Hyperthreading

Hyperthreading is available and enabled on the Pleiades, Aitken, and Electra compute nodes. With hyperthreading, each physical core can function as two logical processors. This means that the operating system can assign two threads per core by assigning one thread to each logical processor.

Note: Hyperthreading is currently off on Aitken Rome nodes.

The following table shows the number of physical cores and potential logical processors available for each processor type:

Processor Model	Physical Cores (N)	Logical Processors (2N)
Sandy Bridge	16	32
Ivy Bridge	20	40
Haswell	24	48
Broadwell	28	56
Skylake	40	80
Cascade Lake	40	80

If you use hyperthreading, you can run an MPI code using 2*N* processes per node instead of *N* process per nodeâ so you can use half the number of nodes for your job. Each process will be assigned to run on one logical processor; in reality, two processes are running on the same physical core.

Running two processes per core can take less than twice the wall-clock time compared to running only one process per coreâ if one process does not keep the functional units in the core busy, and can share the resources in the core with another process.

Benefits and Drawbacks

Using hyperthreading can improve the overall throughput of your jobs, potentially saving standard billing unit (SBU) charges. Also, requesting half the usual number of nodes may allow your job to start running soonerâ an added benefit when the systems are loaded with many jobs. However, using hyperthreading may not always result in better performance.

WARNING: Hyperthreading does not benefit all applications. Also, some applications may show improvement with some process counts but not with others, and there may be other unforeseen issues. Therefore, before using this technology in your production run, you should test your applications with and without hyperthreading. If your application runs more than two times slower with hyperthreading than without, do not use it.

Using Hyperthreading

Hyperthreading can improve the overall throughput, as demonstrated in the following example.

Example

Consider the following scenario with a job that uses 40 MPI ranks on Ivy Bridge. Without hyperthreading, we would specify:

```
#PBS -lselect=2:ncpus=20:mpiprocs=20:model=ivy
```

and the job will use 2 nodes with 20 processes per node. Suppose that the job takes 1000 seconds when run this way. If we run the job with hyperthreading, e.g.:

```
#PBS -lselect=1:ncpus=20:mpiprocs=40:model=ivy
```

then the job will use 1 node with all 40 processes running on that node. Suppose this job takes 1800 seconds to complete.

Without hyperthreading, we used 2 nodes for 1000 seconds (a total of 2000 node-seconds); with hyperthreading, we used 1 node for 1800 seconds (1800 node-seconds). Thus, under these circumstances, if you were interested in getting the best wall-clock time 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, using hyperthreading would save you 10% in SBU charges.

Mapping of Physical Core IDs and Logical Processor IDs

Mapping between the physical core IDs and the logical processor IDs is summarized in the following table. The value of *N* is 16, 20, 24, 28, and 40 for Sandy Bridge, Ivy Bridge, Haswell, Broadwell, and Skylake/Cascade Lake processor types, respectively.

Physical ID	Physical Core ID	Logical Processor ID
0	0	0 ; *N*
0	1	1 ; *N*+1
...	...	.....
0	*N*/2 - 1	*N*/2 - 1; *N* + *N*/2 - 1
1	*N*/2	*N*/2 ; *N* + *N*/2
1	*N*/2 + 1	*N*/2 + 1; *N* + *N*/2 + 1
1	...	...
1	*N* - 1	*N* - 1; 2*N* - 1

Note: For additional mapping details, see the configuration diagrams at the for each processor type, or run the **`cat /proc/cpuinfo`** command on the specific node type.

Article ID: 488
Last updated: 13 May, 2021
Revision: 42
Porting/Building Code -> Optimizing/Troubleshooting -> Process/Thread Pinning -> Hyperthreading
https://www.nas.nasa.gov/hecc/support/kb/entry/488/