N I H  H e l i x  S y s t e m s

# B i o w u l f :  A Beowulf for Bioscience

Steven Fellini
sfellini@nih.gov

Susan Chacko
susanc@helix.nih.gov

CIT
December 7, 2010

1

## Introduction to Biowulf

- Intro to the Biowulf Cluster
- Cluster Basics & Concepts
- Biowulf architecture & hardware configuration
- Why use Clusters/Biowulf?
- Accounts/Passwords
- Connecting to Biowulf, Email
- Storage
- Running Jobs & the Batch System
- Monitoring Jobs
- Node allocation & selection
- Tour of Biowulf Machine Room

2

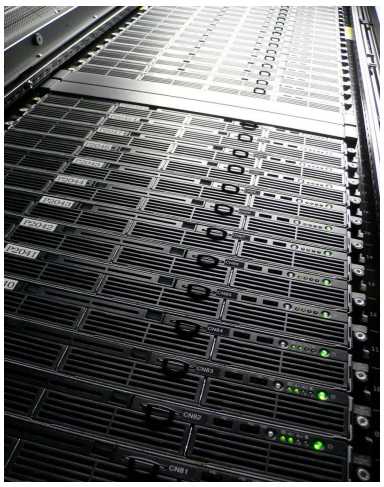
# Introduction to Biowulf

- New class: "NIH Biowulf Cluster Advanced Topics", Jan 19, 2011 1-3 pm
- Slides available as http://biowulf.nih.gov/biowulf-seminar-dec2010.pdf

3

# The NIH Biowulf Cluster

- Central scientific compute resource managed by CIT Helix Staff
- Funded by NIH Management Fund
- Available to all NIH intramural scientists
- Production facility: high availability, data integrity
- Large scale: ~9000 processors; Petabyte storage capacity
- Enabling research not otherwise possible
- General purpose scientific computing (not dedicated to any one application type)
- Cited by over 200 publications in 2008-2010



4

http://biowulf.nih.gov

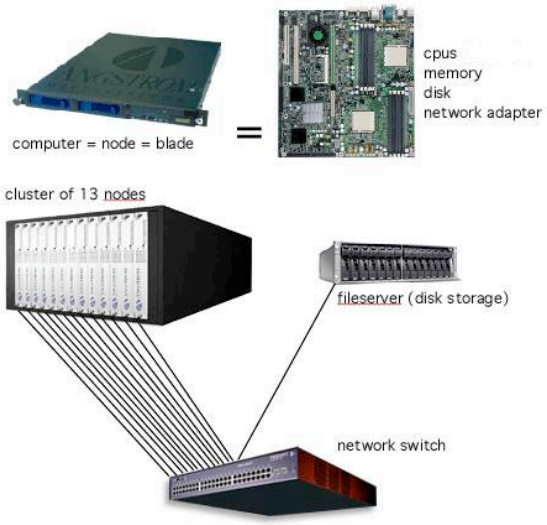
http://biowulf.nih.gov/apps/

http://biowulf.nih.gov/user_guide.html

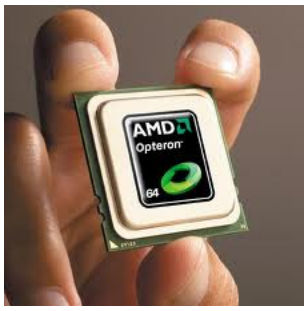
http://biowulf.nih.gov/research.html

RSS

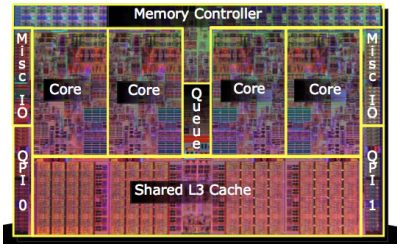
# Cluster Basics



# Terminology/Concepts

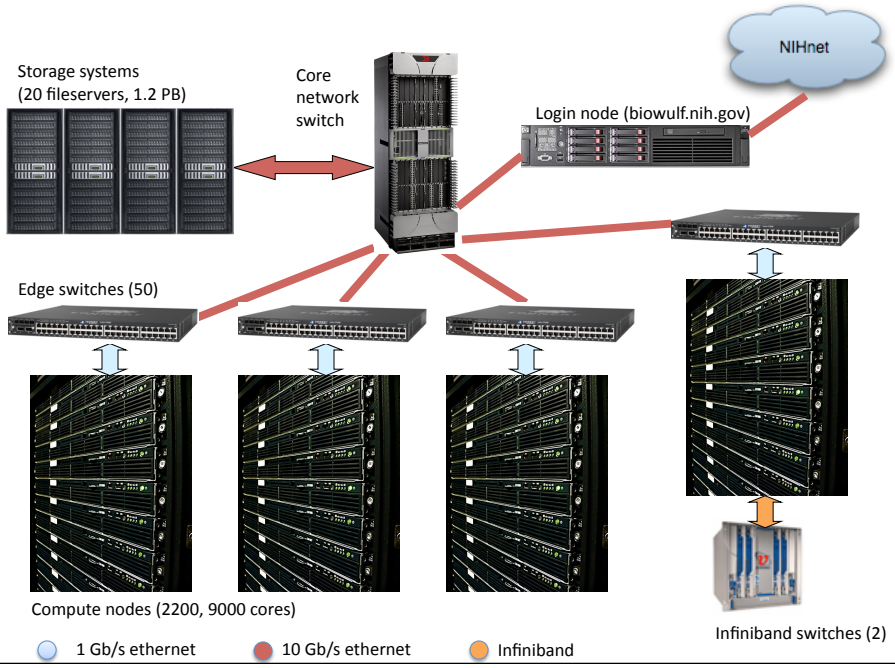



What should we call this?




- Node (all Biowulf nodes have 2 socket motherboards)
- Processor vs. CPU vs. Core
- Process vs. Thread
- 1 thread => 1 core

# NIH Biowulf Cluster Architecture



## Compute Node Technical Specifications

| # nodes | cores per node | Memory | network |
|---|---|---|---|
| 352 | 8 x 2.67 GHz Intel Xeon (X5550)<br>8 MB secondary cache | 320 x 24 GB<br>32 x 72 GB | 1 Gb/s ethernet |
| 1 | 32 x 2.5 GHz AMD Opteron 8380<br>512 KB secondary cache | 512 GB | 1 Gb/s ethernet |
| 250 | 8 x 2.8 GHz Intel Xeon (E5462)<br>12 MB secondary cache | 8 GB | 1 Gb/s ethernet<br>16 Gb/s infiniband |
| 232 | 4 x 2.8 GHz AMD Opteron 290<br>1 MB secondary cache | 8 GB | 1 Gb/s ethernet |
| 289 | 4 x 2.6 GHz AMD Opteron 285<br>1 MB secondary cache | 8 GB | 1 Gb/s ethernet |
| 471 | 2 x 2.8 GHz AMD Opteron 254<br>1 MB secondary cache | 40 x 8 GB<br>226 x 4 GB<br>91 x 2 GB | 1 Gb/s ethernet<br>160 x 8 Gb/s infiniband |
| 389 | 2 x 2.2 GHz AMD Opteron 248<br>1 MB secondary cache | 129 x 2 GB<br>66 x 4 GB | 1 Gb/s ethernet |
| 91 | 2 x 2.0 GHz AMD Opteron 246<br>1 MB secondary cache | 48 x 1 GB<br>43 x 2 GB | 1 Gb/s ethernet |

## Compute Node Configurations, continued

- Heterogeneous: 14 node configurations
- 2-4 GB swap
- 60-200 GB local scratch disk
- RedHat/CentOS 5.x, 64-bit

## Why would you want to use Biowulf?

- Large numbers of jobs (bioinformatics on thousands of sequences)
- Large-memory jobs (genome assemblies)
- Parallel jobs with high-cpu demands (e.g., molecular dynamics on 256 cores)

… in other words, LARGE SCALE

## Unsuitable for Biowulf…(or, why bother?)

- Small numbers of jobs
- One dependent job after the other
- Interactive jobs, graphics etc.

… in other words, desktops can be pretty powerful! (Helix too)

## Class survey

- Level of Unix/Linux experience?
- Helix account?
- Biowulf account?
- Your research application?

## Accounts & Passwords

- Every user must have his/her own account. NO SHARING of accounts
- Requires a pre-existing Helix account (see http://helix.nih.gov/Documentation/accounts.html)
- Registering for a Biowulf account (see https://helix.nih.gov/nih/account_request.html)
- Passwords – initially the same as your Helix password, but not they are not kept in sync. We can sync your Helix and Biowulf passwords on request
- Default login shell: bash
- No $ charges associated with your Biowulf account (/home directory usage is a Helix account charge)

# Connecting to Biowulf & Email

- *ssh* to biowulf.nih.gov (see http://helix.nih.gov/new_users/connect.html)
- Must be on NIH VPN (or login from helix)
- Email goes to your_username@helix.nih.gov
- When you apply for a biowulf account:

**Email Address:**

I will read my email from Biowulf systems and staff at my Helix address:

○ Yes ○ No, my preferred email address is:

- **Important:** make sure your helix email is being properly forwarded if you don't normally read it (see http://helix.nih.gov/Email/)

# Use of Biowulf's login node

- Submitting jobs
- Editing/compiling code
- File management
- File transfer
- **Brief** testing of code or debugging (under 10 minutes cpu time!)
- DO NOT RUN COMPUTE JOBS OF ANY KIND ON THE LOGIN NODE

# Disk storage

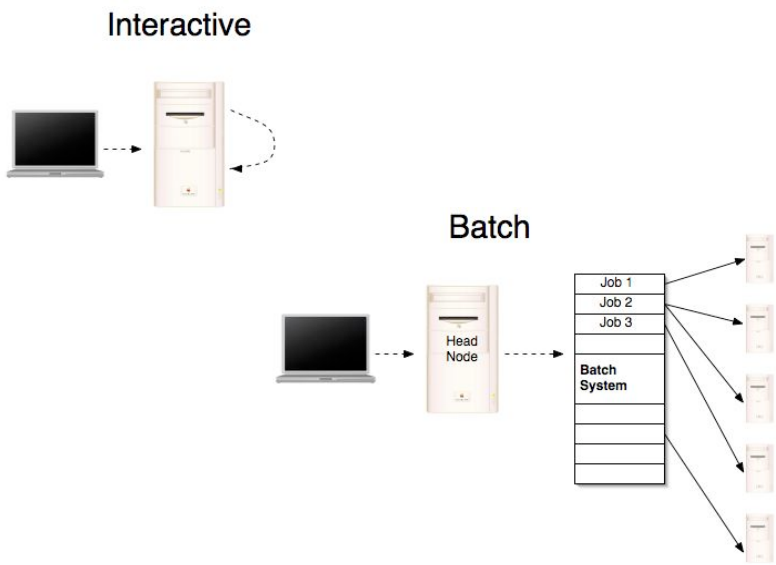
| | Location | Creation | Backups? | Amount of space | Accessible from (*) |
|---|---|---|---|---|---|
| /home | fileservers | with Helix account | Yes | 8 GB (quota) | B, H, C |
| /data | fileservers | with Biowulf account | Yes | 100 GB (quota) | B, H, C |
| /scratch | fileservers | created by user | No | 540 GB shared by all users | B, H |
| /scratch | local disk | created by user | No | varies up to 200 GB, dedicated while node is allocated | C |

(*) H = helix, B = biowulf login node, C = biowulf computational nodes

# Disk Storage

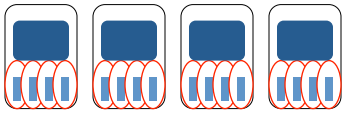
- Use /data/username, ***not*** /gs1/users/username
- *checkquota* command to check usage
- *clearscratch* command to clear out local /scratch on compute nodes
- Login node: use /scratch, ***not*** /tmp or /var/tmp
- Quota increases for /data – request at https://helix.nih.gov/nih/storage_request.html
- access from your workstation (see http://helix.nih.gov/Documentation/transfer.html)
- Snapshots – 6 daily, 2 nightly, 4 weekly on /home directories (less frequent on /data)

# Concepts: Interactive vs. Batch

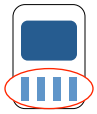



# Kinds of Jobs on a Cluster

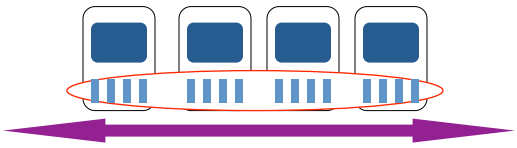
- Hundeds/thousands of single-threaded apps (*swarms*)



- Shared-memory parallel apps (multi-threaded)



- Distributed-memory apps



# Jobs on the Biowulf Cluster

- PBS is the batch system (queuing software)
- Unit of allocation = node (2, 4 or 8 cores)
- Nodes are *dedicated* during the course of the job
- Each allocated core should run a single thread (exception = per thread memory requirements)
- Kinds of Jobs: serial, parallel, multi-threaded, and *large memory*
- Most (almost all) jobs should be run in batch
- There is a maximum core allocation; all cores in a node are charged to you whether you use them or not

# Submitting a simple serial job

```
#!/bin/bash
#
# this file is myjob.sh
#
#PBS -N MyJob
#PBS -m be
#
myprog -n 100 < infile
```

```
% qsub –l nodes=1 myjob.sh
```

- -l nodes=1 is required (even for single node jobs)
- *myprog* output will appear in MyJob.oNNNNNN (STDIN) and MyJob.eNNNNNN (STDERR)

# Essential PBS Commands

- qsub …
- qdel [-Wforce] <jobid>
- qstat –a | -u <user> [-r] [-n] |-f <jobid>
- qselect –u <user> [-s R|Q]

# Serial Jobs, continued

What's inefficient about the previous example?

```
#!/bin/bash
#
# this file is myjob2.sh
#
#PBS -N MyJob
#PBS -m be
#
myprog -a 100 < infile1 > outfile1 &
myprog -a 200 < infile2 > outfile2 &
wait
```

# Submitting swarms of jobs with *swarm*

```
#
# this file is cmdfile
#
myprog -param a < infile-a > outfile-a
myprog -param b < infile-b > outfile-b
myprog -param c < infile-c > outfile-c
myprog -param d < infile-d > outfile-d
myprog -param e < infile-e > outfile-e
myprog -param f < infile-f > outfile-f
myprog -param g < infile-g > outfile-g
```

```
% swarm –f cmdfile
```

# swarms

- *swarm* will place your jobs on 8, 4, 2-core nodes depending on fit and availability
- 1000-line file (1000 threads) => 125 jobs (8-core) or 250 jobs (4-core) or 500 jobs (2-core)
- Entire swarm will be placed on one kind of node only
- Use semi-colons (;) to separate multiple commands on one line
- *swarm* expects tcsh syntax
- *swarmdel* command (see "*man swarmdel*")
- Create complex/long swarm command files with scripts

# Swarm bundles

- For large swarms or swarms with short-lived threads (under 5-10 minutes)
- `swarm –b 20 –f commandfile`
- creates a "bundle" (queue) of 20 threads *per core*
- 8000 lines (8000 threads), -b 20 => 50 jobs (8-core), 100 jobs (4-core), 200 jobs (2-core)
- Bundles *required* for command files with >10,000 lines

# Demo: Biowulf Applications

Checklist:

1. Is it parallel?
2. Is it multi-threaded?
3. How many independent processes do you need to run?
4. How much memory does one process require?
5. Licensed product? (Matlab)

Biowulf Applications webpages:
http://biowulf.nih.gov/apps/

# Demo: R on Biowulf

- Single-threaded
- Large swarms

Demo:

# Interactive Batch Jobs

```
% qsub –l nodes=1 –I [-V]
```

Reasons to use interactive nodes

- Testing/debugging cpu-intensive code
- Graphical application
- Compiling infiniband application

Reasons *not* to use interactive nodes

- Easier than setting up a batch job

## Parallel jobs

- The original driving force driving compute clusters
- N-communicating processes running on N-cores
- Shared-memory vs. distributed-memory
- Message-passing libraries (MPI, PVM)
- Communication and scalability
- High-performance interconnects

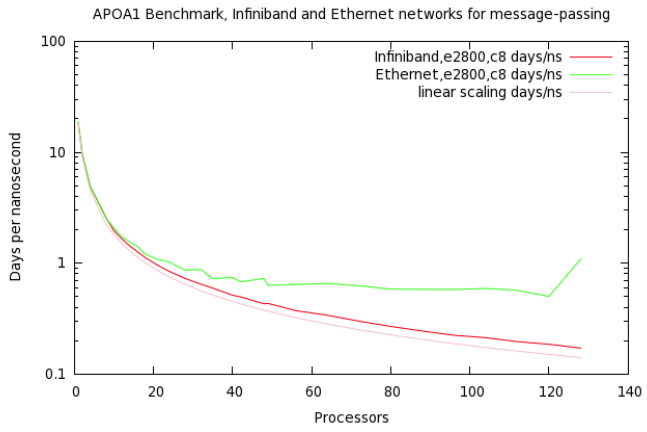
## High-performance Networks

- Parallel (MPI) applications only
- Benchmarking is essential
- Low latency more important than bandwidth
- Bypass TCP/IP stack
- Requires compilation against special libraries
- Two flavors available on Biowulf: Pathscale Infinipath & OFED Infiniband

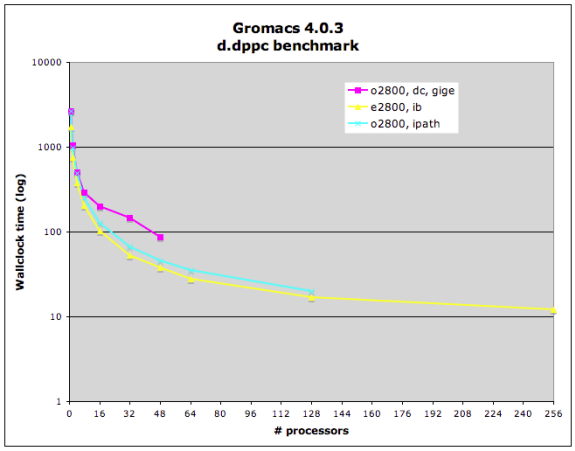
# Biowulf Cluster Networks

| | Gigabit ethernet | Infinipath (Pathscale) | Infiniband |
|---|---|---|---|
| Bandwidth (full duplex) | 1 Gb/s | 8 Gb/s (SDR) | 16 Gb/s (DDR) |
| Latency | "high" | "low" | "low" |
| TCP/IP? | Yes | No | No |
| # nodes | All | 132 | 250 |
| # cores | - | 264 | 2000 |

# NAMD Benchmarks



# GROMACS Benchmarks




# Monitoring Jobs with *jobload*

```
% jobload <jobid>|<user>
```

```
% jobload xxx4
Jobs for  xxx4       Node    Cores   Load
2104618.biobos       p41       2      99%
                     p42       2     100%
                     p43       2      99%
                     p45       2     100%
    Cores: 8   Job Load Average:    99%
2104620.biobos       p19       2     102%
                     p20       2      99%
                     p21       2      99%
                     p22       2     100%
    Cores: 8   Job Load Average:   100%

Core Total: 16     User Load Avg:  100.1%
```

```
% jobload cxx
Jobs for  cxx        Node    Cores   Load
2104868.biobos       p1723     4      99%
2104869.biobos       p1726     4      99%
2104870.biobos       p1728     4      99%
.
.
.
2104993.biobos       p1486     4     100%
2104994.biobos       p1487     4      99%
2104995.biobos       p1488     4      99%

Core Total: 512     User Load Avg:   99.9%
```

```
2106511.biobos       p1666     4      73%
                     p1668     4      70%
                     p1670     4      67%
                     p1672     4      80%
                     p1673     4      77%
                     p1722     4      83%
    Cores: 24   Job Load Average:    75%
```

# Poorly behaving jobs

```
% jobload yxxxxu
Jobs for  yxxxxu    Node    Cores   Load
2103919.biobos      p79       8      12%
                    p887      8       0%
                    p889      8       0%
                    p890      8       0%
    Cores: 32   Job Load Average:     3%
2105929.biobos      p1795     4      50%
2105931.biobos      p1492     4      75%
2105932.biobos      p1493     4      25%
2105933.biobos      p1495     4      50%
2105935.biobos      p1500     4     100%

Core Total: 52     User Load Avg:   34.7%
```

```
# jobload cxxxq
Jobs for  cxxxq     Node    Cores   Load
2105432.biobos      p880      8     299%
2105868.biobos      p490      2     100%

Core Total: 10     User Load Avg:  199.9%
```

# Biowulf User Monitor

- Alternative to *jobload* command
- Demo:

# Nodes for Large-memory Jobs

| Memory | 2-core Opterons | 4-core Opterons | 8-core Xeons | HP DL-785 (32-core Opteron) |
|---|---|---|---|---|
| 8 GB | 40 | 487 | | |
| 24 GB | | | 320 | |
| 72 GB | | | 32 | |
| 512 GB | | | | 1 |

# PBS Properties: node description & selection

| Property (resource)… | Selects… |
|---|---|
| e2666, e2800 | 2.666 (Nahalem)/2.8 GHz processors |
| o2800, o2600, o2200, o2000 | 2.8/2.6/2.2/2.0 GHz Opteron processors |
| c8, c4, c2 | 8/4/2 cores per node |
| m4, m2, m1 | 4/2/1 GB per core |
| g4, g8, g24, g72 | 4/8/24/72 GB per node |
| gige | gigabit ethernet |
| ib, ipath | infiniband, infinipath |
| dc (deprecated!) | dual-core node |

# Node Allocation & Selection

- Default allocation when using qsub: c2 (o2800 > o2200 > o2000)
- Default allocation when using swarm: c8 > c4 > c2
- Any other allocation requires selection based on properties:

```
% qsub –l nodes=1:g24 bigmemjob
% qsub –l nodes=8:o2800:gige namdjob
% qsub –l nodes=1:c8 tophatjob
% qsub –l nodes=4:ib –v np=32 namdjob
```

- Use properties only when needed!
- Properties are not orthagonal

# Available property combinations

| Processor | Cores | **m1** 1 GB mem. per core | **m2** 2 GB mem. per core | **m4** 4 GB mem. per core | **g4** 4 GB total mem. | **g8** 8 GB total mem. | **g24** 24 GB total mem. | **g72** 72 GB total mem. | Network |
|---|---|---|---|---|---|---|---|---|---|
| **e2666** 2.6 GHz Xeon | **c8** | | | | | | | | **gige** Gigabit Ethernet |
| **o2800:dc** 2.8 GHz Opteron | **c4** | | | | | | | | |
| **o2600:dc** 2.6 GHz Opteron | **c4** | | | | | | | | |
| **o2800** 2.8 GHz Opteron | **c2** | | | | | | | | |
| **o2200** 2.2 GHz Opteron | **c2** | | | | | | | | |
| **o2000** 2.0 GHz Opteron | **c2** | | | | | | | | |
| 2.8 GHz Xeon | 8 | | | | | | | | **ib** Infiniband |
| 2.8 GHz Opteron | 2 | | | | | | | | **ipath** Infinipath |

# Demo: NAMD on Biowulf

- Parallel program using MPI
- Typically low memory requirements (< 1GB/ core)
- Benefits from high-performance, low-latency network.

http://biowulf.nih.gov/apps/namd.html

Demo

## Node Availability

```
% freen
Free Nodes by memory-per-core
      Cores       m1      m2      m4   Total
------------------ DefaultPool -----------------------
o2800    2       /      8/210    /      8/210
o2200    2      3/300   0/62     /      3/362
o2000    2      0/38     /       /      0/38
------------------ Infiniband  -----------------------
ib       8    100/245    /       /    100/245
ipath    2       /     91/115    /     91/115
------------------  OnDemand   -----------------------
e2666    8       /      0/318    /      0/318
o2800:dc 4       /      1/210    /      1/210
o2800    2      0/5      /     38/40   38/45
o2600:dc 4      0/41   92/212    /     92/253


Free nodes by total node memory
      Cores       g4      g8     g24     g72   Total
------------------ DefaultPool -----------------------
o2800    2      8/210    /       /       /      8/210
o2200    2      0/62     /       /       /      0/62
------------------  OnDemand   -----------------------
e2666    8       /       /      0/318  11/32   11/350
o2800:dc 4       /      1/210    /       /      1/210
o2600:dc 4       /     92/253    /       /     92/253

--------------------  DL-785  -----------------------
Available: 31 processors, 504.2 GB memory
```

## Node Limits

```
% batchlim
             Max Cores
             Per User
            ---------
nist2       32
nist1       128
ib          288
g72         128
norm        512
gpu         32
burn        2700
ipath       128

            Max Mem    Max Mem
            Per User   on System
            ---------- -----------
DL-785 mem  512gb      512gb
```

## 512 GB node

- Shared node
- 32-cores, but single-core jobs using much/all of memory is ok
- To run a job use the “mem=“ resource:

```
% qsub –l nodes=1,mem=384gb bigmemjob
```

- mem must be >72gb, ≤ 500 gb
- Best not to use “mem=“ for any other jobs!

# Back to *swarm*: large-memory

- "m"-properties most useful for swarm jobs:

```
swarm –l nodes=1:m2 –f cmdfile
```

- When "m4" isn't enough:

```
swarm –n 1 –l nodes=1:g24 cmdfile
```

- Note that "-l nodes=1" is a place holder and has no other effect
- The "g" property is *required* in the example above
- Most nodes are at least "m1", so if that's enough, *don't specify a memory property!*

# Monitoring Memory Usage

```
% jobload –m <jobid>|<user>
```

```
Jobs for  jxxxxxn   Node    Cores   Load    Memory
2102742.biobos      p78       8     100%   23.0/74.2
2102743.biobos      p881      8     100%   36.3/74.2
2107026.biobos      p79       8     100%   26.1/74.2
```

```
# jobload -m jxxxxxxs
Jobs for  jxxxxxxs  Node    Cores   Load    Memory
                                          Used/Total (G)
1898152.biobos      p1747     4      25%    3.7/8.1
1898156.biobos      p1752     4      25%    3.7/8.1
1898157.biobos      p1753     4      25%    3.7/8.1
1898158.biobos      p1754     4      25%    3.7/8.1
1898162.biobos      p1758     4      25%    3.6/8.1
1898163.biobos      p1759     4      25%    3.7/8.1
1898164.biobos      p1762     4      25%    3.7/8.1
1898168.biobos      p1768     4      25%    3.6/8.1
1898171.biobos      p1772     4      25%    3.7/8.1
1898173.biobos      p1774     4      25%    3.7/8.1
1910428.biobos      p1708     4      25%    3.6/8.2

Core Total: 44     User Load Avg:   25.0%
```

```
Jobs for  jxxxxxxe  Node    Cores   Load    Memory
1964089.biobos      p69       8      37%    0.4/74.2
1964090.biobos      p71       8      37%    0.2/74.2
1964091.biobos      p72       8      37%    0.3/74.2
```

Bad!

# Monitoring Memory Usage

- *jobload* gives "instantaneous" usage
- What about maximum usage during the lifetime of the job? Or the "high-water" mark during a job?

```
% qstat -f 2109191 | grep mem
    resources_used.mem = 8392740kb      = 8.4 GB
    resources_used.vmem = 8613136kb     = 8.6 GB
```

```
From: adm
Date: October 13, 2010 9:39:29 AM EDT
To: user3@biobos.nih.gov
Subject: PBS JOB 2024340.biobos

PBS Job Id: 2024340.biobos
Job Name:   memhog2.bat
Execution terminated
Exit_status=0
resources_used.cpupercent=111
resources_used.cput=00:18:32
resources_used.mem=7344768kb                    = 7.3 GB
resources_used.ncpus=1
resources_used.vmem=7494560kb                   = 7.5 GB
resources_used.walltime=00:17:05
```

# Monitoring Memory Usage

- swap space – disk-based extension of memory
- Swapping (paging) jobs should be killed and resubmitted for larger memory nodes

```
% jobload -m luser
Jobs for  luser     Node    Cores   Load    Memory
                                          Used/Total (G)
2206035.biobos      p132      2     142%    1.0/1.0
2206065.biobos      p149      2     136%    1.0/1.0
```

```
% rsh p132 free
             total       used       free     shared    buffers     cached
Mem:       1026532    1020080       6452          0        568       7692
-/+ buffers/cache:    1011820      14712
Swap:      4192956    2482740    1710216
```

## Multi-threaded swarms

```
% swarm –n 1 –l nodes=1:c8 –f bowtie.swarm
% swarm –n 2 –l nodes=1:c4 –f blast.swarm
```

- "-n" switch to swarm requires an additional specification of either "-g" for large-memory jobs or "-c" for multi-threaded jobs (or both)

## Demo: Bowtie on Biowulf

- Multi-threaded
- Can require lots of memory

http://biowulf.nih.gov/apps/bowtie.html

Demo.

# Demo: Matlab on Biowulf

- Can be multi-threaded.
- License-limited

http://biowulf.nih.gov/apps/matlab.html

Demo:

New class: "NIH Biowulf Cluster Advanced Topics", Jan 19, 2011 1-3 pm

- What's new/changed in the last 12 months
- How to benchmark application code
- Optimal selection of nodes
- Methods for monitoring jobs and memory utilization
- High-performance data transfer
- Advanced PBS (job dependencies)
- Advanced use of the swarm command
- GPU cluster
- Compiling code on Biowulf
- Running parallel jobs on Biowulf
- Hyperthreading
- *What would you like to see covered?*