 80 LCKCNT, MOE possible contention for resource PSCDNS_MOE
17:18:27.28 80 LCKCNT, MOE possible contention for resource .&(e
17:18:27.28 80 LCKCNT, CURLY possible contention for resource F11B$s..a%
```

Availability Manager / DECamsds Lock Contention Log File



From AMDS\$SYSTEM:AMDS\$LOCK_LOG.LOG:

Lock Contention Information Start: 7-DEC-1999 09:50:00.90
Time: 7-DEC-1999 09:50:00.91

Master Node: XYZB14

Resource Name: .E..

Parent Resource Name: RMS\$"..%...SS1 ...

RSB Address: 83F0C640, GGMODE: EX, CGMODE: EX

Hex Representation

01CB0D00	E3230200	(Bytes 0 - 7)
00000000	20202020	(Bytes 8 - 15)
2020C583	00000000	(Bytes 16 - 23)
00000000	00000040	(Bytes 24 - 31)

Status: VALID

DECps Example



Full Analysis

XYZB11 (Compaq AlphaServer GS140)

Page 13

PSPA V2.1.5

Wednesday 17-NOV-1999 06:30 to 07:30

CONCLUSION 5.

{R0300}

There are many lock requests per second that are put into the lock wait queue; applications may be experiencing delays. This situation usually indicates that users are contending for shared resources.

Two common reasons for this symptom:

Applications may inherently cause this behavior and not affect the general workload, so be sensitive to response time degradation to rule this out. These applications might be redesigned to lock resources at a lower level, to lower the contention.

Disk volumes (including solid-state devices) might be under too much contention by too many users across the cluster. Response time problems would affect the users of these disks. Try to redistribute the activity to alleviate the contention.

DECps Example



Lastly you may look at the lock wait queue with SDA to isolate the users who are waiting, and resources they are waiting for.

Total number of samples supporting this conclusion: 20

CONDITIONS

1. ENQUE_LOCKS_FORCED_TO_WAIT_RATE .GT. 1.00 + CPU_VUP_RATING
2. OCCURRENCES .GE. 1

EVIDENCE

Enque lck wait rate	Process w/ highest disk I/O:			Volume w/ highest I/O rate	Time of Occurrence
-----	Username	Imagename	Vol w/hgstIO	-----	-----
557.25	WXYZ	ZZSERVER	SYSFILES	SS1	17-NOV 06:32:00
671.42	WXYZ	ZZSERVER	SYSFILES	SS1	17-NOV 06:34:00
599.68	WXYZ	ZZSERVER	SS100000000D	SS1	17-NOV 06:36:00
689.27	WXYZ	ZZSERVER	SS1000000000	SS1	17-NOV 06:38:00
920.70	WXYZ	ZZSERVER	SS1000000007	SS1	17-NOV 06:40:00

DECps Example

```

+----- CLUSTER Lock -----+
!
!           H-Orig      Out      Enq      Dir Op      R-Orig      !
!           Lck Act     Bound     Wait     Incomg     Lck Act     !
!           (/sec)      (%)      (%)      (/sec)      (/sec)      !
!           -----      -----      -----      -----      -----      !
! Node Average      6991.0      26.8      2.6      24.6      1873.3      !
! Node Minimum       42.1      13.9      0.9      0.0      0.1      !
! Node Maximum      14123.3     72.8      9.7      84.7     14658.5     !
! Cluster Total     118844.3    26.8      2.6      417.8    31845.1     !
+-----+

```

DECps Example

```

+----- CLUSTER Lock -----+
!
!           H-Orig      Out      Enq      Dir Op      R-Orig      !
!           Lck Act    Bound    Wait    Incomg    Lck Act    !
!           (/sec)     (%)      (%)      (/sec)     (/sec)     !
!           -----     -----     -----     -----     -----     !
! XYZA03           42.1      37.3      2.2        0.0        0.3      !
! XYZA07           97.4      61.1      3.9        0.0        0.1      !
! XYZA08          102.2      47.6      2.1        0.0        0.3      !
! XYZA09          105.5      51.3      4.2        0.0        0.4      !
! XYZB11          9433.7      24.9      9.7       56.8     14658.5   !
! XYZB12         11385.7      24.0      1.2       16.0      818.8     !
! XYZB13          4738.7      62.0      3.6       32.9     1051.8     !
! XYZB14         14123.3      13.9      4.6       16.3     10279.1   !
! XYZB15         10011.6      26.0      2.1       33.8     2330.8     !
! XYZB16          7148.1      35.7      1.7       24.8     1428.8     !
! XYZB18         13447.1      21.5      1.0       27.2      88.6      !
! XYZB19         13569.2      19.5      0.9       52.5      18.3      !
! XYZB20          6252.1      42.1      2.7       36.5     906.6      !
! XYZB21         12083.0      22.3      1.1       84.7     126.8     !
! XYZB22         12061.4      23.5      1.2       15.6     131.7     !
! XYZA05           608.7      29.3      1.0        0.0        0.2      !
! XYZB23          3637.9      72.8      3.7       20.8      4.3      !

```

Interactive Demo: Introduction to Available Monitoring Tools

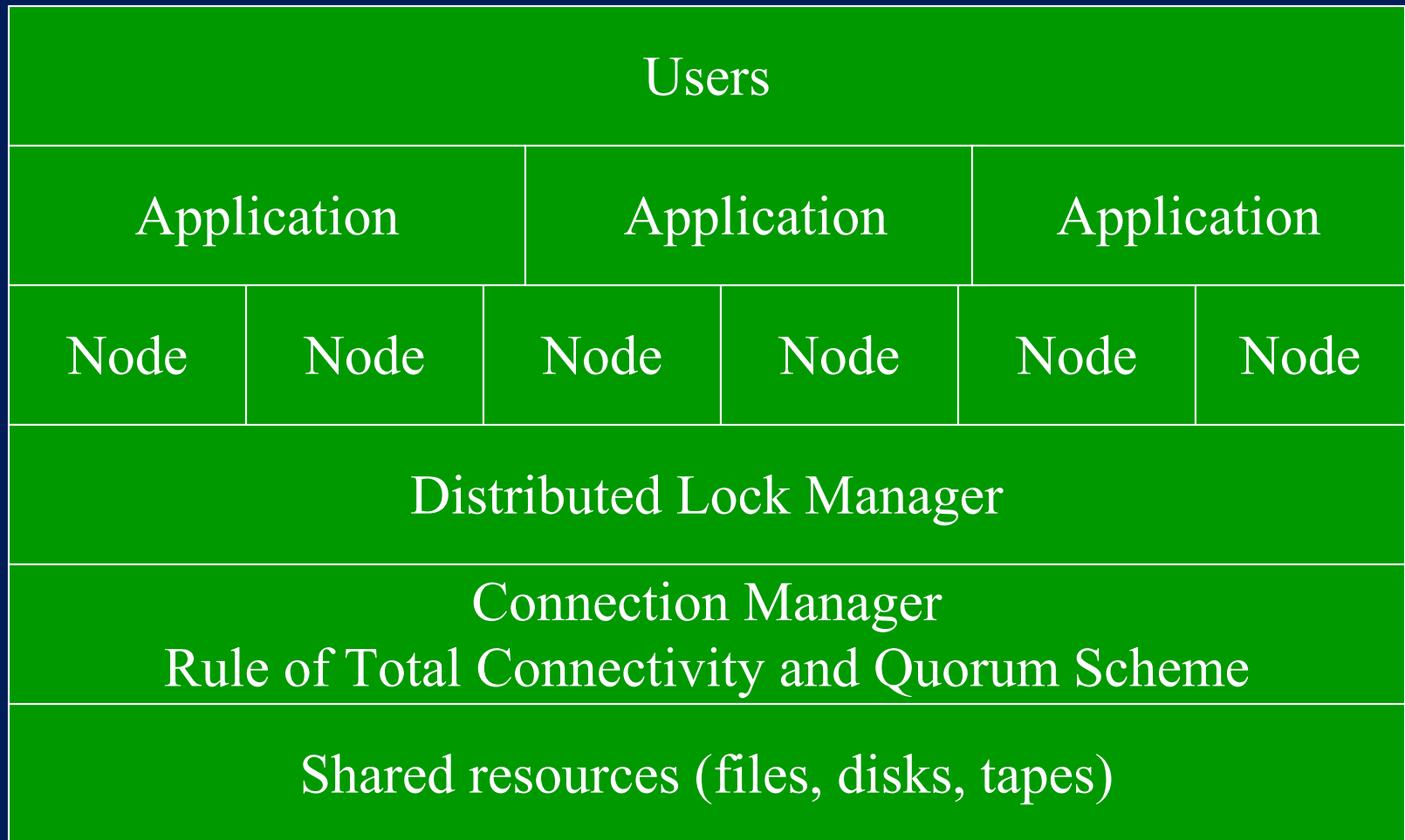


OpenVMS Clusters and the Distributed Lock Manager



- An OpenVMS Cluster is a set of distributed systems which cooperate
- Cooperation requires coordination
- The Distributed Lock Manager is critical to making that coordination possible

Foundation for Shared Access



Distributed Lock Manager



- The Lock Manager provides mechanisms for coordinating access to physical devices, both for exclusive access and for various degrees of sharing

Distributed Lock Manager



- Physical resources that the Lock Manager is used to coordinate access to include:
 - Tape drives
 - Disks
 - Files
 - Records within a fileas well as internal operating system cache buffers and so forth

Distributed Lock Manager



- Physical resources are mapped to symbolic resource names, and locks are taken out and released on these symbolic resources to control access to the real resources

Distributed Lock Manager

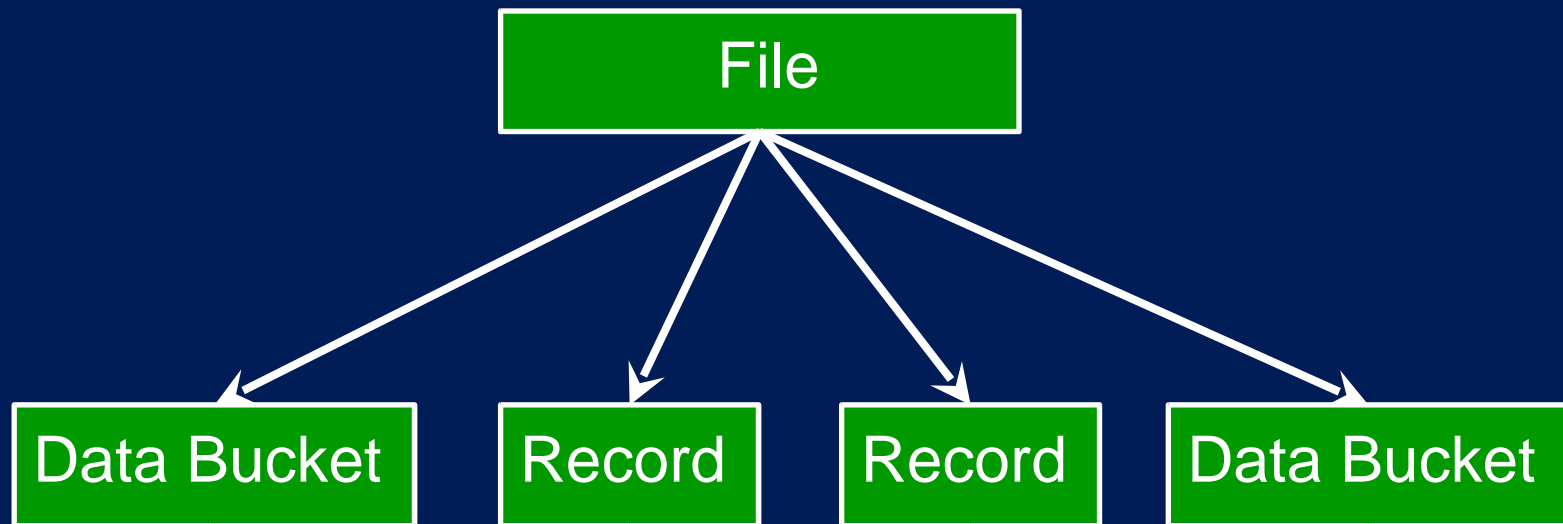


- System services \$ENQ and \$DEQ allow new lock requests, conversion of existing locks to different modes (or degrees of sharing), and release of locks, while \$GETLKI allows the lookup of lock information

OpenVMS Cluster Distributed Lock Manager



- Physical resources are protected by locks on symbolic resource names
- Resources are arranged in trees, e.g.:



- Different resources (disk, file, etc.) are coordinated with separate resource trees, to minimize contention

Symbolic lock resource names

- Symbolic resource names
 - Common prefixes:
 - SYS\$ for OpenVMS executive
 - F11B\$ for XQP, file system
 - RMS\$ for Record Management Services
 - See the book *OpenVMS Internals and Data Structures* by Ruth Goldenberg, et al
 - Appendix H in Alpha V1.5 version
 - Appendix A in Alpha V7.0 version

Resource names

- Example: Device Lock
 - Resource name format is
 - “SYS\$” {Device Name in ASCII text}

3A3333324147442431245F24535953 SYS\$_\$1\$DGA233:

SYS\$ → (SYS\$ facility)

_\$1\$DGA233: → (Device name)

Resource names

- Example: RMS lock tree for an RMS indexed file:
 - Resource name format is
 - “RMS\$” {File ID} {Flags byte} {Lock Volume Name}
 - Identify filespec using File ID
 - Flags byte indicates shared or private disk mount
 - Pick up disk volume name
 - This is label as of time disk was mounted
- Sub-locks are used for buckets and records within the file

Decoding an RMS File Root Resource Name



```
RMS$t.....FDDI_COMMON ...  
000000 204E4F4D4D4F435F49444446 02 000000011C74 24534D52
```

24534D52 RMS\$ → (RMS Facility; RMS File Root Resource)

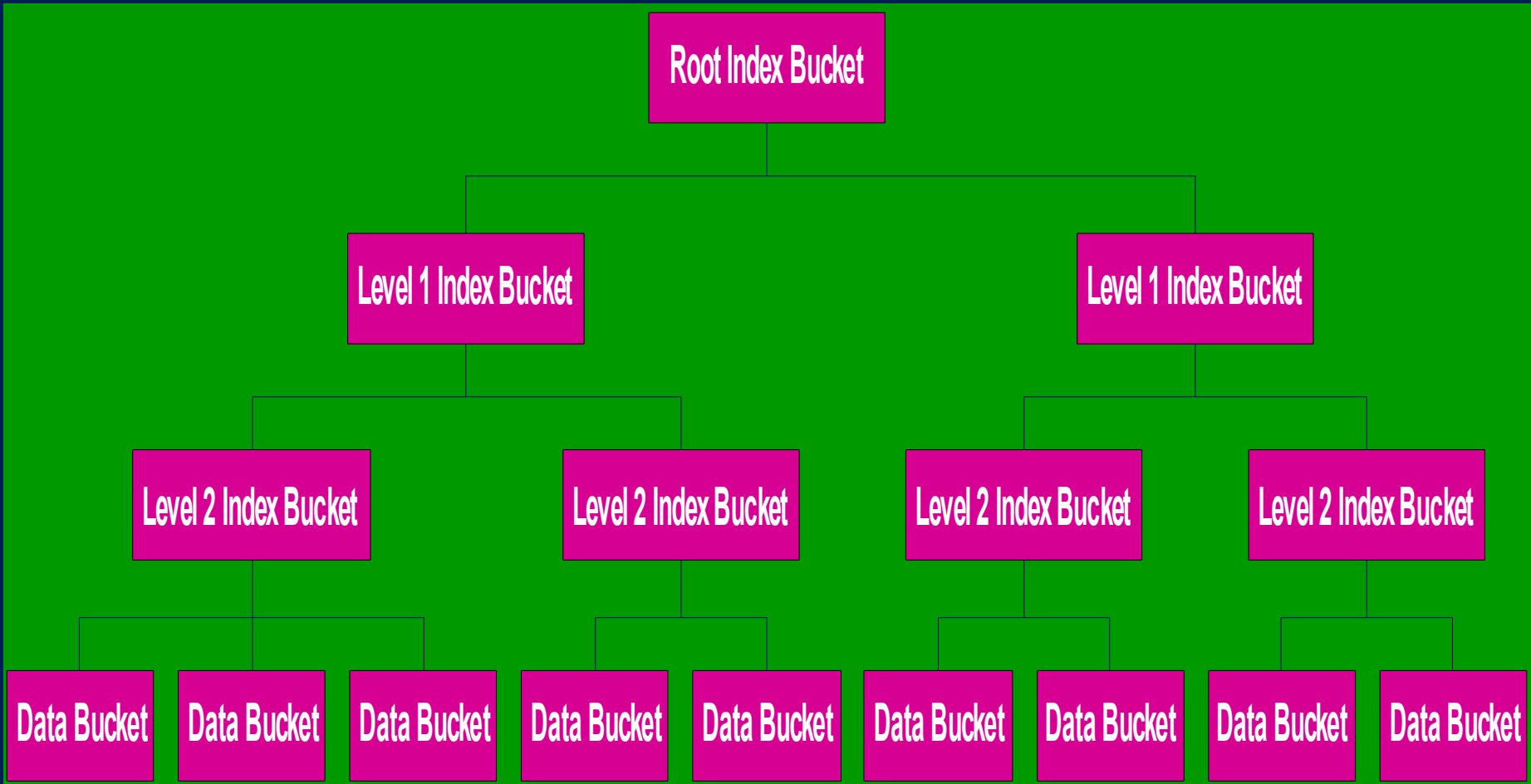
```
00    00    0001          1C74  
RVN   FilX   Sequence_Number   File_Number  
→ File ID [7284,1,0])
```

```
02  
Flags byte (Disk is mounted /SYSTEM)
```

204E4F4D4D4F435F49444446 FDDI_COMMON → (Disk label)

```
$ dump/header/id=7284/block=count=0 -  
    disk$FDDI_COMMON:[000000]indexf.sys  
Dump of file _DSA100:[SYSEXE]SYSUAF.DAT;1 ... → (File)
```

Internal Structure of an RMS Indexed File



RMS Data Bucket Contents

Data Bucket

Data Record	Data Record
Data Record	Data Record
Data Record	Data Record
Data Record	Data Record
Data Record	Data Record

RMS Indexed File Bucket and Record Locks



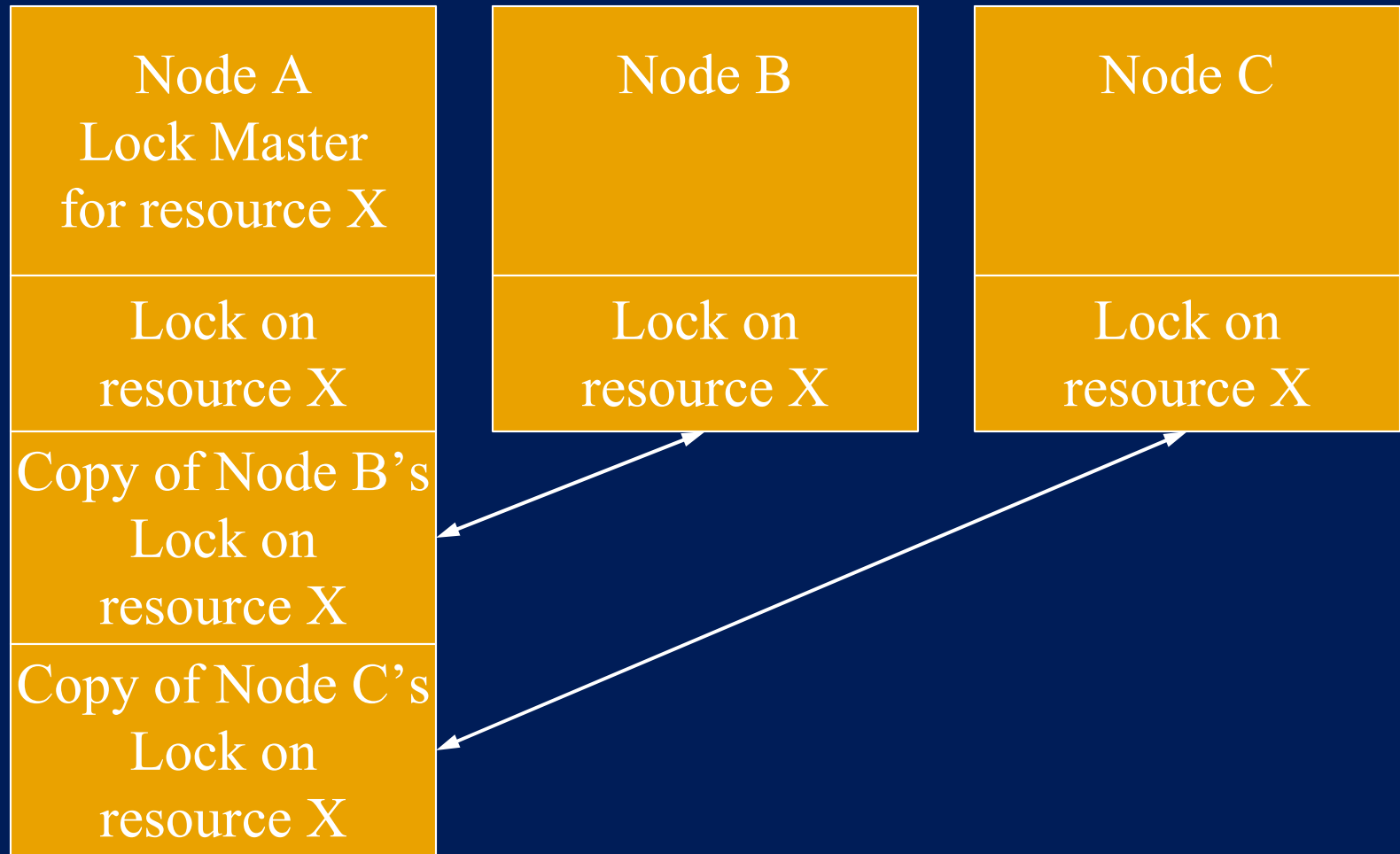
- Sub-locks of RMS File Lock
 - Have to look at Parent lock to identify file
- Bucket lock:
 - 4 bytes: VBN of first block of the bucket
- Record lock:
 - 8 bytes: Record File Address (RFA) of record

Distributed Lock Manager: Lock Master nodes



- OpenVMS assigns a single node at a time to keep track of all the resources in a given resource tree, and any locks taken out on those resources
 - This node is called the ***Lock Master*** node for that tree
 - Different trees often have different Lock Master nodes
 - OpenVMS dynamically moves Lock Mastership duties to the node with the most locking activity on that tree

Distributed Locks



Directory Lookups

- This is how OpenVMS finds out which node is the lock master
- Only needed for 1st lock request on a particular resource tree on a given node
 - Resource Block (RSB) remembers master node CSID
- Basic conceptual algorithm: Hash resource name and index into the lock directory weight vector, which has been created based on LOCKDIRWT values for each node

Lock Directory Weight Vector

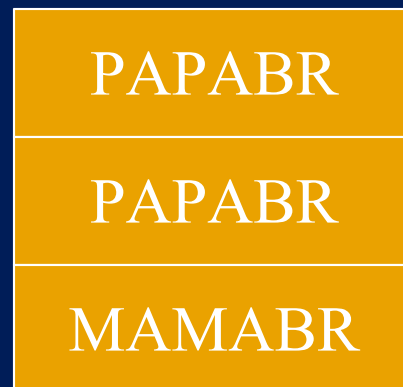


Big node: P AP ABR Lock Directory Weight = 2

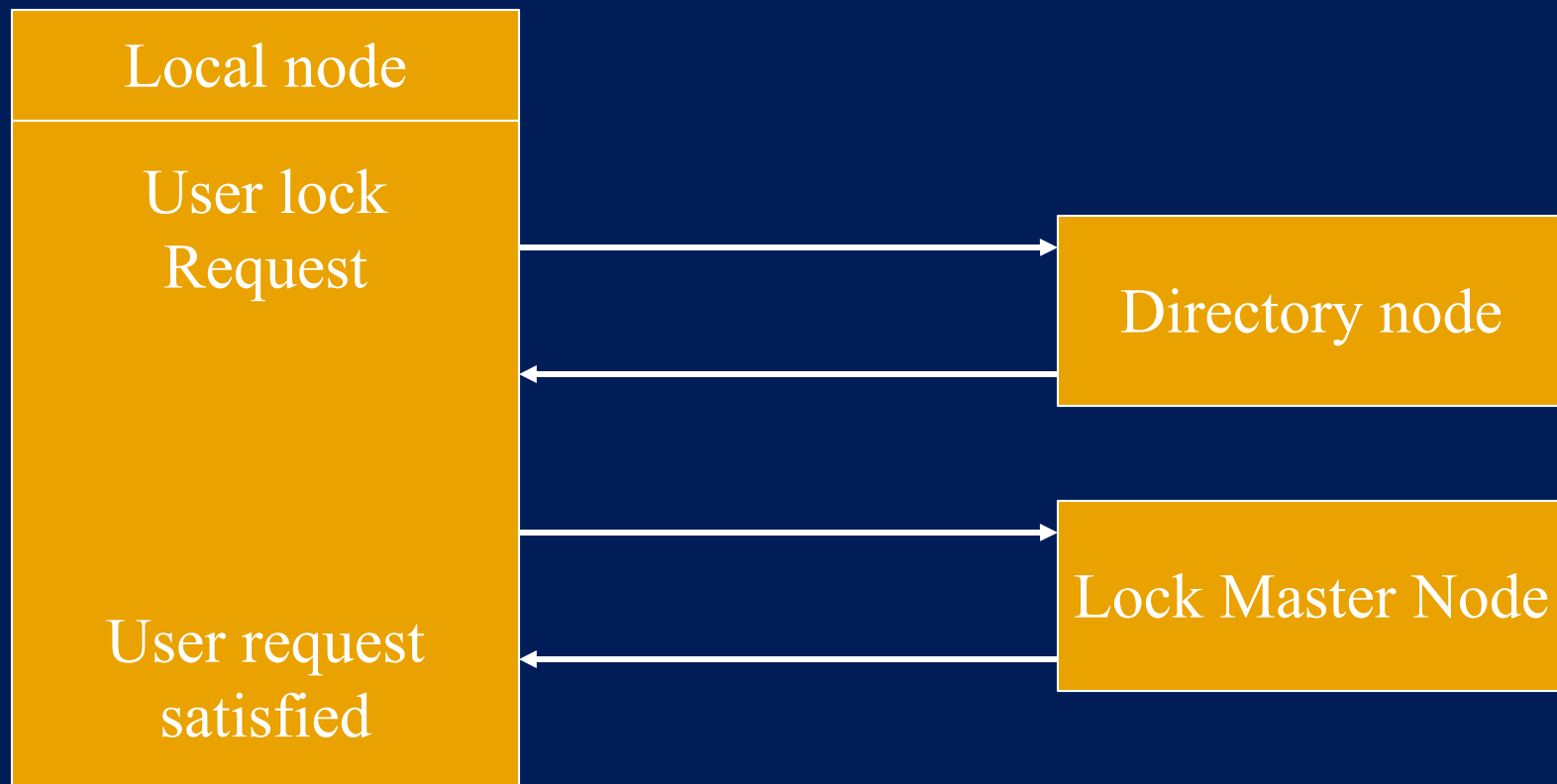
Middle-sized node: M A M ABR Lock Directory Weight = 1

Satellite node: B A B YBR Lock Directory Weight = 0

Resulting lock directory weight vector:



Lock Directory Lookup



Performance Hints

- Avoid \$DEQing a lock which will be re-acquired later
 - Instead, convert to Null lock, and convert back up later
 - Or, create locks as sub-locks of a parent lock that remains held
- This avoids directory lookups
 - and also avoids losing the activity counts in the root RSB used for lock-remastering decisions

- For lock trees which come and go often:
 - A separate program could take out a Null lock on the root resource on each node at boot time, and just hold it “forever”
 - This avoids lock directory lookup operations for that tree

Lock Request Latencies

- Latency depends on several things:
 - Directory lookup needed or not
 - Local or remote directory node
 - \$ENQ or \$DEQ operation (acquiring or releasing a lock)
 - Local (same node) or remote lock master node
 - And if remote, the speed of interconnect used

\$ENQueue

- New lock request (0-2 round trips)
 - No off-node traffic if this node is lock master
 - 1 round trip if:
 - no other node has interest, or
 - directory node is also lock master, or
 - local node is directory node
 - 2 round trips if:
 - directory node is not also the lock master node

\$ENQueue

- Conversion or sub-lock (0 or 1 round-trip)
 - No off-node traffic if this node is lock master
 - 1 round trip to lock master (except 2PC)
 - RSB already contains CSID of lock master node, so we never need to do a directory lookup

\$DEQueue

- 1-way message
- No response expected
- Client doesn't wait
- SCS message guarantee ensures eventual arrival
 - SCS credits may limit number of these in-flight at once

Lock Request Latency: Local vs. Remote



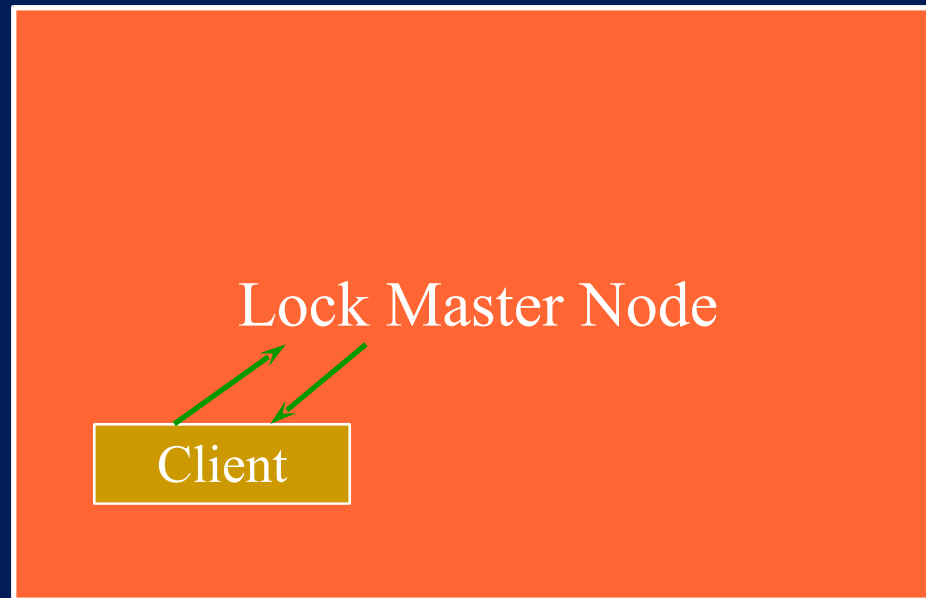
- Local requests are fastest
- Remote requests are significantly slower:
 - Code path ~20 times longer
 - Interconnect also contributes latency
 - Total latency up to 2 orders of magnitude (100x) higher than local requests

Lock Request Latencies

- I used the LOCKTIME program pair written by Roy G. Davis, author of the book *VAXcluster Principles*, to measure latency of lock requests locally and across different interconnects
- LOCKTIME algorithm:
 - Take out 5000 locks on remote node, making it lock master for each of 5000 lock trees (requires ENQLM > 5000)
 - Do 5000 \$ENQs, lock converts, and \$DEQs, and calculate average latencies for each
 - Lock conversion request latency is roughly equivalent to round-trip time between nodes over a given interconnect
- See LOCKTIME.COM from OpenVMS Freeware V6, [KP_LOCKTOOLS] directory

Lock Request Latency: Local

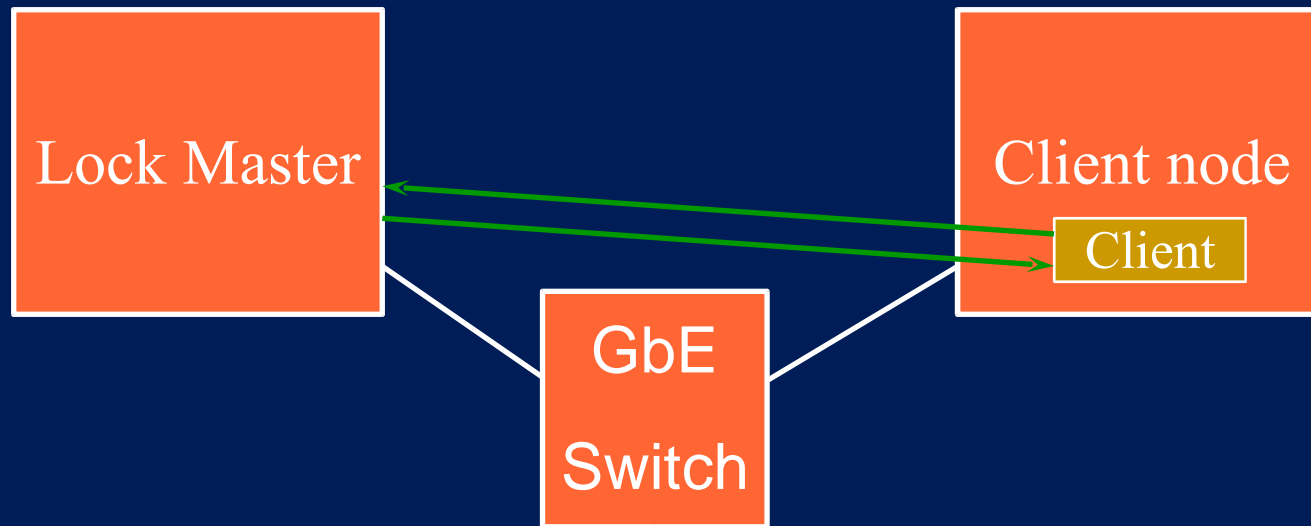
Client process on same node:
2-4 microseconds



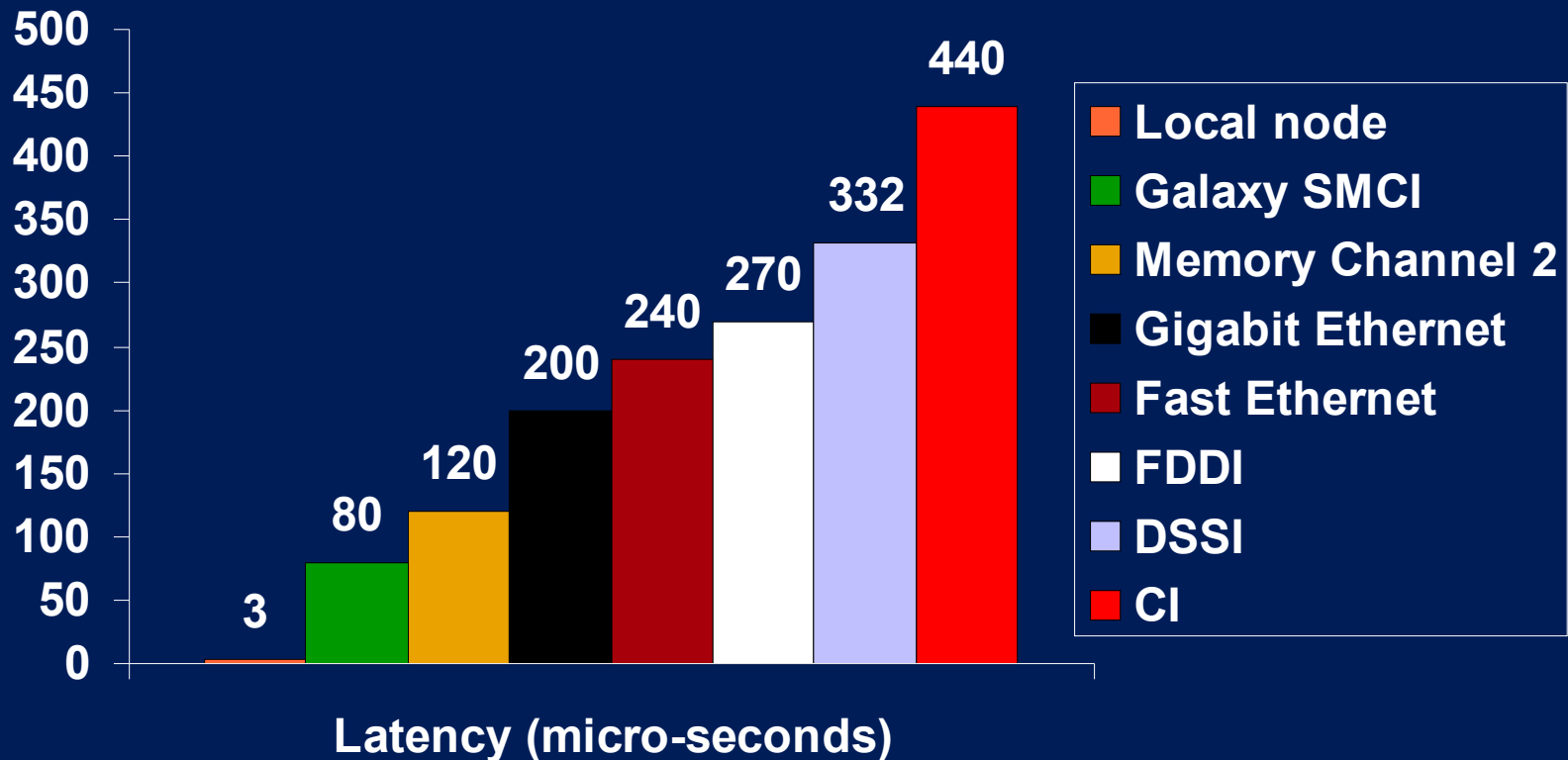
Lock Request Latency: Remote



Client across Gigabit Ethernet:
200 microseconds



Lock Request Latencies



Interactive Demo: Lock Request Latencies



- Local vs. remote
- Different interconnect types

Lock Mastership

- Lock mastership node may change for various reasons:
 - Lock master node goes down -- new master must be elected
 - OpenVMS may move lock mastership to a “better” node for performance reasons
 - LOCKDIRWT imbalance found, or
 - Activity-based Dynamic Lock Remastering
 - Lock Master node no longer has interest

Lock Mastership

- Lock master selection criteria:
 - Interest
 - Only move resource tree to a node which is holding at least some locks on that resource tree
 - Lock Directory Weight (LOCKDIRWT)
 - Move lock tree to a node with interest and a higher LOCKDIRWT
 - Activity Level
 - Move lock tree to a node with interest and a higher average activity level

How to measure locking activity

- OpenVMS keeps counters of lock activity for each resource tree
 - but not for each of the sub-resources
- So you can see the lock rate for an RMS indexed file, for example
 - but not for individual buckets or records within that file
- SDA extension LCK in OpenVMS V7.2-2 and above can show lock rates, and even trace all lock requests if needed. This displays data on a per-node basis.
- Cluster-wide summary is available using LOCK_ACTV*.COM from OpenVMS Freeware V6, [KP_LOCKTOOLS] directory

Lock Remastering

- Circumstances under which remastering occurs, and does not:
 - LOCKDIRWT values
 - OpenVMS tends to remaster to node with higher LOCKDIRWT values, never to node with lower LOCKDIRWT
 - Shifting initiated based on activity counters in root RSB
 - PE1 parameter being non-zero can prevent movement or place threshold on lock tree size
 - Shift if existing lock master loses interest

Lock Remastering

- OpenVMS rules for dynamic remastering decision based on activity levels:
 - assuming equal LOCKDIRWT values
 - 1) Must meet general threshold of experiencing an average of at least 10 lock requests per second (LCK\$GL_SYS_THRSH)
 - 2) New potential master node must have at least 10 *more* lock requests per second than current master (LCK\$GL_ACT_THRSH)

Lock Remastering

- OpenVMS rules for dynamic remastering:
 - 3) Estimated cost to move (based on size of lock tree) must be less than estimated savings (based on lock rate)
 - except if new master meets criteria (2) for 3 consecutive 8-second intervals, cost is ignored
 - 4) No more than 5 remastering operations can be going on at once on a node (LCK\$GL_RM_QUOTA)

Lock Remastering

- OpenVMS rules for dynamic remastering:
 - 5) If PE1 on the current master has a negative value, remastering trees off the node is disabled
 - 6) If PE1 has a positive, non-zero value on the current master, the tree must be smaller than PE1 in size or it will not be remastered

Lock Remastering

- Implications of dynamic remastering rules:
 - LOCKDIRWT must be equal for lock activity levels to control choice of lock master node
 - PE1 can be used to control movement of lock trees OFF of a node, but not ONTO a node
 - RSB stores lock activity counts, so even high activity counts can be lost if the last lock is DEQueued on a given node and thus the RSB gets deallocated

Lock Remastering

- Implications of dynamic remastering rules:
 - With two or more large nodes of equal size running the same application, lock mastership “thrashing” is not uncommon:
 - 10 more lock requests per second is not much of a difference when you may be doing 100s or 1,000s of lock requests per second
 - Whichever new node becomes lock master may then see its own lock rate slow somewhat due to the remote lock request workload

How to Detect Lock Mastership Thrashing

- Detection of remastering activity
 - MONITOR RLOCK in 7.3 and above (not 7.2-2)
 - SDA> SHOW LOCK/SUMMARY in 7.2 and above
 - Change of mastership node for a given resource
 - Check message counters under SDA:
 - SDA> EXAMINE PMS\$GL_RM_RBLD_SENT
 - SDA> EXAMINE PMS\$GL_RM_RBLD_RCVD
 - Counts which increase suddenly by a large amount indicate remastering of large tree(s)
 - SENT: Off of this node
 - RCVD: Onto this node
 - See example procedures WATCH_RBLD.COM and RBLD.COM

SDA> SHOW LOCK/SUMMARY



```
$ analyze/system
```

```
OpenVMS (TM) Alpha system analyzer
```

```
SDA> show lock/summary
```

```
...
```

```
Lock Manager Performance Counters:
```

```
-----
```

```
...
```

```
Lock Remaster Counters:
```

Tree moved to this node	0
Tree moved to another node	0
Tree moved due to higher Activity	0
Tree moved due to higher LOCKDIRWT	0
Tree moved due to Single Node Locks	0
No Quota for Operation	0
Proposed New Manager Declined	0
Operations completed	0
Remaster Messages Sent	0
Remaster Messages Received	0
Remaster Rebuild Messages Sent	0
Remaster Rebuild Messages Received	0

MONITOR RLOCK



OpenVMS Monitor Utility
DYNAMIC LOCK REMASTERING STATISTICS
on node KEITH
6-MAY-2002 18:42:35.07

	CUR	AVE	MIN	MAX
Lock Tree Outbound Rate	0.00	0.00	0.00	0.00
(Higher Activity)	0.00	0.00	0.00	0.00
(Higher LCKDIRWT)	0.00	0.00	0.00	0.00
(Sole Interest)	0.00	0.00	0.00	0.00
Remaster Msg Send Rate	0.00	0.00	0.00	0.00
Lock Tree Inbound Rate	0.00	0.00	0.00	0.00
Remaster Msg Receive Rate	0.00	0.00	0.00	0.00

How to Prevent Lock Mastership Thrashing



- Unbalanced node power
- Unequal workloads
- Unequal values of LOCKDIRWT
- Non-zero values of PE1

Interactive Demo: Lock Tree Remastering



- Inducing remastering by shifting workload
- Lock mastership thrashing between nodes
- Effects of SYSGEN parameter PE1 settings

Impact of Non-zero PE1 Values



- Concern: Locking down remastering with PE1 (to avoid lock mastership thrashing) can result in sub-optimal lock master node selections over time

Mitigating Impact of Non-zero PE1 Values



- Possible ways of mitigating side-effects of preventing remastering using PE1:
 - Adjust PE1 value as high as you can without producing noticeable delays
 - Set PE1 to 0 for short periods, periodically

Deadlock searches

- The OpenVMS Distributed Lock Manager automatically detects lock deadlock conditions, and generates an error to one of the programs causing the deadlock

Deadlock searches

- Deadlock searches can take lots of time and interrupt-state CPU time
- DECps Performance Analysis report can identify when these are occurring
- DEADLOCK_WAIT parameter controls how long we wait before starting a deadlock search

Interrupt-state/stack saturation

- Too much lock mastership, MSCP-serving, etc. workload can saturate the primary CPU on a node
- See with `MONITOR MODES/CPU=n/ALL`
 - where “n” is the number of the Primary CPU as shown by `$SHOW CPU`

Interrupt-state/stack saturation

- FAST_PATH:
 - Can shift interrupt-state workload off primary CPU in SMP systems
 - IO_PREFER_CPUS value of an even number avoids sending interrupts from FAST_PATH devices to the Primary CPU
 - Consider limiting interrupts to a subset of non-primary CPU
 - FAST_PATH for CI since 7.1
 - FAST_PATH for SCSI and FC is in 7.3 and above
 - FAST_PATH for LANs and PEDRIVER in 7.3-2
 - Even with FAST_PATH enabled, the Primary CPU still received the device interrupt, but handed it off immediately via an inter-processor interrupt
 - 7.3-1 allowed FAST_PATH interrupts to bypass the Primary CPU entirely and go directly to a non-primary CPU on hardware platforms which support this
 - No FAST_PATH for Memory Channel (and most likely never will be)
 - No FAST_PATH for Galaxy Shared Memory Cluster Interconnect

Dedicated-CPU Lock Manager

- With 7.2-2 and above, you can choose to dedicate a CPU to do lock management work. This may help reduce MP_Synch time.
- Using this can be helpful when:
 - You have more than 5 CPUs in the system, and
 - You're already wasting more than a CPU's worth in MP_Synch time contending for the LCKMGR spinlock
 - See SDA> SPL extension and SYS\$EXAMPLES:SPL.COM
- LCKMGR_MODE parameter:
 - 0 = Disabled
 - >1 = Enable if at least this many CPUs are running
- LCKMGR_CPUID parameter specifies which CPU to dedicate to LCKMGR_SERVER process

Troubleshooting Real-World Problems



- Techniques:
 - Monitor for
 - High lock rates
 - High lock queues
 - Primary CPU Interrupt-State Saturation
 - SCS credit waits
 - Deadlock Searches and Finds

Interactive Demo: Troubleshooting Real-World Problems



- Induced problems to troubleshoot based on symptoms:
 - High lock queues on directory file
 - High lock rates and large queues on file serialization lock

OpenVMS Cluster DLM Resources



- OpenVMS Documentation on the Web:
 - <http://h71000.www7.hp.com/doc>
 - OpenVMS Cluster Systems
 - Guidelines for OpenVMS Cluster Configurations
- Book “VAXcluster Principles” by Roy G. Davis, Digital Press, 1993, ISBN 1-55558-112-9

Speaker Contact Info:

Keith Parris

- E-mail: Keith.Parris@hp.com or keithparris@yahoo.com
- Web: <http://www2.openvms.org/kparris/>



i n v e n t