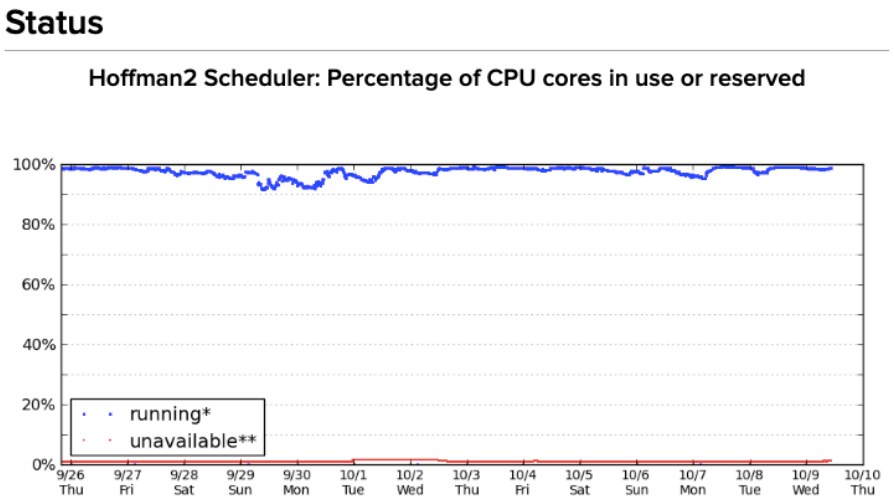
Job Scheduling on Hoffman2 Cluster

Shao-Ching Huang
schuang@idre.ucla.edu

2020-02-26

Hoffman2 Cluster

- 1300+ nodes
- 8~72 cores/node (Intel & AMD)
- Memory (RAM) size: 12GB to O(100) GB
- Some GPU nodes (CUDA)
- Support a range of job types:
  - From single-CPU to MPI-style (multi-node) jobs
- Scheduler (software): Univa Grid Engine
- Operated by IDRE Research Technology Group



https://www.hoffman2.idre.ucla.edu/status/

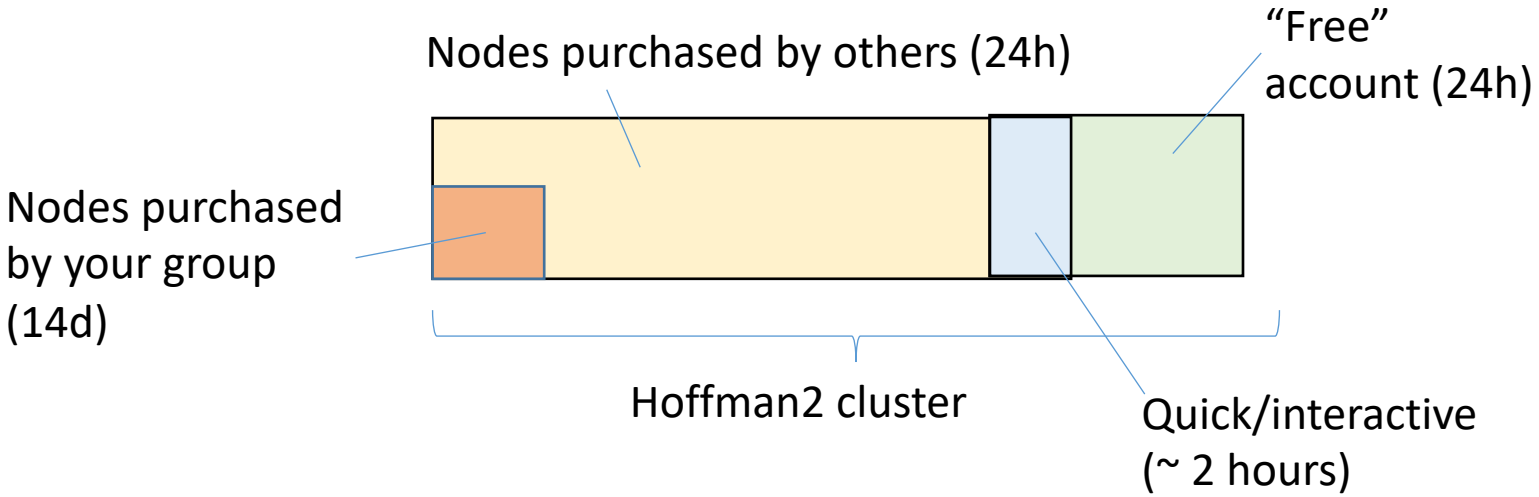
My assumptions

- You have an Hoffman2 cluster account
  - See: https://www.hoffman2.idre.ucla.edu/getting-started/
- You know how to access Hoffman2 cluster from your computer
  - See: http://www.hoffman2.idre.ucla.edu/access/
- You have something to run on Hoffman2 cluster
  - Anything from “hello world” to your research projects
- Use Hoffman2 cluster user support
  - See: http://www.hoffman2.idre.ucla.edu/user-support/
- Ask if you have questions (this class)

Some terminology

- A job is:
  - A program you wrote
  - A program already installed on Hoffman2 (by you or others)
  - A Python script
  - …
- A compute node (or node) is:
  - One of the Hoffman2's servers
  - A node has multiple CPU cores and memory
  - Example: 16 CPU cores, 64GB of memory, 400GB hard disk, connected with Ethernet and Infiniband networking

Access level



- Any UCLA-affiliated person can apply for an account ("free" account)
  - Each account has 20GB of disk space
  - A job can run up to 24 hours (more on this)

- A research group that has purchased Hoffman2 compute nodes can run in two modes:
  - Everything a free account has
  - High priority access on the purchased nodes: up to 14 days
    - Guaranteed to start in 24 hours (when not overusing)
  - Non-high priority access on the nodes other groups purchased: up to 24 hours
    - Many more nodes but start time depends on availability

Job scheduler (software): Univa Grid Engine

- Use “qsub” for batch jobs
- Use “qrsh” for interactive jobs
- Other helper commands to find out job information/status
- Job parameters (discussed in the next few slides)
  - Time limit
  - CPU model
  - Memory size
  - Job type (MPI, shared-memory or sequential)
  - …

Summary info of Hoffman2 nodes: qhost

```
$ qhost
HOSTNAME                ARCH         NCPU  LOAD  MEMTOT  MEMUSE  SWAPTO  SWAPUS
----------------------------------------------------------------------------------
global                  -               -     -       -       -       -       -
n106                    amd-2376        8  1.02    7.8G  886.8M  980.5M  348.4M
n132                    amd-2354        8  4.03    7.8G    1.0G  980.5M  388.2M
n135                    amd-2354        8  6.76   31.5G    2.2G  980.5M   86.6M
n2162                   amd-6136       16 15.02  126.2G    2.9G  980.5M     0.0
n2178                   intel-X5550     8  1.03   23.6G  670.9M  980.5M  746.7M
n2190                   amd-2380        8  1.51   31.5G  791.7M  980.5M  214.9M
n230                    amd-2354        8  3.04    7.8G    1.9G  980.5M  495.1M
n26                     amd-2376        8  5.03   15.7G    2.2G  980.5M  702.7M
n4006                   amd-2382        8  8.08   63.0G    4.7G  980.5M  825.5M
n4026                   intel-X5650    12  8.05   47.3G    1.3G  980.5M  160.0K
n60                     amd-2431       12  8.04   31.5G    1.7G  980.5M  579.7M
n6037                   intel-E5-2650  16  0.41   63.0G    1.1G   15.3G     0.0
n6071                   intel-E5-2650  24 23.87  126.1G    7.1G   15.3G     0.0
n6072                   intel-E5-2650  24 21.00  126.1G    2.8G   15.3G     0.0
n6095                   intel-E5-2670  16  0.01   63.0G    1.1G  980.5M  215.1M
n6162                   intel-E5-2670  16  0.01  252.4G    2.0G  980.5M  158.0M
n6164                   amd-4176       12  2.04  189.3G   20.1G   15.3G     0.0
n9750                   intel-E5530     8  2.24   23.6G    1.0G  980.5M     0.0
n9751                   intel-E5530     8  1.01   23.6G  678.6M  980.5M     0.0
```

1300+ nodes

Note: your acount may not have access to all of these nodes listed by “qhost”.

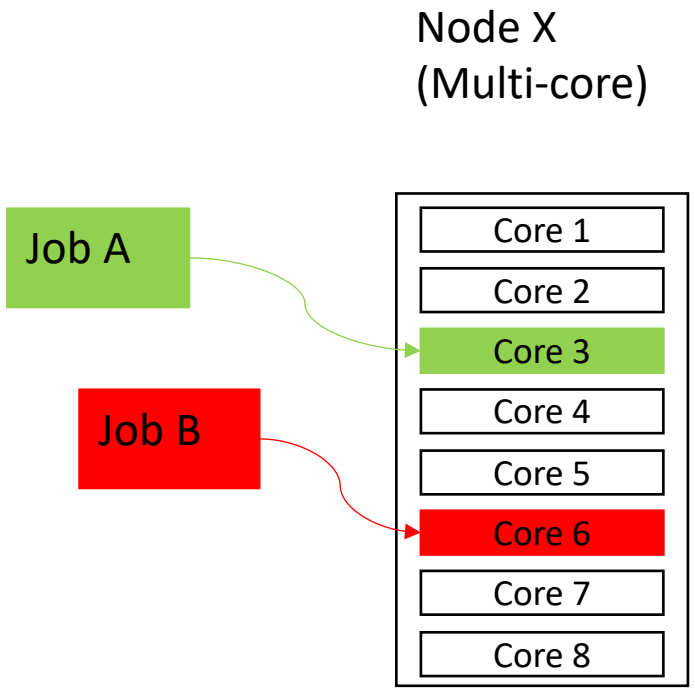
Basic requirements of a job

To inform the scheduler about the job in order to look for matching resources (CPU/memory) to run it.

- Time limit (h_rt)
  - For “free” users, the maximum time is 24 hours per job
  - Up to 14 days per job on the nodes your group purchased
- Memory size (h_data)
  - For multi-core (or multi-node) jobs, this is the per-core memory size
  - Determines what nodes can run your job(s)
- Number of CPU cores
  - Default to 1 if not specified

- More options for more advanced jobs

Sequential jobs



A sequential job uses only one CPU core.

e.g. a “standard” Python/R script, C code, etc.

All Hoffman2 nodes are multi-core, a sequential job uses 1/N of a compute node, sharing the rest with other jobs (unless specified otherwise).

There are other (users’) jobs running on the same node, unless you specify: -l exclusive

Job script example for sequential jobs

No space

```
#!/bin/bash
#$ -l h_rt=8:00:00,h_data=2G
#$ -cwd
./a.out
```

- Time limit, memory size
- Run from current directory (cwd)
- Name of executable
- The lines with leading #$ are scheduler's job parameters
- The file should not contain Windows/DOS carriage return characters (use "dos2unix" to fix it)

Time limit: h_rt

- h_rt sets the wall-clock run time limit of a job
- When h_rt is reached, a job is terminated unconditionally
  - It is possible to alter h_rt before a job starts, using “qalter”
  - It is NOT possible to alter h_rt after a job has started
- A regular job (“free user”) can run up to 24 hours (h_rt=24:00:00)
- A high priority (“-l highp”) job can run up to 14 days (h_rt=336:00:00)
  - Only for groups that have purchased their compute nodes
  - If your group has not purchased nodes, you have no access to this
- Example: `-l h_rt=12:00:00`

Considerations of setting h_rt

- Hoffman2 cluster compute nodes run at different speeds
  - Some old nodes could be much slower than the newer nodes
- If you allow the scheduler to select any nodes, set a larger h_rt so that it is still long enough on slower nodes
- h_rt is the run time limit; wait time is not counted in h_rt
- You may need to experiment a bit to find your optimal setting

Memory size: h_data

- h_data = per-core memory size
- Example: `-l h_data=4G`
- When a job's virtual memory usage exceeds the h_data, the job will be terminated by the scheduler (by a Linux kill signal)
  - To protect other jobs (running on the same node) from your memory overuse
  - Make h_data large enough to run your job, but not too large
- If you set a very large h_data (e.g. –l h_data=256G), your job may never start because the scheduler is unable to find such nodes
  - Unless your group has purchased such large-memory nodes

Job script example for sequential jobs

```
#!/bin/bash
#$ -l h_rt=8:00:00,h_data=2G
#$ -cwd
#$ -N my_job_name
./a.out
```

- Job name (as shown in “qstat”) – useful for your bookkeeping but not required

Job script example for sequential jobs

```
#!/bin/bash
#$ -l h_rt=8:00:00,h_data=2G
#$ -cwd
#$ -N my_job_name
#$ -o $JOB_NAME.o$JOB_ID
#$ -e $JOB_NAME.e$JOB_ID
./a.out
```

- Job name (optional)
- File name for stdout
- File name for stderr

$JOB_NAME is the name specified by –N
$JOB_ID is an unique integer id for a job

After running the job, you get additional files like: my_job_name.e12345 and my_job.o12345

Job script example for sequential jobs

```
#!/bin/bash
#$ -l h_rt=8:00:00,h_data=2G
#$ -cwd
#$ -N my_job_name
#$ -o $JOB_NAME.o$JOB_ID
#$ -j y
./a.out
```

- Combine stdout and stderr in one file

Job script example for sequential jobs

```
#!/bin/bash
#$ -l h_rt=8:00:00,h_data=2G,arch=intel*
#$ -cwd
#$ -N my_job_name
#$ -o $JOB_NAME.o$JOB_ID
#$ -j y
./a.out
```

- Run on Intel CPU

  Even if some AMD nodes are available, this job will still look/wait for availability of Intel nodes

Exclusive mode

- Request to be the only user on a node
  - –l exclusive
- Useful for:
  - Minimize system “noise” caused by other running jobs (e.g. benchmarking)
  - Total control of a compute node’s resources (memory, CPU, etc.) as the sole user at run time
- Wait time may be longer
  - Depending on how busy the cluster is

Job script for sequential jobs

```
#!/bin/bash
#$ -l h_rt=8:00:00,h_data=24G,exclusive
#$ -cwd
#$ -N my_job_name
#$ -o $JOB_NAME.o$JOB_ID
#$ -j y
./a.out
```

- You will be the only one on a node with at least 24GB RAM

Submitting a job: qsub

- Suppose the job script is named “foo.sh”

```
$ qsub foo.sh
Your job 802710 ("test") has been submitted
$
```

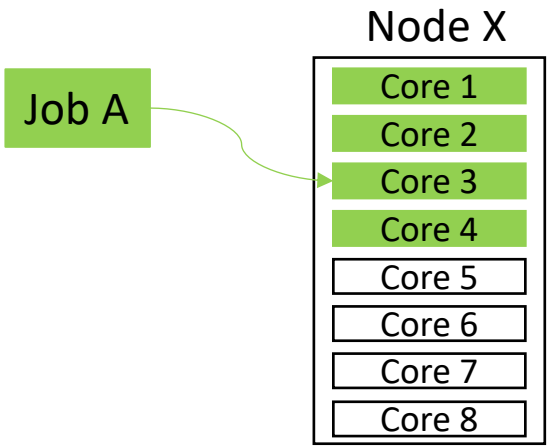
My Job still does not start… Why?

- The scheduler takes some time to put your job(s) into the queue
- The scheduler has not found available compute nodes to run your job(s):
  - Non-existent hardware (requesting too much memory, too long run time)
  - Currently occupied by other jobs
  - maintenance
- You have no access to the requested computing resource
  - E.g. “free” users trying to run longer-than-24-hour jobs
- More on this later

Job parameters are filters

Jobs will not start if the rules are too strict.

- To select the right compute node(s) to run your job
- Examples
  - To run on only Intel CPUs (not AMD):
    `-l arch=intel*`
  - To run on a class of Intel CPUs (but not others):
    `-l arch=intel-E5-*`
  - To request 24GB of RAM – so your job will not go to a small 12GB-RAM node
    `-l h_data=24G`
- The more constraints you set, the smaller subset of nodes your job could use – possibly longer wait time (or not starting at all in some cases)
- Rule of thumb: Request what you need, without being overly restrictive



Shared-memory Jobs

Threaded program that use multiple CPU cores on a single node.

Check with the developers or the user’s manual to see if your program can do this.

Specify # of CPUs using: -pe shared

- Specify the number of CPUs your job will use, e.g.

  -pe shared 8

- See the “NCPU” column of “qhost” output
- If you are a “free” user, your access to 24-core nodes may be limited
  - Potentially long wait or not starting at all

```
$ qhost
HOSTNAME                ARCH         NCPU  LOAD
-------------------------------------------------
global                  -               -     -
n106                    amd-2376        8  1.02
n132                    amd-2354        8  4.03
n135                    amd-2354        8  6.76
n2162                   amd-6136       16 15.02
n2178                   intel-X5550     8  1.03
n2190                   amd-2380        8  1.51
n230                    amd-2354        8  3.04
n26                     amd-2376        8  5.03
n4006                   amd-2382        8  8.08
n4026                   intel-X5650    12  8.05
n60                     amd-2431       12  8.04
n6037                   intel-E5-2650  16  0.41
n6071                   intel-E5-2650  24 23.87
n6072                   intel-E5-2650  24 21.00
n6095                   intel-E5-2670  16  0.01
n6162                   intel-E5-2670  16  0.01
n6164                   amd-4176       12  2.04
n9750                   intel-E5530     8  2.24
n9751                   intel-E5530     8  1.01
```

Job script example for shared-memory jobs

```
#!/bin/bash
#$ -cwd
#$ -l h_data=2G,h_rt=1:00:00
#$ -pe shared 8
./a.out
```

- h_data is “per-core” memory size.
- In this case, you are requesting a total of 2GB*8=16GB memory
- Always check the product of (h_data)*(pe) against the total memory size of a node

Quiz: John submits the following job, but it does not start for days. Why?

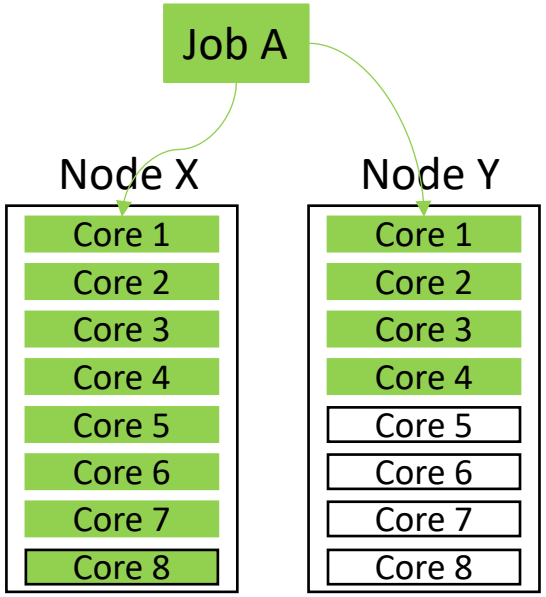
```
#!/bin/bash
#$ -cwd
#$ -l h_data=16G,h_rt=1:00:00
#$ -pe shared 16
./a.out
```

- John is requesting a total memory of 16GB*16=256GB(!) from a node.
- John may not have access to 256GB compute nodes (unless John's group purchases such nodes).

Quiz: John submits the following job, but it sits in the queue forever. Why?

```
#!/bin/bash
#$ -cwd
#$ -l h_data=1G,h_rt=1:00:00
#$ -pe shared 32
./a.out
```

- John is requesting a total memory of 1GB*32=32GB
- John may not have access to compute nodes with 32-core CPUs.



MPI-style (multi-node) jobs

The job runs across multiple nodes. Only programs capable of doing so (e.g. utilizing MPI) can be submitted this way.

Use –pe dc* for multi-node jobs

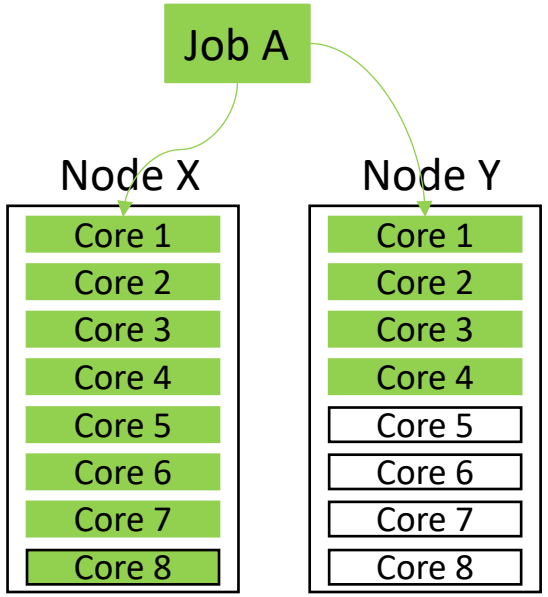
- Jobs will run across multiple nodes
- The "*" in "-pe dc*" is significant
- If your program is multi-threaded (shared-memory), you must use "-pe shared", not "-pe dc*". See previous slides.
- For # of CPU/cores, try using multiple of 8 or 12, e.g.

  -pe dc* 16

  -pe dc* 24

  -pe dc* 32

  To minimize the spreading of nodes used for the job



Job script example for multi-node MPI jobs

```
#!/bin/bash
#$ -cwd
#$ -l h_data=2G,h_rt=1:00:00
#$ -pe dc* 32
source /u/local/Modules/default/init/modules.sh
module load intel/13.cs
mpirun –np $NSLOTS ./a.out
```

Use “module load” because we want to use Intel MPI here

$NSLOTS is the number of cores, specified by –pe

Job script example for multi-node MPI jobs

```
#!/bin/bash
#$ -cwd
#$ -l h_data=2G,h_rt=1:00:00,arch=intel*
#$ -pe dc* 32
#$ -o $JOB_NAME.o$JOB_ID
#$ -j y
source /u/local/Modules/default/init/modules.sh
module load intel/13.cs
mpirun –np $NSLOTS ./a.out
```

Want to use only Intel CPUs

Job Arrays

A way to submit many similar jobs, parameterized by an index

When to use a job array?

- You want to submit a large number of similar jobs
  - Can identify the cases by an integer (index)
  - E.g. Each case has a different set of input data (indexed by the integer)
- Key parameter to use: -t

  Example: to run 1000 cases:

```
#$ -t 1-1000
```

Job array example

```
#!/bin/bash
#$ -cwd
#$ -l h_data=1G,h_rt=1:00:00
#$ -t 1-1000
./a.out  $SGE_TASK_ID
```

The program needs to be able to capture this environment variable

- At run time, the scheduler will launch 1000 “tasks”, each of which is an independent job
- Each task is assigned by a unique ID, $SGE_TASK_ID, ranging from 1 to 1000
- For example, the first task gets $SGE_TASK_ID=1, the second task gets $SGE_TASK_ID=2, and so on.

Pack many short tasks in one run

When a task is very short, it can be take longer to schedule than to run the job.

```
#!/bin/bash
#$ -t 1-2000:100
...
for i in `seq 0 99`; do
    my_task_id=$((SGE_TASK_ID + i))
   ./a.out $my_task_id
done
```

- 1 to 2000 with a step size of 100.
- The first task gets SGE_TASK_ID=1, the second task gets SGE_TASK_ID=201, and so on.
- Each SGE_TASK_ID (task) will run 100 cases one by one in the loop

See: http://www.hoffman2.idre.ucla.edu/faq/#How_do_I_pack_multiple_job-array_tasks_into_one_run

High priority jobs

```
#!/bin/bash
#$ -cwd
#$ -l h_data=1G,h_rt=1:00:00,highp
./a.out
```

- The job will run on your purchased nodes
- Guarantee to start in 24 hours
- … except if your group members are already running long jobs occupying your purchased nodes

Things to check if a job is not starting

- Maximum h_rt allowed
  - For “free” users: up to 24 hours
  - For high-priority users: up to 336 hours (=14 days)
- For sequential jobs (no “-pe” specified)
  - h_data cannot exceed a permissible node’s total RAM size
- For shared-memory (same-node) jobs
  - The requested # of cores must be “realistic”
  - h_data * (# of cores) cannot exceed a node’s total RAM size
- Hoffman2 cluster’s queue is busy almost all the time; do not expect a job to start immediately

How to set memory size

- Job using more than the requested memory size (h_data) at runtime may be killed
- h_data is a per-process limit of virtual memory size
- Depending on the type of your job, different scenarios need to be considered

h_data for sequential and MPI jobs

- h_data is a per-process limit
- Sequential jobs:
  - the peak virtual memory usage during run time
- MPI jobs
  - The peak virtual memory usage per process (e.g. mpirun –n 16 a.out)

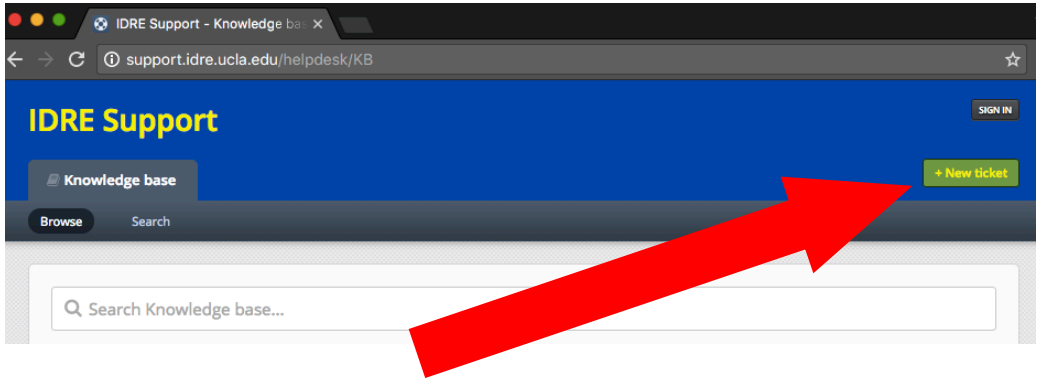
- Note: Virtual memory size is not the same as resident memory size (which is closer to the actual memory usage). Hoffman2 cluster's scheduler (UGE) checks for virtual memory usage, however.

h_data for shared-memory (multithreaded) jobs

- Consider –l h_data=4G –pe shared 8
- h_data is a per-process limit, but the multithreaded job runs as one single process (which may use up to, e.g. 20GB, exceeding h_data=4G)
- Workaround:
  1. -l h_data=4G,exclusive –pe shared 8
  2. -l h_data=4G,h_vmem=32G -pe shared 8

     (The h_vmem value is the product of (h_data)*(-pe shared)

Seeking help



- http://support.idre.ucla.edu/helpdesk
  - Click “New ticket”
- Always provide the following information – making it easier for the people who will help you:
  - Your full name
  - Your Hoffman2 cluster user name
  - Job number(s) if available
  - Specific description of the problem
  - Full error message or screen output
- The consultants are also working on their own projects
- Just saying “help! Urgent! it does not work!” does not help anybody

Summary

- Two groups of users:
  - “Free” users: sharing a smaller pool of nodes. Up to 24 hours per job
  - Hoffman2 contributors (purchasing nodes):
    - High-priority jobs on your purchased nodes (guaranteed start time): 14 days per job
    - Utilize a large pool of compute nodes (24 hours per job)
- Identify your job types
  - Sequential | shared-memory | MPI-style | job array
- Specify the appropriate job parameters
  - Memory size, # of CPU cores, time limit, etc.
  - Check for conflicting or contradicting parameters
- Get help: support.idre.ucla.edu/helpdesk