PERFORMANCE IMPLICATIONS OF NUMA WHAT YOU DON'T KNOW COULD HURT YOU!

CLAIRE CATES SAS INSTITUTE

AGENDA

- Terms
- What Testers need to know about NUMA
- What Developers need to know about NUMA
- What SysAdmins need to know about NUMA
- Tools I’ve used to find areas that are experiencing NUMA problems.

MEMORY TERMS

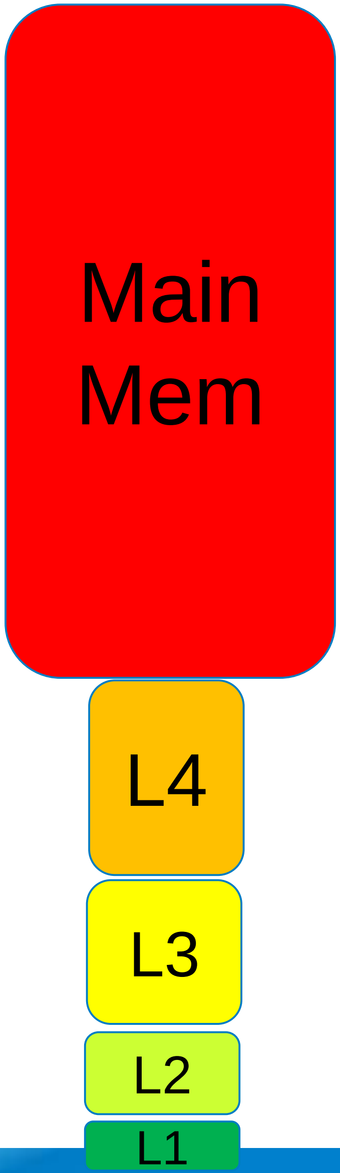
Latency

- The delay to access the memory.
- Usually measured in clock cycles to return the requested data.
- The slower the latency, the slower your program runs.

Bandwidth

- The pipeline carrying the memory from main memory to the processor.
- If you saturate the pipeline, performance will be impacted

THE MEMORY PARKING LOT



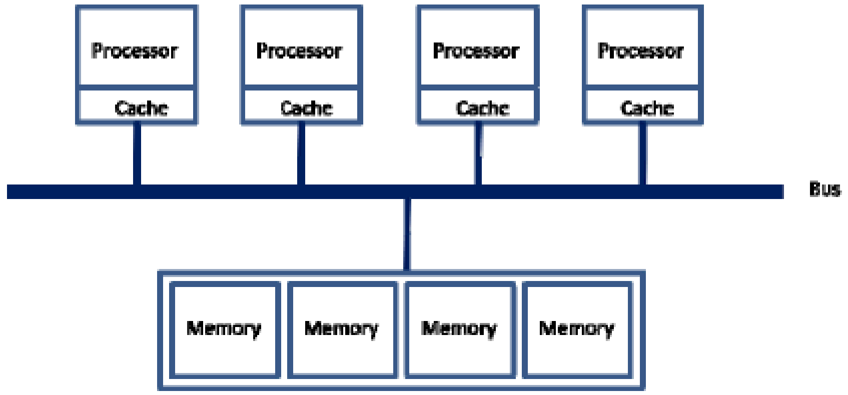
THE METER IS RUNNING BUT YOU'RE NOT MOVING

TYPES OF MEMORY ARCHITECTURES

UMA

UMA – Uniform Memory Access

- All memory is equidistance from all processors therefore memory accesses are uniform
- Timings are consistent between multiple runs.
- As more processor are being added the bandwidth becomes saturated impacting performance. Impedes scalability.

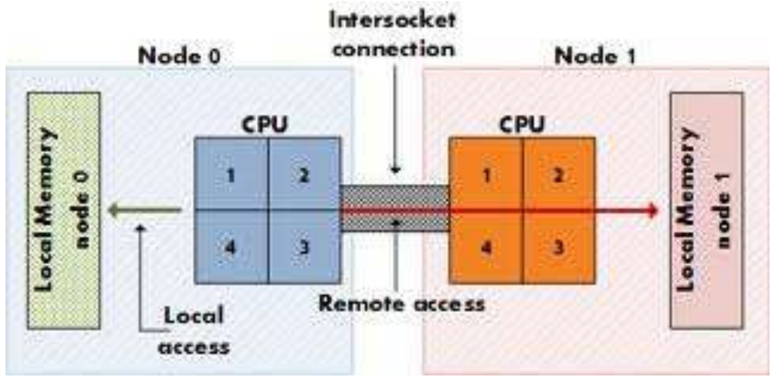


TYPES OF MEMORY ARCHITECTURES

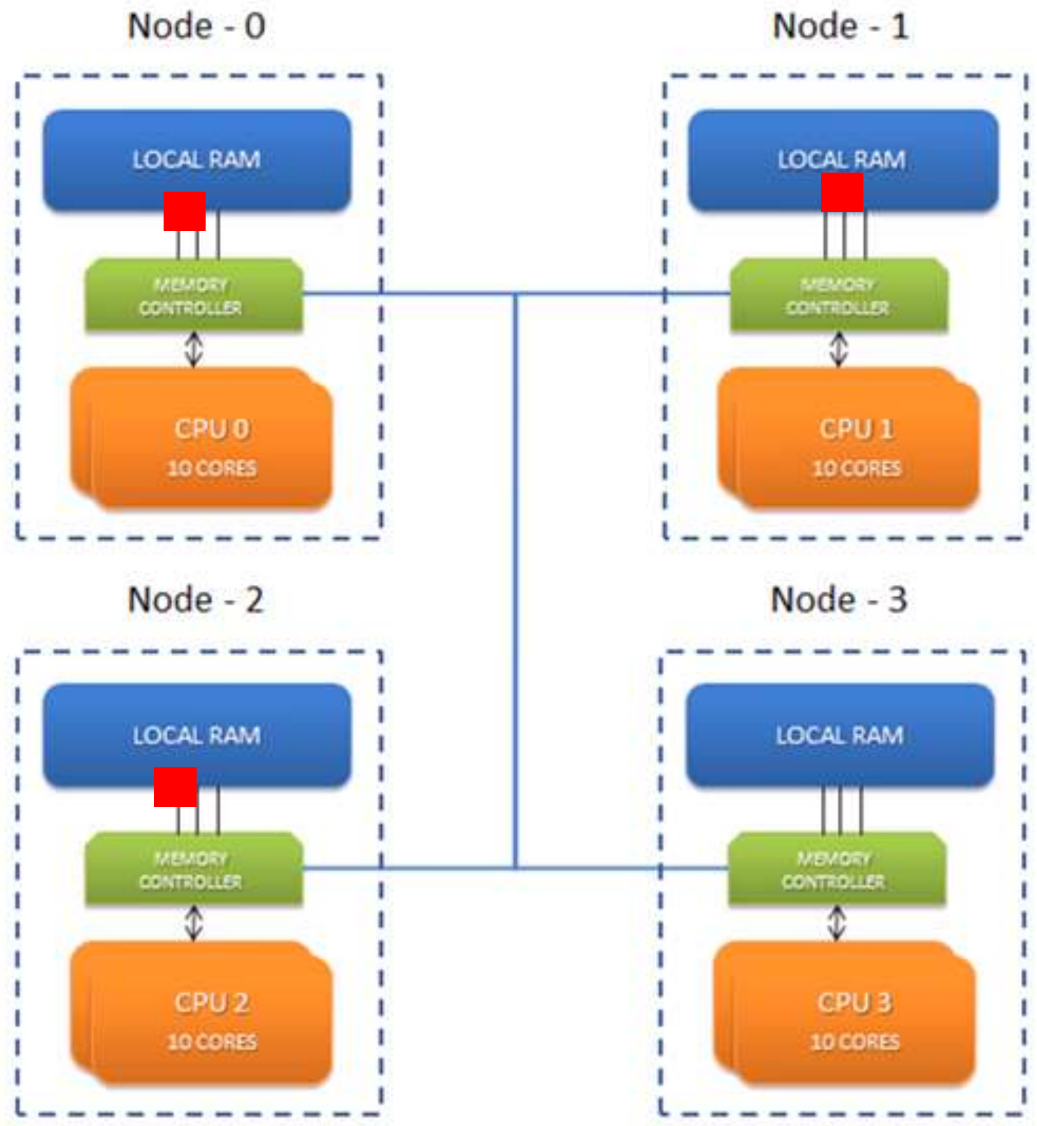
NUMA

NUMA – Non-Uniform Memory Access

- Memory access times very depending on where the memory is located. For best performance you need to co-locate the memory on the same chip as the processor.
- Hardware designers started using NUMA when more and more CPUS where being added to a chip.   The CPUS were running into severe bandwidth issues – "starved" – therefore NUMA improves scalability if used correctly
- Accessing memory on your own chip "socket" is much faster than on the other "sockets". Furthermore, the access time between different sockets can vary.



NUMA


Node - 0
LOCAL RAM
MEMORY CONTROLLER
CPU 0
10 CORES
Node - 1
LOCAL RAM
MEMORY CONTROLLER
CPU 1
10 CORES
Node - 2
LOCAL RAM
MEMORY CONTROLLER
CPU 2
10 CORES
Node - 3
LOCAL RAM
MEMORY CONTROLLER
CPU 3
10 CORES

NUMA SPECIFICS

- Local memory doesn’t have to go thru the interconnect (bandwidth)
- Local memory has less latency than remote
- The closer the memory is to the core the better the performance
- Numa is available on many systems for instance
  - Linux, Windows, Solaris (RedHat pre 6.3 had a bug)
- It may not be optimal for the application to let the OS just place the memory. Varying run times.
- Io cache’s can affect NUMA memory

NUMA SPECIFICS

- When a page is allocated it is not placed on a NUMA node until it is first touched. A hardware fault will be generated when a process touches or writes to an address (*page fault*) that has not been used yet. The physical page is allocated during page-fault handling.

- The default allocation policy is for the OS to place the page on the node where the CPU is running. Remember threads may migrate to another node.

- It is at the page allocation time that the allocation policy occurs

- Varying number of Sockets and varying number of CPUs per socket.

NUMA MEMORY POLICIES

- **Interleave** – place the memory on alternating nodes.
  - page 0 node 0,
  - next page node 1
  - then back to node 0.
- **LocalAlloc**. Memory is placed on the node where thread is running “where it’s first touched”
- **Membind** – allocation only on the nodes and fail if out of memory on the node
- **Preferred** – put memory on this node but if not available, other nodes can be used.

FIND OUT ABOUT YOUR MACHINE – RDCGRD001

- rdcgrd001> lscpu
- Architecture: x86_64
- CPU op-mode(s):   32-bit, 64-bit
- Byte Order: Little Endian
- CPU(s):   32
- On-line CPU(s) list: 0-31
- Thread(s) per core: 2
- Core(s) per socket: 8
- CPU socket(s):   2
- NUMA node(s):   2
- Vendor ID: GenuineIntel
- CPU family: 6
- Model: 45
- Stepping: 7
- CPU MHz: 2700.021
- BogoMIPS: 5399.21
- Virtualization: VT-x
- L1d cache: 32K
- L1i cache: 32K
- L2 cache: 256K
- L3 cache: 20480K
- NUMA node0 CPU(s):   0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30
- NUMA node1 CPU(s):   1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31

NUMADEMO

```
• ptnode604> numademo 4g memset
• 2 nodes available
• memory with no policy memset              Avg 6243.80 MB/s Max 6245.70 MB/s Min 6238.93 MB/s
• local memory memset                       Avg 6241.38 MB/s Max 6243.43 MB/s Min 6239.66 MB/s
• memory interleaved on all nodes memset    Avg 4153.52 MB/s Max 4154.21 MB/s Min 4152.77 MB/s
• memory on node 0 memset                   Avg 3125.44 MB/s Max 3194.80 MB/s Min 3110.37 MB/s
• memory on node 1 memset                   Avg 6258.41 MB/s Max 6260.40 MB/s Min 6256.22 MB/s
```

WHY DOES A TESTER NEED TO KNOW ABOUT NUMA

- If your application is not NUMA aware, then your timings can vary greatly.
- In some examples we've seen as much as a 40% difference in timings
- Recently in one product, the group had problems getting consistent times. It actually looked like a previous release (pre performance changes) was faster than the latest release. The old machines were pre-numa. The new machines were NUMA.
- Single "socket" machines don't show the NUMA effects.
- Most of our performance machines have at least 2 sockets.

WHY DOES A TESTER NEED TO KNOW ABOUT NUMA

- **To get more consistent times**
  - Run with Node interleaved memory – OS puts memory on all sockets – interleaves it.
  - Use virtual machines configured to match the characteristics of a single socket NUMA node.

WHAT DEVELOPERS NEED TO KNOW ABOUT NUMA

- Large applications that use lots of threads and lots of memory need to tune to use NUMA (code changes)
- There is an API available for the application to call
- Memory needs to be co-located to the cpu where the thread is running.

NUMA API

http://linux.die.net/man/3/numa_run_on_node

Some of the routines are

- **numa_max_node**() **–** how many nodes are there
- **numa_alloc_onnode**() – alloc memory on a particular node
- **numa_alloc_local**() – alloc memory on the current “local” node
- **numa_alloc_interleaved**() – only for large memory so that it places it across multiple nodes
- **numa_free**() – free the memory
- **numa_run_on_node**() – run thread and it’s children on this node
- **numa_node_of_cpu**() – what node am I running on currently

WHAT DEVELOPERS NEED TO KNOW ABOUT NUMA

Things to Consider

- If you allocate memory on a node and there is no more memory available, the OS can place the allocation on another node.
- You need to make sure subsystems your code calls which allocate memory are also allocating memory on the node where the thread is running.
- If a master thread is allocating the worker threads and the working memory for the worker threads, the master thread needs to allocate the threads and working memory on the same node.
- Alternatively, the master thread can allocate the threads and have the threads allocate the memory

WHAT DEVELOPERS NEED TO KNOW ABOUT NUMA

Performance Considerations

- Large memory areas shared across CPUs are best placed using interleaving so the objects are distributed over all available nodes.
- Avoid Hot memories - writeable memory, synchronization variables, shared memory,…
- If subset of threads share memory ie one group reads or starts processing data, that is then passed to another set of threads, make sure the threads that are working on the same data are pin'd to the same socket.
- Consider duplicating **read only** memory that is shared across threads – one copy per numa node.

LOCKING A THREAD TO A NUMA SOCKET?

- You can lock a thread to a CPU (affinity) or a NUMA Socket

- **Good**
  - If Memory is allocated on the node the memory is local,
  - The thread won't move so that the memory and cpu are always local

- **Bad**
  - If there are other processes running on the machine or your application has more threads than CPUs, a locked thread can't migrate to take advantage of available CPUs.
    - If it does migrate the thread can take a performance hit.

- How many different processes are running?
- How many threads?

WHAT SYSADMINS NEED TO KNOW ABOUT NUMA

- **Numactl** – Control NUMA policy for Processes or shared memory..
- **Numastat** - Show per-NUMA-node memory statistics for processes and the operating system
- **/proc/<pid>/numa_maps** – show where the memory is located for the pid
- **/proc/<pid>/maps** – shows beginning and ending location of memory blocks
- **/proc/meminfo** - general information on memory
- **/sys/devices/system/node/node<X>/meminfo** general information on the memory by node
- **Taskset** – retrieve or sets a processes' CPU affinity

NUMACTL

- The policy is set for the invocation command and inherited by all of its children. In addition it can set persistent policy for shared memory segments or files.

- Numactl –interleave=0-3 yourapp
- Numactl –cpunodebind=0 –membind=0 your app

```
numactl --hardware
available: 2 nodes (0-1)
node 0 cpus: 0 2 4 6 8 10 12 14
node 0 size: 32768 MB
node 0 free: 7099 MB
node 1 cpus: 1 3 5 7 9 11 13 15
node 1 size: 32755 MB
node 1 free: 7005 MB
node distances:
node   0   1
  0:  10  20
  1:  20  10
```

WHAT SYSADMINS NEED TO KNOW ABOUT NUMA

NUMACTL

Numactl - show **will show the policy**

```
ptnode401.ptest.sas.com> numactl -show
policy: default
preferred node: current
physcpubind: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
cpubind: 0 1
nodebind: 0 1
membind: 0 1
```

Lots of options, look at the man page.

NUMASTAT

Show per-NUMA-node memory statistics for processes and the operating system. This can give you an idea if the memory policy for your system is working

```
                           node0           node1
numa_hit               844535342       785774665
numa_miss              186725577         2219707
numa_foreign             2219707       186725577
interleave_hit           2487781         2487808
local_node             844366323       783409282
other_node             186894596         4585090
```

.

WHAT SYSADMINS NEED TO KNOW ABOUT NUMA

NUMASTAT

	node0	node1
numa_hit	+1	+1
numa_miss	+1	+1
numa_foreign	+1	+1
interleave_hit		
local_node		+1+1
other_node	+1+1	

Assuming running on Node 1

Red wanted on node 1 and got it on node 1

Blue wanted it on node 1 but got it on node 0

Black wanted it on node 0 and got it on node 0

Green wanted on node 0 but it was allocated on node 1

WHAT SYSADMINS NEED TO KNOW ABOUT NUMA

/PROC/<PID>/NUMA_MAPS

- Shows you which node items in your process are using to satisfy memory.

```
7f57406c7000 default anon=9766 dirty=9766 N0=8979 N1=787
7f5742ced000 default anon=4883 dirty=4883 N0=4883
7f5765ea0000 default file=/c/bb04na2a/vol/sasgen/dev/mva-v940m1/tkext/com/laxnd/tkeiplp.so mapped=34 N1=34
7f5766629000 default file=/c/bb04na2a/vol/sasgen/dev/mva-v940m1/tkext/com/laxnd/tkeiplp.so
7f5766829000 default file=/c/bb04na2a/vol/sasgen/dev/mva-v940m1/tkext/com/laxnd/tkeiplp.so anon=9 dirty=9 N1=9
```

/PROC/<PID>/MAPS

- Shows you which beginning and ending locations of the memory block and what it contains.

```
7fe520be0000-7fe520be1000 ---p 00000000 00:00 0
7fe520be1000-7fe521621000 rwxp 00000000 00:00 0
7fe521621000-7fe521622000 ---p 00000000 00:00 0
7fe521622000-7fe522062000 rwxp 00000000 00:00 0
7fe522062000-7fe522077000 r-xp 00000000 00:90 157085388          dev/mva-v9/tkext/com/laxnd/tkloglist.so
7fe522077000-7fe522277000 ---p 00015000 00:90 157085388          dev/mva-v9/tkext/com/laxnd/tkloglist.so
7fe522277000-7fe522278000 rwxp 00015000 00:90 157085388          dev/mva-v9/tkext/com/laxnd/tkloglist.so
7fe522278000-7fe52227d000 r-xp 00000000 00:90 188974557          dev/mva-v9/tkext/com/laxnd/t1a2en.so
7fe52227d000-7fe52247d000 ---p 00005000 00:90 188974557          dev/mva-v9/tkext/com/laxnd/t1a2en.so
7fe52247d000-7fe52248d000 rwxp 00005000 00:90 188974557          dev/mva-v9/tkext/com/laxnd/t1a2en.so
```

/SYS/DEVICES/SYSTEM/NODE/NODE0/MEMINFO

- General information about Memory on the particular Node

```
Node 0 MemTotal:       67062324 kB          Node 0 Mapped:           141780 kB
Node 0 MemFree:         1462268 kB          Node 0 AnonPages:        388524 kB
Node 0 MemUsed:        65600056 kB          Node 0 Shmem:             53976 kB
Node 0 Active:         28869108 kB          Node 0 KernelStack:       11904 kB
Node 0 Inactive:       32415372 kB          Node 0 PageTables:        28128 kB
Node 0 Active(anon):    5850828 kB          Node 0 NFS_Unstable:          0 kB
Node 0 Inactive(anon):    12708 kB          Node 0 Bounce:                0 kB
Node 0 Active(file):   23018280 kB          Node 0 WritebackTmp:          0 kB
Node 0 Inactive(file): 32402664 kB          Node 0 Slab:            2480748 kB
Node 0 Unevictable:           0 kB          Node 0 SReclaimable:    2431168 kB
Node 0 Mlocked:               0 kB          Node 0 SUnreclaim:        49580 kB
Node 0 Dirty:               152 kB          Node 0 HugePages_Total:     0
Node 0 Writeback:             0 kB          Node 0 HugePages_Free:      0
Node 0 FilePages:      55474916 kB          Node 0 HugePages_Surp:      0
```

CLEAR UP FREE SPACE

- Echo 3 | sudo tee /proc/sys/vm/drop_caches

```
numactl --hardware
available: 2 nodes (0-1)
node 0 cpus: 0 2 4 6 8 10 12 14
node 0 size: 65490 MB
node 0 free: 1427 MB
node 1 cpus: 1 3 5 7 9 11 13 15
node 1 size: 65536 MB
node 1 free: 18492 MB
node distances:
node   0   1
  0:  10  20
  1:  20  10
```

```
numactl --hardware
available: 2 nodes (0-1)
node 0 cpus: 0 2 4 6 8 10 12 14
node 0 size: 65490 MB
node 0 free: 57712 MB
node 1 cpus: 1 3 5 7 9 11 13 15
node 1 size: 65536 MB
node 1 free: 55610 MB
node distances:
node   0   1
  0:  10  20
  1:  20  10
```

TOOLS

- Amplifier – with general exploration will tell you some information about the performance. Several of the counters deal with the cache and remote socket access (DRAM).

- ThreadSpotter – It is solely looking at memory usage and will show you the areas in your program where the cache is not utilized thoroughly. This tool shows code areas that experience high latency or saturated bandwidth.

- I use both tools to get a better idea of where we are spending performance cycles and if NUMA is causing problems.

SUMMARY

- Testers, Developers and SysAdmins need to be aware of Numa

- Performance can be improved or degraded because of NUMA. Make sure you take advantage of NUMA to improve performance.   The defaults may hurt performance.

- There are lots of papers and information online

QUESTIONS

CLAIRE.CATES@SAS.COM