Frequently Asked Questions about PVFS2

PVFS2 Development Team

June 8, 2004

Contents

1 Basics

This section covers some basic questions about what PVFS2 is and where it can be used.

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 architectures does PVFS2 support?

The majority of PVFS2 is POSIX-compliant C code that runs in user space. As such, much of PVFS2 could run on most available systems.

However, the part of PVFS2 that hooks to the operating system on clients must be written specifically for the particular operating system. This piece of PVFS2 has only been written for the Linux kernel, version 2.6.xx.

We encourage porting of this component of PVFS2 to other operating systems. At this time we have no plans for porting this component to operating systems other than Linux.

1.3 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.

1.4 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.

1.5 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 Installation

This section covers issues related to installing and configuring PVFS2.

2.1 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.

2.2 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.

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

Running more than one PVFS2 server on the same node 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).

2.4 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.

2.5 Can PVFS2 servers listen on two interfaces (multihome)

PVFS2 servers currently listen on one interface at a time. Multihome support, wehre a PVFS2 server can accept connections from several network interfaces, would be helpful in many situations. We will implement this feature at some point, but not in the next few releases.

2.6 PVFS2 and automount

The Linux automounter needs some help dealing with PVFS2's resource strings. A typical mount command 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://cave0\: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 Reporting Problems

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

3.1 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.

3.2 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.

3.3 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/p 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.

4 Performance

This section covers issues related to the performance of PVFS2.

4.1 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. 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 agressive 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. Try running a parallel file system benchmark like IOR (http://www.llnl.gov/asci/purple/benchmarks/limited/ior/).

4.2 Why is program XXX so slow?

See question 4.1. If the program uses small block sizes to access a PVFS2 file, performance will suffer.

The `TroveSyncMode` config file option can improve performance in some situations, but if the server dies, you will lose data. PVFS2 has an Attribute Cache, which can speed up applications which read a lot of attributes (`ls`, for example). Playing around with the `AttrCache*` config file settings might yield some performance improvements, or it might not.

5 Fault Tolerance

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

5.1 Does PVFS2 support some form of fault tolerance?

Systems can be set up to handle many types of failures for PVFS2. Currently the only failure mode that cannot be handled well is server failure.

5.2 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.

5.3 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 2.2 for more information on this approach.

5.4 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.*

5.5 Can PVFS2 tolerate server failures?

At this time we do not have a solution for this problem. We are currently experimenting with clustering of servers for failover, but we do not have a cookbook recipe for setting this up. We will update this are more information becomes available.

6 Interfaces

This section covers issues related to accessing PVFS2 file systems.

6.1 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

6.2 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...

7 Management

This section covers questions about managing PVFS2 file systems.

7.1 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.

7.2 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 Section 7.3) both in the case of a new server, or if you wanted to migrate data off a server before removing it.

7.3 Are there tools for migrating data between servers?

Not at this time, no.

7.4 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.

7.5 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.

8 Missing Features

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

8.1 Why don't hardlinks work under PVFS2?

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

8.2 Can I `mmap` a PVFS2 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 PVFS2 that would compromise the design goals of simplicity and robustness. This "feature" was intentionally left out, and it will remain so.

8.3 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 9.1 for notes on how you could help get this feature in place.

9 Helping Out

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

9.1 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/develop for more details.

10 Implementation Details

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

10.1 BMI

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

10.1.1 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.

10.1.2 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.