Terremoto - Technical Information

- **Hardware**
  - **Standard Nodes**
  - **High Memory Nodes**
  - **GPU Nodes**
  - **Storage**
  - **Network**
  - **Scheduler**
  - **Fair share**

**Hardware**

Terremoto has 110 total execute nodes, consisting:

- 92 Standard Nodes
- 10 High Memory Nodes
- 8 GPU Nodes

**Standard Nodes**

Terremoto has 92 Standard Nodes with the following specifications:

<table>
<thead>
<tr>
<th>Model</th>
<th>Dell C6420</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel Xeon Gold 6126 2.6 Ghz</td>
</tr>
<tr>
<td>Number of CPUs</td>
<td>2</td>
</tr>
<tr>
<td>Cores per CPU</td>
<td>12</td>
</tr>
<tr>
<td>Total Cores</td>
<td>24</td>
</tr>
<tr>
<td>Memory</td>
<td>192 GB</td>
</tr>
<tr>
<td>Network</td>
<td>EDR Infiniband</td>
</tr>
</tbody>
</table>

**High Memory Nodes**

Terremoto's 10 high memory nodes have 768 GB of memory each. They are otherwise identical to the standard nodes.

**GPU Nodes**

Terremoto has 8 GPU nodes, each with two Nvidia V100 GPU modules. They are otherwise identical to the standard nodes.

**Storage**

430 TB of GPFS parallel file system storage is used for scratch space and home directories.

**Network**

10 Gb/s ethernet connections to the Internet from login nodes and transfer node. 100 Gb/s EDR Infiniband connection between compute nodes.

**Scheduler**
Terremoto uses the **Slurm** scheduler to manage jobs.

## Fair share

Resource allocation on our cluster is based on each group's contribution to computing cores. The Slurm scheduler uses fair share targets and historical resource utilization to determine when jobs are scheduled to run. Also, within-group priority is based on historical usage such that heavier users will have a lower priority than light users. Slurm uses all of a job's attributes - such as wall time, resource constraints, and group membership - to determine the order in which jobs are run.

Using job data such as walltime and resources requested, the scheduler can start other, lower-priority jobs so long as they do not delay the highest priority jobs. Because it works by essentially filling in holes in node space, backfill tends to favor smaller and shorter running jobs more than larger and longer running ones.

There is no preemption in the current system; a job in the queue will never interrupt or stop a job in run state.