Experiences with Large Numascale Shared Memory Systems

Atle Vesterkjaer
Senior Software Engineer, Numascale

with contributions from Dr. Ole W. Saastad
Senior Engineer, University of Oslo

ISC’14
A 1728 core shared memory system is installed at the University of Oslo, compute center. Milestones, tuning, results and long term stability will be discussed.

- One Server - One Operating System - All Applications - Priced like Cluster
- Scales beyond Mainframe - No Virtualization Software required
NumaConnect Platform

Global Shared Memory Space – Single Operating System Image

- Global Shared Memory - Global Shared I/O - Global Shared CPUs - Global Shared GPUs - Global Shared disks

Copyright 2014. All rights reserved.
Numascale ccNuma architecture

Memory

L3 Cache

Mem. Ctrl.

HT Interface

L1&L2 Caches

CPU Cores

NumaChip

NumaCache (2, 4 or 8 GB)

To/From Other nodes in the same dimension

Remote Memory Access, Remote Cache Hit

Remote Memory Access, Remote Cache Miss

Local Memory Access, HT Probe for Shared Data

Local Memory Access, HT Probe for Shared Data

Copyright 2014. All rights reserved.
The current Linux kernels (3.x.y) all have **numa awareness** which provide mechanisms to schedule the many threads efficiently. The distribution comes with tools like **numactl** which help to launch the multi threaded job in an efficient way.

Compiler and threads used:

- Posix threads
- OpenMP
- OpenMPI
- Open64 compiler (maximum core count is 256)
- Intel Compiler
- Gnu Compiler

The **numactl** provide good control over scheduling of threads on the system by providing thread to core, socket and board binding with fine grain control with lists of threads and cores. In addition it can provide memory allocation policy settings, to specify how memory is allocated on the numa memory nodes.
Hardware: “\textit{n-dimensional cabling and pickup module is not trivial. The rest of the hardware is common standard which is easy to install and operate}.”

Performance: “\textit{Running large memory applications can be a challenge. The sheer magnitude of the problem implies long computational times. Anything else than tailor made benchmarks exhibit very long run times}.”

Software: “\textit{Installing a system with over 1600 cores is not commonplace for any Linux distribution. There have been numerous challenges with locks, semaphores etc. These problems have been largely overcome due to great effort by Numascale}.”

Hardware, software and performance achievements will be covered in the next slides.
January 2013: Dr. Ole Saastad on ease of programming shared memory systems

**Concepts:** “The concept of shared memory is very often perceived by the programmers as a uniform very large memory from zero to N. The concept of NUMA (None Uniform Memory Access) is very often not high up on the agenda of the programmer”

**Scaling:** “NUMA systems will happily accept OpenMP or POSIX threads without any NUMA awareness and run perfect, but with poor performance. To overcome these obstacles the programmer must take into account the NUMA architecture. Few OpenMP programs scale to any large core count.”

**Threading libraries, NUMA control and binding:** “A process is normally free to be run on any core within the system, but on NUMA system this is not a good idea. Hence the need to bind the process to a fixed core.”

**MPI:** The shared memory device that comes with OpenMPI is not the best suited for shared memory systems. Numascale has implemented a new shared memory device that is providing superior MPI performance on a Numascale system

Hardware, software and performance achievements will be covered in the next slides.
Lessons learnt: Hardware installation

ISC’14: The NumaManager is the simplest “plug and go” way to deploy a NumaConnect System = less errors, less mistakes, less support and easier than a cluster
The NumaManager automatically detects compatible servers.

A newly detected node will appear as down in the Servers tab.
NumaManager features

- Double click the **Servers** tab icon to power up the node.
The NumaManager will load the multi-server SMP system with correct configuration, including bootloader (using a tftpserver, dhcpserver and a numamanager).

The booting and monitoring environment are seamlessly encapsulated inside the device to minimize manual intervention.
Power control

Power control (shell):
[atlev@numa-prace ~]$ for i in {0..17}; do for j in {0..3}; do echo "ipmi-$j-$i";/site/sbin/itool ipmi-$j-$i power reset; done; done;

Power Control NumaManager:
NumaConnect Architecture Supported in Linux kernel
  - Interprocessor Interrupt (APIC extension HW)

Runs with standard kernel

3.14.X kernels offer great stability on large Numascale systems with recent uptime more than 4 months on the 1728 core UIO system.

Tuned kernel recommended
  - Especially for large systems >8 servers
  - “Custom Kernel” with recommended options

Patches
  - Queue-Based Spin Locks (Scalability)
  - Optimized Timing Framework for NumaConnect Fabric
A NumaConnect system can be programmed just as an “ordinary” computer!

The full memory range is available to all applications.

You can run “top” on a 1.5TB NumaConnect system.

```
top - 09:29:57 up 20 days, 20:07, 6 users, load average: 8.32, 8.31, 8.32 Tasks: 3068 total, 3 running, 3065 sleeping, 0 stopped, 0 zombie Cpu(s): 1.4%us, 0.1%sy, 0.8%ni, 98.5%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st Mem: 1583148160k total, 1185382348k used, 398065812k free, 4k buffers Swap: 3944624k total, 0k used, 3944624k free, 4035576k cached

PID USER   PR NI VIRT  RES   SHR S %CPU %MEM    TIME+ COMMAND
202867 root  20  0 1119g 1.1t 524 R 815 74.1  3758:34 x.mod2as
197024 root  20  0 13512 0360 1204 R 37  0.0  403:40.55 htop
206397 root  20  0 13512 3704 924 R 14  0.0   0:64:48 top
197023 root  20  0 13512 3704 924 S 13  0.0  177:20.24 top
1399 root   20  0  0  0  0 S  3  0.0  1:03.79 ksoftirqd/348
2187 root   20  0  0  0  0 S  3  0.0  0:50.98 ksoftirqd/545
10 root    20  0  0  0  0 S  2  0.0  290:26.05 rcu_sched
11190 root  20  0  0  0  0 S  1  0.0  68:02.65 kworker/40:1
11177 root  20  0  0  0  0 S  1  0.0  173:47.85 kworker/24:1
11241 root  20  0  0  0  0 S  1  0.0  86:23.99 kworker/44:1
11668 root  20  0  0  0  0 S  1  0.0  53:37.71 kworker/348:1
11688 root  20  0  0  0  0 S  1  0.0  59:31.14 kworker/336:1
12035 root  20  0  0  0  0 S  1  0.0  20:37.15 kworker/545:1
11197 root  20  0  0  0  0 S  0  0.0  96:09.80 kworker/36:1
11263 root  20  0  0  0  0 S  0  0.0  148:15.71 kworker/32:1
11269 root  20  0  0  0  0 S  0  0.0  166:31.05 kworker/28:1
11233 root  20  0  0  0  0 S  0  0.0  82:54.52 kworker/48:1
11626 root  20  0  0  0  0 S  0  0.0  32:13.59 kworker/388:1
11710 root  20  0  0  0  0 S  0  0.0  26:04.65 kworker/357:1
11714 root  20  0  0  0  0 S  0  0.0  31:09.39 kworker/354:1
47717 root  20  0  7764 576 484 S  0  0.0  22:06.24 tail
47736 root  20  0  7764 576 484 S  0  0.0  22:13.57 tail
```
ISC’14 Concepts, scaling, threading NUMA control and binding

- Keep the heap on the memory local to the running thread
  - `export LD_PRELOAD=libncalloc.so`
- Keep the Thread Local Storage on the memory local to the running thread
  - `export LD_PRELOAD=libptstack.so`
- Use optimized copy functions
  ```
  struct numachip_sge {
    uint64_t from;
    uint64_t to;
    uint32_t length;
  } __attribute__((aligned(16)));
  /**
   * numachip_sge_copy - Optimized SG Copy
   */
  ```
  ```
  struct numachip_sge sge;
  sge.from = (uint64_t) src;
  sge.to = (uint64_t) dest;
  sge.length = nbytes;
  numachip_sge_copy(&sge, 1);
  ```
- Use affinity settings
  ```
  - OMP_NUM_THREADS=64
  - GOMP_CPU_AFFINITY="0-255:4" bin/ep.C.x
  ```
- Reduce pagefaults due to frequent memory allocation
  ```
  - export MALLOC_TRIM_THRESHOLD_=67108864
  - MALLOC_MMAP_MAX_=67108864
  - MALLOC_TOP_PAD_=67108864
  ```
ISC’14 Lessons learnt: Software

Scaling up to all nodes in the system with help from:
- https://wiki.numascale.com/tips
ISC’14 Lessons learnt: Software

Scaling OpenMP programs to thousand cores on the Numascale architecture

Numascale
Numascale is a European SME specializing in interconnect for high performance and enterprise computing. The differentiator for Numascale’s interconnect is the shared memory and cache coherence mechanisms. These features allow programs to access any memory location and any memory mapped I/O device in a multiprocessor system with high degree of efficiency. It provides scalable systems with a unified programming model that stays the same from the small multi-core machines used in laptops and desktops to the largest imaginable single system image machines that may contain thousands of processors. The architecture is commonly classified as NUMA but the interconnect system can alternatively be used as low latency clustering interconnect. The technology comes as an add-on card to standard servers providing large shared memory at cluster price.

Optimizations for large NUMA systems
- Reduced synchronization overhead:
  - Local data copies where possible
  - Local buffers for synchronize writes
  - Multithreaded memory allocator
  - Thread local random number generator
- NUMA aware memory placement
- Usage of the Adaptive NUMA Scheduler to handle load imbalance and maintain most possible data locality

The Oslo System
The University of Oslo’s Center for Information Technology, USIT features this Numascale installation. The systems is a PRACE prototype and has been used for some of the work between Numascale and JarAHPC.

- 72 IBM x3755 M3 nodes
- 144 AMD 6174 CPUs
- 1728 Cores
- 4.6TB Memory

Memory Bandwidth
Memory bandwidth is essential for many scientific applications. Every processor has its own memory and adds to the total memory bandwidth. STREAM [1] is a standard benchmark to evaluate the memory bandwidth of a system.

TrajSearch
TrajSearch is a code to investigate turbulent fluxes which occur during combustion. It is a post-processing code for dissipation element analysis developed by Peters and Wang [2]. It decomposes a highly resolved three-dimensional turbulent flow field obtained by Direct Numerical Simulation (DNS) into non-constant, space-filling and non-overlapping geometrical elements called “dissipation elements”. Starting from every grid point in the direction of ascending and descending gradient of an underlying diffusion controlled scalar field, a local maximum, respectively, minimum point is found. A dissipation element is defined as a volume from which all trajectories reach the same minimum and maximum point. The dissipation element analysis provides a deeper understanding of turbulence and can be employed to reconstruct important statistical properties as has been shown by Comper and Gobbert [3].

Summary
- The investigated Numascale system combines 1728 cores and 4.5 TB of main memory in a single memory machine.
- The accumulated memory bandwidth rises with the number of nodes to over 2 TB/s.
- The application TrajSearch was optimized to run on large NUMA systems by reducing synchronization, optimizing the memory layout and using the Adaptive NUMA Scheduler for work scheduling.
- Overall the code delivers a speedup of 625 running with 1024 threads.

References
The overhead introduced by MPI is avoided when we are using OpenMP (or Pthreads) on a Shared Memory System.

- The NAS Parallel Benchmarks (NPB)
  - evaluate the performance of parallel supercomputers
  - derived from computational fluid dynamics (CFD) applications
  - LU is a simulated uses symmetric successive over-relaxation (SSOR) method to solve a seven-block-diagonal system resulting from finite-difference discretization of the Navier-Stokes equations in 3-D by splitting it into block Lower and Upper triangular systems.
Flow in porous media scaling

- Example from real data simulation on an 8-node Numascale system
- Numascale enables linear scaling over multiple nodes, while conventional implementations will not scale outside one node
- Large shared memory and number of cores are key
OPM Scaling Work

- 4 weeks to enable OPM scalability

- Initial state:
  - No scaling beyond 4 threads on a single server node

- A few changes after code analysis enabled scalability
  - `#pragma omp parallel for schedule (static)` to `#pragma omp parallel for schedule (dynamic,1)`
  - Created local copies of input file parser and added constructor to take a string input stream instead of filename
  - Reduced excessive use of `malloc/free` by setting environment variables `MALLOC_TRIM_THRESHOLD=-1, MALLOC_MMAP_MAX_=0, MALLOC_TOP_PAD_=536870912 (500MB)`
  - Timer class should use `clock_gettime(CLOCK_MONOTONIC, &now)` instead of `std::clock()` and `getrusage()` avoiding kernel spinlock calls
    - When building use `-DNO_TIMING` in the configuration or modify the code to use calls without spinlocks
Why Numascale

- If you can get scalable OpenMP and MPI performance, ease of programming and ease of administration at commodity cluster price points, why limit yourself to an MPI cluster?

- The NAS Parallel Benchmarks mimic the computation and data movement in CFD applications proving that CFD applications have great performance and scaling on NumaConnect Shared Memory Systems

- Real applications like TrajSearch show better speedup with Numascale technology than:
  - much more expensive hardware solutions
  - software emulations solutions that try to compete with hardware solutions like NumaConnect.

Approved for 1728 cores
Thank you!

Atle Vesterkjaer
Senior Software Engineer, Numascale

with contributions from Dr. Ole W. Saastad
Senior Engineer, University of Oslo

ISC’14
“Any application requiring a large memory footprint can benefit from a shared memory computing environment.”

William W. Thigpen,
Chief, Engineering Branch
NASA Advanced Supercomputing Division

“IBM sees the Numascale technology as a very viable solution for applications that require large scalable memory capacity.”

Dave Jursik,
VP, WW Deep Computing, IBM

“With proper Numa-awareness, applications with high bandwidth requirements will be able to utilize the combined bandwidth of all the memory controllers and still be able to share data with low latency access through coherent shared memory.”

Dr. Ole W. Saastad,
Senior Analyst and HPC Expert, UiO

“Time is an expensive resource, ... A lot of time is lost by having to move data in and out of the machine. We have memory hungry algorithms that can make better pictures of the geology faster given proper memory and processing capacity.”

Trond Jarl Suul,
Senior Manager, High Performance Computing, Statoil
Extra slides

Atle Vesterkjaer
Senior Software Engineer, Numascale

with contributions from Dr. Ole W. Saastad
Senior Engineer, University of Oslo

ISC’14
**Numactl, numastat -p, ipmi**

Monitor the console of one NumaServer:

```bash
[atlev@numa-prace ~]$ ipmitool -e! -I lanplus -H numa-0-ipmi -P ADMIN -U ADMIN sol activate
[SOL Session operational. Use !? for help]
```

CentOS release 6.5 (Final)
Kernel 3.14.4-numascale6+ on an x86_64

numa-0 login:

Monitor the memory access pattern in your Numascale Shared Memory System:

```bash
[atle@x3755 ~]$ numastat
```

<table>
<thead>
<tr>
<th></th>
<th>node0</th>
<th>node1</th>
<th>node2</th>
<th>node3</th>
</tr>
</thead>
<tbody>
<tr>
<td>numa_hit</td>
<td>116198855</td>
<td>22674225</td>
<td>1601598</td>
<td>326121</td>
</tr>
<tr>
<td>numa_miss</td>
<td>0</td>
<td>390637</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>numa_foreign</td>
<td>390637</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>interleave_hit</td>
<td>26017</td>
<td>26221</td>
<td>26298</td>
<td>26107</td>
</tr>
<tr>
<td>local_node</td>
<td>116197682</td>
<td>22647868</td>
<td>1576067</td>
<td>299901</td>
</tr>
<tr>
<td>other_node</td>
<td>1173</td>
<td>416994</td>
<td>25531</td>
<td>26220</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>node4</th>
<th>node5</th>
<th>node6</th>
<th>node7</th>
</tr>
</thead>
<tbody>
<tr>
<td>numa_hit</td>
<td>2484895</td>
<td>4957523</td>
<td>4425252</td>
<td>25191604</td>
</tr>
<tr>
<td>numa_miss</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>numa_foreign</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>interleave_hit</td>
<td>26026</td>
<td>26209</td>
<td>26289</td>
<td>26088</td>
</tr>
<tr>
<td>local_node</td>
<td>2458717</td>
<td>4931194</td>
<td>4398832</td>
<td>25165424</td>
</tr>
<tr>
<td>other_node</td>
<td>26178</td>
<td>26329</td>
<td>26420</td>
<td>26180</td>
</tr>
</tbody>
</table>

[atle@x3755 ~]$
**Numactl, numastat --p, ipmi**

```
[atle@numademo ~]$ numactl --hardware
available: 16 nodes (0-15)

node distances:

<table>
<thead>
<tr>
<th>node</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:</td>
<td>10</td>
<td>16</td>
<td>16</td>
<td>22</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td></td>
</tr>
<tr>
<td>1:</td>
<td>16</td>
<td>10</td>
<td>22</td>
<td>16</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td></td>
</tr>
<tr>
<td>2:</td>
<td>16</td>
<td>22</td>
<td>10</td>
<td>16</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>130</td>
</tr>
<tr>
<td>3:</td>
<td>22</td>
<td>16</td>
<td>16</td>
<td>10</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>130</td>
</tr>
<tr>
<td>4:</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>16</td>
<td>10</td>
<td>22</td>
<td>16</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>5:</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>16</td>
<td>22</td>
<td>10</td>
<td>16</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>6:</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>16</td>
<td>22</td>
<td>10</td>
<td>16</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>7:</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>22</td>
<td>16</td>
<td>16</td>
<td>10</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>8:</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>100</td>
<td>16</td>
<td>16</td>
<td>22</td>
<td>10</td>
<td>16</td>
<td>100</td>
</tr>
<tr>
<td>9:</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>16</td>
<td>10</td>
<td>22</td>
<td>16</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>10:</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>100</td>
<td>16</td>
<td>16</td>
<td>22</td>
<td>10</td>
<td>16</td>
<td>100</td>
</tr>
<tr>
<td>11:</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>22</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>10</td>
<td>16</td>
<td>100</td>
</tr>
<tr>
<td>12:</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>13:</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>14:</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>15:</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>130</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

[atle@numademo ~]$```
Monitor the memory access pattern for a running process.

```bash
[atle@x3755 ~]$ ps -elf | grep bwa
4 S root 25459 25458 99 80 0 - 1274150 futex_10:29 pts/3 00:36:30 /home/demouser04/04_Software/bwa-0.7.4/bwa mem -t 128 -M ../..../00_RAW_DATA/EC_K12../..../00_RAW_DATA/QBICECKH100001W4_GCCAAT_L001_R2_001.fastq
[root@x3755 atle]# numastat -p 25459

Per-node process memory usage (in MBs) for PID 25459 (bwa)

<table>
<thead>
<tr>
<th>Memory Type</th>
<th>Node 0</th>
<th>Node 1</th>
<th>Node 2</th>
<th>Node 3</th>
<th>Node 4</th>
<th>Node 5</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Huge</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Heap</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Stack</td>
<td>0.18</td>
<td>0.09</td>
<td>0.19</td>
<td>0.18</td>
<td>2.52</td>
<td>0.38</td>
<td>0.38</td>
</tr>
<tr>
<td>Private</td>
<td>63.66</td>
<td>38.99</td>
<td>43.42</td>
<td>31.44</td>
<td>82.00</td>
<td>66.58</td>
<td>31.62</td>
</tr>
<tr>
<td>Total</td>
<td>63.85</td>
<td>39.07</td>
<td>43.61</td>
<td>31.62</td>
<td>84.51</td>
<td>66.96</td>
<td>34.45</td>
</tr>
</tbody>
</table>

Node 6       Node 7       Total
<table>
<thead>
<tr>
<th>Memory Type</th>
<th>0.00</th>
<th>0.00</th>
<th>0.00</th>
<th>0.00</th>
<th>0.00</th>
<th>0.00</th>
<th>0.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>Huge</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Heap</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Stack</td>
<td>11.33</td>
<td>0.46</td>
<td>15.32</td>
<td>64.36</td>
<td>3824.57</td>
<td>3839.89</td>
<td>3839.89</td>
</tr>
<tr>
<td>Private</td>
<td>3434.12</td>
<td>64.36</td>
<td>3824.57</td>
<td>3445.45</td>
<td>64.82</td>
<td>3839.89</td>
<td></td>
</tr>
</tbody>
</table>
```

Copyright 2014. All rights reserved.
Affinity, taskset, rankfiles

 OpenMP
- GNU: OMP_NUM_THREADS=64 GOMP_CPU_AFFINITY="0-255:4" bin/ep.C.x
- GNU: OMP_NUM_THREADS=128 GOMP_CPU_AFFINITY="0-127" bin/ep.C.x
- GNU: OMP_NUM_THREADS=128 GOMP_CPU_AFFINITY="0-255:2" bin/ep.C.x
- PGI: OMP_THREAD_LIMIT=128 OMP_PROCS_BIND=true OMP_NUM_THREADS=128 MP_BIND=yes
  MP_BLIST=$(seq -s, 0 2 255) bin_pgfortran/ep.C.x

 MPI
- OMPI_PREFIX_ENV=/dev/shm mpiexec -n 121 -rf rank121_sp -mca btl self,nc bin/sp.D.121;
- OMPI_PREFIX_ENV=/dev/shm mpiexec -n 256 --bind-to-core -mca btl self,nc bin/sp.D.256;
- OMPI_PREFIX_ENV=/dev/shm mpiexec -n 128 -rf rank128_lu -mca btl self,nc bin/lu.D.128;
- OMPI_PREFIX_ENV=/dev/shm taskset -c 0-1727:2 mpiexec -n 864 --bind-to-core -mca btl self,nc bin/lu.E.864;

[ate@nc-demo NPB3.3-MPI]$ cat rank121_sp
rank 0=numademo slot=0:0
rank 1=numademo slot=0:2
rank 2=numademo slot=0:4
rank 3=numademo slot=0:6
rank 4=numademo slot=1:0
rank 5=numademo slot=1:2
rank 6=numademo slot=1:4
rank 7=numademo slot=1:6
......
rank 120=numademo slot=30:0
NumaConnect™ System Architecture

Multi-CPU Node

Memory
Multi-Core CPU

Memory
Multi-Core CPU

I/O Bridge

NumaCache

6 external links - flexible system configurations in multi-dimensional topologies

2-D Torus

3-D Torus
2-D Dataflow
### Boot Process Flow Chart

**System BIOS**
- **Power On**
  - HT routing and configuration
  - PCI enumeration
- Other POST activities
- PXE boot of BootLoader
- Master Node?
  - **y**
  - **n**
- Set "System Ready" CSR flag
- "Go Ahead"?
  - **n**
  - To slave bootstrap code
- **y**
  - NumaChip Expansion ROM (loaded from device, all nodes)

**NumaConnect Bootloader**
- Run Topology Setup
- Make setup code available
- Set "Go Ahead" for all slaves
- All slaves ready?
  - **y**
  - Map all resources, amend ACPI tables
  - Load Operating System
- **n**
  - Parse ACPI a. o. tables
  - General system initialization
  - Startup IPI received
  - Send startup IPIs to all APs
  - Run AP init code
  - Run user processes

- Tally local resources
- Report resources to master node
- Halt CPU to await OS SMP init

**Operating System (loaded from disk on Master Node)**
- Parse ACPI a. o. tables
- General system initialization
- Startup IPI received
- Send startup IPIs to all APs
- Run AP init code
- Run user processes
If you can get scalable OpenMP and MPI performance, ease of programming and ease of administration at commodity cluster price points, why limit yourself to an MPI cluster?
Automobil industry: Palabos cavity3d 800 Open Source Lattice-Boltzmann equation

- Palabos, [http://www.palabos.org/](http://www.palabos.org/), is an open-source CFD solver based on the lattice Boltzmann method

- Cavity3d: Flow in a diagonally lid-driven 3D cavity. In this 3D analog of the 2D cavity, the top-lid is driven with constant velocity in a direction parallel to one of the two diagonals. The benchmark is challenging because of the velocity discontinuities on corner nodes

- The NumaConnect Shared Memory test system used to conduct the tests has:
  - 1TB of memory
  - 256 cores
  - It utilizes 8 servers each equipped with:
    - 2 x AMD Opteron 2.5 GHz 6380 CPUs
    - 16 cores in each CPU
    - 128GB of memory

Mega site updates per second/Threads cavity3d 800 (Higher is better)

The NumaConnect Shared Memory test system used to conduct the tests has:
- 1TB of memory
- 256 cores
- It utilizes 8 servers each equipped with:
  - 2 x AMD Opteron 2,5 GHz 6380 CPUs
  - 16 cores in each CPU
  - 128GB of memory

The graph shows the Time per Iteration in seconds for different numbers of processes, with Lower being better. The data points are as follows:
- 32 processes: 141.77 seconds
- 64 processes: 75.15 seconds
- 128 processes: 43.56 seconds
### Apache Spark™ Benchmark

1B rows, 10 variables, Logistic Regression
4 node distributed cluster vs 4 node NumaQ

<table>
<thead>
<tr>
<th></th>
<th>4 nodes Cluster</th>
<th>NumaQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM</td>
<td>256GB RAM</td>
<td>1TB RAM</td>
</tr>
<tr>
<td>Cores</td>
<td>32 cores each</td>
<td>128 cores</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic Regression</td>
<td>1B rows</td>
<td>108 sec</td>
</tr>
<tr>
<td></td>
<td>10 variables</td>
<td>27 sec</td>
</tr>
</tbody>
</table>
Benchmarks

Scalability of boot_parallel.R code from R parallel package (R=20979,M=20979)