HPC @ CCS

4/29/2013 J. Zysman
Center for Computational Science
Overture

What we will cover
Quick Definitions

4/29/2013 J. Zysman
Center for Computational Science
What we will cover

- HPC and Supercomputers
- Pegasus
- Pegasus Storage
- LSF
- Batch & Interactive Jobs
- Parallel & Serial Jobs
- LSF
  - bsub
  - bjobs
  - bhist
  - bkill
  - bqueues
  - bhosts
  - Bpeek
- Lab – Practice
Pegasus MK II Advanced Compute Environment

**Definitions**

- **Pegasus** – University of Miami’s supercomputer
  - Fastest in Florida
  - 389 in the world
- **Job**
  - The smallest piece of work that can be run on a computer system
  - Parallel Jobs
    - Jobs that talk to each other
  - Serial Jobs
    - Jobs that are independent of each other
- **Node**
  - A computer system
- **CPU**
  - Central Processing unit
- **Core**
  - Smallest processing unit
Act one

HPC

4/29/2013 J. Zysman
Center for Computational Science
What is HPC/HTC

The discipline of designing computing systems and software to work on problems that cannot be addressed by standard commodity computers.
What is a Supercomputer

- Any computing environment that turns a CPU bound program into an IO bound program

- Any computing system that exceeds the capability of standard commodity computers
Merck 9714 – #150 of Top500 Supercomputers in the world 1996

From Computer Desktop Encyclopedia
Reproduced with permission.
© 1996 Cray Research, Inc.

32 CPU
1 TB RAM
800 GB/Sec
58 Gflops
128 bit Vector based
No storage
$39 Million
Pegasus MK II Advanced Compute Environment

Pegasus - #398 of Top500 Supercomputers in the world 2012

- IBM iDataplex and Blade Chassis architecture
- 10,000+ x86 Cores
- 18 TB RAM
- 30 GB/sec local
- 40 Gb/sec distal
- 200 Tflops
- 64 bit scalar
- 500+ TB storage
- $2.6 Million
For clarity purpose not all ISLs from leaf to core are represented.
Many CPU/Many Users

- How do we get programs to work across many computers
  - Scheduling & resource management software
  - All computers have internal scheduler & rm
    - Program execution order
    - Disk access order
    - CPU instructions
    - How much disk
    - How much memory
  - Clusters have an external scheduler as well
    - Which programs run where
  - Clusters have external resource managers as well
    - Memory
    - CPU
    - Disk space
Quick Interlude

Pegasus Storage
LSF
Batch & Interactive Jobs
Parallel & Serial Jobs

4/29/2013 J. Zysman
Center for Computational Science
Pegasus MK II Advanced Compute Environment

Pegasus Storage

- PCs (Windows, Linux, and Macs are all PCs)
  - Hard disk
  - CD/DVD
  - USB

- Clusters use interconnect
  - No internal media
  - USB is for internal support only
## Pegasus Storage

[jzysman@u01 ~]# df -h

<table>
<thead>
<tr>
<th>Filesystem</th>
<th>Size</th>
<th>Used</th>
<th>Avail</th>
<th>Use%</th>
<th>Mounted on</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/cciss/c0d0p1</td>
<td>32G</td>
<td>4.1G</td>
<td>27G</td>
<td>14%</td>
<td>/</td>
</tr>
<tr>
<td>tmpfs</td>
<td>3.9G</td>
<td>0</td>
<td>3.9G</td>
<td>0%</td>
<td>/dev/shm</td>
</tr>
<tr>
<td>ba4:/share</td>
<td>1.2T</td>
<td>698G</td>
<td>469G</td>
<td>60%</td>
<td>/share</td>
</tr>
<tr>
<td>is1.pegasus:/ifs</td>
<td>38T</td>
<td>34T</td>
<td>4.0T</td>
<td>90%</td>
<td>/scratch</td>
</tr>
<tr>
<td>ba3:/home</td>
<td>35T</td>
<td>26T</td>
<td>8.8T</td>
<td>75%</td>
<td>/nethome</td>
</tr>
<tr>
<td>c1:/gpfs/cloud</td>
<td>25T</td>
<td>20T</td>
<td>4.6T</td>
<td>82%</td>
<td>/cloud</td>
</tr>
<tr>
<td>ba4:/s2</td>
<td>44T</td>
<td>44T</td>
<td>1.4G</td>
<td>100%</td>
<td>/smoke</td>
</tr>
</tbody>
</table>
Pegasus Storage

**is1.pegasus:/ifs**
- 38TB Total
- 34TB Used
- 4.0TB Free
- 90% Used
- Attached over the network
- All data & results should go here
- Optimized for read write random
- Designed for transactional
- Purged

**ba3:/home**
- 35TB Total
- 26TB Used
- 8.8TB Free
- 75% Used
- Attached over the network
- Home directory
- All personal programs
- Optimized for read
- Designed to stream
- Quota
Pegasus MK II Advanced Compute Environment

**LSF**

- **Load Sharing Facility**
- The external scheduling and resource management software used by Pegasus to distribute the jobs you submit to the compute nodes for work.
- Currently supports over 1500 users
- Supports 200,000 simultaneous job submissions at once
Pegasus MK II Advanced Compute Environment

**LSF**
Queues

• Queues are the software categories we define in the scheduler to organize work more efficiently.

• We organize these queues using limits like
  • Job size
  • Job length
  • Job purpose
  • Projects
Interactive jobs

Jobs that interact with the user
• Most work on PC is interactive
• Mouse input
• Keyboard input
• User intervention
**Batch Jobs**

Self contained programs that require no intervention to run. Are defined by how many cores they need, how long they need to run, and priority of the user.

- Pegasus uses LSF for scheduling batch jobs according to a schedule and available resources on the computer.
- Users limited to a number of cores by policies.
- Bigger jobs have higher priority than smaller ones.
- Smaller jobs can run longer than bigger ones.
- Parallel and serial jobs co-exist.
- Parallel jobs are tougher to schedule since they are inherently larger.
- Serial jobs can “fit in” to the gaps left by larger jobs.
Programming Models

Serial Programming
One CPU one task. The vast majority of computational work (by number of jobs) is serial. By CPU hour it is not as clear which number is greater.

MPI – Message Passing Interface
A programming interface that allows programs running on different compute nodes to talk to each other.

OpenMP – Open Multi Processing
A programming interface that lets multiple CPU’s on the same compute node work on the same memory.
Act 2

Log in to Pegasus
Running jobs on Pegasus

4/29/2013 J. Zysman
Center for Computational Science
**LSF commands**

Please log in to Pegasus now.
# LSF commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bsub</code></td>
<td>Submits a job via script file</td>
</tr>
<tr>
<td><code>bsub -I</code></td>
<td>Submits a job via script file and logs the job log</td>
</tr>
<tr>
<td><code>bjobs</code></td>
<td>Displays running and pending jobs in the queue</td>
</tr>
<tr>
<td><code>bhist</code></td>
<td>Displays historical information about your finished jobs</td>
</tr>
<tr>
<td><code>bkill</code></td>
<td>Removes/kills a job from the queues</td>
</tr>
<tr>
<td><code>bqueues</code></td>
<td>Shows the queues on Pegasus</td>
</tr>
<tr>
<td><code>bhosts</code></td>
<td>Shows information about the nodes in the cluster</td>
</tr>
<tr>
<td><code>bpeek</code></td>
<td>Displays stderr and stdout from your jobs</td>
</tr>
</tbody>
</table>
**bsub**

Description

Submits a job for batch execution and assigns it a unique numerical job ID.

Runs the job on a host that satisfies all requirements of the job, when all conditions on the job, host, queue, application profile, and cluster are satisfied. If LSF cannot run all jobs immediately, LSF scheduling policies determine the order of dispatch. Jobs are started and suspended according to the current system load.

Sets the user’s execution environment for the job, including the current working directory, file creation mask, and all environment variables, and sets LSF environment variables before starting the job.

When a job is run, the command line and stdout/stderr buffers are stored in the directory home_directory/.lsbatch on the execution host. If this directory is not accessible, /tmp/.lsbtmp user_ID is used as the job’s home directory. If the current working directory is under the home directory on the submission host, then the current working directory is also set to be the same relative directory under the home directory on the execution host.
bsub -h

[jzysman@u02 ~]$ bsub -h
   [-L login_shell] [-c cpu_limit[/host_spec]] [-F file_limit]
   [-P project_name] [-G user_group] [-g job_group]
   [-k chkptnt_dir [init=initial_chkptnt_period] [chkptnt_period]
  [method=method_name]]
   [-q queue_name ...] [-R res Req [-R res Req ...]]
   [-Q requeue_exit_values]
   [-m "host_name[@cluster_name][!] |
   [-n min_processors[,max_processors]] [-J job_name] [-Jd job_description]
   [-b begin_time] [-t term_time] [-u mail_user]
   [-U reservation_id]
**bjobs**

**Bjobs**
**Description**
By default, displays information about your own pending, running and suspended jobs.

*bjobs* displays output for condensed host groups and compute units. The -l and -X options display uncondensed output.
**bjobs**

Bjobs
Description
By default, displays information about your own pending, running and suspended jobs.

bjobs displays output for condensed host groups and compute units. The -l and -X options display uncondensed output.
### bjobs –u all | less

<table>
<thead>
<tr>
<th>JOBID</th>
<th>USER</th>
<th>STAT</th>
<th>QUEUE</th>
<th>FROM_HOST</th>
<th>EXEC_HOST</th>
<th>JOB_NAME</th>
<th>SUBMIT_TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>16247848</td>
<td>zhu</td>
<td>RUN</td>
<td>nrda</td>
<td>u01</td>
<td>n0185</td>
<td>* download</td>
<td>Apr 26 13:54</td>
</tr>
<tr>
<td>16247849</td>
<td>zhu</td>
<td>RUN</td>
<td>nrda</td>
<td>u01</td>
<td>n0187</td>
<td>* download</td>
<td>Apr 26 13:57</td>
</tr>
<tr>
<td>16250032</td>
<td>zhu</td>
<td>RUN</td>
<td>nrda</td>
<td>u01</td>
<td>n0183</td>
<td>* download</td>
<td>Apr 26 22:18</td>
</tr>
<tr>
<td>16250034</td>
<td>zhu</td>
<td>RUN</td>
<td>nrda</td>
<td>u01</td>
<td>n0184</td>
<td>* download</td>
<td>Apr 26 22:32</td>
</tr>
<tr>
<td>16250035</td>
<td>zhu</td>
<td>RUN</td>
<td>nrda</td>
<td>u01</td>
<td>n0186</td>
<td>* download</td>
<td>Apr 26 22:32</td>
</tr>
<tr>
<td>16250036</td>
<td>zhu</td>
<td>RUN</td>
<td>nrda</td>
<td>u01</td>
<td>n0188</td>
<td>* download</td>
<td>Apr 26 22:32</td>
</tr>
<tr>
<td>16250037</td>
<td>zhu</td>
<td>RUN</td>
<td>nrda</td>
<td>u01</td>
<td>n0189</td>
<td>* download</td>
<td>Apr 26 22:32</td>
</tr>
<tr>
<td>16250042</td>
<td>zhu</td>
<td>RUN</td>
<td>nrda</td>
<td>u01</td>
<td>n0190</td>
<td>* download</td>
<td>Apr 26 23:39</td>
</tr>
<tr>
<td>16250043</td>
<td>zhu</td>
<td>RUN</td>
<td>nrda</td>
<td>u01</td>
<td>n0191</td>
<td>* download</td>
<td>Apr 26 23:39</td>
</tr>
<tr>
<td>15963468</td>
<td>amehta</td>
<td>RUN</td>
<td>small</td>
<td>u01</td>
<td>n0182</td>
<td>bgzip</td>
<td>Apr 10 17:34</td>
</tr>
<tr>
<td>15963526</td>
<td>amehta</td>
<td>RUN</td>
<td>small</td>
<td>u01</td>
<td>n0182</td>
<td>bgzip</td>
<td>Apr 10 17:34</td>
</tr>
<tr>
<td>16178531</td>
<td>rduncan</td>
<td>RUN</td>
<td>small</td>
<td>u03</td>
<td>1*n0084</td>
<td>mpijob</td>
<td>Apr 23 09:13</td>
</tr>
</tbody>
</table>
bhist

Description
By default:

Displays information about your own pending, running and suspended jobs.
Groups information by job

CPU time is not normalized

Displays events occurring in the past week, but this can be changed by setting the environment variable LSB_BHIST_HOURS to an alternative number of hours
```
[jzysman@u02 ~]$ bhist -a
Summary of time in seconds spent in various states:

<table>
<thead>
<tr>
<th>JOBID</th>
<th>USER</th>
<th>JOB_NAME</th>
<th>PEND</th>
<th>PSUSP</th>
<th>RUN</th>
<th>USUSP</th>
<th>SSUSP</th>
<th>UNKWN</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>16251929</td>
<td>jzysman</td>
<td>*??</td>
<td>7</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>16251931</td>
<td>jzysman</td>
<td>*@@@@@@@</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>16251932</td>
<td>jzysman</td>
<td>*rialjob</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>16251933</td>
<td>jzysman</td>
<td>*rialjob</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>16251934</td>
<td>jzysman</td>
<td>*rialjob</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>16251935</td>
<td>jzysman</td>
<td>*rialjob</td>
<td>6</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>16251936</td>
<td>jzysman</td>
<td>*rialjob</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
</tbody>
</table>
```
**bkill**

By default, sends a set of signals to kill the specified jobs. On UNIX, SIGINT and SIGTERM are sent to give the job a chance to clean up before termination, then SIGKILL is sent to kill the job. The time interval between sending each signal is defined by the JOB_TERMINATE_INTERVAL parameter in lsb.params(5).

By default, kills the last job submitted by the user running the command. You must specify a job ID or -app, -g, -J, -m, -u, or -q. If you specify -app, -g, -J, -m, -u, or -q without a job ID, bkill kills the last job submitted by the user running the command. Specify job ID 0 (zero) to kill multiple jobs.

On Windows, job control messages replace the SIGINT and SIGTERM signals (but only customized applications can process them) and the TerminateProcess() system call is sent to kill the job.

Exit code 130 is returned when a dispatched job is killed with bkill.
bqueues

Description
Displays information about queues.

By default, returns the following information about all queues: queue name, queue priority, queue status, job slot statistics, and job state statistics.

CPU time is normalized.

CPU time output is not consistent with bacct
bacct displays the sum of CPU time consumed by all past jobs in event files, regardless of the execution host type and run time (unless you indicate a begin and end time.) For a specified job, bacct and bhist have the same result.

Because the value of CPU time for bqueues is used by mbatchd to calculate fairshare priority, it does not display the actual CPU time for the queue, but a CPU time normalized by CPU factor. This results in a different CPU time output in bacct and bqueues.
<table>
<thead>
<tr>
<th>QUEUE_NAME</th>
<th>PRIOR</th>
<th>STATUS</th>
<th>MAX</th>
<th>JL/U</th>
<th>JL/P</th>
<th>JL/H</th>
<th>NJOBS</th>
<th>PEND</th>
<th>RUN</th>
<th>SUSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug</td>
<td>200</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>hptest</td>
<td>200</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nrda</td>
<td>200</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>9</td>
<td>0</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>small</td>
<td>100</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>6375</td>
<td>5833</td>
<td>542</td>
<td>0</td>
</tr>
<tr>
<td>biostats</td>
<td>100</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>medium</td>
<td>90</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>356</td>
<td>0</td>
<td>356</td>
<td>0</td>
</tr>
<tr>
<td>large</td>
<td>80</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>512</td>
<td>256</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>xlarge</td>
<td>70</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>bigmem</td>
<td>70</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>longrun</td>
<td>60</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>407</td>
<td>0</td>
<td>407</td>
<td>0</td>
</tr>
<tr>
<td>amd</td>
<td>60</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>eris</td>
<td>60</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>hermes</td>
<td>60</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>atlas</td>
<td>60</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>kronos</td>
<td>60</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ib</td>
<td>60</td>
<td>Open:Active</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>128</td>
<td>0</td>
<td>128</td>
<td>0</td>
</tr>
</tbody>
</table>
bhosts

Description
By default, returns the following information about all hosts: host name, host status, job state statistics, and job slot limits.

bhosts displays output for condensed host groups and compute units. These host groups and compute units are defined by CONDENSE in the HostGroup and ComputeUnit section of lsb.hosts. Condensed host groups and compute units are displayed as a single entry with the name as defined by GROUP_NAME or NAME in lsb.hosts.

When LSF adds more resources to a running resizable job, bhosts displays the added resources. When LSF removes resources from a running resizable job, bhosts displays the updated resources.

The -l and -X options display uncondensed output.

The -s option displays information about the numeric resources (shared or host-based) and their associated hosts.

With MultiCluster, displays the information about hosts available to the local cluster. Use -e to view information about exported hosts.
<table>
<thead>
<tr>
<th>HOST_NAME</th>
<th>STATUS</th>
<th>JL/U</th>
<th>MAX</th>
<th>NJOBS</th>
<th>RUN</th>
<th>SSUSP</th>
<th>USUSP</th>
<th>RSV</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1</td>
<td>ok</td>
<td>-</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>n0001</td>
<td>ok</td>
<td>-</td>
<td>8</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>n0002</td>
<td>ok</td>
<td>-</td>
<td>8</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>n0003</td>
<td>ok</td>
<td>-</td>
<td>8</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>n0004</td>
<td>ok</td>
<td>-</td>
<td>8</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>n0005</td>
<td>ok</td>
<td>-</td>
<td>8</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>n0006</td>
<td>ok</td>
<td>-</td>
<td>8</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>n0007</td>
<td>closed</td>
<td>-</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>n0008</td>
<td>ok</td>
<td>-</td>
<td>8</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>n0009</td>
<td>ok</td>
<td>-</td>
<td>8</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>n0010</td>
<td>closed</td>
<td>-</td>
<td>8</td>
<td>8</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>n0011</td>
<td>ok</td>
<td>-</td>
<td>8</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>n0012</td>
<td>ok</td>
<td>-</td>
<td>8</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>n0013</td>
<td>ok</td>
<td>-</td>
<td>8</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>n0014</td>
<td>ok</td>
<td>-</td>
<td>8</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>n0015</td>
<td>closed</td>
<td>-</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>n0016</td>
<td>closed</td>
<td>-</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>n0017</td>
<td>closed</td>
<td>-</td>
<td>8</td>
<td>8</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>
bpeek

Description
Displays the standard output and standard error output that have been produced by one of your unfinished jobs, up to the time that this command is invoked.

By default, displays the output using the command cat.

This command is useful for monitoring the progress of a job and identifying errors. If errors are observed, valuable user time and system resources can be saved by terminating an erroneous job.
bpeek  -h

[jzysman@u02 ~]$ bpeek -h
Usage: bpeek [-h] [-V] [-f] [-m "host_name ..." | -q queue_name | -J job_name | jobld | "jobld[index]"]
Intermission

Lunch
Stretch break

4/29/2013 J. Zysman
Center for Computational Science
Act 3
Practical

Work with files and directories

4/29/2013 J. Zysman
Center for Computational Science
Basic Instructions

[jzysman@u04 ~]$ pwd
/nethome/jzysman

[jzysman@u04 ~]$ ls -l
total 324192
-rw-rw-r--  1 jzysman hpc       342 Apr  28 12:45 16251933.err
-rw-rw-r--  1 jzysman hpc       539 Apr  28 12:45 16251933.out
drwxrwxr-x 11 jzysman hpc     2048 Jan  14  2011 Database
-rw-rw-r--  1 jzysman hpc       216 Oct  10  2012 java.log.15501
-rw-rw-r--  1 jzysman hpc       216 Oct  10  2012 java.log.15712
-rw-rw-r--  1 jzysman hpc       216 Oct  10  2012 java.log.16064
-rw-rw-r--  1 jzysman hpc       216 Oct  10  2012 java.log.4205
-rw-r--r--  1 jzysman hpc       207 Apr  28 12:44 job_sample
drwxrwxr-x  4 jzysman hpc     2048 Apr  28 12:33 Linux_Class
drwxrwxr-x  4 jzysman hpc     2048 Oct  10  2012 Matlab
Basic Instructions

[jzysman@u04 ~]$ cd /scratch/Linux_Class/
[jzysman@u04 Linux_Class]$ pwd
/scratch/Linux_Class
[jzysman@u04 Linux_Class]$
[jzysman@u04 Linux_Class]$
[jzysman@u04 Linux_Class]$
[jzysman@u04 Linux_Class]$ ls
Examples
[jzysman@u04 ~]$ pwd
/nethome/jzysman
[jzysman@u04 ~]$ cp -a /scratch/Linux_Class/Examples/ .
[jzysman@u04 ~]$ cd Examples
[jzysman@u04 Examples]$ ls -a
.. cpi.c Makefile pi pif_b.f90 pif.f90 pif_i.f90 ppif.f90 ppif.in README test_mpi.job test_serial.job
Basic Instructions

[jzysman@[jzysman@u04 Examples]$ more README
pif_i.f90      Serial interactive fortran code
Pif_b.f90      Serial interactive fortran code
ppif.f90       Parallel fortran code (uses modules)
cpi.c          Parallel C code (uses modules)

TO make executables
make pif
make ppif
make cpi

To run executables on 4 processors:
mpiexec -l -n 4 cpi < ppif.in
mpiexec -l -n 4 pif < ppif.in
mpiexec -l -n 4 ppif < ppif.in
Basic Instructions

[jzysman@u02 Examples]$ more Makefile
FC = /share/apps/mpich2/1.2.1/gcc/4.1.2/bin/mpif90
CC = /share/apps/mpich2/1.2.1/gcc/4.1.2/bin/mpicc
FFLAGS = -g
CFLAGS = -g

# precedence rule
%.o: %.f90
  $(FC) -c $(FFLAGS) $<
%:%.f90
  $(FC) -o $@ $(FFLAGS) $<
%.o: %.c
  $(CC) -c $(CFLAGS) $<
%:%.c
  $(CC) -o $@ $(CFLAGS) $<

clean:
  rm *.o ppif pif cpi pif_i pif_b
Act 4
Practical

Loading modules
Making programs
Running programs
Submitting jobs

4/29/2013 J. Zysman
Center for Computational Science
Basic Instructions

[jzysman@u02 Examples]$ module load LSF mpich2/gcc
[jzysman@u02 Examples]$
[jzysman@u02 Examples]$
[jzysman@u02 Examples]$
[jzysman@u02 Examples]$ make pif_i
/share/apps/mpich2/1.2.1/gcc/4.1.2/bin/mpif90 -o pif_i -g pif_i.f90
[jzysman@u02 Examples]$
[jzysman@u02 Examples]$
[jzysman@u02 Examples]$
[jzysman@u02 Examples]$ ./pif_i
  Enter the number of intervals
9
pi is approximately: 3.1426214565576127  Error is: 0.0010288029678196
[jzysman@u02 Examples]$ ./pif_i
  Enter the number of intervals
9
pi is approximately: 3.1426214565576127  Error is: 0.0010288029678196
[jzysman@u02 Examples]$ ./pif_i
  Enter the number of intervals
999999999
pi is approximately: 3.1415926535898007  Error is: 0.0000000000000075
[jzysman@u02 Examples]$

Basic Instructions

[jzysman@u02 Examples]$ time ./pif_i
  Enter the number of intervals
9
pi is approximately: 3.1426214565576127  Error is: 0.0010288029678196

real  0m4.607s
user  0m0.002s
sys   0m0.003s

[jzysman@u02 Examples]$ time ./pif_i
  Enter the number of intervals
999999999
pi is approximately: 3.1415926535898007  Error is: 0.00000000000000075

real  0m30.707s
user  0m22.779s
sys   0m1.907s
Basic Instructions

[jzysman@u04 Examples]$ more test_serial.job
#!/bin/bash
#BSUB -J serialjob
#BSUB -o %J.out
#BSUB -e %J.err
#BSUB -W 1:00
#BSUB -q small
#BSUB -n 1
#BSUB -B
#BSUB -N
#
# Run serial executable on 1 cpu of one node
cd /scratch/Linux_Class/Examples/
./pif_b
Basic Instructions

Here is a detailed line-by-line breakdown of the keywords and their assigned values listed in this script:

#!/bin/bash:

Specifies that the Bash shell is to be used when executing the command portion of the script.

#BSUB -J serialjob
assigns a name to job. The name of the job will show in the bjobs output.

#BSUB -o %J.out
redirect std output to a specified file. In this example, %J is the JobID.

#BSUB -e %J.err
redirect std error to a specified file

#BSUB -W 1:00
set wallclock time limit of 1 hour
Basic Instructions

#BSUB -q small
specify queue to be used

#BSUB -n 1
specify number of processors. For serial job, it would be 1.

#BSUB -B
Send email at job start

#BSUB -N
Send email at job end
Independent Exercise

Edit the test_serial.job script

Add the line #BSUB –U rsv#13
Change the queue from nrda to small
Change the jobname to yournamejob
Change the program name to pif_b
Correct the path (hint: it should have Examples in it)

Make the pif_b.f90 program (hint: don’t include the .f90)

Submit the job using bsub

[jzysman@u04 Examples]$ bsub < test_serial.job
Independent Exercise

Edit the test_serial.job script

Add the line #BSUB –U rsv#13
Change the queue from nrda to small
Change the jobname to yournamejob
Change the program name to pif_b
Correct the path (hint: it should have Examples in it)

Make the pif_b.f90 program (hint: don’t include the .f90)

Submit the job using bsub

[jzysman@u04 Examples]$ bsub < test_serial.job
Independent Exercise part 2

Edit pif_i.f90 and add the following line just before the stop line at the end:

Call Sleep(60)

Use bjobs to see the status of the job
Use bpeek to check the job’s output
Use bkill to kill your job
Finale

Running MPI jobs

4/29/2013 J. Zysman
Center for Computational Science
Basic Instructions

[jzysman@u04 Examples]$ more test_serial.job
#!/bin/bash
#BSUB -J serialjob
#BSUB -o %J.out
#BSUB -e %J.err
#BSUB -W 1:00
#BSUB -q small
#BSUB -n 32
#
# Run an MPI job with the "mpirun.lsf" MPI job starter.
mpirun.lsf ./test.x mpi.in
Independent Exercise part 2

Correct the test_mpi.job script to correct
• Queue - small
• Executable name (pif_b),
• MPI wrapper name (ppif.in)
• Make pif_b
• Submit the script