this file: using\_perlmutter.docx

|  |  |  |  |
| --- | --- | --- | --- |
|  | Cori Haswell | Cori KNL | Perlmutter CPU |
| Physical cores | 32 | 64 | 128 |
| Logical CPUs per physical core | 2 | 4 | 2 |
| logical CPUs per node | 64 | 272 | 256 |
| NUMA domains | 2 | 1 | 8 |
| -c value for srun | floor(tpn)\*2 | floor(68/tpn)\*4 | floor(128/tpn)\*2 |

Table 1. tpn = tasks per node

Sample Permutter batch script for pure MPI, 10 nodes, 1280 MPI tasks

#!/bin/bash

#!bin/bash

#SBATCH –qos=regular

#SBATCH –constraint=cpu

#SBATCH –time=1:00:00

$SBATCH –nodes=10

export OMP\_NUM\_THREADS=1

srun -n 1280 -c 2 –cpu-bind=cores ./mycode.ext

From: <https://slurm.schedmd.com/srun.html> : srun options

**-c**, **--cpus-per-task**=<*ncpus*>

Request that *ncpus* be allocated **per process**. This may be useful if the job is multithreaded and requires more than one CPU per task for optimal performance. Explicitly requesting this option implies **--exact**. The default is one CPU per process and does not imply **--exact**.

If **-c** is specified without **-n**, as many tasks will be allocated per node as possible while satisfying the **-c** restriction. For instance on a cluster with 8 CPUs per node, a job request for 4 nodes and 3 CPUs per task may be allocated 3 or 6 CPUs per node (1 or 2 tasks per node) depending upon resource consumption by other jobs. Such a job may be unable to execute more than a total of 4 tasks.

**WARNING**: There are configurations and options interpreted differently by job and job step requests which can result in inconsistencies for this option. For example *srun -c2 --threads-per-core=1 prog* may allocate two cores for the job, but if each of those cores contains two threads, the job allocation will include four CPUs. The job step allocation will then launch two threads per CPU for a total of two tasks.

**WARNING**: When srun is executed from within salloc or sbatch, there are configurations and options which can result in inconsistent allocations when -c has a value greater than -c on salloc or sbatch. The number of cpus per task specified for salloc or sbatch is not automatically inherited by srun and, if desired, must be requested again, either by specifying --cpus-per-task when calling srun, or by setting the SRUN\_CPUS\_PER\_TASK environment variable.

This option applies to job and step allocations.

**-n**, **--ntasks**=<*number*>

Specify the number of tasks to run. Request that **srun** allocate resources for *ntasks* tasks. The default is one task per node, but note that the **--cpus-per-task** option will change this default. This option applies to job and step allocations.

see: <https://www.youtube.com/watch?v=SrumAJj4UjU> for a discussion of nodes, cores, threads, tasks

* a node contains one or more processors and each processor can contain multiple cores.
* tasks and threads can be used interchangeably.
* You need to reserve one core for each task (or thread)
* shared memory jobs are jobs that do not use MPI to pass information between nodes, and so shared memory jobs must be run on a single node.
* distributed memory jobs do use MPI and so can be spread across multiple nodes.

mpirun options: taken from <https://www.open-mpi.org/doc/v4.0/man1/mpirun.1.php>

**-N <num>**

Launch num processes per node on all allocated nodes (synonym for npernode).

**-c, -n, --n, -np <#>**

Run this many copies of the program on the given nodes. This option indicates that the specified file is an executable program and not an application context. If no value is provided for the number of copies to execute (i.e., neither the "-np" nor its synonyms are provided on the command line), Open MPI will automatically execute a copy of the program on each process slot (see below for description of a "process slot"). This feature, however, can only be used in the SPMD model and will return an error (without beginning execution of the application) otherwise. **-<>**

* Launch N times the number of objects of the specified type on each node.

from: <https://docs.oracle.com/cd/E19708-01/821-1319-10/ExecutingPrograms.html#50593798_62903>

* mpirun -np x progam1 🡪 starts ‘x’ copies of the program program1
* Using Environmental variables witn the mpirun command

setenv DISPLAY myworkstation:0

mpirun -x DISPLAY -x LD\_LIBRARY\_PATH=/opt/SUNWhpc/HPC8.2.1.c/sun/lib -np 4 a.out

examples

task: run multiple parallel pcoresses: -c or -np <number of processes>

* To Direct mpirun By using an Appfile

You can use a type of text file (called an appfile) to direct mpirun. The appfile specifies the nodes on which to run, the number of processes to launch on each node, and the programs to execute in a parallel application. When you use the  
--app option, mpirun takes all its direction from the contents of the appfile and ignores any other nodes or processes specified on the command line.

For example the following shows an appfile called my\_appfile:

# Comments are supported; comments begin with #

# Application context files specify each sub-application in the

# parallel job, one per line. The first sub-application is the 2

# a.out processes:

-np 2 a.out

# The second sub-application is the 2 b.out processes:

-np 2 b.out

To use the –app option with the mpirun command, specify the name and path of the appfile on the command line. For example:

mpirun –app my\_appfile

This command produces the same results as running a.out and b.out from the command line.

Mapping MPI processes to Nodes

To determine where the processes will run, ORTE uses the following criteria:

* Available hosts (also referred to as nodes), specified by a hostfile or by the --host option
* Scheduling policy (round-robin or by-slot)
* Default and maximum numbers of slots available on each host
* ORTE also checks to see whether the current environment/shell runs with any third-party launcher (such as Sun Grid Engine or PBS) to determine where the processes will launch.

### Specifying Available Hosts

You specify the available hosts to Open MPI in three ways:

* Through the batch scheduler in your resource management software. This option is described in detail in [Chapter 6](https://docs.oracle.com/cd/E19708-01/821-1319-10/ExecutingBatchPrograms.html#50593817_32725).
* By using a hostfile with the --hostfile option. The hostfile is a text file that contains the names of hosts, the number of available slots on each host, and the maximum slots on each host.
* By using the --host option. Use this option to specify which hosts to include or exclude.