Silicon Graphics, Inc.

Scalable Single System Image

SGI Altix 3700, 512p

Architecture and Software Environment

Presented by:

Jean-Pierre Panziera
Principal Engineer
Altix 3700 SSSI - Architecture and Software Environment

• SMP and NUMA
• Directory based Shared Memory
• SGI Altix 3700, scaling Hardware to 512p
• Scaling Linux to 512p
• Scaling Applications to 512p
• next
Symmetric Multiprocessor (SMP)
Non Uniform Memory Access (NUMA)

Interconnection
Sharing data on a **Cache Coherent NUMA system** (ccNUMA)

**Node m**
- Reg
- L2
- L3

**Node n**
- Reg
- L2
- L3

**Node k**
- Mem

**Interconnection Fabric**
MESI protocol
Cache Coherence on the CPU side

- For each cacheline (128Bytes) coherency state is either:
  - **M**odified: Dirty Exclusive (DEX)
  - **E**xclusive: Clean Exclusive (CEX)
  - **S**hared: Shared (SHD)
  - **I**nvalid: Invalid (INV)
Directory based coherency
Cache Coherence on the Memory side

Directory information is in memory DIMMs
Directory takes only 3% on total memory

For each CacheLine available information:

- Line state (Unowned, Exclusive, Shared, Busy, ...) 2 bits
- Sharing Vector, list of nodes (256 max) 24 bits
- Priority
- Protection
- ECC
Directory based Cache Coherence
sequences of sharing messages

NodeN

NodeK

Memory

NodeM

BRL(shared)

SHD

SHRD

INV

EXCL

EXCL

INV

load

store

load

INTER

DNGRD

READ

ESPEC

INTER

XFER

BRL(shared)

INTER

INV

INV

EXCL

EXCL

INV

BUSY

ERPLY

N

M

K

N

M

K

N

M

K
SGI Altix 3700 Hardware
CPU-Memory brick

SC-Brick (3 U)

Node 0

Itanium®2

FSB

Itanium®2

Memory

Itanium®2

SHub

SHub

Memory

Itanium®2

Node 1

2x16 GB

Memory

6.4 GB/s

6.4 GB/s

XIO™ 2.4 GB/s

FSB

Numalink™ 3/4

3.2 / 6.4 GB/s

NL4

SGI Proprietary
SGI Altix 3700

32 Processors in a Rack

NUMAlink 3 Routers

2x1.6 = 3.2 GB/s

Bisection BW
12.8 GB/s
400 MB/s/p
Altix 3700
fat-tree topology for 512p/4TB system

Bisection BW
204.8 GB/s
400 MB/s/p

Image courtesy: NASA Ames
Linux for Altix 3000 / Linux 2.4 kernel

- **Differentiating Features and Functionality**
  - CPU sets / Memory placement
  - MPT & Array Services
  - Hierarchical storage management tools
  - Partitioning
  - XVM, XSCSI
  - PCP

- **Enabling Features and Functionality**
  - Latest bugfixes, other supported device drivers, etc
  - Comprehensive system accounting (CSA)
  - Job containers (PAGG)
  - XFS

- **Base OS and Common Open Source Apps**
  - Platform support, error handling, scaling, NUMA
  - O(1) Scheduler

- **Device Drivers, SGI XFS installer**

- **SGI Value-Added Enhancements**
- **SGI Open Source Enhancements**
- **Standard Linux® Distribution**
- **Boot/Driver CD**
Linux for Altix 3000

• NUMA support
  – discontiguous memory support
  – VM Support
  – text replication
  – process to processor binding, local memory allocation

• Partitioning support (shared memory clusters)
  e.g. configure 256p as 4x64p

• Scalability enhancements

• Fixes
O(1) Scheduler

• Scheduler improvements required
  – Standard Linux scheduler
    • poor cache usage; too many task migrations
    • heavy contention for runqueue_lock
  – O(1) scheduler
    • tasks “stick” to processors
    • no global runqueue_lock (multiple run queues)
    • 6x improvement on some benchmarks
    • Still needs work
      – migration livelock with lots of idle processors (fixed)
      – NUMA awareness (being worked by us & community)
      – Fairness in overcommitted workload situations
      – no gang scheduling concept
CPU and Memory Allocation

• Required to get good NUMA performance
  – Want memory allocated locally if possible
    • exploit local memory bandwidth
  – Want to bind processes to specific processors
    • to avoid cache damage caused by migration
    • to keep processes close to where data allocated
  – Want processors working on single HPC application to be “close to one another”
    • to minimize memory latency when cross-processor memory references are required
  – Don’t want to have to recompile the application
CPU and Memory Allocation

• `dplace` command: `dplace -c0-7 -s3 program`
  - assumes 9 processes will be created
    • 8 will be bound to CPUs 0-7
    • 3rd process created will be “skipped”
      not bound, e.g. for pthreads shepherd processes
  - Storage will be allocated locally
    “first-touch” allocation rule

• `cpusets` provides finer grain control, e.g. :
  - “boot” `cpuset` for interactive jobs and kernel data
  - “compute” `cpuset(s)` to isolate applications
## Stream Triad Benchmark Results (GB/s)

<table>
<thead>
<tr>
<th>System</th>
<th>Performance (GB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGI Altix 3000 (512P)</td>
<td>1007.83</td>
</tr>
<tr>
<td>NEC SX-7 (32P)</td>
<td>872.3</td>
</tr>
<tr>
<td>NEC SX-5 (16P)</td>
<td>583.1</td>
</tr>
<tr>
<td>SGI Altix 3000 (256P)</td>
<td>488.3</td>
</tr>
<tr>
<td>NEC SX-4 (32P)</td>
<td>437</td>
</tr>
<tr>
<td>HP AlphaServer GS1280 (64P)</td>
<td>377.7</td>
</tr>
<tr>
<td>Cray T3E (32P)</td>
<td>359.3</td>
</tr>
<tr>
<td>SGI Altix 3000 (128P)</td>
<td>255</td>
</tr>
<tr>
<td>NEC SX-6 (8P)</td>
<td>213</td>
</tr>
<tr>
<td>SGI Altix 3000 (64P)</td>
<td>127.3</td>
</tr>
<tr>
<td>Cray C90 (16P)</td>
<td>103.8</td>
</tr>
<tr>
<td>SGI Origin 3800 (256P)</td>
<td>99.7</td>
</tr>
<tr>
<td>HP Integrity SuperDome (64P)</td>
<td>84.2</td>
</tr>
<tr>
<td>SGI Altix 3000 (32P)</td>
<td>63.7</td>
</tr>
<tr>
<td>IBM eServer p690+ (32P)</td>
<td>58.9</td>
</tr>
<tr>
<td>Sun Fire15K (72P)</td>
<td>50.7</td>
</tr>
<tr>
<td>SGI Origin 2000 (256P)</td>
<td>49.3</td>
</tr>
<tr>
<td>Cray SV1 (32P)</td>
<td>47.8</td>
</tr>
<tr>
<td>HP AlphaServer ES80 (8P)</td>
<td>44.5</td>
</tr>
<tr>
<td>IBM eServer p670+ (16P)</td>
<td>36.8</td>
</tr>
<tr>
<td>IBM eServer p690 Turbo (32P)</td>
<td>32.2</td>
</tr>
<tr>
<td>SGI Altix 3000 (16P)</td>
<td>31.9</td>
</tr>
<tr>
<td>Cray Y-MP (8P)</td>
<td>26.8</td>
</tr>
<tr>
<td>HP SuperDome 750 (64P)</td>
<td>26.5</td>
</tr>
<tr>
<td>IBM eServer p690 HPC (16P)</td>
<td>25.1</td>
</tr>
</tbody>
</table>

### Measures memory bandwidth performance
- Simple loops
- Embarrassingly parallel
- Easy for compiler to generate scalable code
- No fancy flags: -O3

**STREAM Triad:**

```cpp
!$OMP PARALLEL DO
DO j = 1,n
   a(j) = b(j) + s*c(j)
CONTINUE
```
MPI across partitions: Latency vs. Distance

MPI Point-to-Point Latency (from CPU 0)
2x64p 1.5 GHz Altix 3700 Supercluster
Using MPT 1.10 on ProPack 2.4

8 byte transfer size

Time (usec)

Destination CPU

Host A

Host B
IFS (ECMWF) a scalable MPI application

- Total Time
- Communication Time
- Parallel Speedup
- ideal speedup

efficiency 78%
Scaling OpenMP Cart3D (Aerodynamics) on SGI Altix at NASA Ames

CPUs Used

CPUs Achieved

88% efficiency

Source: NASA Ames Research Center
Efficient Shared Memory Parallelism SMP
Pthreads / OpenMP

- Highest possible level parallelism (forget incremental)
- Limit “global” commons/arrays to necessary
- Make “sharing” explicit
- Use “[thread]private” local variables/arrays
- Replicate global data to local space
- Remove unnecessary synchronization
- Remove big global locks
  individual locks for critical code or data
- SMP enables Dynamic Load Balancing
SGI® Altix® 3000  future evolutions

- Faster Itanium®2 processors
- NUMAl ink™4 routers (2xNL3 BW)
- Higher processor density
- Reduced Remote Latency
- Faster/Denser Memory DIMMs
- Faster FSB (front side bus)
- Larger Cache Coherence domains
- Larger Shared Memory systems
Next generation: UV Architecture Vision

UV Petascale GAM

- Globally Addressable Memory
- Low Latency
- High Bandwidth
- O(10K) Ports
Altix 3700 SSSI - Architecture and Software environment

http://www.sgi.com/servers/altix/

Questions ?