SGI Altix 3000

- Introduced January 2003
  - Red Hat- or Suse-Linux compatible Operating System:
  - 512 CPU SSI Linux released
  - Intel Itanium2 processors (Madison) in all variants
  - Over 55000 processors sold, systems from 2 PEs to >512 PEs
  - Huge shared memory
    - Several orders for < 100 PEs with >2 TB shared memory.
  - Two OS Variants:
    - SGI enhanced Red Hat AS 2.1 based Linux OS
    - Standard SUSE SLES 9
  - Most traditional SGI value-adds available:
    - CXFS Client/Server, DMF, MPT,SCSL
    - Multipipe GFX available
SGI ccNuma Balanced System Architecture
SGI Altix3000

• SGI Altix3000 computer system is characterized by:

  – (Scalable) cache coherent shared memory (SMP)
  – Intel Itanium-2 processors
  – Standard Linux operating system
Parallel Architectures

Shared Memory (S.M.)

- Easy to Program
- Difficult to Scale
- ~ 32p

NUMA

- Easy to Program
- Scales Well
- ~ 1024p

Cluster

- Difficult to Program
- Highly Scalable
- ~ 4096p

Distributed Memory (D.M.)
SGI Scalable ccNUMA Architecture

- 16 GB Shared Physical Memory
- 8 GB Physical Memory x 2
- Interconnect
ccNuma: Distributed Shared Memory

• ccNuma:
  – Memory is physically distributed but logically shared
  – Memory is kept coherent automatically by hardware
  – Coherent memory: memory is always valid (caches hold copies)
  – Granularity is L3 cacheline (128 B)

• Directory memory:
  – For each cacheline access information is stored:
  – Who has valid copies
  – Which processor has write access
  – Hardware revokes access rights automatically

• In contrast snoop bus protocols do not scale well
  – Access requests are broadcasted

• Directory information is stored in main memory
  – Directory entry is 4 byte wide for each 128 byte cache line
ccNuma: Distributed Shared Memory

- Schematic view onto a full directory based coherence scheme


- http://www.cse.ucsd.edu/classes/fa00/cse240/lectures/Lecture18.html
SGI Altix 3700BX2
SGI® Altix™ Product Family

SGI® Altix™ 350
- Mid-range server
- Scales up to 32P
- Modular "expand on demand" architecture

Altix 1350

SGI® Altix™ 3700
- 32 processors per rack
- NUMALink 3 (3.2GB/sec)
- Madison

SGI® Altix™ 3700 Bx2
- 64 processors per rack
- NUMALink 4 (6.4 GB/sec)
- Madison 9M

SGI® Altix™ 4700
- 128++ processors per rack
- NUMALink ++
- Blades
- Montecito

System Scalability
SGI® Altix™ 3000BX2 CPU-Module

Node 0

Memory

Itanium®2

FSB

Itanium 2

NUMAlink™
6.4GB/sec

SHUB

6.4GB/sec

Memory
10.2GB/sec

Two network adapters

Two Itanium 2 processors

large processor caches

up to 24 GB memory

6.4 GB/s FrontsideBus

XIO™
2.4GB/sec

24GB max

Memory

Node 0
SGI® Altix™ 3000BX2 Memory

Each CPU module:

- 4 banks of up to 3 DDR-SDRAM dimms

- Dimms are 512 MB, 1GB or 2GB in size
  
  • PC2100  = 133MHz (DDR226)  Altix BW = 8.5 GB/s  - 7.5 ns
  • PC2700  = 166MHz (DDR333)  Altix BW = 10.2 GB/s  - 6.0 ns
  • PC3200  = 200MHz (DDR400)  Altix BW = 12.8 GB/s  - 5.0 ns
SGI Altix™ 3700 Bx2 Platform Introduction:
CR-Brick - Components

CPU Module

CR-Brick

Node 0

Proces

sor

(Intel
Madison 9M

6.4GB/s

Asic

(Shub1.2)

Nl4
Network

6.4GB/s

Full Duplex

Nd
Board

P

P

Node Board

P

P

Node Board

P

P

Node Board

I/O

Nl4

I/O

Nl4

I/O

Router

Router

Router

Router

I/O
Numaflex-4 Router:

- Microarchitecture elements of Cray T3E
  - Enhanced hardware support synchronization primitives
- 8 bidirectional ports
- 3.2 GB/s per direction per port
- Low latency about 50 nsec per router
- Dual plane configuration:
  - 2 x 6.4GB/sec total bandwidth between C-bricks
SGI Altix 3700 BX2 CR-Brick

Bottom-side IP57 node boards (2)

Rear router assembly (includes brick chassis)
SGI Altix 3700 BX2 CR-Brick
SGI Altix BX2 Configurations
Configurations

8 CPU System

16 CPU System
IX-Brick

• IX-brick
  Function
  – provides basic I/O functionality, houses the system disk and one option drive maximum
  – DVD-ROM, 2 serial ports, and 10/100/1000Bt
  – 10 PCIX expansion slots on 5 buses + 1 PCI 66MHz slot
  – optional 2 serial port expansion card
  – 1 per partition required
  Connectivity to a C-brick
  – 1 or 2 XIO™ + input ports with up to 4.8GB/sec aggregate B/W
PX-Brick

Function
- Provides PCIX expansion without basic I/O overhead
- 12 PCIX expansion slots on 6 buses
Connectivity to a C-brick
- 1 or 2 XIO + input ports with up to 4.8GB/sec aggregate B/W
SGI Altix™ 3700 BX2 Platform Introduction: I/O System Topology of CR-Brick to I/O Bricks

SGI Altix 3700 and Bx2 Topology of CR-Bricks with IX, PX-Bricks

SGI Altix 3700 Bx2 Topology of CR-Bricks with PA-Brick (Peer I/O)
SGI Altix™ 3700 Bx2 Platform Introduction: Benefits of Peer I/O

• Available with the PA-Brick on SGI Altix 3700 Bx2 Platform
• Direct connection of I/O into NUMAlink fabric:
  – NL4 at 6.4GB/s vs XIO at 2.4GB/s
  Total flexibility in ratio of compute to I/O capability
• Allows I/O channel performance to scale concurrently with NUMAlink improvements
Excursion on PCI

• Peripheral Component Interconnect
  – Invented by Intel
  – Started as 32-bit bus
  – Bus is buffered and works asynchronously
  – Supports Plug and Play configuration (PnP)

• PCIX, extension to width of 64 bits, up to 133 MHz

• Some performance data

<table>
<thead>
<tr>
<th>PCI</th>
<th>PCI-X</th>
</tr>
</thead>
<tbody>
<tr>
<td>33 MHz</td>
<td>66 MHz</td>
</tr>
<tr>
<td>132 MB/s</td>
<td>256 MB/s</td>
</tr>
<tr>
<td>66 MHz</td>
<td>100 MHz</td>
</tr>
<tr>
<td>512 MB/s</td>
<td>800 MB/s</td>
</tr>
<tr>
<td>133 MHz</td>
<td>1000 MB/s</td>
</tr>
</tbody>
</table>

• [http://arstechnica.com/articles/paedia/hardware/pcie.ars/1](http://arstechnica.com/articles/paedia/hardware/pcie.ars/1)
64-CPU Configuration
128-CPU Configuration
256 CPUs

Tornado (Shub 1.2) 200-256 Processor Dual-Plane Fat Tree

CR-Brick Configuration
- 4 IP57 Nodes
- 4 Shub 1.2s
- 8 Processor Sockets
- 8 Processors (Madison)
- 2 Routers

06-18-04
384 CPUs
Altix 3700 BX2 at TU Dresden
Altix 3700 BX2 at TU Dresden

- Two partitions
  - venus, 128 CPUs
  - merkur, 64 CPUs
Intel® Itanium® 2 - Why it is important?

- **High Bandwidth**
- **Many functional units**
- **Large on-chip caches**
- **Large physical address space**

### System Bus
- 128 bits wide
- 200 MHz/400 MT/sec
- 6.4GB/sec

### Width
- 2 bundles per clock
- 6 integer units
- 2 loads and 2 stores per clock
- 11 issue ports
- 4 FPMultiply Adds per Clock

### Caches
- L1: 2X16KB — 1 clock latency
- L2: 256K — 5 clock latency
- L3: 3-9MB — 12 clk
  - 32GB/sec bandwidth

### Addressing
- 50-bit physical addressing
- 64-bit virtual addressing
- Maximum page size of 4GB
Explicitly Parallel Instruction Computing (EPIC)

• EPIC
  – New instruction set (with IA-32™ compatibility)
  – 3 predicated instructions into 1 bundle (128bit)
  – 2 bundles per cycle
  – 128 general (integer) registers; up to 96 rotating
  – 128 floating-point registers; up to 96 rotating
  – 64 1-bit predicate registers; up to 48 rotating
  – 8 branch registers
  – 128 application registers (e.g., loop or epilog counters for pipelining)
  – Performance Monitor Unit (PMU) (> 100 Performance Counters)
  – Advanced Load Address Table (ALAT)
  – 6 integer units
  – 2 loads and 2 stores per clock cycle, speculative loads
  – 11 issue ports
  – Special instructions (multimedia, popcnt)
IA-64™ Instruction Bundles

1 instruction coded on 41 bits
3 instructions grouped into 1 bundle (128 bits)

Bundle type is specified through 5-bit template:

\{
  .mfi
  // template (mem-fp-int)
  (p16) ld fd f39=[r2],16 // load fp, post-increment
  (p19) fnma.d.s0 f49=f42,f6,f45 // multiply Add
  (p16) adds r32=16,r33 ; // integer add immediate
\}

\{
  .mib
  // template (mem-fp-br)
  (p16) ld fd f42=[r33] // load fp, post-increment
  (p16) adds r40=8,r33
  br.ctop.dptk.few .BB13_mp_ortho2_ ; ; ; // counted loop branch
Branch Optimization

Predication allows to remove (small) branches:

```c
if (i == j) {
    cmp.eq p1,p2=r32,r33
    (p1) add r1 = r1, r3
    (p1) fpma.d f31 = f3, f4, f2
    k += l;
    x = y + a * b;
} else {
    (p2) sub r1 = 3, r4
    (p2) ldfd f31=[r34], 8
    k = m - 3;
    y = *p_fp ++ ;
}
```

<table>
<thead>
<tr>
<th>cycle 0</th>
<th>cycle 1</th>
<th>cycle 1</th>
<th>cycle 1</th>
<th>cycle 1</th>
<th>cycle 1</th>
<th>cycle 1</th>
</tr>
</thead>
</table>

IA-64™
IA-64™ HW for Loop Optimization

Counted loops are optimized with HW support:

– Loop counter
– Epilog counter
– Predication registers for each instruction
– Rotation of registers
IA-64™ HW for Loop Optimization

\{ .mfi
(P16)
(P24)
(P19)
\}

\{ .mib
(P30)
(P20)
\}

br.ctop ;;


Itanium™2 - Execution Units

- 6 ALU ALU0-5
- 2 Integer I0,I1
- 1 ISHIFT
- 4 Port Data Cache Unit (2ld[fp]+2st or 4ldf)
- 6 Multimedia PALU0-5
- 2 Parallel shift PSMU0,1
- 1 Parallel Multiply PMUL
- 1 POPCNT
- 2 FP multiply-add FMAC
- 2 FP other operations FMISC
- 3 Branch
Itanium™2 - Instructions Latency

- Integer Instructions 1 cycle
- Floating Point Instructions 4 cycles
- MultiMedia 2 cycles
- FP Multiply-Add/sub  \textit{fma}/fnma/fms 4 cycles
- FP Multiply or Add  \textit{(fma} x*y+0 \textit{or} x*1+y) 4 cycles
- no FP Div, use approx[256] \textit{frcpa} 4 cycles
- no FP RSQRT, use approx[256] \textit{frsqua} 4 cycles
- no integer mult, use \textit{setf}/xma/getf 6/4/5 cycles
- no integer Mod, Div use \textit{setf}/frcpa/…/getf 6/4/5 cycles
Itanium™2 - FP Macros Latency

x/y, 1/sqrt(x), sqrt(x) do not translate into HW instructions. Instead the compiler combines fma/frcpa/frsqua (Newton iterations). Similarly integer *, /, %(modulo) are expanded through macros. Latency will vary depending with compiler efficiency:

<table>
<thead>
<tr>
<th></th>
<th>Single</th>
<th>Double</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP cycles</td>
<td></td>
<td></td>
</tr>
<tr>
<td>y = a+y</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>y = ay</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>y = a+b*y</td>
<td>28</td>
<td>32</td>
</tr>
<tr>
<td>y = b+a/y</td>
<td>36</td>
<td>37</td>
</tr>
<tr>
<td>y = a/sqrt(y)</td>
<td>43</td>
<td>55</td>
</tr>
<tr>
<td>y = sqrt(y)</td>
<td>36</td>
<td>37</td>
</tr>
<tr>
<td>y = y/sqrt(y)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

|                      |        |        |
| Int cycles           |        |        |
| i = i + c            | 1      | 1      |
| i = a*i              | 15     | 15     |
| i = a+b*i            | 16     | 16     |
| i = b + a/i          | 37     | 37     |
| i = b + a % i        | 42     | 42     |

|                      |        |        |
|                      | Single | Double |
| FP cycles            | 4      | 4      |
| y = a+y              | 4      | 4      |
| y = ay               | 4      | 4      |
| y = a+b*y            | 28     | 32     |
| y = b+a/y            | 36     | 37     |
| y = a/sqrt(y)        | 43     | 55     |
| y = sqrt(y)          | 36     | 37     |
| y = y/sqrt(y)        |        |        |

|                      |        |        |
| Int cycles           |        |        |
| i = i + c            | 1      | 1      |
| i = a*i              | 15     | 16     |
| i = a+b*i            | 16     | 56     |
| i = b + a/i          | 37     | 42     |
| i = b + a % i        | 42     | 42     |
Itanium™2 Data Flow (1.5 GHz)

- **128 Integer Registers**: 1kB
- **128 FP Registers**: 1kB
- **L1D cache**: 16kB
- **L2U cache**: 256kB
- **L3U cache**: 3-6 MB
- **Data Flow Rates**:
  - 41.6GB/s
  - 20.8GB/s
  - 16Rd / 6Wr
  - 2 cycles, 5 cycles, 12 cycles, 5+1 cycles, 12+1 cycles
  - 6.4 GB/s

---

**November 22, 2005**

Page 43
Itanium™2 L1/L2 Data Cache

L1D is 16kByte, 64Byte/line, 4way, WriteThrough, GRegisters only:
- 1 cycle latency (2 for load, pointer chasing), no FP cached in L1D
- Store uses 8x8 bytes array. Updates L1D only if hit.
- 8 (unique) outstanding misses

L2U is 256kByte, 128Byte/line, 8way, WriteBack, NotRecentlyUsed
- 5,7,9.../6,8,10... latency for Int/FP
- 16 banks - 16bytes/bank (??? 256Byte stride/alignment ???)
- 16 (unique) outstanding misses
- L2 is not inclusive of L1D and L1I
Itanium™2 L3U Cache/Memory

L3U: 1.5/3MByte, 128Byte/line, 6/12way, WriteBack, LeastRecentlyUsed
- 12,16…/13,17… latency for Int/FP
- 16 (unique) read misses
- 6 write

Local/remote memory is accessed through SHub/NUMAflex:

Local latency 132 ns
Same brick / other node 180 ns
NL4 router ~50 ns
1 Meter cable ~10 ns