PVM Administration at RFPK

The administration of the Parallel Virtual Machine (PVM) in the RFPK lab is described.

Table of Contents

Introduction

The Parallel Virtual Machine (pvm) is a software system which allows spk to be distributed over any arbitrary collection of computers running the linux or unix operating system. Each computer is a node in the pvm, capable of participating in the spk computation. With the system correctly configured, nodes can be added and deleted at will.

This document assumes that all nodes in the pvm are running the same or closely related versions of the same operating system. In the current version of this document, that operating system is RedHat Enterprise Linux, Version 3 (RHEL3).

A key feature of the configuration described is that all of the software and many of the configuration files are shared via the Network Files System (NFS). This greatly reduces the overall system administration required.

Another important feature of this configuration is that different processor architectures are accommodated in the same configuration.

Network Security

An important part of the pvm architecture is a set of daemons, one on each host, which provide task life-cycle and inter-task communications services. In the basic configuration, these daemons communicate between themselves using the *rsh* internet service, without passwords. This is very insecure.

As an alternative, *ssh* can be used and that is the approach described in this document. With ssh, communication between nodes is secure, even if they are on a subnetwork that is open to the public internet. Public key encryption is used, so that a passcode is required only when the head node is booted. When, on the other hand, an ordinary node is booted, the daemon on the head node discovers that it can no longer communicate with the daemon on the ordinary node, and deletes it from the PVM. After the ordinary node is back up, the pvm console program can be used to add it to the pvm once again.

Shared File Hierarchy

At RFPK, the pvm software is installed centrally on cspkserver. Each of the other hosts accesses the software via NFS. The shared file hierarchy is mounted on the directory node normally referred to by the environment variable `PVM_ROOT`. Within this hierarchy, items which differ from one supported architecture to another are differentiated by the inclusion of architecture specific nodes in the pathnames. For example, pvm system executables are found in the directory `$PVM_ROOT/lib/$PVM_ARCH`. Substituting values for environment variables in use for Pentium machines at RFPK, this path resolves to `/usr/share/pvm3/lib/LINUXI386`. Similarly, spk executables reside in the library `$PVM_ROOT/bin/$PVM_ARCH`.

Software Installation

All pvm software at RFPK is installed on cspkserver, which acts as the head node of the pvm. At present, two processor architectures are supported. Pvm designates the Intel Pentium architecture as LINUXI386 and the Amd/Intel 64-bit architecture as LINUXX86_64. The software for both of these architectures is installed on cspkserver so that both architectures are available to the other hosts via NFS. If additional architectures were to be added, their software would also be installed on cspkserver.

Because our systems all run RedHat Enterprise Linux, Version 3 (RHEL3), the easiest and most reliable way to install and update the software is to download packages as `.rpm` files and to install them with the RedHat Packet Manager (rpm).

Here is how the software was installed on cspkserver. If, for some reason, the software had to be reinstalled from scratch, this would be the procedure to follow.

1. Download the most recent pvm packages from RedHat. At the time of writing (February 4, 2005), the most recent were `pvm-3.4.4-22.i386.rpm` and `pvm-3.4.4-22.x86_64.rpm`.
2. Install the package for the LINUXX86_64 architecture. As root, in a terminal window do the following:

   ```
   rpm -Uhv pvm-3.4.4-22.x86_64.rpm
   ```

3. Install the package for the LINUXI386 architecture:

   ```
   rpm -Uhv pvm-3.4.4-22.i386.rpm
   ```

NFS Configuration Required for the All Nodes

As root, in a terminal window, do the following:

1. Start NFS, if it is not already running:o

   ```
   /etc/rc.d/init.d/nfs start
   ```

2. Insure that NFS will restart automatically, each time the system boots:

   ```
   /sbin/chkconfig nfs on
   ```

3. Restart the port mapper:

   ```
   /etc/rc.d/init.d/portmap restart
   ```

Additional NFS Configuration for the Head Node

As root in a terminal window, perform the following:

1. Each node must have an entry in the `/etc/hosts` file. For any node that is not already so declared, use your favorite text editor to add a line such as the following, which relates the IP address with the name by which you wish to refer to it:

```
192.168.1.5          rose
```

2. Edit the `/etc/exports` file to list all of the nodes that will need access to the software. For *each* such node, add a line such as this:

```
/usr/share/pvm3        rose(rw,sync,no_root_squash)
```

   Note that `/usr/share/pvm3` is the file hierarchy which will be shared, and *rose* is the host name of a node which will be allowed to access it.

3. Export the file hierarchy:

```
/usr/sbin/exportfs -a
```

Additional NFS Configuration on Ordinary Nodes

This section covers NFS configuration for all nodes other than the head node (cspkserver). All the following should be performed by the root user, from a terminal window.

1. Hide a preexisting installation of pvm on the node, if one exists:

```
mv /usr/share/pvm3 /usr/share/pvm3.hide
```

2. The `/etc/hosts` file must contain an entry for *cspkserver*. If such an entry is not already there, edit `/etc/hosts` to add the line (or similar, if this IP address is incorrect):

```
192.168.1.18     cspk cspkserver
```

3. Create a node on which to mount `PVM_ROOT`:

```
mkdir /usr/share/pvm3
```

4. All nodes other than the head node must perform an NFS mount of PVM_ROOT. As root, edit `/etc/fstab`, adding the following line:

```
cspkserver:/usr/share/pvm3  /usr/share/pvm3  nfs rsize=8194,wsize=8192,timeo=14,ir
```

5. If, on the *head node*, there is no entry for this ordinary node in the `/etc/exports` file, add an entry now ; then export it.

6. Verify that you can mount PVM_ROOT by executing the following command, as root:

   ```
   mount /usr/share/pvm3
   ```

   then use **ls** to verify that the directory contains files.

Configuration Required for Every Node

This section covers configuration steps that must be taken for *all nodes, including the head node*.

Correct `/etc/hosts`

In linux, the host name of a system as well as its aliases can be associated in the `/etc/hosts` file with either `127.0.0.1`, which is the IP address of the *lo* interface, or with the IP address of a network interface, such as *eth0*. For pvm to work correctly, only the latter choice is acceptable. In other words, the host name and its aliases must be associated with the IP address by which the node is addressed by the other nodes.

As root, edit `/etc/hosts` so that the localhost entry looks like this:

```
127.0.0.1       localhost localhost.localdomain
```

and the entry for a node (*anynode*, for example):

```
192.168.1.18    anynode
```

Create a home directory for pvm

As part of the installation of RedHat Enterprise Linux, Version 3 (RHEL3), a user named pvm is created. It is given `PVM_ROOT` as its home directory. This will not work for our purposes, because we share the `PVM_ROOT` file hierarchy via NFS. Certain configuration files are changed by the nodes during their normal operation. If these files were shared, the nodes would interfere with each other. For this reason, it is necessary to create a separate home directory for every node.

As root:

```
mkdir /home/pvm
usermod -d /home/pvm pvm
```

Install keychain

ssh is used for communication between nodes of the pvm. Normally, ssh requires a password every time that a communications channel is established. If a program called keychain is installed on each node, however, the password only needs to be provided for the first communication between two hosts after a system boot by the

destination host. An `.rpm` package can be downloaded from the keychain web site [1] or it can be copied from whitechuck (fileserver):

```
# as an ordinary user
scp 'fileserver:/opt/download/keychain*.rpm' .
```

Once you have the file on the node, as root you can install it with the command

```
# as root
rpm -Uhv keychain*.rpm
```

Configuration for the Head Node

In the following, it is assumed that you will be working as the *root* user, from a terminal window, and that for convenience sake, you have defined `PVM_ROOT`:

```
PVM_ROOT=/usr/share/pvm3
```

Copy skeleton files to `$PVM_ROOT`

We will need some configuration files to enable the pvm user to run the pvm console program. The following commands will copy the necessary files from the skeleton directory to `PVM_ROOT`. They will later be linked into the home directory by means of symbolic links.

```
cp /etc/skel/.bash* $PVM_ROOT
cat $PVM_ROOT/lib/bashrc.stub >> $PVM_ROOT/.bashrc
```

Edit `/home/pvm/.bashrc`

We can now edit `$PVM_ROOT/.bashrc` to define several necessary environment variables. Because PVM_ROOT is NFS mounted on all nodes, these variables will be defined for the pvm user on all nodes.

Using your favorite text editor, add the following lines to the `$PVM_ROOT/.bashrc` just after the section at the beginning which sources global definitions from `/etc/bashrc`:

```
PVM_ROOT=/usr/share/pvm3
PVM_RSH=/usr/bin/ssh
export PVM_ROOT PVM_RSH
```

Of course, if the value of your `PVM_ROOT` is not `/usr/share/pvm3`, let the command that you use reflect that.

With your text editor, uncomment the lines that correspond to the following commands by deleting the lead # sign:

```
export PATH=$PATH:$PVM_ROOT/lib/$PVM_ARCH  # arch-specific
```

```
export PATH=$PATH:$PVM_ROOT/bin/$PVM_ARCH
```

Create a shared `.ssh` directory

```
cd $PVM_ROOT
mkdir .ssh
chmod 775 .ssh
```

Create an ssh key pair

```
ssh-keygen -t dsa -f .ssh/id_dsa
```

Create a shared `authorized_keys` file

cat .ssh/id_dsa.pub >> .ssh/authorized_keys chmod 644 .ssh/authorized_keys

Configure keychain

The *keychain* program makes it possible to initiate multiple ssh sessions without having to enter a pass-code more than once. Between reboots of the machine, the pass-code is entered the first time a user logs in. To set this up, we edit the `.bash_profile` file for the pvm user.

Using your favorite text editor, append these lines to `$PVM_ROOT/.bash_profile`:

```
keychain ~/.ssh/id_dsa
. ~/.keychain/${HOSTNAME}-sh
```

SSH Configuration for Ordinary Nodes

In the following, it is assumed that you will be working as the *root* user, from a terminal window, on the ordinary node.

For convenience sake, define `PVM_ROOT`:

```
PVM_ROOT=/usr/share/pvm3
```

Just like the head node, each ordinary node needs a `.ssh` file.

```
cd /home/pvm
mkdir .ssh
```

```
chmod 755 .ssh
```

Symbolic Links for Every Node

We have placed several configuration files in `$PVM_ROOT` rather than `/home/pvm` so that they can be shared by all nodes. Now we have to place symbolic links in each home directory so that they can be found.

```
cd /home/pvm
ln -s $PVM_ROOT/.bash_logout         .bash_logout
ln -s $PVM_ROOT/.bash_profile        .bash_profile
ln -s $PVM_ROOT/.bashrc              .bashrc
cd .ssh
ln -s $PVM_ROOT/.ssh/id_dsa          id_dsa
ln -s $PVM_ROOT/.ssh/id_dsa.pub      id_dsa.pub
ln -s $PVM_ROOT/.ssh/authorized_keys authorized_keys
chown -R pvm.pvm /home/pvm
```

Adding an Ordinary Node

This section contains a check list for adding an ordinary node.

1. As root, in a terminal window, define `PVM_ROOT`:
   ```
   PVM_ROOT=/usr/share/pvm3
   ```
2. NFS Configuration Required for All Nodes
3. Additional NFS Configuration on Ordinary Nodes .
4. Configuration Required for Every Node.
5. SSH Configuration for Ordinary Nodes.
6. Symbolic Links for Every Node.
7. Test your configuration. Suppose that the name of the new node is *newnode*.
   a. Start as the user *pvm* on cspkserver. Use ssh to log into newnode.
      ```
      ssh pvm@newnode
      ```
      You will need to supply the pvm pass*phrase*.
   b. Now, in the same window, log into cspkserver.
      ```
      ssh cspkserver
      ```
      You should be asked to supply the pvm pass*phrase*.
   c. Next, in the same window (which is now displaying a shell running on cspkserver), log into newnode:
      ```
      ssh newnode
      ```
      If everything is working, this time you will not be asked for either a password or a passphrase.

d. Finally, from newnode and still in the same window, log into cspkserver:

```
ssh cspkserver
```

Again, you should not be asked for either a password or a passphrase.

Notes

1. http://www.gentoo.org/proj/en/keychain.xml