Using Hoffman2 Cluster

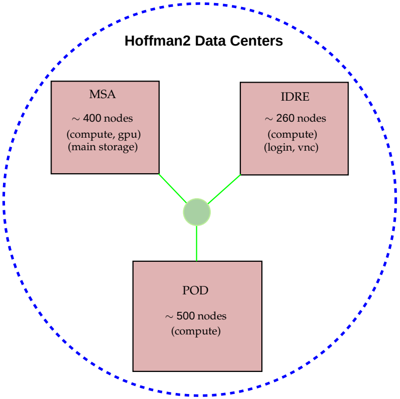
Qiyang Hu

UCLA Institute for Digital Research & Education

May 8th, 2013

Hoffman2: largest and most powerful cluster in UC

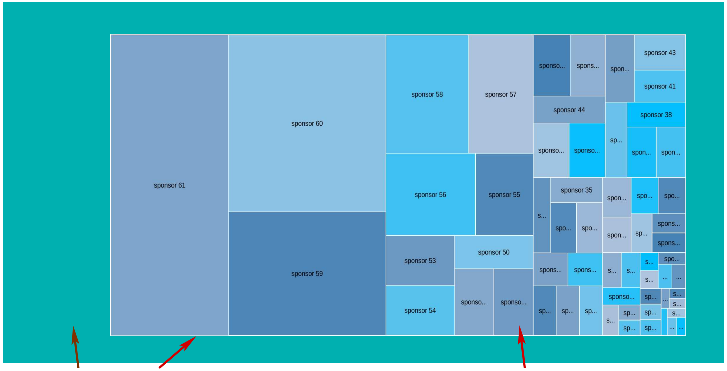
Total:	$>$ 1100 machine
	$\sim$ 11,000 cores
	$>$ 300 GPUs
	$\sim$ 102 Tflops
Storage:	$>$ 1.5 Petabytes
CPU:	8, 12, 16 cores, 2.2 $\sim$ 3.0 GHz
Memory:	1G, 4G, 8G /core
OS:	CentOS Linux 6.4



We have 1,200+ users, 175+ research groups

Research Virtual Shared Cluster

~ 11,000 cores and increasing!



Open to ALL campus users in shared base for <24 hour jobs

Contributed researchers can use their own cores to run longer–hour jobs

Before accessing Hoffman2 cluster

Hoffman2's official site:

`https://idre.ucla.edu/hoffman2`

Need to have a log-in ID first

- via Grid Identity Manager: `gim.ats.ucla.edu`
- open to all current student, staff and faculty member with a valid UCLA log-on ID.

Two modes to use hoffman2

1. Web-service mode:
   - UCLA Grid Portal: grid.ucla.edu
   - Easiest way for windows users
   - Slow, function-limited, depricating
2. Command-line mode :
   - by logging into login nodes
   - efficient, more control
   - might need some basic unix knowledge + SGE commands

Two modes to use hoffman2

1. Web-service mode:
   - UCLA Grid Portal: grid.ucla.edu
   - Easiest way for windows users
   - Slow, function-limited, depricating
2. Command-line mode :
   - by logging into login nodes
   - efficient, more control
   - might need some basic unix knowledge $+$ SGE commands

Outline

Outline

Outline

For Unix, Linux or Mac users

- Using `ssh` command:
  - `ssh` *`login-id`*`@hoffman2.idre.ucla.edu`
  - "Yes" for fingerprint for the first time:
  - One of the physical login nodes will be randomly assigned
    - `login1` $\sim$ `login4`
- Working with interactive GUI applications:
  - Activate X11 forwarding:
    `ssh` *`login-id`*`@hoffman2.idre.ucla.edu -X`

Check Hoffman2 Login Node Fingerprints:

`http://fuji.ats.ucla.edu/for-transfer/hoffman2-cluster/access/login_node.htm#fingerprint`

For Windows users

- Getting `ssh` software: PuTTY, Xshell, Cygwin, Tunnelier, ...

Or just using web browser!

- Chrome Web Store → Secure Shell
- Firefox Add-ons → FireSSH

- Working with interactive GUI application:
  - get an X server: PuTTY + XMing, Cygwin + Cygwin+X
- using NX client (not recommended for Mac user):
  1. download and install NX client (NoMachine Player for Mac Lion)
  2. get the key from /etc/nxserver/client.id_dsa.key
  3. configure login info and input key

Do not run computation on login node!

Additional commands from login nodes

- Change your password by command:

  ```
  passwd
  ```

- Configure your shell (command-line interpreter/scripts):
  - Check your shell: `echo $SHELL`
- Text editor:
  - `vim, emacs, gedit` ...

Tip: to keep ssh session alive:

1. `vi ~/.ssh/config`

   Add: `ServerAliveInterval 5`
2. `vi ~/.bashrc`

   Add: `export TMOUT=36000`

Data storage on Hoffman2

- Home directory: `cd $HOME`
  - 20 GB quota for general campus user
  - backed up for 30 days
- Temporary use:
  - on each node: `cd $TMPDIR`
    100 GB, keep only during the job's run
  - `cd $SCRATCH`
    2 TB limit, keep for 7 days

Remember to check your quota in your home directory

1. ATS scripts: `get_pan_quotas $USER`
2. Linux command: `cd $HOME; du -sh`

Major storage architecture change in June

- absolute home path will change
- distinguish home and sponsored directories
- bash profiles need to be copied into new home

Transferring files

- from Linux/Mac terminal:
  - Use `dtn2` to transfer!
  - `ssh dtn2`
    `scp [-r]` *`source`*`:path/file` *`target`*`:local_path`

- from Windows/Mac/Linux GUI:
  - Drag & drop FTP software
    (e.g. `Cyberduck`, `Macfusion`, `FileZilla`, etc.)

For bigger files, use our high-speed GlobusOnline service!

`http://www.ucgrid.org/go/go.html`

UCLA GlobusOnline guidelines

- Multipoint, multi-stream data transfer
  - fault-tolerant
  - fire-and-forget for server-to-server transfer,
  - 5x faster than `scp`
- 4 quick steps to use Web UI:
  1. have your UCLA grid account ready
  2. create a globus online account at Argonne National Lab
  3. install and run GlobusConnect (if to your desktop)
  4. sign in the globusonline.org and go!
- Command-line interface with restricted SSH
- Caveats:
  - may not be appropriate for file transfers to/from a laptop
  - need Cygwin on Windows XP

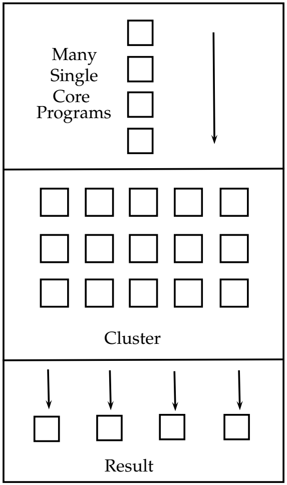
Don’t forget Modules

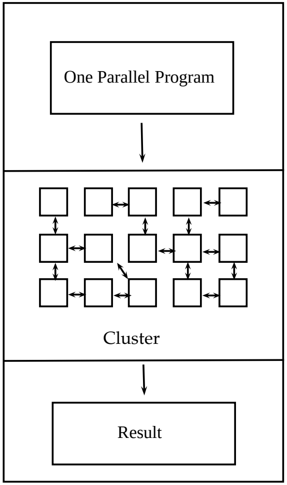
- Modules is for setting environmental variables
  - `$PATH`
  - `$LD_LIBRARY_PATH`
  - other additional env variables.
- Common commands:

current:	`module list`
available:	`module available`
load:	`module load matlab`
unload:	`module unload matlab`
show:	`module show matlab`

Outline

Taking advantage of Clusters





Before submission, you need to know:

1. the type of your job
    - serial or multithreaded or distributed?
2. the group you belong to
3. the name of your executable or input file
    - Your own code:
        - Compile $\longrightarrow$ Link $\longrightarrow$ Executable
    - Precompiled program (FSL, etc):
        - Check the name of the executable
    - Application:
        - Matlab: m file
        - R: R file
        - LAMMPS, etc: input files

Job types on Hoffman2

Serial job:	single thread, single core
Shared memory job:	multi-threaded, single node
MPI Parallel job:	distributed, multiple node
Hybrid job:	MPI and OpenMP
Array job:	serial or multi-thread same executable, different input

For code programmer:

- Intel compiler: 11.1(default), 12.0, 12.1, 13.0
- GCC: 4.4(default), 4.3, 4.7
- Python: 2.4, 2.6(default), 2.7, 3.1
- Java: 1.6.0_23
- matlab: 7.7, 7.11, 7.14(default)
- R: 2.9, 2.12.0, 2.12.1, 2.12.2(default), 2.13.2, 2.15

To check the software, libs installed on the cluster:

- `module av`
- `ls /u/local/apps/`
- ask us!

Which group you are in

Different groups, different policies

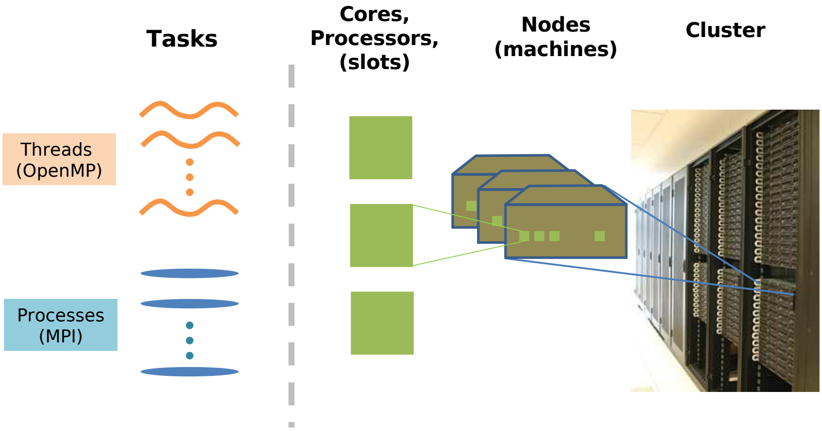
1. Research group:
    - up to 14 days
    - can use their own group's core
2. Others:
    - up to 24 hours
    - 4 GB memory/core
    - up to a certain number of cores/user at a moment

Use `mygroup` command to check!

- See "highp" for your own group's resource
- "queues": rules to jobs for requested resources.

Outline

Of cores, nodes and tasks



In Hoffman2, we will always have: $\boxed{N \text{ tasks} \Rightarrow N \text{ cores}}$

Using ATS queue scripts for 5 types of jobs

- Serial Jobs
  `job.q`
- Serial Array Jobs
  `jobarray.q`
- Multi-threaded Jobs
  `openmp.q`
- Distributed Jobs

`mpi.q`	for MPI
`mpishm.q`	for OpenMP with MPI

- Application Jobs
  `application``.q`

A quick transcript to submit a job

1. have the executable or control file ready
2. type `scriptName.q` and answer prompted questions:
   - type b to build cmd file
   - input executable or control file name
   - input how much memory, how long time
   - for parallel job: how many cpus
   - ...
   - type y to submit
3. type `myjobs` to check the job status

Tip

- Can submit cmd file later by `qsub`

Example: running Serial C++ program

(1) Log into hoffman2: `ssh hoffman2.idre...`

(2) Write C++ code: `vi testSerial.cpp`

(3) Compile and link:
```
g++ -o  target testSerial.cpp
```

(4) Submit & build jobs: just type `job.q`

- Type b to build
- Enter program name: `target`
- Enter memory request: `1024` MB for default
- Enter time limit: `24` hours for default
- Use your own group's cores: y for default
- Enter arguments: if needed
- Enter to submit

Example: running MPI C++ program

(1) Log into hoffman2: `ssh -l ...`

(2) Write C++ code w/ MPI2 API: `vi testMPI.cpp`

(3) Compile and link: `Makefile` → `make` or
`mpiCC -o[-c] target testMPI.c`

(4) Submit & build jobs in hoffman2: `mpi.q`

- Type b to build
- Enter program name: `target`
- Enter memory request: `1024` MB for default
- Enter time limit: `24` for default
- Use your own group's cores: y for default
- Enter task numbers: 8 for our example
- Enter arguments: if needed
- Enter to submit

Example: running Matlab using queue scripts

(1) Log into login node: `ssh -l...`

(2) Write `.m` file : `vi test.m`

(3) Submit & build jobs in hoffman2: `matlab.q`

- Type b to build
- Enter control file name: `test`
- Enter a message option: `bea` for default
- Enter memory request: `1024` MB for default
- Enter time limit: `24` for default
- Use your own group's cores: `y` for default
- Enter arguments: if needed
- Enter to submit

Most-commonly-used SGE commands

- To submit a job:
  `qsub` *`myjob`*`.cmd`
- To determine the status of a job:
  `myjobs` (ATS scripts)
- To cancel a job:
  `qdel [-f]` *`jobNum`*

For more information

Use `man` command, for example: `man qsub`

Outline

Outline

`mygroup` command

Group	Job time	Max slots/usr	Guaranteed start time	Job type	SGE `-l` option
Contribute	$0 \sim 14$ days	as purchased	< 24 hrs if grp not full	any	`highp`
	$0 \sim 24$ hrs	unlimited may change later	none	any	
General Campus	$0 \sim 24$ hrs	400 may change	none	any	
	$0 \sim 2$ hrs	600	none usually < 5 min	serial, shm, jobarray	`express`

Interactive	$0 \sim 24$ hrs	8	immediately if available	any	`i`

More SGE commands

- To changes the attributes of submitted but pending jobs:
  ```
  qalter
  ```
- To hold a queued job to prevent it running:
  ```
  qhold [jobid:taskid]
  ```
- To release a held job:
  ```
  qrls
  ```
- To display node information:
  ```
  qhost [-j]
  ```

More words on checkpoints

- Job's running time on Hoffman2 is limited
  - for campus group: as short as 24 hours.
- Application-level solution: (recommended)
  - Save your data periodically in a restart file.
  - Resubmit and restart from where it left off.
- System-level solution:
  - Cluster has BLCR kernal library installed.
  - Serial & shared-mem jobs OK.
  - MPI job in test.

Outline

Two steps in interactive-scheduler mode

1. Obtain an interactive session
   - interactive session = reservation of resources.
   - Reservation should correspond to your job requirements.
   - Can request nodes up to 24 hours.
   - Using SGE command: `qrsh`
2. Submit/run your job upon

Do not use login node to run your program!

Just use “`qrsh -l i`” to do test-running.

Using `qrsh` to request resources

- `-l` options:
- commonly used parameters:

`i` (or `interactive`):	Request use the int-session nodes
`time` (or `h_rt`):	Wall-clock time limit (default = 2 hrs)
`mem` (or `h_data`):	Request memory size per core
	Max 1 GB/core for campus user

- parameters separated by commas without any space.

Example: request a single core for 2 hours

- `qrsh -l i,mem=1G,time=2:00:00`
- `qrsh -l i,h_data=1024M,h_rt=2:00:00`

Using `qrsh` to request multiple cores

- `-pe` option

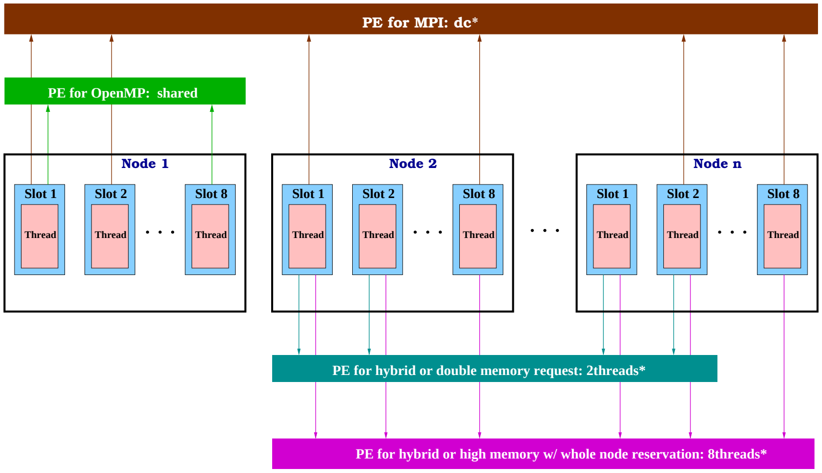
OpenMP:	`-pe` **`shared`** `8`
MPI:	`-pe` **`dc*`** `22`
Hybrid:	`-pe` ***`4threads*`*** `20`

- To reserve an entire node:
  - `-l exclusive=True`

Examples: request an entire node for 4 hour

- `qrsh -l i,mem=1G,time=4:00:00,exclusive=True`

Better understanding for PEs on Hoffman2


PE for MPI: dc*
PE for OpenMP: shared
Node 1
Node 2
Node n
Slot 1
Slot 2
Slot 8
Thread
PE for hybrid or double memory request: 2threads*
PE for hybrid or high memory w/ whole node reservation: 8threads*

After request, we can submit/run jobs:

- Serial, shared-mem, most app job:
    - same as in a local machine.
- MPI job: extra work needed

Example: MPI job (`test`) with 12 cores, 1GB/CPU, for 2 hours

1. Request CPUs from any nodes:
```
qrsh -l i,mem=1G,time=2:00:00 -pe dc* 12 -now n
```
2. Obtain SGE environment variables:
```
source /u/local/bin/set_qrsh_env.sh
```
3. Launch MPI job in OpenMPI:
```
mpiexec -n $NSLOTS ./test
```

Outline

R as a typical CLI application

- Run R interactively:
  - If running R serially: `qrsh -l i`
  - If running R parallelly: `qrsh -l i,exclusive`
  - `module load R`
  - `R`
- Run R in batch:
  - `R.q`
- About R in parallel:
  - multithreading (R 2.14): `multicore`
  - distributed (MPI) (R 2.12): `snow(Rmpi)` or `npRmpi`
  - both need to modify cmd file.

Further info to check

- `https://idre.ucla.edu/hoffman2/software/r`
- specific library docs

If you want to run a Matlab application:

- Multithreading is default from v7.11
  - take either one slot or one whole node.
- We only have limited number of licenses.
  - 6 for general, 4 for compiler
  - 2 for statistical toolbox, ...
- 2 ways to run matlab jobs
  1. in matlab GUI, via interactive session
  2. in batch, via matlab.q or matexe.q

Special note on newly installed Matlab 7.14

- Parallel toolbox: temporarily not working.
- Singlethread by compiler: temporarily not working.

Matlab GUI

1. enable X11-forwarding when login:
   - `ssh hoffman2.idre.ucla.edu -X -l ...`
2. request an interactive session:
   - `qrsh -l i,exclusive=True,time=4:00:00`
3. load matlab module:
   - `module load matlab`
4. run matlab:
   - `matlab`

To start matlab with single-thread mode

```
matlab
```

Mablab batch mode

- `matlab.q` will do 2 steps:
  1. compiling your `m` script to an executable
  2. run the executable as a general job
- Can do 2 steps separately:
  1. run `matmcc.q`
  2. run `matexe.q`
- Parallel Compute Toolbox will need extra work.
  1. export configuration file
  2. add extra code and modify your m file

Tip: sometimes hybriding does a better job!

- `qrsh` + `mcc` do the compilation.
- `matexe.q` do the job submission.

Outline

Short answers for survey questions

- How to interactively run my application?
  Ans: use NX client + qrsh
- How to submit my job which will run in batch mode?
  Ans: use ATS queue scripts (job.q, mpi.q, etc)
- How to do HUGE data transferring?
  Ans: use GlobusOnline.
- How to let my job wait less?
  Ans: request short time and less mem.
- How to make my job running faster?
  Ans: request the whole node if possible.
- How to submit a bunch of jobs simultaneously?
  Ans: use job array or `qsub -hold_jid [JobName]`.
- How to use my group's contributed cores?
  Ans: use highp.
- How to run jobs with longer times (>24 hours)?
  Ans: add checkpoints.

IDRE is helping you!

- Hosting: resources
  - clusters (Hoffman2, UC C^2)
  - storages (high performance, archival)
  - web services (RIM, Grid, GlobusOnline)
- Participating: research projects
- Consulting: supports, code clinics, helps
- Tutoring: classes, virtual summer school

Contact HPC consultant

`hpc@ucla.edu`

Backup slides

UCLA GlobusOnline guidelines

- Data transfer with fault-tolerant, fire-and-forget, 5x faster than `scp`
- 4 quick steps to use Web UI:
  1. have your UCLA grid account ready
  2. create a globus online account at Argonne National Lab
  3. install and run Globus Connect software
  4. sign in the globusonline.org and go!
- Command-line interface with restricted SSH is available for client-side scripting.
- Caveats:
  - may not be appropriate for file transfers to/from a laptop
  - need Cygwin on Windows XP

Running array jobs with `jobarray.queue`

- Job array: same executable, different input (variables/files)
- Each core/task $\Longleftrightarrow$ specific input file.
- Input files:
  - names must include a sequence number.
  - save in the same directory.
- Your code must identify the ID of tasks in Hoffman2.
  - For compiled code:
    — use `getenv("SGE_TASK_ID")` function
  - For script program:
    — use the *$SGE_TASK_ID* environment variable
- Run `jobarray.q` and submit

Example: writing C++ program for array jobs

in1.dat

```
1     2
```

out1.dat

```
i = 1
j = 2
```

in2.dat

```
3     4
```

out2.dat

```
i = 3
j = 4
```

in3.dat

```
5     6
```

out3.dat

```
i = 5
j = 6
```

testArray.cpp

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
using namespace std;

int main( int argc, char *argv[] ){
    stringstream testID;
    testID << getenv( "SGE_TASK_ID" );

    string infileName="in";
    string outfileName="out";
    infileName += testID.str()+".dat";
    outfileName += testID.str()+".dat";

    ifstream infile( infileName.c_str() );
    ofstream outfile( outfileName.c_str() );

    int i, j;
    infile >> i >> j ;
    outfile << "i = " << i << endl;
    outfile << "j = " << j << endl;
    return 0;
}
```

Advanced job array submission

- More controls on job ids and task ids.
- 3 switches:
  - `-hold_jid`: to specify dependency on a job_id
  - `-hold_jid_ad`: to specify dependencies between tasks of different arrray jobs
  - `-tc`: to define max number of concurrent jobs in the job array
- example:

```
Qsub -N job1 -t 1:25 script1.cmd
Qsub -hold_jid job1 -N job2 -t 26:50 script1.cmd
Qsub -hold_jid job2 -N job3 -t 51:52 script1.cmd
```

Running high-memory jobs

- Serial:
  - No need to combine the options of `-pe` and `-l`!
  - Example: Serial job require 3GB memory $\rightarrow$

```
#$ -l mem=3G
```

- MPI (e.g. 3 mpi procs, 8GB per proc):
  - use `mpi.q` to allocate 3 core, 8G/core
  - `h_data` or `mem`: still per core