

# AGILA HPCS USERS GUIDE

WILLIAM EMMANUEL S. YU AND RAFAEL P. SALDAÑA

## 1. INTRODUCTION

The Ateneo High Performance Computing Group(AHPC) is a professional interdisciplinary research group that provides the Ateneo de Manila University community with high performance computing services. In August 2000, the AHPC developed a Beowulf-class cluster computer called the AGILA HPCS, which stands for the Ateneo Gigaflops-Range, Linux OS, and Athlon Processors High Performance Computing System. It 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.

Located in the Advanced Computing Laboratory of the Mathematics Department, the AGILA HPCS is an interdisciplinary project (mathematics, physics, engineering, and computer science) aimed at supporting the computational science and engineering research of the Ateneo de Manila University. It is also intended 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.

## 2. GETTING STARTED

**2.1. Requesting for an AGILA HPCS Access Account.** To get access to the AGILA HPCS, the user must first complete the AGILA HPCS Access Request Form(see Appendix A). The form must be filled out completely and submitted to the AHPC. The request will then be reviewed before access is finally granted. The AHPC reserves the right to grant and deny requests made.

**2.2. Connecting via Secure Shell.** The AGILA HPCS can be accessed via secure shell to the following address:

*agila.math.admu.edu.ph*

For Linux/Unix Users:

Secure shell can be accessed via the command *ssh*. If Secure shell is not installed in your system please contact the administrator of your system. Secure Shell Clients can be downloaded from:

*http://www.math.admu.edu.ph/ahpc/downloads/linux/ssl/*  
*http://www.math.admu.edu.ph/ahpc/downloads/linux/ssh/*

Select the appropriate version of Secure shell for your system. Redhat Package Manager(RPMs) are made available to ease installation. Install the Secure Socket Layer(SSL) package first and then install the SSH packages. For other systems the source code version of SSH can be found at <http://www.openbsd.org/>

For Windows Users:

Secure shell can be accessed via a windows SSH client. If your system does not have Secure Shell installed please contact the administrator of your system. A free software Secure Shell Client called **putty** for Win95/98/2000/NT can be obtained at:

<http://www.math.admu.edu.ph/ahpc/downloads/win32/ssh/putty.exe>

<http://www.math.admu.edu.ph/ahpc/downloads/win32/ssh/pscp.exe>

This client is maintained by Simon Tatham. Newer versions of this software can be found at <http://www.chiark.greenend.org.uk/~sgtatham/putty/>. Download the two files and save them on either c:\windows for win9x and c:\winnt for the winNT family. It is also possible to save these programs in different directories as long as these directories are in the executable search path. Load a command console or the run programs menu and type **putty**. Then you will be presented with a configuration dialog seen in Figure 1.

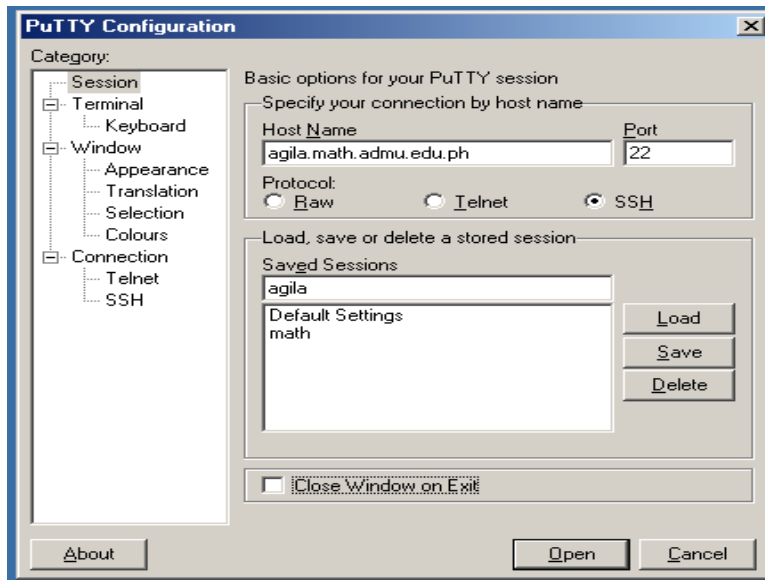


FIGURE 1. Logging in via win32 ssh

The settings below must be used to configure the SSH client to access the AGILA HPCS.

Host Name: agila.math.admu.edu.ph

Port: 22

Profile Name: agila

Click on the SSH tab to the left and configure the settings accordingly. The dialog will be similar to that presented in Figure 2. This screen deals with the type of encryption that the SSH client is going to use. The proper settings can be found below:

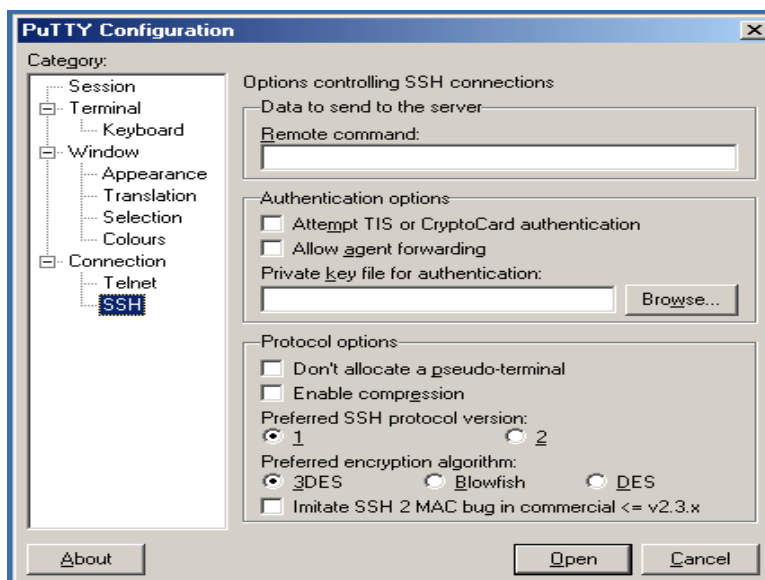


FIGURE 2. Configuring SSH client

Preferred Encryption Algorithm: Blowfish  
 Preferred SSH Protocol Version: 2  
 Enable Compression: Selected

Once completed, go back to the session tag in the left and click on the **save** button then click on **open**. You will now be prompted for a username and password.

**2.3. Uploading files to the AGILA HPCS.** It is advisable to write your code in a remote workstation. To upload your code to the AGILA HPCS system, you should use an scp client to move files from one machine to another. On a unix system the client is called **scp**. On a win32 system, a popular scp client is called **pscp**. This client can be downloaded together with the **putty** ssh client.

**scp** and **pscp** are command line tools for moving to and from a system using the SSH protocol. The AGILA HPCS does not support FTP anymore due to some security reasons and would encourage its users to use **scp** instead.

Usage:

- Move the file 'test.txt' from the local computer called local to the users current directory in the AGILA HPCS.
 

```
scp test.txt username@agila.math.admu.edu.ph:~/.
```

```
scp test.txt username@agila.math.admu.edu.ph:.
```

```
scp test.txt username@agila.math.admu.edu.ph:/home/username/.
```
- Move a file 'remote.txt' from the user's home directory in the AGILA HPCS to the current directory in the local machine.
 

```
scp username@agila.math.admu.edu.ph:~/remote.txt .
```

```
scp username@agila.math.admu.edu.ph:~/remote.txt .
```

```
scp username@agila.math.admu.edu.ph:/home/username/remote.txt .
```

For Win32 users, use **pscp** instead of **scp**. In a Unix system, the scp client contains a large number of options such as compressions and encryption algorithm selection. These options can be set in either the ssh configuration file (/etc/ssh/ssh.conf) or via command line options.

**2.4. Filesystem.** Home directories for AGILA users will be stored in /home/username where username is the UID of the AGILA user. The entire /home is shared via NFS. This means that this home directory can be utilized by the entire cluster without the need to explicitly move files to the individual nodes. Changes made to the files in the /home directories will be propagated throughout the cluster.

**2.5. Logging In.** The AGILA HPCS uses Linux as its operating system, therefore, logging into the system will entail connecting to the system via Secure shell as outlined above. The users will then be presented with a command line prompt as show below:

```
Last login: Wed Nov 15 08:52:24 2000 from 10.2.17.240
[username,~,09:13]$
```

The first line is simply a last login display. It shows the user when and where he logged on the last time. The second line is called the prompt. In the AGILA HPCS, the prompt is divided into three parts separated by commas. The first part is the login name of the user. The second part is the current directory. The third part is the current time. From here the user can type any linux command, edit, compile and run programs.

### 3. NAVIGATING THRU THE AGILA

**3.1. Basic Linux Commands.** Linux is a unix-like operating system and as such it contains commands that are similar if not the same as other unices such as SUN Solaris, IBM AIX and others. Below is a summary of some important unix commands:

- ★ **pwd** - displays the path of the current directory
- ★ **ls** [directory] - displays the contents of the specified directory
- ★ **cd** [directory] - changes the current directory to the specified directory
- ★ **cp** [source] [destination] - copies a file from specified source to the specified destination
- ★ **mv** [source] [destination] - moves a file from specified source to the specified destination
- ★ **rm** [source] - deletes a file from the specified source
- ★ **cat** [file] - displays the contents of the specified file
- ★ **mkdir** [directory] - creates a directory
- ★ **rmdir** [directory] - removes a directory
- ★ **less** [file] - displays the contents of the specified file and prompts the user when the display is filled
- ★ **man** [command] - displays the manual pages of the specified command
- ★ **exit** - to logout of the system

For a more detailed description of the following commands and other commands use the **man** command.

**3.2. Text Editors.** A handful of text editors are available for users who wish to make slight modifications to their code before compiling it in the system. However, the use of text editors is highly discouraged. Please refer to the next section for more instructions on how to upload files:

Available Text Editors:

- ★ **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.
- ★ **emacs** is a full screen text editor that is built on top of the lisp interpreter. One comment about emacs is it is "everything but the kitchen sink".
- ★ **joe** is a powerful ASCII-text screen editor. It has a "mode-less" user interface which is similar to many user-friendly PC editors. Users of Micro-Pro's WordStar or Borland's "Turbo" languages will feel at home. JOE is a full featured UNIX screen-editor though, and has many features for editing programs and text.
- ★ **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.

#### 4. EXECUTING PROGRAMS

There are many tools provided by the AGILA HPCS to execute programs in parallel. The AGILA HPCS provides the users with LAM-MPI, MPICH-MPI and PETSc as primary tools for development.

Examples on using these Message Passing Environments are found in the `/home/apps/samples` directory.

**4.1. Compiling Systems.** The AGILA HPCS supports Fortran 77, C and C++. It uses the free/open source GNU compilers that are part of the GNU Compiler Collection(GCC). Examples on compiling programs are shown below:

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

**4.2. MPI.** AGILA HPCS uses both the LAM and MPICH implementations of MPI. However, the user is encouraged to use the LAM implementation due to its debugging and profiling tools. MPI is a standard library for performing parallel programming using a distributed-memory model.

**4.2.1. Programming Notes:** Each program file using MPI must include the MPI header file. The following statement must appear near the beginning of each C/C++ or Fortran file, respectively.

```
#include <mpi.h>
include 'mpif.h'
```

4.2.2. *Starting LAM.* Before starting a LAM session, it is important to check if all nodes are properly working. In order to do this, execute the following commands. There must be no error message after running this.

```
[username,~,09:13]$ recon -v
```

Now that we know that the nodes are ready, the LAM environment must be started in order to run programs compiled using MPI. This LAM environment is initialized per user and thus each user will have his/her own. Each LAM environment does not affect another users LAM environment. To start LAM:

```
[username,~,09:13]$ lamboot -v
```

If this command is successful you will see the nodes initialized one at a time. This is only executed for LAM users. MPICH users will simply run the program without doing this form of initialization. However, it is recommended that the programmer uses LAM.

4.2.3. *Compiling for Parallel Runs.* To compile an MPI program, use the MPI wrappers around the gcc compilers. Here are some examples.

For LAM:

```
[username,~,09:13]$ hf77 sample.f -o sample
[username,~,09:13]$ hcc sample.c -o sample
[username,~,09:13]$ hcp sample.cpp -o sample
```

For MPICH:

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

4.2.4. *Running Parallel Programs.* To run programs in MPI, the *mpirun* command is used for MPICH and the *lamrun* command is used for LAM. The example below instructs the cluster to run the program *myprogram* in all 8 nodes of the cluster.

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

4.2.5. *Stopping LAM.* After running LAM programs, the LAM environment must be cleaned up and close. In order to do, this simply execute:

```
[username,~,09:13]$ wipe -v
```

Other details of using MPI will be provided in a separate document. Please consult the Ateneo High Performance Computing Group Website(<http://www.math.admu.edu.ph/ahpc/>) for more information.

## 5. DEBUGGING

The GNU compiler collection comes with a debugger called *gdb* for the console based debugger and *xgdb* for the graphical debugger. These debuggers can be used to provide user runtime information. In order to debug, programmers must first compile their programs with the *-g* option. Currently, the debugging methods presented below are specific to LAM only.

```
[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
```

To debug the program interactively, run the program with the appropriate debugging script. Below is an example.

```
[username,~,09:13]$ mpirun -np 8 runfdb.tcsh my_program_name
```

where *runfdb.tcsh* is a shell script. The contents of this shell script are as follows:

```
#!/bin/kcsh -f

if ("$LAMRANK" == "0") then
  gdb $*
endif exit 0
```

For GUI debuggers, you will probably need to export the *DISPLAY* environment variable, unless you are using *ssh* for LAM to launch remote programs. *ssh* sets the *DISPLAY* environment variable on remote nodes automatically; you probably do not want to override it (lest your X traffic be sent unencrypted). To debug the program interactively in X, run the debugger with the appropriate debugging script.

```
[username,~,09:13]$ mpirun -np 8 -x DISPLAY runfdbx.tcsh
my_program_name
```

where *runfdbx.tcsh* is a shell script. The contents of this shell script are as follows:

```
#!/bin/kcsh -f

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

## 6. SOFTWARE

The AGILA HPCS uses a variety of free or open source software development tools. Presented in Table 1 is a summary of the installed software currently available.

	Software Used
Linux Distribution	Redhat Linux 6.2 (Zoot)
Linux Kernel	Linux Kernel v. 2.2.18 w/ patches
Compilers	GCC 2.95.3 PGI Compilers 3.2-4
Message Passing Libraries	PVM 3.4.3, MPI-MPICH 1.2.1, MPI-LAM 6.5.2
Scientific Libraries	LAPACK, SCALAPACK, PETSc, FFTW
Benchmark Suites	HPL, NPB
Others	MPI-POVRAY, MM5 GAP, NWCHEM

TABLE 1. Software Installed in the AGILA Cluster as of Nov. 10, 2001

## 7. TRAINING

The Ateneo High Performance Computing Group provides regular training seminars for those interested in high performance computing. For more information about training contact:

The Coordinator  
 Ateneo High Performance Computing Group  
 Mathematics Department  
 Ateneo de Manila University  
 Loyola Heights, Quezon City  
 63(2)4266125, 63(2)4266001 local 5680/5683  
<http://www.math.admu.edu.ph/ahpc/>  
[raf@mathsci.math.admu.edu.ph](mailto:raf@mathsci.math.admu.edu.ph)

The Ateneo High Performance Computing Group Website also contains some documentation and other relevant information.

## 8. ACKNOWLEDGEMENT

This project was funded by the CHED Center of Excellence in Science and Mathematics Grant of the Ateneo de Manila University.

## REFERENCES

- [1] R. Saldaña, J. Garcia, F. Muga II, W. Yu. 2000. "AGILA: the Ateneo High Performance Computing System". Proceedings of the First Philippine Computing Science Congress, Manila, Philippines, Nov. 2000.
- [2] F. Muga II, and W. Yu. 2000. "A Proposed Topology for a 192-Processor Symmetric Cluster with a Single-Switch Delay". Proceedings of the First Philippine Computing Science Congress, Manila, Philippines, Nov. 2000.
- [3] R. Saldaña, J. Garcia, F. Muga II, W. Yu. 2000. "Development of a Beowulf-Class High Performance Computing System for Computational Science Application". Proceedings of the 18th National Physics Congress, Palawan, Philippines, Oct. 27-29,2000.

- [4] T. Sterling, D. Becker, D. Savarse, M. Bery, and C. Reshke. 1996. "Achieving a Balanced Low-Cost Architecture for Mass Storage Management through Multiple Fast Ethernet Channels on the Beowulf Parallel Workstation." *Proceedings of the International Conference on Parallel Processing*, IPPS96.
- [5] T. Sterling, D. Becker, and D. Savarese. 1995. Beowulf: A Parallel Workstation for Scientific Computation. *Proceedings of the Fourth 1995 International Conference on Parallel Processing (ICPP)*, August 1995, vol. 1, pp. 111-14.
- [6] D. Ridge, D. Becker, P. Merkey, T. Sterling. 1997. "Harnessing the Power of Parallelism in a Pile-of-PCs." *IEEE Aerospace*.
- [7] D. Becker, T. Sterling, D. Savarese, B. Fryxell, K. Olson. 1995. "Communication Overhead for Space Science Applications on the Beowulf Parallel Workstation." *High Performance and Distributed Computing*.
- [8] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, V. Sunderam. 1994. *PVM: Parallel Virtual Machine A User's Guide and Tutorial for Parallel Networked Computing*(Massachusetts:MIT 1994).
- [9] W. Gropp, E. Lusk, A. Skjellum. 1994. *USING MPI: Portable Parallel Programming with the Message-Passing Interface*(Massachusetts:MIT 1994).
- [10] M. Snir, S. Otto, S. Hans-Lederman, D. Walker, J. Dongarra. 1996. *MPI: The Complete Reference*(Massachusetts:MIT 1996).
- [11] W. Gropp, E. Lusk. "User's Guide for MPE: Extensions for MPI Programs". Available on the Internet as <http://http://www-unix.mcs.anl.gov/mpi/mpich/docs/mpeguide/paper.htm>.
- [12] Ohio Supercomputing Center. 1996. "MPI Primer/Developing with LAM." *National Science Foundation Grant CCR-9510016*.