Ateneo de Manila University

Using the AGILA HPCS

Ateneo High Performance Computing Group
1st Semester 2001-2002
http://www.math.admu.edu.ph/ahpc/
william.s.yu@ieee.org
Section I

Introduction
Ateneo High Performance Computing Group

★ is a professional interdisciplinary research group that provides the Ateneo de Manila University community with high performance computing services

★ in August 2000, developed a Beowulf-class cluster computer called the AGILA HPCS

★ to support the parallel computing courses offered by University, particularly those computational subjects offered in the applied mathematics/computational science program of the Mathematics Department
AGILA HPCS

★ Ateneo Gigaflops-Range, Linux OS, and Athlon Processors High Performance Computing System

★ Located in the Advanced Computing Laboratory of the Mathematics Department

★ consists of eight (8) compute nodes connected by a 100Mbps Fast Ethernet and supports parallel programming using message passing and scientific computing software such as LAM-MPI, PVM and PETSc
Prerequisites

★ background in any programming language

★ familiar with basic unix commands
  ★ ls
  ★ cp
  ★ rm
  ★ cd

★ able to use unix compile and build tools
  ★ gcc
  ★ g++
  ★ make
  ★ libtool

★ familiar with basic parallel programming
Section II

Getting Started
Getting an AGILA Access Account

★ fill out the AGILA Access Request Form

★ take down the access information that will be provided

★ the AHPCG reserves the right to allow and deny access to the AGILA HPCS
Finding SSH

★ connect to the AGILA HPCS via a SSH client or Secure Shell Client

★ please notify you system administrator if you cannot locate the SSH client in your system

★ Linux/Unix SSH clients can be downloaded at:

★ Windows SSH clients can be downloaded at:
Configuring SSH

Profile Name: agila
Host Name: agila.math.admu.edu.ph
Port: 22
UID: username
Cipher Type: Blowfish
Section III

Navigating thru the System
Linux/Unix Commands

1. `man [command]` - displays the manual pages of the specified command
2. `pwd` - displays the path of the current directory
3. `ls [directory]` - displays the contents of the specified directory
4. `cd [directory]` - changes the current directory to the specified directory
5. `cp [source] [destination]` - copies a file from specified source to the specified destination
6. `mv [source] [destination]` - moves a file from specified source to the specified destination
7. `cat [file]` - displays the contents of the specified file
8. `less [file]` - displays the contents of the specified file and prompts the user when the display is filled
9. `exit` - to logout of the system
Text Editing Tools

★ vim is a text editor that is upwards compatible to Vi. It can be used to edit any ASCII text. It is especially useful for editing programs.

★ jed is a programmer’s text editor that provides color syntax highlighting. Emulation of Emacs, EDT, Wordstar, and Brief editors. Extensible in a language resembling C. Completely customizable. Editing TeX files with AUC-TeX style editing (BiBTeX support too). Folding support, and much more...

★ ed is a line-oriented text editor. It is used to create, display, modify and otherwise manipulate text files.
Uploading files to the AGILA HPCS

★ locate the FTP client

★ configure the FTP client
  
  ftp server: agila.math.admu.edu.ph
  port: 22
  name: username

★ enter user name and password
FTP Commands

★ ls [directory] - displays the contents of the specified directory

★ cd [directory] - changes the current directory to the specified directory

★ put [file] - upload a file to the AGILA server

★ mput [file] - upload a file to the AGILA server; this command supports wildcard characters

★ get [file] - gets a file from the AGILA server

★ mget [file] - gets a file from the AGILA server; this command supports wildcard characters

★ bye - to exit the system
Section IV

Building your Program
GNU Compiler Collection

- gcc - GNU ANSI-complaint C compiler
- g++ - GNU C++ compiler
- g77 - GNU ANSI-complaint Fortran 77 compiler

[username,~,09:13]$ g77 sample.f -o sample
[username,~,09:13]$ gcc sample.c -o sample
[username,~,09:13]$ g++ sample.cpp -o sample
Compiling LAM MPI Programs

★ hcc - LAM MPI C compiler wrapper

★ hcp - LAM MPI C++ compiler wrapper

★ hf77 - LAM MPI Fortran 77 compiler wrapper

[username,~,09:13]$ hf77 sample.f -o sample
[username,~,09:13]$ hcc sample.c -o sample
[username,~,09:13]$ hcp sample.cpp -o sample
Compiling MPICH MPI Programs

★ mpicc - MPICH MPI C compiler wrapper

★ mpiCC - MPICH MPI C++ compiler wrapper

★ mpif77 - MPICH MPI Fortran 77 compiler wrapper

[username,~,09:13]$ mpif77 sample.f -o sample
[username,~,09:13]$ mpicc sample.c -o sample
[username,~,09:13]$ mpiCC sample.cpp -o sample
Running MPI Programs

★ lamrun - LAM MPI execute binary wrapper

★ mpirun - MPICH MPI execute binary wrapper

[username,~,09:13]$ mpirun -np 8 myprogram
[username,~,09:13]$ lamrun -np 8 myprogram
Section V

Debugging your Program
Compile programs for debugging

★ add the -g compiler flag

★ call standard compiler/compiler wrappers to compile and build the programs

[username, ~, 09:13]$ hf77 -g sample.f -o sample
[username, ~, 09:13]$ hcc -g sample.c -o sample
[username, ~, 09:13]$ hcp -g sample.cpp -o sample
Running the Debuggers in Console Mode

Running the Debugger

[username,~,09:13]$ mpirun -np 8 rungdb.tcsh
  myprogram

rungdb.tcsh Debugger Script

#!/bin/kcsh -f

if ("$LAMRANK" == "0") then
  gdb $*
endif exit 0
Running the Debuggers in X-Windows

Running the Debugger

[username, ~, 09:13]$ mpirun -np 8 -x DISPLAY rungdbx.tcsh myprogram

rungdbx.tcsh Debugger Script

#!/bin/kcsh -f

echo "Running GDB on node 'hostname'"
xterm -e gdb $* exit 0
Section VI

Exercises