A Quick Start Guide to the Parallel Virtual File System

Philip H. Carns, Robert B. Ross, Walter B. Ligon III

Jan 17, 2001

Contents

1 How to use this document

The quick start guide is intended to be a reference on how to quickly install and configure the Parallel Virtual File System. This is broken into three parts. The first describes how to download and compile the PVFS software. The next section walks through the steps of configuring PVFS to store and access files on a single host. This of course is not a cluster environment, but may be useful for testing, evaluation, or just familiarizing yourself with PVFS. The final section of this document describes how to install and configure PVFS in a true cluster environment. It makes a clearer distinction between which components are needed for clients and which are needed for servers. This section also covers the details of multi-server configuration.

1.1 Versions

This document coincides with the following versions of the PVFS software:

- PVFS 1.5.0
- pvfs-kernel 0.9.0

1.2 Related documents

A more in depth discussion of PVFS usage can be found in the PVFS user guide (http://parlweb.parl.clemson.edu/pvfs/user_guide.html). Various technical papers documenting the design and performance of PVFS can be found at http://parlweb.parl.clemson.edu/pvfs/papers.html. Finally, documentation for the ROMIO MPI-IO package which may be used in conjunction with PVFS is located at http://www.mcs.anl.gov/romio/.

2 Downloading and compiling PVFS

2.1 Obtaining the source

PVFS is freely available under the GNU General Public License. There are two official download locations available on the internet:

- ftp://ftp.parl.clemson.edu:/pub/pvfs/
- ftp://mirror.chpc.utah.edu:/pub/pvfs/ (mirror site)

Visit one of these sites and download the files pvfs-<v1>.tgz and pvfs-kernel-<v2>.tgz, where <v1> and <v2> are version numbers. Please obtain the latest version of each (note that the version numbers of these two packages do not match).

2.2 Untarring the packages

The two PVFS packages may be untarred into the same directory to ease compilation. The following example uses the /usr/src/ directory:

```
[root@testhost /root]# cp pvfs-1.5.0.tgz pvfs-kernel-0.9.0.tgz /usr/src
[root@testhost /root]# cd /usr/src
[root@testhost /usr/src]# tar -xzf pvfs-1.5.0.tgz
```

```
[root@testhost /usr/src]# tar -xzf pvfs-kernel-0.9.0.tgz
[root@testhost /usr/src]# ln -s pvfs-1.5.0 pvfs
[root@testhost /usr/src]# ls -lF
total 476
lrwxrwxrwx   1 root    root         15 Dec 14 17:42 pvfs -> pvfs-1.5.0/
drwxr-xr-x  12 root    root        512 Dec 14 10:11 pvfs-1.5.0/
-rw-r--r--   1 root    root     371535 Dec 14 17:41 pvfs-1.5.0.tgz
drwxr-xr-x   6 root    root       1024 Dec 14 10:10 pvfs-kernel-0.9.0/
-rw-r--r--   1 root    root     105511 Dec 14 17:41 pvfs-kernel-0.9.0.tgz
```

2.3 Building the packages

Once the packages have been untarred, you may proceed to build and install them:

```
[root@testhost /usr/src]# cd pvfs
[root@testhost /usr/src/pvfs-1.5.0]# ./configure
[root@testhost /usr/src/pvfs-1.5.0]# make
[root@testhost /usr/src/pvfs-1.5.0]# make install
```

You should now have the server binaries, client libraries, include files, utilities, and man pages for PVFS installed on your system. The next step builds the kernel module and support programs needed for compatibility with existing system programs. The build process looks for kernel header files in /usr/src/linux by default, so make sure that this directory points to a copy of the kernel version that you wish to compile the module for.

```
[root@testhost /usr/src/pvfs-1.5.0]# cd ../pvfs-kernel-0.9.0
[root@testhost /usr/src/pvfs-kernel-0.9.0]# ./configure --with-libpvfsdir=./pvfs/lib
[root@testhost /usr/src/pvfs-kernel-0.9.0]# make
[root@testhost /usr/src/pvfs-kernel-0.9.0]# make install
```

At this point you need to copy `pvfs.o` into the correct directory for your system. This is ususally /lib/modules/<kernel-version>/misc/. For example:

```
[root@testhost /usr/src/pvfs-kernel-0.9.0]# cp pvfs.o /lib/modules/2.2.16/misc/
```

Now that you have successfully compiled the PVFS and pvfs-kernel software packages, you may now proceed to either the "Installing PVFS on a single host" or the "Installing PVFS on a cluster" sections of this document.

3 Installing PVFS on a single host

This section documents the steps required to configure PVFS on a system in which a single machine acts as both the client and server for all PVFS operations. It assumes that you have completed the above section entitled "Downloading and compiling PVFS". The hostname of the example machine is "testhost" and will be referenced as such in the following examples. We will store the PVFS metadata in the /pvfs-meta directory and the file data in the /pvfs-data directory. /mnt/pvfs will serve as the mount point for the file system. For more details about the purpose of these directories please see the PVFS users guide.

3.1 Server Directories and Configuration Files

There are two directories that need to be created to support PVFS daemons. The first is for PVFS metadata (used for storing directory structure, permissions, etc.) and the second is for PVFS file data. Create the two directories as follows:

```
[root@testhost /usr/src/pvfs-kernel-0.9.0]# cd /
[root@testhost /]# mkdir /pvfs-meta
[root@testhost /]# mkdir /pvfs-data
[root@testhost /]# chmod 700 /pvfs-data
[root@testhost /]# chown nobody.nobody /pvfs-data
```

You can now create the configuration files needed for the mgr daemon by running `mkmgrconf` (these files will reside in the /pvfs-meta directory).

```
[root@testhost /]# cd /pvfs-meta

[root@testhost /pvfs-meta]# /usr/local/bin/mkmgrconf
This script will make the .iodtab and .pvfsdir files
in the metadata directory of a PVFS file system.

Enter the root directory:
/pvfs-meta
Enter the user id of directory:
root
Enter the group id of directory:
root
Enter the mode of the root directory:
777
Enter the hostname that will run the manager:
testhost
Searching for host...success
Enter the port number on the host for manager:
(Port number 3000 is the default)
3000
Enter the I/O nodes:  (can use form node1, node2, ...  or
nodename#-#,#,#)
testhost
Searching for hosts...success
I/O nodes:  testhost
Enter the port number for the iods:
(Port number 7000 is the default)
7000
Done!

[root@testhost /pvfs-meta]# ls -al
total 9
drwxr-xr-x    2 root     root           82 Dec 17 15:01 ./
drwxr-xr-x   21 root     root          403 Dec 17 15:01 ../
```

```
-rwxr-xr-x    1 root     root           84 Dec 17 15:01 .iodtab*
-rwxr-xr-x    1 root     root           43 Dec 17 15:01 .pvfsdir*
```

Finally, you must create a configuration file for the I/O daemon. For now we will assume that you wish to use the defaults, thus it suffices to just copy the example configuration from the source directory:

```
[root@testhost /]# cp /usr/src/pvfs/system/iod.conf /etc/iod.conf
[root@testhost /]# ls -al /etc/iod.conf
-rwxr-xr-x    1 root     root           57 Dec 17 11:22 /etc/iod.conf
```

3.2 Starting the PVFS daemons

The PVFS manager handles access control to files on PVFS file systems. It must be running in order to access PVFS files. Start it up with:

```
[root@testhost /root]# /usr/local/sbin/mgr
```

The I/O daemon handles actual file reads and writes for PVFS. It too needs to be started:

```
[root@testhost /root]# /usr/local/sbin/iod
```

The server side portion of PVFS is now ready for access. If at any point you wish to verify that the iod or mgr is running correctly on your system you may check it by running the iod-ping or mgr-ping utilities, respectively. These test programs establish communications with the daemons in question and return a status report:

```
[root@testhost /root]# /usr/local/bin/iod-ping -h testhost
testhost:7000 is responding.
[root@testhost /root]# /usr/local/bin/mgr-ping -h testhost
testhost:3000 is responding.
```

3.3 Client configuration

There are two primary methods for accessing a PVFS file system. The first is the native PVFS interface which is made available through `libminipvfs`. This also happens to be the same interface used by ROMIO if you configure your system to use MPI-IO. The second method relies on a kernel module to provide standard Linux file system compatability. This interface allows the user to use existing binaries and system utilities on PVFS without recompiling. We will cover how to configure both access methods here.

We must create a mount point for the file system as well as an `/etc/pvfstab` entry that will be used by the pvfs libraries to locate the file system. The `pvfstab` file is analogous to the `/etc/fstab` file that most linux systems use to keep up with file system mount points.

```
[root@testhost /root]# mkdir /mnt/pvfs
[root@testhost /root]# touch /etc/pvfstab
[root@testhost /root]# chmod a+r /etc/pvfstab
```

Now edit this file so that it contains the following:

```
testhost:/pvfs_meta  /mnt/pvfs  pvfs  port=3000  0  0
```

This tells clients to look for a PVFS file system mounted on the /mnt/pvfs directory. The metadata is being served from the local machine's /pvfs_meta directory.
At this point, applications which utilize the PVFS library directly should work fine. However, we still need to setup the PVFS kernel module for standard Linux utilities such as ls to work correctly. This requires four steps: creating a special device file, loading the module, starting the client daemon, and mounting the file system. The device file need only be created once for each machine and does not need to be repeated at each boot.

```
[root@testhost /root]# /bin/mknod /dev/pvfsd c 60 0
[root@testhost /root]# /sbin/insmod pvfs
[root@testhost /root]# /usr/local/sbin/pvfsd
[root@testhost /root]# /sbin/mount.pvfs testhost:/pvfs-meta /mnt/pvfs
```

3.4 Testing your installation

If your installation up to this point has been successful, you should be able to read and write files onto PVFS just as you would any other file system.

```
[root@testhost /root]# cp /etc/iod.conf /mnt/pvfs/
[root@testhost /root]# cat /mnt/pvfs/iod.conf
# Blank IOD config file -- IOD will use builtin defaults
[root@testhost /root]#
```

You can also check the status of any PVFS file using a PVFS utility called pvstat. pvstat queries a PVFS file and returns information regarding the stripe size, base node, and other file parameters which you can find out more about by reading the users guide.

```
[root@testhost /root]# /usr/local/bin/pvstat /mnt/pvfs/iod.conf
/mnt/pvfs/iod.conf: base = 0, pcount = 1, ssize = 65536
```

Congratulations! You now have a fully functional PVFS configuration.

4 Installing PVFS on a cluster

This section documents the steps required to configure PVFS on a beowulf cluster consisting of more than one node (For more information on beowulf clusters, please visit http://www.beowulf-underground.org or http://www.beowulf.org). It assumes that you have completed the preceeding section entitled "Downloading and compiling PVFS", that your cluster is already up and running, and that you have a mechanism for logging into each node and copying files to each node. If you are uncertain about any of these concepts please review the above mentioned websites for documentation before proceeding.

4.1 Machine configuration

It is important to have in mind the roles that machines (a.k.a. nodes) will play in the PVFS system. There are three potential roles that a machine might play:

- metadata server
- I/O server
- client

A metadata server is a node that keeps up with metadata (such as permissions and time stamps) for the file system. An I/O server is a node that actually stores a portion of the PVFS file data. A client is a node that can read and write PVFS files. Your applications will typically be run on PVFS clients so that they can access the file system.

A machine can fill one, two, or all of these roles simultaneously. Each role requires a specific set of binaries and configuration information. There will be one metadata server for the PVFS file system. There can be many I/O servers and clients. In this section we will discuss the components and configuration files needed to fulfill each role.

We will configure our example system so that the
"head" node provides metadata service, eight other nodes (named "node0" through "node7") provide I/O service, and all nodes can act as clients.

4.2 Installation file locations

This portion of the document assumes that you have installed all of the PVFS files from the first section of this document on each node. This is not strictly necessary. If storage space is a concern, you may wish to check the users guide to determine which files are needed for each class of nodes and just copy out the required files for each. For the purposes of this document it will be clearer to assume that each node contains a full PVFS and pvfs-kernel installation, however.

4.3 Configuring the metadata server

On the machine that will act as your metadata server, you must create a directory that will be used to store file system metadata. You can then run a script called `mkmgrconf` that will configure this directory and provide information about the layout of your system. Take special note of the portion of the script that asks you to list each I/O node on the system.

```
[root@head /]# mkdir /pvfs-meta
[root@head /]# cd /pvfs-meta

[root@head /pvfs-meta]# /usr/local/bin/mkmgrconf
This script will make the .iodtab and .pvfsdir files
in the metadata directory of a PVFS file system.

Enter the root directory:
/pvfs-meta
Enter the user id of directory:
root
Enter the group id of directory:
root
Enter the mode of the root directory:
777
```

```
Enter the hostname that will run the manager:
head
Searching for host...success
Enter the port number on the host for manager:
(Port number 3000 is the default)
3000
Enter the I/O nodes:  (can use form node1, node2, ...  or
nodename#-#,#,#)
node0-7
Searching for hosts...success
I/O nodes:  node0 node1 node2 node3 node4 node5 node6 node7
Enter the port number for the iods:
(Port number 7000 is the default)
7000
Done!
```

```
[root@head /pvfs-meta]# ls -al
total 9
drwxr-xr-x    2 root     root           82 Dec 17 15:01 ./
drwxr-xr-x   21 root     root          403 Dec 17 15:01 ../
-rwxr-xr-x    1 root     root           84 Dec 17 15:01 .iodtab*
-rwxr-xr-x    1 root     root           43 Dec 17 15:01 .pvfsdir*
```

The `.iodtab` and `.pvfsdir` files were created by the `mkmgrconf` script and contain information about the configuration of the file system.

4.4 Configuring the I/O servers

On each of the machines that will act as I/O servers, you need to create a directory that will be used to store PVFS file data and set the appropriate permissions on that directory.

```
[root@node0 /]# mkdir /pvfs-data
[root@node0 /]# chmod 700 /pvfs-data
[root@node0 /]# chown nobody.nobody /pvfs-data
```

You must create a configuration file for the I/O daemon on each of your I/O servers. For now we will assume that you wish to use the defaults, thus it suffices to just copy the example configuration from the source directory. Each I/O server in the cluster can use an identical configuration file.

```
[root@node0 /]# cp /usr/src/pvfs/system/iod.conf /etc/iod.conf
[root@node0 /]# ls -al /etc/iod.conf
-rwxr-xr-x    1 root     root           57 Dec 17 11:22 /etc/iod.conf
```

4.5 Starting the PVFS daemons

The PVFS manager handles access control to files on PVFS file systems. It must be running on your metadata server in order to access PVFS files.

```
[root@head /root]# /usr/local/sbin/mgr
```

The I/O daemon handles actual file reads and writes for PVFS. It must be started on every I/O server in your cluster. You may log on to each one and start the daemon as follows:

```
[root@node0 /root]# /usr/local/sbin/iod
```

The server side portion of PVFS is now ready for access. If at any point you wish to verify that one of the iod's or mgr is running correctly on your system you may check it by running the iod-ping or mgr-ping utilities, respectively. These test programs establish communications with the daemons in question and return a status report:

```
[root@head /root]# /usr/local/bin/iod-ping -h node0
node0:7000 is responding.
[root@head /root]# /usr/local/bin/iod-ping -h node1
node1:7000 is responding.
[root@head /root]# /usr/local/bin/mgr-ping -h head
head:3000 is responding.
```

4.6 Client configuration

There are two primary methods for accessing a PVFS file system. The first is the native PVFS interface which is made available through `libminipvfs`. This also happens to be the same interface used by ROMIO if you configure your system to use MPI-IO. The second method relies on a kernel module to provide standard Linux file system compatability. This interface allows the user to use existing binaries and system utilities on PVFS without recompiling. We will cover how to configure both access methods here.

We must create a mount point for the file system as well as an `/etc/pvfstab` entry that will be used by the pvfs libraries to locate the file system. The `pvfstab` file is analogous to the `/etc/fstab` file that most linux systems use to keep up with file system mount points. You should create this file on each of your client nodes (node0 through node7 and head in this case). The contents of this file can be identical for each machine.

```
[root@head /root]# mkdir /mnt/pvfs
[root@head /root]# touch /etc/pvfstab
[root@head /root]# chmod a+r /etc/pvfstab
```

Now edit this file so that it contains the following:

```
head:/pvfs_meta  /mnt/pvfs  pvfs  port=3000  0  0
```

This tells clients to look for a PVFS file system mounted on the `/mnt/pvfs` directory. The metadata is being served from the head machine's `/pvfs_meta` directory.

At this point, applications which utilize the PVFS library directly should work fine. However, we still need to setup the PVFS kernel module for standard Linux utilities such as `ls` to work correctly. This requires four steps: creating a special device file, loading the module, starting the client daemon, and mounting the file system. Again, perform these steps on each of the machines that will be accessing the file system as a client. The device file need only be created once for each machine and does not need to be repeated at each boot.

```
[root@head /root]# /bin/mknod /dev/pvfsd c 60 0
[root@head /root]# /sbin/insmod pvfs
[root@head /root]# /usr/local/sbin/pvfsd
[root@head /root]# /sbin/mount.pvfs head:/pvfs-meta /mnt/pvfs
```

4.7 Testing your installation

If your installation up to this point has been successful, you should be able to read and write files onto PVFS just as you would any other file system by using the /mnt/pvfs mount point on each client node.

```
[root@head /root]# cp /etc/iod.conf /mnt/pvfs/
[root@head /root]# cat /mnt/pvfs/iod.conf
# Blank IOD config file -- IOD will use builtin defaults
[root@head /root]#
```

You can also check the status of any PVFS file using a PVFS utility called pvstat. pvstat queries a PVFS file and returns information regarding the stripe size, base node, and other file parameters which you can find out more about by reading the users guide.

```
[root@head /root]# /usr/local/bin/pvstat /mnt/pvfs/iod.conf
/mnt/pvfs/iod.conf: base = 0, pcount = 1, ssize = 65536
```

Congratulations! You now have a fully functional PVFS configuration. You should now be able to write parallel applications that generate concurrent accesses to the PVFS file system mounted on /mnt/pvfs.