Lecture 11
computer clusters

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

Lecture 11 outline

Last time: regex substitutions

This time: clusters

cluster basics

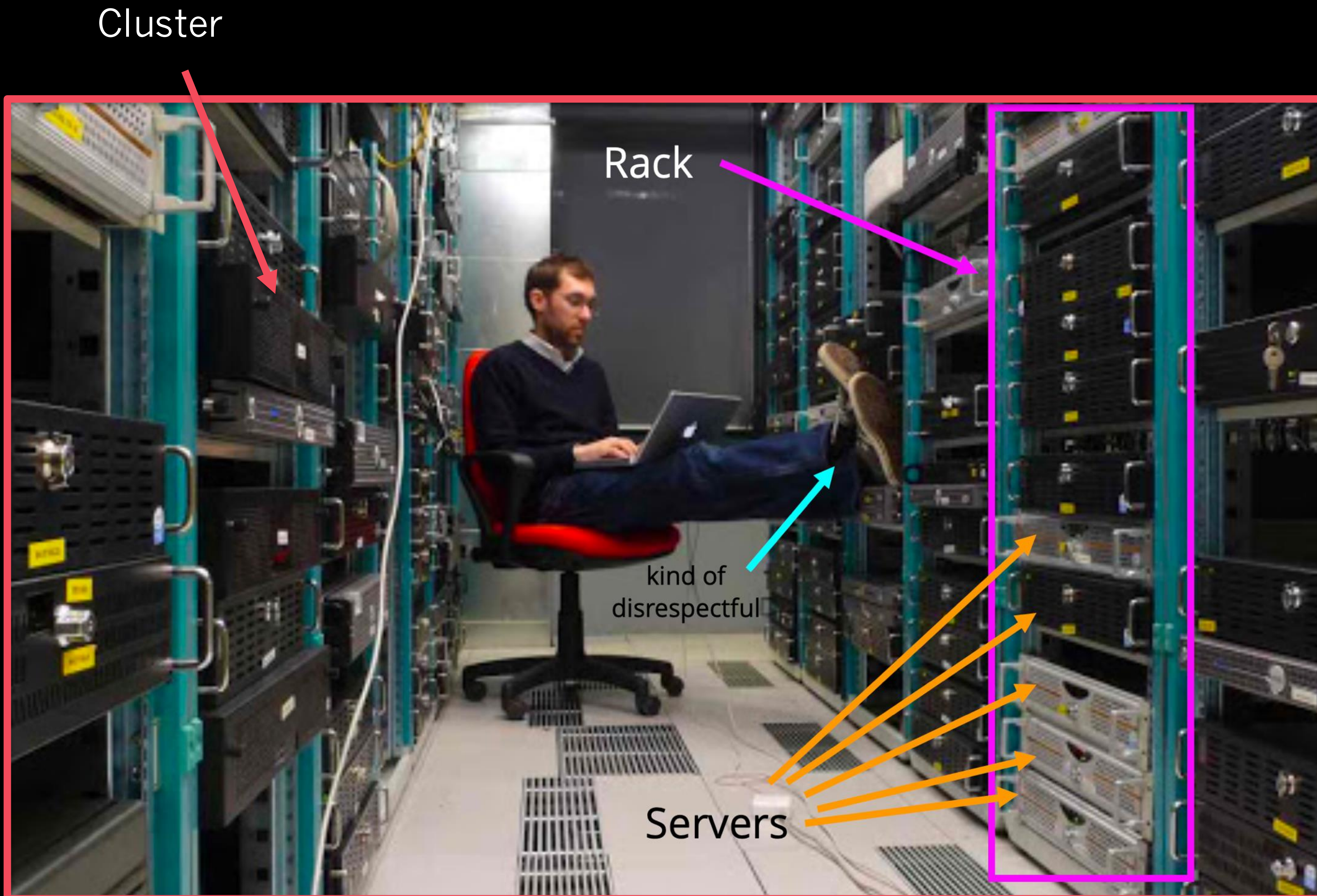
- cluster anatomy
- using clusters
- sharing data w/ clusters

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


Cluster
Rack
kind of
disrespectful
Servers

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
- centralized security

Cluster anatomy

Common cluster features:

- Many ***users***, each with individual needs
- ***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***
  - cores are the units of ***parallelization***
  - some jobs might request nodes with special hardware, such as ***graphics processing unit cards (GPUs)***

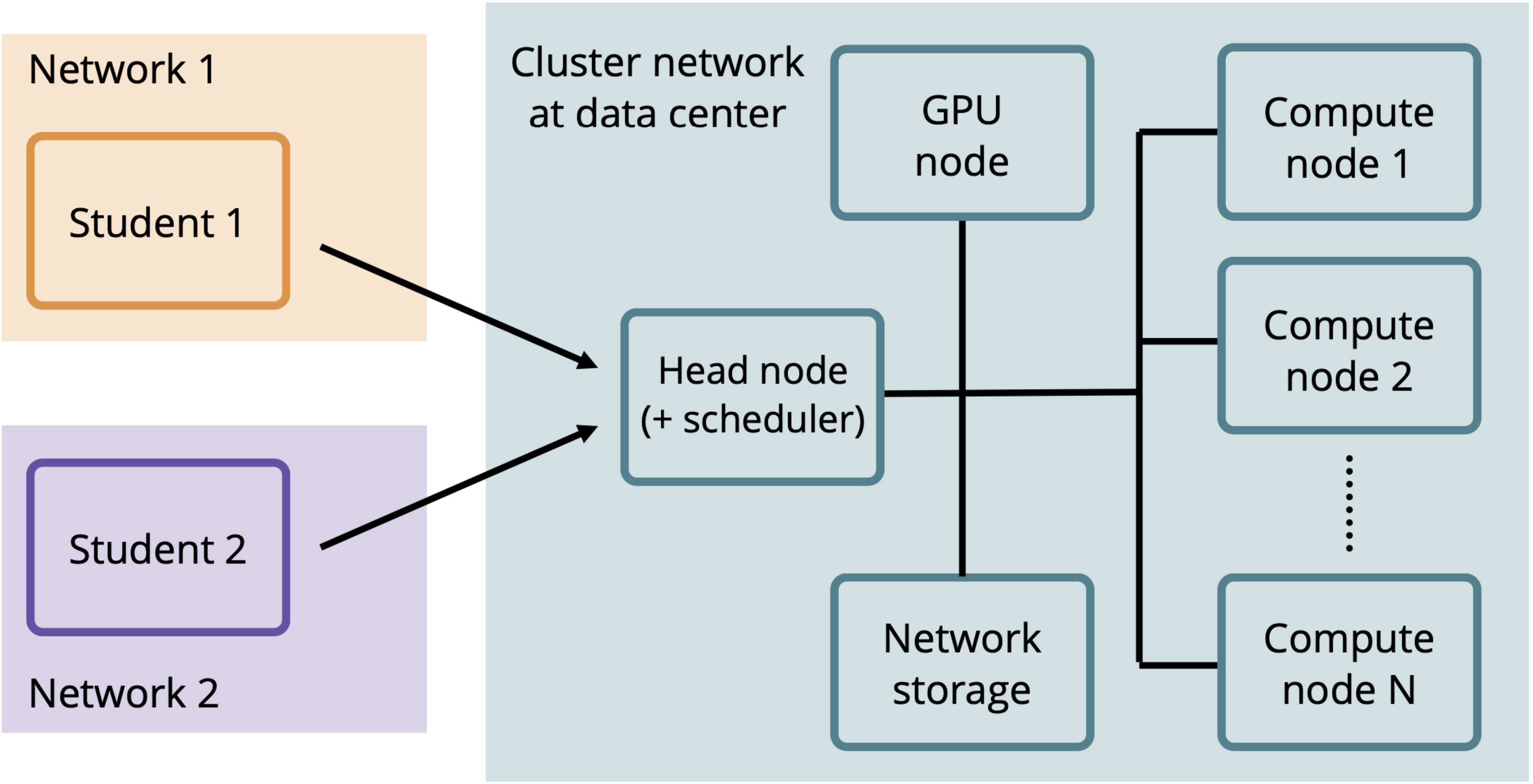
Cluster anatomy

Common cluster features (cont'd):

- 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.

Cluster anatomy



Five fastest clusters

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,699,904	1,206.00	1,714.81	22,786
2	**Aurora** - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	9,264,128	1,012.00	1,980.01	38,698
3	**Eagle** - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Azure Microsoft Azure United States	2,073,600	561.20	846.84	
4	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
5	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,752,704	379.70	531.51	7,107

PFlop/s, 1e15 floating point operations/s

https://www.top500.org/lists/top500/2024/06/

WashU RIS cluster

- Base system
  - 5,000 Intel Cascade Lake Cores
  - 120 NVIDIA Tesla V100 GPUs
  - 300TB DDN high-performance scratch space
  - 100Gbit Mellanox HDR Network

Rough estimate?

CPUs	~0.57 PFlop/s
GPUs	~1.88 PFlop/s

WashU cluster resources

- Resources webpage
  https://ris.wustl.edu/resources/

- Manual
  https://docs.ris.wustl.edu/index.html

- Knowledgebase
  https://washu.atlassian.net/jira/servicedesk/projects/ITSD/knowledge/articles

- Help desk
  http://servicedesk.ris.wustl.edu/

Using clusters

Typical workflow for cluster

1. Connect to cluster network (*e.g.* VPN)
2. Log into cluster through *head node* using *ssh*
3. Set up files and profile on cluster
4. Configure 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***

tragedy of the commons

Job scheduler

Example scenario

Both jobs require 6400 CPU-hours
- User 1 submits 6400x 1-core 1-hour jobs
- User 2 submits 10x 64-core 10-hour jobs

Easier to secure resources for User 1 (many, small) than User 2 (few, large)

Scheduler priorities, managed through policy:
- User 1 jobs decrease in priority as more jobs are run
- User 2 jobs increase in priority the longer they wait

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/docs/en/spectrum-lsf/10.1.0

bsub

Submit a job to queue

interactive shell

compute group

job queue

compute environment

submitted command

```
$ bsub -Is -G compute-michael.landis -q general-interactive
       -a 'docker(alpine)' date

Job <92893> is submitted to queue <general-interactive>.
<<Waiting for dispatch ...>>
<<Starting on compute1-exec-187.ris.wustl.edu>>
Using default tag: latest
latest: Pulling from library/alpine
188c0c94c7c5: Pull complete
Digest: sha256:c0e9560cda118f9ec63ddefb4a173a2b2a0347082d7dff7
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest
Sun Oct 25 17:09:17 UTC 2020
$
```

interactive session prints `date`

bjobs

List all queued jobs

job
status

job
name

```
$ bjobs -u all
JOBID   USER    STAT    QUEUE     FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
1004    user1   RUN     short     hostA       hostA       job0       Dec 16 09:23
1235    user3   PEND    priority  hostM                   job1       Dec 11 13:55
1234    user2   SSUSP   normal    hostD       hostM       job3       Dec 11 10:09
1250    user1   PEND    short     hostA                   job4       Dec 11 13:59
```

unique job
identifier

user who
submitted
job

job
queue

compute nodes
processing job

job submit
time

bkill

Terminate a job in the queue

```
$ 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 for cluster
Helps identify underused queues

priority of jobs in queue

total # job slots

pending jobs

```
$ bqueues | head
QUEUE_NAME       PRIO STATUS          MAX JL/U JL/P JL/H NJOBS  PEND   RUN  SUSP
datatransfer      10  Open:Active       -    -    -    -     0     0     0     0
general           10  Open:Active       -    -    -    - 90935 90912    23     0
general-interac   10  Open:Active       -    -    -    -     8     0     8     0
workshop          10  Open:Active       -    2    -    -     0     0     0     0
workshop-intera   10  Open:Active       -    1    -    -     0     0     0     0
alex.holehouse    10  Open:Active       -    -    -    -   212     0   212     0
alex.holehouse-   10  Open:Active       -    -    -    -     0     0     0     0
anastasio         10  Open:Active       -    -    -    -     0     0     0     0
anastasio-inter   10  Open:Active       -    -    -    -     0     0     0     0
```

queue name

slots/user

slots/proc.

slots/host

running jobs

scp

Secure copy allows you to transfer files
between local and remote filesystems

```
# 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
```

Overview for Lab 11