A Quick Start Guide to PVFS2

PVFS2 Development Team

Last Updated: June 2004

Contents

1 How to use this document

The quick start guide is intended to be a reference on how to quickly install and configure a PVFS2 file system. It is broken down into three parts. The first describes how to download and compile the PVFS2 software. The next section walks through the steps of configuring PVFS2 to store and access files on a single host, which may be useful for simple testing and evaluation. The final section of this document describes how to install and configure PVFS2 in a true cluster environment with multiple servers and/or clients.

1.1 Versions

This document only applies to the most recent snapshot of PVFS2.

2 Downloading and compiling PVFS2

At this time, tarballs (and CVS access) are provided to trusted parties. If you're reading this, you should have a method of obtaining the PVFS2 source code. Once the source code is downloaded, compiling the PVFS2 source code is a matter of running './configure', followed by 'make' from the top level source directory. More detailed instruction for building and installing are provided below.

2.1 Dependencies

The following software packages are currently required by PVFS2:

- Berkely DB (version 3 or 4)
- aio support (provided by glibc and librt)
- pthreads
- gcc 2.96 or newer (DO NOT USE gcc 2.95! gcc 3.x recommended)
- GNU Make

The following software packages are currently recommended for use with PVFS2:

- GNU Libc (glibc) 2.3.2 [or later]
- Linux kernel version 2.6.0, or later (NOTE: not necessary for running PVFS2 servers, only the client kernel module)
- A GNU/Linux environment (heterogenous configuration are supported)

ROMIO supports PVFS2. It is not provided with pvfs2, but can be found as part of the following MPI implementations:

- MPICH2-0.96p2 or newer

2.2 Untarring the packages

All source code is contained in one tarball: pvfs2-x.x.x.tar.gz. The following example assumes that you will be building in the /usr/src directory, although that is not required:

```
[root@testhost /root]# cp pvfs2-x.x.x.tar.gz /usr/src
[root@testhost /root]# cd /usr/src
[root@testhost /usr/src]# tar -xzf pvfs2-x.x.x.tar.gz
[root@testhost /usr/src]# ln -s pvfs2-x.x.x pvfs2
[root@testhost /usr/src]# ls -lF
total 476
lrwxrwxrwx   1 root    root         15 Aug 14 17:42 pvfs2 -> pvfs2-x.x.x/
drwxr-xr-x  12 root    root        512 Aug 14 10:11 pvfs2-x.x.x/
-rw-r--r--   1 root    root     371535 Aug 14 17:41 pvfs2-x.x.x.tar.gz
```

2.3 Building and installing the packages

The default steps for building and installing PVFS2 are as follows:

```
[root@testhost /usr/src]# cd pvfs2
[root@testhost /usr/src/pvfs2-XXX]# ./configure
[root@testhost /usr/src/pvfs2-XXX]# make
[root@testhost /usr/src/pvfs2-XXX]# make install
```

Here are some optional configure arguments which may be of interest:

- –prefix=<path>: installs all files in the specified directory (/usr/local/ is the default if –prefix is not specified)
- –with-kernel=<path to 2.6.x kernel source>: this enables compilation of the PVFS2 Linux kernel driver [Requires Linux Kernel 2.6.0 or later]
- –with-kernel24=<path to 2.4.x kernel source>: this enables compilation of the PVFS2 Linux kernel driver [Requires Linux Kernel 2.4.19 or later]
- –with-mpi=<path to mpi installation>: this enables compilation of MPI based test programs
- –with-efence: automatically links in Electric Fence for debugging assistance

Also note that the pvfs2 2.6.x kernel source supports out of tree builds if you prefer to use that technique.

3 Configuring PVFS2 for a single host

This section documents the steps required to configure PVFS2 on a system in which a single machine acts as both the client and server for all PVFS2 operations. It assumes that you have completed the above sections on building and installation already. The hostname of the example machine is “testhost” and will be referenced as such in the following examples. IMPORTANT: if you intend to use the provided rc scripts to handle startup and shutdown of the PVFS2 server, then you must specify a valid hostname as reported by the `hostname` command line tool in the configuration. For this reason, we recommend that you *not* use “localhost” as the hostname of your server, even if you intend to only test one machine. We will store all PVFS2 data in /pvfs2-storage-space. /mnt/pvfs2 will serve as the mount point for the file system. For more details about the purpose of these directories please see the PVFS2 users guide.

3.1 Server configuration

Since this is a single host configuration, we only have to configure one server daemon. In the original PVFS, the metadata and I/O servers were separated into two separate programs (mgr and iod). PVFS2, however, has only a single daemon called pvfs2-server which serves both roles.

The most important part of server configuration is simply generating the configuration files. These can be created using the pvfs2-genconfig script. This is an interactive script which will ask several questions to determine your desired configuration. Please pay particular attention to the listing of the metadata servers and I/O servers. In this example we will use "testhost" for both.

The pvfs2-genconfig tool will generate two configuration files. One is a file system configuration file that will be identical for all servers (if we had more than one). The second is a server specific configuration file that will be different for each server. The server specific files have the hostname of the server that they belong to appended to the file name. This script should be excuted as root, so that we can place the configuration files in their default /etc/ locations.

In this simple configuration, we can accept the default options for every field. We will use the hostname "testhost" rather than "localhost" however.

```
root@testhost:~# /usr/bin/pvfs2-genconfig  \
/etc/pvfs2-fs.conf /etc/pvfs2-server.conf
*********************************************************************
Welcome to the PVFS2 Configuration Generator:

This interactive script will generate configuration files suitable
for use with a new PVFS2 file system.  Please see the PVFS2 quickstart
guide for details.

*********************************************************************

You must first select the network protocol that your file system will use.
The only currently supported options are "tcp", "gm", and "ib".

* Enter protocol type [Default is tcp]:

Choose a TCP/IP port for the servers to listen on.  Note that this
script assumes that all servers will use the same port number.

* Enter port number [Default is 3334]:

Next you must list the hostnames of the machines that will act as
I/O servers.  Acceptable syntax is "node1, node2, ..." or "node{#-#,#,#}".

* Enter hostnames [Default is localhost]: testhost

Now list the hostnames of the machines that will act as Metadata
servers.  This list may or may not overlap with the I/O server list.

* Enter hostnames [Default is localhost]: testhost
```

```
Configured a total of 1 servers:
1 of them are I/O servers.
1 of them are Metadata servers.

* Would you like to verify server list (y/n) [Default is n]?

Choose a file for each server to write log messages to.

* Enter log file location [Default is /tmp/pvfs2-server.log]:

Choose a directory for each server to store data in.

* Enter directory name: [Default is /pvfs2-storage-space]:

Writing fs config file... Done.
Writing 1 server config file(s)... Done.

Configuration complete!
```

3.2 Starting the server

Before you run pvfs2-server for the first time, you must run it with a special argument that tells it to create a new storage space if it does not already exist. In this example, we must run the server as root in order to create a storage space in /pvfs2-storage-space as specified in the configuration files.

```
bash-2.05b# /usr/sbin/pvfs2-server /etc/pvfs2-fs.conf \
/etc/pvfs2-server.conf-testhost -f
```

Once the above step is done, you can start the server in normal mode as follows:

```
bash-2.05b# /usr/sbin/pvfs2-server /etc/pvfs2-fs.conf \
/etc/pvfs2-server.conf-testhost
```

All log messages will be directed to /tmp/pvfs2-server.log, unless you specified a different location while running pvfs2-genconfig. If you would prefer to run pvfs2-server in the foreground and direct all messages to stderr, then you may run the server as follows:

```
bash-2.05b# /usr/sbin/pvfs2-server /etc/pvfs2-fs.conf \
/etc/pvfs2-server.conf-testhost -d
```

3.2.1 Automatic server startup and shutdown

Like most other system services, PVFS2 may be started up automatically at boot up time through the use of rc scripts. We have provided one such script that is suitable for use on RedHat (or similar) rc systems. The following example demonstrates how to set this up:

```
bash-2.05b# cp /usr/src/pvfs2/examples/pvfs2-server.rc \
    /etc/rc.d/init.d/pvfs2-server
bash-2.05b# chmod a+x /etc/rc.d/init.d/pvfs2-server
```

```
bash-2.05b# chkconfig pvfs2-server on
bash-2.05b#  ls -al /etc/rc.d/rc3.d/S35pvfs2-server
lrwxrwxrwx  1 root  root   22 Sep 21 13:11 /etc/rc.d/rc3.d/S35pvfs2-server \
    -> ../init.d/pvfs2-server
```

This script will now automatically launch on startup and shutdown to ensure that the pvfs2-server is started and stopped gracefully. To manually start the server, you can run the following command:

```
bash-2.05b# /etc/rc.d/init.d/pvfs2-server start
Starting PVFS2 server:                                     [  OK  ]
```

To manually stop the server:

```
bash-2.05b# /etc/rc.d/init.d/pvfs2-server stop
Stopping PVFS2 server:                                     [  OK  ]
```

3.3 Client configuration

There are two primary methods for accessing a PVFS2 file system. The first is the native PVFS2 interface which is made available through `libpvfs2`. 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 compatibility. This interface allows the user to use existing binaries and system utilities on PVFS2 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/pvfs2tab` entry that will be used by the PVFS2 libraries to locate the file system. The `pvfs2tab` 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/pvfs2
[root@testhost /root]# touch /etc/pvfs2tab
[root@testhost /root]# chmod a+r /etc/pvfs2tab
```

Now edit this file so that it contains the following, except that you should substitute your host name in place of “testhost”:

```
tcp://testhost:3334/pvfs2-fs /mnt/pvfs2 pvfs2 default 0 0
```

3.4 Testing your installation

PVFS2 currently includes (among others) the following tools for manipulating the file system using the native PVFS2 library: pvfs2-ping, pvfs2-import, pvfs2-ls, and pvfs2-export. These tools check the health of the file system, import local files, list the contents of directories, and export files, respectively. Their usage can best be summarized with the following examples:

```
bash-2.05b# ./pvfs2-ping -m /mnt/pvfs2

(1) Searching for /mnt/pvfs2 in /etc/pvfs2tab...

   Initial server: tcp://testhost:3334
   Storage name: pvfs2-fs
   Local mount point: /mnt/pvfs2
```

```
(2) Initializing system interface and retrieving configuration from server...

   meta servers (duplicates are normal):
   tcp://testhost:3334

   data servers (duplicates are normal):
   tcp://testhost:3334

(3) Verifying that all servers are responding...

   meta servers (duplicates are normal):
   tcp://testhost:3334 Ok

   data servers (duplicates are normal):
   tcp://testhost:3334 Ok

(4) Verifying that fsid 9 is acceptable to all servers...

   Ok; all servers understand fs_id 9

(5) Verifying that root handle is owned by one server...

   Root handle: 0x00100000
   Ok; root handle is owned by exactly one server.

=============================================================

The PVFS2 filesystem at /mnt/pvfs2 appears to be correctly configured.

bash-2.05b# ./pvfs2-ls /mnt/pvfs2/

bash-2.05b# ./pvfs2-import /usr/lib/libc.a /mnt/pvfs2/testfile
PVFS2 Import Statistics:
*******************************************************
Destination path (local): /mnt/pvfs2/testfile
Destination path (PVFS2 file system): /testfile
File system name: pvfs2-fs
Initial config server: tcp://localhost:3334
*******************************************************
Bytes written: 2555802
Elapsed time: 0.416727 seconds
Bandwidth: 5.848920 MB/second
*******************************************************

bash-2.05b# ./pvfs2-ls /mnt/pvfs2/
testfile
```

```
bash-2.05b# ./pvfs2-ls -alh /mnt/pvfs2/
drwxrwxrwx    1 pcarns  users            0 2003-08-14 22:45 .
drwxrwxrwx    1 pcarns  users            0 2003-08-14 22:45 .. (faked)
-rw-------    1 root    root            2M 2003-08-14 22:47 testfile

bash-2.05b# ./pvfs2-export /mnt/pvfs2/testfile /tmp/testfile-out
PVFS2 Import Statistics:
********************************************************
Source path (local): /mnt/pvfs2/testfile
Source path (PVFS2 file system): /testfile
File system name: pvfs2-fs
Initial config server: tcp://localhost:3334
********************************************************
Bytes written: 2555802
Elapsed time: 0.443431 seconds
Bandwidth: 5.496690 MB/second
********************************************************

bash-2.05b# diff /tmp/testfile-out /usr/lib/libc.a
```

4 Installing PVFS2 on a cluster

It is important to have in mind the roles that machines (a.k.a. nodes) will play in the PVFS2 system. There are three potential roles that a machine might play: metadata server, I/O server, or 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 PVFS2 file data. A client is a node that can read and write PVFS2 files. Your applications will typically be run on PVFS2 clients so that they can access the file system.

A machine can fill one, two, or all of these roles simultaneously. Unlike PVFS-1, each role requires just the pvfs2-server binary. It will consult the cluster-wide config file and the node-specific config file when it starts up to know what role pvfs2-server should perform on this machine.

We currently support just one metadata server (this limit will be raised in the future). 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 node “cluster1” provides metadata information, eight nodes (named “cluster1” through “cluster8”) provide I/O services, and all nodes act as clients.

4.1 Server configuration

We will assume that at this point you have either performed a make install on every node, or else have provided the pvfs2 executables, headers, and libraries to each machine by some other means.

Installing PVFS2 on a cluster is quite similar to installing it on a single machine, so familiarize yourself with Section 3. We are going to generate one master config file and 8 smaller node-specific config files. Again, remember that it is critical to list correct hostnames for each machine, and to make sure that these hostnames match the output of the `hostname` command on each machine that will act as a server.

```
root@cluster1:~# /usr/local/pvfs2/bin/pvfs2-genconfig  \
/etc/pvfs2-fs.conf /etc/pvfs2-server.conf
```

```
*****************************************************************
        Welcome to the PVFS2 Configuration Generator:

This interactive script will generate configuration files suitable
for use with a new PVFS2 file system.  Please see the PVFS2 quickstart
guide for details.

*****************************************************************

You must first select the network protocol that your file system will use.
The only currently supported options are "tcp" and "gm".

* Enter protocol type [Default is tcp]:

Choose a TCP/IP port for the servers to listen on.  Note that this
script assumes that all servers will use the same port number.

* Enter port number [Default is 3334]:

Next you must list the hostnames of the machines that will act as
I/O servers.  Acceptable syntax is "node1, node2, ..." or "node{#-#,#,#}".

* Enter hostnames [Default is localhost]: cluster{1-8}

Now list the hostnames of the machines that will act as Metadata
servers.  This list may or may not overlap with the I/O server list.

* Enter hostnames [Default is localhost]: cluster1

Configured a total of 8 servers:
8 of them are I/O servers.
1 of them are Metadata servers.

* Would you like to verify server list (y/n) [Default is n]? y

****** I/O servers:
tcp://cluster1:3334
tcp://cluster2:3334
tcp://cluster3:3334
tcp://cluster4:3334
tcp://cluster5:3334
tcp://cluster6:3334
tcp://cluster7:3334
tcp://cluster8:3334

****** Metadata servers:
tcp://cluster1:3334
```

```
* Does this look ok (y/n) [Default is y]? y

Choose a file for each server to write log messages to.

* Enter log file location [Default is /tmp/pvfs2-server.log]:

Choose a directory for each server to store data in.

* Enter directory name: [Default is /pvfs2-storage-space]:

Writing fs config file... Done.
Writing 8 server config file(s)... Done.

Configuration complete!
```

We have now made all the config files for an 8-node storage cluster:

```
root@cluster1:~# ls /etc/pvfs2/foo/
pvfs2-fs.conf               pvfs2-server.conf-cluster5
pvfs2-server.conf-cluster1  pvfs2-server.conf-cluster6
pvfs2-server.conf-cluster2  pvfs2-server.conf-cluster7
pvfs2-server.conf-cluster3  pvfs2-server.conf-cluster8
pvfs2-server.conf-cluster4
```

Now the config files must be copied out to all of the server nodes. If you use the provided (Redhat style) rc scripts, then you can simply copy all config files to every node; each server will pick the correct config files based on its own hostname at startup time. The following example assumes that you will use scp to copy files to cluster nodes. Other possibilities include rcp, bpcp, or simply storing the configuration files on an NFS volume. Please note, however, that the rc script should be modified if you intend to store config files in any location other than the default /etc/.

At this time, we also will copy out the example rc script an enable it on each machine.

```
root@cluster1:~# for i in ‘seq 1 8‘; do
> scp /etc/pvfs2-server.conf-cluster\${i} cluster\${i}:/etc/
> scp /etc/pvfs2-fs.conf cluster\${i}:/etc/
> scp /usr/src/pvfs2/examples/pvfs2-server.rc \
    cluster\${i}:/etc/rc.d/init.d/pvfs2-server
> ssh cluster\${i} /sbin/chkconfig pvfs2-server on
> done
```

4.2 Starting the servers

As with the single-machine case, you must run pvfs2-server with a special argument to create the storage space on all the nodes if it does not already exist. Run the following command on every metadata or IO node in the cluster:

```
root@cluster1# /usr/sbin/pvfs2-server /etc/pvfs2-fs.conf \
/etc/pvfs2-server.conf -f
```

Then once the storage space is created, start the server for real with a command like this on every metadata or IO node in the cluster:

```
root@cluster1# /usr/sbin/pvfs2-server /etc/pvfs2-fs.conf \
/etc/pvfs2-server.conf
```

If you want to run the server in the foreground (e.g. for debugging), use the -d option.

If you wish to automate server startup and shutdown with rc scripts, refer to the corresponding section 3.2.1 from the single server example.

4.3 Client configuration

Setting up a client for multiple servers is the same as setting up a client for a single server. Refer to section 3.3.

4.4 Testing your Installation

Testing a multiple-server pvfs2 installation is the same as testing a single-server pvfs2 installation. Refer to section 3.4

5 The PVFS2 Linux Kernel Interface

5.1 Finding an Appropriate Kernel Version

Now that you've mastered the download and installation steps of managing the userspace PVFS2 source code, configuring the PVFS2 Linux Kernel Interface is relatively straight forward. We assume at this point that you are familiar with running the server and that a PVFS2 storage space has already been created on the node that you would like to configure for use with the VFS.

A Linux 2.6.0 kernel or later is recommended for the kernel interface, although 2.4.x kernel support has been added for systems that require it. If you're using a 2.4.x kernel, you must be running 2.4.19 or later, as previous versions are NOT (and will not be) supported.

The following examples assume that you've already downloaded, compiled, and are now running the Linux kernel located in the /usr/src/linux-2.x.x directory on your system.

Before compiling the kernel module against your running kernel, check to make sure that you are running an appropriate kernel version. You can do this in the following manner:

```
lain linux # cat /proc/version
Linux version 2.6.6 (root@lain.mcs.anl.gov) (gcc version 3.3.3
20040412 (Gentoo Linux 3.3.3-r5, ssp-3.3-7, pie-8.7.5.3)) #3 SMP Wed
May 26 16:22:11 CDT 2004
```

By issuing that command, we are able to inspect the output to ensure that we're running an appropriate kernel version. If your kernel is older than 2.6.0 (for 2.6.x kernels) or 2.4.19 (for 2.4.x kernels), please download and install a later kernel version (or submit a request to your site's System Administrator).

For reference, you can download Linux kernels at:

```
2.6.x kernels: http://www.kernel.org/pub/linux/kernel/v2.6/
2.4.x kernels: http://www.kernel.org/pub/linux/kernel/v2.4/
```

Once you're convinced the Linux kernel version is appropriate, it's time to compile the PVFS2 kernel module.

5.2 Preparing Linux Kernel 2.6.x configurations

To generate the Makefile, you need to make sure that you run ’./configure’ with the ’–with-kernel=path’ argument. An example is provided here for your convenience:

```
gil:/usr/src/pvfs2# ./configure --with-kernel=/usr/src/linux-2.6.0
```

After this command is issued, build the PVFS2 source tree if it has not yet been built.

Building the 2.6.x kernel module requires an extra step. Since current kernels require writing a few files in the kernel source directory to build a module, you may have to become root to compile the kernel module. To build the module, type “make kmod”.

At this point, we have a valid PVFS2 2.6.x Kernel module. The module itself is the file `pvfs2.ko` in subdirectory `src/kernel/linux-2.6` in your build tree. You may install it to the standard system location with “make kmod_install”, again you will likely have to be root to do this. Or you may override the install location by setting the variable `KMOD_DIR` variable when you install.

5.3 Preparing Linux Kernel 2.4.x configurations

To generate the Makefile, you need to make sure that you run ’./configure’ with the ’–with-kernel24=path’ argument. An example is provided here for your convenience:

```
gil:/usr/src/pvfs2# ./configure --with-kernel24=/usr/src/linux-2.4.26
```

After this command is issued, build the PVFS2 source tree if it has not yet been built.

Building the 2.4.x kernel module requires an extra step. Since current kernels require writing a few files in the kernel source directory to build a module, you may have to become root to compile the kernel module. To build the module, type “make kmod24”.

At this point, we have a valid PVFS2 2.4.x Kernel module. The module itself is the file `pvfs2.o` in subdirectory `src/kernel/linux-2.4` in your build tree. You may install it to the standard system location with “make kmod24_install”, again you will likely have to be root to do this. Or you may override the install location by setting the variable `KMOD_DIR` variable when you install.

5.4 Testing the Kernel Interface

Now that you’ve built a valid PVFS2 kernel module, there are several steps to perform to properly use the file system.

The basic steps are as follows:

- Create a mount point on the local filesystem
- Load the Kernel Module into the running kernel
- Start the PVFS2 Server application
- Start the PVFS2 Client application
- Mount your existing PVFS2 volume on the local filesystem
- Issue VFS commands

First, choose where you’d like to mount your existing PVFS2 volume. Create this directory on the local file system if it does not already exist. Our mount point in this example is /mnt/pvfs2.

```
gil:~# mkdir /mnt/pvfs2
```

Now load the kernel module into your running kernel. You can do this by using the 'insmod' program, or modprobe if you've copied your module into the appropriate /lib/modules directory for your running kernel.

5.4.1 Loading the kernel module

For 2.6.x kernels ONLY:

```
gil:~# insmod /usr/src/pvfs2/src/kernel/linux-2.6/pvfs2.ko
```

For 2.4.x kernels ONLY:

```
gil:~# insmod /usr/src/pvfs2/src/kernel/linux-2.4/pvfs2.o
```

You should verify that the module was loaded properly using the command "lsmod". Also, you can use the "rmmod" to remove the PVFS2 module after it's been loaded. Only remove the module when you have safely unmounted all mounted file systems (if any) and stopped the pvfs2-client software.

At this point, we need to start the PVFS2 server and the PVFS2 client applications before trying to mount a PVFS2 volume. See previous sections on how to properly start the PVFS2 server if you're unsure. Starting the PVFS2 client is covered below.

The PVFS2 client application consists of two programs. "pvfs2-client-core" and "pvfs2-client". DO NOT run "pvfs2-client-core" by itself. "pvfs2-client" is the PVFS2 client application. This application cannot be started unless the PVFS2 server is already running. Here is an example of how to start the PVFS2 client:

```
gil:/usr/src/pvfs2# cd src/apps/kernel/linux-2.6/
gil:/usr/src/pvfs2/src/apps/kernel/linux-2.6# ./pvfs2-client -f -p ./pvfs2-client-c
pvfs2-client starting
Spawning new child process
About to exec ./pvfs2-client-core
Waiting on child with pid 17731
```

The -f argument is not required. For reference, this keeps the PVFS2 client application running in the foreground.

The -p argument is required unless the pvfs2-client-core is installed and can be found in your PATH.

Now that the module is loaded, and the pvfs2-server and pvfs2-client programs are running, we can mount our PVFS2 file system (and verify that it's properly mounted) as follows:

5.4.2 Mounting a PVFS2 volume under 2.6.x

For 2.6.x kernels ONLY:

```
lain pvfs2 # mount -t pvfs2 tcp://testhost:3334/pvfs2-fs /mnt/pvfs2
lain pvfs2 # mount | grep pvfs2
tcp://lain.mcs.anl.gov:3334/pvfs2-fs on /tmp/mnt type pvfs2 (rw)
```

5.4.3 Mounting a PVFS2 volume under 2.4.x

For 2.4.x kernels ONLY:

```
gil:~# mount -t pvfs2 pvfs2 /mnt/pvfs2 -o tcp://testhost:3334/pvfs2-fs
gil:~# mount | grep pvfs2
pvfs2 on /mnt/pvfs2 type pvfs2 (rw)
```

NOTE: The device of the format tcp://testhost:3334/pvfs2-fs MUST be specified, as we need to know a valid running pvfs2-server and file system name to dynamically mount a pvfs2 volume. These values can be read from your configuration files. As a side note, you can use "umount" to unmount the PVFS2 volume when you're ready.

Now that a PVFS2 volume is mounted, normal VFS operation can be issued on the command line. An example is provided below:

```
gil:/usr/src/pvfs2/src/kernel/linux-2.6# mkdir /mnt/pvfs2/newdir
gil:/usr/src/pvfs2/src/kernel/linux-2.6# ls -al /mnt/pvfs2/newdir
total 1
drwxr-xr-x    2 root     root            0 Aug 15 13:29 .
drwxr-xr-x    3 root     root            0 Aug 15 13:21 ..
gil:/usr/src/pvfs2/src/kernel/linux-2.6# cp pvfs2.ko
/mnt/pvfs2/newdir/foo
gil:/usr/src/pvfs2/src/kernel/linux-2.6# ls -al /mnt/pvfs2/newdir
total 2
drwxr-xr-x    2 root     root            0 Aug 15 13:29 .
drwxr-xr-x    3 root     root            0 Aug 15 13:21 ..
-rw-r--r--    1 root     root       330526 Aug 15 13:30 foo
```

A Notes on running PVFS2 without root access

The preceding documentation assumes that you have root access on the machine(s) that you wish to install the file system. However, this is not strictly required for any component except for the kernel VFS support. The servers, client libraries (such as MPI-IO), and administrative tools can all be used by non-priviledged users. This may be particularly useful for evaluation or testing purposes.

In order to do this, you must make the following adjustments to the installation and configuration process:

- Use the –prefix option at configure time to choose an alternate directory (one that you have write access to) for installation. An example would be /home/username/pvfs2-build.
- When generating the server config files, choose a data storage directory that you have write access to, but preferably not NFS mounted. An example would be /tmp/pvfs2-test-space.
- Place the pvfs2tab file in an alternate location, such as /home/username/pvfs2-build/pvfs2tab, instead of /etc/pvfs2tab. Then set the PVFS2TAB_FILE environment variable to the full path to this file. A tcsh example would be: "setenv PVFS2TAB_FILE /home/username/pvfs2-build/pvfs2tab".

B Debugging your PVFS2 configuration

Bug reports and questions should be directed to the PVFS2 users mailing list for best results (see the PVFS2 web site for details: http://www.pvfs.org/pvfs2/lists.html). It is helpful to include a description of your problem, the PVFS2 version number, and include relevant log information from /var/log/messages and /tmp/pvfs2-server.log.

People who wish to find more verbose information about what the file system is doing can enable extra logging messages from the server. This is done by adjusting the "EventLogging" field in the file system configuration file. By default it is set to "none". You can set it to a comma seperated list of log masks to get more information. An example would be "EventLogging storage,network,server", which will result in verbose messages from the storage subsystem, the network subsystem, and server state machines. *WARNING: this may result in extremely large log files!* The logging masks can also be set at runtime using the pvfs2-set-debugmask command line tool. Usage information and a list of supported masks will be shown if it is run with no arguments.

Similarly, run-time client debugging information can be gathered by using environment variables before running the client application. The default client logging method is to set the variable PVFS2_DEBUGMASK to values such as "client,network". Many of the supported client debugging masks overlap the server masks that can be verified using pvfs2-set-debugmask. By default, setting PVFS2_DEBUGMASK emits debugging information to stderr, often intermixed with the client program output. If you'd like to redirect client debugging to a file, set the PVFS2_DEBUGFILE environment variable to a valid file name. This causes all debug information specified by the PVFS2_DEBUGMASK to be stored in the file specified, no longer intermixing the output with the client program.

C ROMIO Support

Building ROMIO with PVFS2 support can be a bit tricky, and is certainly not well documented. While ROMIO has been updated with PVFS2 support, only MPICH2 has included a recent snapshot of ROMIO.

First, get the software. Download MPICH2 from http://www.mcs.anl.gov/mpi/mpich2/. While MPICH2 contains a ROMIO with PVFS2 support, the PVFS2 API has continued to stabilize since MPICH2 was last released. Patches to syncronize the ROMIO in MPICH2 with PVFS2 can be found at http://www.mcs.anl.gov/romio/pvfs2-patches.html. There is also a patch in the PVFS2 source in `doc/coding`. For example's sake, assume all software was downloaded to `${HOME}/src`.

Unpack mpich2, then change to the src/mpi/romio directory. Apply the ROMIO patch. The patch makes changes to the ROMIO configure scripts, so you'll have to re-run `autoconf` to generate a new configure file.

```
prompt% tar xzf ~/src/mpich2-0.96p2.tar.gz     # unpack mpich2 source
prompt% cd mpich2-0.96p2/src/mpi/romio         # change to ROMIO dir
prompt% patch -p0 < ~/src/romio-<CORRECT_VERSION>.diff   #apply patch
prompt% autoconf                               # create a new 'configure'
prompt% cd ../../..                            # return to top of src
prompt%
```

In order to build MPICH2 with a ROMIO that speaks PVFS2, a few changes have to be made to the normal configure process. MPICH2 will need to know the path to the PVFS2 installation. Modify the `CFLAGS`, `LDFLAGS` and `LIBS` environment variables.

```
prompt% export CFLAGS="<other desired flags> -I/usr/local/pvfs2/include"
```

```
prompt% export LDFLAGS="-L/usr/local/pvfs2/lib"
prompt% export LIBS="-lpvfs2 -lpthread"
prompt%
```

The MPICH2 configure script needs a some additional arguments to build ROMIO correctly. The `enable-romio` flag builds ROMIO and the `with-file-system` flag tells ROMIO which file systems to support.

```
configure --enable-romio --with-file-system=ufs+nfs+pvfs2 [other flags]
```

Now compile and install MPICH2 as you normally would. Applications accessing PVFS2 through MPI-IO will bypass the kernel interface and talk to PVFS2 servers directly.