

Columbia Configuration Details

Category: Columbia

Columbia Phase Out:

As of Feb. 27, 2013, the Columbia21, Columbia23, and Columbia24 nodes have been taken offline as part of the Columbia phase out process. Columbia22 is still available. If your script requires a specific node, please make the appropriate changes in order to ensure the success of your job.

Current Columbia compute nodes, Columbia21-24, are SGI Altix 4700 systems. Detailed information about the processor and memory subsystems of these compute nodes are provided in this article.

Processor and Memory Subsystems Statistics

Below are configuration statistics for the processor and memory subsystems for Columbia21-24:

Columbia Processor and Memory Subsystems Statistics			
Hostname	Columbia21	Columbia22	Columbia23-24
Function	compute	compute	compute
Architecture	Altix 4700 (bandwidth configuration)	Altix 4700 (density configuration)	Altix 4700 (density configuration)
Dual-Core Processor			
Processor	Itanium2 9040 (Montecito)	Itanium2 9040 (Montecito)	Itanium2 9150M (Montvale)
Core-Clock	1.6 GHz	1.6 GHz	1.67 GHz
# of Cores/Node	2	4	4
Nodes/Blade	1	1	1
Total # of Blades	256	512	256
Total # of Cores	512	2048	1024
Memory			
Local Memory/Node (2 Cores for C21 and 4 Cores for C22, C23-24)	~3.8 GB	~7.6 GB	~7.6 GB
Total Memory	~ 1000 GB	~ 4000 GB	~ 2000 GB

L1 Cache Size/Core	32KB (split into instruction and data cache)	32KB (split into instruction and data cache)	32KB (split into instruction and data cache)
L1 Cache Associativity	4-way	4-way	4-way
L1 Cache Line Size	64 bytes	64 bytes	64 bytes
L2 Cache Size/Core	1MB: instructions 256KB: data	1MB: instructions 256KB: data	1MB: instructions 256KB: data
L2 Cache Associativity	8-way	8-way	8-way
L2 Cache Line Size	128 bytes	128 bytes	128 bytes
L3 Cache Size/Core	9MB	9MB	9MB
L3 Cache Associativity	9-way	9-way	9-way
Default Page Size	16 KB	64 KB	16 KB

Itanium-64 Processors Facts

The Itanium chip is based on the IA-64 (Intel Architecture, 64 bit) architecture that implements the EPIC (Explicit Parallel Instruction set Computing) technology. With EPIC, an Itanium processor family compiler turns sequential code into parallelized 128-bit bundles that can be directly or explicitly processed by the CPU without having to interpret it further. This explicit expression of parallelism allows the processor to concentrate on executing parallel code as fast as possible, without further optimizations or interpretations. On the contrary, a regular (non-Itanium's processor family) compiler takes a sequential code and examines and optimizes it for parallelism, but then has to regenerate sequential code in a such a way that the processor can re-extract the parallelization from it. The processor then has to read this implied parallelism from the machine code, re-build it, and run it. The parallelism is there, but it is not as obvious to the processor, and more work has to be done by the hardware before it can be utilized.

Unlike the RISC processors (as used in the SGI Origins) that dedicate an enormous amount of chip real estate and logic to hide cache misses (by allowing instructions to be executed out of order, which works well when the ratio of CPU frequency to memory frequency is relatively small), the EPIC processors rely on the software to make sure that the data is in the proper cache at the proper time. Instructions are issued in order, so there is no hardware mechanism to hide a cache miss.

The Itanium processors use long instruction words. Specifically, three instructions are grouped into a 128-bit bundle. Each instruction is 41 bits wide. The least significant 5 bits encode a bundle template. The template field encodes (1) the execution units (integer units I, memory units M, floating point units F, and branch units B) needed by the three instructions, and (2) which instructions can be executed in parallel. For the Itanium 2 chips, two bundles can be executed per cycle.

Four memory-load operations per cycle can be delivered from the L2 cache to the floating-point register file. This will completely support two floating-point operations per cycle; this translates into 4 flops per cycle using the FMA operation.

Branch Predication

Without predication, parallelism would be impossible. Instead of waiting for each section of a complex calculation to finish, it is faster if the processor can predict the outcome and proceed on the basis of that prediction. These prediction points are called branches, and current processors try to guess which branch to take. If it predicts correctly, the whole calculation is validated. If it predicts incorrectly, the string has to be thrown out and the calculation starts over. The Itanium processor family architecture minimizes wasted calculations by taking both possible paths to the next branch, where it follows both branches again. When it comes to the correct result it drops the other branch path that it doesn't need, keeps the branch that it does and it continues on with the calculation.

Speculative Loads

A processor needs to access the memory to get code to execute, but while it fetches this code it is not executing instructions. A processor based on the Itanium processor family architecture specification can look ahead at its instruction and load the required data from the memory early; so, when those instructions begin to execute, they have the required data, even if the loaded data changes.

Processor Details

- 128 integer registers; up to 96 rotating

Note: 32 registers are fixed and 96 are "stacked". A procedure call can allocate up to 96 of the stacked registers and still has access to the 32 common registers. Each procedure has its own register frame, which is flexible in size. Since most procedure calls will allocate only a few new registers, many calls can be made before the physical limits of the register file are exceeded. A dedicated piece of hardware called the Register Stack Engine (RSE) will quickly and automatically spill older registers to free up space in the register stack for the new request. The RSE will also restore spilled registers as needed.

- 128 floating-point registers; up to 96 rotating
- 64 1-bit predicate registers; up to 48 rotating
- 8 branch registers
- 128 application registers (for example, loop or epilog counters for loop optimization)
- Performance Monitor Unit (PMU)
- Advanced Load Address Table (ALAT) ALAT keeps track of speculative, or advance loads. However, an excessive number of ALAT comparisons that result in a failed

- advance load will seriously degrade performance
- 3 predicated instructions in a single 128-bit bundle
- 2 bundles (that is, 6 instructions) per clock cycle
- 6 integer units
- 2 loads and 2 stores per clock cycle
- 11 issue ports

Main Memory - Global Shared Memory

SGI Altix systems dramatically reduce the time and resources required to run applications by managing extremely large data sets in a single, system-wide, shared-memory space called global shared memory. Global shared memory means that a single memory address space is visible to all system resources, including microprocessors and I/O, across all nodes. Systems with global shared memory allow access to all data in the system's memory directly and efficiently, without having to move data through I/O or network bottlenecks. On the contrary, clusters with multiple nodes without global shared memory must pass copies of data, often in the form of messages, which can greatly complicate programming and slow down performance by increasing the time processors must wait for data.

If an Altix system is configured as a multi-partition cluster, global shared memory can be achieved by using a sophisticated system memory interconnect like SGI's NUMALink and application libraries that enable shared-memory calls, such as MPT and XPMEM (a driver which allows shared memory across partitions) from SGI.

To configure an Altix system as a single system image machine, special versions of a scalable operation system from SGI is used and no XPMEM is needed. The current version of the OS used is "2.6.16.60-0.42.9.1-nasa64k #1 SMP".

The SGI Altix systems use the non-uniform memory access (NUMA) model. Memory subsystems from different nodes are connected through SHUB and NUMALink interconnects.

Latency

The local memory latency (within a node) is about 145 nanoseconds (ns). Latency from the other node of the same C-brick is 290 ns. Each additional router hop adds 45 - 50 ns (for NUMALink 3 protocol). Each meter of NUMALink cable adds 10 ns.

Maximum number of router hops:

- 16 CPUs - 3 hops
- 32 CPUs - 4 hops
- 64 CPUs - 5 hops

- 128 CPUs - 5 hops
- 256 CPUs - 7 hops

Bandwidth

The Altix memory subsystem uses PC-style double data rate (DDR) SDRAM DIMMs. Each SHUB supports four DDR buses. Each DDR bus may contain up to four DIMMs. The four memory buses are independent and can operate simultaneously to provide up to 12.8 GB/sec of memory bandwidth. (Local memory bandwidth for DIMM type PC2700 is 10.2 GB/sec; and for type PC3200, it is 12.8 GB/sec). While the local processor bus has a peak bandwidth (between L3 cache and memory) of 6.4 GB per second, the local memory subsystem has enough bandwidth to fully saturate the local processor demands while leaving available bandwidth to service remote processor and I/O memory requests.

Article ID: 83

Last updated: 14 Feb, 2013

Computing at NAS -> Computing Hardware -> Columbia -> Columbia Configuration Details
<http://www.nas.nasa.gov/hecc/support/kb/entry/83/?ajax=1>