What MARCC Does

- **Maryland**
- **Advanced**
- **Research**
- **Computing**
- **Center**
Slides available online

- [www.marcc.jhu.edu/training](http://www.marcc.jhu.edu/training)

- MARCC HELP
- marcc-help@marcc.jhu.edu
  - Include as much information as possible, for example:
    - The jobid of the job with problems
    - Full path to the batch submission script
    - Any specific error messages
    - If possible a snapshot with errors

- Frequently Asked Questions
- [https://www.marcc.jhu.edu/getting-started/faqs/](https://www.marcc.jhu.edu/getting-started/faqs/)
High Performance Computing Analogy

Research Project
Ingredients/Recipe
Scientific Applications

Oven = MARCC
MARCC/Bluecrab
Model & Funding

• Grant from the State of Maryland to Johns Hopkins University to build an HPC/big data facility.
• Building, IT stack and Networking
• Operational cost covered by 5 schools:
  • Krieger School of Arts & Sciences (JHU)
  • Whiting Schools of Engineering (JHU)
  • School of Medicine (JHU)
  • Bloomberg Schools of Public Health (JHU)
  • University of Maryland at College Park (UMCP)
Definitions

- **Public Interface**: Front end connecting to Master Node(s)
- **Private Interface**: Back end connecting to Storage Path
- **Computer Nodes**: Connections to Public Interface
- **Storage Path**: Connection to Private Interface
- **NAS**: Accessible via Ethernet or SAN
- **Common FS**: May be connected via Ethernet or SAN
- **FC SAN**: (or iSCSI)

**Multi-core Processor**
- **Core 1**: Individual Memory
- **Core 2**: Individual Memory
- **Core 3**: Individual Memory
- **Core 4**: Individual Memory
- **Shared Memory**
- **Bus Interface**
- **Off-Chip Components**
- **Chip Boundary**
# Definitions

<table>
<thead>
<tr>
<th>Cluster (High Performance Compute Cluster)</th>
<th>Aggregation of servers with high speed connectivity and file systems attached to the servers.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Central Processing Unit aka “Processor”</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphical Processing Unit</td>
</tr>
<tr>
<td>Node (login/compute/management/others)</td>
<td>A server with some amount of memory, cores, CPUs/GPUs</td>
</tr>
<tr>
<td>Core</td>
<td>a “processing unit” within a node (24 our 28 cores)</td>
</tr>
<tr>
<td>Memory/RAM</td>
<td>Amount of memory per core and node (128/96)</td>
</tr>
<tr>
<td>File system</td>
<td>Storage attached to the cluster</td>
</tr>
<tr>
<td>Network</td>
<td>Connectivity between nodes and file systems/communication</td>
</tr>
<tr>
<td>Software</td>
<td>Scientific applications (Python, Matlab, Samtools, Gaussian)</td>
</tr>
</tbody>
</table>
## Compute Nodes

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Type</th>
<th>Description</th>
<th>Total No cores</th>
<th>TFLOPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>648</td>
<td>Regular compute nodes</td>
<td>Haswell 24-core 128GB RAM</td>
<td>15,552</td>
<td>622</td>
</tr>
<tr>
<td>50</td>
<td>Large Mem nodes</td>
<td>Ivy Bridge 48-core 1024GB RAM</td>
<td>2,400</td>
<td>57.6</td>
</tr>
<tr>
<td>48</td>
<td>GPU nodes</td>
<td>Haswell 24-core, 2 Nvidia K80s</td>
<td>1,152</td>
<td>225.5</td>
</tr>
<tr>
<td>-</td>
<td>Lustre</td>
<td>2 PetaByte File system</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>ZFS</td>
<td>13 TB (formatted)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>Regular compute nodes</td>
<td>Broadwell, 28-core 128GB RAM</td>
<td>1,344</td>
<td>55.91</td>
</tr>
<tr>
<td>24</td>
<td>GPU nodes</td>
<td>Broadwell, 28-core 2 Nvidia K80s</td>
<td>672</td>
<td>117.75</td>
</tr>
<tr>
<td>4</td>
<td>GPU P100</td>
<td>Broadwell, 28-core plus 2 P100 per node</td>
<td>112</td>
<td>4.65 + 4/7/gpu</td>
</tr>
<tr>
<td>2</td>
<td>GPU V100</td>
<td>64-core and 28-core</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Condo</td>
<td>Haswell 24-core 128GB RAM</td>
<td>672</td>
<td>26.88</td>
</tr>
<tr>
<td>8</td>
<td>Condo</td>
<td>Broadwell, 28-core 128GB RAM</td>
<td>224</td>
<td>9.32</td>
</tr>
<tr>
<td>52</td>
<td>Condo</td>
<td>Skylake Gold 6126 24 cores</td>
<td></td>
<td>1152</td>
</tr>
<tr>
<td></td>
<td>Total Resources as of 2/22/2019</td>
<td></td>
<td>+23,300 cores</td>
<td>1.5+ PFLOPs</td>
</tr>
</tbody>
</table>

+23,300 cores | 1.5+ PFLOPs
HPC Resources & Model

• Approx 21,120 cores and 15 PB storage

KSAS: 13.4 M Quarter

WSE: 13.4 M Quarter

SOM: 6.8 M Quarter

BSPH: 2.6 M Quarter

UMCP: 6.4 M Quarter

Reserve: 2.0 M/Q
Allocations

• Deans requested applications from all faculty members
• Allocations granted according to available resources

• http://marcc.jhu.edu/request-access/marcc-allocation-request/
Remarks

• **MARCC is free of cost to PIs.** The schools pay for the operations.
• Authentication: via Two-factor authentication.
• Open-data or any kind of confidential data. dbGaP is fine, in most cases.
• Secure Research Environment (MSE) for HIPAA data.
• If additional resources (allocation) needed plan to add a condo (compute nodes).
Storage

```bash
lrwxrwxrwx 1 root root 14 Jun 16 2017 data -> /data/wojcie1
```

```bash
drwxr-xr-x 3 browan@jhu.edu wojcie1 3 Nov 14 2017 intel
```

```bash
drwxr-xr-x 2 browan@jhu.edu wojcie1 3 Nov 9 2017 privatemodules
```

```bash
drwxr-xr-x 141 browan@jhu.edu wojcie1 141 Mar 3 2018 Rpackages
```

```bash
lrwxrwxrwx 1 root root 30 Jun 16 2017 scratch -> /scratch/users/browan1@jhu.edu
```

```bash
lrwxrwxrwx 1 root root 24 Jun 16 2017 work -> /scratch/groups/wojcie1
```

```bash
lrwxrwxrwx 1 root root 23 Oct 4 2017 work2 -> /scratch/groups/pzandil
```

## Storage

<table>
<thead>
<tr>
<th>Directory</th>
<th>Quota</th>
<th>Backup (cost)</th>
<th>Additional storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>$HOME</td>
<td>20 GBytes on ZFS</td>
<td>YES (no cost)</td>
<td>NO</td>
</tr>
<tr>
<td>~/scratch</td>
<td>1 TB per group on Lustre, user access</td>
<td>NO</td>
<td>YES (&gt;10TB, Vice Dean)</td>
</tr>
<tr>
<td>~/work</td>
<td>shared quota with ~/scratch, group access</td>
<td></td>
<td></td>
</tr>
<tr>
<td>~/data</td>
<td>1 TB default per PI, up to 10 TB per group on ZFS, request MARCC</td>
<td>YES (no cost)</td>
<td>N/A</td>
</tr>
<tr>
<td>~/work-zfs</td>
<td>50 TB per group on ZFS, request Vice Dean</td>
<td>YES ($40/TB/yr)</td>
<td>$40/TB/yr + backup cost</td>
</tr>
<tr>
<td>~/project</td>
<td>&lt; 6 months, upon request (Vice Dean)</td>
<td>NO</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Temporary files

• Temporary files go in ~/scratch
• Please do **not** use /tmp or /dev/shm
• If needed, please clean files after job is completed
• Please if at all possible do not do heavy I/O to “data”. Use scratch/work
Connecting

• Windows: Putty, bash, XSHELL
• Mac: terminal, XQuartz,
• VNC (limited); OpenOnDemand (OOD)
• `ssh [-YX] gateway2.marcc.jhu.edu -l userid`
• `ssh -Y login.marcc.jhu.edu -l userid`
1. Use Filezilla  
   https://www.marcc.jhu.edu/getting-started/faqs/

2. ssh dtn.marcc.jhu.edu -l userid

3. Use “aspera”  (FAQ)

4. Use globus connect
   • Download client or use website
   • Create end point (if needed)
   • Authenticate using JHU single sign-on
sshfs Mounts (Basic)

• Fuse/sshfs is now enabled on the cluster
• It follows the two factor authentication protocol
• Check with your local IT person to find out how to mount different file systems or MARCC’s web site for an example:
  • https://www.marcc.jhu.edu/getting-started/basic/
MARCC manages software availability using the “environment modules” (Lmod from TACC)

- module -- help
- module avail
- module spider python
- module list (ml)
- module load gaussian
More on Modules

• Module “load” changes the user’s path to prepend the package being loaded to the user’s environment.
• Example: python
  • >which python
  • /usr/bin/python
  • python version that comes with the OS
module spider

[jcombar1@login-node03 ~]$ ml spider python

python:

------------------------------------------------------------------------------------------------------------------------------------------

Versions:
  python/2.7-anaconda
  python/2.7-anaconda53
  python/2.7
  python/3.6-anaconda
  python/3.6
  python/3.7-anaconda
  python/3.7

Other possible modules matches:
  biopython

------------------------------------------------------------------------------------------------------------------------------------------

To find other possible module matches execute:

  $ module -r spider '.*python.*'

------------------------------------------------------------------------------------------------------------------------------------------

For detailed information about a specific "python" module (including how to load the modules) use the module's full name. For example:

  $ module spider python/3.7
module show

[jcombar1@login-node03 ~]$ ml show python/2.7

/software/lmod/modulefiles/apps/python/2.7.lua:

help({[ anaconda - loads the anaconda software & application environment

This adds /software/apps/python/2.7 to several of the environment variables.

After loading, to see system installed packages for this Python
installation please type:
conda list

See MARCC's help page at: https://www.marcc.jhu.edu/getting-started/local-python-packages/])

whatis("loads the Python 2.7 package")
always_load("centos7")
prepend_path("PATH","/software/apps/python/2.7/bin")
prepend_path("LD_LIBRARYPATH","/software/apps/python/2.7/lib")
prepend_path("LD_LIBRARYPATH","/software/apps/python/2.7/lib/python2.7")
prepend_path("LD_LIBRARYPATH","/software/apps/python/2.7/lib/python2.7/site-packages")
Modules Examples

- `[jcombar1@login-node01 ~]$ module load python
- `[jcombar1@login-node01 ~]$ module list
- Currently Loaded Modules:
  - 1) gcc/4.8.2  2) slurm/14.11.03  3) python/2.7.10 [jcombar1@login-node01 ~]
- `$ which python
- `/software/apps/python/2.7
- `$ echo $PATH
- '/software/apps/python/2.7/bin:/software/apps/marcc/bin:/software/centos7/bin:/software/apps/slurm/current/sbin:/software/apps/slurm/current/bin:/software/apps/mpi/openmpi/3.1.3/intel/18.0/bin:/software/apps/compilers/intel/itac/2018.3.022/intel64/bin:/software/apps/compilers/intel/clck/2018.3/bin/intel64:/software/apps/compilers/intel/compilers_and_libraries_2018.3.222/linux/bin/intel64:/software/apps/compilers/intel/compilers_and_libraries_2018.3.222/linux/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/software/apps/compilers/intel/parallel_studio_xe_2018.3.051/bin:/home-0/jcombar1/pdsh/bin:/home-0/jcombar1/bin
#### Module Specific Help for "gaussian/g09"

This is a computational chemistry application.

- Send mail to marcc-help@marcc.jhu.edu
- Request access to the g09 group

---

**web site:** http://www.gaussian.com  
**Manual on line:** http://www.gaussian.com/g_tech/g_ur/g09_help.htm
---

- On MARCC Gaussian 09 runs using threads. It does not use Linda libraries
- Please do not run Gaussian over more than one Node. Follow the example below

---

To run it in batch mode use a script like this one:

```bash
#!/bin/bash
#SBATCH --time=1:0:0
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=8
#SBATCH --partition=shared
#SBATCH --mem=120000MB
###THE ABOVE REQUESTS 120GB RAM
module load gaussian
gscratch=/scratch/users/$USER
mkdir -p $gscratch/$SLURM_JOBID
export GAUSS_SCRDIR=$gscratch/$SLURM_JOBID
date
time g09 water(.com)
```
Job management tool
Queuing system

- MARCC allocates resources to users using a transparent and fair process by means of a “queueing system”.
- SLURM (Simple Linux Universal Resource Manager)
- Open source, adopted at many HPC centers and HHPC (similar environments)
SLURM Commands

- `man slurm`
- [http://marcc.jhu.edu/getting-started/running-jobs/](http://marcc.jhu.edu/getting-started/running-jobs/)
## Queues/Partitions

<table>
<thead>
<tr>
<th>Partition</th>
<th>Default/Max Time</th>
<th>Default/Max Cores</th>
<th>Default/Max Mem</th>
<th>Serial/Parallel</th>
<th>Backfilling</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHARED</td>
<td>1 hr/ 72hr</td>
<td>1/24</td>
<td>5 GB / 120GB</td>
<td>Serial/Parallel</td>
<td>Shared</td>
</tr>
<tr>
<td>UNLIMITED</td>
<td>Unlimited</td>
<td>1/24/48</td>
<td>5 Gb/120GB</td>
<td>Serial/Parallel</td>
<td>Shared</td>
</tr>
<tr>
<td>PARALLEL</td>
<td>1 hr/ 72hr</td>
<td>1/24</td>
<td>5 GB / 120GB</td>
<td>Parallel</td>
<td>Exclusive</td>
</tr>
<tr>
<td>GPUK80/gpup100/</td>
<td>1 hr/ 72hr</td>
<td>6/24</td>
<td>5 GB / 120GB</td>
<td>Serial/Parallel</td>
<td>Shared</td>
</tr>
<tr>
<td>gpuv100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LRGMEM</td>
<td>1 hr/ 72hr</td>
<td>48</td>
<td>21GB / 1024GB</td>
<td>Serial/Parallel</td>
<td>Shared</td>
</tr>
<tr>
<td>Scavenger</td>
<td>Max 6 hr</td>
<td>1/24</td>
<td>5Gb / 120GB</td>
<td>Serial/Parallel</td>
<td>Shared</td>
</tr>
</tbody>
</table>
Partitions/Shared

- May share compute nodes for jobs
- Serial or parallel jobs
- time limit 1hr to 100-hours
- 1 - 24 cores
- one node

- `#SBATCH -N 1`
- `#SBATCH -ntasks-per-node=12`
- `#SBATCH -partition=shared`
Partitions/Unlimited

- Unlimited time !!!
- Jobs that need to run for more than 100-hours
- If the system/node crashes the job will be killed
- one to 24 cores, one or multi node
- serial or parallel

#SBATCH -N n (n 1 or more)
#SBATCH -partition=unlimited
#SBATCH -walltime=15-00:00:00 (fifteen days)
#SBATCH -ntasks-per-node=m
#SBATCH —mem=0 !!!!!!!!!!!!!!!!!!!!
Partitions/Parallel

- Dedicated queue
- exclusive nodes
- single and multi-node jobs
- 1hr to 100 hours
- Parallel jobs only

- `#SBATCH -N 4`
- `#SBATCH -ntasks-per-node=24` (96 cores)
- `#SBATCH -partition=parallel`
- `#SBATCH --mem=0`
Partitions/scavenger

• Must use with qos=scavenger
• #SBATCH - -qos=scavenger
• #SBATCH - -partition=scavenger

• Low priority jobs
• Time maximum 6 hours
• Use only if your allocation ran out
# SLURM Flags

<table>
<thead>
<tr>
<th>Description</th>
<th>FLAG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Script directive</td>
<td><code>#SBATCH</code></td>
</tr>
<tr>
<td>Job Name</td>
<td><code>#SBATCH --job-name=Any-name</code></td>
</tr>
<tr>
<td>Requested time</td>
<td><code>#SBATCH -t minutes</code></td>
</tr>
<tr>
<td></td>
<td><code>#SBATCH -t [days-hrs:min:sec]</code></td>
</tr>
<tr>
<td>Nodes requested</td>
<td><code>#SBATCH -N min-Max</code></td>
</tr>
<tr>
<td></td>
<td><code>#SBATCH --nodes=Number</code></td>
</tr>
<tr>
<td>Number of cores per node</td>
<td><code>#SBATCH --ntasks-per-node=12</code></td>
</tr>
<tr>
<td>Number of cores per task</td>
<td><code>#SBATCH --cpus-per-task=2</code></td>
</tr>
<tr>
<td>Mail</td>
<td><code>#SBATCH --mail-type=end</code></td>
</tr>
<tr>
<td>User’s email address</td>
<td><code>#SBATCH --mail-user=userid@jhu.edu</code></td>
</tr>
<tr>
<td>Memory size</td>
<td>`#SBATCH --mem=[mem</td>
</tr>
<tr>
<td>Job Arrays</td>
<td><code>#SBATCH --array=[array_spec]</code></td>
</tr>
<tr>
<td>Request specific resource</td>
<td><code>#SBATCH --constraint=&quot;XXX&quot;</code></td>
</tr>
<tr>
<td>Description</td>
<td>Variable</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>JobID</td>
<td>$SLURM_JOBID</td>
</tr>
<tr>
<td>Submit Directory</td>
<td>$SLURM_SUBMIT_DIR</td>
</tr>
<tr>
<td>Submit Host</td>
<td>$SLURM_SUBMIT_HOST</td>
</tr>
<tr>
<td>Node List</td>
<td>$SLURM_JOB_NODELIST</td>
</tr>
<tr>
<td>Job Array Index</td>
<td>$SLURM_ARRAY_TASK_ID</td>
</tr>
</tbody>
</table>

> printenv | grep SLURM
SLURM Scripts

```bash
# cp -r /scratch/public/scripts . (Copy directory)

#!/bin/bash
#SBATCH --job-name=MyJob
#SBATCH --time=8:0:0
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=12
#SBATCH --mail-type=end
#SBATCH - -mail-user=userid@jhu.edu
#SBATCH - - partition=shared
module load mvapich2/gcc/64/2.0b  #### load mvapich2 module
time mpiexec ./code-mvapich.x > OUT-24log
```
Running Jobs

- `sbatch` (qsub) script -name
- `squeue` (qstat -a) -u userid [jcombar1@jhu.edu]$

```bash
qstat -a -u jcombar1@jhu.edu (sqme)
```

```
login-vmnode01.cm.cluster:

<table>
<thead>
<tr>
<th>Job id</th>
<th>Username</th>
<th>Queue</th>
<th>Name</th>
<th>Req'd</th>
<th>Req'd</th>
<th>Elap</th>
<th>SessID</th>
<th>NDS</th>
<th>TSK</th>
<th>Memory</th>
<th>Time</th>
<th>Use</th>
<th>S</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>jcombar1</td>
<td>shared</td>
<td>MyJob</td>
<td>--</td>
<td>1</td>
<td>12</td>
<td>--</td>
<td></td>
<td></td>
<td></td>
<td>08:00</td>
<td></td>
<td></td>
<td>R 00:00</td>
</tr>
</tbody>
</table>
```
SLURM commands

- `scancel (qdel) jobid`
- `scontrol show job jobID`
- `sinfo`
- `sinfo -p shared`
- `sqme`
- `sacct`
Interactive work

• interact -usage

- usage: interact [-n cores] [-t walltime] [-m memory] [-p queue]
  [-o outfile] [-X] [-f featurelist] [-h hostname] [-g ngpus]
- Interactive session on a compute node
- options:
  - -n cores (default: 1)
  - -t walltime as hh:mm:ss (default: 30:00)
  - -m memory as #[k|m|g] (default: 5GB)
  - -p partition (default: 'def')
  - -o outfile save a copy of the session’s output to outfile (default: off)
  - -X enable X forwarding (default: no)
  - -f featurelist CCV-defined node features (e.g., 'e5-2600'), combined with '&' and '|' (default: none)
  - -h hostname only run on the specific node 'hostname'
  - (default: none, use any available node)
Job arrays

•#!/bin/bash
#SBATCH --job-name=job-array
#SBATCH --time=1:0:0
#SBATCH --array=1-240
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --partition=shared
#SBATCH --mem=4.9GB
#SBATCH --mail-type=end
#SBATCH --mail-user=userid@jhu.edu

•# run your job

•echo "Start Job $SLURM_ARRAY_TASK_ID on $HOSTNAME"

•...
GPUs and Interactive Jobs

• `#SBATCH -p gpu --gres=gpu:4`
• `#SBATCH --ntasks-per-node=4`
• `#SBATCH --cpus-per-task=6`

• `interact -p debug -g 1 -n 1 -c 6`
Compilers/Compiling

- Intel compilers
- module list
- ifort -O3 -openmp -o my.exe my.f90
- icc -g -o myc.x myc.c
- GNU
- gfortran -O4 -o myg.f90
- gcc -O4 -o myc.x myc.c
- PGI Compilers (ml pgi )
- pgcc -help
MPI jobs

- module spider mpi
- module load mvapich2
- mpif90 or mpicc code(.f90 or .c)
- mpiexec code.x (within compute node)
- USE mpiicc or mpif90 (Intel-mpi)
- mpif90 and mpicc (use gcc)
Warning

• No refunds so make sure you are using MARCC resources effectively
Information

- marcc-help@marcc.jhu.edu
- Web site  marcc.jhu.edu