Frequently Asked Questions about PVFS

PVFS Development Team

December 30, 2025

Contents

1 Basics **3**
- 1.1 What is PVFS? . . . 3
- 1.2 What is the history of PVFS? . . . 3
- 1.3 What is OrangeFS? . . . 4
- 1.4 What is Omnibond? . . . 5
- 1.5 What does the "V" in PVFS stand for? . . . 5
- 1.6 Is PVFS an attempt to parallelize the *NIX VFS? . . . 5
- 1.7 What are the components of PVFS that I should know about? . . . 5
- 1.8 What is the format of the PVFS version string? . . . 5

2 Supported Architectures and Hardware **5**
- 2.1 Does PVFS require any particular hardware? . . . 6
- 2.2 What architectures does PVFS support? . . . 6
- 2.3 Does PVFS work across heterogeneous architectures? . . . 6
- 2.4 Does running PVFS require a particular kernel or kernel version? . . . 6
- 2.5 What specific hardware architectures are supported by the PVFS kernel module? . . . 6
- 2.6 Does the PVFS client require a patched Linux kernel? . . . 6
- 2.7 Can I build the PVFS kernel code directly into the kernel, rather than as a module? . 6
- 2.8 Is there a MacOS X/Cygwin/Windows client for PVFS? . . . 7

3 Installation **7**
- 3.1 How do I install PVFS? . . . 7
- 3.2 How can I store PVFS data on multiple disks on a single node? . . . 7
- 3.3 How can I run multiple PVFS servers on the same node? . . . 7
- 3.4 Can I use multiple metadata servers in PVFS? . . . 7
- 3.5 Does using multiple metadata servers reduce the chance of file system corruption during hardware failures? . . . 8
- 3.6 How many servers should I run? . . . 8
- 3.7 Can PVFS servers listen on two network interfaces simultaneously (i.e. multihome)? 8
- 3.8 How can I automount PVFS volumes? . . . 8
- 3.9 Can I mount more than one PVFS file system on the same client? . . . 8
- 3.10 How can I upgrade from PVFS v1 to PVFS v2? . . . 9

4 Reporting Problems **9**
4.1 Where can I find documentation? . . . 9
4.2 What should I do if I have a problem? . . . 9
4.3 How do I report a problem with PVFS? . . . 9

5 Problems and Solutions **10**
5.1 When I try to mount, I get 'wrong fs type, bad option, bad superblock...' . . . 10
5.2 PVFS server consumes 100% of the CPU . . . 10
5.3 PVFS write performance slows down dramatically . . . 11
5.4 I get "error while loading shared libraries" when starting PVFS programs . . . 11
5.5 PVFS performance gets really bad once a day, then gets better again . . . 11
5.6 Make kmod24 fails with "structure has no member..." errors . . . 11
5.7 When i try to mount a pvfs2 file system, something goes wrong. . . . 12
5.8 I did all three of the above steps and I still can't mount pvfs2 . . . 12
5.9 I'm running Redhat and the pvfs2-server can't be killed! What's wrong? . . . 12
5.10 Why do you single out Redhat users? What's so different about Redhat than other distributions? . . . 13
5.11 Where is the kernel source on a Fedora system? . . . 13
5.12 What are extended attributes? How do I use them with PVFS? . . . 13
5.13 What are Access Control Lists? How do I enable Access Control Lists on PVFS? . . . 14
5.14 On SLES 9, 'make kmod' complains about `mmgrab` and `flush_icache_range` being undefined . . . 14
5.15 Everything built fine, but when I try to compile programs that use PVFS, I get undefined references . . . 14
5.16 Can we run the Apache webserver to serve files off a PVFS volume? . . . 15
5.17 Trove-dbpf metadata format version mismatch! . . . 15
5.18 Problems with pre-release kernels . . . 15
5.19 Does PVFS work with Open-MX? . . . 15

6 Performance **15**
6.1 I configured PVFS with support for multiple intercdonnects (e.g. Infiniband and TCP), but see low performance . . . 16
6.2 I ran Bonnie and/or IOzone and the performance is terrible. Why? Is there anything I can do? . . . 16
6.3 Why is program XXX so slow? . . . 16
6.4 NFS outperforms PVFS for application XXX. Why? . . . 16
6.5 Can the underlying local file system affect PVFS performance? . . . 17
6.6 Is there any way to tune particular directories for different workloads? . . . 17
6.6.1 Distribution . . . 17
6.6.2 Distribution parameters . . . 18
6.6.3 Number of datafiles . . . 18
6.7 My app still runs more slowly than I would like. What can I do? . . . 18

7 Fault Tolerance **18**
7.1 Does PVFS support some form of fault tolerance? . . . 18
7.2 Can PVFS tolerate client failures? . . . 19
7.3 Can PVFS tolerate disk failures? . . . 19
7.4 Can PVFS tolerate network failures? . . . 19

7.5 Can PVFS tolerate server failures? . . . 19

8 File System Interfaces **19**
8.1 How do I get MPI-IO for PVFS? . . . 19
8.2 Can I directly manipulate PVFS files on the PVFS servers without going through some client interface? . . . 19

9 Management **19**
9.1 How can I back up my PVFS file system? . . . 20
9.2 Can I add, remove, or change the order of the PVFS servers on an existing PVFS file system? . . . 20
9.3 Are there tools for migrating data between servers? . . . 21
9.4 Why does df show less free space than I think it should? What can I do about that? . . . 21
9.5 Does PVFS have a maximum file system size? If so, what is it? . . . 21
9.6 Mouning PVFS with the interrupt option . . . 21

10 Missing Features **21**
10.1 Why don't hardlinks work under PVFS? . . . 21
10.2 Can I `mmap` a PVFS file? . . . 21
10.3 Will PVFS store new files on servers with more space, allowing files to be stored when one server runs out of space? . . . 22
10.4 Does PVFS have locks? . . . 22

11 Helping Out **22**
11.1 How can I contribute to the PVFS project? . . . 22

12 Implementation Details **22**
12.1 BMI . . . 22
12.1.1 What is the maximum packet size for BMI? . . . 22
12.1.2 What happens if I try to match a BMI send with a BMI receive that has too small a buffer? . . . 23

1 Basics

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

1.1 What is PVFS?

PVFS 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 PVFS a perfect research testbed as well.

1.2 What is the history of PVFS?

PVFS was first developed at Clemson University in 1993 by Walt Ligon and Eric Blumer as a parallel file system for Parallel Virtual Machine (PVM). It was developed as part of a NASA grant to study the I/O patterns of parallel programs. PVFS version 0 was based on Vesta, a parallel file system

developed at IBM T. J. Watson Research Center. Starting in 1994 Rob Ross re-wrote PVFS to use TCP/IP and departed from many of the original Vesta design points. PVFS version 1 was targeted to a cluster of DEC Alpha workstations networked using switched FDDI. Like Vesta, PVFS striped data across multiple servers and allowed I/O requests based on a file view that described a strided access pattern. Unlike Vesta, the striping and view were not dependent on a common record size. Ross' research focused on scheduling of disk I/O when multiple clients were accessing the same file. Previous results had show than scheduling according the best possible disk access pattern was preferable. Ross showed that this depended on a number of factors including the relative speed of the network and the details of the file view. In some cases a scheduling that based on network traffic was preferable, thus a dynamically adaptable schedule provided the best overall performance.

In late 1994 Ligon met with Thomas Sterling and John Dorband at Goddard Space Flight Center (GSFC) and discussed their plans to build the first Beowulf computer. It was agreed that PVFS would be ported to Linux and be featured on the new machine. Over the next several years Ligon and Ross worked with the GSFC group including Donald Becker, Dan Ridge, and Eric Hendricks. In 1997 at a cluster meeting in Pasadena, CA Sterling asked that PVFS be released as an open source package.

In 1999 Ligon proposed the development of a new version of PVFS initially dubbed PVFS2000 and later PVFS2. The design was initially developed by Ligon, Ross, and Phil Carns. Ross completed his PhD in 2000 and moved to Argonne National Laboratory and the design and implementation was carried out by Ligon, Carns, Dale Witchurch, and Harish Ramachandran at Clemson University, Ross, Neil Miller, and Rob Lathrum at Argonne National Laboratory, and Pete Wyckoff at Ohio Supercomputer Center. The new file system was released in 2003. The new design featured object servers, distributed metadata, views based on MPI, support for multiple network types, and a software architecture for easy experimentation and extensibility.

PVFS version 1 was retired in 2005. PVFS version 2 is still supported by Clemson and Argonne. Carns completed his PhD in 2006 and joined Axicom, Inc. where PVFS was deployed on several thousand nodes for data mining. In 2008 Carns moved to Argonne and continues to work on PVFS along with Ross, Latham, and Sam Lang. Brad Settlemyer developed a mirroring subsystem at Clemson, and later a detailed simulation of PVFS used for researching new developments. Settlemyer is now at Oak Ridge National Laboratory. in 2007 Argonne began porting PVFS for use on an IBM Blue Gene/P. In 2008 Clemson began developing extensions for supporting large directories of small files, security enhancements, and redundancy capabilities. As many of these goals conflicted with development for Blue Gene, a second branch of the CVS source tree was created and dubbed "Orange" and the original branch was dubbed "Blue." PVFS and OrangeFS tracked each other very closely, but represent two different groups of user requirements.

1.3 What is OrangeFS?

Simply put, OrangeFS is PVFS. OrangeFS is a branch of PVFS created by the Clemson team PVFS developers to investigate new features and implementations of PVFS. As of fall 2010 OrangeFS has become the main branch of PVFS. So why the name change? PVFS was originally conceived as a research parallel file system and later developed for production on large high performance machines such as the BG/P at Argonne National Lab. OrangeFS is taking a slightly different approach to support a broader range of large and medium systems and a number of issues PVFS was not concerned with including security, redundancy, and a broader range of applications. The new name reflects this new focus, but for now at least, OrangeFS is PVFS.

The PVFS web site is still maintained. The PVFS mailing lists for users and developers have not changed and will be used for OrangeFS. At some point in the future another group may decide to

branch from the main but the PVFS site will remain the home for the community.

1.4 What is Omnibond?

Omnibond is a software company that for years has worked with Clemson University to market software developed at the university. As of fall 2010 Omnibond is offering commercial support for OrangeFS/PVFS. OrangeFS is open source and will always be free; and the code, as always, is developed and maintained by the PVFS community. Omnibond is offering profesional services to those who are intersted in it, and directly supports the PVFS community. Omnibond offers its customers the option of dedicated support services and the opportunity to support the development of new features that they feel are critical. Omnibond gives back to the community through their support and development.

1.5 What does the "V" in PVFS 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.

1.6 Is PVFS an attempt to parallelize the *NIX VFS?

No, and we're not even sure what that means! The design of PVFS 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.

1.7 What are the components of PVFS that I should know about?

The PVFS Guide (http://www.pvfs.org/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.

1.8 What is the format of the PVFS version string?

PVFS uses a three-number version string: X.Y.Z. The first number (X) represents the high level design version of PVFS. The current design version is 2, and will likely remain there. The second number (Y) refers to the major version of the release. Major versions are incremented with new features, protocol changes, public API changes, and storage format changes. The third number (Z) refers to the minor version of the release, and is incremented primarily for bug fix releases.

With our 2.6.0 release, we changed the release version and name from PVFS2 1.x.x, to PVFS 2.x.x. Users familiar with 'PVFS2' and had been using PVFS2 1.5.1 will find the same software in PVFS version 2.6.0 or later (with updates and new features of course).

Users of PVFS version 1 can still go to: http://www.parl.clemson.edu/pvfs, although we highly encourage you to upgrade to PVFS version 2, if you are still using version 1.

2 Supported Architectures and Hardware

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

2.1 Does PVFS require any particular hardware?

Other than hardware supported by the Linux OS, no. PVFS uses existing network infrastructure for communication and can currently operate over TCP, Myrinet, and InfiniBand. Disk local to servers is used for PVFS 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 PVFS support?

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

The (optional) part of PVFS 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 PVFS work across heterogeneous architectures?

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

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

You can run the userspace PVFS 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 PVFS 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 PVFS 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 PVFS kernel module?

To our knowledge, PVFS 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 PVFS client require a patched Linux kernel?

No. The kernel module source included with PVFS is generally targeted toward the official "Linus" kernels (found at kernel.org). Patches for the PVFS 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 PVFS 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 PVFS?

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

3 Installation

This section covers issues related to installing and configuring PVFS.

3.1 How do I install PVFS?

The PVFS 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 PVFS 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 PVFS 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 PVFS 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 PVFS servers on the same node?

If you do decide to run more than one PVFS 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 PVFS?

Absolutely! Any PVFS 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 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 PVFS 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 PVFS 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 PVFS servers listen on two network interfaces simultaneously (i.e. multihome)?

Yes! PVFS servers can listen on more than one interface at a time. Multihome support was added shortly before the PVFS2 1.0 release.

3.8 How can I automount PVFS volumes?

The Linux automounter needs some help dealing with PVFS'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 PVFS 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 PVFS v1 to PVFS v2?

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 PVFS 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 PVFS is the PVFS web site at http://www.pvfs.org/. Documentation (including this FAQ) is also available in the `doc` subdirectory of the PVFS source distribution. Please reference `pvfs2-logging.txt` to understand more about PVFS' informational messages, where the logs exist, and how to turn logging on and off.

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 PVFS 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 PVFS?

First you will need to join the PVFS2 Users Mailing list at http://www.beowulf-underground.org/mailman/listin 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 PVFS
- 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 PVFS 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 PVFS 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 PVFS 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 PVFS. gdb indicates that the server is spending a lot of time in the glibc routine `'.handle_kernel_aio'`. 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. We have also seen this behavior on ppc64 systems running glibc-2.3.3-18.ydl.4 . If you encounter this problem and your distribution does not have an updated glibc package, you can configure pvfs2 with `--disable-aio-threaded-callbacks`, though this will result in a performance hit. An alternate workaround is to set `LD_ASSUME_KERNEL` to 2.4.1 before running pvfs2-server. pvfs2-server will then use an older (and not as optimized) thread library that does not have this bug.

At this time we do not know which of the two suggested workarounds is better from a performance standpoint. The `LD_ASSUME_KERNEL` method might make more sense: when/if the system's

glibc is upgraded, you will only have to restart pvfs2-server with the environment variable unset. You would not have to rebuild pvfs2 to take advantage of the fix.

5.3 PVFS write performance slows down dramatically

Phil Carns noticed that on some kernels, write-heavy workloads can trigger a kernel bug. The symptoms are that the PVFS server will only be able to deliver a few KB/s, and the CPU utilization will be close to 100%. The cause appears to be related to ext3's "reservation" code (designed to reduce fragmentation). The solution is to either mount the filesystem with the 'noreservation' option, or upgrade your kernel.

For more information, including URLs to several other reports of this issue, see Phil's original post: http://www.beowulf-underground.org/pipermail/pvfs2-developers/2006-March/001885.html

5.4 I get "error while loading shared libraries" when starting PVFS programs

PVFS needs several libraries. If those libraries aren't in the default locations, you might need to add flags when running PVFS's configure script. At configure time you can, for example, pass `--with-db=/path/to/db --with-gm=/path/to/gm` to compile with Berkeley DB and Myiricom GM libraries. The configure options let the compiler know where to find the libraries at compile time.

Those compile-time options, however, aren't enough to find the libraries at run-time. There are two ways to teach the system where to find libraries:

- add /usr/local/BerkeleyDB.4.3/lib to the /etc/ld.so.conf config file and re-run 'ldconfig' OR
- add /usr/local/BerkeleyDB.4.3/lib to the `LD_LIBRARY_PATH` environment variable.

I would suggest the ld.so.conf approach, since that will work for all users on your system.

5.5 PVFS performance gets really bad once a day, then gets better again

Several sites have reported poor PVFS performance early in the day that eventually goes away until the next day, when the cycle begins again. Daily cron jobs might be the culprit in these cases. In particular, most Linux distributions have a daily cron job (maybe called 'slocate', 'locate' or 'updatedb') that indexes the entire file system. Networked file systems such as NFS are often excluded from this indexing.

The exact steps to remove PVFS from this indexing process vary among distributions. Generally speaking, there should be a cron script in `/etc/cron.daily` called 'slocate' or 'updatedb'. That script should have a list of excluded file systems (like /var/run and /tmp) and flie types (like 'proc' and 'nfs'). Either add 'pvfs2' to the list of file types or add the pvfs2 mount point to the list of excluded file systems. Be sure to do this on all machines in your cluster.

5.6 Make kmod24 fails with "structure has no member..." errors

On some Redhat and Redhat-derived distributions, "make kmod24" might fail with errors like this:

```
console]:make kmod24
 CC [M]  /usr/src/pvfs2/src/kernel/linux-2.4/pvfs2-utils.o
pvfs2-utils.c: In function ‘mask_blocked_signals’:
pvfs2-utils.c:1063: structure has no member named ‘sig’
```

```
pvfs2-utils.c:1070: structure has no member named ‘sigmask_lock’
pvfs2-utils.c:1073: too many arguments to function ‘recalc_sigpending’
pvfs2-utils.c: In function ‘unmask_blocked_signals’:
pvfs2-utils.c:1082: structure has no member named ‘sigmask_lock’
pvfs2-utils.c:1084: too many arguments to function ‘recalc_sigpending’
make[1]: *** [pvfs2-utils.o] Error 1
make: *** [kmod24] Error 2
```

Redhat, and derived distributions, have a linux-2.4 based kernel with many linux-2.6 features backported. These backported features change the interface to the kernel fairly significantly. PVFS versions newer than 1.0.1 have a new configure option `--enable-redhat24`. With this option, we will be able to accommodate the backported features (and the associated interface changes).

5.7 When i try to mount a pvfs2 file system, something goes wrong.

- First, are all the userspace components running? If `pvfs2-ping` doesn't work, the VFS interface won't, either.
- Make sure the pvfs2 kernel module is loaded
- Make sure pvfs2-client and pvfs2-client core are running
- Take a look at dmesg. `pvfs2_get_sb -- wait timed out` could indicate a problem with `pvfs2-client-core`. See the next question.

5.8 I did all three of the above steps and I still can't mount pvfs2

There's one last thing to check. Are you you are using a Redhat or Fedora distribution, but running with a stock kernel.org 2.4 kernel? If so, you need to set the environment variable `LD_ASSUME_KERNEL` to 2.4.1 or `pvfs2-client-core` will try to use the NPTL thread library. NPTL requires a 2.6 kernel (or a heavily backported 2.4 kernel, which Redhat provides). Redhat systems expect to have such a kernel, so running a stock kernel.org 2.4 kernel can cause issues with any multi-threaded application. In this particular case, the `pvfs2-client-core` failure is hidden and can be tricky to diagnose.

5.9 I'm running Redhat and the pvfs2-server can't be killed! What's wrong?

On some Redhat systems, for compatibility reasons, the pvfs2-server program is actually a script that wraps the installed pvfs2-server binary. We do this ONLY if we detect that PVFS is being installed on a system with an NPTL implementation that we're incompatible with. Specifically, the script exports the LD_ASSUME_KERNEL=2.2.5 environment variable and value to avoid using the NPTL at run-time. The script quite literally exports this variable and then runs the installed pvfs2-server binary which is named `pvfs2-server.bin`. So to properly shutdown or kill the pvfs2-server application once it's running, you need to issue a `killall pvfs2-server.bin` command instead of the more common `killall pvfs2-server` command.

5.10 Why do you single out Redhat users? What's so different about Redhat than other distributions?

Some Redhat versions (and probably some other less popular distributions) use a heavily modified Linux 2.4.x kernel. Due to the changes made in the memory manager and signal handling, our default Linux 2.4.x kernel module will not even compile! We have compatibility code that can mend the differences in place, but we have to be able to detect that you're running such a system. Our configure script tries hard to determine which version you're running and matches it against a known list. If you suspect you need this fix and our script does not properly detect it, please send mail to the mailing list and include the contents of your /etc/redhat-release file.

In addition, some Redhat versions ship with an NPTL (threading library) implementation that PVFS is not compatible with. We cannot explain why the errors we're seeing are occurring, as they appear to be in glibc and the threading library itself. In short, we disable the use of the NPTL on these few Redhat systems. It should be noted that we are fully compatible with other distributions that ship NPTL libraries (such as Gentoo and Debian/unstable).

5.11 Where is the kernel source on a Fedora system?

Older systems used to split up the kernel into several packages (`kernel`, `kernel-headers`, `kernel-source`). Fedora kernels are not split up that way. Everything you need to build a kernel module is in /lib/modules/‘uname -r‘/build. For example, Fedora Core 3 ships with linux-2.6.9-1.667. When configuring PVFS, you would pass `--with-kernel=/lib/modules/2.6.9-1.667/build` to the configure script.

In Fedora Core 4 things changed a little bit. In order to build the pvfs2 kernel module, make sure you have both a `kernel` and `kernel-devel` package installed. If you have an SMP box, then you'll need to install the -smp versions of both – i.e. `kernel-smp` and `kernel-smp-devel`. After both packages are installed, /lib/modules/‘uname -r‘/build will once again contain a correctly configured kernel source tree.

5.12 What are extended attributes? How do I use them with PVFS?

Extended attributes are name:value pairs associated with objects (files and directories in the case of PVFS). They are extensions to the normal attributes which are associated with all objects in the system (i.e. the stat data). A complete overview of the extended attributes concepts can be found in man pages section 5 for attr. On supported 2.4 kernels and all 2.6 kernels, PVFS allows users to store extended attributes on file-system objects through the VFS as well as through the system interface. Example usage scenarios are shown below, To set an extended attribute (”key1”, ”val1”) on a PVFS file foo,

```
prompt# setfattr -n key1 -v val1 /path/to/mounted/pvfs2/foo
```

To retrieve an extended attribute for a given key (”key1”) on a PVFS file foo,

```
prompt# getfattr -n key1 /path/to/mounted/pvfs2/foo
```

To retrieve all attributes of a given PVFS file foo,

```
prompt# getfattr -m "" /path/to/mounted/pvfs2/foo
```

Note that PVFS uses a few standard names for its internal use that prohibit users from reusing the same names. A list of such keys are as follows at the time of writing of this document ("dir_ent", "root_handle", "datafile_handles", "metafile_dist", "symlink_target"). Further, Linux also uses a set of reserved keys to hold extended attributes that begin with the prefix "system.", thus making them unavailable for regular usage.

5.13 What are Access Control Lists? How do I enable Access Control Lists on PVFS?

Recent versions of PVFS support POSIX Access Control Lists (ACL), which are used to define fine-grained discretionary access rights for files and directories. Every object can be thought of as having associated with it an ACL that governs the discretionary access to that object; this ACL is referred to as an access ACL. In addition, a directory may have an associated ACL that governs the initial access ACL for objects created within that directory; this ACL is referred to as a default ACL. Each ACL consists of a set of ACL entries. An ACL entry specifies the access permissions on the associated object for an individual user or a group of users as a combination of read, write and search/execute permissions.

PVFS supports POSIX ACLs by storing them as extended attributes. However, support for access control based permission checking does not exist on 2.4 Linux kernels and is hence disabled on them. Most recent version of the Linux 2.6 kernels do allow for such permission checks, and PVFS enables ACLs on such kernels. However, in order to use and enforce access control lists on 2.6 kernels, one must mount the PVFS file system by specifying the "acl" option in the mount command line. For example,

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

Please refer to the man pages of "setfacl", "getfacl" or section 5 acl for detailed usage information.

5.14 On SLES 9, 'make kmod' complains about `mmgrab` and `flush_icache_range` being undefined

SLES 9 (and possibly other kernels) makes use of internal symobls in some inlined kernel routines. PVFS2-1.3.2 or newer has the configure option `--disable-kernel-aio`. Passing this option to configure results in a pvfs2 kernel module that uses only exported symbols.

5.15 Everything built fine, but when I try to compile programs that use PVFS, I get undefined references

The `libpvfs2` library requires a few additional libraries. Usually "-lpthread -lcrypto -lssl" are required. Further, Myrinet and Infiniband have their own libraries. If you do not link the required libraries, you will probably get errors such as `undefined reference to 'BIO_f_base64'`.

The easiest and most portable way to ensure that you link in all required libraries when you link `libpvfs2` is to use the `pvfs2-config` utility. `pvfs2-config --libs` will give you the full set of linker flags needed. Here's an example of how one might use this tool:

```
$ gcc -c $(pvfs2-config --cflags) example.c
$ gcc example.o -o example $(pvfs2-config --libs)
```

5.16 Can we run the Apache webserver to serve files off a PVFS volume?

Sure you can! However, we recommend that you turn off the EnableSendfile option in httpd.conf before starting the web server. Alternatively, you could configure PVFS with the option `--enable-kernel-sendf` Passing this option to configure results in a pvfs2 kernel module that supports the sendfile callback. But we recommend that unless the files that are being served are large enough this may not be a good idea in terms of performance. Apache 2.x+ uses the `sendfile` system call that normally stages the file-data through the page-cache. On recent 2.6 kernels, this can be averted by providing a `sendfile` callback routine at the file-system. Consequently, this ensures that we don't end up with stale or inconsistent cached data on such kernels. However, on older 2.4 kernels the `sendfile` system call streams the data through the page-cache and thus there is a real possibility of the data being served stale. Therefore users of the `sendfile` system call are warned to be wary of this detail.

5.17 Trove-dbpf metadata format version mismatch!

In PVFS2-1.5.0 or newer the format of the metadata storage has change from previous versions (1.4.0 or earlier). This affects users that have created file systems with the earlier versions of pvfs2, and wish to upgrade to the most recent version. We've provided a migration tool that must be run (a one-time only procedure) to convert the file system from the old format to the new one. The migration tool can be used as follows:

```
$PVFS_INSTALL/bin/pvfs2-migrate-collection --all fs.conf server.conf-<hostname>
```

This command finds all the pvfs2 storage collections specified in the configuration files and migrates them to the new format. Instead of using `--all`, the option `--fs` can be used to specify the name of the storage collection that needs to be migrated (usually there's only one storage collection, with the default name of 'pvfs2-fs').

5.18 Problems with pre-release kernels

For better or worse, the Linux kernel development process for the 2.6 series does not make much effort to maintain a stable kernel API. As a result, we often find we need to make small adjustments to the PVFS kernel module to track recent kernel additions or changes.

If you are using a pre-release kernel (anything with -rc in the name), you stand a good chance of running into problems. We are unable to track every pre-release kernel, but do make an effort to publish necessary patches once a kernel is officially released.

5.19 Does PVFS work with Open-MX?

Yes, PVFS does work with Open-MX. To use Open-MX, configure PVFS with the the same arguments that you would use for a normal MX installation: "–disable-bmi-tcp" and "–with-mx=PATH". In addition, however, you must set the "MX_IMM_ACK" environment variable to "1" before starting the pvfs2-server or pvfs2-client daemons. This is necessary in order to account for differences in how MX and Open-MX handle message progression by default.

6 Performance

This section covers issues related to the performance of PVFS.

6.1 I configured PVFS with support for multiple intercdonnects (e.g. Infiniband and TCP), but see low performance

When multiple interconnects are enabled, PVFS will poll both interfaces. This gives PVFS maximum flexiblity, but does incur a performance penalty when one interface is not being used. For highest performance, configure PVFS with only one fast method. Consult the `without-bmi-tcp` option or omit the `with-<METHOD>` option when configuring PVFS.

Note that it can sometimes be useful to have multiple interconnects enabled. The right choice depends a lot on your situation.

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

We designed PVFS 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 PVFS 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. PVFS has several more aggressive optimizations that can be turned on, but those optimizations require that applications accessing PVFS can cope with out-of-sync caches.

In the future, PVFS 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 PVFS. Furthermore, because of their serial nature, PVFS will be unable to deliver its full performance. Instead try running a parallel file system benchmark like IOR (ftp://ftp.llnl.gov/pub/siop/ior/).

6.3 Why is program XXX so slow?

See Question 6.2. If the program uses small block sizes to access a PVFS 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. If you set the value to no and the server is terminated unexpectedly, you will likely lose data (or access to it). Also, PVFS has a transparent server side attribute cache (enabled by default), which can speed up applications which read a lot of attributes (`ls`, 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.

6.4 NFS outperforms PVFS for application XXX. Why?

In an environment where there is one client accessing a file on one server, NFS will outperform PVFS 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 PVFS, at which point we will be playing on a level field, so to speak. However, if you insist on benchmarking PVFS and NFS in a single-client test, there are some immediate adjustments you can make.

The easiest way to improve PVFS performance is to increase the block size of each access. Large block sizes help most file systems, but for PVFS 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 PVFS configuration file, the server will sync data to disk only when a flush or close operation is called. The `TroveSyncMeta` 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 PVFS is not a wise choice of file system for you. PVFS is not designed for serial workloads, particularly one with small accesses.

6.5 Can the underlying local file system affect PVFS performance?

Yes! However, the interaction between the PVFS 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 PVFS 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.

6.6 Is there any way to tune particular directories for different workloads?

Yes. This can be done by using extended attributes to set directory hints. Three hints are currently supported, and they allow you to specify the distribution, distribution parameters, and number of datafiles to stripe across. They will not change the characteristics of existing files, but they will take effect for any newly created files within the directory. These hints will also be inherited by any new subdirectories.

6.6.1 Distribution

The distribution can be set as follows:

```
prompt# setfattr -n "user.pvfs2.dist_name" -v "basic_dist" /mnt/pvfs2/directory
```

Supported distribution names can be found by looking in the pvfs2-dist-* header files.

6.6.2 Distribution parameters

Some distributions allow you to set parameters that impact how the distribution behaves. These parameters can be set as follows:

```
prompt# setfattr -n "user.pvfs2.dist_params" -v "strip_size:4096" /mnt/pvfs2/directory
```

You can specify more than one ”parameter:value” pair by seperating them with commas.

6.6.3 Number of datafiles

You can also specify the number of datafiles to stripe across:

```
prompt# setfattr -n "user.pvfs2.num_dfiles" -v "1" /mnt/pvfs2/directory
```

PVFS defaults to striping files across each server in the file system. However, you may find that for small files it is advantages to limit each file to only a subset of servers (or even just one).

6.7 My app still runs more slowly than I would like. What can I do?

If you ask the mailing list for help with performance, someone will probably ask you one or more of the following questions:

- Are you running servers and clients on the same nodes? We support this configuration – sometimes it is required given space or budget constraints. You will not, however, see the best performance out of this configuration. See Section 3.6.
- Have you benchmarked your network? A tool like netpipe or ttcp can help diagnose point-to-point issues. PVFS will tax your bisection bandwidth, so if ppossible, run simultaneous instances of these network benchmarks on multiple machine pairs and see if performance suffers. One user realized the cluster had a hub (not a switch, a hub) connecting all the nodes. Needless to say, performance was pretty bad.
- Have you examined buffer sizes? On linux, the settings /proc can make a big difference in TCP performance. Set `/proc/sys/net/core/rmem_default` and `/proc/sys/net/core/wmem_default`

Tuning applications can be quite a challenge. You have disks, networks, operating systems, PVFS, the application, and sometimes MPI. We are working on a document to better guide the tuning of systems for IO-intensive workloads.

7 Fault Tolerance

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

7.1 Does PVFS support some form of fault tolerance?

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

7.2 Can PVFS tolerate client failures?

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

7.3 Can PVFS 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 PVFS tolerate network failures?

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

7.5 Can PVFS tolerate server failures?

Yes. We currently have a recipe describing the hardware and software needed to set up PVFS in a high availability cluster. Our method is outlined in the 'pvfs2-ha.{ps,pdf}' file in the doc subdirectory of the PVFS 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 PVFS file systems.

8.1 How do I get MPI-IO for PVFS?

The ROMIO MPI-IO implementation, as provided with MPICH2 and others, supports PVFS. 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 PVFS files on the PVFS servers without going through some client interface?

You can, yes, but you probably should not. The PVFS 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 PVFS file systems.

9.1 How can I back up my PVFS file system?

The default storage implementation for PVFS (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 PVFS v1, and is one of the many improvements present in the new system.

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

You can add and change the order of PVFS servers for an existing PVFS 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?

PVFS 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 PVFS data is stored). The `pvfs2-statfs` utility, included with PVFS, can be used to check the amount of free space on each server, as can the `karma` GUI.

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

PVFS 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 PVFS server storage and multiply these values by the number of servers you are using.

9.6 Mouning PVFS with the interrupt option

The PVFS kernel module supports the `intr` option provided by network file systems. This allows applications to be sent kill signals when a filesystem is unresponsive (due to network failures, etc.). The option can be specified at mount time:

```
mount -t pvfs2 -o intr tcp://hosta:3334/pvfs2-fs /pvfs-storage/
```

10 Missing Features

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

10.1 Why don't hardlinks work under PVFS?

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

10.2 Can I `mmap` a PVFS file?

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

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

No. Currently PVFS 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 PVFS 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 PVFS. 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 PVFS project.

11.1 How can I contribute to the PVFS project?

There are lots of ways to directly or indirectly contribute to the PVFS project. Reporting bugs helps us make the system better, and describing your use of the PVFS system helps us better understand where and how PVFS 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 PVFS. See http://www.pvfs.org/pvfs2/devel 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 PVFS.

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 PVFS 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.