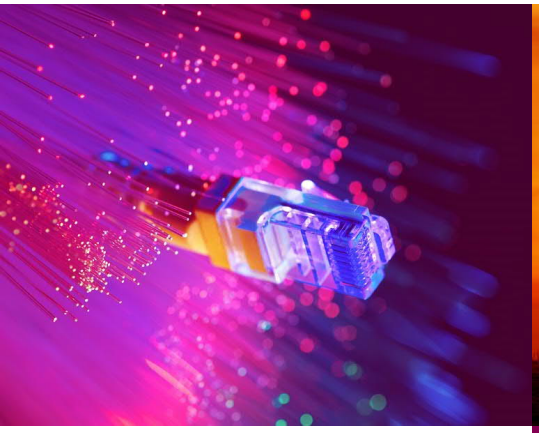
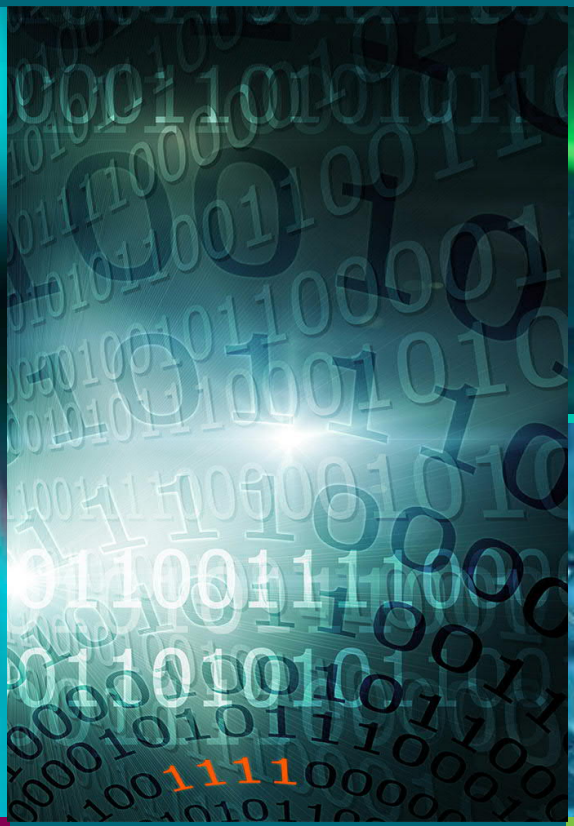
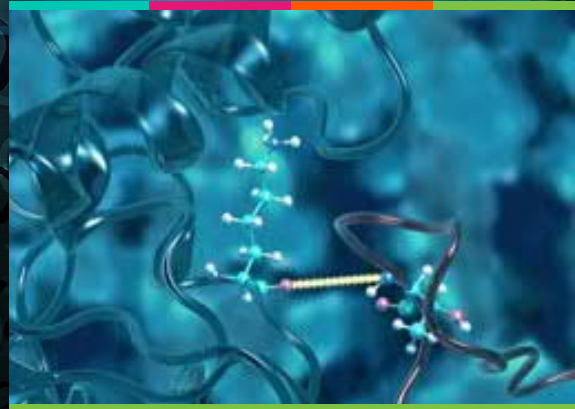


Running Singularity Containers in HPC Environments

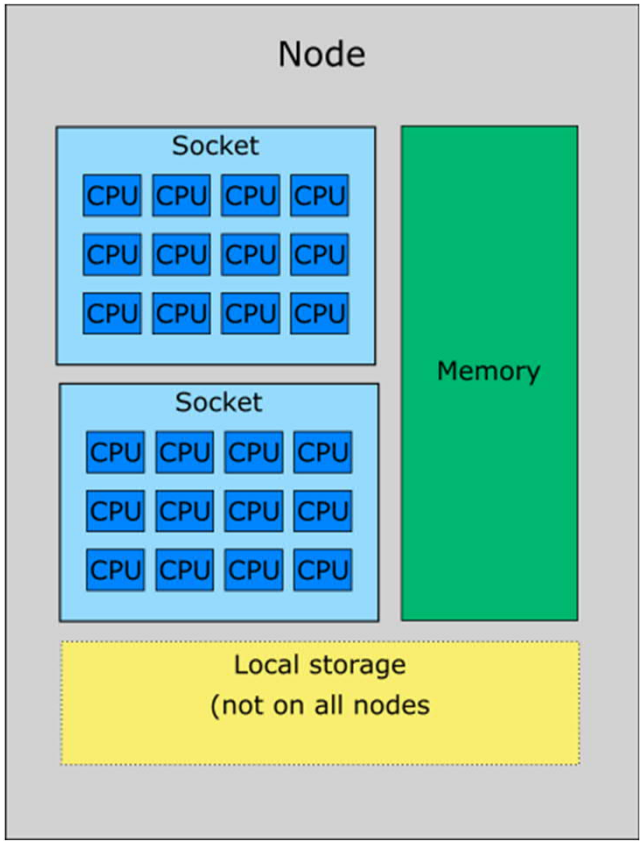
Brief introduction to HPC environments





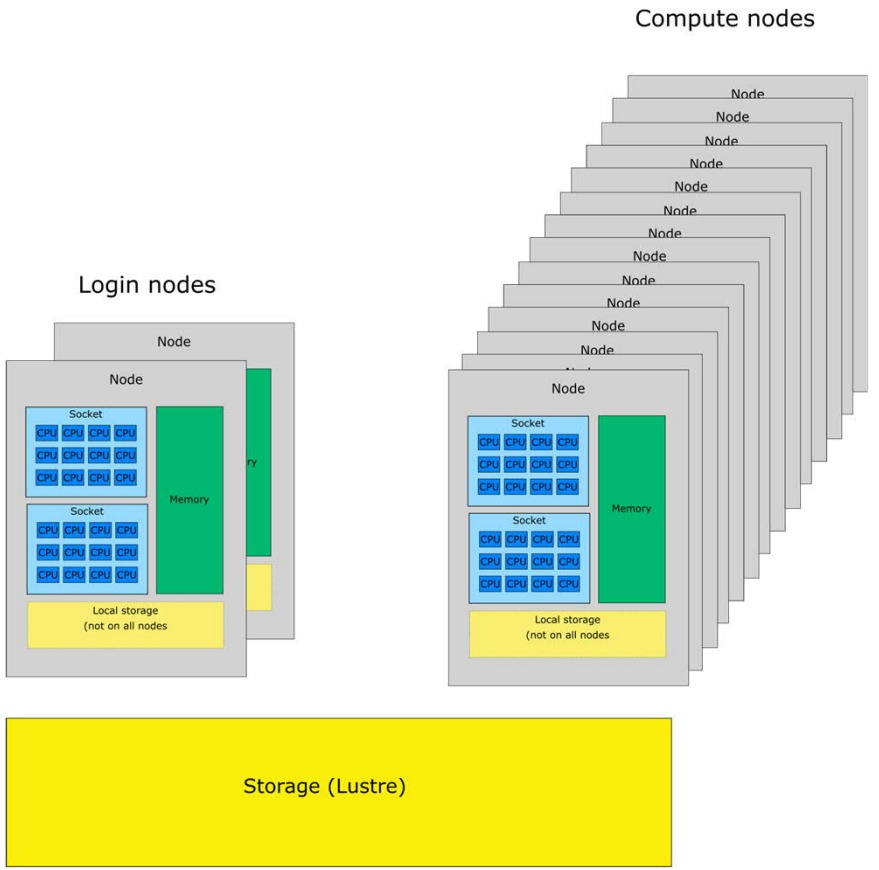
Some notes on vocabulary

computer ~= node
processor ~= socket
core ~= CPU



Cluster systems

- Login nodes are used to set up the jobs
- Jobs are run in the compute nodes
- A batch job system (aka scheduler) is used to run and manage the jobs
  - On this course we use Slurm
  - Other common systems include SGE and Torque/PBS
  - Syntax is different, but basic operation is similar



- To be able to plan your jobs efficiently, you need to familiarize yourself with the available resources
- Each system is different, so check the documentation
- Things to check
  - What batch job system is used
  - What kind of nodes are available?
    - Number of cores
    - Size of memory
    - Extra hardware, e.g GPU, fast local storage
  - What partitions (queues) are available
    - Job sizes, max length, etc
  - Provisioning policy
    - Per core/per node/other

Puhti nodes

	Type	CPU	CPU cores	Memory	Number of nodes
Puhti CPU parition	M	Xeon Gold 6230	2 x 20 cores @ 2.1 GHz	192 GB	532
	L	Xeon Gold 6230	2 x 20 cores @ 2.1 GHz	384 GB	92
	IO	Xeon Gold 6230	2 x 20 cores @ 2.1 GHz	384 GB + 3.2 TB NVMe	40
	XL	Xeon Gold 6230	2 x 20 cores @ 2.1 GHz	768 GB	12
	BM	Xeon Gold 6230	2 x 20 cores @ 2.1 GHz	1.5TB	6
Puhti-AI GPU partition	GPU	Xeon Gold 6230 4 x V100 32 GB	2 x 20 cores @ 2.1 GHz	384 GB (Host) 128 GB (GPUs) 3.2 TB NVMe	80

Puhti CPU partitions

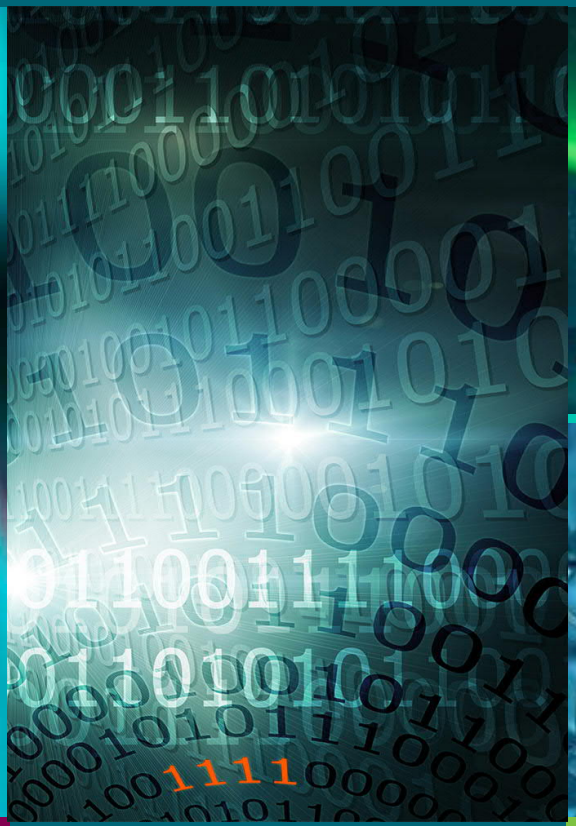
Partition	Time limit	Max tasks	Max nodes	Node types	Max memory	Max local storage
test	15 min	80	2	M	382 GB	
interactive	7 days	1	2	IO	16 GB	160 GB
small	3 days	40	1	M, L, IO	382 GB	3600 GB
large	3 days	4000	100	M, L, IO	382 GB	3600 GB
longrun	14 days	40	1	M, L, IO	382 GB	3600 GB
hugemem	3 days	160	4	XL, BM	1534 GB	
hugemem_longrun	14 days	40	1	XL, BM	1534 GB	

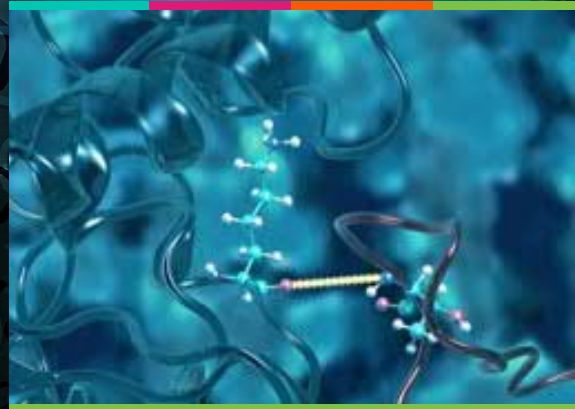
Puhti GPU partitions

Partition	Time limit	Max GPUs	Max nodes	Node types	Max memory	Max local storage
gputest	15 min	8	2	GPU	382 GB	3600 GB
gpu	3 days	80	20	GPU	382 GB	3600 GB

Note that for each GPU, you should reserve at most 10 cores/task.

Running Singularity containers in Puhti





Singularity in Puhti

- Singularity installed only in compute nodes
  - Singularity jobs need to run as batch jobs or with **`sinteractive`**
  - No need to load a module
- Users can run their own containers
- Some CSC software installations provided as containers
  - See software pages for details

- Documentation (under construction):
  - https://docs.csc.fi/computing/containers/run-existing/

singularity_wrapper

- Some software available as CSC installed singularity containers
- Some of these come with helper script singularity_wrapper
  - Takes care of --bind commands, sets path for correct image file, set necessary flags (e.g. --nv) etc
  - Instead of e.g.

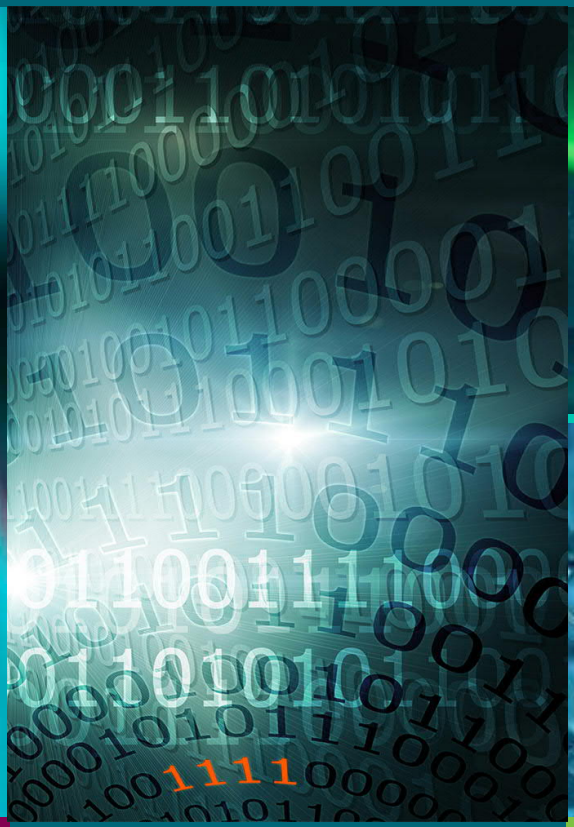
```
srun python3 myprog <options>
```

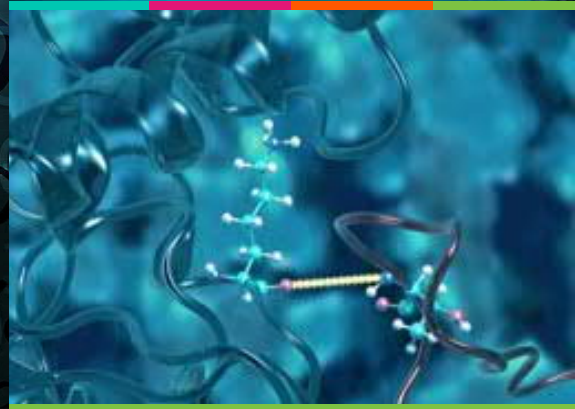
  You can simply do:

```
srun singularity_wrapper exec python3 myprog <options>
```

- See software documentation

Example batch job





```
#!/bin/bash
#SBATCH --job-name=myprog-test
#SBATCH –account=project_12345
#SBATCH --output=output_%j.txt
#SBATCH --error=errors_%j.txt
#SBATCH --time=02:00:00
#SBATCH --ntasks=1
#SBATCH --nodes=1
#SBATCH --cpus-per-task=8
#SBATCH --mem=16G
#SBATCH --partition=small
#

c
```

`#!/bin/bash`

- Tells the computer this is a script that should be run using bash shell
- Everything starting with ”#SBATCH” is passed on to the batch job system
- All #SBATCH lines should be grouped together atthe beginning of the script
  - Comments are allowed, but there should not be empty lines or any other non-comment line
- Everything starting with ”# ” is considered a comment
- Everything else is executed as a command

`#SBATCH --job-name=myprog-test`

- Sets the name of the job
- Job names can be used to manage jobs, but unlike jobids they are not necessarily unique, so care should be taken
  - E.g

    ```
    scancel –n bowtie2
    ```

- By default `squeue`, only shows 8 first characters of job names
  - Can be controlled with `--format` option, e.g:

    ```
    squeue --format="%.18i %.9P %.30j %.8u %.2t %.10M %.6D %R"
    ```

#SBATCH --account=<project>

System specific
Required for CSC systems

- Billing project needs to be specifed in the batch job script
- If omitted, job submission will fail
- Use project name. Project names typically look something like ”project_12345”
- You can check the projects you are a member of with command: `id`

```
uid=10004231(training027) gid=10004155(4training)
groups=10004155(4training),2000745(project_2000745)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

- In CSC systems you can also use command csc-workspace
  - Or check web portal: https://my.csc.fi/myProjects

```
#SBATCH --output=output_%j.txt
#SBATCH --error=errors_%j.txt
```

Optional
–but highly recommended

- When running a command interactively, text output from the command to the shell is delivered via the stdout (standard out) stream. Error messages from the command are sent through the stderr (standard error) stream.
- When running a batch, job these have to be saved to a file. In puhti both stdout and stderr are directed to file ”slurm-<jobid>.out” by default.
- You can choose to save them separately by specifying options `--output` and `–error`
  - `%j` is replaced with the job id number in the actual file name
- What gets written to stdout and stderr depends on the program. If you are unfamiliar with the program, it’s always safest to capture both

`#SBATCH --time=02:00:00`

- Time reserved for the job in hh:mm:ss
- When the time runs out the job will be terminated!
- With longer reservations the job might spend longer in the queue
- Maximum time determined by the queue

```
#SBATCH –-ntasks=1
#SBATCH --nodes=1
#SBATCH --cpus-per-task=8
```

- In this case we are running a shared memory program. It must run inside one node, so we specify:
  - 1 task (`--ntasks`)
  - 1 node (`--nodes`)
  - 8 cores (`--cpus-per-task`)
- For a MPI program we would not need to run inside one node, so we might specify something like:
  - `#SBATCH --ntasks=36`
- Check software documentation
  - Many bioinformatics software can not utilize more than one core
  - Some can use threads and run as a shared memory job
  - Only very few utilize MPI

`#SBATCH --mem=16G`

- The amount of memory reserved for the job
- For MPI jobs use `--mem-per-cpu`
- For shared memory (OpenMP) jobs --mem is easiest
  - `--mem-per-cpu` can be used, but remember to adjust if changing core number
- Keep in mind the specifications for the nodes. Jobs with impossible requests are rejected
- If you reserve too little memory the job will fail
- If you reserve too much memory, your job will spend much longer in queue

#SBATCH --partition=small

- The queue (partition) the job should be submitted to
- You can check the available queues with command

```
sinfo [-p <partition_name>]
```

or

```
scontrol show partition [<partition_name>]
```

Additional resources

System specific
These examples for Puhti

- Local storage:

  `#SBATCH --gres=nvme:<local_storage_space_per_node>`

  - Environment variable $LOCAL_SCRATCH
  - Make sure to copy the results at then end of the jobs. The local storage is emptied after the job has finished
  - Can be helpful when running heavily I/O bound applications
  - Most nodes have no local storage
  - Available in `small`, `large` and `longrun` partitions

- GPUs

  `#SBATCH --gres=gpu:v100:<number_of_gpus_per_node>`

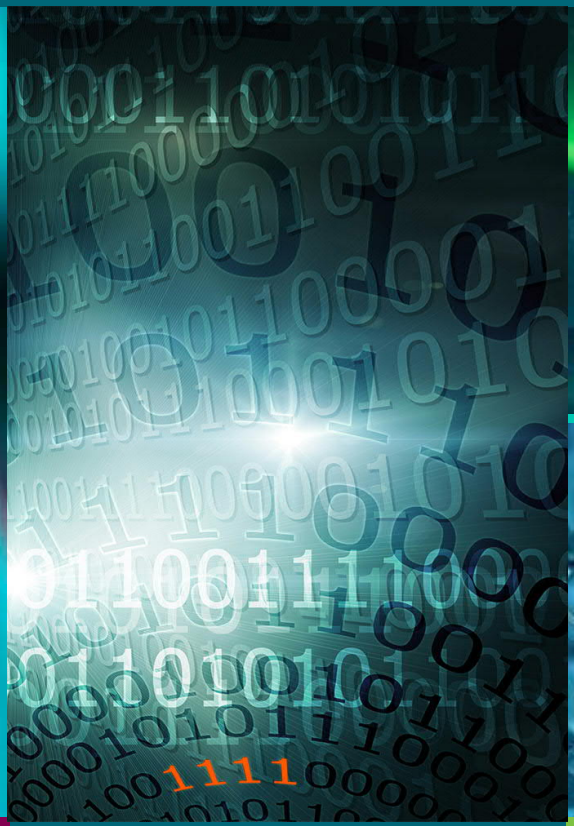
  - The --gres reservation is on a per node basis. There are 4 GPUs per GPU node
  - Available in gpu and gputest partitions
  - Remember to add option `--nv` to Singularity command line

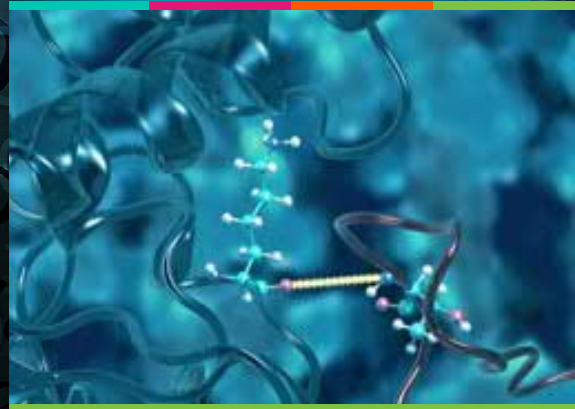
```
singularity exec /path/to/myimage.sif myprog <options>
```

- By default the working directory is the directory where you submitted the job
  - If you include a `cd` command, make sure it points to correct directory
- Command syntax depends on the software
  - It's not enough to reserve the cores: Also remember to tell the program to use them!
  - See application documentation for correct syntax
  - You can use system variable $SLURM_CPUS_PER_TASK
- For MPI jobs, remember to use srun

```
srun mympiprog <options>
```

Managing batch jobs





Submitting and cancelling jobs

- The script file is submitted with command

```
sbatch batch_job.file
```

- sbatch options are usually listed in the batch job script, but they can also be specified on command line, e.g.

```
sbatch --job-name=test2 --time=00:05:00 batch_job_file.sh
```

- Job can be deleted with command

```
scancel <jobid>
```

Queues

- The job can be followed with command squeue:

`squeue`	(shows all jobs in all queues)
`squeue –p <partition>`	(shows all jobs in single queue (partition))
`squeue –u <username>`	(shows all jobs for a single user)
`squeue –j <jobid>`	(shows status of a single job)

- To estimate the start time of a job in queue

```
scontrol show job <jobid>
```

  - row "StartTime=..." gives an estimate on the job start-up time, e.g.

```
StartTime=2019-11-12T19:46:44 EndTime=Unknown
```

Available queues

- You can check available queues on each machine with command:

```
sinfo -l
```

```
PARTITION       AVAIL  TIMELIMIT   JOB_SIZE ROOT OVERSUBS     GROUPS  NODES       STATE NODELIST
small*             up 3-00:00:00          1   no       NO        all     41     drained r17c[01-24],r18c[13-28,32]
small*             up 3-00:00:00          1   no       NO        all     50       mixed
r01c[07,18],r02c[14,21,40],r03c[24,34,39,47],r04c[05,18,21,23,28,39,47],r05c[10-
11,30,32,43,47],r06c[02,06,09,19,23,28,30,34,37,46,50,58],r07c[09,24,30,33],r13c[15,19,31,37],r14c[10-
11],r15c[19,44],r16c[08,25],r18c[33-34]
small*             up 3-00:00:00          1   no       NO        all    567   allocated r01c[01-06,08-17,19-48],r02c[01-13,15-
20,22-39,41-48],r03c[01-23,25-33,35-38,40-46,48],r04c[01-04,06-17,19-20,22,24-27,29-38,40-46,48],r05c[01-09,12-29,31,33-42,44-
46,48-64],r06c[01,03-05,07-08,10-18,20-22,24-27,29,31-33,35-36,38-45,47-49,51-57,59-64],r07c[07-08,10-23,25-29,31-32,34-
56],r13c[01-14,16-18,20-30,32-36,38-48],r14c[01-09,12-48],r15c[01-18,20-43,45-48],r16c[01-07,09-24,26-48],r17c[25-48],r18c[01-
12,29-31,35-48]
..
..
```

Available nodes

- You can check available nodes in each queue with command:

```
sjstat -c
```

```
Scheduling pool data:
-------------------------------------------------------------
Pool        Memory  Cpus  Total Usable   Free  Other Traits
-------------------------------------------------------------
small*     382000Mb    40     92     92      0  type_l
small*     190000Mb    40    526    485      0  type_m
small*     382000Mb    40     40     40      0  type_io
large      382000Mb    40     92     92      0  type_l
large      190000Mb    40    526    485      0  type_m
large      382000Mb    40     40     40      0  type_io
test       190000Mb    40      6      6      4  type_m
longrun    382000Mb    40     92     92      0  type_l
longrun    190000Mb    40    526    485      0  type_m
longrun    382000Mb    40     40     40      0  type_io
fmi        190000Mb    40    238    236    188  type_m
fmitest    190000Mb    40      2      2      2  type_m
hugemem    764000Mb    40     12     12      0  type_xl
hugemem   1532000Mb    40      6      6      4  type_bigmem
hugemem_l  764000Mb    40     12     12      0  type_xl
hugemem_l 1532000Mb    40      6      6      4  type_bigmem
gputest    382000Mb    40      2      2      1  type_gpu
gpu        382000Mb    40     78     74      1  type_gpu
```

Most frequently used SLURM commands

Command	Description
`srun`	Run a parallel job.
`salloc`	Allocate resources for interactive use.
`sbatch`	Submit a job script to a queue.
`scancel`	Cancel jobs or job steps.
`sinfo`	View information about SLURM nodes and partitions.
`squeue`	View information about jobs located in the SLURM scheduling queue
`smap`	Graphically view information about SLURM jobs, partitions, and set configuration parameters
`sjstat`	display statistics of jobs under control of SLURM (combines data from sinfo, squeue and scontrol)
`scontrol`	View SLURM configuration and state.
`sacct`	Displays accounting data for batch jobs.