OpenVMS Lock Performance

Jeffrey S. Jalbert
JCC Consulting, Inc.

Introduction

- Most computer systems face one of three limits:
  - CPU
  - Memory
  - Disk I/O
- However some Rdb applications add a fourth potential performance wall
  - Performance of OpenVMS Lock Manager

Questions That Need Answers

- How many locks per second can an Open VMS system do before MP Synch swallows the system
- How long does it take to perform a locking operation
- What are the effects of
  - Using the Dedicated Lock Manager
  - Distributed Locking
  - Different hardware
  - Threaded cores on Integrity
  - Relative performance of Alpha and Integrity servers
  - Running on various emulators
  - Exchanging lock value blocks
  - Effects of sizing of RESHASTBL and LOCKIDTBL

How to Develop Answers

- Being an empiricist by training, the only way to answer these questions is to actually experiment on a variety of machines
  - Alpha and Integrity
  - OpenVMS 8.4
  - Alpha hardware
  - Alpha emulators
    - Emulators were booted from the same SAN disks as the Alpha hardware

Methodology

- Construct a simple C program that exercises the lock manager
  - Uses a lock space of 1 million locks
  - Sub-locks of a parent lock
    - To ensure all locks in cluster mastered by one node
  - Randomly gets/promotes/releases locks in different modes
  - Handle BLAST's
  - No sleeps. Drives as hard as it can
  - Optionally exchanges lock value blocks of specified size
- Run on systems and use VMS monitor to evaluate
  - Single node & cluster
  - Processor mode utilization
  - Total locking rates
- Record locking operations/second and CPU utilization in all modes while increasing the number of processes
  - Allow rates to stabilize at each change
- Methodology repeatable within a few percent upon being repeated

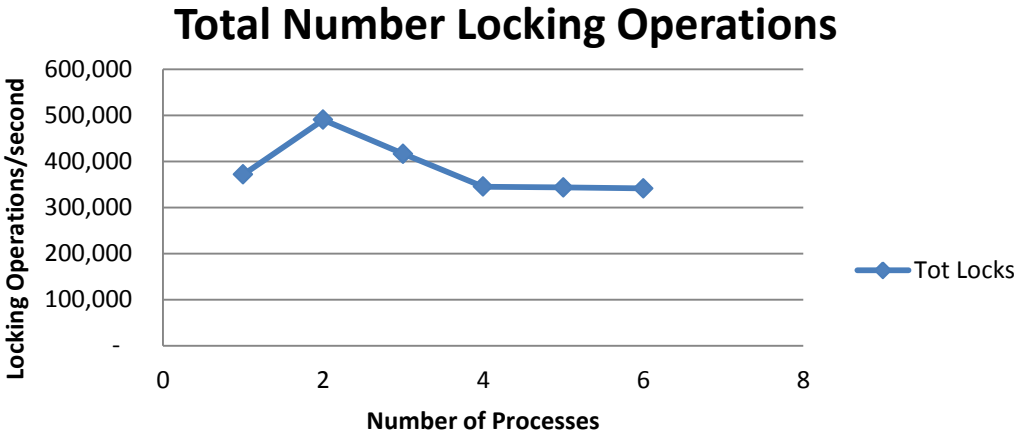
Issues with the Lock Manager

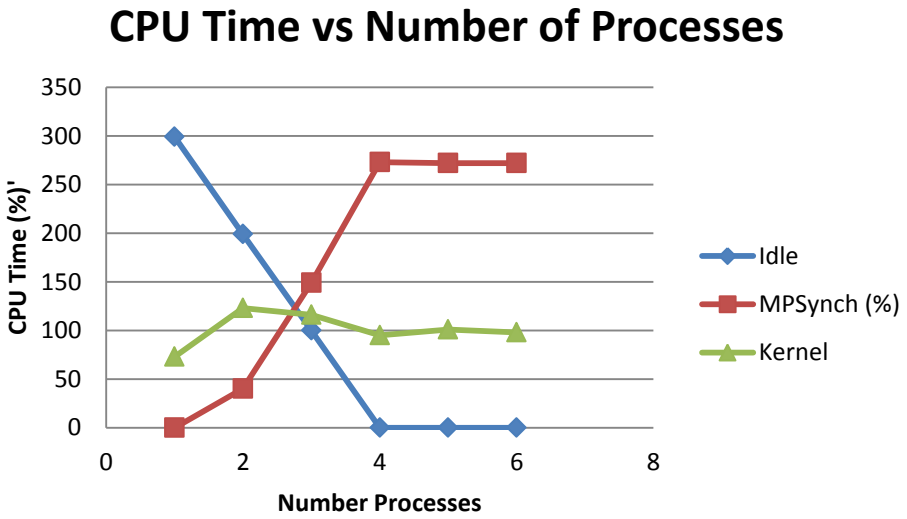
- The lock database consists of multiple data structures
  - In a multi-CPU environment updates are not simple
    - Readers could see inconsistent structures without control
    - Multiple writers could create permanent corruption
  - The data structures are "guarded" by a VMS Spinlock
    - Allows one-at-a-time access
    - Implies potential stalls and limits to performance
  - Remote locking in a cluster will have additional overhead due to communications between nodes

Sample Dataset

HP BL870c (1.59GHz/9.0MB) - 4 cores											
# processes	Idle	MPSynch	Kernel	Exec	Super	User	Total	new ENQ	Convert ENQ	DEQ	Tot Lock
1	299	0	73	0	0	27	399	93025	185311	93021	371,357
2	199	40	123	0	0	39	401	122454	245715	122030	490,199
3	100	149	116	0	0	32	397	104101	207995	104071	416,167
4	0	273	95	0	0	28	396	85999	172520	86361	344,880
5	0	272	101	0	0	30	403	85986	171529	85668	343,183
6	0	272	98	0	0	30	400	85317	170886	85113	341,316

Observations for BL870c

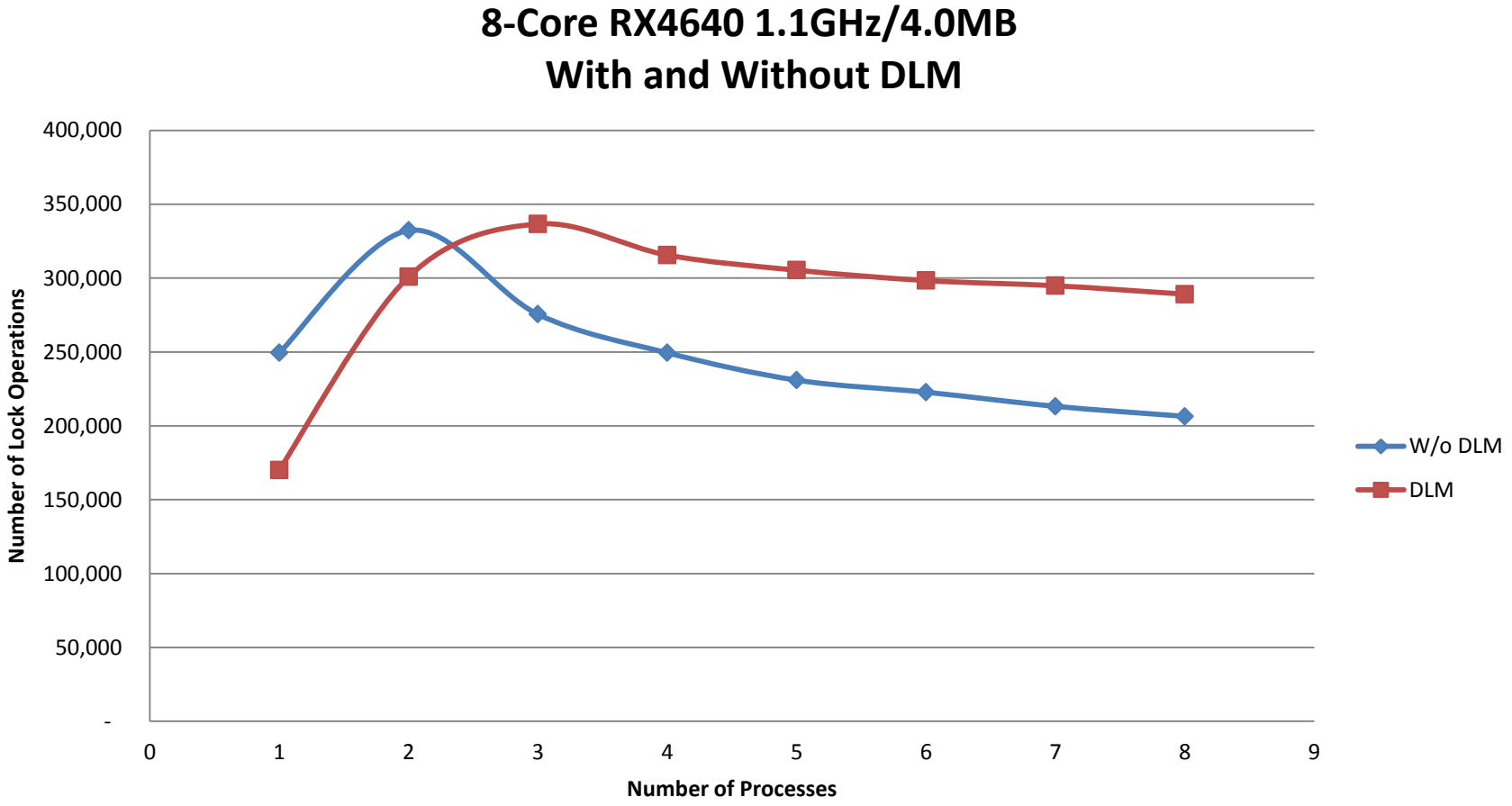




Lessons from BL870

- Max lock operations/second approximately 490,000 per second
  - Your mileage may vary
- Maximum number of lock operations occur after “a little” MP Synch occurs
- Trying to drive a system “harder” results in less work being done

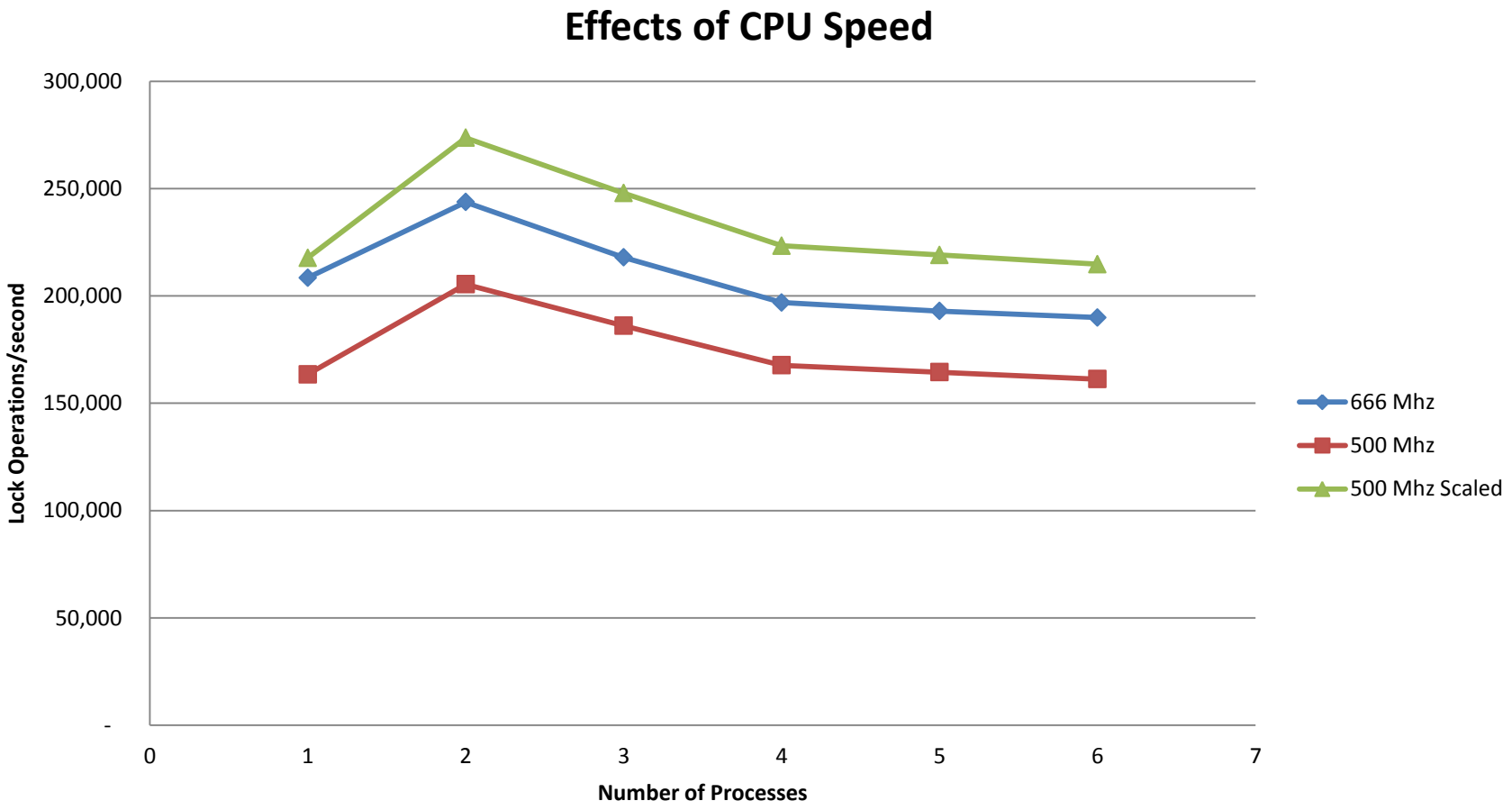
Effects of The Dedicated Lock Manager



Lessons from RX4640

- The DLM consumes 1 core
  - Only 7 cores remain for application
- Max lock operations with DLM is slightly higher than without DLM
- A system with DLM degrades much more gracefully
  - More CPU resources available for other work

Performance of 2 4-CPU ES40's

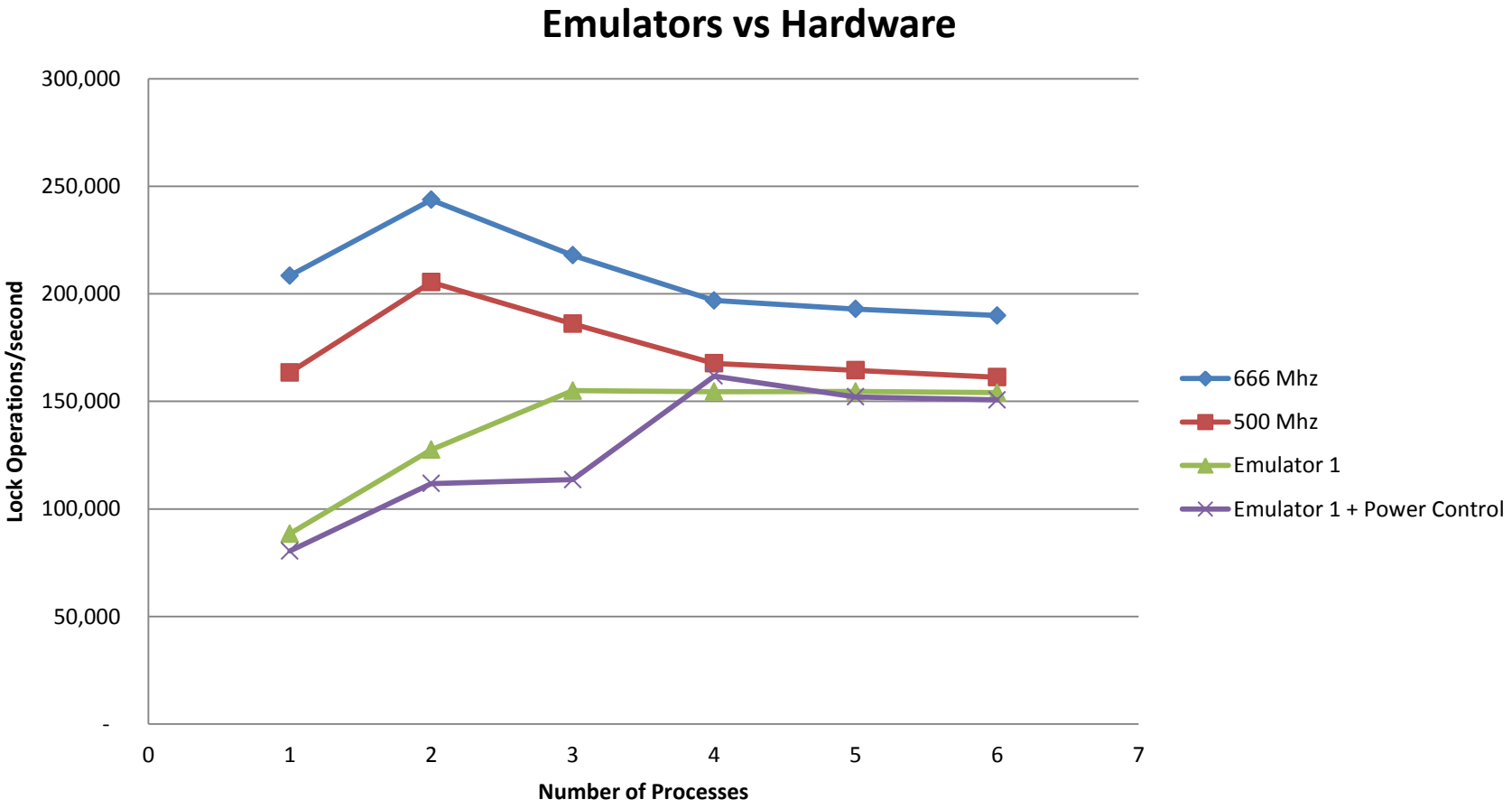


Lessons from 2 ES40's

This is really a CPU performance comparison when on one node.

- CPU speed does make a difference
- But it does not tell the entire story
  - The scaled numbers (multiply 500MHz results by 666/500) are consistently higher than those for the 666 MHz processor
    - CPU Cache
    - Memory access speed
- The entire system architecture is important

Emulators vs. Hardware (ES40 Hardware and Emulator)



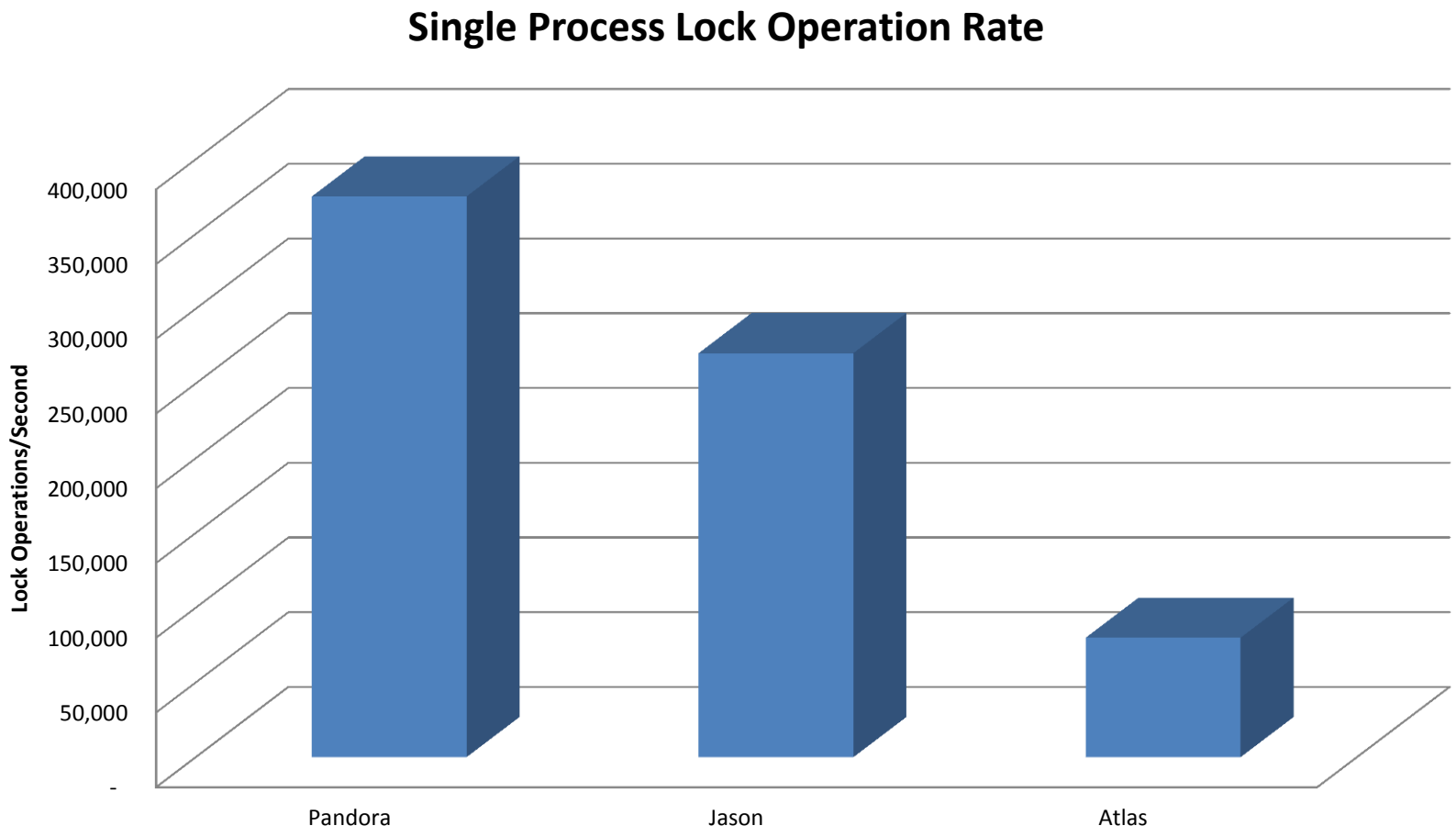
Lessons from Emulator vs. Hardware

- Emulator CPU makes a huge difference
  - The emulator was running on a relatively slow system (AMD Opteron 6174 @ 2.2 GHz) which was selected for cores rather than performance
    - 2 Alphas on this system with 4 ES40 CPUs. 12 cores total
    - Independent network controllers
- Still, in this case with an obsolete version of the emulator software, performance was about 40% of the 666 MHz ES40 hardware
- With saturation, the emulator performance approaches the 500 MHz hardware
- Altering the idle process handling can reduce performance since cores are switched on and off
  - But saves power when system has idle time

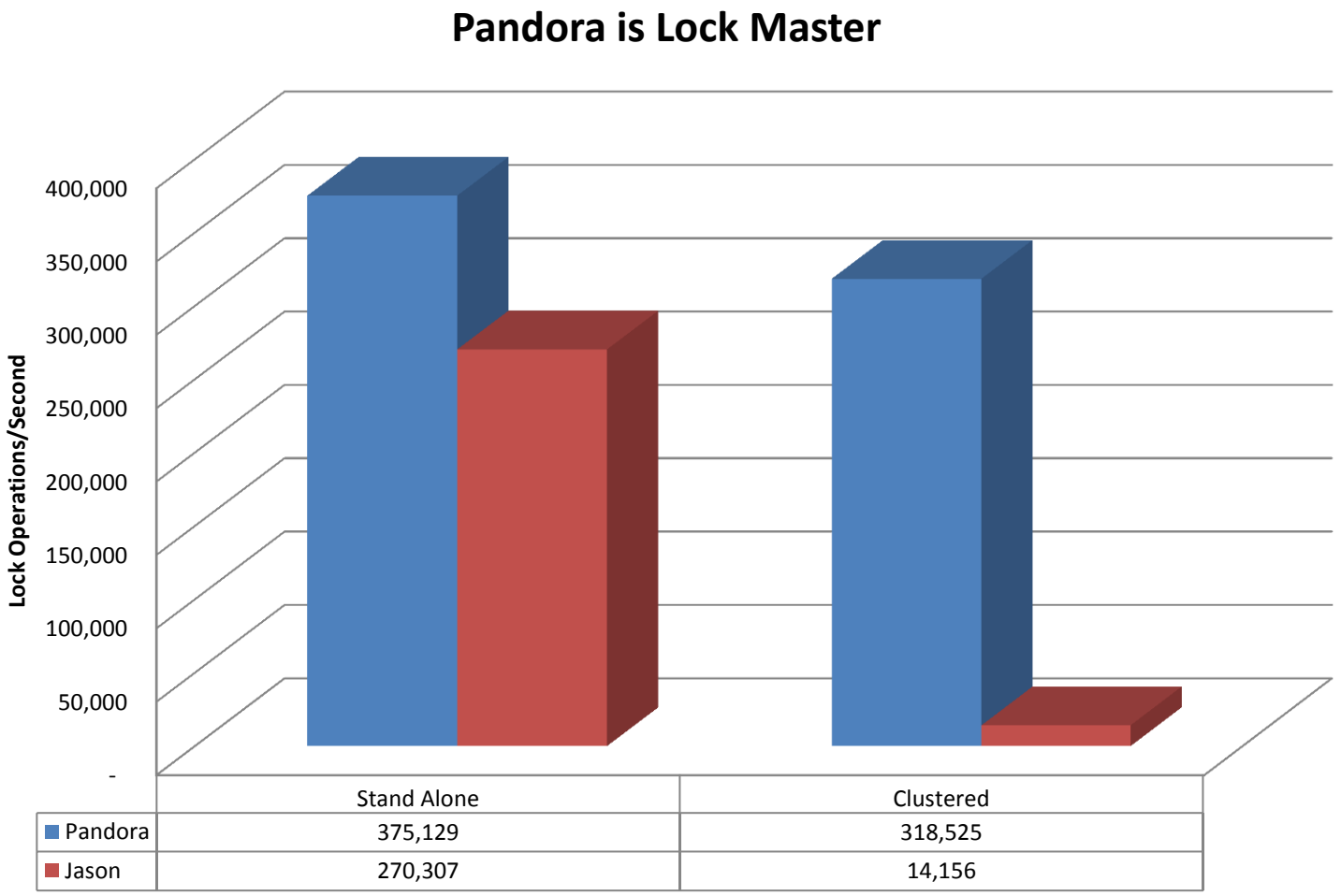
Effect of Remote Locking

- This experiment explored what happens when one is running on multiple nodes in a cluster
  - Three nodes were used, one Alpha emulator and two different Integrity systems
  - One BL870c (1.59 Ghz,/9.0MB) (Pandora – 4 cores)
  - One RX4640 (1.10 GHz/4.0MB) (Jason – 8 cores)
  - ES40 Alpha Emulator running on AMD Opteron 6174 @ 2.2 GHz) (Atlas) 4 Emulated CPUs
- Only one process was run per node
- The experiment was started on different nodes to see the effect of the lock mastering of each CPU
- Ran with 1 node, 2 Integrity nodes and all three nodes

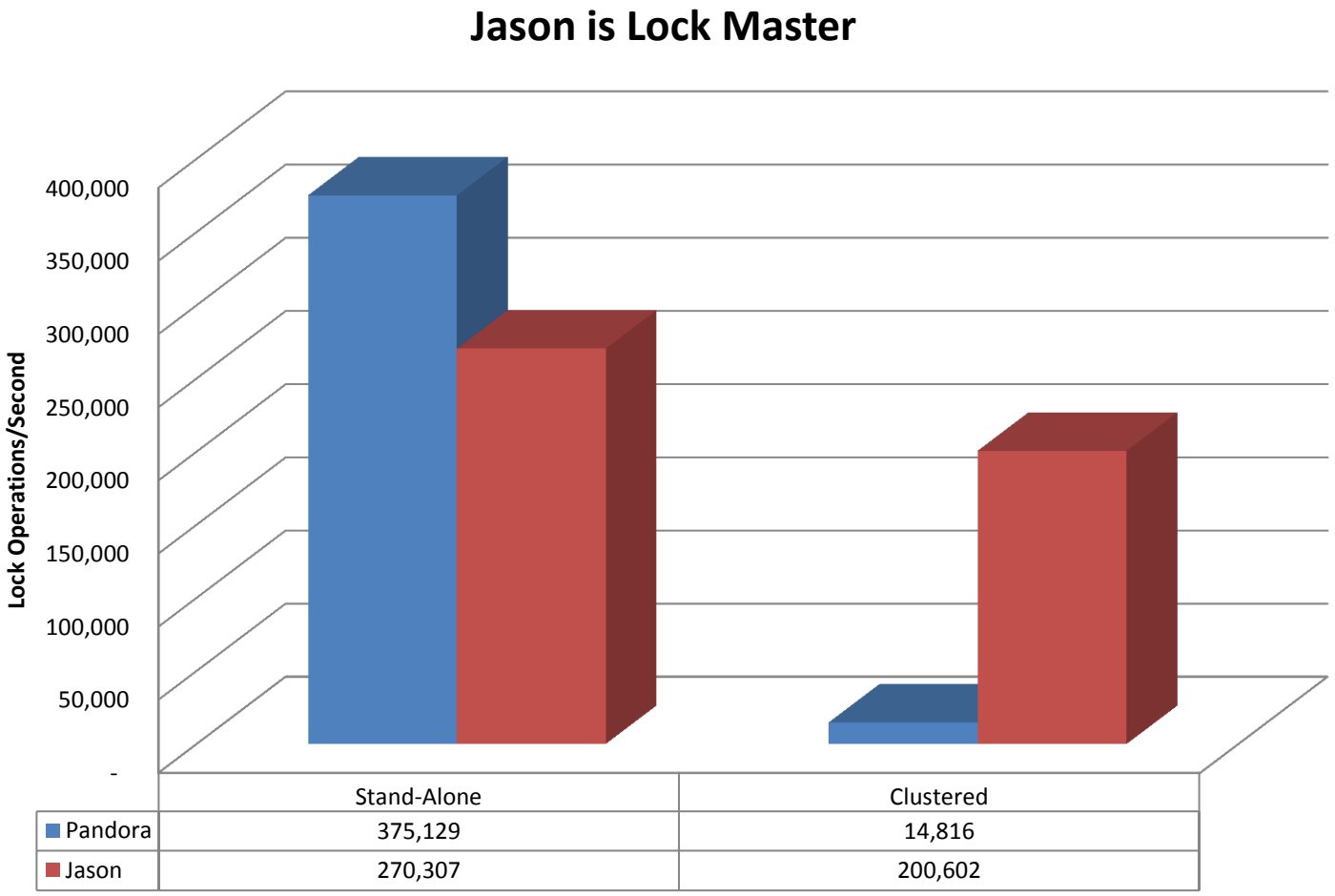
Very Different Node Performance



Pandora and Jason

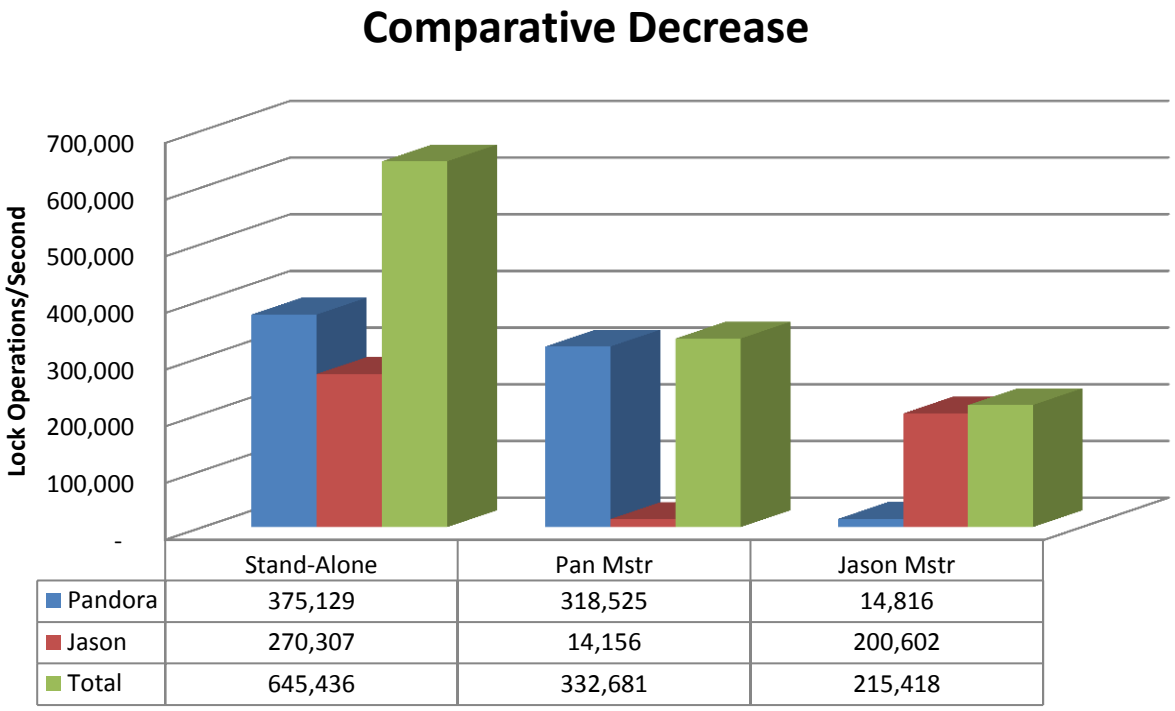


Pandora and Jason



Remote Locking Comparison

When using distributed locking, the aggregate throughput is considerably reduced



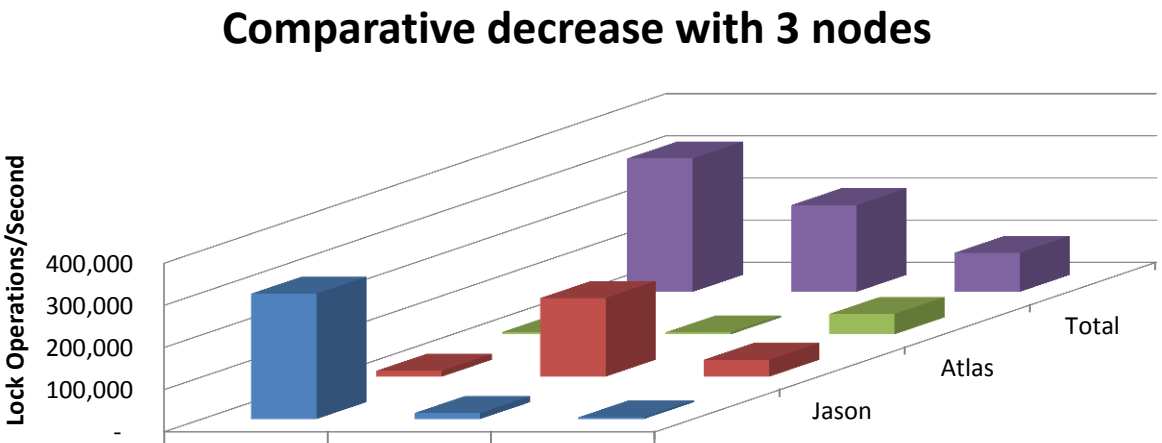
	Stand-Alone	Pan Mstr	Jason Mstr
Pandora	375,129	318,525	14,816
Jason	270,307	14,156	200,602
Total	645,436	332,681	215,418

Results from Pandora and Jason

- Total locking rate is reduced in the cluster compared to what the fastest node can do
  - 85% of fastest node
- When the slower node is the lock master the net throughput is significantly reduced
  - 57% of fastest node

Comparison With 3 Nodes

- Atlas here is an emulated system. Columns indicate lock master



	Pandora	Jason	Atlas
Pandora	297,460	14,543	4,236
Jason	14,151	186,449	40,294
Atlas	5,178	4,773	48,236
Total	316,789	205,765	92,766

Results from Three Nodes

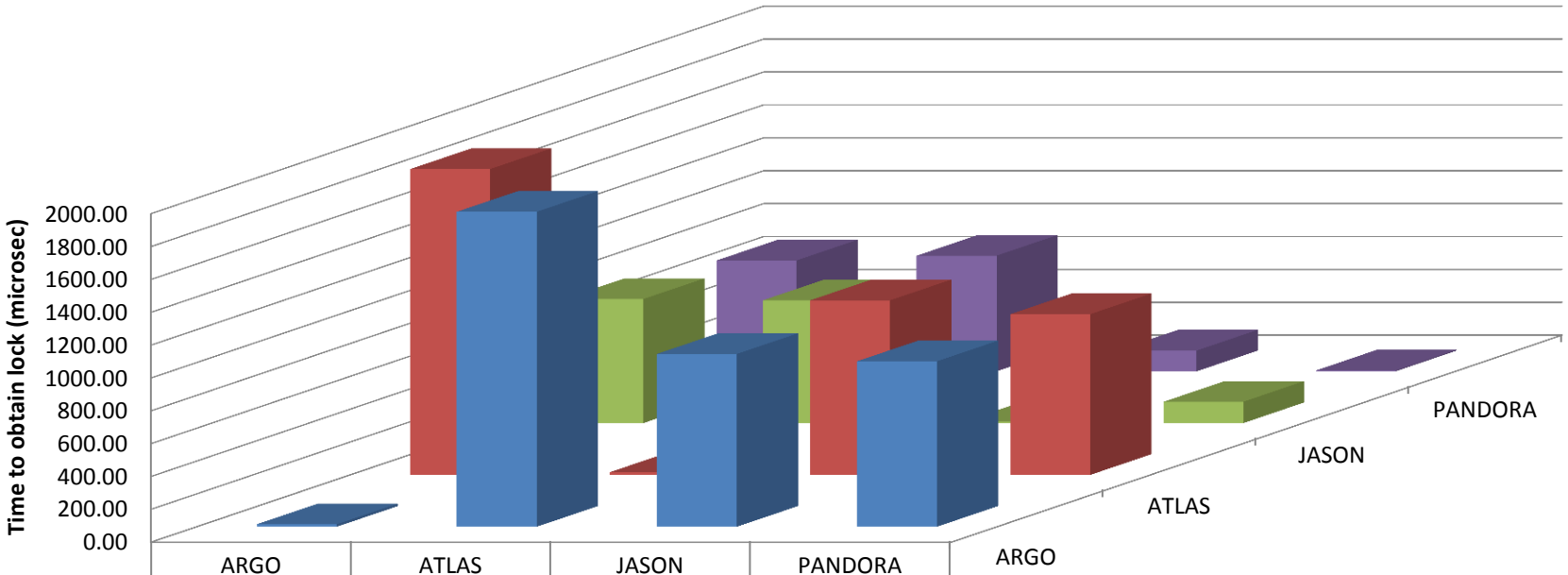
- Total locking rate is reduced again with the addition of a third (slow) node
  - 84% when Pandora is lock master
  - 55% when Jason is lock master
  - 25% when Atlas (emulator) is lock master
- The slower the lock master node the fewer the total locking operations

Time to Obtain Lock

- The OpenVMS Tools Freeware CD V6 Contains tools to measure the time it takes to obtain a lock when different nodes are lock masters.
- SYSGEN parameter set to PE1 = -1 to avoid re-mastering
- Runs many lock requests between nodes one of which, for each test is the lock master
- Measures time in microseconds to acquire a lock
- Done with the 2 emulated systems running on a single host described earlier
- Atlas hardware is 666 MHz ES40
- Argo hardware is 500 MHz ES40
- Columns in graph indicate lock master node

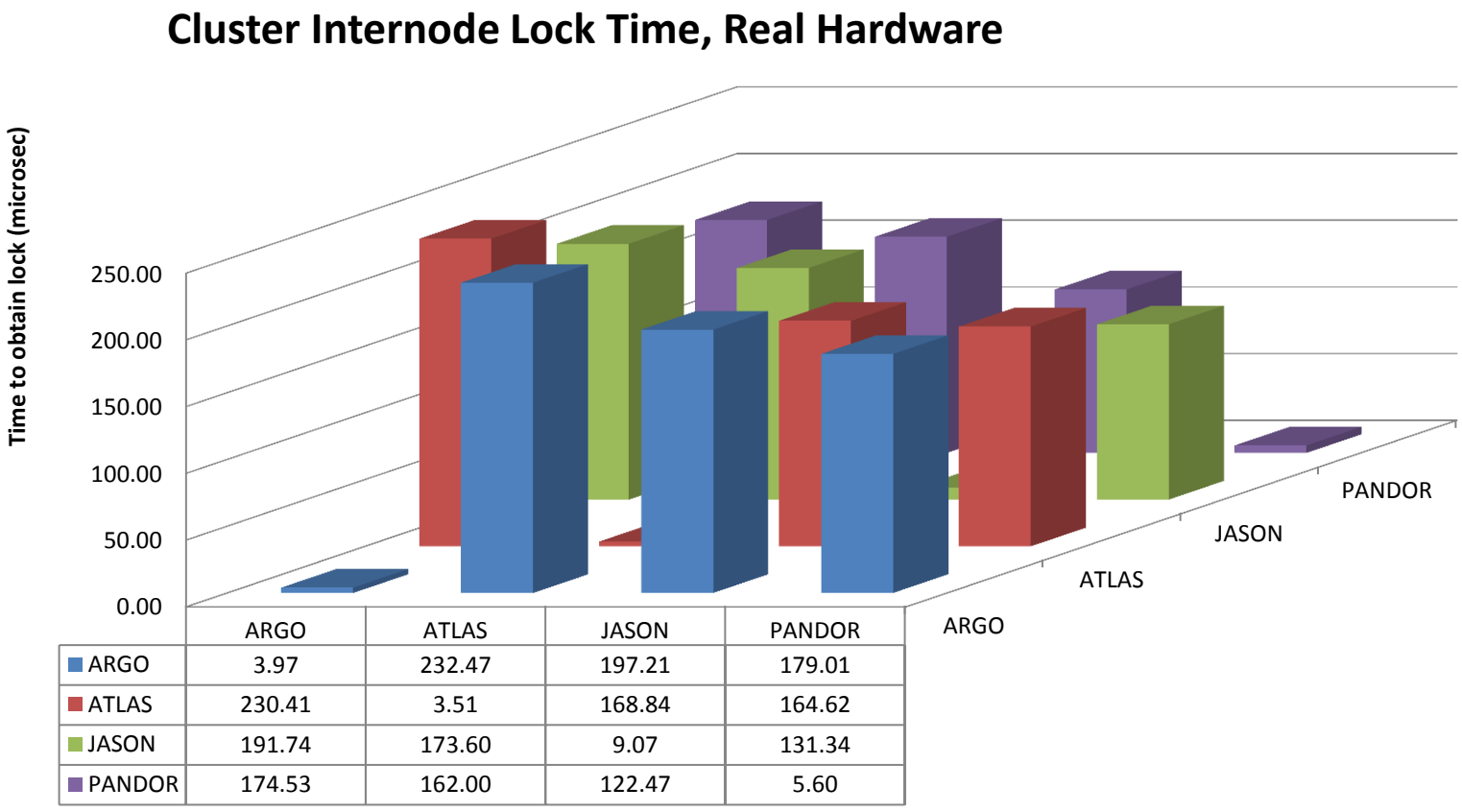
Argo and Atlas are Emulated Systems

Cluster Internode Lock Time, Argo & Atlas are Emulators



	ARGO	ATLAS	JASON	PANDORA
ARGO	14.97	1917.37	1052.37	1007.59
ATLAS	1861.10	17.19	1063.53	978.94
JASON	755.64	749.03	8.87	130.01
PANDORA	675.14	704.14	126.40	5.53

Time to Obtain Lock with Alpha hardware



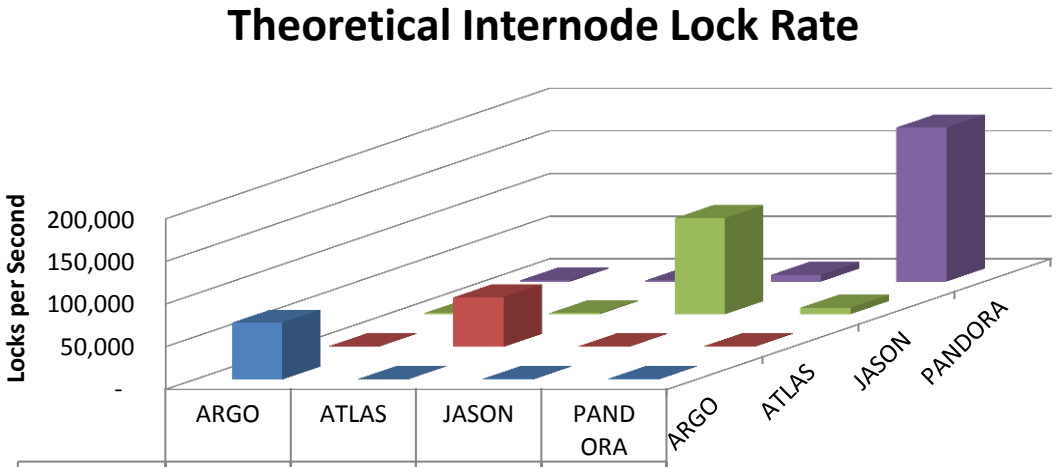
	ARGO	ATLAS	JASON	PANDOR
ARGO	3.97	232.47	197.21	179.01
ATLAS	230.41	3.51	168.84	164.62
JASON	191.74	173.60	9.07	131.34
PANDOR	174.53	162.00	122.47	5.60

Observations on Internode Lock Time

- Remote locks are vastly “more expensive” than local locks
- Using an emulator as lock master significantly increases internode latency
  - Compared to the hardware
  - We have observed similar network-related tardiness in another emulator as well

Potential Max Locks/Second One CPU

- If one uses the previous data to determine the max potential locking rate you get the following (1 process per node)



	ARGO	ATLAS	JASON	PANDORA
ARGO	66,787	522	950	992
ATLAS	537	58,185	940	1,022
JASON	1,323	1,335	112,775	7,692
PANDORA	1,481	1,420	7,911	180,723

Lessons From Cluster

- Distributed locking will significantly reduce the throughput of the cluster's lock management
- Total locking rate is improved if lock master is fastest node

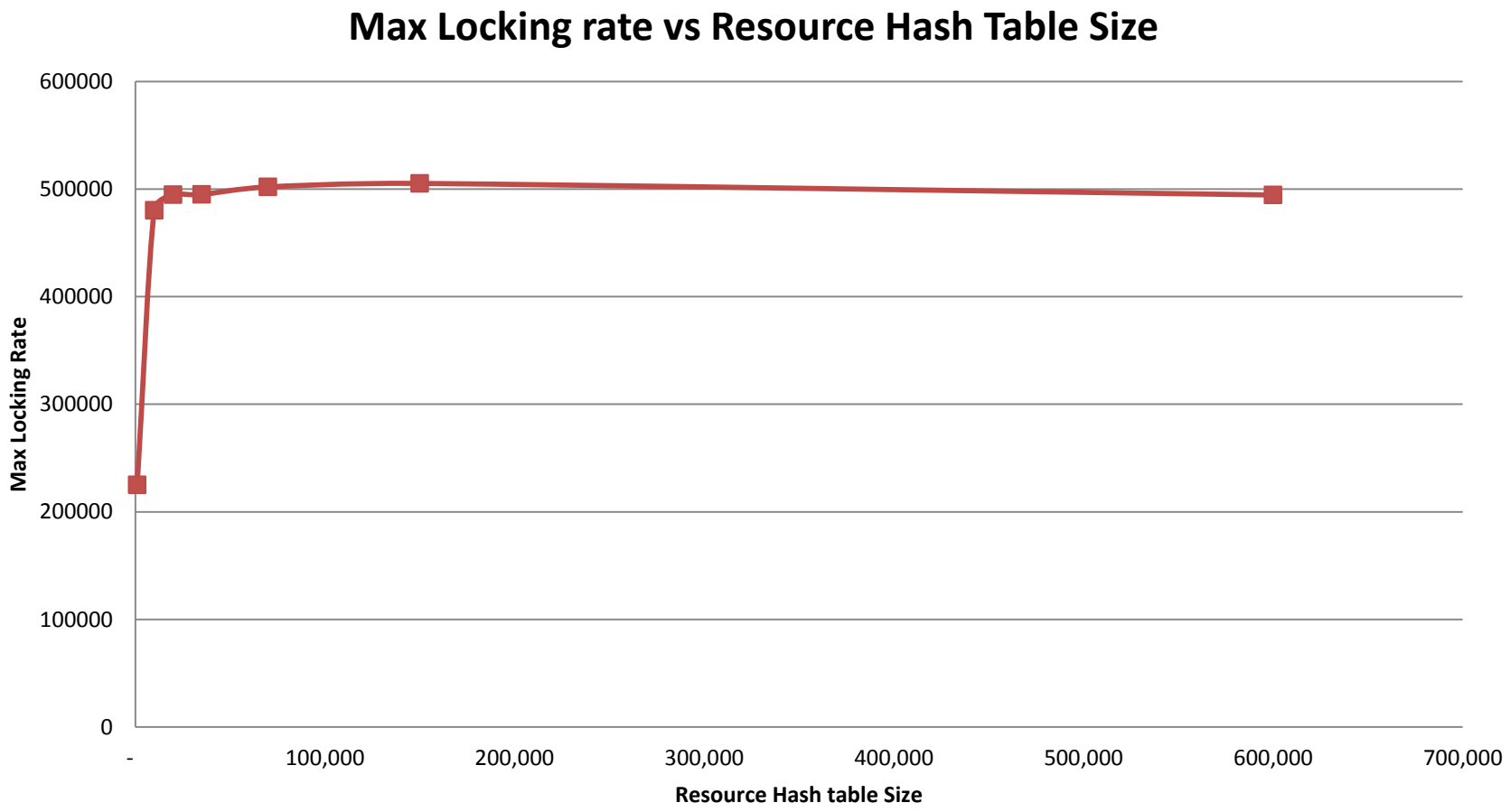
Additional Notes for Cluster

- The three nodes in the cluster were adjacent to one another in a rack.
  - Total length of Ethernet less than 15 meters
  - Best signal propagation time is ~ $5*10^{-8}$ seconds plus switching latency (times two for round trip)
  - Time for a lock on the BL870 is (1/371,357) seconds or $2.7*10^{-6}$ or about 27 times longer than internode signal time
  - If the nodes were separated by 25 KM the total signal propagation would be $1.25*10^{-3}$ seconds plus latency (times two) or a factor of 92 times longer than a lock operation itself, including reduced time because light is slower in glass than air!
- Distributed locking in a wide-area cluster will be about 100 times worse than in the data shown on earlier slides.
  - Entirely due to signal propagation time

Effects of Varying the Resource Hash Table

- One data structure used by the Lock Manager is the Resource Hash Table (RHT)
  - Essentially a hash index on lock names
  - If the table shrinks there are more collisions
  - Collisions have to be sifted individually to find the required lock name
    - Consumes invisible CPU
- Data presented here is the result of varying the size of the resource hash table on the BL870 machine

Effects of Varying Resource Hash Table Size

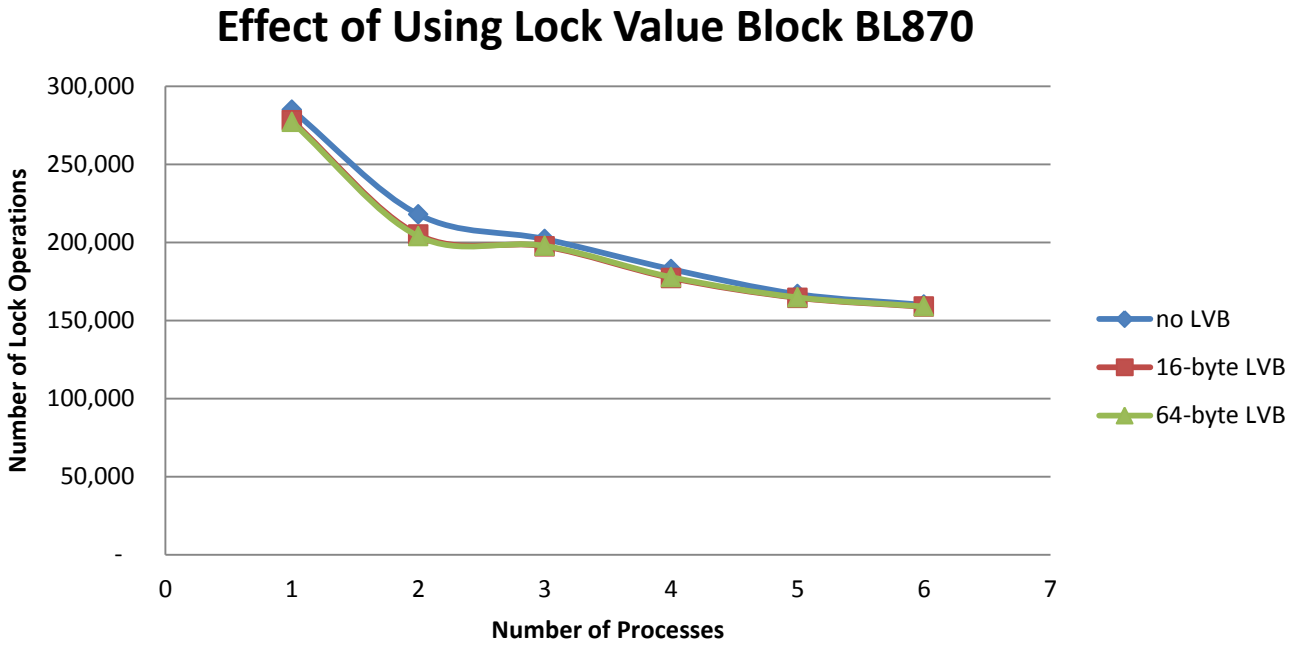


Conclusion Regarding RHT Size

- A significantly undersized table can reduce throughput by a large amount
- Conclusions are unclear regarding a table that is too large
  - Test is limited
  - CPU cache may influence the test with smaller RHT size
  - A larger hash table is not a contributor to performance of the lock manager but does consume memory
    - More memory for OpenVMS = less memory for application
- Do not be afraid to have a large RHT but one that is extremely small is *very* bad

Effect of Lock Value Block

- Every lock may have a lock value block associated with it. The LVB may be 16 or 64 bytes with recent versions of OpenVMS



Conclusions from LVB Data

- There is a slight (1%) reduction in maximum locking rates within one node for using an LVB at all
- An even smaller cost (0.5%) for using the larger LVB

Effects of Hyper Threading

- Integrity cores each host a number of units that do fundamental operations
  - Arithmetic
  - Logic
  - Etc.
- There are a variable number of units of each type
- Instructions are executed in bundles and all instructions in a bundle are executed concurrently
  - May include some NOOP instructions, each element of bundle uses a different unit within the core
- That still allows many units to be idle at any given point
- More bang may be achieved if more units are kept busy
  - Thus the introduction of Hyper threading

What is Hyper Threading?

- Each core presents itself as two logical cores or so-called co-threads
- The idea is that each physical core can thereby keep more of its fundamental parts concurrently busy
- However there is inevitable contention and each co-thread will likely run slower than an unthreaded core
- How much slower depends on the workload

VMS Utilization of CPUs

- VMS sees each co-thread as a CPU
- On VMS all CPUs are ***always*** busy
  - If only doing the NULL process, viewed as IDLE time on $ monitor/modes
- Thus threading a VMS system, if it is lightly loaded, has the possibility of slowing it down
  - Competing with the NULL job
- Also, other critical components of the VMS operating system can experience interference, e.g. Fastpath or the Dedicated Lock Manager

Disabling Co-thread CPUs

- In a VMS system you can see what each CPU is dedicated to:

```
pandor > show fast
Fast Path preferred CPUs on PANDOR 15-JUN-2013 06:31:40.13HP BL870c
(1.59GHz/9.0MB) with 8 active CPUs
Device:            Fastpath CPU:

EWA0                        4
EWB0                        3
EWC0                        2
EWD0                        1
FGB0                        5
FGA0                        7
PEA0                        5
PKA0                        0
OpenVMS TCP/IP is currently running on CPU 4
OpenVMS Lock Manager is currently running on CPU 6
```

There are Conflicts for Cores

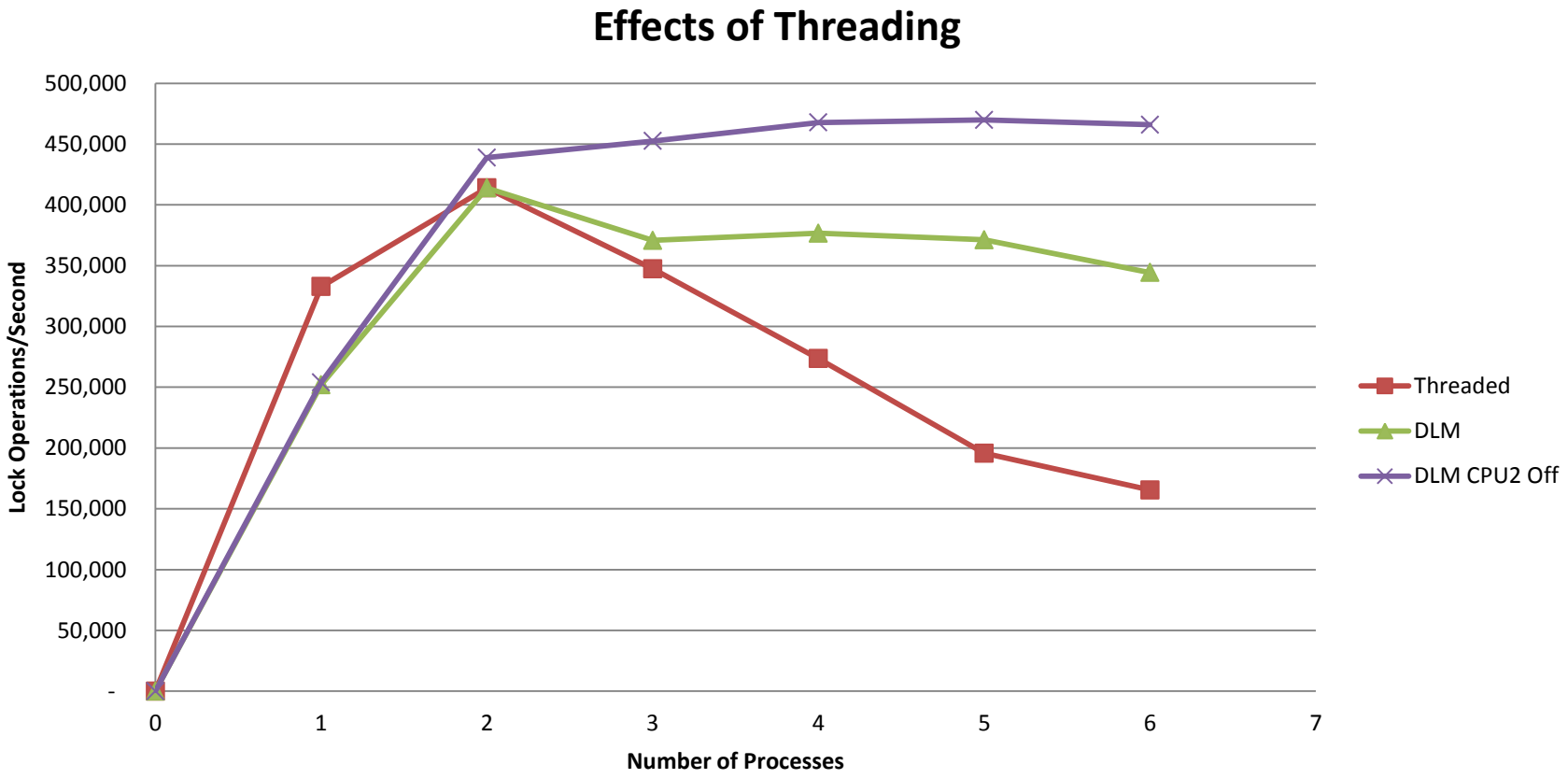
- The Lock Manager shares a core with a network device EWC0:

```
pandor > show cpu 6/full

System: PANDOR, HP BL870c  (1.59GHz/9.0MB)


CPU 6    State: RUN                CPUDB: 891CEB00     Handle: 000063D0

         Owner: 000004C8         Current: 000004C8     Partition 0
(PANDOR)
        ChgCnt:          2          State: Present, Reassignable

         Cothd:          2   <===

       Process: LCKMGR_SERVER        PID: 3E606EC1
```

Interference

- Network I/O therefore will interact with the operation of the Lock Manager
- Can be fixed by stopping CPU 2
  - No contention for Lock Manager core

Threaded Systems Can Perform Worse Without Management



Comparisons

- The maximum with DLM and CPU2 off was 499,842 lock operations per second
  - Maximum number of lock operations for the unthreaded machine was 490,000 which is essentially the same given measurement error
- The time to perform a lock operation was $3.9*10^{-6}$ seconds
  - This compares to $2.7*10^{-6}$ seconds for a lock operation with an unthreaded machine (single process)

What Does All of this Mean for Rdb Customers?

- Lock Manager can be a bottleneck
- Use a Resource Hash Table large enough the handle the number of resources you are locking
- Use Local locking wherever possible
  - Allow Rdb to use partitioned lock trees if the database must be open on multiple nodes
  - Performance is often better if database is open on only a single node
    - And is marked for being open-able on only one node
      - Triggers additional optimizations
    - If you do that Rdb record caches can eliminate a large number of page locks
  - Using Rdb remote can often improve throughput in a cluster
    - Transfers some work to network which itself has potential bottlenecks

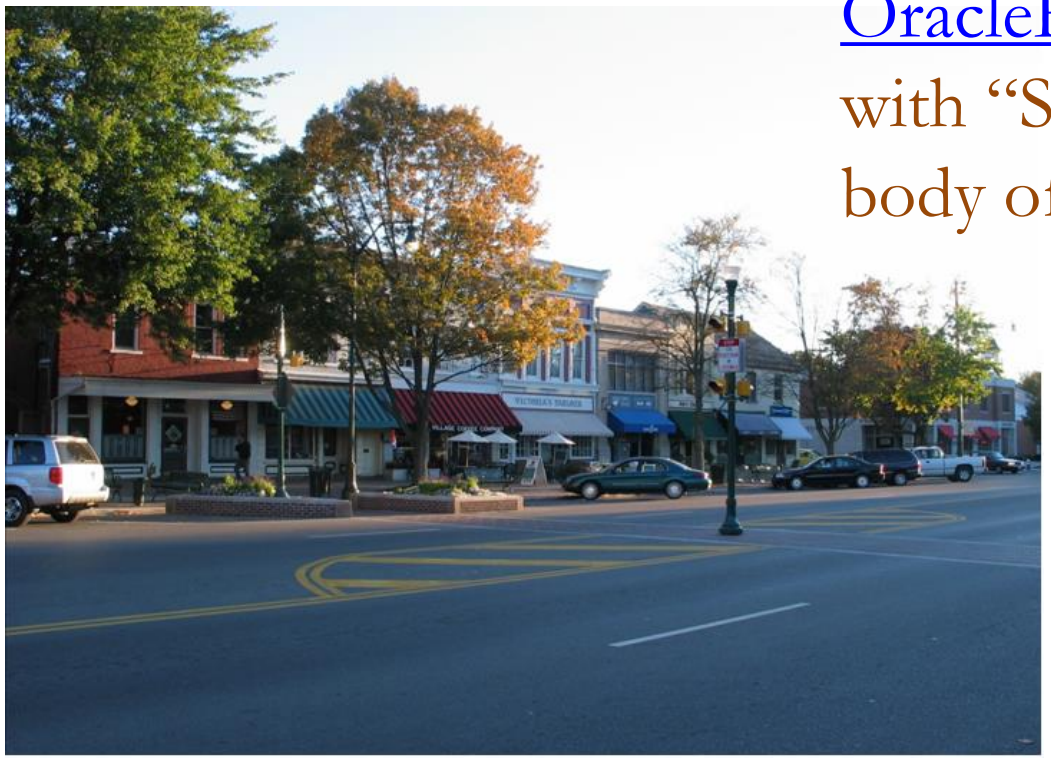
What Does this Mean for Rdb Customers?

- Put Dedicated Lock Manager on the fastest system and ensure that is the lock master for the database tree
- Tune applications and databases to minimize number of lock operations
- Threading your Integrity CPUs may not be an advantage
  - If you do, then you must pay careful attention as to how each co-thread CPU is used
- Since your applications are not doing 100% locking, your results WILL vary.

Join the Conversation

Join the worldwide Rdb community. Send mail to

OracleRdb-request@JCC.com with "SUBSCRIBE" in the body of the message.



Questions?

http://www.jcc.com/training.htm

At break,
please ask questions
and share ideas

Send your input and requests to Info@JCC.com