

Frequently Asked Questions about PVFS2

PVFS2 Development Team

November 9, 2004

Contents

1	Basics	3
1.1	What is PVFS2?	3
1.2	What does the “V” in PVFS2 stand for?	3
1.3	Is PVFS2 an attempt to parallelize the *NIX VFS?	3
1.4	What are the components of PVFS2 that I should know about?	3
2	Supported Architectures and Hardware	3
2.1	Does PVFS2 require any particular hardware?	3
2.2	What architectures does PVFS2 support?	3
2.3	Does PVFS2 work across heterogeneous architectures?	4
2.4	Does running PVFS2 require a particular kernel or kernel version?	4
2.5	What specific hardware architectures are supported by the PVFS2 kernel module?	4
2.6	Does the PVFS2 client require a patched Linux kernel?	4
2.7	Can I build the PVFS2 kernel code directly into the kernel, rather than as a module?	4
2.8	Is there a MacOS X/Cygwin/Windows client for PVFS2?	4
3	Installation	4
3.1	How do I install PVFS2?	4
3.2	How can I store PVFS2 data on multiple disks on a single node?	5
3.3	How can I run multiple PVFS2 servers on the same node?	5
3.4	Can I use multiple metadata servers in PVFS2?	5
3.5	Does using multiple metadata servers reduce the chance of file system corruption during hardware failures?	5
3.6	How many servers should I run?	5
3.7	Can PVFS2 servers listen on two network interfaces simultaneously (i.e. multihome)?	6
3.8	How can I automount PVFS2 volumes?	6
3.9	Can I mount more than one PVFS2 file system on the same client?	6
3.10	How can I upgrade from PVFS1 to PVFS2?	6
4	Reporting Problems	6
4.1	Where can I find documentation?	7
4.2	What should I do if I have a problem?	7
4.3	How do I report a problem with PVFS2?	7

5 Problems and Solutions	7
5.1 When I try to mount, I get 'wrong fs type, bad option, bad superblock...'	8
5.2 PVFS2 server consumes 100% of the CPU	8
6 Performance	8
6.1 I ran Bonnie and/or IOzone and the performance is terrible. Why? Is there anything I can do?	8
6.2 Why is program XXX so slow?	9
6.3 NFS outperforms PVFS2 for application XXX. Why?	9
6.4 Can the underlying local file system affect PVFS2 performance?	9
7 Fault Tolerance	10
7.1 Does PVFS2 support some form of fault tolerance?	10
7.2 Can PVFS2 tolerate client failures?	10
7.3 Can PVFS2 tolerate disk failures?	10
7.4 Can PVFS2 tolerate network failures?	10
7.5 Can PVFS2 tolerate server failures?	10
8 File System Interfaces	10
8.1 How do I get MPI-IO for PVFS2?	11
8.2 Can I directly manipulate PVFS2 files on the PVFS2 servers without going through some client interface?	11
9 Management	11
9.1 How can I back up my PVFS2 file system?	11
9.2 Can I add, remove, or change the order of the PVFS2 servers on an existing PVFS2 file system?	11
9.3 Are there tools for migrating data between servers?	12
9.4 Why does df show less free space than I think it should? What can I do about that?	12
9.5 Does PVFS2 have a maximum file system size? If so, what is it?	12
10 Missing Features	12
10.1 Why don't hardlinks work under PVFS2?	12
10.2 Can I mmap a PVFS2 file?	13
10.3 Will PVFS2 store new files on servers with more space, allowing files to be stored when one server runs out of space?	13
10.4 Does PVFS2 have locks?	13
11 Helping Out	13
11.1 How can I contribute to the PVFS2 project?	13
12 Implementation Details	13
12.1 BMI	13
12.1.1 What is the maximum packet size for BMI?	14
12.1.2 What happens if I try to match a BMI send with a BMI receive that has too small a buffer?	14

1 Basics

This section covers some basic questions for people who are unfamiliar with PVFS2.

1.1 What is PVFS2?

PVFS2 is an open-source, scalable parallel file system targeted at production parallel computation environments. It is designed specifically to scale to very large numbers of clients and servers. The architecture is very modular, allowing for easy inclusion of new hardware support and new algorithms. This makes PVFS2 a perfect research testbed as well.

1.2 What does the “V” in PVFS2 stand for?

The “V” in PVFS stands for virtual. This is a holdover from the original (PVFS1) project that built a parallel file system on top of local file systems, which we still do now. It isn’t meant to imply virtualization of storage, although that is sort of what the file system does.

1.3 Is PVFS2 an attempt to parallelize the *NIX VFS?

No, and we’re not even sure what that means! The design of PVFS2 does not depend on the design of the traditional *NIX Virtual Filesystem Switch (VFS) layer, although we provide a compatibility layer that allows access to the file system through it.

1.4 What are the components of PVFS2 that I should know about?

The PVFS2 Guide (<http://www.pvfs.org/pvfs2/pvfs2-guide.html>) has more information on all of these components, plus a discussion of the system as a whole, the code tree, and more.

2 Supported Architectures and Hardware

This section covers questions related to particular system architectures, operating systems, and other hardware.

2.1 Does PVFS2 require any particular hardware?

Other than hardware supported by the Linux OS, no. PVFS2 uses existing network infrastructure for communication and can currently operate over TCP, Myrinet, and InfiniBand. Disk local to servers is used for PVFS2 storage, so no storage area network (SAN) is required either (although it can be helpful when setting up fault tolerant solutions; see Section 7).

2.2 What architectures does PVFS2 support?

The majority of PVFS2 is POSIX-compliant C code that runs in user space. As such, much of PVFS2 can run on most available systems. See Question 2.5 for more information on particular hardware.

The (optional) part of PVFS2 that hooks to the operating system on clients must be written specifically for the particular operating system. Question 2.4 covers this issue.

2.3 Does PVFS2 work across heterogeneous architectures?

Yes! The “language” that PVFS2 uses to talk between clients and servers is encoded in a architecture-independent format (little-endian with fixed byte length parameters). This allows different PVFS2 components to interact seamlessly regardless of architecture.

2.4 Does running PVFS2 require a particular kernel or kernel version?

You can run the userspace PVFS2 servers and administration tools on every major GNU/Linux distribution out of the box, and we intend to keep it that way. However, the kernel module that allows client access to the PVFS2 system does depend on particular kernel versions because it builds against the running one (in the same manner as every other Linux module). The kernel dependent PVFS2 client support has been written for Linux kernel versions 2.4.19 (and greater) and 2.6.0 (and greater). At this time only Linux clients have this level of support.

2.5 What specific hardware architectures are supported by the PVFS2 kernel module?

To our knowledge, PVFS2 has been verified to be working on x86/IA-32, IA-64, AMD64, PowerPC (ppc), and Alpha based GNU/Linux distributions.

2.6 Does the PVFS2 client require a patched Linux kernel?

No. The kernel module source included with PVFS2 is generally targetted toward the official “Linus” kernels (found at kernel.org). Patches for the PVFS2 kernel module code may be provided for major distributions that have modified their kernel to be incompatible with the officially released kernels. The best place to find out more information about support for a kernel tied to a particular distribution is on the PVFS2-developers mailing list.

2.7 Can I build the PVFS2 kernel code directly into the kernel, rather than as a module?

No, this is currently not supported nor recommended.

2.8 Is there a MacOS X/Cygwin/Windows client for PVFS2?

At this time we have no plans for porting the code to operating systems other than Linux. However, we do encourage porting efforts of PVFS2 to other operating systems, and will likely aid in the development.

3 Installation

This section covers issues related to installing and configuring PVFS2.

3.1 How do I install PVFS2?

The PVFS2 Quick Start Guide (<http://www.pvfs.org/pvfs2/pvfs2-quickstart.html>) provides an overview of both a simple, single-server installation, and a more complicated, multi-server configuration.

3.2 How can I store PVFS2 data on multiple disks on a single node?

There are at least two ways to do this.

In general the best solution to this problem is going to be to get the disks logically organized into a single unit by some other OS component, then build a file system on that single logical unit for use by the PVFS2 server on that node.

There are a wide array of hardware RAID controllers that are capable of performing this task. The Multiple Devices (MD) driver is a software component of Linux that can be used to combine multiple disk drives into a single logical unit, complete with RAID for fault tolerance. Using the Logical Volume Management (LVM) component of the Linux OS is another option for this (see the HOWTO at <http://www.tldp.org/HOWTO/LVM-HOWTO.html>). LVM would also allow you to add or remove drives at a later time, which can be quite convenient. You can of course combine the MD and LVM components in interesting ways as well, but that's outside the scope of this FAQ. There's an EVMS program that can be used for managing local storage; this might be useful for setting up complicated configurations of local storage prior to starting up PVFS2 servers.

A second solution would be to use more than one server on the same node, each using a different file system to store its data. This might lead to resource contention issues, so we suggest trying other options first.

3.3 How can I run multiple PVFS2 servers on the same node?

If you do decide to run more than one PVFS2 server on the same node, setting things up is as simple as setting up servers on different nodes. Each will need its own entry in the list of Aliases and its own server-specific configuration file, as described in the Quick Start (<http://www.pvfs.org/pvfs2/pvfs2-quickstart.html>).

3.4 Can I use multiple metadata servers in PVFS2?

Absolutely! Any PVFS2 server can store either metadata, data, or both. Simply allocate unique MetaHandleRanges for each server that you would like to store metadata; the clients will handle the rest.

3.5 Does using multiple metadata servers reduce the chance of file system corruption during hardware failures?

Unfortunately, no. While using multiple metadata servers distributes metadata, it does not replicate or store redundant information across these servers. For information on better handling failures, see Section 7.

3.6 How many servers should I run?

Really, the answer is "it depends", but here are some factors you should take into account.

Running multiple metadata servers might help if you expect to have a lot of small files. The metadata servers are not involved in data access (file contents) but do have a role in file creation and lookup. Multiple clients accessing different files will likely access different metadata servers, so you could see a load balancing effect.

A good rule of thumb is you should run as many data servers as possible. One common configuration is to have some nodes with very high-performance disks acting as servers to the larger cluster. As you use more servers in this configuration, the theoretical peak performance of PVFS2

increases. The clients, however, have to make very large requests in order to stripe the I/O across all the servers. If your clients will never write large files, use a smaller number of servers. If your clients are writing out gigantic checkpoint files or reading in huge datasets, then use more servers.

It is entirely possible to run PVFS2 servers on the same nodes doing computation. In most cases, however, you will see better performance if you have some portion of your cluster dedicated to IO and another portion dedicated to computation.

3.7 Can PVFS2 servers listen on two network interfaces simultaneously (i.e. multi-home)?

PVFS2 servers currently listen on one interface at a time. Multihome support, where a PVFS2 server can accept connections from several network interfaces, would be helpful in many situations. We plan to implement this feature in the future.

3.8 How can I automount PVFS2 volumes?

The Linux automounter needs some help dealing with PVFS2's resource strings. A typical mount command (on Linux 2.6) would look like this:

```
mount -t pvfs2 tcp://server0:3334/pvfs2-fs /mnt/pvfs2
```

The entry in the automount config file should look like this:

```
pvfs -fstype=pvfs2 :tcp://server0\:3334/pvfs2-fs
```

Note the backslash-escape of the colon before the port number. Without that escape, the automounter will get confused and replace 'tcp:/' with 'tcp:/'

3.9 Can I mount more than one PVFS2 file system on the same client?

Yes. However, when setting up the two file systems it is important that both file systems have unique Name and ID values (in the file system configuration file). This means that you can't simply make a copy of the `fs.conf` generated by `pvfs2-genconfig`; you will need to edit the files a bit. This editing needs to be performed *before* you create the storage spaces!

3.10 How can I upgrade from PVFS1 to PVFS2?

Hans Reiser summarized the upgrade approach from reiserfs V3 to V4 with the following:

To upgrade from reiserfs V3 to V4, use tar, or sponsor us to write a `convertfs`.

Similarly, there are no tools currently provided by the PVFS2 team to upgrade from PVFS1 to PVFS2, so tar is your best bet.

4 Reporting Problems

This section outlines some steps that will help the developers figure out what has happened when you have a problem.

4.1 Where can I find documentation?

The best place to look for documentation on PVFS2 is the PVFS2 web site at <http://www.pvfs.org/pvfs2>. Documentation (including this FAQ) is also available in the doc subdirectory of the PVFS2 source distribution.

4.2 What should I do if I have a problem?

The first thing to do is to check out the existing documentation and see if it addresses your problem. We are constantly updating documentation to clarify sections that users have found confusing and to add to this document answers to questions that we have seen.

The next thing to do is to check out the PVFS2 mailing list archives at <http://www.pvfs.org/pvfs2/lists.html>. It is likely that you are not the first person to see a particular problem, so searching this list will often result in an immediate answer.

If you still haven't found an answer, the next thing to do is to mail the mailing list and report your problem.

If you enjoy using IRC, you can also join us on irc.freenode.net in the #pvfs2 channel.

4.3 How do I report a problem with PVFS2?

First you will need to join the PVFS2 Users Mailing list at <http://www.beowulf-underground.org/mailman/listinfo/pvfs2-users>. You must be a user to post to the list; this is necessary to keep down the amount of spam on the list.

Next you should gather up some information regarding your system:

- Version of PVFS2
- Version of MPI and MPI-IO (if you're using them)
- Version of Linux kernel (if you're using the VFS interface)
- Hardware architecture, including CPU, network, storage
- Any logs that might be useful to the developers

Including this information in your first message will help the developers most quickly help you. You are almost guaranteed that if you do not include this information in your first message, you will be asked to provide it in the first reply, slowing down the process.

You should be aware that you are also likely to be asked to try the newest stable version if you are not running that version. We understand that this is not always possible, but if it is, please do.

Note: Please do not send your message to both the PVFS2 Users List and the PVFS2 Developers List; the lists serve different purposes. Also, please do not send your message directly to particular developers. By keeping discussion of problems on the mailing lists we ensure that the discussion is archived and that everyone has a chance to respond.

5 Problems and Solutions

This section covers error conditions you might encounter, what they might mean, and how to fix them.

5.1 When I try to mount, I get 'wrong fs type, bad option, bad superblock...'

First, make 100% sure you typed the mount command correctly. As discussed in the PVFS2 quick-start, different mount commands are needed for linux-2.4 and linux-2.6. A linux-2.6 mount command will look like this:

```
prompt# mount -t pvfs2 tcp://testhost:3334/pvfs2-fs /mnt/pvfs2
```

Under linux-2.4, the mount command looks slightly different:

```
prompt# mount -t pvfs2 pvfs2 /mnt/pvfs2 -o tcp://testhost:3334/pvfs2-fs
```

This error could also mean a pvfs2-client process is not running, either because it was not started before the mount command, or was terminated at some point. If you can reliably (or even intermittently) cause the pvfs2-client to exit abnormally, please send a report to the developers.

This error can also occur if you attempt to mount a second PVFS2 file system on a client, where the new file system has the same name or ID as one that is already mounted. If you are trying to mount more than one file system on the same client and have problems, please see question 3.9.

Finally, be sure there are no typos in your command line, as this is commonly the case!

5.2 PVFS2 server consumes 100% of the CPU

On some systems, the pvfs2-server will start consuming 100% of the CPU after you try to read or write a file to PVFS2. Please check to see if your distribution has an updated glibc package. RHEL3, for example, will exhibit this behavior with glibc-2.3.2-95.6, but not with the updated glibc-2.3.2-95.20 package.

6 Performance

This section covers issues related to the performance of PVFS2.

6.1 I ran Bonnie and/or IOzone and the performance is terrible. Why? Is there anything I can do?

We designed PVFS2 to work well for scientific applications in a cluster environment. In such an environment, a file system must either spend time ensuring all client-side caches are in sync, or not use a cache at all (which is how PVFS2 currently operates). The `bonnie` and `bonnie++` benchmarks read and write very small blocks – on the order of 1K. These many small requests must travel from the client to the server and back again. Without client-side caching, there is no sane way to speed this up.

To improve benchmark performance, specify a bigger block size. PVFS2 has several more aggressive optimizations that can be turned on, but those optimizations require that applications accessing PVFS2 can cope with out-of-sync caches.

In the future, PVFS2 is looking to provide optional semantics for use through the VFS that will allow some client-side caching to speed these kinds of serial benchmarks up. By offering a way to explicitly sync data at any given time or by providing 'close-to-open' semantics, these kinds of caching improvements become an option for some applications.

Bear in mind that benchmarks such as IOzone and Bonnie were meant to stress local file systems. They do not accurately reflect the types of workloads for which we designed PVFS2. Furthermore, because of their serial nature, PVFS2 will be unable to deliver its full performance. Instead try running a parallel file system benchmark like IOR (<http://www.llnl.gov/asci/purple/benchmarks/limited/ior/>).

6.2 Why is program XXX so slow?

See Question 6.1. If the program uses small block sizes to access a PVFS2 file, performance will suffer.

Setting both (or either of) the `TroveSyncMeta` and `TroveSyncData` options to `no` in the config file can improve performance in some situations. By default these values are both set to `yes`, which means that server will issue a sync call after each metadata or data update, respectively. If you set the value to `no` and the server is terminated unexpectedly, you will likely lose data (or access to it). Also, PVFS2 has a transparent server side attribute cache (enabled by default), which can speed up applications which read a lot of attributes (1s, for example). Playing around with the `AttrCache*` config file settings may yield some performance improvements. If you're running a serial application on a single node, you can also use the client side attribute cache (disabled by default). This timeout is adjustable as a command line argument to `pvfs2-client`.

6.3 NFS outperforms PVFS2 for application XXX. Why?

In an environment where there is one client accessing a file on one server, NFS will outperform PVFS2 in many benchmarks. NFS has completely different consistency semantics, which work very well when just one process accesses a file. There is some ongoing work that will optionally offer similar consistency semantics for PVFS2, at which point we will be playing on a level field, so to speak. However, if you insist on benchmarking PVFS2 and NFS in a single-client test, there are some immediate adjustments you can make.

The easiest way to improve PVFS2 performance is to increase the block size of each access. Large block sizes help most file systems, but for PVFS2 they make a much larger difference in performance than they do for other file systems.

Also, if the `TroveSyncMeta` and `TroveSyncData` options are set to `no` in your PVFS2 configuration file, the server will sync data to disk only when a flush or close operation is called. This option is set to `yes` by default, to limit the amount of data that could be lost if a server is terminated unexpectedly. With this option enabled, it is somewhat analogous to mounting your NFS volume with the `sync` flag, forcing it to sync data after each operation.

As a final note on the issue, if you plan on running application XXX, or a similar workload, and the NFS consistency semantics are adequate for what you're doing, then perhaps PVFS2 is not a wise choice of file system for you. PVFS2 is not designed for serial workloads, particularly one with small accesses.

6.4 Can the underlying local file system affect PVFS2 performance?

Yes! However, the interaction between the PVFS2 servers and the local file system hosting the storage space has not been fully explored. No doubt a great deal of time could be spent on different file systems and their parameters.

People have looked at sync performance for a variety of file systems. Some file systems will flush all dirty buffers when `fsync` is called. Other file systems will only flush dirty buffers belonging to the

file. See the threads starting at <http://www.parl.clemson.edu/pipermail/pvfs2-developers/2004-July/000740.html> and at <http://www.parl.clemson.edu/pipermail/pvfs2-developers/2004-July/000741.html>.

These tests demonstrate wide variance in file system behavior. Interested users are encouraged to experiment and discuss their findings on the PVFS2 lists.

If you're looking for a quick suggestion for a local file system type to use, we suggest ext3 with "journal data writeback" option as a reasonable choice.

7 Fault Tolerance

This section covers issues related to fault tolerance in the context of PVFS2.

7.1 Does PVFS2 support some form of fault tolerance?

Systems can be set up to handle many types of failures for PVFS2. Given enough hardware, PVFS2 can even handle server failure.

7.2 Can PVFS2 tolerate client failures?

Yes. One of the benefits of the PVFS2 design is that client failures are not a significant event in the system. Because there is no locking system in PVFS2, and no shared state stored on clients in general, a client failure does not affect either the servers or other clients.

7.3 Can PVFS2 tolerate disk failures?

Yes, if configured to do so. Multiple disks on each server may be used to form redundant storage for that server, allowing servers to continue operating in the event of a disk failure. See section 3.2 for more information on this approach.

7.4 Can PVFS2 tolerate network failures?

Yes, if your network has redundant links. Because PVFS2 uses standard networks, the same approaches for providing multiple network connections to a server may be used with PVFS2. *Need a reference of some sort.*

7.5 Can PVFS2 tolerate server failures?

Yes. We currently have a recipe describing the hardware and software needed to set up PVFS2 in a high availability cluster. Our method is outlined in the 'pvfs2-ha.{ps,pdf}' file in the doc subdirectory of the PVFS2 distribution. This configuration relies on shared storage and commodity "heartbeat" software to provide means for failover.

Software redundancy offers a less expensive solution to redundancy, but usually at a non-trivial cost to performance. We are studying how to implement software redundancy with lower overhead, but at this time we provide no software-only server failover solution.

8 File System Interfaces

This section covers issues related to accessing PVFS2 file systems.

8.1 How do I get MPI-IO for PVFS2?

The ROMIO MPI-IO implementation, as provided with MPICH2 and others, supports PVFS2. You can find more information in the ROMIO section of the pvfs2-quickstart: <http://www.pvfs.org/pvfs2/pvfs2-quickstart.html#sec:romio>

8.2 Can I directly manipulate PVFS2 files on the PVFS2 servers without going through some client interface?

You can, yes, but you probably should not. The PVFS2 developers are not likely to help you out if you do this and something gets messed up...

9 Management

This section covers questions about managing PVFS2 file systems.

9.1 How can I back up my PVFS2 file system?

The default storage implementation for PVFS2 (called Trove DBPF for “DB Plus Files”) stores all file system data held by a single server in a single subdirectory. In that subdirectory is a directory tree containing UNIX files with file data and metadata. This entire directory tree can be backed up in any manner you like and restored if problems occur.

As a side note, this was not possible in PVFS1, and is one of the many improvements present in the new system.

9.2 Can I add, remove, or change the order of the PVFS2 servers on an existing PVFS2 file system?

You can add and change the order of PVFS2 servers for an existing PVFS2 file system. At this time, you must stop all the servers in order to do so.

To add a new server:

1. Unmount all clients
2. Stop all servers
3. Edit your config file to:
 - (a) Add a new Alias for the new server
 - (b) Add a new DataHandleRange for the new server (picking a range you didn’t previously use)
4. Deploy the new config file to all the servers, including the new one
5. Create the storage space on the new server
6. Start all servers
7. Remount clients

To reorder the servers (causing round-robin to occur in a different relative order):

1. Unmount all clients
2. Stop all servers
3. Edit your config file to reorder the DataHandleRange entries
4. Deploy the new config file to all the servers
5. Start all servers
6. Remount clients

Note that adding a new server will *not* cause existing datafiles to be placed on the new server, although new ones will be (by default). Migration tools are necessary to move existing datafiles (see Question 9.3) both in the case of a new server, or if you wanted to migrate data off a server before removing it.

9.3 Are there tools for migrating data between servers?

Not at this time, no.

9.4 Why does df show less free space than I think it should? What can I do about that?

PVFS2 uses a particular algorithm for calculating the free space on a file system that takes the minimum amount of space free on a single server and multiplies this value by the number of servers storing file data. This algorithm was chosen because it provides a lower-bound on the amount of data that could be stored on the system at that point in time.

If this value seems low, it is likely that one of your servers has less space than the others (either physical space, or because someone has put some other data on the same local file system on which PVFS2 data is stored). The `pvfs2-statfs` utility, included with PVFS2, can be used to check the amount of free space on each server, as can the `karma` GUI.

9.5 Does PVFS2 have a maximum file system size? If so, what is it?

PVFS2 uses a 64-bit value for describing the offsets into files, so theoretically file sizes are virtually unlimited. However, in practice other system constraints place upper bounds on the size of files and file systems.

To best calculate maximum file and file system sizes, you should determine the maximum file and file system sizes for the local file system type that you are using for PVFS2 server storage and multiply these values by the number of servers you are using.

10 Missing Features

This section discusses features that are not present in PVFS2 that are present in some other file systems.

10.1 Why don't hardlinks work under PVFS2?

We didn't implement hardlinks, and there is no plan to do so. Symlinks are implemented.

10.2 Can I mmap a PVFS2 file?

Private, read-only mmaping of files is supported. Shared mmaping of files is not. Supporting this would force a great deal of additional infrastructure into PVFS2 that would compromise the design goals of simplicity and robustness. This “feature” was intentionally left out, and it will remain so.

10.3 Will PVFS2 store new files on servers with more space, allowing files to be stored when one server runs out of space?

No. Currently PVFS2 does not intelligently place new files based on free space. It’s a good idea, and possible, but we have not done this yet. See Section 11.1 for notes on how you could help get this feature in place.

10.4 Does PVFS2 have locks?

No. Locking subsystems add a great deal of shared state to a parallel file system implementation, and one of the primary design goals was to eliminate shared state in PVFS2. This results in a simpler, more fault tolerant overall system than would have been possible had we integrated locking into the file system.

It’s possible that an add-on locking subsystem will be developed at some point; however, there is no plan to build such a system at this time.

11 Helping Out

This section covers ways one could contribute to the PVFS2 project.

11.1 How can I contribute to the PVFS2 project?

There are lots of ways to directly or indirectly contribute to the PVFS2 project. Reporting bugs helps us make the system better, and describing your use of the PVFS2 system helps us better understand where and how PVFS2 is being deployed.

Even better, patches that fix bugs, add features, or support new hardware are very welcome! The PVFS community has historically been a friendly one, and we encourage users to discuss issues and exchange ideas on the mailing lists.

If you’re interested in this type of exchange, we suggest joining the PVFS2 Developers List, grabbing the newest CVS version of the code, and seeing what is new in PVFS2. See <http://www.pvfs.org/pvfs2/development> for more details.

12 Implementation Details

This section answers questions regarding specific components of the implementation. It is most useful for people interested in augmenting or modifying PVFS2.

12.1 BMI

This section specifically covers questions about the BMI interface and implementations.

12.1.1 What is the maximum packet size for BMI?

Each BMI module is allowed to define its own maximum message size. See `BMI_tcp_get_info`, `BMI_gm_get_info`, and `BMI_ib_get_info` for examples of the maximum sizes that each of the existing modules support. The maximum should be reported when you issue a `get_info` call with the option set to `BMI_CHECK_MAXSIZE`. Higher level components of PVFS2 perform these checks in order to make sure that they don't choose buffer sizes that are too large for the underlying network.

12.1.2 What happens if I try to match a BMI send with a BMI receive that has too small a buffer?

If the receive buffer is too small for the incoming message, then the communication will fail and an error will be reported if possible. We don't support any semantics for receiving partial messages or anything like that. Its ok if the receive buffer is too big, though.