Installation

What is HUBzero?

HUBzero is a platform used to create dynamic web sites for scientific research and educational activities. With HUBzero, you can easily publish your research software and related educational materials on the web. Powerful middleware serves up interactive simulation and modeling tools via your web browser. These tools can connect you with rendering farms and powerful Grid computing resources.

Minimum System Requirements

HUBzero installations require one or more dedicated physical hosts running Debian GNU/Linux 5.0.

> Other distributions might theoretically work with some modification, although they would be totally unsupported.

A typical starter HUBzero installation might consist of a single physical server with dual 64-bit quad-core CPUs, 16 Gigabytes of RAM and a terabyte of disk

It is possible to run HUBzero inside of a virtual machine such as ones created by VMware and VirtualBox. While fully functional there will would significant performance and resource limitations in such an environment.

Target Audience

This document and the installation of a HUBzero system has a target audience of experienced Linux administrators (preferably experienced with Debian GNU/Linux).

What's Included

HUBzero is composed of the following packages (subject to change):

Package Name	*Purpose*
hubzero-addrepo	Creates project areas for tool development
hubzero-apache2	Apache 2.2 Site Configuration Files for HUBzero
hubzero-app	HUBzero App Installer
hubzero-app-invoke	HUBzero application (tool) invocation scripts
hubzero-app-workspace	HUBzero App providing a lightweigth Linux desktop, for app/tool development

hubzero-cms	The HUBzero Content Management System (based on Joomla! framework)
hubzero-cms-joomla	Joomla! framework used by HUBzero
hubzero-cms-setup	Customizes the HUBzero CMS database
hubzero-config	Configures a HUBzero server and provides an automated installation script
hubzero-expire-sessions	Expires unused app/tool sessions
hubzero-filexfer	Transfer files between App Sessions and user's desktop
hubzero-filexfer-xlate	Support daemon for the filexfer program
hubzero-firewall	HUBzero firewall that protects app/tool sessions
hubzero-icewm	Linux ICE window manager configuration, used in workspaces
hubzero-icewm-captive	Linux ICE window manager specially crafted to support tools in a sessions
hubzero-icewm-themes	The HUBzero Linux ICE window manager theme, used in workspaces
hubzero-mw-client	HUBzero middleware - client
hubzero-mw-service	HUBzero middleware - execution host session manager
hubzero-mw-session	HUBzero middleware - per session tools
hubzero-mysql	MySQL configuration package for HUBzero
hubzero-openldap	OpenLDAP configuration package for HUBzero
hubzero-python	HUBzero python API module
hubzero-rappture	The Rapid APPlication infrastrucTURE toolkit for building scientific tools
hubzero-rappture-session	Session support packages for Rappture
hubzero-ratpoison-captive	Linux window manager, used in app/tool sessions
hubzero-submit-client	The session based part of the job submission server
hubzero-submit-distributor	Part of the job submission server
hubzero-submit-server	Part of the job submission server
hubzero-telequotad	Disk quota monitor
hubzero-texvc	Helper utility to generate math fomulas for wiki markup
hubzero-trac-mysqlauthz	Plug-in for MySQL user auth in project development areas
hubzero-twm-captive	Linux TWM window manager, used in app/tool sessions
hubzero-use	Command for configuring the environment within a workspace
hubzero-usermap	File permission mapping FUSE filesystem used by WebDAV
hubzero-vncproxy	Routes vnc between web server and app/tool session

icewm	Modified Linux ICE window manager, used in workspaces
tightvnc-java	Modified VNC Client that receives app/tool sessions within a web browser
vnc4server	Modified VNC Server that sends app/tool sessions to the web browser

Source Code

You'll find most of the source code within the web root of a working hub. But you can get source code for the middleware and all other parts by installing source code via the package mechanism. Please refer to Section 1.2.7 for instructions.

Debian Build

Install Basic Operating System

The latest version of Debian GNU/Linux 5.0 (5.0.8 as of this writing at http://www.debian.org/releases/lenny/debian-installer/) should be installed on each physical host used by a HUBzero installation. HUBzero has packaging support for i386 (32bit) and amd64 (64bit) Intel architectures. However the 32bit version gets more limited testing. Debian 6.0 is not supported at this time.

It is strongly recommended that you install from one of the ISO disk images listed in the above link. VM images (other than those distributed by HUBzero) and other boutique Debian installs are known to fail due to incomplete dependencies in the current HUBzero packaging.

Locale Settings

At the initial boot menu choose the command line install.

Choose your language, country, and keyboard (English, United States, and American English will be the default)

The next step is to enter the hostname and domain name for the system.

Choose the time zone for the system (the time zones will be limited to the country that was selected previously).

Disk Partitions

The recommended minimum for partitions sizes are 100 GB bootable partition for the root filesystem, 50 GB export filesystem for each HUBzero server, and an appropriately sized swap partition.

Partitioning method: (Guided - use entire disk)

Select disk to partition: (SCSI1 (0,0,0))

- Partitioning scheme: (All files in one partition)
- Select each partition and delete
- Select #1 - then Delete the partition
- Select #5 - then Delete the partition
- Select free space
- create a new partition
- set size (ex: 100 GB)

- type is primary
- location is beginning
- Done setting up partition
- Select free space
- create a new partition
- set size (ex: 1 GB)
- type is primary
- location is beginning
- Select Use As: set to swap area
- Done setting up partition
- Select free space
- create a new partition
- use all available space left
- type is primary
- Select mount point, change to do not mount it

Done setting up partition

Finish partitioning and write changes to disk

Want to return - say No

Write the changes to disks? Select Yes

Base System Accounts

The install will begin installing the base system then ask for the creation of the root password and to create a new user. It is suggested to skip the step of creating a new user.

- Skip the step of creating a new user
- Select (takes to root password screen)
- Select (takes to Debian installer main menu)
- Select Configure the package manager. (next one down)

Package Manager

Choose the package manager archive to use for downloads. ftp.us.debian.org is a good choice for the archive mirror for installs based in the U.S. and leave the HTTP proxy information blank unless your network requires it.

The install will begin retrieving a list of package options and configure base components. The

install may prompt to participate in the package usage survey. The decision to participate or not will not affect the install and is up to the individual system administrator.

In the Software Selection option, deselect â€˜Desktop environmentâ€™ but leave â€˜Standard systemâ€™ then choose to install the GRUB boot loader to the master boot record and finish the installation (when the installation is complete your system will reboot into a minimal Debian GNU/Linux system).

Debian GNU/Linux

Install Basic Operating System

The latest version of Debian GNU/Linux 5.0 (5.0.8 as of this writing) should be installed on each physical host used by a HUBzero installation.

To install Debian GNU/Linux, you can easily obtain a copy, and then follow the installation instructions for your architecture.

Installing Debian GNU/Linux using a a small bootable CD is the simplest method.

When installing Debian GNU/Linux be sure to do the following:

- Ensure the disk(s) are partitioned to have at least:
    - A bootable partition at least 100.0 GB in size for the root filesystem.
    - An empty partition at least 50.0 GB in size (note the device name of this partition for later)
    - An appropriately sized swap partition.
- When prompted to select an installation package just select "Standard System", other packages will be added later

When the installation is complete your system will reboot into a minimal Debian GNU/Linux system.

Don't forget to remove your installation media and/or change your server's boot media order if you changed them prior to installation.

Set hostname

Optional. If you didn't specify the fully qualified domain name when running setup you will need to set it here.

HUBzero expects the `hostname` (and `hostname -f`) command to return the fully qualified hostname for the system.

```
# hostname myhub.org
```

To make the change permanent you must also edit the file /etc/hostname, which can be done simply with:

```
# echo "myhub.org" > /etc/hostname
```

Fix hosts

Now edit /etc/hosts by making sure that a line exists that looks like

```
192.168.2.10    full-host-name optional-short-alias-of host
```

The number 192.168.2.10 must be replaced by the current IP number of your server. This is normally reported by the 'ifconfig eth0' command.

Do not remove the line that looks like

```
127.0.0.1     localhost
```

Delete local user

If you created a local user account when installing the operating system now would be a good time to delete it before it causes you future problems.

```
# deluser username
```

Configure Networking

Optional. If you didn't configure networking during installation you will need to do so now.

For help with networking setup try this link.

Setting up your IP address.

The IP addresses associated with any network cards you might have are read from the file **/etc/network/interfaces**. This file has documentation you can read with:

```
# man interfaces
```

A sample entry for a machine with a static address would look something like this:

```
# The loopback network interface
```

```
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
 address 192.168.1.90
 gateway 192.168.1.1
 netmask 255.255.255.0
 network 192.168.1.0
 broadcast 192.168.1.255
```

Here we've setup the IP addresss, the default gateway, and the netmask.

For a machine running DHCP the setup would look much simpler:

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface - use DHCP to find our address
auto eth0
iface eth0 inet dhcp
```

(If you're using a DHCP based setup you must have a DHCP client package installed - usually one of pump or dhcp-client.)

If you make changes to this file you can cause them to take effect by running:

```
# /etc/init.d/networking restart
```

Setting up DNS

Use whatever nameserver and other options as recommended by your ISP. If you used DHCP to set up networking it is likely this has already been set.

When it comes to DNS setup Debian doesn't differ from other distributions. To cause your machine to consult with a particular server for name lookups you simply add their addresses to

/etc/resolv.conf.

For example a machine which should perform lookups from the DNS server at IP address 192.168.1.10 would have a resolv.conf file looking like this:

```
nameserver 192.168.1.10
```

Configure Advanced Package Tool

Now configure what debian distribution mirror to use and the location of the HUBzero package repository by editing /etc/apt/sources.list to look like:

```
deb http://ftp.us.debian.org/debian/ lenny main
deb-src http://ftp.us.debian.org/debian/ lenny main

deb http://security.debian.org/ lenny/updates main
deb-src http://security.debian.org/ lenny/updates main

deb http://volatile.debian.org/debian-volatile lenny/volatile main
deb-src http://volatile.debian.org/debian-volatile lenny/volatile main

deb http://packages.hubzero.org/deb buck main contrib non-free
deb-src http://packages.hubzero.org/deb buck main contrib non-free
```

You will need to get and install the hubzero archive key to be able to verify packages from the hubzero archive:

```
# wget http://packages.hubzero.org/deb/hubzero-signing-
key.asc -q -O - | apt-key add -
```

Once the public key for http://packages.hubzero.org has been install you can then upgrade the current packages to their latest releases.

```
# apt-get update
# apt-get upgrade
```

SSH

Next we install fail2ban and ssh

```
# apt-get install fail2ban ssh
```

At this point you can continue configuration and setup remotely if that is more convenient.

Enable OpenVZ

If you are installing this in a VirtualBox VM you must enable PAE/NX support. Go to system -> processor of your VM, select "Enable PAE/NX".

To use OpenVZ you must use an OpenVZ enabled kernel which is easily installed.

HUBzero makes extensive use of OpenVZ containers so it is recommended to just use the OpenVZ enabled kernel on all HUBzero servers. To install a 64 bit kernel run the command:

```
# apt-get install linux-image-2.6-openvz-amd64
```

For 32 bit kernels, run the command:

```
# apt-get install linux-image-2.6-openvz-686
```

You will need to reboot the server to activate the new kernel.

```
# reboot
```

Once you have rebooted you can verify the new kernel is active

```
# uname -a
Linux myhub.hubzero.org 2.6.26-2-openvz-
amd64 #1 SMP Thu Nov 5 03:06:00 UTC 2009 x86_64 GNU/Linux
```

or for 32 bit kernels

```
# uname -a
Linux myhub.hubzero.org 2.6.26-2-openvz-686 #1 SMP Thu Nov 5 03:06:00
UTC 2009 x86_64 GNU/Linux
```

With the new kernel active you can optionally remove the old one. For 64 bit kernels run:

```
# apt-get purge linux-image-2.6.26-2-amd64
```

or for 32 bit kernels run:

```
# apt-get purge linux-image-2.6.26-2-686
```

Prepare Fileystem

The root filesystem ('/') runs with quotas disabled and contains the primary operating system for the server and for each OpenVZ container hosted on the server.

Each HUBzero server may use an addition partition for use appropriate to the function of the server (web document root, project data, home directories, etc).

If you did not create an empty partition during setup, create one now using your favorite disk partitioning tool. Be sure to note the device name for the partition you create as it will be used below.

Once you have an empty partition ready we can install a filesystem. Replace "/dev/PART" with the device name for the empty partition you have created (e.g., /dev/sda2). The command "fdisk -l" will list all paritions the system knows about.

```
# mke2fs -j /dev/PART
# e2fsck -f -C 0 /dev/PART
# mkdir /export
```

Then make sure the following line appears in /etc/fstab

```
/dev/PART     /export      ext3    defaults,quota,errors=remount-
ro     0     2
```

Then mount the new filesystem

```
# mount /export
```

Bind mount /home

Create a 'home' directory in our new /export filesystem. move the contents of the default home directory to the new location, then bind mount new location over the old.

```
# mkdir -p /export/home/myhub
# mv /home/* /export/home
# mount --bind /export/home /home
```

Bind mount /opt

Currently HUBzero uses the /opt directory for storing subversion and trac data as well as some of hubzero supporting software as well. We recognize this may not be the best organization.

Create a 'opt' directory in our new /export filesystem. move the contents of the default /opt directory to the new location, then bind mount new location over the old.

```
# mkdir /export/opt
# mv /opt/* /export/opt
# mount --bind /export/opt /opt
```

Bind mount /apps

Currently HUBzero uses the /apps directory for storing installed tools and other software that needs to be available to each execution container.

Create a 'apps' directory in our new /export filesystem and in the root filesystem. Then bind mount /export/apps over /apps.

```
# mkdir /export/apps
# mkdir /apps
# mount --bind /export/apps /apps
```

Bind mount /www

HUBzero uses the /www directory for storing the document root and supporting directories needed by the web server.

Create a 'www' directory in our new /export filesystem and in the root filesystem. Then bind mount /export/www over /www.

```
# mkdir -p /export/www
# mkdir /www
# mount --bind /export/www /www
```

Update /etc/fstab

Now edit **/etc/fstab** with the bind mounts we created above by adding the following lines

```
/export/opt  /opt        none    bind,defaults
0     0
/export/apps /apps       none    bind,defaults
0     0
/export/home /home       none    bind,defaults
0     0
/export/www  /www        none    bind,defaults
0     0
```

Hubzero Installation

Hubzero Software Installation

This section describes the automated installation of the HUBzero software. It assumes that the server has been setup with a basic Debian installation as described in the previous sections of the documentation.

Note It is very important that the server hostname be described properly in the file /etc/hostname, that the server's fully qualified domain name be accurately reported by the 'hostname -f' command and that the file /etc/hosts contain an accurate mapping of the server's IP address to fully qualified hostname. Please refer to the previous section of the documentation for details.

To begin the process of installing HUBzero software, run the following command:

```
# apt-get install hubzero-config
```

This package will ask 12 questions - 6 pertaining to system variables that need to be set and another 6 passwords needed by the system. If the fully qualified domain name is set properly, 5 of the first 6 questions will contain the proper value by default. The other question is for an email address that will be used in several places for system notification messages.

The six passwords all require some kind of input. Note that the final password is for the super administrator *admin* user. This is the only one that will normally be used by the site maintainer.

Once the questions are answered the package will leave a script called *hz-install* in the /root directory. This script will use the information previously entered to fully construct a working HUBzero server. Run the command by typing:

```
# bash ./hz-install
```

The software installation process will then start and can take quite a while to finish depending on the server's network speed. Once the process is complete, you should be able to access your HUBzero server using a browser. You can log in with the *admin* account and begin customizing your site.

Mail

Install

We need to reconfigure exim4 to enable outgoing email (exim4 got installed earlier as a prerequisite for the mysql server).

```
# dpkg-reconfigure exim4-config
```

- Select "internet site; mail is sent and received directly using SMTP" then configure as appropriate for your site.
- Enter the FQDN of your site when asked
- Listen on all IP addresses (i.e., make list blank)
- Other destinations for which mail is accepted: should be made blank
- Domains to relay mail for: should be made blank
- Machines to relay mail for: should be made blank
- Keep num,ber of DNS-queries minimal (Dial-on-Demand): No
- Delivery method for local mail: mbox format in /var/mail/
- Split configuration into small files? No

This is just an example. Mail should be configured however the site needs. The CMS just expects to be able send outgoing email.

Rappture

Install

The Rappture application is install in the apps directory along with proper links.

```
# apt-get install hubzero-rappture
```

Session Installation

Rappture is used from inside a container and needs several other packages installed to allow use of all its features. This process has been simplified by using the hubzero-rappture-session with only contains the dependencies needed to pull in these other packages.

```
# chroot /var/lib/vz/template/debian-5.0-amd64-maxwell
# apt-get update
# apt-get upgrade
# apt-get install hubzero-rappture-session
```

This is also a good time to add some default paths to your session environment so that it doesn't need to be whenever someone logs in. Modify the /etc/profile file as follows:

Change

```
if [ "`id -u`" -eq 0 ]; then
  PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
else
  PATH="/usr/local/bin:/usr/bin:/bin:/usr/games"
fi
```

To

```
if [ "`id -u`" -eq 0 ]; then
  PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
else
  PATH="/apps/rappture/bin:/apps/bin:
/usr/local/bin:/usr/bin:/bin:/usr/games"
```

```
fi
```

Be sure to exit the chroot environment when you are done:

```
# exit
```

A workspace may need to be opened and closed a few times before the changes to the session template appear in a workspace.

Test

Rappture comes with several demostration scripts that can effectively test many parts of the package. These demonstrations must be copied to a user's home directory within a workspace before running.

```
$ mkdir examples
$ cp -r /apps/rappture/examples/* examples/.
$ cd examples
$ ./demo.bash
```

A window should open on the workspace showing that part of the demonstration. Close that window to see the next demonstration. Some demonstrations may need something inputted to work properly (such as the graphing calculator).

Filexfer

Install filexfer

Install the filexfer packages

```
# apt-get install hubzero-filexfer
```

```
# apt-get install hubzero-filexfer-xlate
```

Configure Apache for filexfer

Modify the hub site file at /etc/apache2/sites-available/hub-ssl

The apache hub site configuration files are preconfigured to support this. Uncommented the two highlighted lines as shown below. Note that the relative placement of these lines is important.

Edit **/etc/apache2/sites-available/hub-ssl**

```
<VirtualHost *:443>
        RewriteEngine   on
        RewriteMap      xlate           prg:/usr/lib/mw/bin/filexfer-
xlate
        ...
        ...
        ...
        <Directory /www/myhub>
                Options FollowSymLinks
                AllowOverride None
                Order allow,deny
                Allow from all

                RewriteEngine   On

RewriteRule     ^filexfer/(.*) ${xlate:$1|nothing} [P,QSA,L]
```

Then restart apache

```
# /etc/init.d/apache2 restart
```

Submit

Introduction

The submit command provides a means for HUB end users to execute applications on remote resources. The end user is not required to have knowledge of remote job submission mechanics. Jobs can be submitted to traditional queued batch systems including PBS and Condor.

Installation

```
# apt-get install hubzero-app-submit
# apt-get install hubzero-submit-server
# apt-get install hubzero-submit-distributor
```

At completion of the apt-get install commands several files will be located in the directory /opt/submit. Excluding python files the directory listing should like the following:

Configuration

submit provides a mechanism to execute jobs on machines outside the HUB domain. To accomplish this feat some configuration is required on the HUB and some additional software must be installed and configured on hosts in remote domains. Before attempting to configure submit it is necessary to obtain access to the target remote domain(s). The premise is that a single account on the remote domain will serve as an execution launch point for all HUB end users. It is further assumes that access to this account can be made by direct ssh login or using an ssh tunnel (port forwarding).

Having attained account access to one or more remote domains it is possible to proceed with submit configuration. To get started the ssh public generated by the installation should be transferred to the remote domain host(s).

HUB Configuration

The behavior of submit is controlled through a set of configuration files. The configuration files contain descriptions of the various parameters required to connect to a remote domain, exchange files, and execute simulation codes. There are separate files for defining remote sites, staged tools, multiprocessor managers, permissible environment variables, remote job monitors, and ssh tunneling. Most parameters have default values and it is not required that all

parameters be explicitly defined in the configuration files. A simple example is given for each category of configuration file.

Sites

Remote sites are defined in the file sites.dat. Each remote site is defined by a stanza indicating an access mechanism and other account and venue specific information. Defined keywords are

- [name] - site name. Used as command line argument (-v/--venue) and in tools.dat (destinations)
- venues - comma separated list of hostnames. If multiple hostnames are listed one site will chosen at random.
- tunnelDesignator - name of tunnel defined in tunnels.dat.
- siteMonitorDesignator - name of site monitor defined in monitors.dat.
- venueMechanism - possible mechanisms are ssh and local.
- remoteUser - login user at remote site.
- remoteBatchAccount - some batch systems requirement that an account be provided in addition to user information.
- remoteBatchSystem - the possible batch submission systems include CONDOR, PBS, and LSF. SCRIPT may also be specified to specify that a script will be executed directly on the remote host.
- remoteBatchQueue - when remoteBatchSystem is PBS the queue name may be specified.
- remoteBatchPartition - slurm parameter to define partition for remote job
- remoteBatchPartitionSize - slurm parameter to define partition size, currently for BG machines.
- remoteBatchConstraints - slurm parameter to define constraints for remote job
- remoteBinDirectory - define directory where shell scripts related to the site should be kept.
- remoteApplicationRootDirectory - define directory where application executables are located.
- remoteScratchDirectory - define the top level directory where jobs should be executed. Each job will create a subdirectory under remoteScratchDirectory to isolated jobs from each other.
- remotePpn - set the number of processors (cores) per node. The PPN is applied to PBS and LSF job description files. The user may override the value defined here from the command line.
- remoteManager - site specific multi-processor manager. Refers to definition in managers.dat.
- remoteHostAttribute - define host attributes. Attributes are applied to PBS description files.
- stageFiles - A True/False value indicating whether or not files should be staged to remote site. If the the job submission host and remote host share a file system file staging may not be necessary. Default is True.

- passUseEnvironment - A True/False value indicating whether or not the HUB 'use' environment should passed to the remote site. Default is False. True only makes sense if the remote site is within the HUB domain.
- arbitraryExecutableAllowed - A True/False value indicating whether or not execution of arbitrary scripts or binaries are allowed on the remote site. Default is True. If set to False the executable must be staged or emanate from /apps.
- members - a list of site names. Providing a member list gives a layer of abstraction between the user facing name and a remote destination. If multiple members are listed one will be randomly selected for each job.
- state - possible values are enabled or disabled. If not explicitly set the default value is enabled.
- failoverSite - specify a backup site if site is not available. Site availability is determined by site probes.
- checkProbeResult - A True/False value indicating whether or not probe results should determine site availability. Default is True.
- restrictedToUsers - comma separated list of user names. If the list is empty all users may garner site access. User restrictions are applied before group restrictions.
- restrictedToGroups - comma separated list of group names. If the list is empty all groups may garner site access.
- logUserRemotely - maintain log on remote site mapping HUB id, user to remote batch job id. If not explicitly set the default value is False.
- undeclaredSiteSelectionWeight - used when no site is specified to choose between sites where selection weight > 0.

An example stanza is presented for a site that is accessed through ssh.

```
[cluster]
venues = cluster.campus.edu
remotePpn = 8
remoteBatchSystem = PBS
remoteBatchQueue = standby
remoteUser = yourhub
remoteManager = mpich-intel64
venueMechanism = ssh
remoteScratchDirectory = /scratch/yourhub
siteMonitorDesignator = clusterPBS
```

Tools

Staged tools are defined in the file tools.dat. Each staged tool is defined by a stanza indicating an where a tool is staged and any access restrictions. The existence of a staged tool at multiple sites can be expressed with multiple stanzas or multiple destinations within a single stanza. If the tool requires multiprocessors a manager can also be indicated. Defined keywords are

- [name] - tool name. Used as command line argument to execute staged tools. Repeats are permitted to indicate staging at multiple sites.
- destinations - comma separated list of destinations. Destination may exist in sites.dat or be a grid site defined by a ClassAd file.
- executablePath - path to executable at remote site. The path may be given as an absolute path on the remote site or a path relative to remoteApplicationRootDirectory defined in sites.dat.
- restrictedToUsers - comma separated list of user names. If the list is empty all users may garner tool access. User restrictions are applied before group restrictions.
- restrictedToGroups - comma separated list of group names. If the list is empty all groups may garner tool access.
- environment - comma separated list of environment variables in the form e=v.
- softenvExtensions - per site softenv environment declaration for TeraGrid sites running GRAM4.
- remoteManager - tool specific multi-processor manager. Refers to definition in managers.dat. Overrides value set by site definition.
- state - possible values are enabled or disabled. If not explicitly set the default value is enabled.

An example stanza is presented for a staged tool maintained in the yourhub account on a remote site.

```
[earth]
destinations = cluster
executablePath = ${HOME}/apps/planets/bin/earth.x
remoteManager = mpich-intel

[sun]
destinations = cluster
executablePath = ${HOME}/apps/stars/bin/sun.x
remoteManager = mpich-intel
```

Monitors

Remote job monitors are defined in the file monitors.dat. Each remote monitor is defined by a stanza indicating where the monitor is located and to be executed. Defined keywords are

- [name] - monitor name. Used in sites.dat (siteMonitorDesignator)
- venue - hostname upon which to launch monitor daemon. Typically this is a cluster headnode.
- venueMechanism - monitoring job launch process. The default is ssh.
- tunnelDesignator - name of tunnel defined in tunnels.dat.
- remoteUser - login user at remote site.
- remoteMonitorCommand - command to launch monitor daemon process.
- state - possible values are enabled or disabled. If not explicitly set the default value is enabled.

An example stanza is presented for a remote monitor tool used to report status of PBS jobs.

```
[clusterPBS]
venue = cluster.campus.edu
remoteUser = yourhub
remoteMonitorCommand = ${HOME}/SubmitMonitor/monitorPBS.py
```

Multi-processor managers

Multiprocessor managers are defined in the file managers.dat. Each manager is defined by a stanza indicating the set of commands used to execute a multiprocessor simulation run. Defined keywords are

- [name] - manager name. Used in sites.dat and tools.dat.
- computationMode - indicate how to use multiple processors for a single job. Recognized values are mpi, parallel, and matlabmpi. Parallel application request multiprocess have there own mechanism for inter process communication. Matlabmpi is used to enable the an Matlab implementation of MPI.
- preManagerCommands - comma separated list of commands to be executed before the manager command. Typical use of pre manager commands would be to define the environment to include a particular version of MPI amd/or compiler, or setup MPD.
- managerCommand - manager command commonly mpirun. It is possible to include strings that will be sustituted with values defined from the command line.
- postManagerCommands - comma separated list of commands to be executed when the manager command completes. A typical use would be to terminate an MPD setup.
- mpiRankVariable - define environment variable set by manager command to define

process rank. Recognized values are: MPIRUN_RANK, GMPI_ID, RMS_RANK, MXMPI_ID, MSTI_RANK, PMI_RANK, and OMPI_MCA_ns_nds_vpid. If no variable is given an attempt is made to determine process rank from command line arguments.
- environment - comma separated list of environment variables in the form e=v.
- moduleInitialize - initialize module script for sh
- modulesUnload - modules to be unloaded clearing way for replacement modules
- modulesLoad - modules to load to define mpi and other libraries
- state - possible values are enabled or disabled. If not explicitly set the default value is enabled.

An example stanza is presented for a typical MPI instance. The given command should be suitable for /bin/sh execution.

```
[mpich-intel]
preManagerCommands = . ${MODULESHOME}/init/sh, module load mpich-
intel/11.1.038
managerCommand = mpirun -machinefile ${PBS_NODEFILE} -np NPROCESSORS
```

The token NPROCESSORS is replaced by an actual value at runtime.

Environment variables

Legal environment variables are listed in the file environmentwhitelist.dat. The objective is to

prevent end users from setting security sensitive environment variables while allowing application specific variables to be passed to the remote site. Environment variables required to define multiprocessor execution should also be included. The permissible environment variables should be entered as a simple list - one entry per line. An example file allowing use of a variables used by openmp and mpich is presenter here.

```
# environment variables listed here can be specified from the command
line with -e/--env option. Attempts to specify other environment varia
bles will be ignored and the values will not be passed to the remote s
ite.

OMP_NUM_THREADS
MPICH_HOME
```

Tunnels

In some circumstances access to clusters is restricted such that only a select list of machines is allowed to communicate with the cluster job submission node. The machines that are granted such access are sometimes referred to as gateways. In such circumstances ssh tunneling or port forwarding can be used to submit HUB jobs through the gateway machine. Tunnel definition is specified in the file tunnels.dat. Each tunnel is defined by a stanza indicating gateway host and port information. Defined keywords are

- [name] - tunnel name.
- venue - tunnel target host.
- venuePort - tunnel target port.
- gatewayHost - name of the intermediate host.
- gatewayUser - login user on gatewayHost.
- localPortOffset - local port offset used for forwarding. Actual port is localPortMinimum + localPortOffset

An example stanza is presented for a tunnel between the HUB and a remote venue by way of

```
[cluster]
venue = cluster.campus.edu
venuePort = 22
gatewayHost = gateway.campus.edu
gatewayUser = yourhub
localPortOffset = 1
```

Initialization Scripts and Log Files

The submit server and job monitoring server must be started as daemon processes running on the the submit host. If ssh tunneling is going to be used an addition server must be started as a daemon process. Each daemon process writes to a centralized log file facilitating error recording and debugging.

Initialize daemon scripts

Scripts for starting the server daemons are provided and installed in /etc/init.d. The default settings for when to start and terminate the scripts are adequate.

Log files

Submit processes log information to files located in the /var/log/submit directory tree. The exact location varies depending on the vintage of the installation. Each process has its own log file. The three most important log files are submit.log, distributor.log, and monitorJob.log.

submit.log

The submit.log file tracks when the submit server is started and stopped. Each connection from the submit client is logged with the command line and client ip address reported. All log entries are timestamped and reported by client ip address or submit ID once an ID has been assigned. Entries from all jobs are simultaneously reported and intermingled. The submit ID serves as a good search key when tracing problems. Examples of startup, job execution, and termination are given here. The job exit status and time metrics are also recorded in the MyQSL database JobLog table.

```
[Sat Jan 21 14:32:39 2012] Startup: Using configdir /opt/submit
[Sat Jan 21 14:32:39 2012] Startup: Backgrounding process.
[Sat Jan 21 14:32:39 2012] Startup: Listening: protocol='tcp', host=''
```

```
, port=830
[Sat Jan 21 14:32:39 2012] Startup: Listening: protocol='tls', host=''
, port=831
```

```
[Thu Feb  2 11:55:57 2012] 128.46.19.176: Connection to tls://:831 fro
m ('128.46.19.176', 49737)
[Thu Feb  2 11:55:57 2012] 128.46.19.176: Server will time out in 60 s
econds.
[Thu Feb  2 11:55:57 2012] 128.46.19.176: Server will time out in 60 s
econds.
[Thu Feb  2 11:55:58 2012] 128.46.19.176: Args are:['/apps/bin/submit'
, '-n', '8', '-v', 'coates', '-w', '4.000000:00:00', '-i', 'BSLAB_512_
RUN', 'bandstrlab-r2091', '/home/nanohub/clarksm/data/sessions/460378L
/BSLAB_512_RUN/OMEN_input_1_512.cmd']
[Thu Feb  2 11:55:58 2012] 2190962: The filesystem is shared.
[Thu Feb  2 12:07:58 2012] 2190962: Job Status: venue=1:sshPBS:3934644
:coates.rcac.purdue.edu status=0 cpu=200.610000 real=36.000000
 wait=619.000000
[Thu Feb  2 12:07:58 2012] 2190962: Server exiting.
```

```
[Thu Feb  2 09:38:30 2012] Startup: Server was terminated by a signal.
[Thu Feb  2 09:38:30 2012] Startup: Job Status: venue=any status=65534
 cpu=0.000000 real=0.000000 wait=0.000000
[Thu Feb  2 09:38:30 2012] Startup: EXCEPTION IN MAINLOOP: int argumen
t required
[Thu Feb  2 09:38:30 2012] Startup: Server fell out of mainloop().
[Thu Feb  2 09:38:30 2012] Startup: Server exiting.
```

distributor.log

The distributor.log file tracks each job as it progresses from start to finish. Details of remote site assignment, queue status, exit status, and command execution are all reported. All entries are timestamped and reported by submit ID. The submit ID serves as the key to join data reported in submit.log. An example for submit ID 2190962 is listed here. Again the data for all jobs are intermingled.

```
[Thu Feb  2 11:55:59 2012] 2190962: command = tar vchf 02190962_01_inp
```

```
ut.tar --exclude='*.svn*' -C /home/nanohub/clarksm/data/sessions/46037
8L .__local_jobid.02190962_01 BSLAB_512_RUN -C /home/nanohub/clarksm/d
ata/sessions/460378L/BSLAB_512_RUN OMEN_input_1_512.cmd
[Thu Feb  2 11:55:59 2012] 2190962: remoteCommand bandstrlab-
r2091 ./BSLAB_512_RUN/OMEN_input_1_512.cmd
[Thu Feb  2 11:55:59 2012] 2190962: command = genuserid
[Thu Feb  2 11:55:59 2012] 2190962: IDENTITY = /tmp/id.uBYdxy4FUw
[Thu Feb  2 11:55:59 2012] 2190962: command = update-known-
hosts coates.rcac.purdue.edu
[Thu Feb  2 11:55:59 2012] 2190962: workingDirectory /scratch/lustreA/
n/nano0/nanoHUBjobs/1328219759_02190962_01
[Thu Feb  2 11:55:59 2012] 2190962: command = tar vrhf 02190962_01_inp
ut.tar --exclude='*.svn*' -C /home/nanohub/clarksm/data/sessions/46037
8L/02190962_01 02190962_01.pbs 02190962_01.sh
[Thu Feb  2 11:55:59 2012] 2190962: command = nice -n 19 gzip 02190962
_01_input.tar
[Thu Feb  2 11:55:59 2012] 2190962: command = cat /home/nanohub/clarks
m/data/sessions/460378L/02190962_01/02190962_01_input.tar.gz | ssh -T
-x -a -i /tmp/id.uBYdxy4FUw nano0@coates.rcac.purdue.edu "${HOME}/bin/
receiveinput.sh /scratch/lustreA/n/nano0/nanoHUBjobs/1328219759_021909
62_01 .__timestamp_transferred.02190962_01"
[Thu Feb  2 11:56:01 2012] 2190962: .__local_jobid.02190962_01
[Thu Feb  2 11:56:01 2012] 2190962: command = ssh -T -x -a -i /tmp/id.
uBYdxy4FUw nano0@coates.rcac.purdue.edu "${HOME}/bin/submitbatchjob.sh
 /scratch/lustreA/n/nano0/nanoHUBjobs/1328219759_02190962_01 ./0219096
2_01.pbs"
[Thu Feb  2 11:56:01 2012] 2190962: remoteJobId = 3934644.coates-
adm.rcac.purdue.edu
[Thu Feb  2 11:56:01 2012] 2190962: confirmation: S(1):N Job
[Thu Feb  2 11:56:01 2012] 2190962: status:Job N coates
[Thu Feb  2 11:56:06 2012] 2190962: status:Simulation Q coates
[Thu Feb  2 12:07:42 2012] 2190962: status:Simulation D coates
[Thu Feb  2 12:07:42 2012] 2190962: waitForBatchJobs: nCompleteRemoteJ
obIndexes = 1, nIncompleteJobs = 0, abortGlobal = False
[Thu Feb  2 12:07:42 2012] 2190962: command = ssh -T -x -a -i /tmp/id.
uBYdxy4FUw nano0@coates.rcac.purdue.edu "${HOME}/bin/transmitresults.s
h /scratch/lustreA/n/nano0/nanoHUBjobs/1328219759_02190962_01" | tar x
zmf - --ignore-case --exclude '*hub-
proxy.*' -C /home/nanohub/clarksm/data/sessions/460378L/02190962_01
[Thu Feb  2 12:07:57 2012] 2190962: command = ssh -T -x -a -i /tmp/id.
uBYdxy4FUw nano0@coates.rcac.purdue.edu "${HOME}/bin/cleanupjob.sh /sc
ratch/lustreA/n/nano0/nanoHUBjobs/1328219759_02190962_01"
[Thu Feb  2 12:07:58 2012] 2190962: venue=1:sshPBS:3934644:coates.rcac
.purdue.edu status=0 cputime=200.610000 realtime=36.000000 waittime=61
9.000000 ncpus=8
```

monitorJob.log

The monitorJob.log file tracks the invocation and termination of each remotely executed job monitor. The remote job monitors are started on demand when job are submitted to remote sites. The remote job monitors terminate when all jobs complete at a remote site and no new activity has been initiated for a specified amount of time - typically thirty minutes. A typical report should look like:

```
[Thu Feb  2 11:05:53 2012] (22140) **********************************
[Thu Feb  2 11:05:53 2012] (22140) * distributor job monitor started *
[Thu Feb  2 11:05:53 2012] (22140) **********************************
[Thu Feb  2 11:05:53 2012] (22140) loading active jobs
[Thu Feb  2 11:05:53 2012] (22140) 73 jobs loaded from DB file
[Thu Feb  2 11:05:53 2012] (22140) 73 jobs loaded from dump file
[Thu Feb  2 11:05:53 2012] (22140) 2 jobs purged
[Thu Feb  2 11:05:53 2012] (22140) 71 monitored jobs
[Thu Feb  2 11:10:33 2012] (22311) Launching coates
[Thu Feb  2 11:10:33 2012] (22140) 72 monitored jobs
[Thu Feb  2 11:10:44 2012] (22140) Update message received from coates
[Thu Feb  2 11:12:14 2012] (22140) Update message received from coates
[Thu Feb  2 11:18:22 2012] (22629) Launching steele-fe01
[Thu Feb  2 11:18:22 2012] (22140) 73 monitored jobs
[Thu Feb  2 11:19:53 2012] (22140) Update message received from steele-
fe01
[Thu Feb  2 11:21:28 2012] (22140) Update message received from steele-
fe01
[Thu Feb  2 11:50:02 2012] (22629) Closing steele-fe01
[Thu Feb  2 11:51:28 2012] (1420) **********************************
[Thu Feb  2 11:51:28 2012] (1420) * distributor job monitor stopped *
[Thu Feb  2 11:51:28 2012] (1420) **********************************
```

It is imperative that the job monitor be running in order for notification of job progress to occur. If users report that their job appears to hang check to make sure the job monitor is running. If necessary take corrective action and restart the daemon.

monitorTunnel.log

The monitorTunnel.log file tracks invocation and termination of each ssh tunnel connection. If users report problems with job submission to sites accessed via an ssh tunnel this log file should be checked for indication of any possible problems.

Remote Domain Configuration

For job submission to remote sites via ssh it is necessary to configure a remote job monitor and a set of scripts to perform file transfer and batch job related functions. A set of scripts can be used for each different batch submission system or in some cases they may be combined with appropriate switching based on command line arguments. A separate job monitor is need for each batch submission system. Communication between the HUB and remote resource via ssh

requires inclusion of a public key in the authorized_keys file.

Job monitor daemon

A remote job monitor runs a daemon process and reports batch job status to a central job monitor located on the HUB. The daemon process is started by the central job monitor on demand. The daemon terminates after a configurable amount of inactivity time. The daemon code needs to be installed in the location declared in the monitors.dat file. The daemon requires some initial configuration to declare where it will store log and history files. The daemon does not require any special privileges any runs as a standard user. Typical configuration for the daemon looks like this:

The directory defined by MONITORLOGLOCATION needs to be created before the daemon is started. Sample daemon scripts used for PBS, LSF, Condor, Load Leveler, and Slurm batch systems are included in directory BatchMonitors.

File transfer and batch job scripts

The simple scripts are used to manage file transfer and batch job launching and termination. The location of the scripts is entered in sites.dat.

Examples scripts suitable for use with PBS, LSF, Condor, Load Leveler, and Slurm are included

in directory Scripts. After modifications are made to monitors.dat the central job monitor must be notified. This can be accomplished by stopping and starting the submon daemon or a HUP signal can be sent to the monitorJob.py process.

File transfer - input files

Receive compressed tar file containing input files required for the job on stdin. The file transferredTimestampFile is used to determine what newly created or modified files should be returned to the HUB.

```
receiveinput.sh jobWorkingDirectory transferredTimestampFile
```

Batch job script - submission

Submit batch job using supplied description file. If arguments beyond job working directory and batch description file are supplied an entry is added to the remote site log file. The log file provides a record relating the HUB end user to the remote batch job identifier. The log file should be placed at a location agreed upon by the remote site and HUB.

```
submitbatchjob.sh jobWorkingDirectory jobDescriptionFile
```

The jobId is returned on stdout if job submission is successful. For an unsuccessful job submission the returned jobId should be -1.

File transfer - output files

Return compressed tar file containing job output files on stdout.

```
transmitresults.sh jobWorkingDirectory
```

File transfer - cleanup

Remove job specific directory and any other dangling files

```
cleanupjob.sh jobWorkingDirectory
```

Batch job script - termination

Terminate given remote batch job. Command line arguments specify job identifier and batch system type.

```
killbatchjob.sh jobId jobClass
```

Access Control Mechanisms

By default tools and sites are configured so that access is granted to all HUB members. In some cases it is desired to restrict access to either a tool or site to a subset of the HUB membership. The keywords restrictedToUsers and restrictedToGroups provide a mechanism to apply restrictions accordingly. Each keyword should be followed by a list of comma separated values of userids (logins) or groupids (as declared when creating a new HUB group). If user or group restrictions have been declared upon invocation of submit a comparison is made between the restrictions and userid and group memberships. If both user and group restrictions are declared the user restriction will be applied first, followed by the group restriction.

In addition to applying user and group restrictions another mechanism is provided by the boolean keyword arbitraryExecutableAllowed in the sites configuration file. In cases where the executable program is not pre-staged at the remote sites the executable needs to be transferred along with the user supplied inputs to the remote site. Published tools will have their executable program located in the /apps/tools/revision/bin directory. For this reason submitted programs that reside in /apps are assumed to be validated and approved for execution. The same cannot be said for programs in other directories. The common case where such a situation arises is when a tool developer is building and testing within the HUB workspace environment. To grant a tool developer the permission to submit such arbitrary applications the site configuration must allow arbitrary executables and the tool developer must belong the system group submit.