KFUPM HPC Workshop

April 29-30 2015

Mohamed Mekias

HPC Solutions Consultant

Agenda

Agenda-Day 1

- HPC Overview
  - What is a cluster?
    - Shared v.s. Distributed
    - Parallel v.s. Massively Parallel
  - Interconnects (Infiniband, tcp)
- HPC System @ KFUPM
  - Software stack
- Parallel programming & Tools of the trade
  - Parallelism with OpenMP and libraries
    - Open source (gcc,gfortran)
    - Commercial tools (Intel Parallel Studio (Optimizing compilers, IMPI, MKL, IPP, profiling tools, Intel and PGI),
  - Parallelism with MPI
    - Open source openmpi, mvapich
    - Commercial tools (Intel MPI library, Intel Parallel Debugger)
  - **Massive parallelism with GPUs**
    - **NVIDIA CUDA, PGI and OpenACC and NSIGHT and thrust library**
    - **This might be pushed to Day 2**

Agenda-Day 2

- Job submissions (CLI)
  - Job Scheduler
  - What is LSF?
  - Basic LSF scripts for submitting jobs
    - Submitting to a single machine (SMP, OpenMP jobs)
    - Submitting to multiple machines (Distributed)
    - Lab scripts using familiar applications like MATLAB
    - Write simple MPI applications and submit jobs to multiple machines
  - Some useful LSF commands
  - Managing LSF jobs (killing, suspending/stopping and resuming), job history, job resource consumption
  - Submitting a GPU job
- Job Submission via the Web portal
  - Submit jobs based on published applications
  - Submit generic jobs
  - Submit jobs to GPUs

HPC jargon

- Basic Concepts of Parallelism
  - SISD (Instruction Level Parallelism, threads, compilers role)
  - SIMD (Vector processors, GPUs, SSE/AVX instructions)
  - MISD (rare breed, Multiple Instruction Single Data)
  - MIMD (Multiple Instruction Multiple Data, clusters)
- Communication and different interconnects
  - TCP (used for embarrassingly parallel programs, like Monte Carlo)
  - Infiniband (and others) used for inter-process communication across machines
  - Topologies

HPC System @ KFUPM is MIMD

- Brief description of HW
  - Management node
  - CPU compute nodes
  - GPU nodes
  - Interconnect (Intel IB network)
- File system structure
- Software stack
  - Compilers and development tools (We cover Intel compilers, PGI and gcc/gfortran)
  - Libraries (MKL, IPP, ...)
  - Compiling programs and linking them to libraries
- Development environment
- Batch scheduling (time sharing) with LSF

Hardware (details)

- 1x Head Node (hpcmaster)
  - IP address to access the node 10.146.2.1
  - This is your entry to the cluster if you want to submit jobs on the command line
  - 2x 10 cores Intel Xeon CPU E-2680 v2 2.8 GHz and 25MB shared cache (total 20 cores)
  - 128GB DDR3 ECC RAM
- 32x compute nodes with
  - 2x 10 cores Intel Xeon CPU E-2680 v2 2.8 GHz and 25MB shared cache (total 20 cores)
  - 64GB DDR3 ECC RAM
  - Names are
    - Node[000-019]
    - gpu[00-11]

Hardware

- 12 Compute nodes gpu[00-11] are equiped with NVIDIA GPUs
  - TESLA K20Xm [GK110GL]
- Fast QDR Infiniband Network
  - QLOGIC/INTEL
  - Used for MPI
  - Infinipath (PSM) protocol is preferred for best performance

Software
Most of the software packages are installed on /shared

/shared
|-- ansys_inc
|-- cuda-samples
|-- etc
|-- g09
|-- gcc-4.9.2
|-- GEOS
|-- gmp-6.0.0
|-- gromacs
|-- hdf5
|-- ibm
|-- intel
|-- IntelIB-Basic.RHEL6-x86_64.7.3.0.0.26
|-- lammps-9Dec14
|-- MaterialStudio
|-- matlab
|-- MatlabR2014a
|-- modulefiles
|-- mpc-1.0.3
|-- mpfr-3.1.2
|-- netcdf
|-- octave
|-- OpenFoam
|-- pgi_install
|-- R
|-- scripts
|-- SEM2DPACK

Exceptions (due to limited licenses):

- CUDA Toolkit is installed in the default /usr directory on the GPU nodes
- PGI compiler is installed on gpu00 and gpu01 installed on /opt
  - Compile in these nodes
  - Can execute PGI generated binaries on all the gpu nodes if using CUDA or on all nodes if CUDA is not used
- Intel Cluster Toolkit
  - Only installed on master node
  - Compile on master and execute on all nodes
    - Runtime libraries are royalty free

Software-- Modulefiles

- All relevant user software is loaded via module files
  - Loaded into user environment using module files
- To see what is available
  - Type the command 'module avail'

```
[mmekias@hpcmaster ~]$ module avail

--------------------------------------------------------------------------------- /usr/share/Modules/modulefiles ----------------------------
-----------------------------------------------
PMPI/modulefile dot             module-cvs      module-info     modules         null            use.own

--------------------------------------------------------------------------------- /shared/modulefiles ---------------------------------------
-----------------------------------------------
Intel_Compilers         cuda6.5                 mpich-x86_64            mvapich2-1.8.1-qlc_gcc  openmpi-1.8.1-qlc_intel
pgi/14.10               pgi/shared
Intel_MPI               gcc-4.9.2               mvapich-1.2.0-qlc_gcc   octave                  openmpi-psm-x86_64      pgi/2014
LAMMPS                  matlab                  mvapich-1.2.0-qlc_intel openmpi-1.8.1-qlc_gcc   openmpi-x86_64
pgi/mpich
```

Loading/Unloading a module

- To load a module issue the command “module load <<name>>”
  - For example: “module load Intel_MPI”, will load the environment variables needed to run an MPI job with Intel MPI
- To unload a module issue the comamnd “module unload <<name>>”
  - For example : “module unload Intel_MPI”
- To view the loaded modules, issue the command “module list”
  - For example : “module list”
  - >> Currently Loaded Modulefiles:
    - >> 1) Intel_MPI

Choices for Parallel Programming

- Shared memory machine
  - Compiler
  - pthreads
  - OpenMP
  - cuda (if machine has cuda device)
  - Even MPI
  - Exploit parallel libraries (transparent to users)
    - MKL
    - IPP
    - TBB
    - CILK

Introduction to Parallel Programming—Message Passing

- Essential MPI programming
  - What is MPI?
  - Different flavors of MPI libraries
  - Base MPI library
  - Basic Communication
  - Blocking vs Non-Blocking
  - Collective Operations

Introduction to Parallel Programming--GPUs

- Overview of GPU architecture
- What is CUDA?
- Inside the CUDA Toolkit
- Using PGI and OpenACC directives
- Using Thrust library

Batch Scheduling

- Introduction to LSF
- Basic commands
- Job submission of basic commands
- Submitting parallel jobs
- LSF scripting
- Job management
  - Load on system
  - Job status
  - Listing currently running jobs
  - Jobs history
  - Killing jobs
  - Suspend/stop jobs
  - Resuming jobs
- Jobs with specific resource requirements
  - Number of cores required
  - Memory required
  - GPU requirement

Web Portal

- Introduction to the web portal
- Submit jobs via web portal
- Viewing/managing jobs via portal