Introduction to
High Performance Computing (HPC)
on Lewis and Clark clusters

Research Computing Support Services (RCSS)
http://docs.rnet.missouri.edu

RCSS CIE team
Ashkan Mirzaee, Asif Magdoom
Brian Marxkors, Christina Roberts
Predrag Lazic
A supercomputer is a computer with a high level of performance as compared to a general-purpose computer.

Purpose: massive parallelization because life is too short!

The world's fastest 500 supercomputers run Linux-based operating systems.
Lewis and Clark

Lewis

- A large-scale cluster for requesting high amount of resources
- Great for parallel programming
- GPU resources
- No cost for MU members for general usage
- Investment option is available to receive more resource (more fairshare)

Clark

- Great for learning and teaching
- No need for registration and it is available to all MU members by MU username and PawPrint
- Usually less crowded - receive resources very fast
- No cost for MU members
## Lewis and Clark Partitions

<table>
<thead>
<tr>
<th>Partition Name</th>
<th>Time Limit</th>
<th>Nodes</th>
<th>Cores (per node*)</th>
<th>Cores (total)</th>
<th>Memory in GB (per nodes*)</th>
<th>Processors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interactive</td>
<td>4:00:00</td>
<td>4</td>
<td>24+</td>
<td>144</td>
<td>251+</td>
<td>Intel(R) Xeon(R) CPU E5-2695 v2 @ 2.40GHz+</td>
</tr>
<tr>
<td>General</td>
<td>4:00:00</td>
<td>187</td>
<td>24+</td>
<td>5636</td>
<td>122+</td>
<td>Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz+</td>
</tr>
<tr>
<td>BioCompute</td>
<td>2:00:00:00</td>
<td>37</td>
<td>56</td>
<td>2072</td>
<td>509</td>
<td>Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz</td>
</tr>
<tr>
<td>Lewis</td>
<td>2:00:00:00</td>
<td>90</td>
<td>24+</td>
<td>3564</td>
<td>122+</td>
<td>Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz+</td>
</tr>
<tr>
<td>hpc3</td>
<td>2:00:00:00</td>
<td>22</td>
<td>24</td>
<td>1296</td>
<td>122+</td>
<td>Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz</td>
</tr>
<tr>
<td>hpc4</td>
<td>2:00:00:00</td>
<td>37</td>
<td>28</td>
<td>1260</td>
<td>251+</td>
<td>Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz</td>
</tr>
<tr>
<td>hpc4rc</td>
<td>2:00:00:00</td>
<td>34</td>
<td>28</td>
<td>1008</td>
<td>251</td>
<td>Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz</td>
</tr>
<tr>
<td>hpc6</td>
<td>2:00:00:00</td>
<td>61</td>
<td>48</td>
<td>2976</td>
<td>379+</td>
<td>Intel(R) Xeon(R) Gold 6252 CPU @ 2.10GHz</td>
</tr>
<tr>
<td>hpc5</td>
<td>2:00:00:00</td>
<td>35</td>
<td>40</td>
<td>1320</td>
<td>379</td>
<td>Intel(R) Xeon(R) Gold 6138 CPU @ 2.00GHz</td>
</tr>
<tr>
<td>Gpu</td>
<td>2:00:00</td>
<td>16</td>
<td>16+</td>
<td>372</td>
<td>122+</td>
<td>Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz+</td>
</tr>
<tr>
<td>gpu3</td>
<td>2:00:00:00</td>
<td>13</td>
<td>16+</td>
<td>284</td>
<td>122+</td>
<td>Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz</td>
</tr>
<tr>
<td>gpu4</td>
<td>2:00:00:00</td>
<td>3</td>
<td>40+</td>
<td>124</td>
<td>379+</td>
<td>Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz</td>
</tr>
<tr>
<td>Serial</td>
<td>2:00:00:00</td>
<td>1</td>
<td>64</td>
<td>64</td>
<td>1,025</td>
<td>AMD EPYC 7601 32-Core Processor</td>
</tr>
<tr>
<td>Dtn</td>
<td>2:00:00:00</td>
<td>2</td>
<td>16+</td>
<td>36</td>
<td>66+</td>
<td>Intel(R) Xeon(R) CPU X5550 @ 2.67GHz+</td>
</tr>
</tbody>
</table>

* plus sign (+) indicates a mixed environment. The number before the plus represents the minimum
[http://docs.rnet.missouri.edu/policy/partition-policy/](http://docs.rnet.missouri.edu/policy/partition-policy/)
Slurm is a system for cluster management and job scheduling. All RCSS clusters use Slurm (https://slurm.schedmd.com).

- Slurm is a workflow scheduler and has set of tools for submitting and monitoring jobs.
- Slurm is a resource management system and has many tools to find available resources in the cluster.
- All Slurm commands start with letter “s”. In this course we will learn many of them.
- Resource allocation depends on your fairshare i.e. priority in the queue.
Login node

All users connect to Clark and Lewis clusters through the login nodes.

```
[user@lewis4-r630-login-node675 ~]$ 
[user@clark-r630-login-node907 ~]$
```

All jobs must be run using Slurm submitting tools to prevent running on the Lewis login node. Jobs that are found running on the login node will be immediately terminated followed up with a notification email to the user.
# Review

## Unix commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>man</td>
<td>manual</td>
</tr>
<tr>
<td>cd</td>
<td>change directory</td>
</tr>
<tr>
<td>pwd</td>
<td>print working directory</td>
</tr>
<tr>
<td>ls -la</td>
<td>long list of all files</td>
</tr>
<tr>
<td>mkdir</td>
<td>make directory</td>
</tr>
<tr>
<td>cp</td>
<td>copy</td>
</tr>
<tr>
<td>hostname</td>
<td>host name</td>
</tr>
<tr>
<td>nproc</td>
<td>number of processors</td>
</tr>
<tr>
<td>lscpu</td>
<td>list CPU architecture</td>
</tr>
<tr>
<td>free</td>
<td>free memories</td>
</tr>
<tr>
<td>df</td>
<td>disk free</td>
</tr>
<tr>
<td>echo $USER</td>
<td>echo user id</td>
</tr>
<tr>
<td>cat</td>
<td>concatenate files and print output</td>
</tr>
<tr>
<td>top</td>
<td>information about processes</td>
</tr>
<tr>
<td>Tab</td>
<td>tab completion</td>
</tr>
<tr>
<td>clear</td>
<td>clear</td>
</tr>
<tr>
<td>Ctrl + l</td>
<td>clear</td>
</tr>
<tr>
<td>exit</td>
<td>exit</td>
</tr>
<tr>
<td>Ctrl + d</td>
<td>exit</td>
</tr>
<tr>
<td>Ctrl + z</td>
<td>suspend</td>
</tr>
<tr>
<td>fg</td>
<td>return</td>
</tr>
<tr>
<td>Ctrl + z</td>
<td>suspend</td>
</tr>
<tr>
<td>Esc + :w</td>
<td>write</td>
</tr>
<tr>
<td>Esc + :q</td>
<td>quit</td>
</tr>
<tr>
<td>:q!</td>
<td>no change</td>
</tr>
<tr>
<td>cp</td>
<td>copy</td>
</tr>
<tr>
<td>echo $USER</td>
<td>echo user id</td>
</tr>
<tr>
<td>history</td>
<td>history</td>
</tr>
<tr>
<td>hostname</td>
<td>host name</td>
</tr>
<tr>
<td>nproc</td>
<td>number of processors</td>
</tr>
<tr>
<td>lscpu</td>
<td>list CPU architecture</td>
</tr>
<tr>
<td>free</td>
<td>free memories</td>
</tr>
<tr>
<td>df</td>
<td>disk free</td>
</tr>
<tr>
<td>echo $USER</td>
<td>echo user id</td>
</tr>
<tr>
<td>cat</td>
<td>concatenate files and print output</td>
</tr>
<tr>
<td>top</td>
<td>information about processes</td>
</tr>
<tr>
<td>Tab</td>
<td>tab completion</td>
</tr>
<tr>
<td>clear</td>
<td>clear</td>
</tr>
<tr>
<td>Ctrl + l</td>
<td>clear</td>
</tr>
<tr>
<td>exit</td>
<td>exit</td>
</tr>
<tr>
<td>Ctrl + d</td>
<td>exit</td>
</tr>
<tr>
<td>Ctrl + z</td>
<td>suspend</td>
</tr>
<tr>
<td>fg</td>
<td>return</td>
</tr>
<tr>
<td>Ctrl + z</td>
<td>suspend</td>
</tr>
<tr>
<td>Esc + :w</td>
<td>write</td>
</tr>
<tr>
<td>Esc + :q</td>
<td>quit</td>
</tr>
<tr>
<td>:q!</td>
<td>no change</td>
</tr>
</tbody>
</table>

## Text editors

<table>
<thead>
<tr>
<th>Text editor</th>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>emacs -nw</td>
<td>Ctrl-k</td>
<td>kill (cut)</td>
</tr>
<tr>
<td></td>
<td>Ctrl-y</td>
<td>yank (paste)</td>
</tr>
<tr>
<td></td>
<td>Alt-w</td>
<td>copy</td>
</tr>
<tr>
<td></td>
<td>Shift-arrow keys</td>
<td>select</td>
</tr>
<tr>
<td></td>
<td>Ctrl-z</td>
<td>suspend fg return</td>
</tr>
<tr>
<td></td>
<td>Ctrl-x + Ctrl-s</td>
<td>save</td>
</tr>
<tr>
<td></td>
<td>Ctrl-x + Ctrl-c</td>
<td>close</td>
</tr>
<tr>
<td></td>
<td>Ctrl-k</td>
<td>cut</td>
</tr>
<tr>
<td></td>
<td>Ctrl-u</td>
<td>uncut (paste)</td>
</tr>
<tr>
<td></td>
<td>Alt-6</td>
<td>copy</td>
</tr>
<tr>
<td></td>
<td>Alt-m + alt-a</td>
<td>Select</td>
</tr>
<tr>
<td></td>
<td>Ctrl-z</td>
<td>suspend fg return</td>
</tr>
<tr>
<td></td>
<td>Ctrl-o</td>
<td>write out</td>
</tr>
<tr>
<td></td>
<td>Ctrl-x</td>
<td>exit</td>
</tr>
<tr>
<td>nano -z</td>
<td>i</td>
<td>insert</td>
</tr>
<tr>
<td></td>
<td>Esc</td>
<td>dd delete line</td>
</tr>
<tr>
<td></td>
<td>Esc</td>
<td>yy yank (paste)</td>
</tr>
<tr>
<td></td>
<td>Esc u</td>
<td>undo Esc p paste</td>
</tr>
<tr>
<td></td>
<td>Esc</td>
<td>Ctrl + z suspend fg ret.</td>
</tr>
<tr>
<td></td>
<td>Ctrl-o</td>
<td>write out</td>
</tr>
<tr>
<td></td>
<td>Ctrl-x</td>
<td>exit</td>
</tr>
<tr>
<td></td>
<td>Ctrl-x</td>
<td>exit</td>
</tr>
<tr>
<td>vim</td>
<td>i</td>
<td>insert</td>
</tr>
<tr>
<td></td>
<td>Esc</td>
<td>dd delete line</td>
</tr>
<tr>
<td></td>
<td>Esc</td>
<td>yy yank (paste)</td>
</tr>
<tr>
<td></td>
<td>Esc u</td>
<td>undo Esc p paste</td>
</tr>
<tr>
<td></td>
<td>Ctrl + z</td>
<td>suspend fg ret.</td>
</tr>
<tr>
<td></td>
<td>Esc +  :w</td>
<td>write</td>
</tr>
<tr>
<td></td>
<td>Esc + :q</td>
<td>quit :q! no change</td>
</tr>
</tbody>
</table>

## SSH commands

- Connect to Lewis by ssh or:
- Connect to Clark by ssh and:
- Copy training files to your home directory
Cluster info

Scheduling pool data:

<table>
<thead>
<tr>
<th>Pool</th>
<th>Memory</th>
<th>Cpus</th>
<th>Total Usable</th>
<th>Free</th>
<th>Other</th>
<th>Traits</th>
</tr>
</thead>
<tbody>
<tr>
<td>r630-hpc3</td>
<td>122534Mb</td>
<td>24</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>hpc3</td>
<td>122534Mb</td>
<td>24</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>General</td>
<td>122534Mb</td>
<td>24</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

CPUS/NODES(A/I/O/T) count of CPUS/nodes in the form "available/idle/other/total"

S:C:T counts number of "sockets, cores, threads"

The Scheduling data contains information pertaining to the:

- Pool: a set of nodes
- Memory: the amount of memory on each node
- Cpus: the number of cpus on each node
- Total: the total number of nodes in the pool
- Usable: total usable nodes in the pool
- Free: total nodes that are currently free
Users info

**sshare - sacctmgr - df - lfs quota**

sshare -u show your fairshare and accounts
sacctmgr show assoc user=$USER format=acc,user,share,qos,maxj your quality of services (QOS)
groups show your groups
df -h /home/$USER home storage quota
lfs quota -hg $USER /storage/hpc data/scratch storage quota
lfs quota -hg <group-name> /storage/hpc data/scratch storage quota
./userq.py show user’s fairshare, accounts, groups and QOS

- Resource allocation depends on your fairshare. If your fairshare is 0.000000 you have used the cluster more than your fair share and will be de-prioritized by the queuing software
- Users have 5GB at their home directory /home/$USER and 100GB at /storage/hpc/data/$USER
- **Do not** use home directory for running jobs, storing data or virtual environments
- Clark users have 100G on their home storage. The above methods does not apply for Clark
- The RCSS team reserves the right to delete anything in /scratch and /local/scratch at any time for any reason
- There are no backups of any storage. The RCSS team is not responsible for data integrity and data loss. **You are responsible for your own data and data backup**
- Review our storage policy at [http://docs.rnet.missouri.edu/policy/storage-policy/](http://docs.rnet.missouri.edu/policy/storage-policy/)

**Note:** $USER is a default environmental variable that returns your user id, try echo $USER
Job submission

All jobs must be run using `srun` or `sbatch` to prevent running on the Lewis login node.

**srun: requesting resources to run jobs**

- `srun <slurm-options> <software-name/path>`
- `srun --pty /bin/bash` requesting a pseudo terminal of bash shell to run jobs interactively
- `srun -p Interactive --qos interactive --pty /bin/bash` requesting a p.t. of bash shell in Interactive node on Lewis
- `srun -p <partition-name> --mem 16G --pty /bin/bash` req. 4 tasks and 16G memory
- `srun -p <partition-name> -n 4 --mem 16G --pty /bin/bash` req. 4 tasks and 16G memory
- `srun -p <partition-name> -N 1 --nodes-per-node 8 --pty /bin/bash` req. 1 GPU and 1 node for running 8 tasks on Gpu partition

*Slurm by default allocates 1 CPU per tasks

**sbatch: submitting jobs**

- Batch file is a shell script (`#!/bin/bash`) including Slurm options (`#SBATCH`) and computational tasks
- `sbatch <batch-file>` submitting a batch file

After job completion we will receive outputs (`slurm-jobid.out`)
Slurm options

**man srun - man sbatch**

-`p` --partition `<partition-name>`
-`--mem` `<memory>`
-`-n` --ntasks `<number of tasks>`
-`-N` --nodes `<number-of-nodes>`
-`-c` cpus-per-task `<number-of-cpus>`
-`-w` --nodelist `<list-of-node-names>`

**Environmental Variables**

- `$SLURM_JOB_ID`
- `$SLURM_JOB_NAME`
- `$SLURM_JOB_NODELIST`
- `$SLURM_CPUS_ON_NODE`
- `$SLURM_SUBMIT_HOST`
- `$SLURM_SUBMIT_DIR`

**Example**

**test.py**
```python
#!/usr/bin/python3
import os

os.system('""
    echo hostname: $(hostname)
    echo number of processors: $(nproc)
    echo data: $(date)
    echo job id: $SLURM_JOB_ID
    echo submit dir: $SLURM_SUBMIT_DIR
    """

print("Hello world")
```

**jobpy.sh**
```bash
#!/bin/bash

#SBATCH -p hpc3
#SBATCH --n 4
#SBATCH --mem 8G
python3 test.py
```

**srun**
```
srun -p Interactive --qos interactive -n 4 --mem 8G --pty bash
```

**Output**
```
hostname: lewis4-lenovo-hpc2-node282
number of processors: 4
data: Sun Jun 28 13:27:39 CDT 2020
job id: 21437062
submit dir: /home/user/training
Hello world
```
Monitor jobs

sacct -X show your jobs in the last 24 hours
sacct -X -S <yyyy-mm-dd> show your jobs since a date
sacct -X -S <yyyy-mm-dd> -E <yyyy-mm-dd> -s <R/PD/F/CA/CG/CD> show running/pending/failed/cancelled/completing/completed jobs in a period
sacct -j <jobid> show info about the jobid
squeue -u <username> show a user jobs (R/PD/CD) in the queue
squeue -u <username> --start show estimation time to start pending jobs
scancel <jobid> cancel jobs

./jobstat.py <day/week/month/year> show info about running, pending and completed jobs of a user within a time period (default is a week)
Monitor CPU and Memory

```
sacct -j <jobid> -o User,Acc,AllocCPUS,Elaps,CPUPer,TotalCPU,AveDiskRead,AveDiskWrite,ReqMem MaxRSS

info about CPU and virtual memory for completed jobs

seff <jobid> show jobs CPU and memory efficiency

sacct - seff

[jantwc@lewiss-r630-login-node675 -]$ sacct -j 20785018 -o User,Acc,AllocCPUS,Elaps,CPUPer,TotalCPU,AveDiskRead,AveDiskWrite,ReqMem MaxRSS

User: amtwc
Account: general
 AllocCPUS  Elaps   CPUPer  TotalCPU  AveDiskRead  AveDiskWrite  ReqMem  MaxRSS
amtwc       16  00:48:39  12:58:24  01:49.774  66.88M      44.75M      64Gn   216K
```

CPU Efficiency = CPU Utilization / Core-waittime
Core-waittime = Core per node * Elapsed time

```
sstat -srun

sstat <jobid> -o AveCPU,AveDiskRead,AveDiskWrite,MaxRSS info about CPU and memory for running jobs (srun only)
srun --jobid <jobid> --pty /bin/bash attach to a srun/sbatch session and run `top` command to see information about processes

Test

```
emacs -nw test.py
emacs -nw jobpy.sh

Ctrl+x+s to save and Ctrl+x+c to exit
```

```
sbatch jobpy.sh
```

```
sstat <jobid>
srun <jobid> --pty /bin/bash attach to a srun/sbatch session and run `top` command to see information about processes
```

We are using 0% CPU and less than 6MB memory
**Modules**

module avail available modules
module load loaded modules
module show show modules info
module unload unload loaded modules
module list list loaded modules
module purge unload all loaded modules

For example, let’s run R and MATLAB interactively:

**R**

```bash
srun -p Interactive --qos interactive --mem 4G --pty /bin/bash
# srun -p hpc3 --mem 4G --pty bash # for Clark
```

```bash
module load R
module list
  1) R/R-3.3.3
R
```

**MATLAB**

```bash
srun -p Interactive --qos interactive --mem 4G -L matlab
--pty /bin/bash
```

```bash
module load matlab
module list
  1) matlab/matlab-R2020a
matlab -nodisplay
```

**Never** load modules in the login node. It makes login node slow for all users. **Many modules don’t work in the login node.** We can load modules in batch files, for example:

**test.R**

```r
#!/usr/bin/R

for (i in 1:3) {
  cat("Hello world", i,"\n")
}
```

**runr.sh**

```bash
#!/bin/bash

#SBATCH -p hpc3
#SBATCH --mem 4G
module load R
Rscript test.R
```

**sbatch**

```bash
dsbatch runr.sh
```

**Output**

```
Hello world 1
Hello world 2
Hello world 3
```
## What is next:

### Version control

### Job dependencies
- Slurm dependency option (--dependency) [https://slurm.schedmd.com/sbatch.html](https://slurm.schedmd.com/sbatch.html)
- Snakemake [https://snakemake.readthedocs.io/](https://snakemake.readthedocs.io/)

### Virtual Environments

### Software Installation

### Parallel programming
- Scientific languages C/Fortran
  - OpenMP
  - OpenACC for GPU parallelization
  - MPI for massive parallelization
- Python
  - NumPy
  - Numba
  - mpi4py

### Links
- RCSS Documentation [http://docs.rnet.missouri.edu](http://docs.rnet.missouri.edu)
- XSEDE [https://www.xsede.org/for-users/training](https://www.xsede.org/for-users/training)
- Software Carpentry [https://software-carpentry.org/lessons/](https://software-carpentry.org/lessons/)
- HPC Carpentry [https://hpc-carpentry.github.io](https://hpc-carpentry.github.io)
- Data Carpentry [https://datacarpentry.org/lessons/](https://datacarpentry.org/lessons/)
- Cornell Virtual Workshop [https://cvw.cac.cornell.edu/topics](https://cvw.cac.cornell.edu/topics)
- Pittsburgh Supercomputing Center (PSC) [https://www.psc.edu/resources-for-users/training/](https://www.psc.edu/resources-for-users/training/)
- TACC Learning Portal [https://learn.tacc.utexas.edu/course/](https://learn.tacc.utexas.edu/course/)

### Feedback and Questions
mudoitrcss@missouri.edu