DINO DE LAURENTIIS
Presents
FLASH GORDON

SAM J. JONES · ORNELLA MUTI · MELODY ANDERSON
as Flash Gordon · as Princess Aura · as Dale Arden

MAX VON SYDOW · TOPOL · TIMOTHY DALTON · BRIAN BLESSED
as The Emperor Ming · as Dr. Hans Zarkov · as Prince Barin · as Prince Vultan

PETER WYNGARDE · MARIANGELA MELATO · Music Composed, Performed and Produced by
as Klytus · as Kala · QUEEN

Screenplay by LORENZO SEMPLE, JR. · Producer DINO DE LAURENTIIS · Director MIKE HODGES

Music by QUEEN on EMI Records and Tapes. EMC 3351

[illegible]

A Member of the Thorn EMI Group EMI

Released by COLUMBIA · EMI · WARNER Distributors Limited

Gordon

- 1024 compute nodes
  - 2 x 8 core (2.6 GHz EM64T Xeon E5)
  - 64 GB RAM (DDR3-1333)
- 64 I/O nodes
  - 2 x 6 core
  - 48 GB (DDR3-1333)
  - 4.8 TB flash storage (16 x 300 GB Intel 710 SSDs)
- 40 Gbps interconnects
- 341 TFlops
- 87 TB/s memory bandwidth
- 1.5 PB storage

Connect to Gordon using your XSEDE username / pass

```
[14:36 dskola@kerner ~] > ssh phage@gordon.sdsc.edu

The authenticity of host 'gordon.sdsc.xsede.org (198.202.104.118)' can't be established.

RSA key fingerprint is 5e:47:dc:37:3e:2e:62:2a:31:57:41:a3:35:d1:82:22.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added 'gordon.sdsc.xsede.org,198.202.104.118' (RSA) to the list of known hosts.

Password:
```

-

Modules

- Modules are predefined sets of environment variables to configure the system for a particular application.

```
$ module avail
```

    list the available modules

```
$ module load <module name>
```

    load the specified module

```
$ module display <module name>
```

    see what that module does

```
$ module purge
```

    unload all modules

- Go ahead and load the biotools module . . .

```
$ module load biotools
```

My account

```
$ show_accounts

ID name      project      used     available    used_by_proj

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

phage        csd399       0        40000        0
```

To charge your job to one of these projects replace << project >> with one from the list and put this PBS directive in your job script:

```
#PBS -A << project >>
```

Account “billing”

- 1 SU = 1 CPU-hour
- Each node has 16 CPUs and can't share jobs
- So even if you only use 1 CPU on a node that's 16 SU / hour
- 4 x 10^5 / 18 students / 16 cores = 139 node-hours.
- Hopefully enough for your group project ;-)

Running jobs

- Controlled by TORQUE Resource Manager
- qsub executes batch scripts that define the jobs

```
$ qsub my_batch_script
```

- Sample-template batch script:

```
#!/bin/bash
#PBS -q normal
#PBS -l nodes=<2>:ppn=<16>:native
#PBS -l walltime=<1:00:00>
#PBS -N <jobname>
#PBS -o <my.out>
#PBS -e <my.err>
#PBS -A <abc100>
#PBS -M <email_address>
#PBS -m abe
#PBS -V
# Start of user commands - comments start with a hash sign (#)
cd /oasis/scratch/$USER/temp_project
<usercommands>
```

Header parameters: queue

```
#!/bin/bash
#PBS -q normal
#PBS -l nodes=<2>:ppn=<16>:native
#PBS -l walltime=<1:00:00>
#PBS -N <jobname>
#PBS -o <my.out>
#PBS -e <my.err>
#PBS -A <abc100>
#PBS -M <email_address>
#PBS -m abe
#PBS -V
# Start of user commands - comments start with a hash sign (#)
cd /oasis/scratch/$USER/temp_project
<usercommands>
```

- '-q' specifies which queue to use.
- In our case we want 'normal'

Header parameters: resources

```
#!/bin/bash
#PBS -q normal
#PBS -l nodes=<2>:ppn=<16>:native
#PBS -l walltime=<1:00:00>
#PBS -N <jobname>
#PBS -o <my.out>
#PBS -e <my.err>
#PBS -A <abc100>
#PBS -M <email_address>
#PBS -m abe
#PBS -V
# Start of user commands - comments start with a hash sign (#)
cd /oasis/scratch/$USER/temp_project
<usercommands>
```

- '-l nodes' specifies how many nodes and CPUs to request
  - here it is 2 nodes with 16 processors per node
- '-l walltime' specifies the max running time (wall time as in clock-on-the wall)
  - Try to be realistic - too long will delay your job

Header parameters: personalization

```
#!/bin/bash
#PBS -q normal
#PBS -l nodes=<2>:ppn=<16>:native
#PBS -l walltime=<1:00:00>
#PBS -N <jobname>
#PBS -o <my.out>
#PBS -e <my.err>
#PBS -A <abc100>
#PBS -M <email_address>
#PBS -m abe
#PBS -V
# Start of user commands - comments start with a hash sign (#)
cd /oasis/scratch/$USER/temp_project
<usercommands>
```

- '-N' naming your job makes it easier to keep track of (obviously)
- '-o', '-e' redirect stdout and stderr (respectively) to these files.
- '-M' send email notifications to this address
- '-m' which notifications?
  - 'a' when the job **a**borts
  - 'b' when the job **b**egins
  - 'e' when the job **e**nds
- '-V' export your current environment variables to the job environment

Header parameters: personalization

```
#!/bin/bash
#PBS -q normal
#PBS -l nodes=<2>:ppn=<16>:native
#PBS -l walltime=<1:00:00>
#PBS -N <jobname>
#PBS -o <my.out>
#PBS -e <my.err>
#PBS -A <abc100>
#PBS -M <email_address>
#PBS -m abe
#PBS -V
# Start of user commands - comments start with a hash sign (#)
cd /oasis/scratch/$USER/temp_project
<usercommands>
```

- '-N' naming your job makes it easier to keep track of (obviously)
- '-o', '-e' redirect stdout and stderr (respectively) to these files.
- '-M' send email notifications to this address
- '-m' which notifications?
  - 'a' when the job **a**borts
  - 'b' when the job **b**egins
  - 'e' when the job **e**nds
- '-V' export your current environment variables to the job environment

Checking on jobs

- What's running?
  - qstat
- Whoa, too much. What am *I* running?
  - qstat -u $USER

```
                                                                                  Req'd    Req'd       Elap
Job ID                 Username    Queue    Jobname          SessID NDS   TSK    Memory   Time     S   Time
--------------------- ----------- -------- ---------------- ------ ----- ------ ------ --------- - ---------
934931.gordon-fe       phage       normal   alignA549           --      2     32    --   00:10:00  Q       --
```

Checking on jobs

- Kill it
  - qdel *jobid*
  - *jobid* can be found with qstat

We're doing it live!

- Log in to Gordon
- Use wget to download the following data set:
  - wget http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeSydhTfbs/wgEncodeSydhTfbsA549Bhlhe40IggrabRawDataRep2.fastq.tgz

We're doing it live!

- Create a directory called “data” in /oasis/projects/nsf/csd399/$USER.
- Use tar to extract the sequence data
  - tar xvfz wgEncodeSydhTfbsA549Bhlhe40IggrabRawDataRep2.fastq.tgz
- Move the extracted files to your data directory

We're doing it live!

- Bowtie2 needs an index to align against (a reference genome)
  - wget ftp://ftp.ccb.jhu.edu/pub/data/bowtie2_indexes/hg19.zip
- Unzip it (unzip *file*) into a directory called “bt2” off of /oasis/projects/nsf/csd399/$USER

We're doing it live!

- We have the raw reads, we have the reference genome.
- Let's map them using bowtie2
- In the github repository is a script called “align_test.pbs”
- Download and edit to put your email address on the -M line

We're doing it live!

- The moment of truth (or consequences):
  - qsub align_test.pbs
- Is it running?
  - qstat -u *user*
- What appears in your home folder?