User Environment on Super Mike II

Le Yan & Bhupender Thakur & Alex Pacheco

User Services
HPC @ LSU
Outline

● User environment
  - System access and file systems
  - Software stack
  - Compiling programs on Super Mike II – GPU-accelerated code and Open MPI
  - Job management

● Applications benchmark results
SuperMikell

• Ranked 250 in Nov 2012 Top 500 list
• 440 nodes, dual 8-core Intel Sandy Bridge Xeon processors @ 2.6GHz, including
  – 382 nodes with 32 GB RAM
  – 50 GPU nodes with 64 GB RAM and dual NVIDIA Tesla M2090 GPUs
  – 8 nodes with 256 GB RAM, capable of aggregation into a single virtual symmetric multiprocessing (vSMP) using ScaleMP
• 146 CPU TFLOPS and 66 GPU TFLOPS (double-precision)
• 364 TB of storage space
• Mellanox Infiniband network
Accessing the Clusters

• Host name: *mike.hpc.lsu.edu*

• Use ssh to connect
  – Unix-alike and Mac: “`ssh <host name>`” in a terminal window
  – Windows: use **Putty** or other ssh clients

• The default log in shell is bash
  – Supported: bash, tcsh, csh, ksh and sh
Connection with X11 Forwarding

• Some software packages have GUI, which requires X11 forwarding to be established with the ssh connection
• Unix/Linux users
  – Use the “-X” option of ssh
• Mac users
  – Use the “X11” application
• Windows users
  – Install X server for Windows (e.g. Xming)
  – Enable X11 forwarding in the ssh client
File Systems

<table>
<thead>
<tr>
<th>Distributed</th>
<th>Throughput</th>
<th>File lifetime</th>
<th>Best used for</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home</td>
<td>Yes</td>
<td>Low</td>
<td>Unlimited</td>
</tr>
<tr>
<td>Work/Scratch</td>
<td>Yes</td>
<td>High</td>
<td>30 or 60 days</td>
</tr>
<tr>
<td>Local scratch</td>
<td>No</td>
<td>Job duration</td>
<td>Temporary files</td>
</tr>
</tbody>
</table>

- **Tips**
  - Never let your job write output to your home directory
  - Do not write temporary files to /tmp
    - Write to the local scratch or work space
    - The work space is not for long-term storage
    - Files are purged periodically
  - Use “rmpurge” to delete large amount of files
Disk Quota

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Home Access point</th>
<th>Home Quota</th>
<th>Work Access Point</th>
<th>Work Quota</th>
<th>Local scratch</th>
</tr>
</thead>
<tbody>
<tr>
<td>SuperMikeII</td>
<td>/home/$USER</td>
<td>5 GB</td>
<td>/work/$USER</td>
<td>None</td>
<td>/var/scratch</td>
</tr>
</tbody>
</table>

- No quota is enforced on the /work space on Super Mike II (146TB total)
  - Again, it is subject to a purge every 30 to 60 days
- The work directory is created within an hour after the first login
- Command to check current disk usage: `showquota`
Storage Allocation on /project

• One can apply for extra disk space on the /project volume (218TB total on Super Mike II) if
  – your research requires some files to remain on the cluster for a fairly long period of time; **and**
  – their size exceeds the quota of the /home
• The unit is 100 GB
• Storage allocations are good for 6 months, but can be extended based on the merit of the request
• Examples of valid requests
  – I am doing a 6-month data mining project on a large data set
  – The package I am running requires 10 GB of disk space to install
• Examples of invalid requests
  – I do not have time to transfer the data from my scratch space to my local storage and I need a temporary staging area
File Transfer

• From/to a Windows machine
  – Use a client that supports the scp protocol (e.g. WinSCP, Filezilla)

• From/to a Unix/Linux/Mac machine (including between the clusters)
  – scp command
    • Syntax: scp <options> <source> <destination>
  – rsync command
    • Syntax: rsync <options> <source> <destination>
Software Stack

• Similar to other LSU HPC Linux clusters
  – Installed under /usr/local/packages

• Compiler
  – Default: Intel 13.0.0
  – Also available: Intel 12.1.4, PGI-12.8, PGI-11.5, GCC-4.7.2, GCC 4.4.6

• MPI
  – Default: OpenMPI 1.6.2
  – Also available: Mvapich2 1.8.1, Mvapich2 1.9a2
Software Stack Cont’d

• Script language
  – Python 2.7.3
  – Perl 5.10.1
  – Ruby 1.9.3

• Application software
  – NAMD, Matlab, Mathematica, DDT...

• A complete list available at http://www.hpc.lsu.edu/docs/guides/index.php#Supermike2
  – Or use the softenv command to find out

• User requested software will be installed in the user’s home space
  – Unless requested by multiple users, in which case it will be installed under /usr/local/packages
Using SOFTENV

- Environment variables
  - PATH: where to look for executables
  - LD_LIBRARY_PATH: where to look for shared libraries
  - Other custom environment variables needed by various software

- SOFTENV is a software that helps users set up environment variables properly to use software packages
  - Much more convenient than setting environment variables in .bashrc or .cshrc
Listing All Packages

- Command “softenv” lists all packages that are managed by SOFTENV

[lyan1@mikel ~]$ softenv
SoftEnv version 1.6.2

... These are the keywords explicitly available:

+ImageMagick-6.7.9-gcc-4.4.6 @types: library @name: ImageMagick @version: 6.7.9 @build: ImageMagick-6.7.9-gcc-4.4.6 @internal: @external: http://www.imagemagick.org @about: A software suite to create, edit, and compose bitmap images.

+Intel-12.1.4 @types: Compiler @name: Intel compiler suite @version: 12.1.4 @build: Binary installation @internal: @external: http://software.intel.com/en-
Searching A Specific Package

- Use "-k" option with "softenv"

```
[lyan1@mike1 ~]$ softenv -k fftw
...
```

These are the keywords explicitly available:

- `+fftw-3.3.2-Intel-13.0.0` @types: library @name: fftw @version: 3.3.2 @build: Intel-13.0.0 @internal: @external: www.fftw.org @about: A fast, free C FFT library; includes real-complex, multidimensional, and parallel transforms.

- `+fftw-3.3.3-Intel-13.0.0` @types: library @name: fftw @version: 3.3.3 @build: Intel-13.0.0 @internal: @external: www.fftw.org @about: A fast, free C FFT library; includes real-complex, multidimensional, and parallel transforms.
Setting up Environment via Softenv – permanent change

- Set up the environment variables to use a certain software package
  - First add the key to $HOME/.soft
  - Then execute resoft at the command line
  - The environment will be the same next time you log in

```
[lyan1@mike1 ~]$ cat .soft
#
# This is the .soft file.
+portland-12.8
@default

[lyan1@mike1 ~]$ resoft

[lyan1@mike1 ~]$ which pgf90
/usr/local/compilers/pgi/linux86-64/12.8/bin/pgf90
```
Setting up Environment via Softenv – one time change

- Set up the environment variables to use a certain software package in the current login session only
  - Add a package: soft add <key>
  - Remove a package: soft delete <key>

```
[lyan1@mike1 ~]$ which gcc
/usr/bin/gcc
[lyan1@mike1 ~]$ gcc --version
gcc (GCC) 4.4.6 20110731 (Red Hat 4.4.6-3)
[lyan1@mike1 ~]$ soft add +gcc-4.7.2
[lyan1@mike1 ~]$ which gcc
/usr/local/compilers/GNU/gcc-4.7.2/bin/gcc
[lyan1@mike1 ~]$ gcc --version
gcc (GCC) 4.7.2
```
Querying a Softenv key

- Command “soft-dbq” shows which variables are set by a SOFTENV key

```bash
$ soft-dbq +portland-12.8
```

Name: +portland-12.8
Description: @types: Compiler @name: The Portland Group Compilers @version: 12.8 @about: Portland Group's high-performance compilers and tools.
-------------------------------------------

On the Linux architecture,
the following will be done to the environment:

The following environment changes will be made:
- `CUDA_NIC_INTEROP = 1`
- `LD_INCLUDE_PATH = ${LD_INCLUDE_PATH}:/usr/local/compilers/ pgi/linux86-64/12.8/include`
- `LD_LIBRARY_PATH = ${LD_LIBRARY_PATH}:/usr/local/compilers/pgi/linux86-64/12.8/lib`
- `MANPATH = ${MANPATH}:/usr/local/compilers/pgi/linux86-64/12.8/man`
- `PATH = ${PATH}:/usr/local/compilers/pgi/linux86-64/12.8/bin`
- `PGI = /usr/local/compilers/pgi`
Compilers

<table>
<thead>
<tr>
<th>Language</th>
<th>Vendor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel</td>
<td>PGI</td>
</tr>
<tr>
<td>Fortran</td>
<td>ifort</td>
</tr>
<tr>
<td>C</td>
<td>icc</td>
</tr>
<tr>
<td>C++</td>
<td>icpc</td>
</tr>
<tr>
<td>CUDA</td>
<td></td>
</tr>
</tbody>
</table>

- Usage: `<compiler> <options> <your_code>`
  - Example: `icc -O3 -o myexec mycode.c`
Compiling CUDA Programs

• Add the softenv keys for CUDA and NVIDIA driver
  – Currently `+cuda-4.2.9` and `+nvidiadriver-default`
• Use the `nvcc` command to compile

```bash
[lyan1@mike401 asyncAPI]$ nvcc -I../../common/inc/ asyncAPI.cu
[lyan1@mike401 asyncAPI]$ ./a.out
[./a.out] - Starting...
GPU Device 0: "Tesla M2090" with compute capability 2.0

CUDA device [Tesla M2090]
time spent executing by the GPU: 22.31
time spent by CPU in CUDA calls: 0.04
CPU executed 132214 iterations while waiting for GPU to finish
```
Compiling CUDA Fortran Programs

- CUDA Fortran
  - Fortran interface to CUDA
  - PGI v12.8 is the only compiler that supports it on Super Mike II

```bash
[lyan1@mike405 cudaFortran]$ which pgf90
/usr/local/compilers/pgi/linux86-64/12.8/bin/pgf90
[lyan1@mike405 cudaFortran]$ pgf90 matmul.CUF
[lyan1@mike405 cudaFortran]$ ./a.out
arrays sized           512  by          1024  by           512
calling mmul
Kernel time excluding data xfer: 2665.000 microseconds
Megaflops excluding data xfer: 100726.3
Total time including data xfer: 219162.0 microseconds
Megaflops including data xfer: 1224.827
C(1,1) = 3.5791874E+11
C(2,2) = 3.5739933E+11
No errors found
```
Compiling OpenACC Programs

• OpenACC
  – Directive-based programming for code running on accelerators with syntax similar to OpenMP
  – Available in Fortran, C and C++
• Supported via PGI 12.8

[lyan1@mike401 openacc]$ pgf90 vecadd.f90 -ta=nvidia -Minfo
vector_add:
  11, Generating copyout(c(1:10))
  Generating copyin(a(1:10))
  Generating copyin(b(1:10))
  Generating compute capability 1.0 binary
  Generating compute capability 2.0 binary
  12, Loop is parallelizable
  Accelerator kernel generated
  12, !$acc loop gang, vector(32) ! blockidx%x threadIdx%x
  CC 1.0 : 7 registers; 52 shared, 12 constant, 0 local memory bytes
  CC 2.0 : 14 registers; 0 shared, 68 constant, 0 local memory bytes
SHOC Benchmark

• **Scalable Heterogeneous Computing Benchmark Suite**
  – Multiple benchmarks in both CUDA and OpenCL
  – Supports multiple devices per node
  – Inter-node parallel benchmarks with MPI
  – Both performance tests and stability tests
SHOC – Data Movement over PCIe

SHOC 1.1.4, Serial (single node) benchmarks, Size 4

Very slow!
SHOC – Coalesced vs Strided Memory Access

SHOC Benchmark - Global Memory Access

<table>
<thead>
<tr>
<th>Block Size</th>
<th>CoalescedRead</th>
<th>StridedRead</th>
<th>CoalescedWrite</th>
<th>StridedWrite</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>110</td>
<td>90</td>
<td>80</td>
<td>70</td>
</tr>
<tr>
<td>64</td>
<td>120</td>
<td>100</td>
<td>90</td>
<td>80</td>
</tr>
<tr>
<td>128</td>
<td>130</td>
<td>110</td>
<td>100</td>
<td>90</td>
</tr>
<tr>
<td>256</td>
<td>140</td>
<td>120</td>
<td>110</td>
<td>100</td>
</tr>
<tr>
<td>512</td>
<td>150</td>
<td>130</td>
<td>120</td>
<td>110</td>
</tr>
</tbody>
</table>
SHOC - Tesla M2090 vs M2070

SHOC Benchmark - Single Node

Benchmark

SHOC 1.1.4, Serial (single node) benchmarks, Size 4
SHOC – Single vs Multiple Devices

SHOC 1.1.4, Parallel (MPI) benchmarks, Size 4
Debugging GPU Programs

- NVIDIA cuda-gdb
  - Command line debugger that comes with CUDA toolkit
  - CUDA only
- DDT
  - Parallel debugger that supports CUDA
    - Step through CUDA kernel
    - Examine data in GPU memory
**MPI Compilers (1)**

<table>
<thead>
<tr>
<th>Language</th>
<th>Compiler command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fortran</td>
<td>mpif77, mpif90</td>
</tr>
<tr>
<td>C</td>
<td>mpicc</td>
</tr>
<tr>
<td>C++</td>
<td>mpiCC</td>
</tr>
</tbody>
</table>

- **Usage:** similar to what we have seen
  - Example: `mpif90 -O2 -o myexec mycode.f90`
- **On Super Mike II**
  - The default MPI is OpenMPI 1.6.2
  - **Keys:**
    + openmpi-1.6.2-Intel-13.0.0
    + openmpi-1.6.2-gcc-4.4.6
    + openmpi-1.6.2-gcc-4.7.2
    + openmpi-1.6.2-pgi-12.8
The MPI compilers are **wrappers**
- They still use the compilers we've seen on the previous slide
  - Intel, PGI or GNU
- They take care of everything we need to build MPI codes
  - Head files, libraries etc.
- What they actually do can be revealed by the `--show` option

It's extremely important that you compile and run your code with the same version of MPI
- Use the default version (+openmpi-1.6.2-Intel-13.0.0) if possible
OpenMPI - mpirun

• Use `mpirun` or `mpiexec` (identical for OpenMPI) to run
• Options
  – `-np`: number of processes
  – `-hostfile` (or `-machinefile`): name of the host file
  – `--mca <parameter> <value>`: specify run-time environment
    • Back-end network, resource manager support etc.
    • Ex: “`--mca btl ^tcp`” prevents tcp from being used for point-to-point communication
OpenMPI – mpirun (cont’d)

• Options (cont’d)
  – --bind-to-core: bind processes to specific cores (default is not to)
  • Performance consideration mainly for MPI-OpenMP hybrid programs
  • --byslot: assign processes round-robin by slot (default)
  • --bysocket: assign processes round-robin by socket
  • --bynode: assign processes round-robin by node
OpenMPI Commands

• Useful commands other than mpirun
  – ompi_info: tells everything about the OpenMPI installation
    • Ex: ompi_info -param btl all
  – ompi-top: Diagnostic to provide process info similar to the popular "top" program
  – ompi-ps: Displays information about the active jobs and processes in Open MPI
OpenMPI Configuration

• Determined by (in the given order)
  – mpirun command line
  – Environment variables
  – File
    • $HOME/.openmpi/mca-params.conf
    • <OpenMPI installation path>/etc/openmpi-mca-params.conf
  – Default

• Defined by openmpi-mca-params.conf on Super Mike II
  – FCA is turned off by default
    • Because of firmware issues, but
    • Consider to turn it on to reduce latency when you have many collective MPI calls with large message size (--mca coll coll_fca_enable 1)
  – MPI_LEAVE_PINNED is turned off by default
    • May hang multi-threaded MPI programs, but
    • Consider to turn it on to sustain high bandwidth when you have lots of large MPI messages (--mca mpi_leave_pinned 1)
OSU MPI Benchmarks – Bandwidth

![Graph showing OSU Bandwidth Test](image)

- Bandwidth(Bi)
- Put(Bi)
- Bandwidth(Uni)
- Put(Uni)
- Get(Uni)
OSU MPI Benchmark – Latency
Batch Queuing System

- A software that manages resources (CPU time, memory etc.) and schedules job execution
  - Super Mike II: Torque (PBS)/Moab
- The batch queuing system determines
  - The order jobs are executed
  - On which node(s) jobs are executed
Job Queues

- There are more than one job queue
- The main purpose is group similar jobs to maximize utilization
- Each job queue differs in
  - Number of available nodes
  - Max run time
  - Max running jobs per user
  - Node characteristics
  - ...
## Job Queues on Super Mike II

<table>
<thead>
<tr>
<th>Machine</th>
<th>Queue</th>
<th>Max Runtime</th>
<th># of nodes</th>
<th>Max running jobs per user</th>
<th>Max nodes per job</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>SuperMikell</td>
<td>workq</td>
<td>3 days</td>
<td>346</td>
<td>48</td>
<td>128</td>
<td>Unpreemptable</td>
</tr>
<tr>
<td></td>
<td>ckpt</td>
<td>262</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>bigmem</td>
<td>2 days</td>
<td>8</td>
<td></td>
<td>2</td>
<td>Preemptable</td>
</tr>
<tr>
<td></td>
<td>gpu</td>
<td>1 day</td>
<td>50</td>
<td></td>
<td>2</td>
<td>Big memory</td>
</tr>
<tr>
<td></td>
<td>single</td>
<td>3 days</td>
<td>2</td>
<td></td>
<td>1</td>
<td>Job using GPU</td>
</tr>
<tr>
<td>Mwfa</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Only open to certain users</td>
</tr>
<tr>
<td>Lasigma</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Priority</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Admin</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Job Types

- **Interactive job**
  - Set up an interactive environment on compute nodes for users
    - Advantage: can run programs interactively
    - Disadvantage: must be present when the job starts
  - Purpose: testing and debugging
    - Do not run on the head node!!!
    - Try not to run interactive jobs with large core count, which is a waste of resources

- **Batch job**
  - Executed without user intervention using a job script
    - Advantage: the system takes care of everything
    - Disadvantage: can only execute one sequence of commands which cannot changed after submission
  - Purpose: production run
Basic Commands

- Queue querying
  - Check how busy the cluster is

- Job submission
  - Submit a job to run

- Job monitoring
  - Check job status (estimated start time, remaining run time etc.)

- Job manipulation
  - Cancel/hold jobs
Queue Querying

- **Command:** qfree
  - Show the number of free, busy and queued nodes

```
[lyan1@mike1 ~]$ qfree
PBS total nodes: 453,  free: 118,  busy: 315,  down: 20,  use: 69%
PBS workq nodes: 168,  free: 30,  busy: 31,  queued: 0
PBS checkpt nodes: 252,  free: 30,  busy: 17,  queued: 248
```
Submitting Jobs

- **Interactive job**
  - `qsub -I -V -l walltime=<hh:mm:ss>,nodes=<# of nodes>:ppn=16 -A <your allocation> -q <queue name>`
  - Add `-X` to enable X11 forwarding
  - Limited to a maximum of 12 hours on Super Mike II

- **Batch job**
  - `qsub <job script>`

- **On Super Mike II `ppn` must be 16 except for serial jobs**
PBS Job Script – Parallel Jobs

```bash
#!/bin/bash
#PBS -l nodes=4:ppn=16 Number of nodes and processor
#PBS -l walltime=24:00:00 Maximum wall time
#PBS -N myjob Job name
#PBS -o <file name> File name for standard output
#PBS -e <file name> File name for standard error
#PBS -q checkpt Queue name
#PBS -A <hpc_allocation> Allocation name
#PBS -m e Send mail when job ends
#PBS -M <email address> Send mail to this address

<shell commands>
mpirun -machinefile $PBS_NODEFILE -np 16 <path_to_executable> <options>
<shell commands>
```
Running GPU-enabled Applications

• One needs to specify
  – How many devices per node
  – How many CPU cores per device

• Varies from one application to another
  – Ex: NAMD will automatically use all devices on the node(s) if built with CUDA

  • The user needs to determine how many CPU cores to use by
    – Creating an appropriate host file (with mpirun) or
    – Using the “+p<n>” and “+ppn<n>” option (with charmrun)

• The user can indicate which device(s) to use with the “+devices” option (with charmrun)
Example: Running NAMD on GPU

• Running NAMD on 16 nodes with 2 devices per node and 1 CPU process for device:
  – With mpirun

    mpirun -np 32 -hostfile $newhostfile <path to namd executable> apoa1.namd

  – With charmrun

    charmrun +p32 ++nodelist <path to host file> <path to namd executable> apoa1.namd ++ppn 2 +idlepoll ++remote-shell ssh
NAMD Benchmark – CPU Only

NAMD 2.9, Apoa1 benchmark

http://www.ks.uiuc.edu/Research/namd/performance.html
NAMD Benchmark – With GPU

NAMD 2.9, Apoa1 benchmark
http://www.ks.uiuc.edu/Research/namd/performance.html
Job Monitoring

- **Command:** `showstart <job_id>`
  - Check when a job is estimated to start

- Things that can change the estimated start time
  - Higher priority jobs are submitted
  - Running jobs terminate earlier than the system expects
  - The system has trouble starting your job
Job Monitoring cont'd

● **Command:** `qstat <options> <job_id>`
  - Show information on job status
  - All jobs are displayed if `<job_id>` is omitted
  - Show jobs submitted by a specific user: `qstat -u <username>`
  - Display in the alternative format: `qstat -a <job_id>`

● **Command:** `qshow <job_id>`
  - Show information on a running job
    ● On which node(s) the job is running
    ● CPU load
    ● Memory usage
Job Manipulation

- **Command:** `qdel <job_id>`
  - Cancel a running or queued job
  - May take some time depending on the size of the job

- **Command:** `qhold <job_id>`
  - Put a queued job on hold

- **Command:** `qrls <job_id>`
  - Resume a held job
Benchmark Results on Super Mike II
NAS Parallel Benchmarks

- Developed by NASA Advanced Supercomputing (NAS)
- Derived from CFD applications
  - MG: Multi-grid method for a 3D Poisson problem
  - CG: Conjugate-gradient solver for an unstructured sparse linear system
  - BT: Block tri-diagonal solver
  - SP: Scalar penta-diagonal solver
  - LU: Lower-upper Gauss-Seidel solver
NAS Parallel Benchmarks

Compiler flag: -O3 -xHost

NPB-3.2, problem size D, pure MPI, 256 processes
LAMMPS Benchmark – CPU Only
GROMACS Benchmark

GROMACS 4.5.4, d.dppc benchmark
Questions?