Mon, Oct 26

Lecture 07A:

Computer clusters

Practical Bioinformatics (Biol 4220)
Instructor: Michael Landis
Email: michael.landis@wustl.edu

Lecture 7A outline

1. Working with a cluster
2. Using the scheduler
3. Lab 7A overview

Servers and clusters

A ***server*** is a computer with ample resources that are shared among many ***clients***

A ***cluster*** is a group of servers that are configured to distribute tasks that involve large numbers of jobs and/or parallelized jobs

Information Technology (IT) departments manage security, access, maintenance, and expansion of servers and clusters

Why clusters?

Large institutions that rely on computation invest in clusters for many reasons

- massive parallelization of jobs
- large shared storage
- more efficient (less idling)
- economy of scale
- greater hardware capacity
- security benefits

Cluster anatomy

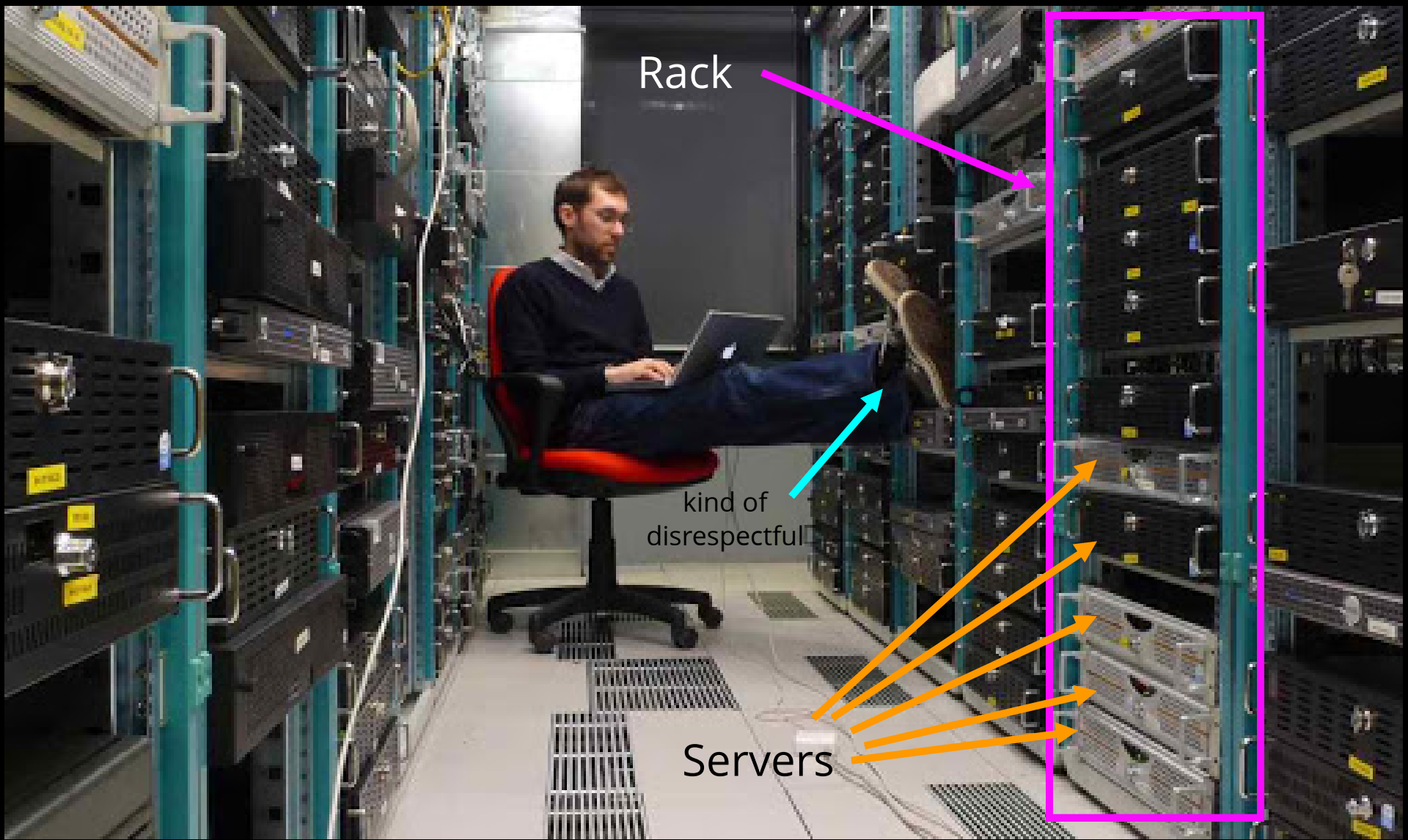
Common cluster features

- Large numbers of ***users***, each with individual needs
- A ***private network*** to secure resources and monitor bandwidth
- ***Nodes*** are servers with well-defined roles, depending on hardware and network access
  - users log-in to and submit jobs through the ***head node***
  - the ***scheduler*** determines when to process each job
  - jobs are processed by the large pool of ***compute nodes***;
  - each compute node generally has 16-64 ***compute cores***, where cores are the units of parallelization
  - some jobs might request nodes with special hardware (e.g. ***GPU nodes***)

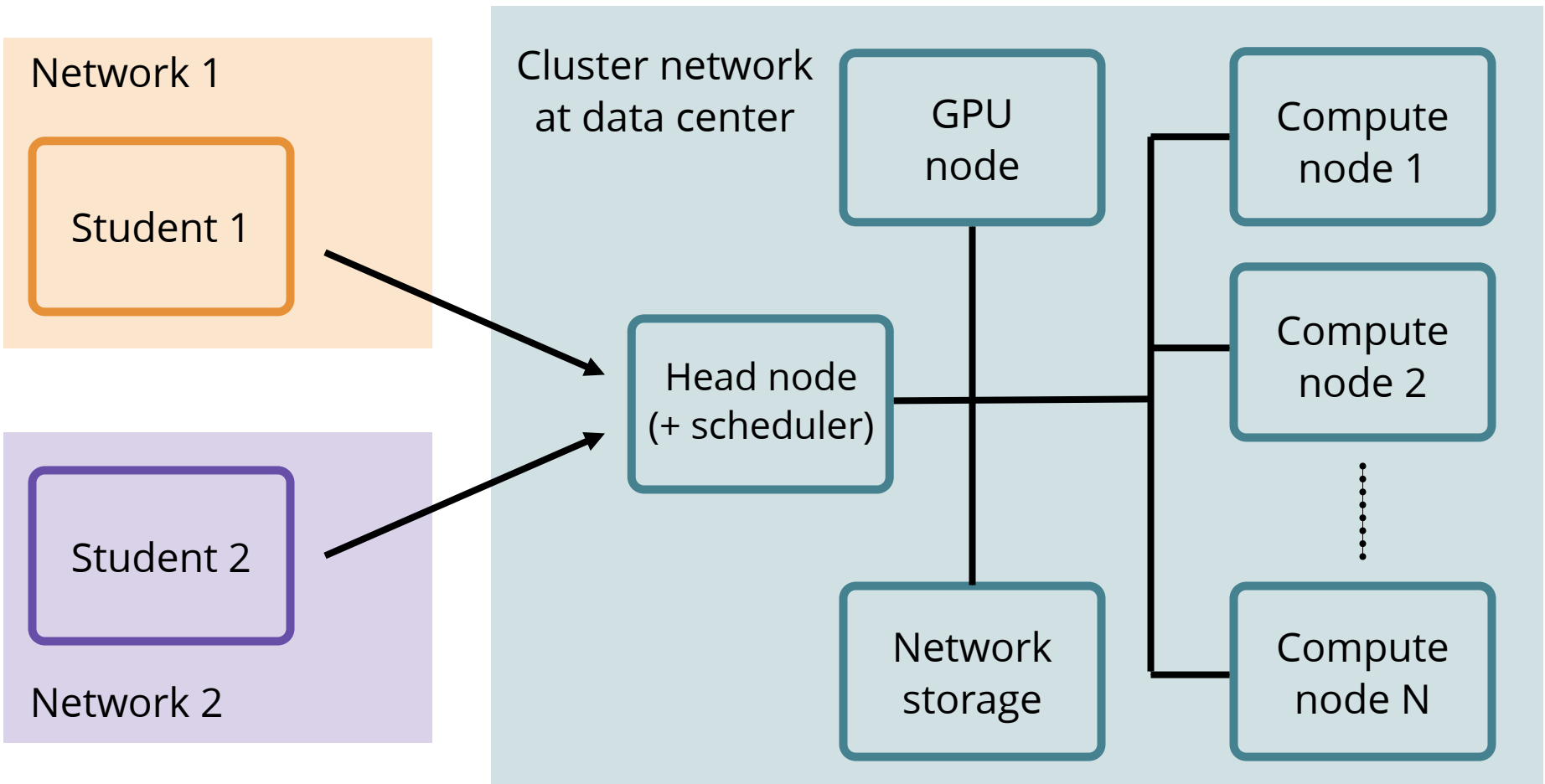
Cluster anatomy

Common cluster features

- All users and nodes share access to ***network storage***
  - ***OS storage*** for e.g. utilities, libraries, home directories; small, fast, permanent, and free
  - ***Scratch storage*** for job output, *etc.* that is periodically deleted; large, slow, temporary, and free
  - ***Persistent storage*** for large irreplaceable datasets; large, slow, permanent, and costly
- Nodes, storage devices, network devices, *etc.* are installed in ***racks*** in ***data centers***, where IT manages network, backup, storage, power, etc.


Rack
kind of
disrespectful
Servers

Cluster anatomy



Five fastest clusters

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148,600.0	200,794.9	10,096
2	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
3	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway, NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
4	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000, NUDT National Super Computer Center in Guangzhou China	4,981,760	61,444.5	100,678.7	18,482
5	**Frontera** - Dell C6420, Xeon Platinum 8280 28C 2.7GHz, Mellanox InfiniBand HDR, Dell EMC Texas Advanced Computing Center/Univ. of Texas United States	448,448	23,516.4	38,745.9	

TFlop/s, 1e12 floating point operations/s

https://www.top500.org/lists/top500/list/2019/11/

WUSTL RIS cluster

- Base system
  - 5,000 Intel Cascade Lake Cores
  - 120 Nvidia Telsa V100 GPUs
  - 300TB DDN high-performance scratch space
  - 100Gbit Mellanox HDR Network
- Batch computing across thousands of CPU and hundreds of GPUs
- Integration with WashU Research Network (WURN) (40Gbit)

https://ris.wustl.edu/services/compute/

WUSTL cluster resources

Useful resources:

- Manual https://confluence.ris.wustl.edu/display/RSUM/RIS+Services+User+Manual
- Knowledgebase https://confluence.ris.wustl.edu/display/ITKB/RIS+IT+Knowledge+Base
- Help desk https://jira.ris.wustl.edu/servicedesk/customer/portal/1/create/101

(note: RIS cluster still in *Early Access* phase)

Using the cluster

Typical workflow for cluster

1. connect to cluster network (*e.g.* VPN)
2. log into cluster through *head node*
3. set up files and profile on cluster
4. reconfigure scripts on cluster to work with programs and filesystem
5. submit jobs to *scheduler*
6. wait for *compute nodes* to run jobs
7. retrieve output from cluster

We'll practice this workflow in lab

Job scheduler

Clusters are endowed with an extensive array of hardware, *e.g.* CPU, GPU, memory, storage

The ***job scheduler*** ensure that resources are shared fairly among users by managing multiple ***job queues***

Users submit their job(s) to a queue, the queue then processes jobs based on ***queue policies***

Job scheduler

Example scenario

- Cluster has 500 cores
- User 1 submits 10x 64-core 10-hour jobs
- User 2 submits 6400x 1-core 1-hour jobs

Scheduler will have more difficulty securing the resources for User 1's jobs than User 2's jobs

The longer it takes to run User 1's jobs, the higher their priority will grow; User 2's jobs will decrease in priority as more jobs are run

Job scheduler

Example scenario

- User 1 submits **10x 64-core 10-hour** jobs
- User 2 submits **6400x 1-core 1-hour** jobs

Scheduler will have more difficulty securing the resources for User 1's jobs than User 2's jobs

- The longer it takes to run User 1's jobs, the higher their priority will grow
- User 2's jobs will decrease in priority as more jobs are run

Job scheduler

Different clusters use different software for job scheduling.
Examples: Slurm, Torque, LSF

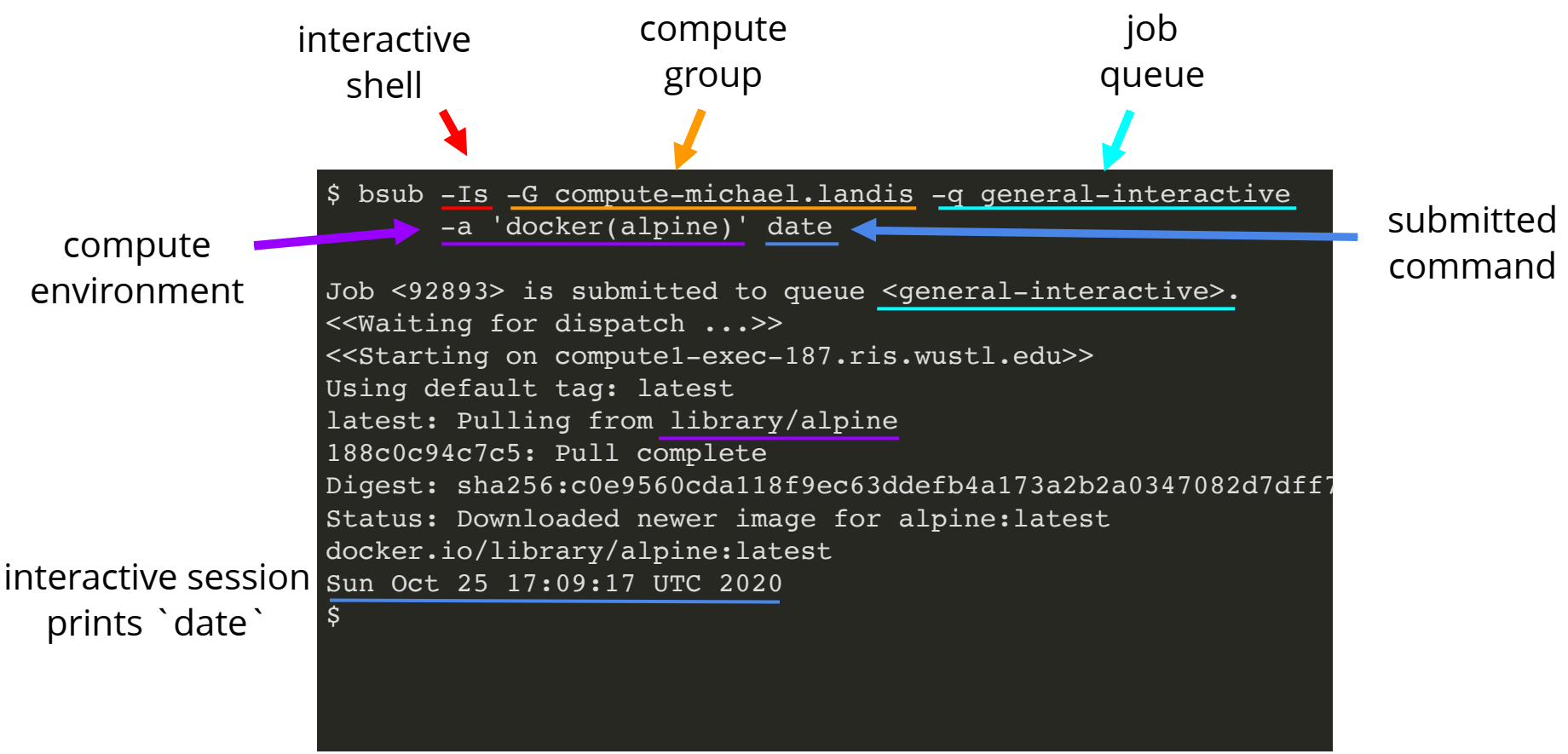
How users interact with the job scheduler varies in terms of syntax, but workflow is fairly consistent across platforms:

- submit jobs
- monitor jobs
- cancel jobs

The RIS cluster uses LSF:
https://www.ibm.com/support/knowledgecenter/SSWRJV_10.1.0/lsf_welcome/lsf_kc_cmd_ref.html

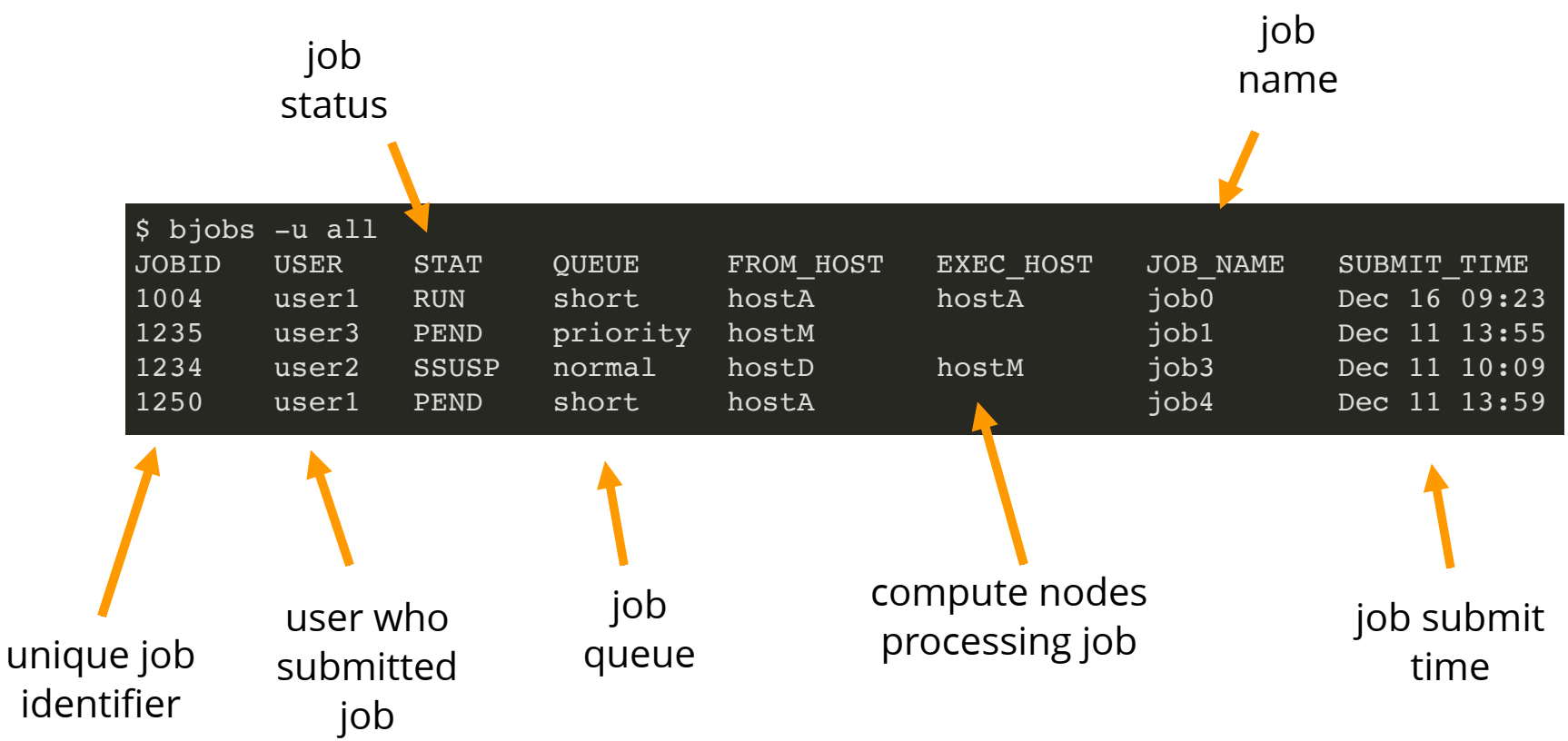
bsub

Submits job to a queue



bjobs

Lists all queued jobs



bkill

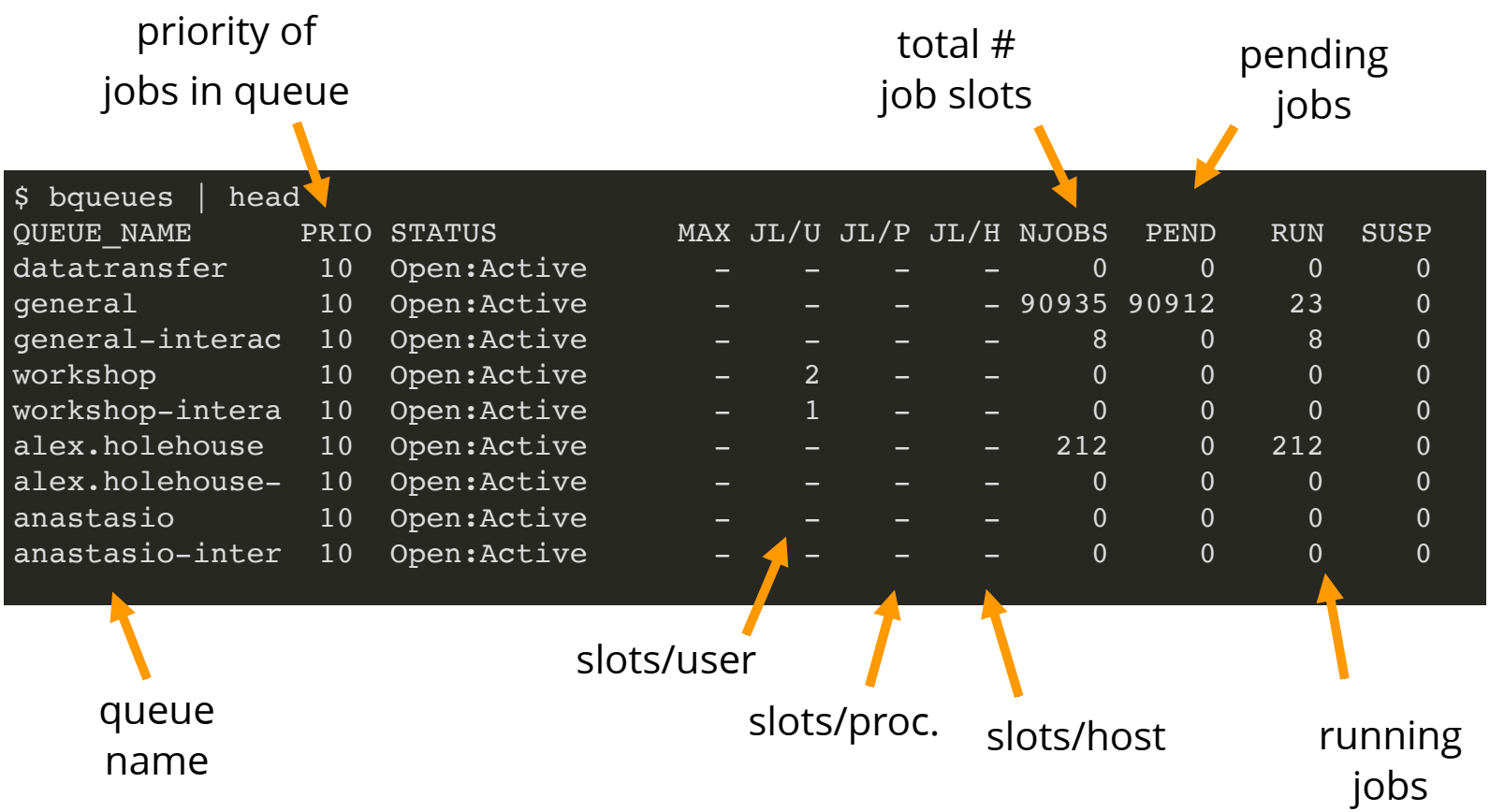
Cancel a job in the queue;
prevents waste of resources,
preserves your fair share score

```
$ bsub -G compute-michael.landis \
> -q general \
> -a 'docker(alpine)' date
Job <92898> is submitted to queue <general>.
# list jobs
$ bjobs
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
92898   michael PEND  general    compute1-cl             date       Oct 25 12:23
# kill job
$ bkill 92898
Job <92898> is being terminated
# job no longer running
$ bjobs
No unfinished job found
```

bqueues

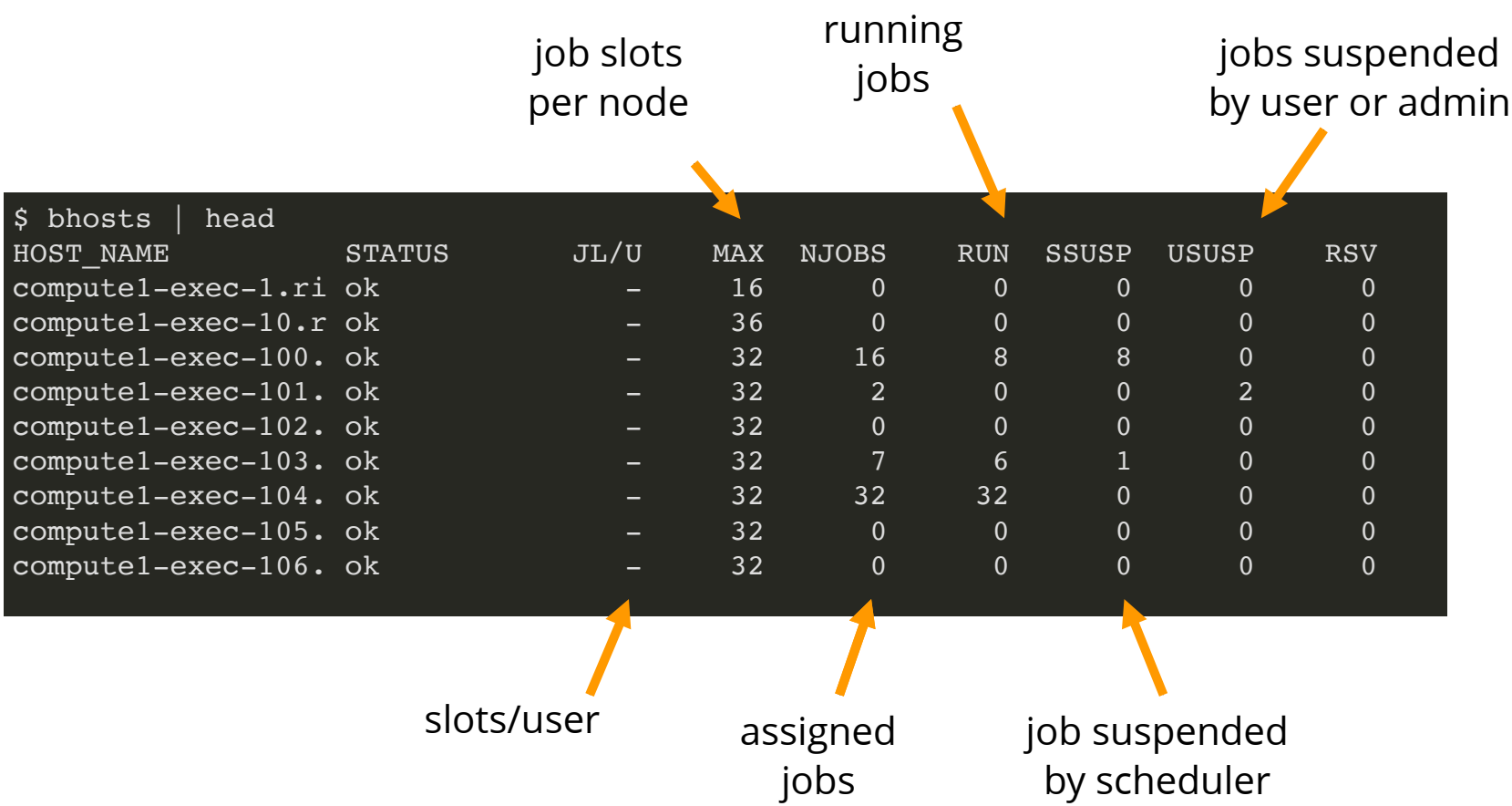
List all queues available on the cluster;

helps identify load across queues



bhosts

List all compute nodes on the cluster



scp

Copy files between two remote computers using SSH protocol

```
# copy file1.txt FROM remote server TO local machine
$ scp mlandis@some.server.org:/home/mlandis/file.txt .
mlandis@some.server.org's password: <password>
file1.txt                                     100%  413    22.7KB/s   00:00

# copy file2.txt TO remote server FROM local machine
$ scp file.txt mlandis@some.server.org:/home/mlandis
mlandis@some.server.org's password: <password>
file2.txt                                     100%  777    22.9KB/s   00:00

# recursively copy my_dir TO remote server FROM local machine
$ scp -r my_dir mlandis@some.server.org:/home/mlandis

michael.landis@compute1-client-1.ris.wustl.edu's password:
file1.txt                                     100%    0     0.0KB/s   00:00
file2.txt                                     100%  413    24.0KB/s   00:00

# remote server now contains the files/directories:
# /home/mlandis/my_dir/file1.txt
# /home/mlandis/my_dir/file2.txt
```

Lab 7A

github.com/WUSTL-Biol4220/home/labs/lab_07A.md