Table of Contents

Computing Hardware **1**

Pleiades **2**
- Pleiades: Introduction 2
- Pleiades Configuration Details 3
- Harpertown Processors 6
- Nehalem-EP Processors 8
- Westmere Processors 11
- Pleiades Home Filesystem 13
- Pleiades Lustre Filesystems 15
- Pleiades Front-End Usage Guidelines 17
- Pleiades Interconnect 19
- GPU Basics 21
- Sandy Bridge Processors 23

Columbia **26**
- Columbia: Introduction 26
- Columbia Hardware Overview 27
- Columbia Configuration Details 29
- Columbia Home Filesystems 34
- Columbia CXFS Filesystems 36
- Columbia Front-End Usage Guidelines 38

Computing Hardware

Pleiades

Pleiades: Introduction

Pleiades is the primary supercomputer at NAS. Originally installed in 2008 with 51,200 cores, it has been further expanded at various stages. The following articles provide hardware information at varying levels of detail:

- Pleiades Hardware Overview - a high-level overview of the Pleiades system architecture, including resource summaries of the compute and front-end nodes, the interconnect, and the storage capacity.

- Pleiades Configuration Details - focuses on the hardware hierarchy (from the processors to the whole cluster) and provides more detailed configuration statistics on the processors and their associated memory.

- Harpertown Processors, Nehalem-EP Processors, Westmere Processors, and Sandy Bridge Processors (four articles) - provide configuration diagrams and additional information such as core labeling, instruction set, hyperthreading, and Turbo Boost, for each of Pleiades' four processor types.

- Pleiades Home Filesystem - information on quota and backup policies on the home filesystem.

- Pleiades Lustre Filesystems - details the configurations of the Lustre filesystems and users' quotas on these filesystems.

- Pleiades Interconnect - information on the topology, latency, and bandwidth of the Pleiades InfiniBand fabric.

- Pleiades Front-End Usage Guidelines - guidelines on using the front-end nodes and bridge nodes.

Pleiades Configuration Details

Pleiades Hardware Summary

- 184 racks: 64 Harpertown, 20 Nehalem, 72 Westmere, 2 Westmere+GPU, and 26 Sandy Bridge
- 11,920 nodes
- 129,024 CPU cores + 32,768 GPU cores
- 237 TB total memory
- 1.79 petaflops theoretical peak performance (CPU) + 43 teraflops (GPU)

Hostnames

The hostname of each of the 11,920 Pleiades compute nodes (accessible only through PBS jobs) is based on the physical rack and individual rack unit (IRU) it resides in and its node position in the IRU. There are 4 IRUs in each rack. For Harpertown, Nehalem, and Westmere, there are 16 nodes per IRU. For Sandy Bridge, there are 18 nodes per IRU.

For the Harpertown, Nehalem, and Westmere racks, each rack is controlled by a rack leader, while for Sandy Bridge every two racks are controlled by a rack leader. The hostnames of each group of 144 (2 racks x 4 IRUs x 18 nodes) Sandy Bridge nodes reflect their control by the same rack leader.

The hostnames are:

- Harpertown nodes - r[1-64]i[0-3]n[0-15]
- Nehalem nodes - r[161-170,177-186]i[0-3]n[0-15]
- Westmere nodes - r[129-160,171-176,187-218, 219, 221-222]i[0-3]n[0-15]
- Sandy Bridge nodes - r[3xx]i[0-7]n[0-17], where 3xx are *odd* numbers between 301-311 and 317-329.

Processor, Memory and Network Subsystems Statistics

Below are detailed configuration statistics for the processor and memory subsystems for all Pleiades compute nodes:

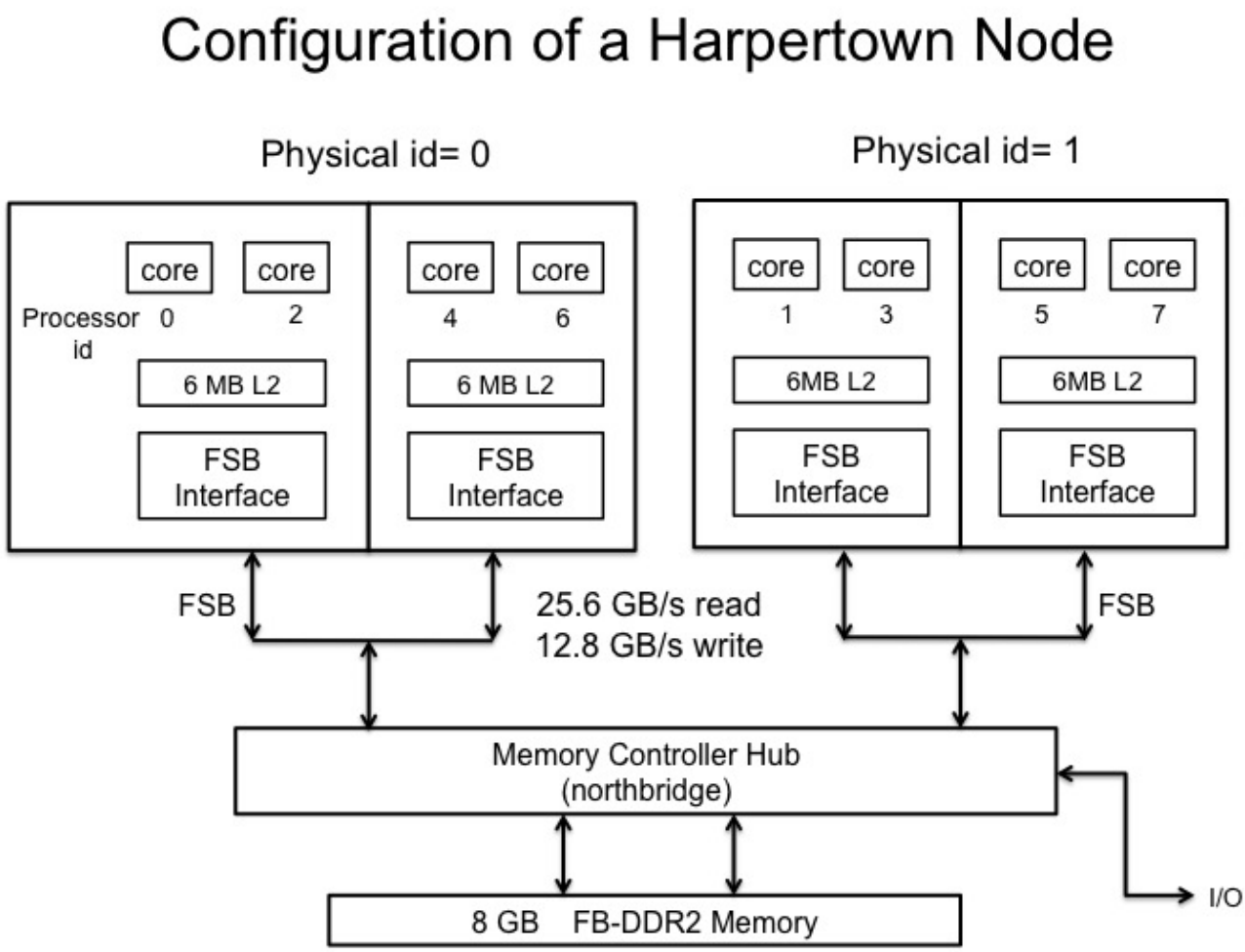
Pleiades Processor, Memory, and Network Subsystems Statistics

Architecture	ICE 8200EX	ICE 8200EX	ICE 8400EX	ICE X
		Processor		
CPU	**Harpertown**	**Nehalem-EP**	**Westmere**	**Sandy Bridge**
	Quad-Core	Quad-Core	6-Core	8-core
	Xeon E5472	Xeon X5570	Xeon	Xeon E5-2670

			X5670/X5675 (r221-222)	
Newest Instruction Set	SSE 4.1	SSE 4.2	SSE 4.2	AVX
Hyper-Threading	N/A	ON	ON	ON
TurboBoost	N/A	ON	ON	ON
CPU-Clock	**3.00 GHz**	**2.93 GHz**	**2.93/3.06 (r221-222) GHz**	**2.6 GHz**
Maximum Double Precision Floating Point Operations per Cycle per Core	4	4	4	**8**
# of Cores/node	8	8	12	16
Total # of Nodes	4,096	1,280	4,672	1,872
Total # of Cores	32,768	10,240	56,064	29,952
Total Double Precision TFlops	393	120	658	623
		Memory		
L1 Cache	Local to each core; Instruction cache: 32K Data cache: 32K; Associativity: 8 ; Cache line size: 64 B	Local to each core; Instruction cache: 32K Data cache: 32K; Associativity: 4 (Instruction) or 8 (Data); Cache line size: 64 B	Local to each core; Instruction cache: 32K Data cache: 32K; Associativity: 4 (Instruction) or 8 (Data); Cache line size: 64 B	Local to each core; Instruction cache: 32K Data cache: 32K; Associativity: 8; Cache line size: 64 B
L2 Cache	12MB on-die for the Quad-Core; 6MB per core pair; shared by the two cores. L2 Cache speed: 3 GHz Associativity: 24; Cache line size: 64 B	256 KB per core; Associativity: 8; Cache line size: 64 B	256 KB per core Associativity: 8; Cache line size: 64 B	256 KB per core; Associativity: 8; Cache line size: 64 B
L3 Cache	N/A	8 MB shared by the four cores; Associativity: 16;	12 MB shared by the six cores; Associativity: 16; Cache line size:	20 MB shared by the eight cores; Associativity:

		Cache line size: 64 B	64 B	20; Cache line size: 64 B
TLB	local to each core	local to each core	local to each core	local to each core
Default Page Size	4 KB	4 KB	4 KB	4 KB
Memory/Core	**1 GB; DDR2**	**3 GB; DDR3**	**2 GB; DDR3**	**2 GB; DDR3**
Total Memory/node	8 GB; nodes in r32 at 16 GB	24 GB	24 GB; 17 nodes at 48 GB; 4 nodes at 94 GB	32 GB
Memory Speed and Bandwidth	1600 MHz; 25.6 GB/sec read 12.8 GB/sec write	1333 MHz; 3 channels; 32 GB/sec read/write	1333 MHz; 3 channels; 32 GB/sec read/write	1600 MHz; 4 channels; 51.2 GB/sec read/write
QuickPath Interconnect	N/A	6.4 GT/s or 25.6 GB/sec	6.4 GT/s or 25.6 GB/sec	8.00 GT/s or 32 GB/sec
Inter-node Network				
IB Device on node	4x DDR HCA; 20 Gbits/s	4x DDR HCA; 20 Gbits/s	4x QDR HCA; 40 Gbits/s	4x FDR dual-port IB Mezzanine card; 56 Gbits/s
IB Switches between nodes	4x DDR; 20 Gbits/s	4x QDR; 40 Gbits/s	4x QDR; 40 Gbits/s	4x FDR; 56 Gbits/s

Harpertown Processors



Core Labeling

The core labeling as shown in this diagram is obtained from the command **`cat /proc/cpuinfo`**. Note that in the first socket (that is, phyiscal id=0), the four cores are labeled 0, 2, 4, and 6, and are not contiguous. Similarly, in the second socket (physical id=1), they are labeled as 1, 3, 5, and 7. In addition, each core pair (0,2), (4,6), (1,3) and (5,7) shares a 6 MB L2 cache.

For performance consideration, care must be taken if one tries to use tools such as **`dplace`** to pin processes to specific processors. Be aware of the non-contiguous nature of the labeling and the sharing of L2 cache per core pair. Also, when using the SGI MPT library, the environment variable **`MPI_DSM_DISTRIBUTE`** has been set to OFF for the Harpertown nodes since setting **`MPI_DSM_DISTRIBUTE`** to ON causes the processes to be pinned to processors in a contiguous order. For example, MPI ranks 0-7 are pinned to processors 0-7, respectively. This results in bad performances for most applications.

SSE4 Instruction Set

Intel's Streaming SIMD Extensions 4.1 (SSE4.1) instruction set is included in the Harpertown processors.

Since the instruction set is upward compatible, an application which is compiled with **`-xSSE4.1`** (with Intel version 11 compiler) can run on Harpertown, Nehalem-EP, Westmere, or Sandy Bridge processors. An application which is compiled with **`-xSSE4.2`** can run only on Nehalem-EP or Westmere processors. An application that is compiled with **`-xAVX`** can run only on Sandy Bridge processors.

TIP: If you want to have a single executable that will run on any of the four Pleiades processor types, with suitable optimization to be determined at runtime, you can compile your application with **`-O3 -ipo -axAVX -xSSE4.1`**.

Hyperthreading

Not available.

Turbo Boost

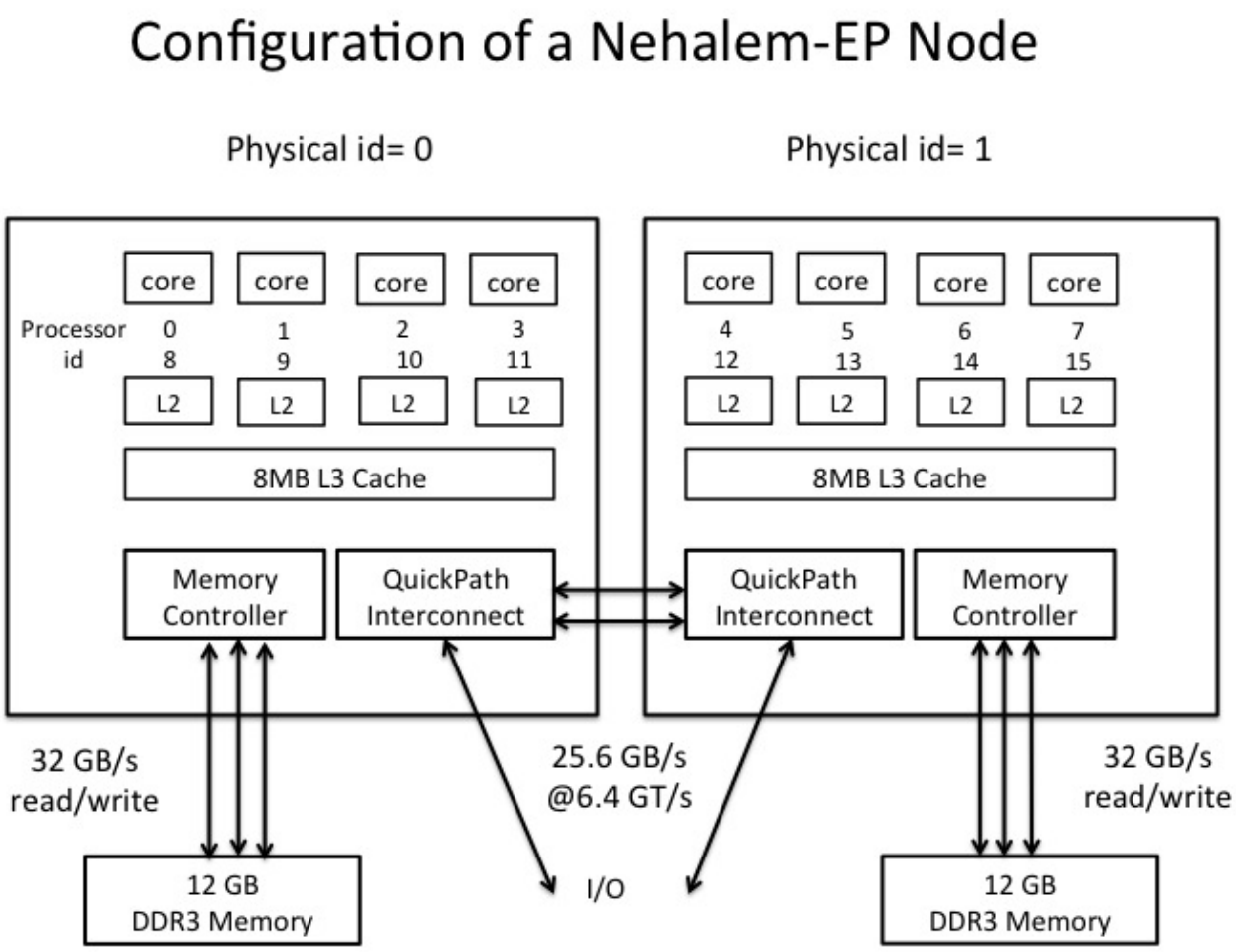
Not available.

Front-Side Bus

The Harpertown (quad-core Intel Xeon E5472) processors use 1600 MHz Front-Side Bus (FSB). The processor transfers data four times per bus clock (4x data transfer rate, as in AGP 4x). Along with the 4x data bus, the address bus can deliver addresses two times per bus clock and is referred to as a double-clocked or a 2x address bus. In addition, the Request Phase completes in one clock cycle. Working together, the 4x data bus and 2x address bus provide a data bus bandwidth of up to 12.8 GB per second. The FSB is also used to deliver interrupts.

Nehalem-EP Processors

The Nehalem-EP processors are part of the Pleiades compute subsystem, among the four types of nodes on which you can execute jobs. When you run a job on Pleiades, you choose which types of nodes to use; Nehalem ("nuh-HAY-lem") is one of your choices.



Core Labeling

Unlike the Harpertown nodes, the core labeling in Nehalem-EP (and Westmere) is contiguous. That is, cores 0-3 are in the first socket and cores 4-7 are in the second socket.

When using the SGI MPT library, the environment variable **`MPI_DSM_DISTRIBUTE`** is by default set to ON for the Nehalem-EP (and Westmere) nodes.

SSE4 Instruction Set

Intel's Streaming SIMD Extensions 4.2 (SSE4.2) instruction set is included in the Nehalem-EP processors.

Since the instruction set is forward compatible, an application that is compiled with **`-xSSE4.1`** (with Intel version 11 compiler) can run on the Harpertown, Nehalem-EP, Westmere, or Sandy Bridge processors. An application compiled with **`-xSSE4.2`** can run only on Nehalem-EP, Westmere, or Sandy Bridge processors, while an application compiled with **`-xAVX`** can run only on Sandy Bridge processors.

TIP: If you want to have a single executable that will run on any of the four Pleiades processor types, with suitable optimization to be determined at runtime, you can compile your application with **`-O3 -ipo -axAVX -xSSE4.1`**.

Hyperthreading

On Nehalem-EP (and Westmere) nodes, hyperthreading is available by user request -- for example, by asking for more than 8 MPI ranks per Nehalem-EP node.

When hyperthreading is requested, the OS views each physical core as two logical processors and can assign two threads to it.

Preliminary benchmarking by NAS staff shows that many jobs would benefit from using hyperthreading, so this feature is currently turned ON, meaning that it is available if a job requests it.

Mapping of Physical Cores and Logical Processor IDs

Physical id	Core id	Processor id Hyperthreading OFF	Processor id Hyperthreading ON
0	0	0	0 ; 8
0	1	1	1 ; 9
0	2	2	2 ; 10
0	3	3	3 ; 11
1	4	4	4 ; 12
1	5	5	5 ; 13
1	6	6	6 ; 14
1	7	7	7 ; 15

With hyperthreading, you can run an MPI code with 16 processes per Nehalem-EP node, instead of just 8. Each of the 16 processes will be assigned to run on one logical processor. In reality, two processes are running on the same physical core. If one process does not keep the functional units in the core busy at all time, and can share the resources in the core with another process, then running in this mode will take less than two times the wall-time compared to running only one process on the core. This can improve the overall

throughput, as demonstrated in the following example:

Consider the following scenario for a job that uses 16 MPI ranks. Without hyperthreading we would use:

```
#PBS -lselect=2:ncpus=8:mpiprocs=8 -lplace=scatter:excl
```
and the job will use 2 nodes with 8 processes per node. Suppose the job takes 1,000 seconds when run this way. If we run the job with hyperthreading, for example:

```
#PBS -lselect=1:ncpus=16:mpiprocs=16 -lplace=scatter:excl
```
then the job will use 1 node with all 16 processes running on that node.

Now suppose this job takes 1,800 seconds to complete. Without hyperthreading, we used 2 nodes for 1,000 seconds (a total of 2,000 node-seconds); with hyperthreading we used 1 node for 1,800 seconds (1,800 node-seconds). Under these circumstances, if you were interested in getting the best wall-clock performance for a single job, you would use two nodes without hyperthreading. However, if you were interested in minimizing resource usage, especially with multiple jobs running simultaneously, use of hyperthreading would save you 10% utilization.

An added benefit of using fewer nodes with hyperthreading, is that when Pleiades is loaded with many jobs, then asking for half as many nodes may allow your job to start running sooner, with a resulting improvement in your job throughput.

Note that hyperthreading does not benefit all applications. Some applications may also show improvement with some process counts but not with others. For example, a 256-process OVERFLOW job shows benefit with hyperthreading, while a 32-process OVERFLOW job does not.

TIP: You may experience unforeseen issues with hyperthreading. We suggest testing your applications with and without hyperthreading before making a choice for production runs. If your application runs more than two times slower with hyperthreading, then it doesn't make sense to use it.

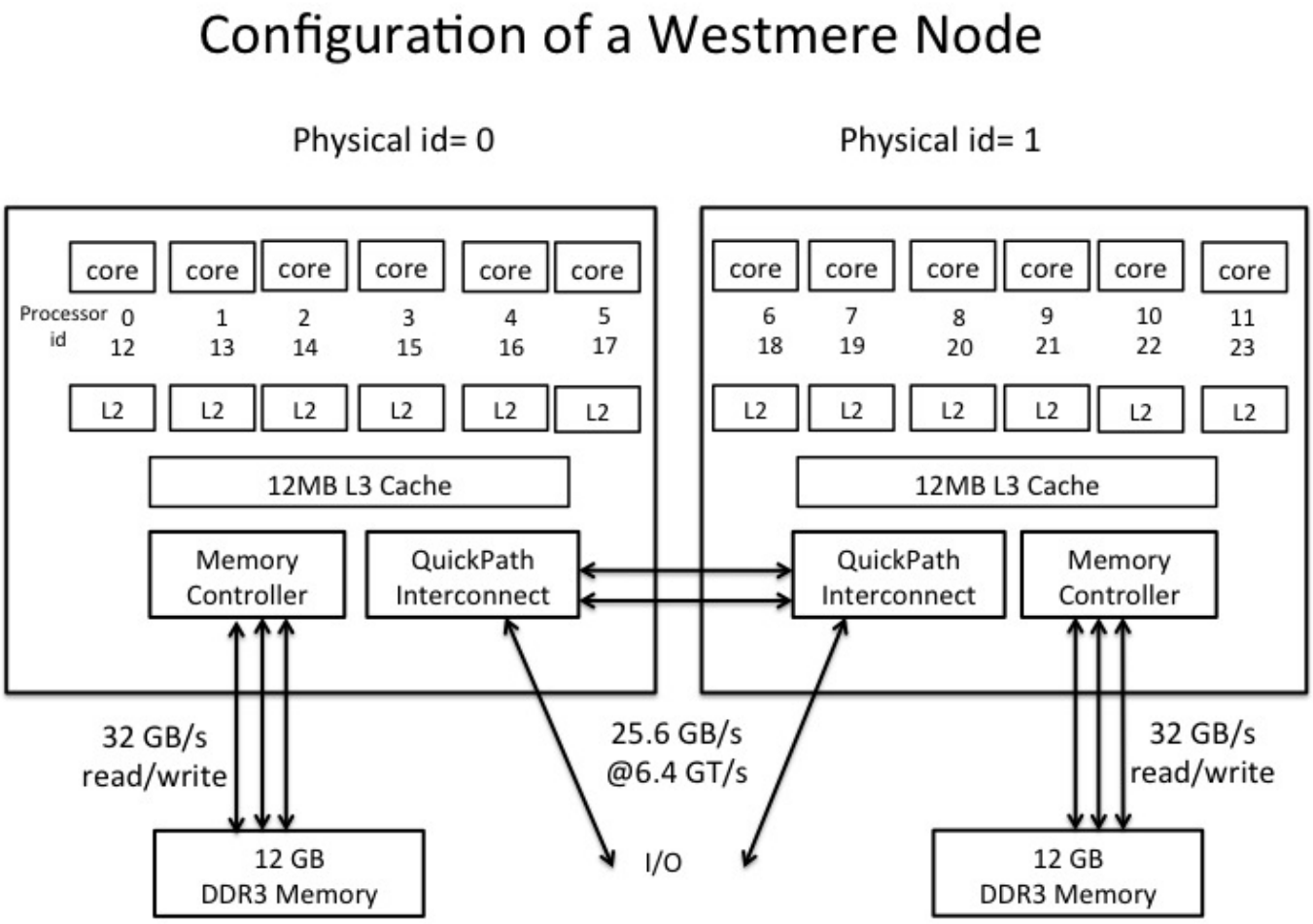
Turbo Boost

Turbo Boost is available on Nehalem-EP nodes (and on Westmere and Sandy Bridge nodes).

When Turbo Boost is enabled, idle cores are turned off and and power is channeled to the cores that are active, making them faster. The net effect is that the active cores perform above their nominal clock speed (this is known as overclocking).

Turbo Boost mode is set up in the system BIOS, and is currently set to ON.

Westmere Processors



Core Labeling

Unlike Harpertown, the core labeling in Westmere is contiguous. That is, cores 0-5 are in the first socket and cores 6-11 are in the second socket.

When using the SGI MPT library, the environment variable **`MPI_DSM_DISTRIBUTE`** is set to ON by default for the Westmere nodes.

SSE4 Instruction Set

Intel's Streaming SIMD Extensions 4.2 (SSE4.2) instruction set is included in the Westmere processors.

Since the instruction set is upward compatible, an application that is compiled with **`-xSSE4.1`** (with Intel version 11 compiler) can run on Harpertown, Nehalem-EP, Westmere, or Sandy Bridge processors. An application that is compiled with **`-xSSE4.2`** can run only on Nehalem-EP or Westmere processors. An application that is compiled with **`-xAVX`** can run only on Sandy Bridge processors.

TIP: If you want to have a single executable that will run on any of the four Pleiades processor types, with suitable optimization to be determined at runtime, you can compile your application with **`-O3 -ipo -axAVX -xSSE4.1`**.

Hyperthreading

Hyperthreading is available by user request on the Westmere nodes (for example, by asking for more than 12 ranks per node).

Turbo Boost

Turbo Boost is set to ON on the Westmere nodes.

Pleiades Home Filesystem

Summary: You automatically have a home directory on Pleiades, with a quota of 8-10 GB of storage. For temporary storage of larger files, use the /nobackupp file systems. For long-term storage, use the Lou mass storage systems.

The home filesystem on Pleiades is an SGI NEXIS 9000 filesystem. It is NFS-mounted on all of the Pleiades front-end systems (PFEs), bridge[*1-4*], and compute nodes.

Once you are granted an account on Pleiades, your home directory is set up automatically during your first login.

Quota Limits and Policy

Disk space quota limits are enforced on the home filesystem. By default, the soft limit is 8 GB and the hard limit is 10 GB. There are no inode limits on the home filesystem.

To check your quota and usage on your home filesystem, do the following:

```
%quota -v
Disk quotas for user username (uid xxxx):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
saturn-ib1-0:/mnt/home2
                7380152  8000000 40000000          190950       0       0
```

The NAS quota policy states that if you exceed the soft quota, an email will be sent to inform you of your current usage and how much of the grace period remains. It is expected that you will occasionally exceed your soft limit; however, after 14 days, users who are still over their soft limit will have their batch queue access to Pleiades disabled. If you believe that you have a long-term need for higher quota limits on Pleiades, send an email justification to support@nas.nasa.gov. This will be reviewed by the HECC Deputy Project Manager, Bill Thigpen, for approval.

NOTE: For temporary storage of larger files, or a large number of files, use your /nobackupp directory. For normal long-term file storage, transfer your files to the Lou mass storage systems.

See also: Quota Policy on Disk Space and Files.

TIP: If you receive the following error when logging into Pleiades...

```
/usr/X11R6/bin/xauth:  error in locking authority file /u/username/.Xauthority
```

...you won't be able to run X applications. This error is most likely caused by your home filesystem quota being exceeded and you will have to decrease your disk usage to

eliminate this error.

Backup Schedule

Files on the home filesystem are backed up daily.

Pleiades Lustre Filesystems

Summary: The Lustre filesystems on Pleiades are called "nobackup." As the name suggests, these filesystems are for temporary use, and are not backed up. Lustre can handle many large files, but you cannot store those files on Pleiades; if you want to save them, move them to Lou.

Pleiades has several Lustre filesystems (**`/nobackupp[1-6]`**) that provide a total of about 6.795 petabytes of storage and serve thousands of cores. These filesystems are managed under Lustre software version 1.8.6.

Lustre filesystem configurations are summarized at the end of this article.

WARNING: As the names suggest, these filesystems are not backed up, so any files that are removed *cannot* be restored. Essential data should be stored on Lou[*1-2*] or on other, more permanent storage.

Which /nobackup Should I Use?

Once you are granted an account on Pleiades, you will be assigned to use one of the Lustre filesystems. Find out which Lustre filesystem you have been assigned to by typing the following:

```
pfe1% ls -l /nobackup/your_username
lrwxrwxrwx 1 root root 19 Feb 23  2010 /nobackup/username -> /nobackupp2/username
```

In the above example, the symlink from **`/nobackup`** to **`/nobackupp2`** shows that the user's assigned nobackup system is **`/nobackupp2`**.

Default Quota and Policy on /nobackup

Disk space and inodes quotas are enforced on the /nobackup filesystems. The default soft and hard quota limits for inodes are 75,000 and 100,000, respectively. Those for the disk space are 500 gigabytes and 1 terabyte, respectively. To check your disk space and inodes usage and quota on your /nobackup, use the **`lfs`** command and type the following:

```
%lfs quota -u username /nobackup/username
Disk quotas for user username (uid nnnn):
   Filesystem  kbytes       quota   limit   grace   files   quota   limit   grace
/nobackup/username 1234  530000000 1100000000    -     567   75000  100000       -
```

The NAS quota policy states that if you exceed the soft quota, an email will be sent that lists your current usage and remaining grace period. It is expected that users will occasionally exceed their soft limit, as needed; however after 14 days, users who are still over their soft

limit will have their batch queue access to Pleiades disabled.

If you anticipate having a long-term need for higher quota limits, please send a justification via email to support@nas.nasa.gov. This will be reviewed by the HECC Deputy Project Manager for approval.

For more information, see also, Quota Policy on Disk Space and Files.

NOTE: If you reach the hard limit while your job is running, the job will die prematurely without providing useful messages in the PBS output/error files. A Lustre error with code **`-122`** in the system log file indicates that you are over your quota.

In addition, when a Lustre filesystem is full, the jobs writing to it will hang. A Lustre error with code **`-28`** in the system log file indicates that the filesystem is full. The NAS Control Room staff normally will send out emails to those using the most space, asking them to clean up their files.

Lustre File Systems Configurations

In the table below, /nobackupp[*1-6*] are abbreviated as nbp[*1-6*]. P=Petabytes; T=Terabytes

Pleiades Lustre Configurations

Filesystem	nbp1	nbp2	nbp3	nbp4	nbp5	nbp6
# of MDSes	1	1	1	1	1	1
# of MDTs	1	1	1	1	1	1
size of MDTs	0.9T	0.9T	0.6T	0.6T	0.8T	0.9T
# of usable inodes on MDTs	~256x10^6	~256x10^6	~173x10^6	~173x10^6	~512x10^6	~256x10^6
# of OSSes	8	8	8	8	8	8
# of OSTs	120	120	60	60	120	120
size/OST	15T	15T	7.1T	7.1T	15T	7.1T
Total Space	1.7P	1.7P	424T	424T	1.7P	847T
Default Stripe Size	4M	4M	4M	4M	4M	4M
Default Stripe Count	1	1	1	1	1	1

NOTE: After January 13, 2011, directories without an explicit stripe count and/or stripe size adopted the new stripe count of 1 and stripe size of 4MB. However, old files in that directory retain their old default values. New files that you create in these directories will adopt the new default values.

Each Pleiades Lustre filesystem is shared among many users. To get good I/O performance for your applications and avoid impeding the I/O operations of other users, read the related articles listed below.

Pleiades Front-End Usage Guidelines

Summary: Use the Pleiades front-end systems (PFEs) and the bridge nodes for file editing, compiling, short debugging/testing sessions, and batch job submissions.

The PFEs and the bridge nodes are the front-end systems to Pleiades. They provide an environment that allows you to get quick turnaround while performing file editing, file transferring, compiling, short debugging/testing sessions, and batch job submission via PBS to a subset of the Pleiades compute nodes.

WARNING: The new Pleiades front-ends (pfe[*20-27*]) use the Intel Sandy Bridge processors. If you use a PGI compiler to build your executable, be aware that by default the executable is optimized for Sandy Bridge and will not necessarily execute on Harpertown, Nehalem-EP, or Westmere processors. To generate a single executable that will work on all Pleiades processor types, use the option **`-tp=penryn-64,nehalem-64,sandybridge-64`** during compilation with PGI compilers. See PGI Compilers and Tools for more information.
You cannot "ssh" to the compute nodes except for the subset of nodes your PBS job is running on.

The bridge nodes are recommended for the following functions:

Pre- and/or Post-Processing

The large amount of memory on the bridge nodes allows pre- and post-processing applications such as Tecplot, IDL, and Matlab to run faster than on the PFEs. Note that the bridge nodes have the same software as the PFEs. For a list of available applications, run the command **`module avail`**.

File Transfers Between Pleiades and Columbia

Both the Pleiades Lustre filesystems /nobackup and the Columbia CXFS filesystems /nobackup are mounted on the bridge nodes. To copy files between the Pleiades Lustre and Columbia CXFS filesystems, log into a bridge node and use the **`cp, mcp,`** or **`shiftc`** command to perform the transfer.

File Transfers to Mass Storage

The Pleiades /nobackup filesystems are mounted on Lou2. Thus, the easiest way to transfer files between Pleiades and Lou2 is to initiate a command such as **`cp`**, **`mcp`**, **`tar`**, or **`shiftc`** on Lou2. For example:

```
lou% mcp /nobackup/username/foo $HOME
```

If you initiate the transfer on Pleiades, the commands **scp**, **bbftp**, **bbscp**, and **shiftc** are available to do the transfers between a Pleiades front-end or bridge node and Lou. Since **bbscp** uses almost the same syntax as **scp**, but performs faster than **scp**, we recommend using **bbscp** in cases where you do not require the data to be encrypted. For very large file transfers, we recommend the Shift utility, developed at NAS.

See also File Transfer Overview, and File Transfer Commands.

File transfers from the Pleiades compute nodes to Lou must go through one of the PFEs or bridge nodes first. See Streamlining File Transfers from the Pleiades Compute Nodes to Lou for more information.

When sending data to Lou, keep your largest individual file size under 1 TB, as large files will occupy all of the tape drives, preventing other file restores and backups.

Additional Restrictions on Front-end Systems

- No MPI (Message Passing Interface) jobs are allowed to run on the PFEs or the bridge nodes
- A job on bridge[1-2] should not use more than 56 GB; when it does, a courtesy email is sent to the owner of the job
- A job on bridge[3-4] should not use more than 192 GB; when it does, a courtesy email is sent to the owner of the job

Before starting a large-memory session, it is a good idea to check to make sure there is enough memory available. You can run the command **top**, hit "M", and check under the "RES" column for other large memory applications that may be running.

Pleiades Interconnect

Topology

InfiniBand (IB) is used for inter-node communication among all of the Pleiades nodes. A key feature of InfiniBand permits remote direct memory access (RDMA) between processing nodes, allowing direct access to other nodes' memory. This allows developers and application owners to bypass the TCP/IP stack, accelerating the application performance. Two devices are involved in the interconnect: the Mellanox ConnectX host channel adapter chip (HCA) on the motherboard of each node and the Mellanox IB switches. There are two IB switches per half- IRU (which includes 8 nodes). One of the switches is involved in the ib0 fabric, which is used mainly for MPI communication. The other is involved in the ib1 fabric which is used mainly for I/O. InfiniBand uses subnet manager (SM) software to manage the InfiniBand fabric and to monitor interconnect performance and health at the fabric level.

The network topology of each IB fabric of Pleiades is a partial 11-D hypercube. In a 11-D hypercube, each switch has 11 direct connections with 11 other specific switches in the network.

The ib1 hypercube fabric is extended by a set of nine switches connected to the Lustre servers (one for the MDSes, and eight for the OSSes). The plan is for each rack to connect directly to one of the OSS switches and each group of eight racks to connect directly to the MDS switch. Currently, most of the racks are connected this way, but some remain to be connected.

Another set of nine switches on the ib1 fabric provides direct access between hyperwall visualization nodes and Pleiades nodes and Lustre servers.

Latency

The shortest communication path in a Pleiades IB fabric will be for any two nodes located in the same half-IRU of the same rack such that the communication only needs to go through 1 switch. For a fully populated 11-D hypercube, the optimum communication path between any two nodes which are not in the same half-IRU varies from going through 2 to 12 switches, depending on which racks and half-IRUs the two nodes reside. Since the Pleiades IB fabric is not a full 11-D hypercube, some connections are missing that would facilitate the optimum path between some nodes, therefore, it is possible that some communications may go over more than 12 switches.

MPI half Ping-Pong latency starts around 1000 to 1500 ns for communication going through two switches. Each additional switch adds ~100 ns (QDR) to ~150 ns (DDR) to the latency.

Bandwidth

The HCA on each node uses either 4x DDR (double data rate) links or 4x QDR (quad data rate) links. Each link is bi-directional and contains 1 send channel and 1 receive channel. For each direction, the raw data transfer rates for 4x DDR and 4x QDR are 20 Gb/s and 40 Gb/s, respectively. These links use 8b/10b encoding such that every 10 bits sent carry 8 bits of useful data. Thus, for each direction, the effective maximum bandwidth for each node is 16 Gb/s (i.e, 2 GB/s) if 4x DDR HCA is used or 32 Gb/s (i.e. 4 GB/sec) if 4x QDR HCA is used.

The IB switch which every 8 nodes in each half-IRU share through a single path also has similar effective data transfer limits per port: 16 Gb/s for 4x DDR IB switches and 32 Gb/s for 4x QDR IB switches.

The Harpertown and Nehalem-EP nodes use 4x-DDR HCAs while the Westmere nodes use 4x-QDR HCAs. The Harpertown racks use 4x DDR IB switches while the Nehalem-EP and Westmere racks use 4x QDR switches.

These limits also apply to each OSS in the Lustre filesystem. For /nobackupp20 and /nobackupp50, QDR switches are used to connect to the IB fabric and DDR switches are used to connect to the DDNs of the hard disks. For /nobackupp[10,30,40,60], DDR switches are used to connect to both IB fabric and to the DDNs. With DDR switches, the theoretical bandwidth of each OSS is 2 GB/s for each direction. With 8 OSSes per Lustre filesystem, the theoretical peak aggregate bandwidth for each filesystem for each direction would be 16 GB/s. This bandwidth however is reduced to 10 GB/s due to bandwidth that the DDNs can provide. The best benchmark performance obtained for each Pleiades Lustre filesystem is 8 - 10 GB/sec (all read or all write).

The actual I/O bandwidth a user's application experiences is far less than the theoretical peak or even the benchmark data due to factors such as the I/O pattern the application is doing (for example, serial or parallel; for parallel, if the I/O requests are from nodes of different half-IRUs), the number of stripe count used (this affects the maximum aggregate bandwidth provided by the OSSs), how busy the Lustre is handling requests from many users, if there are bad links in the network, etc.

Follow the tips listed in Lustre Best Practices if you are not getting good performances out of the Lustre filesystem.

GPU Basics

A graphics processing unit (GPU) is a hardware device that may be able to accelerate an algorithm or computer code. If you want general information on GPUs, you might start with the Wikipedia article "GPGPU" and the GPGPU website, which has information for developers.

If you have an application that can take advantage of GPU technology, you can use the GPU nodes on Pleiades. Specifically, the 64 Pleiades Westmere nodes on rack 219 (r219i[0-3]n[0-15]) include one NVIDIA Tesla M2090 GPU per node. Each M2090 computing module comprises a computing subsystem with a Tesla 20-series GPU and high speed memory, and is connected to the Westmere node via a PCI Express bus.

The NVIDIA Developer Zone has specific information about NVIDIA GPUs and the programming model for them.

To use the Westmere+GPU nodes, specify the processor model type **`model=wes_gpu`** in your PBS script:

```
#PBS -l select=xx:ncpus=yy:model=wes_gpu
```

Submit your PBS jobs to the **`gpu`** queue as follows:

```
% qsub -q gpu job_script
```

To check the status of your jobs submitted to the **`gpu`** queue:

```
% qstat gpu -u your_username
```

Get basic hardware information about the Pleiades GPUs, as follows:

```
pfe20% qsub -I -q gpu
r219i0n0% /usr/bin/nvidia-smi -q
and/or
r219i0n0% module load comp-pgi/11.6
r219i0n0% pgaccelinfo
```

The output from **`pgaccelinfo`** shows:

```
CUDA Driver Version:           4000
NVRM version: NVIDIA UNIX x86_64 Kernel Module  275.09.07  Wed Jun  8 14:16:46 PDT 2011

Device Number:                 0
Device Name:                   Tesla M2090
Device Revision Number:        2.0
Global Memory Size:            5636554752
Number of Multiprocessors:     16
Number of Cores:               512
Concurrent Copy and Execution: Yes
Total Constant Memory:         65536
Total Shared Memory per Block: 49152
Registers per Block:           32768
```

```
Warp Size:                      32
Maximum Threads per Block:      1024
Maximum Block Dimensions:       1024, 1024, 64
Maximum Grid Dimensions:        65535 x 65535 x 65535
Maximum Memory Pitch:           2147483647B
Texture Alignment:              512B
Clock Rate:                     1301 MHz
Initialization time:            20609 microseconds
Current free memory:            5552726016
Upload time (4MB):              1547 microseconds ( 761 ms pinned)
Download time:                  1113 microseconds ( 681 ms pinned)
Upload bandwidth:               2711 MB/sec (5511 MB/sec pinned)
Download bandwidth:             3768 MB/sec (6159 MB/sec pinned)
```

To use the Pleiades GPUs you have two possibilities:

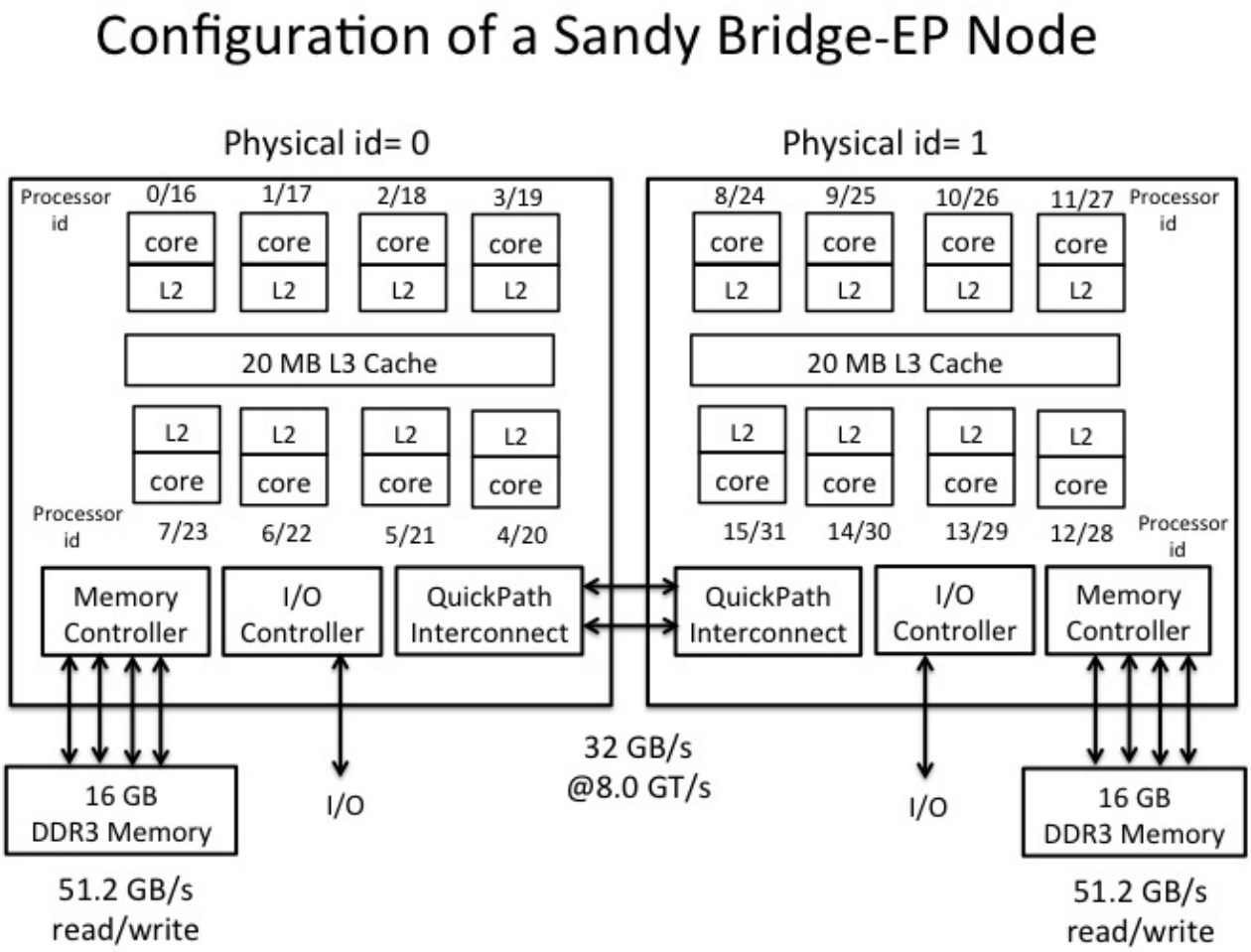
1. The PGI accelerator model requires you to annotate your source code (Fortran or C) with directives describing sections of your code that are to be executed on the GPU. You need to use one of the PGI compiler modules (for example, **`module load comp-pgi/11.6`**). Refer to the Portland Group website for information on their accelerator compilers.
2. You may write (or rewrite) portions of your code in CUDA (Compute Unified Device Architecture; see the NVIDIA Developer Zone) or in CUDA Fortran. If you choose the latter, you will need to use a PGI compiler module (such as **`module load comp-pgi/11.6`**). If you select the former, you will need to load a **`cuda`** module (such as **`module load cuda/4.0`**) and use the CUDA tools. You will also need to determine how to link your new CUDA code with the rest of your program. If you need assistance, contact the NAS Control Room, and our consultants will lend a hand.

Currently, no direct communication exists between a GPU on one node and a GPU on another node. If such communication is required, the data must go via the PCI Express bus from the GPU to the Westmere CPU and via MPI from one CPU to another.

If you delve into CUDA programming, the following book may be useful:

CUDA by Example: An Introduction to General-Purpose GPU Programming, Jason Sanders and Edward Kandrot

Sandy Bridge Processors



Core Labeling

The core labeling in Sandy Bridge is contiguous. That is, cores 0-7 are in the first socket and cores 8-15 are in the second socket.

When using the SGI MPT library, the environment variable **`MPI_DSM_DISTRIBUTE`** is set to ON by default for the Sandy Bridge nodes.

Instruction Set

A Sandy Bridge processor's execution hardware contains the Advanced Vector Extensions (AVX), a set of instructions for doing Single Instruction Multiple Data (SIMD) operations on Intel architecture processors. These extensions widen the vector registers from 128 bits to 256 bits, so the floating-point hardware can sustain 16 single-precision and 8

double-precision floating point operations per cycle. As a result, even though the CPU clock speed of the Sandy Bridge processor (2.6 GHz) is lower than that of the Harpertown (3.0 GHz), Nehalem-EP, and Westmere (2.93 GHz) processors, the floating-point performance can be higher for some applications.

AVX is supported in Intel compilers starting with version 11.1. However, Intel version 12 compilers provide more optimizations for AVX and are recommended over version 11.1.

AVX is also supported in the GNU Compiler Collection starting with version 4.6. An application that is compiled with **`-xAVX`** can run *only* on Sandy Bridge.

TIP: If you want to have a single executable that will run on any of the four Pleiades processor types, with suitable optimization to be determined at run time, you can compile your application with **`-O3 -ipo -axAVX -xSSE4.1`**.

Hyperthreading

Hyperthreading is turned ON.

Turbo Boost

Turbo Boost is turned ON.

Memory Subsystems

The memory hierarchy of Sandy Bridge is as follows:

- L1 instruction cache: 32 KB, private to each core
- L1 data cache: 32 KB, private to each core
- L2 cache: 256 KB, private to each core
- L3 cache: 20 MB, shared by 8 cores in each socket
- Memory: 16 GB per socket, total of 32 GB per node

There are four 1600-MHz memory channels per socket. Each channel can be connected with up to two memory DIMMs. Of the eight memory DIMM slots for each socket, four are populated with 4-GB Error Correcting Code (ECC) registered DDR3 memory, for a total of 16 GB per socket. With two sockets in a node, the total memory per node is 32 GB. If there is a user requirement, some nodes could be configured with larger amounts of memory.

Connecting the two sockets are two Intel QPI links running at a speed of 8.0 Giga-transfers (GT) per second. Each link contains separate lanes for the two directions. The total bandwidth (2 links x 2 directions) is 32 GB/sec.

Network Subsystem

The Sandy Bridge nodes are connected to the two fabrics (ib0 and ib1) of the Pleiades InfiniBand (IB) network via the dual-port, four-link fourteen data rate (4x FDR) IB Mezzanine card on each node, as well as the Mellanox FDR IB switches in the SGI ICE X IB Premium Blade. The FDR runs at 14 Gb/sec per lane. With four links, the total bandwidth is 56 Gb/sec or about 7 GB/sec.

On each node, the IB Mezzanine card sits on a sister board next to the mother board, which contains the two processor sockets.

There are 18 nodes per Individual Rack Unit. These 18 nodes are connected to two Mellanox FDR IB switches in an SGI ICE X IB Premium Blade to join the ib0 fabric. Another set of connection between the 18 nodes and a second Premium Blade is established for ib1.

Columbia

Columbia: Introduction

Columbia Phase Out:

As of Feb. 8, 2013, the Columbia21 node has been taken offline as part of the Columbia phase out process. Columbia22-24 are still available. If your script requires a specific node, please make the appropriate changes in order to ensure the success of your job.

Columbia, an SGI Altix supercomputer named to honor the crew of Space Shuttle Columbia flight STS-107, has been in production since 2004. In March 2008, the system had 14,136 cores in 24 nodes (Columbia1-Columbia24). When the Pleiades system came into production, the original 20 Columbia nodes (1-20) were retired. Columbia currently comprises 1 front-end node (cfe2) and 4 compute nodes (Columbia21-Columbia24).

The following few articles provide Columbia hardware information at varying levels of detail:

Columbia Hardware Overview provides a high-level overview of the Columbia system architecture, including resource summaries of the compute- and front-end nodes, the interconnect, and storage capacity.

Columbia Configuration Details focuses on more detailed configuration statistics of the processors and their associated memory.

The article Columbia Home Filesystem - provides information on the quota and backup policies on the home filesystem.

The article Columbia CXFS Filesystems - details the configurations of the CXFS filesystems and users' quotas on these filesystems.

In addition, the article Columbia Front-End Usage Guidelines provides guidelines on using the front-end node (cfe2).

Columbia Hardware Overview

Columbia Phase Out:

As of Feb. 8, 2013, the Columbia21 node has been taken offline as part of the Columbia phase out process. Columbia22-24 are still available. If your script requires a specific node, please make the appropriate changes in order to ensure the success of your job.

The Columbia supercluster, which ranked 2nd (51.87 teraflops) in the November 2004 TOP500 list, has been in service at NAS for many years. Most of the earlier Columbia nodes (Columbia1 - Columbia20) have been retired. The remaining Columbia nodes (Columbia21-24) continue to serve the NASA community to achieve breakthroughs in science and engineering for the agency's missions and vision for Space Exploration.

Current Columbia System Facts

Manufacturer: SGI

List of nodes for Columbia system			
Nodes	**Type**	**Speed**	**Cache**
1 Altix 4700 (512 cores)	Montecito	1.6 GHz	9MB
1 Altix 4700 (2048 cores)	Montecito	1.6 GHz	9MB
2 Altix 4700 (1024 cores)	Montvale	1.6 GHz	9MB
4 Total Compute Nodes (4,608 Total Cores)			

System Architecture

- 40 compute node cabinets
- 30 teraflops theoretical peak (original 10,240 system: 63 teraflops)

Subsystems

- 1 front-end node

Memory

- Type - double data rate synchronous dynamic random access memory (DDR SDRAM)
- Per Processor (core) - 2GB
- Total Memory - 9TB

Interconnects

- SGI NUMAlink interconnected single-system image compute nodes
- Internode
    - ♦ InfiniBand - 4x (Single Data Rate, Double Data Rate)
    - ♦ 10Gb Ethernet LAN/WAN interconnect
    - ♦ 1Gb Ethernet LAN/WAN interconnect

Storage

- Online - DataDirect Networks & LSI RAID, 1PB (raw)
    - ♦ 1 SGI CXFS domains
    - ♦ Local SGI XFS fileystems
- Archival - Attached to high-end computing SGI CXFS SAN filesystem

Operating Environment

- Operating system - SUSE Linux Enterprise
- Job Scheduler - PBS
- Compilers - C, Intel Fortran, SGI MPT

Columbia Configuration Details

Columbia Phase Out:

As of Feb. 8, 2013, the Columbia21 node has been taken offline as part of the Columbia phase out process. Columbia22-24 are 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.

Columbia Home Filesystems

Columbia Phase Out:

As of Feb. 8, 2013, the Columbia21 node has been taken offline as part of the Columbia phase out process. Columbia22-24 are still available. If your script requires a specific node, please make the appropriate changes in order to ensure the success of your job.

Columbia's home fileystem (**`/u/username`**) is NFS-mounted on the Columbia front-end (cfe2) and compute nodes (Columbia*21-24*).

Once a user is granted an account on Columbia, the home directory is set up automatically during his/her first login.

Quota and Policy

Disk space quota limits are enforced on the home filesystem. By default, the soft limit is 4 GB and the hard limit is 5 GB. There are no inode limits on the home filesystem.

To check your quota and usage on your home filesystem, do:

```
%quota -v
Disk quotas for user username (uid xxxx):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
  ch-rg1:/home6    4888  4000000 5000000            294       0       0
```

The quota policy for NAS states that if you exceed the soft quota, an email will be sent to inform you of your current usage and how much of your grace period remains. It is expected that a user will occasionally exceed their soft limit as needed; however after 14 days, users who are still over their soft limit will have their batch queue access to Pleiades disabled. If you believe that you have a long-term need for higher quota limits, you should send an email justification to support@nas.nasa.gov. This will be reviewed by the HECC Deputy Project Manager, Bill Thigpen, for approval.

The quota policy for NAS can be found here.

WARNING: If you receive the following error when logging into Columbia:

```
/usr/X11R6/bin/xauth:  error in locking authority file
/u/username/.Xauthority
```

you won't be able to run X applications. This error is most likely caused by your home filesystem quota being exceeded and you will have to decrease your disk usage to eliminate this error.

Backup Policy

Files on the home filesystem are backed up daily.

Columbia CXFS Filesystems

Columbia Phase Out:

As of Feb. 8, 2013, the Columbia21 node has been taken offline as part of the Columbia phase out process. Columbia22-24 are still available. If your script requires a specific node, please make the appropriate changes in order to ensure the success of your job.

Columbia CXFS filesystems (**/nobackup[*1-2*][*a-i*]**) are shared and accessible from cfe2 and Columbia*21-24*. This allows user jobs to be load-balanced across Columbia's systems without forcing users to move their data to a particular Columbia system.

Users will have a **/nobackup** directory on one of these shared file systems. To find out where your **/nobackup** directory is, log in to the front-end node and type the following shell command:

```
cfe2% ls -d /nobackup[1-2][a-i]/$USER
/nobackup1f/username/
```

In this example, the user is assigned to **/nobackup1f**.

Default Quota and Policy on /nobackup

Disk space and inodes quotas are enforced on the CXFS **/nobackup[*1-2*][*a-i*]** filesystems. The default soft and hard limits for inodes are 25,000 and 50,000, respectively. Those for disk space are 200GB and 400GB, respectively. To check your disk space and inodes usage and quotas on your CXFS filesystem, do the following:

```
cfe2% quota -v
Disk quotas for user username (uid xxxx):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
/dev/cxvm/nobackup1f
                1673856  210000000 420000000              10973   25000   50000
```

The NAS quota policy states that if you exceed the soft quota, an email will be sent to inform you of your current usage and how much of your grace period remains. It is expected that users will occasionally exceed their soft limit, as needed; however after 14 days, users who are still over their soft limit will have their batch queue access to Columbia disabled.

If you anticipate having a long-term need for higher quota limits, please send a justification via email to support@nas.nasa.gov. This will be reviewed by the HECC Deputy Project Manager for approval.

For more information, see also, Quota Policy on Disk Space and Files.

WARNING: As the names suggest, these filesystems are not backed up, so any files that are removed *cannot* be restored. Essential data should be stored on Lou*1-2* or onto other more permanent storage.

Columbia Front-End Usage Guidelines

The front-end system, cfe2, provide an environment that allows users to get quick turnaround while performing the following: file editing, file management, short debugging and testing sessions, and batch job submission to the compute systems.

Running long and/or large (in terms of memory and/or number of processors) debugging or production jobs interactively or in the background of cfe2 is considered to be inconsiderate behavior to the rest of the user community. If you need help submitting such jobs to the batch systems, please contact a the Control Room at (650) 604-4444 or (800) 331-USER or send e-mail to: support@nas.nasa.gov

Jobs that cause significant impact on the system load of the Columbia front-end machine (cfe2) are candidates for removal in order to bring the front-end systems back to a normal and smooth environment for all users. A cron job regularly monitors the system load and determines if job removal is necessary. The criteria for job removal are described below. Owners of any removed jobs will receive a notification e-mail.

1. To be eligible for removal, the number of processors a front-end interactive job uses can be one (1) or more. Exceptions to this are those programs, utilities, etc. common to users and/or NASA missions that are listed in an "exception file". Examples of these would be: **`bash, cp, csh, emacs, gzip, rsync, scp, sftp, sh, ssh, tar`**, and **`tcsh`**. Users can submit program names to be added to this exception file by mailing requests to: support@nas.nasa.gov.
2. For qualifying processes, the CPU time usage of each process in a job has, on the average, exceeded a threshold defined as: (20 min x 8 / number of processes for the job). That is, a baseline for removal is a job with 8 processors running for more than 20 minutes. The maximum amount of time allowed for each processor in a job is scaled using the formula: 20 min x 8 cpu / number-of-processes. Therefore, the following variations are possible:

    - 160 minutes = (20 * 8) / 1 cpu
    - 80 minutes = (20 * 8) / 2 cpu
    - 40 minutes = (20 * 8) / 4 cpu
    - 20 minutes = (20 * 8) / 8 cpu
    - 10 minutes = (20 * 8) / 16 cpu
    - 5 minutes = (20 * 8) / 32 cpu
    - 2.5 minutes = (20 * 8) / 64 cpu

The conditions of removal are subject to change, when necessary.