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

# NIH Biowulf Cluster: Advanced Topics

Steven Fellini, Ph.D sfellini@nih.gov
Susan Chacko, Ph.D susanc@helix.nih.gov
David Hoover, Ph.D hooverdm@helix.nih.gov

CIT
January 19, 2011

1

## Topics for Today

- Recent Hardware Upgrades
- 2010 Statistics
- High-performance Data Transfer
- Advanced PBS Batch Features
- Advanced *swarm* Features
- Benchmarking Parallel Codes
- GPU nodes
- Hyperthreading
- Optimal Node Selection
- Monitoring Memory Usage
- Reminders
- Tour of Biowulf Machine Room

This presentation available as
http://biowulf.nih.gov/AdvBiowulf-Jan2011.pdf

2

# Helix & Biowulf Walk-in Consults

- Tuesday January 25, 2011, 9 am – noon
- Building 12B, 2nd Floor Conference Room

3

## Recent Hardware Upgrades

FY2008

- Infiniband 4X DDR
- +250 nodes (2000 cores)

FY2009

- High-performance storage system 600 TB (raw)

- +Large-memory compute nodes: 72 GB & 512 GB

FY2010

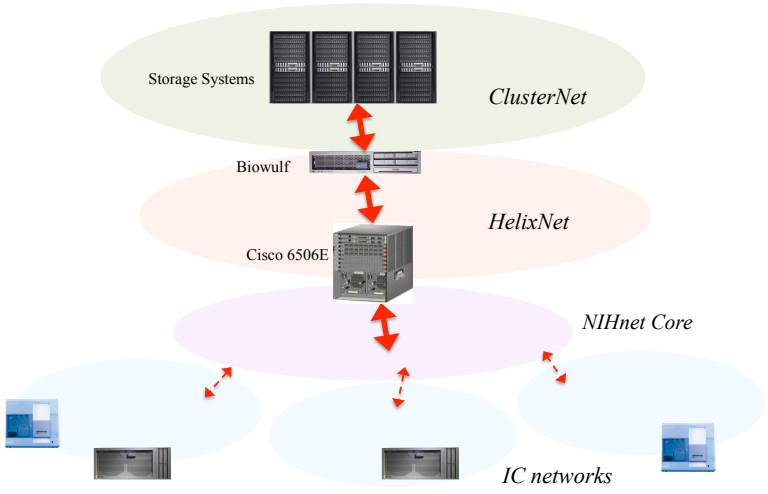
- +336 nodes (2688 cores)

- Replace ClusterNet core switch

- Expand storage system by 600 TB (raw)

- 800 TB (raw) storage system

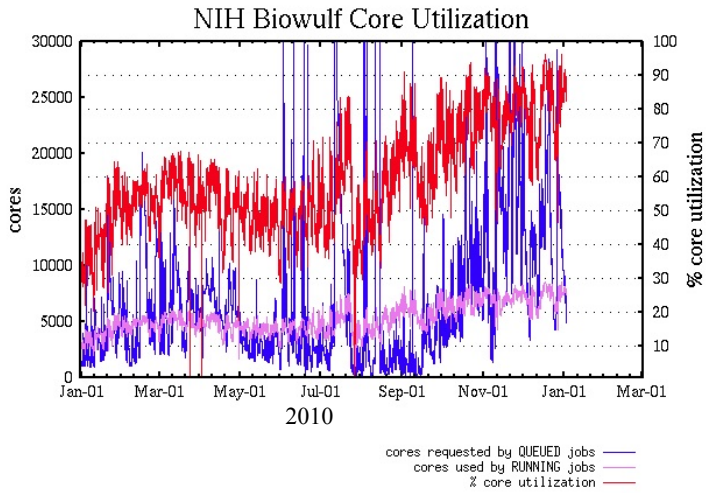
## FY10: 10Gb/s Network Support to NIH Core



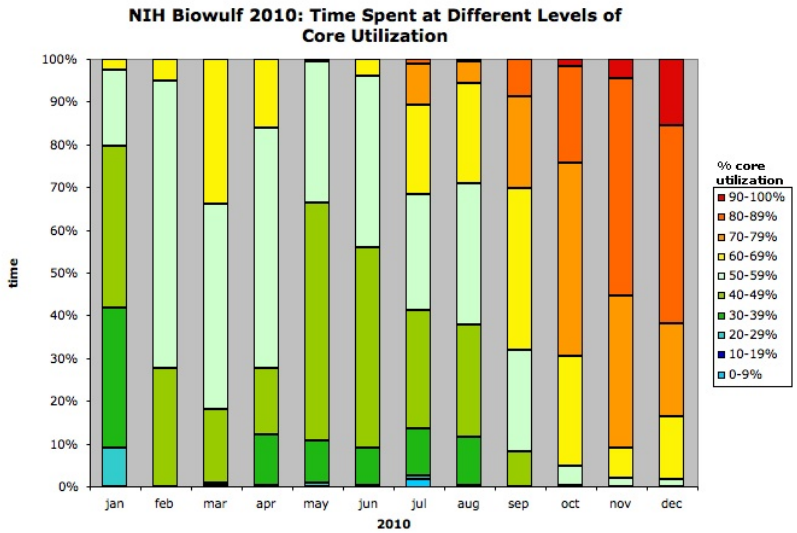

5

## FY11: Compute Nodes ?

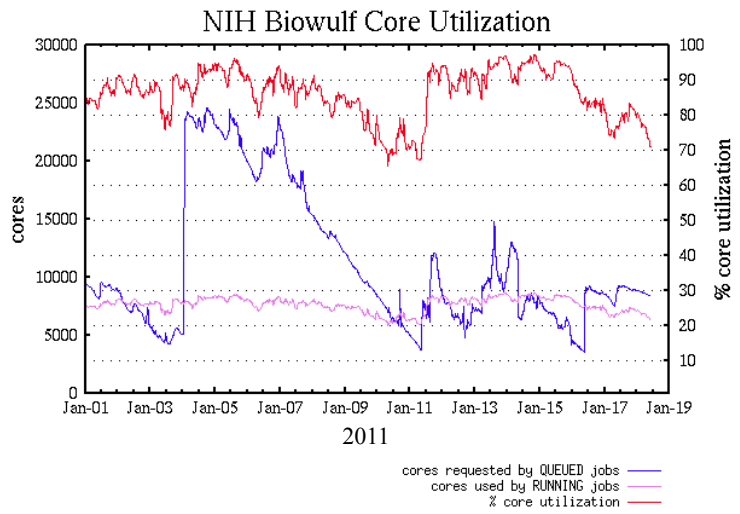
6


NIH Biowulf Core Utilization
cores
% core utilization
30000
25000
20000
15000
10000
5000
0
100
90
80
70
60
50
40
30
20
10
Jan-01
Mar-01
May-01
Jul-01
Sep-01
Nov-01
Jan-01
Mar-01
2010
cores requested by QUEUED jobs
cores used by RUNNING jobs
% core utilization

7


NIH Biowulf 2010: Time Spent at Different Levels of Core Utilization
time
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
jan
feb
mar
apr
may
jun
jul
aug
sep
oct
nov
dec
2010
% core utilization
90-100%
80-89%
70-79%
60-69%
50-59%
40-49%
30-39%
20-29%
10-19%
0-9%

8




9

# High-Performance Data Transfer Using *scp-hpn*

- Modified version of *scp* to be used on your Linux workstation or server
- Push/Pull files from your workstation or server to/from biowulf.nih.gov
- Download from http://biowulf.nih.gov/hpn.tar.gz into your home directory
- Syntax is identical to *scp*

# scp-hpn

```
$ tar xfvp hpn.tar.gz
scp-hpn
ssh-hpn
README

$ ./scp-hpn
usage: scp [-1246BCpqrv] [-c cipher] [-F ssh_config] [-i identity_file]
           [-l limit] [-o ssh_option] [-P port] [-S program]
           [[user@]host1:]file1 ... [[user@]host2:]file2

$ ./scp-hpn README steve@biowulf.nih.gov:/home/steve
steve@biowulf.nih.gov's password:
README                              100%  798     0.8KB/s   0.8KB/s   00:00
```

# hpn-scp Performance

| | 1 Gb/s | | 10 Gb/s | |
|---|---|---|---|---|
| | Mb/s | MB/s | Mb/s | MB/s |
| scp | 340.4 | 42.6 | 411.2 | 51.4 |
| scp-hpn | 824.8 | 103.1 | 1844.4 | 230.6 |
| increase | 2.4X | | 4.5X | |

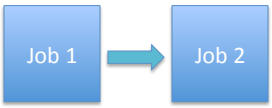
# Advanced PBS: Job Dependencies

Jobs that depend on the completion of other jobs.

Examples:

- A 8-threaded job on a single node followed by a parallel job on 80 cores
- A data-processing swarm job followed by a single summary job.

# Job Dependencies




qsub –W depend=afterany:Job1 –l nodes=1 Job2script

Job2 will run after Job1 has exited, with or without errors.

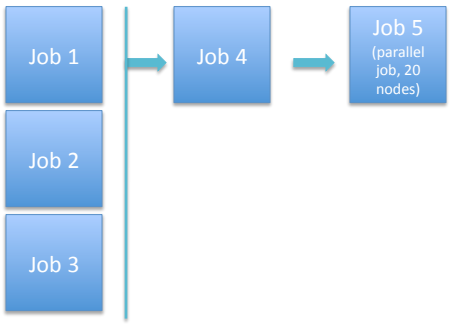
after (after Job1 has started)
afterok (after Job1 has completed successfully)
afternotok (after Job 1 has terminated with errors)
Beforeok, beforeany, beforenotok

```
[susanc@biowulf ~]$ qstat -u susanc

biobos:
                                                                  Req'd  Req'd   Elap
Job ID          Username Queue    Jobname    SessID NDS TSK Memory Time  S Time
--------------- -------- -------- ---------- ------ --- --- ------ ----- - -----
2484503.biobos  susanc   norm     Job1  13686   1   1    --    --  R 00:00
2484504.biobos  susanc   norm     Job2     --   1   1    --    --  H   --
```

```
[susanc@biowulf ~]$ qstat -f 2484504
    Job Id: 2484504.biobos
    Job_Name = sw1n17048
    Job_Owner = susanc@biowulf-e1
    job_state = H
    queue = norm
    server = biobos
    Checkpoint = u
    ctime = Mon Jan 10 09:44:55 2011
    depend = afterok:2484503.biobos
    […]
```

# Job Dependencies




qsub –W depend=afterok:Job1:Job2:Job3 -l nodes=1 Job4script
qsub –W depend=afterok:Job4 –l nodes=20:o2800 Job5script

```
[susanc@biowulf ~]$ qstat -u susanc

biobos:
                                                            Req'd  Req'd   Elap
Job ID          Username Queue    Jobname    SessID NDS TSK Memory Time  S Time
--------------- -------- -------- ---------- ------ --- --- ------ ----- - -----
2484503.biobos  susanc   norm     Job1        13686   1   1    --    --  R 00:00
2484504.biobos  susanc   norm     Job2          --    1   1    --    --  R 00:00
2484505.biobos  susanc   norm     Job3          --    1   1    --    --  Q --
2484506.biobos  susanc   norm     Job4          --    1   1    --    --  H --
2484507.biobos  susanc   norm     Job5          --   20   1    --    --  H --
```

```
[susanc@biowulf ~]$ qstat -f 2484506
    Job Id: 2484504.biobos
    Job_Name = sw1n17048
    Job_Owner = susanc@biowulf-e1
    job_state = H
    queue = norm
    server = biobos
    Checkpoint = u
    ctime = Mon Jan 10 09:44:55 2011
    depend = afterok:2484503.biobos:2484504.biobos:2484505.biobos,
             beforeok:2484507.biobos
    […]
```

```
#!/bin/bash
FIRST=`qsub first_1.sh`
echo $FIRST
SECOND=`qsub -W depend=afterok:$FIRST second_1.sh`
echo $SECOND
THIRD=`qsub -W depend=afterok:$SECOND third_1.sh`
echo $THIRD
FOURTH=`qsub -W depend=afterok:$THIRD fourth_1.sh`
echo $FOURTH
```

Caveats:

If you delete Job1, Job 2 will remain in the queue forever.
PBS completion status is shaky.

# Advanced PBS: Job Arrays

- Many jobs based on the same job script.
- One qsub command to create/delete all the jobs.

```
qsub –J 0-4 –l nodes=1 job_script
```

-> 5 jobs in a single job array

-> $PBS_ARRAY_INDEX set to index number 0,1,2,3,4

```
qsub –J 10-14 –l nodes=1 job_script
```

-> 5 jobs in a single job array

-> $PBS_ARRAY_INDEX set to index number 10,11,12,13,14

# Job Arrays

Demo

```
biowulf% qstat –u username
biobos:
                                                          Req'd  Req'd   Elap
Job ID          Username Queue    Jobname    SessID NDS TSK Memory Time  S Time
--------------- -------- -------- ---------- ------ --- --- ------ ----- - -----
2485230[].biobo teacher  norm     jobscript      --    1   1    --    --  Q   --
```

```
2485230[].biobo teacher  norm     jobscript      --    1   1    --    --  B   1:00
```

```
biowulf% qstat –u username –t
biobos:
                                                          Req'd  Req'd   Elap
Job ID          Username Queue    Jobname    SessID NDS TSK Memory Time  S Time
--------------- -------- -------- ---------- ------ --- --- ------ ----- - -----
2485230[0].biob teacher  norm     jobscript      --    1   1    --    --  X   --
2485230[1].biob teacher  norm     jobscript      --    1   1    --    --  R  0:00
2485230[2].biob teacher  norm     jobscript      --    1   1    --    --  R  0:00
2485230[3].biob teacher  norm     jobscript      --    1   1    --    --  Q   --
2485230[4].biob teacher  norm     jobscript      --    1   1    --    --  Q   --
```

# Job Arrays

```
#!/bin/bash

echo $HOSTNAME
echo $PBS_ARRAY_INDEX
```

```
biowulf% qsub –J 0-4 –l nodes=1 jobscript
2485230[].biobos
```

```
biowulf% more jobscript.o2485230.3
p1323
3
-------- running PBS epilogue script (2485230[3].biobos p1323 teacher) --------
housekeeping: p1323
------------- done PBS epilogue script -------------
```

# Job Arrays

Example: 100 directories containing data to be processed.

```
biowulf% ls /data/user/raw_data/
NGS_mouse_200
NGS_mouse_202
NGS_mouse_204
[…]
NGS_mouse_400
```

# Job Arrays

```
#!/bin/bash
# This file is arrayscript
# runs Bowtie in multithreaded mode for a single set of data
#PBS options

cd /data/user/raw_data/NGS_mouse_$PBS_ARRAY_INDEX
bowtie –t –s –p $np hg18 file1.fastq file2.fastq
```

```
qsub –J 200-400:2 –v np=8 –l nodes=1:c8 arrayscript
```

| | |
|---|---|
| `-J 200-400:2` | (array index goes from 200 – 400 in steps of 2) |
| `-v np=8` | (variable called $np that is passed to the script. It tells Bowtie how many threads to run) |
| `-l nodes=1:c8` | (8 threads need to be run on nodes with 8 cores) |

# Job Arrays

Limitations:

- Single script
- Only suitable for many iterations of the same script
- Only one variable $PBS_ARRAY_INDEX

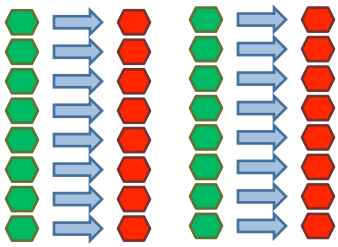
Swarm is more flexible, but job arrays are convenient to consolidate a lot of jobs.

# Advanced Use of Swarm

## swarm

- Commands are written in a single file
- A line can have multiple serial commands (;)

```
command < 1.txt > 1.out
command < 2.txt > 2.out
command < 3.txt > 3.out
command < 4.txt > 4.out
command < 5.txt > 5.out
command < 6.txt > 6.out
command < 7.txt > 7.out
command < 8.txt > 8.out
command < 9.txt > 9.out
command < 10.txt > 10.out
command < 11.txt > 11.out
command < 12.txt > 12.out
command < 13.txt > 13.out
command < 14.txt > 14.out
command < 15.txt > 15.out
command < 16.txt > 16.out
```



# Standard swarm Options

- -f, --file [file]

  File containing commands to execute, one process per line

- -n, --procpernode [int]

  Number of processes to run per node

- -b, --bundle [int]

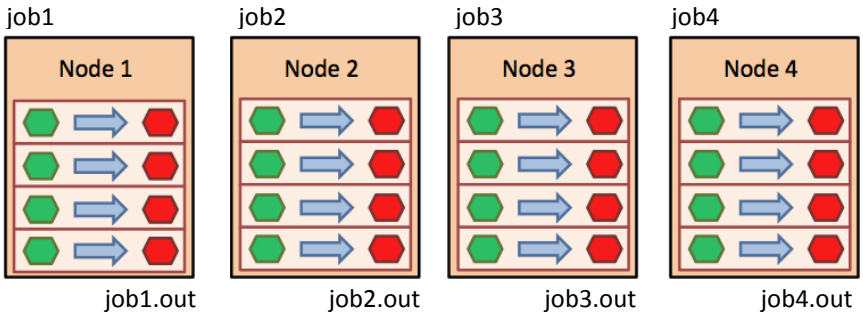
  Number of processes to assign to run serially on each core

- -d, --debug [int]

  Show what commands would have run, preserve scripts, don't submit

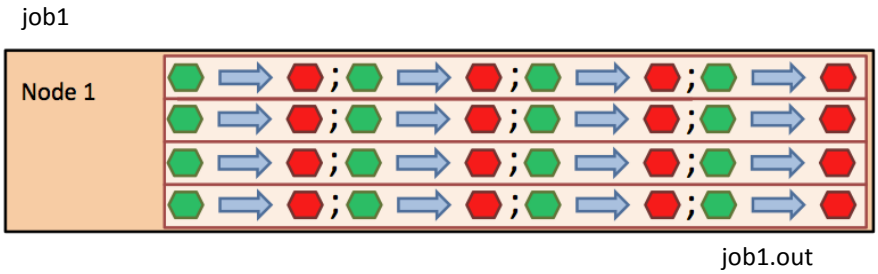
# swarm

```
[biowulf]$ swarm -f file
```



# Bundled swarm

**[biowulf]$ swarm -f file -b 4**




# --test Option

- See what might happen

```
[biowulf]$ swarm -f swarmfile -b 2 --test
[main] command file is 'swarmfile'
[main] -b commands per processor is '2'

  Node: Free  Cores /  All  Cores
----------------------------------
    c2:  162 ( 324) /  779 (1558)
    c4:   51 ( 204) /  476 (1904)
    c8:    5 (  40) /  318 (2544)

[main] loptions is '-l nodes=1:c2'
[main] 1000 commands on 250 c2 nodes using 2 procs per node
```

# Under The swarm Hood

- swarm creates a subdirectory .swarm

```
[biowulf]$ ls .swarm
cmd-swarmb1n0p1816  cmd-swarmb1n5p1816  cmd-swarmb2n2p1816  cmd-swarmb2n7p1816
cmd-swarmb1n1p1816  cmd-swarmb1n6p1816  cmd-swarmb2n3p1816  ctrl-swarmb1n1816
cmd-swarmb1n2p1816  cmd-swarmb1n7p1816  cmd-swarmb2n4p1816  ctrl-swarmb2n1816
cmd-swarmb1n3p1816  cmd-swarmb2n0p1816  cmd-swarmb2n5p1816
cmd-swarmb1n4p1816  cmd-swarmb2n1p1816  cmd-swarmb2n6p1816
```

- ctrl- scripts launch sets of cmd- scripts (only with bundled swarm)
- cmd- scripts do the heavy lifting

# Under The swarm Hood

- swarm runs under /bin/tcsh by default
- Commands are run in current working directory

```
$ head .swarm/cmd-swarmb1n1p1816
#!/bin/tcsh
#
# [script generated by swarm Jan 11 2011 14:12:09]
#
# this file is /data/user/swarm/.swarm/cmd-swarmb1n1p1816
#
cd /data/user/swarm
command < 1.in > 1.out
command < 2.in > 2.out
command < 3.in > 3.out
```

# Under The swarm Hood

- Jobnames have format sw[#]n[PID]
- Bundled swarms use swb[#]n[PID]

```
[biowulf]$ qstat -u user

biobos:
                                                          Req'd  Req'd   Elap
Job ID          Username Queue    Jobname    SessID NDS TSK Memory Time  S Time
--------------- -------- -------- ---------- ------ --- --- ------ ----- - -----
2502073.biobos  user     norm     sw1n11558   20372   1   1    --    --  R 00:23
2502074.biobos  user     norm     sw2n11558    8808   1   1    --    --  R 00:23
2502075.biobos  user     norm     sw3n11558   17620   1   1    --    --  R 00:23
2502076.biobos  user     norm     sw4n11558   11130   1   1    --    --  R 00:23
2502077.biobos  user     norm     sw5n11558   22779   1   1    --    --  R 00:23
2502078.biobos  user     norm     sw6n11558   11058   1   1    --    --  R 00:23
2502079.biobos  user     norm     sw7n11558    2796   1   1    --    --  R 00:23
2502080.biobos  user     norm     sw8n11558   24265   1   1    --    --  R 00:23
```

# Under The swarm Hood

- By default swarm chooses nodes based on what is available (c8 then c4 then c2)
- If nothing is available, then swarm chooses randomly
- `-n` is required if node type is specified with `-l nodes=1`

```
-l nodes=1:g24 -n 4
-l nodes=1:c4 -n 1
```

- `-n` not required for memory properties

```
-l nodes=1:m4
```

## Stopping a swarm

- swarmdel acts like qdel, but for a set of swarm jobs

```
[biowulf]$ swarmdel sw4n12345
```

- Will kill all jobs with name format `sw#n12345` and owned by user running swarmdel

## swarm Hidden Options

- There are hidden swarm options, somewhat still experimental

--usebash
--autobundle
--prologue
--epilogue
--singleout
--jobarray

## swarm Hidden Options

- `--usebash` causes swarm to run under /bin/bash instead of /bin/tcsh

```
export ENV_VAR=10; command > out 2>&1
export ENV_VAR=20; command > out 2>&1
export ENV_VAR=30; command > out 2>&1
export ENV_VAR=40; command > out 2>&1
export ENV_VAR=50; command > out 2>&1
export ENV_VAR=60; command > out 2>&1
export ENV_VAR=70; command > out 2>&1
export ENV_VAR=80; command > out 2>&1
```

- Uses ~/.bashrc, ~/.bash_profile, and ~/.profile instead of ~/.cshrc, ~/.tcshrc, and ~/.login

## swarm Hidden Options

- `--autobundle` automatically determines a bundle value and runs bundled swarm
- `-b` value depends on max number of cores available per user
- Useful for massive (>10,000 commands) swarms

## swarm Hidden Options

- `--prologue` and `--epilogue` allows a script to be run once per job
- Especially useful for `clearscratch` and other methods dealing with local /scratch files

## swarm Hidden Options

- --singleout squishes all the swarm output and error files into one file
- from

```
[biowulf]$ ls
sw1n1234.o  sw3n1234.o  sw5n1234.o  sw7n1234.o
sw1n1234.e  sw3n1234.e  sw5n1234.e  sw7n1234.e
sw2n1234.o  sw4n1234.o  sw6n1234.o  sw8n1234.o
sw2n1234.e  sw4n1234.e  sw6n1234.e  sw8n1234.e
```

- to

```
[biowulf]$ ls
sw0n1234.o  sw0n1234.e
```

# swarm Hidden Options

- --jobarray is a very powerful option that allow a swarm to be run as a single job
- Useful for job dependencies

```
[biowulf]$ qsub -l nodes=1 first.sh 1001.biobos
[biowulf]$ swarm --jobarray -f swarmfile.txt \
  -W depend=afterany:1001.biobos 1002[].biobos
[biowulf]$ qsub -l nodes=1 -W depend=afterany:1002[].biobos last.sh 1003.biobos
```

- Job id different format: `1002[].biobos`
- Monitoring different as well, must use `-t` option for qstat to display job array subjobs

# Effect of --jobarray

```
[biowulf]$ qstat -u user

biobos:
                                                                Req'd  Req'd   Elap
Job ID          Username Queue    Jobname    SessID NDS TSK Memory Time  S Time
--------------- -------- -------- ---------- ------ --- --- ------ ----- - -----
2505306[].biobo user     norm     sw0n17837     --    1   1    --    --  B  --

[biowulf]$ qstat -u user -t

biobos:
                                                                Req'd  Req'd   Elap
Job ID          Username Queue    Jobname    SessID NDS TSK Memory Time  S Time
--------------- -------- -------- ---------- ------ --- --- ------ ----- - -----
2505306[1].biob user     norm     sw0n17837   15969   1   1    --    --  X  --
2505306[2].biob user     norm     sw0n17837    2806   1   1    --    --  X  --
2505306[3].biob user     norm     sw0n17837   26076   1   1    --    --  R 00:05
2505306[4].biob user     norm     sw0n17837    5459   1   1    --    --  R 00:05
2505306[5].biob user     norm     sw0n17837   21522   1   1    --    --  R 00:05
2505306[6].biob user     norm     sw0n17837   23018   1   1    --    --  R 00:05
2505306[7].biob user     norm     sw0n17837   24090   1   1    --    --  R 00:05
2505306[8].biob user     norm     sw0n17837   29353   1   1    --    --  R 00:05
2505306[9].biob user     norm     sw0n17837   18617   1   1    --    --  Q  --
2505306[10].bio user     norm     sw0n17837   18617   1   1    --    --  Q  --
```

# Effect of --jobarray

- The output files are different as well:

```
[biowulf]$ ls
sw0n32691.e.10  sw0n32691.e.18  sw0n32691.e.7   sw0n32691.o.14  sw0n32691.o.3
sw0n32691.e.11  sw0n32691.e.19  sw0n32691.e.8   sw0n32691.o.15  sw0n32691.o.4
sw0n32691.e.12  sw0n32691.e.2   sw0n32691.e.9   sw0n32691.o.16  sw0n32691.o.5
sw0n32691.e.13  sw0n32691.e.20  sw0n32691.o.1   sw0n32691.o.17  sw0n32691.o.6
sw0n32691.e.14  sw0n32691.e.3   sw0n32691.o.10  sw0n32691.o.18  sw0n32691.o.7
sw0n32691.e.15  sw0n32691.e.4   sw0n32691.o.11  sw0n32691.o.19  sw0n32691.o.8
sw0n32691.e.16  sw0n32691.e.5   sw0n32691.o.12  sw0n32691.o.2   sw0n32691.o.9
sw0n32691.e.1   sw0n32691.e.17  sw0n32691.e.6   sw0n32691.o.13  sw0n32691.o.20
```

# Effect of --jobarray

- Stopping job array swarm is simple

```
[biowulf]$ qdel 2505306[]
```

# Effect of --jobarray

- Not all monitoring tools understand different format

# Supercomplex swarm Example

```
[biowulf]$ qsub -l nodes=1 makeconfig.sh
123456.biobos
[biowulf]$ export PATH=/data/user/funkyapp/bin:$PATH
[biowulf]$ cat swarmfile
cd /scratch; funkycmd -in /data/user/config -seed 1; funkyscript.sh 1
cd /scratch; funkycmd -in /data/user/config -seed 2; funkyscript.sh 2
cd /scratch; funkycmd -in /data/user/config -seed 3; funkyscript.sh 3
cd /scratch; funkycmd -in /data/user/config -seed 4; funkyscript.sh 4
...
cd /scratch; funkycmd -in /data/user/config -seed 256; funkyscript.sh 256
[biowulf]$ swarm -f swarmfile -l nodes=1:g24 -n 2 -b 4 --prologue /usr/
local/bin/clearscratch --epilogue 'funkymerge.sh -dir /data/user' -V -W
depend=afterany:123456.biobos --jobarray --singleout --usebash
123489[].biobos
[biowulf]$ qsub -l nodes=1:g72 -W depend=afterany:123489[].biobos
finalanalysis.sh
```

# Supercomplex swarm Example

```
[biowulf]$ qsub -l nodes=1 makeconfig.sh
123456.biobos
[biowulf]$ export PATH=/data/user/funkyapp/bin:$PATH
[biowulf]$ cat swarmfile
cd /scratch; funkycmd -in /data/user/config -seed 1; funkyscript.sh 1
cd /scratch; funkycmd -in /data/user/config -seed 2; funkyscript.sh 2
cd /scratch; funkycmd -in /data/user/config -seed 3; funkyscript.sh 3
cd /scratch; funkycmd -in /data/user/config -seed 4; funkyscript.sh 4
...
cd /scratch; funkycmd -in /data/user/config -seed 256; funkyscript.sh 256
[biowulf]$ swarm -f swarmfile -l nodes=1:g24 -n 2 -b 4 --prologue /usr/
local/bin/clearscratch --epilogue 'funkymerge.sh -dir /data/user' -V -W
depend=afterany:123456.biobos --jobarray --singleout --usebash
123489[].biobos
[biowulf]$ qsub -l nodes=1:g72 -W depend=afterany:123489[].biobos
finalanalysis.sh
```

Create the config file based on prewritten script makeconfig.sh

# Supercomplex swarm Example

```
[biowulf]$ qsub -l nodes=1 makeconfig.sh
123456.biobos
[biowulf]$ export PATH=/data/user/funkyapp/bin:$PATH
[biowulf]$ cat swarmfile
cd /scratch; funkycmd -in /data/user/config -seed 1; funkyscript.sh 1
cd /scratch; funkycmd -in /data/user/config -seed 2; funkyscript.sh 2
cd /scratch; funkycmd -in /data/user/config -seed 3; funkyscript.sh 3
cd /scratch; funkycmd -in /data/user/config -seed 4; funkyscript.sh 4
...
cd /scratch; funkycmd -in /data/user/config -seed 256; funkyscript.sh 256
[biowulf]$ swarm -f swarmfile -l nodes=1:g24 -n 2 -b 4 --prologue /usr/
local/bin/clearscratch --epilogue 'funkymerge.sh -dir /data/user' -V -W
depend=afterany:123456.biobos --jobarray --singleout --usebash
123489[].biobos
[biowulf]$ qsub -l nodes=1:g72 -W depend=afterany:123489[].biobos
finalanalysis.sh
```

Add the funkyapp bin directory to your path for easy access. You could add it to your .bashrc file, but this causes interference with other activities.

# Supercomplex swarm Example

```
[biowulf]$ qsub -l nodes=1 makeconfig.sh
123456.biobos
[biowulf]$ export PATH=/data/user/funkyapp/bin:$PATH
[biowulf]$ cat swarmfile
cd /scratch; funkycmd -in /data/user/config -seed 1; funkyscript.sh 1
cd /scratch; funkycmd -in /data/user/config -seed 2; funkyscript.sh 2
cd /scratch; funkycmd -in /data/user/config -seed 3; funkyscript.sh 3
cd /scratch; funkycmd -in /data/user/config -seed 4; funkyscript.sh 4
...
cd /scratch; funkycmd -in /data/user/config -seed 256; funkyscript.sh 256
[biowulf]$ swarm -f swarmfile -l nodes=1:g24 -n 2 -b 4 --prologue /usr/
local/bin/clearscratch --epilogue 'funkymerge.sh -dir /data/user' -V -W
depend=afterany:123456.biobos --jobarray --singleout --usebash
123489[].biobos
[biowulf]$ qsub -l nodes=1:g72 -W depend=afterany:123489[].biobos
finalanalysis.sh
```

You are a clever user who uses the local /scratch directory for temporary files. This avoids any quota overruns, and may speed up the execution by reducing the load on fileservers and network.

# Supercomplex swarm Example

```
[biowulf]$ qsub -l nodes=1 makeconfig.sh
123456.biobos
[biowulf]$ export PATH=/data/user/funkyapp/bin:$PATH
[biowulf]$ cat swarmfile
cd /scratch; funkycmd -in /data/user/config -seed 1; funkyscript.sh 1
cd /scratch; funkycmd -in /data/user/config -seed 2; funkyscript.sh 2
cd /scratch; funkycmd -in /data/user/config -seed 3; funkyscript.sh 3
cd /scratch; funkycmd -in /data/user/config -seed 4; funkyscript.sh 4
...
cd /scratch; funkycmd -in /data/user/config -seed 256; funkyscript.sh 256
[biowulf]$ swarm -f swarmfile -l nodes=1:g24 -n 2 -b 4 --prologue /usr/
local/bin/clearscratch --epilogue 'funkymerge.sh -dir /data/user' -V -W
depend=afterany:123456.biobos --jobarray --singleout --usebash
123489[].biobos
[biowulf]$ qsub -l nodes=1:g72 -W depend=afterany:123489[].biobos
finalanalysis.sh
```

Because funkycmd and funkyscript.sh will be in your path, there is no need to give the entire path to the executables. You are also using the config file generated in the earlier batch job 123456.biobos.

# Supercomplex swarm Example

```
[biowulf]$ qsub -l nodes=1 makeconfig.sh
123456.biobos
[biowulf]$ export PATH=/data/user/funkyapp/bin:$PATH
[biowulf]$ cat swarmfile
cd /scratch; funkycmd -in /data/user/config -seed 1; funkyscript.sh 1
cd /scratch; funkycmd -in /data/user/config -seed 2; funkyscript.sh 2
cd /scratch; funkycmd -in /data/user/config -seed 3; funkyscript.sh 3
cd /scratch; funkycmd -in /data/user/config -seed 4; funkyscript.sh 4
...
cd /scratch; funkycmd -in /data/user/config -seed 256; funkyscript.sh 256
[biowulf]$ swarm -f swarmfile -l nodes=1:g24 -n 2 -b 4 --prologue /usr/
local/bin/clearscratch --epilogue 'funkymerge.sh -dir /data/user' -V -W
depend=afterany:123456.biobos --jobarray --singleout --usebash
123489[].biobos
[biowulf]$ qsub -l nodes=1:g72 -W depend=afterany:123489[].biobos
finalanalysis.sh
```

Each job requires 12GB of memory, so you wisely allocate two processes per 24GB node.

# Supercomplex swarm Example

```
[biowulf]$ qsub -l nodes=1 makeconfig.sh
123456.biobos
[biowulf]$ export PATH=/data/user/funkyapp/bin:$PATH
[biowulf]$ cat swarmfile
cd /scratch; funkycmd -in /data/user/config -seed 1; funkyscript.sh 1
cd /scratch; funkycmd -in /data/user/config -seed 2; funkyscript.sh 2
cd /scratch; funkycmd -in /data/user/config -seed 3; funkyscript.sh 3
cd /scratch; funkycmd -in /data/user/config -seed 4; funkyscript.sh 4
...
cd /scratch; funkycmd -in /data/user/config -seed 256; funkyscript.sh 256
[biowulf]$ swarm -f swarmfile -l nodes=1:g24 -n 2 -b 4 --prologue /usr/
local/bin/clearscratch --epilogue 'funkymerge.sh -dir /data/user' -V -W
depend=afterany:123456.biobos --jobarray --singleout --usebash
123489[].biobos
[biowulf]$ qsub -l nodes=1:g72 -W depend=afterany:123489[].biobos
finalanalysis.sh
```

The swarm is bundled to improve efficiency of node use.

# Supercomplex swarm Example

```
[biowulf]$ qsub -l nodes=1 makeconfig.sh
123456.biobos
[biowulf]$ export PATH=/data/user/funkyapp/bin:$PATH
[biowulf]$ cat swarmfile
cd /scratch; funkycmd -in /data/user/config -seed 1; funkyscript.sh 1
cd /scratch; funkycmd -in /data/user/config -seed 2; funkyscript.sh 2
cd /scratch; funkycmd -in /data/user/config -seed 3; funkyscript.sh 3
cd /scratch; funkycmd -in /data/user/config -seed 4; funkyscript.sh 4
...
cd /scratch; funkycmd -in /data/user/config -seed 256; funkyscript.sh 256
[biowulf]$ swarm -f swarmfile -l nodes=1:g24 -n 2 -b 4 --prologue /usr/
local/bin/clearscratch --epilogue 'funkymerge.sh -dir /data/user' -V -W
depend=afterany:123456.biobos --jobarray --singleout --usebash
123489[].biobos
[biowulf]$ qsub -l nodes=1:g72 -W depend=afterany:123489[].biobos
finalanalysis.sh
```

Prior to the commands being run on a node, clear the /scratch area. Since the intermediate files will be very large, this will allow enough diskspace. After all the commands have run, merge the intermediate files and transfer them to your /data directory.

# Supercomplex swarm Example

```
[biowulf]$ qsub -l nodes=1 makeconfig.sh
123456.biobos
[biowulf]$ export PATH=/data/user/funkyapp/bin:$PATH
[biowulf]$ cat swarmfile
cd /scratch; funkycmd -in /data/user/config -seed 1; funkyscript.sh 1
cd /scratch; funkycmd -in /data/user/config -seed 2; funkyscript.sh 2
cd /scratch; funkycmd -in /data/user/config -seed 3; funkyscript.sh 3
cd /scratch; funkycmd -in /data/user/config -seed 4; funkyscript.sh 4
...
cd /scratch; funkycmd -in /data/user/config -seed 256; funkyscript.sh 256
[biowulf]$ swarm -f swarmfile -l nodes=1:g24 -n 2 -b 4 --prologue /usr/
local/bin/clearscratch --epilogue 'funkymerge.sh -dir /data/user' -V -W
depend=afterany:123456.biobos --jobarray --singleout --usebash
123489[].biobos
[biowulf]$ qsub -l nodes=1:g72 -W depend=afterany:123489[].biobos
finalanalysis.sh
```

Import all environment variables, especially $PATH, to the swarm jobs.

# Supercomplex swarm Example

```
[biowulf]$ qsub -l nodes=1 makeconfig.sh
123456.biobos
[biowulf]$ export PATH=/data/user/funkyapp/bin:$PATH
[biowulf]$ cat swarmfile
cd /scratch; funkycmd -in /data/user/config -seed 1; funkyscript.sh 1
cd /scratch; funkycmd -in /data/user/config -seed 2; funkyscript.sh 2
cd /scratch; funkycmd -in /data/user/config -seed 3; funkyscript.sh 3
cd /scratch; funkycmd -in /data/user/config -seed 4; funkyscript.sh 4
...
cd /scratch; funkycmd -in /data/user/config -seed 256; funkyscript.sh 256
[biowulf]$ swarm -f swarmfile -l nodes=1:g24 -n 2 -b 4 --prologue /usr/
local/bin/clearscratch --epilogue 'funkymerge.sh -dir /data/user' -V -W
depend=afterany:123456.biobos --jobarray --singleout --usebash
123489[].biobos
[biowulf]$ qsub -l nodes=1:g72 -W depend=afterany:123489[].biobos
finalanalysis.sh
```

Don't start the swarm until the previous job 123456.biobos is finished.

# Supercomplex swarm Example

```
[biowulf]$ qsub -l nodes=1 makeconfig.sh
123456.biobos
[biowulf]$ export PATH=/data/user/funkyapp/bin:$PATH
[biowulf]$ cat swarmfile
cd /scratch; funkycmd -in /data/user/config -seed 1; funkyscript.sh 1
cd /scratch; funkycmd -in /data/user/config -seed 2; funkyscript.sh 2
cd /scratch; funkycmd -in /data/user/config -seed 3; funkyscript.sh 3
cd /scratch; funkycmd -in /data/user/config -seed 4; funkyscript.sh 4
...
cd /scratch; funkycmd -in /data/user/config -seed 256; funkyscript.sh 256
[biowulf]$ swarm -f swarmfile -l nodes=1:g24 -n 2 -b 4 --prologue /usr/
local/bin/clearscratch --epilogue 'funkymerge.sh -dir /data/user' -V -W
depend=afterany:123456.biobos --jobarray --singleout --usebash
123489[].biobos
[biowulf]$ qsub -l nodes=1:g72 -W depend=afterany:123489[].biobos
finalanalysis.sh
```

Run the swarm as a jobarray, squish all the .o and .e files into single files, and use /bin/bash as the shell

# Supercomplex swarm Example

```
[biowulf]$ qsub -l nodes=1 makeconfig.sh
123456.biobos
[biowulf]$ export PATH=/data/user/funkyapp/bin:$PATH
[biowulf]$ cat swarmfile
cd /scratch; funkycmd -in /data/user/config -seed 1; funkyscript.sh 1
cd /scratch; funkycmd -in /data/user/config -seed 2; funkyscript.sh 2
cd /scratch; funkycmd -in /data/user/config -seed 3; funkyscript.sh 3
cd /scratch; funkycmd -in /data/user/config -seed 4; funkyscript.sh 4
...
cd /scratch; funkycmd -in /data/user/config -seed 256; funkyscript.sh 256
[biowulf]$ swarm -f swarmfile -l nodes=1:g24 -n 2 -b 4 --prologue /usr/
local/bin/clearscratch --epilogue 'funkymerge.sh -dir /data/user' -V -W
depend=afterany:123456.biobos --jobarray --singleout --usebash
123489[].biobos
[biowulf]$ qsub -l nodes=1:g72 -V -W depend=afterany:123489[].biobos
finalanalysis.sh
```

Submit another job which will wait until the swarm has finished, and which requires a 72GB node. Make sure to also import your $PATH variable using `-V`.

# Pitfalls With swarm

- Don't delete .swarm directory
- Commands in swarmfile are not executed in order, assume random
- swarm uses tcsh by default, not bash; use the option `--usebash`
- Don't overwhelm the scheduler, use `-b`
- Be aware of how many files you will create

# Questions?

- `[biowulf]$ man swarm`

- http://biowulf.nih.gov/apps/swarm.html
- http://biowulf.nih.gov/swarm_extra.html

# Benchmarking Parallel Code

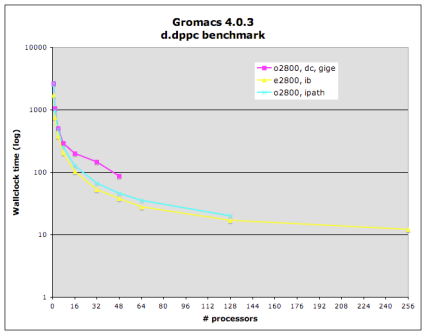
To determine the most efficient way to run a job.

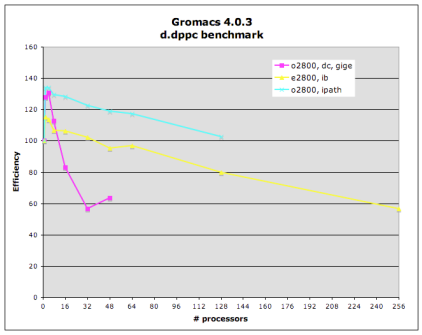
| # cores | Time taken | Efficiency |
|---|---|---|
| 1 | 128 m | 100% |
| 2 | 65 m | 98% |
| … | | |
| 16 | 20 m | 40% |

$$\text{Efficiency} = \frac{100 * (\text{time on 1core})}{n * (\text{time on n cores})}$$

# Benchmarking Parallel Code

Jobs should be run on the # cores where you get 80% efficiency.



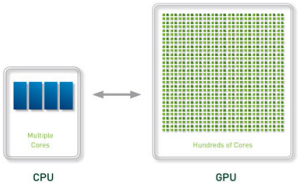

# GPU: Graphical Processing Unit

- Microprocessor used in video cards
- Development driven by video gaming
- "Hard-wired" for graphics operations

62

# GPUs in HPC

- Mainstream processors (Intel, AMD): multi-core rather than higher clock speeds
- New generations of GPUs: use as a general-purpose matrix or array processor



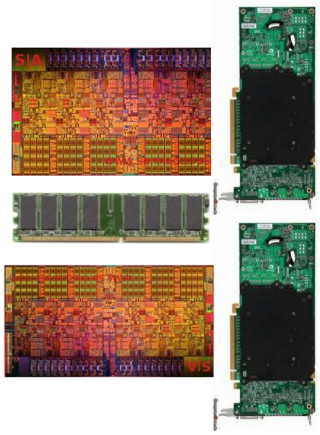

- Problems: DP vs SP floating-point performance, lack of ECC memory, programming model
- 2010: NVIDIA Tesla Fermi and CUDA 3.1

63

# Biowulf GPU Pilot Hardware/Software

16 compute nodes, each with…

- 2 x Intel Xeon X5650 (total: 12 cores)
- 2 x NVIDIA M2050 (total: 880 cores)
- 48 GB memory
- QDR Infiniband
- CUDA 3.1



64

# Targeted Applications

Molecular Dynamics

- NAMD
- Amber
- Gromacs

Bioinformatics

- CUDA Blastp
- GPU BLAST (pblast)
- CUDA Meme
- MUMmer GPU
- GPU HMMER

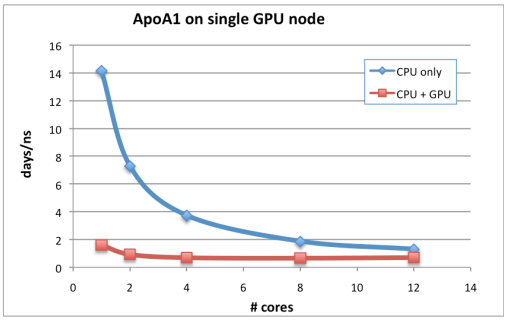
Allocating a GPU node:

qsub –l nodes=1:gpu2050 myjobscript

65

# NAMD Benchmarks, single-node

better




| #CPU cores | days/ns (%eff) CPU only | days/ns (%eff) CPU+GPU | CPU/ GPU ratio | Speedup with GPUs |
|---|---|---|---|---|
| 1 | 14.15 (100) | 1.60 (100) | 1 | 8.84 |
| 2 | 7.29 (97) | 0.94 (85) | 1 | 7.79 |
| 4 | 3.72 (95) | 0.70 (57) | 2 | 5.32 |
| 8 | 1.85 (95) | 0.67 (30) | 4 | 2.78 |
| 12 | 1.29 (92) | 0.71 (19) | 6 | 1.82 |

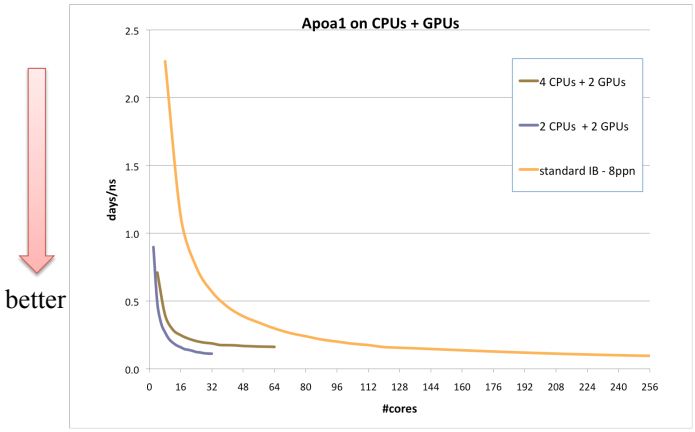
- ApoA1 92K atoms
- STMV 1M atoms

- 1:1 CPU:GPU optimal, adding additional CPUs increases performance but at very reduced efficiency
- STMV gives similar results

66

# NAMD Benchmarks, multi-node

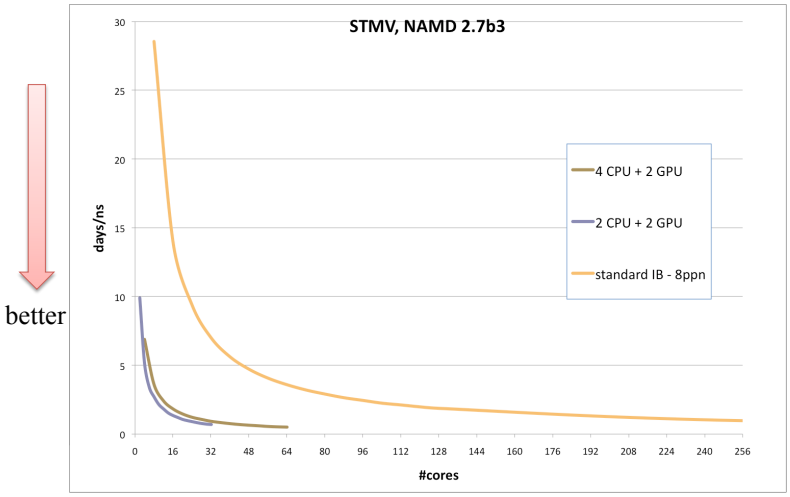
Apoa1: 92,224 atoms, 12A cutoff + PME every 4 steps, periodic. OutputEnergies=600.




67

# NAMD Benchmarks, multi-node

STMV: 1,066,628 atoms, periodic, PME. OutputEnergies=100.

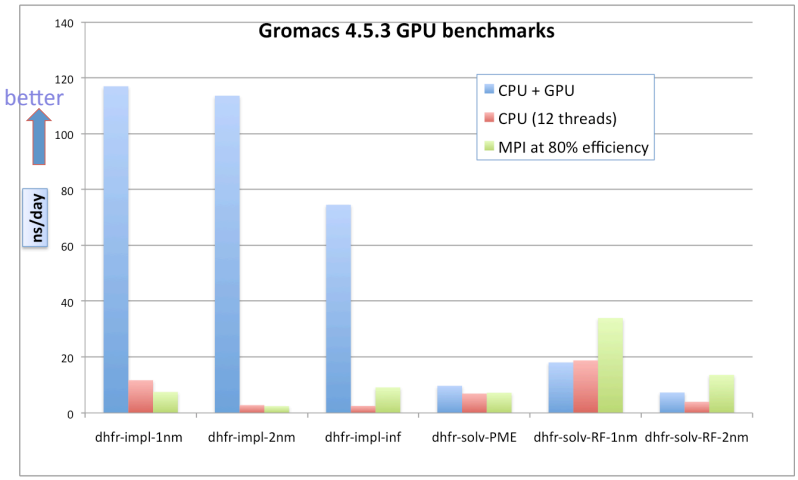



68

# NAMD Benchmarks, multi-node summary

| | ApoA1 (days/ns) | STMV (days/ns) |
|---|---|---|
| 16 GPU nodes, 64 cpus + 32 GPUs (2:1) | 0.16 | 0.52 |
| 16 GPU nodes, 32 cpus + 32 GPUs (1:1) | 0.11 | 0.71 |
| 32 STD nodes, 256 cpus | 0.10 | 0.94 |
| 64 STD nodes, 512 cpus | 0.06 | 0.49 |

69




Caveats:
Not parallel
Limited set of features implemented
Can't do multi-million atom simulations
File I/O is expensive.
Different parameters/settings.

70

# CUDA Meme Benchmarks

| # CPUs | CUDA meme (+ 1 GPU) | meme (standard build) |
|---|---|---|
| 1 | 1205 s | 9442 s |
| 8 | | 1239 s |
| 12 | | 930 s |

- *meme* is multi-threaded, GPU-implementation is single threaded
- GPU-implementation gives 7.8X "single"-threaded performance
- But…

71

# Conclusions

- Single-threaded performance on GPUs is great.
- But not compared to multi-node performance on standard nodes.
- Only NAMD can run multi-node GPU jobs so far.
- GPUs are a win for large molecular simulations with NAMD, implicit-solvent simulations with Gromacs.
- Need to drive down CPU:GPU ratio for future compute node configurations

72

# Next Steps

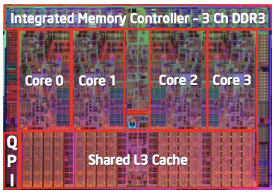
- Identifying user simulations that will benefit from using GPUs
- Support any user initiatives to port or develop GPU-enabled applications
- Monitor the field for GPU-enabled applications in other computational biology domains

*New #1 in the Supercomputing TOP500: Tianhe-1A (7168 NVIDIA GPUs, 14336 Intel processors) NUTD, China*

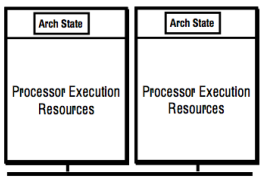
73

# Coming Soon: Hyperthreading

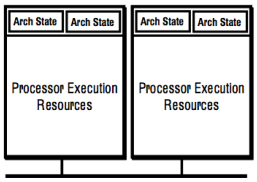
processor



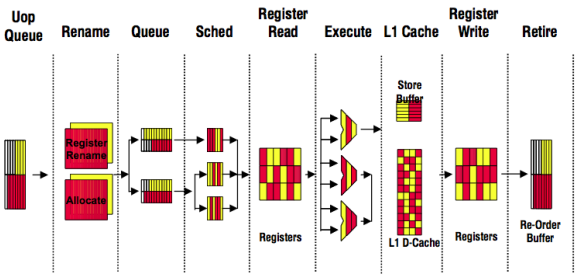

cores




no hyperthreading




hyperthreading enabled



# Hyperthreading

- Only on certain Intel processors
- Submit to h16 nodes (== c8)

```
% qsub –l nodes=1:h16 –v np=16 myjob
```

- Use *ht* to verify hyperthreading is enabled

```
% rsh p1000 ht
0
% rsh p76 ht
1
```

- "charged" for real cores, not virtual cores

# Hyperthreading Performance on c8 (h16) nodes

- **bowtie**

| threads | time (s) | efficiency |
|---|---|---|
| 1 | 10129 | 100 |
| 2 | 5009 | 101 |
| 4 | 2519 | 101 |
| 8 | 1290 | 98 |
| 12 | 1056 | 80 |
| 16 | 900 | 70 |

- **R**

| Processes | time | over 1p | work per unit time |
|---|---|---|---|
| 1 | 1734 | 1.00 | 1.0 |
| 2 | 1734 | 1.00 | 2.0 |
| 4 | 1748 | 1.01 | 4.0 |
| 8 | 1805 | 1.04 | 7.7 |
| 12 | 2773 | 1.60 | 7.5 |
| 16 | 2853 | 1.65 | 9.7 |

# Hyperthreading Performance on c8 (h16) nodes

- **NAMD**

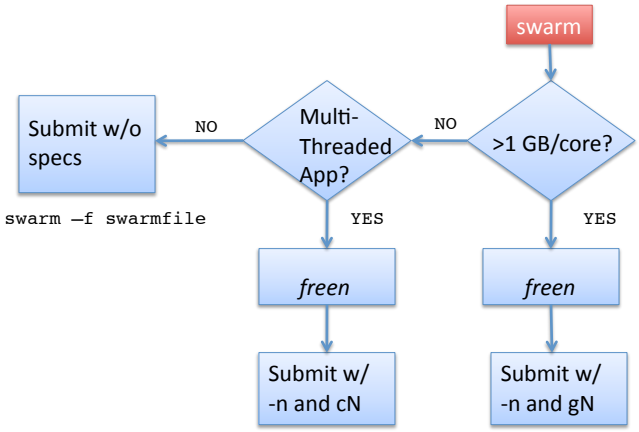
| Processes | d/ns | efficiency |
|---|---|---|
| 1 | 18.1 | 100 |
| 2 | 10.1 | 90 |
| 4 | 4.92 | 92 |
| 8 | 2.39 | 95 |
| 10 | 2.28 | 79 |
| 12 | 2.16 | 70 |
| 16 | 1.92 | 59 |

- **GROMACS**

| Processes | ns/d | efficiency |
|---|---|---|
| 1 | 0.325 | 100 |
| 2 | 0.648 | 100 |
| 4 | 1.255 | 97 |
| 8 | 2.79 | 107 |
| 12 | 2.345 | 60 |
| 16 | 3.056 | 59 |

16 (2:ib) 4.742

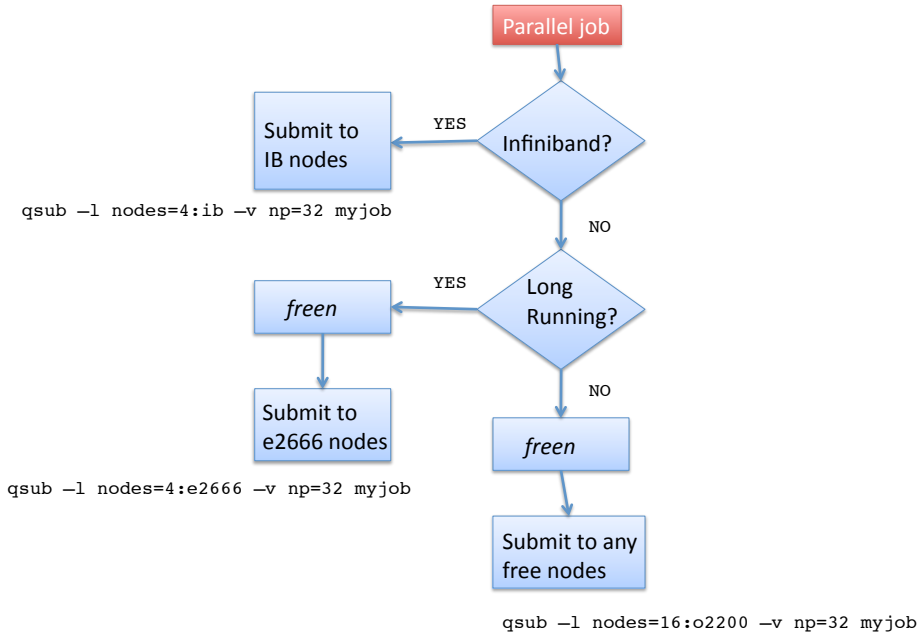
# Node Selection




(for 4 threads/process)
swarm –n 2 –l nodes=1:c8 –f swarmfile

(for a single 8 GB process)
swarm –n 1 –l nodes=1:g8 –f swarmfile

# Node Selection




# Node Selection: 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 |

# 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
```

# Reminders

- Use /data/username, not /data3/c/username
- Use *jobload* to check the behavior of your jobs
- Login node: use /scratch, not /tmp or /var/tmp
- No compute-intensive jobs on the login node
- Make sure /home/*username*/.forward points to your correct email address
- Questions and problems to *staff@biowulf.nih.gov* (not to individual staff)