**VASP**

**VASP and Slurm**

In Slurm, there is big difference between --ntasks and --cpus-per-task

For the purposes of VASP, --ntasks-per-node should always equal NCORE (in your INCAR file). Then --nodes should be equal to the total number of cores you want, divided by --ntasks-per-node.

VASP has two parameters for controlling processor layouts, NCORE and NPAR, but you only need to set one of them. If you set NCORE, you don’t need to set NPAR. Instead VASP will automatically set NPAR.

In your mpirun line, you should specify the number of MPI tasks as:

```
mpirun -n $SLURM_NTASKS vasp_std
```

**Cores Layout Examples**

If you want 40 cores (2 nodes and 20 cpus per node):

in your submission script:

```
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=20

mpirun -n 2 vasp_std
```

in INCAR:

```
NCORE=20
```
You may however find that the wait time to get 20 cores on two nodes can be very long since cores request via `--cpus-per-task` can’t span multiple nodes. Instead you might want to try breaking it up into smaller chunks. Therefore, try:

in your submission script:

```bash
#SBATCH --nodes=4
#SBATCH --ntasks-per-node=10
mpirun -n 4 vasp_std
```

in INCAR:

```plaintext
NCORE=10
```

which would likely spread over 4 nodes using 10 cores each and spend less time in the queue.

**Grace mpi partition**

On Grace’s `mpi` partition, since cores are assigned as whole 24-core nodes, `NCORE` should always be equal to 24 and then you can just request `ntasks` in multiples of 24.

in your submission script:

```bash
#SBATCH --ntasks=48 # some multiple of 24
mpirun -n $SLURM_NTASKS vasp_std
```

in INCAR:

```plaintext
NCORE=24
```

**Additional Performance**

Some users have found that if they actually assign 2 MPI tasks per node (rather than 1), they see even better performance because the MPI tasks don’t span the two sockets on
the node. To try this, set $NCORE$ to half of your nodes’ core count and increase `mpirun -n` to twice the number of nodes you requested.