Project Report for Project Shared Parallel File System

Fangbin Liu
fliu@os3.nl

04-02-2006

Abstract

In recent years, shared parallel file system has been a new hot area for the high throughput computing. Many systems have been developed for this purpose, which include PVFS2 and GFS. These systems were being studied in my project.

In this project, the concept of such a shared parallel file system has been built up by installing it on four nodes of the LISA cluster at SARA (Stichting Academische Rekeningcentrum Amsterdam). After the Installation of PVFS2 and GFS, a number of experiments have been organized to compare the performance of two file systems. The results are discussed in the experiment section of this report.

The installation of PVFS2 and GFS as well as how to configure such file systems on the LISA cluster nodes are also described in this report.

In this report, the following aspects have been included:

1. Background information about shared parallel file system.
2. The PVFS2 files system.
3. The GFS file system.
4. The comparison of two file systems.
5. What problems could be encountered when exploring the possibility of the maximum performance of two systems.
6. At the end, the conclusion of this project comparing two file systems and the recommendation for the usage by SARA.

Contents

Chapter 1

Background information on Parallel File System

In this chapter, the concept of parallel file system is introduced. Then, a short review about the present development of distributed file systems will be given, as well as the utilities.

With the fast development of software production, the speed of CPU and raw computering power of industry conmodity PCs tends to get more and more out of pace. Following this trend, many solutions for this problem have been implemented worldwide. High Throughput Computing is one area emerging out of this trend. From now on, this set of words are presented here with an abbreviation as "HTC".

For HTC, the importance does not sit in the achievement of the performance of the system per second or per hour. Rather than that, interest here stands in the quantities of operations a system is capable of handling within a longer period, like one month or even more. To achieving an optimal performance for this kind of evaluations, a huge number of work stations are assembled in a parallel manner to optimize the performance. Also, to improve the throughput of a system, it is believable that different sort of resources should be utilized. In this way, a high throughput can be still achieved even though some resources might defect during the job execution for all kinds of unknown reasons.

In such an environment, a massive amount of data will be generated. These data must be stored in such a way that the performance of the work

stations can not be violated. Also, at the same time, these data can also be accessed by multiple programs on diverse nodes in the system. As a result, the storage devices in such a environment must be able to supply services for all the nodes in the system in a optimal manner. For this purpose, effect of I/O operations on the system performance must be minimized.

At this moment, many system use NFS as its file system which stands for Network File System. In this system, multiple clients have to communicate with one server where all the data are stored and accessed. As a result, this server has been a bottleneck for the performance of whole system. when the number of clients increases, the load of server would become extremely huge.

Some new file systems have been implemented in the last few years worldwide, among which PVFS2 and GFS sit, to solve the problem mentioned above. These two file systems are both good at supplying a parallel manner for file storage and access management. They both provide a interface for the nodes in the system to communicate with storage device, and a storage device management entity. With this structure of the system, storage devices can be extended gradually and integrated into a kind storage "pool" for the utilization later. More than just a storage management system, both of these systems are able to supply a monitoring function. They are able to monitor the states of storage devices in the system. By distributing the job among diverse devices, the load balancing effect between different storage nodes can be realized.

In the recent years, the parallel file systems have been used in many different network systems in companies and research institutes. In this way, a huge monetary save has been realized through utilizing multiple low-cost work stations than supercomputers.

Chapter 2

PVFS2 File System

As a new parallel file system developed in the recent year, the PVFS2 system will be introduced in this section. A global description of this system will be given and also the main functionalities the system supplies us. More details about PVFS2 system could be found in book [1].

2.1 Introduction of PVFS2 File System

As stated in [1], PVFS2 is a parallel file system. It can be used for multiple programs to access one shared data storage device in a coordinated manner. In PVFS2 file system, many servers can be used to supply the access to the stored data. In this way, multiple paths to the data storage can be realized. Also, a high performance can be supplied since the data in this system can be accessed in a really parallel manner. This feature can not be found in many other distributed file systems.

In PVFS2 file system, every server can be used in two ways as shown in figure 2.1. In the first way, the server can be used as one of the parallel data servers. In this way, the server is responsible for the management and access of data stored on the local disks. In the other way, a server can play a role as metadata server which does not manage the file data itself but the meta data about the file. No matter which role the server plays, the data managed by this server will all be stored in the local disk. When the server starts, it will read the configuration file to find out which role it need to play. In the PVFS2 file system, the file data is just stored in the UNIX files and the metadata is mostly stored in the Berkeley DB database. An API called Trove has been built in the PVFS2 file system to hide the

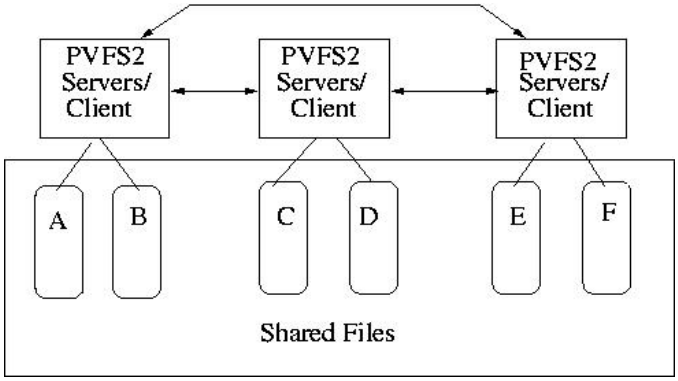

Figure 2.1: PVFS2 File System

management of these file data and metadata.

At this moment, two low−level I/O interface has been supplied by PVFS2. The client can use normal UNIX API presented on his system to communicate with PVFS2 server. In this manner, a kernel module must be loaded. It will export the functions to user−space. Then, a UNIX process, called pvfs2−client, will handle the communication with the PVFS2 servers. The other interface is called MPI−IO interface. For this interface, the ROMIO MPI−IO implementation is used to link directly to low−level PVFS2 application interface to access data on servers.

In the PVFS2 file system, no locking algorithm is used. Instead of it, a access sequence algorithm has been implemented. The PVFS2 file system force all the operations who intend to modify the file system hierarchy to be carried out in a special manner. For this manner, every access will be performed in a number of steps. As a result, every access can be viewed as a atomic operation and the file system hierarchy is always in a consistent state. In this way, the file system consistency can be held successfully.

2.2 Features of PVFS2 File System

PVFS2 file system supplies many new features in the latest version. Some of them will be listed below:

- Ease of installation.
- As mentioned about, multiple interfaces are supported by PVFS2 such as MPI−IO and UNIX IO.
- For PVFS2, no special network and storage hardware needed to be used. Commodity networks and storage hardware can be applied overall.
- Another strong point of PVFS2 file system is that it is designed in modular manner. For network communication, a Buffered Messaging Interface (BMI) has been used. For local data management, the storage interface called Trove supplies API for it.
- In PVFS2, the metadata does not need to be stored on one server. Instead they can be stored on multiple servers. In this way, applications accessing different data do not need to impact each other any more.

2.3 Deployment of PVFS2 File system

To set up a PVFS2 file system over a cluster. The next steps need to be completed. More details can be found in the book [2].

1. Untarring the packages downloaded from the website.
2. Building and installing the packages with “configure”, “make”, “make install”.
3. Configuring the server with command “pvfs2-genconf”.
4. Transferring the configuration files over the servers in cluster.
5. Starting the servers with command “pvfs2-server”.
6. Configuring the client in the file “/etc/pvfs2tab”.
7. Preparing Linux kernel with “configure”.
8. Loading the kernel module “pvfs2.ko”.

9. Starting the clients with command “pvfs2-client”.
10. Mounting the PVFS2 file system

Chapter 3

GFS File System

As another parallel file system developed in the recent year, the GFS file system will be introduced in this section. Like in previous chapter, a global description of this system will be given and also the main functionalities the system supplies us will be discussed here. More details about GFS file system could be found in book [3].

3.1 Introduction of GFS File System

As stated in the document [3], Red Hat GFS is a cluster file system that is available with Red Hat Cluster Suite. In GFS file system, the cluster node is managed by Red Hat cluster management tools. GFS will supply a uniform and consistent view of the data storage among the Red hat cluster nodes. The cluster node does not need to care about how the data storage is implemented under the GFS file system or which infrastructure is used to store data.

Also, multiple paths are supported in the GFS file system. This feature can be realized with a SAN network for the storage devices. In This way, the storage devices can be applied by many different servers. This feature could help file system to realize the load balancing amount many servers. Also, it can help to built up a more robust system, allowing unpredictable server failure.

GFS can be installed either directly on the SAN devices, or on GNBD (Global Network Block Device) storages. A superior performance and scalability can be achieved by SAN storage, whereas a easy establishment

can be realized with GNBD servers connected with LAN.

A GFS file system is built up with some logical volumes. These logical volumes are managed by CLVM (Cluster Logical Volume Manager), a cluster enabled logical volume manager. CLVM creates the logical volumes using physical volumes located on diverse servers. These physical volumes is held by alternative servers distributed across the network. In this way, users will not need to care where the data is stored and how to access it. Users will directly access data located in the logical devices in the same way as accessing the data on local disk.

In this project, a GFS file system has been built up with GNBD devices connected with LAN as shown in the figure 3.1.

3.2 Deployment of GFS File system

Before loading GFS file system, a number of packages from diverse developer teams need to be installed. The main steps will be discussed below. More details can be found in appendix.

1. First of all, a cluster must be established. The cluster configuration tools and servers can be achieved from Cluster Configuration System package, called ccs.
2. Then, GNBD package need to be installed to set up GNBD tools and servers.
3. Further, the Cluster LVM package needs to be installed to achieve the LVM cluster support.
4. To be able to run a cluster, the cluster management package "cman" need to be installed.
5. Also, fence package needs to be installed to isolate the failing servers.
6. In the end, a lock algorithm must be selected. There are normally two options: Distributed Lock Manager (dlm) and Grand Unified Locking Manager (gulm).
7. To start the GFS file system, a cluster must be formed through the distributed joining to the cluster.

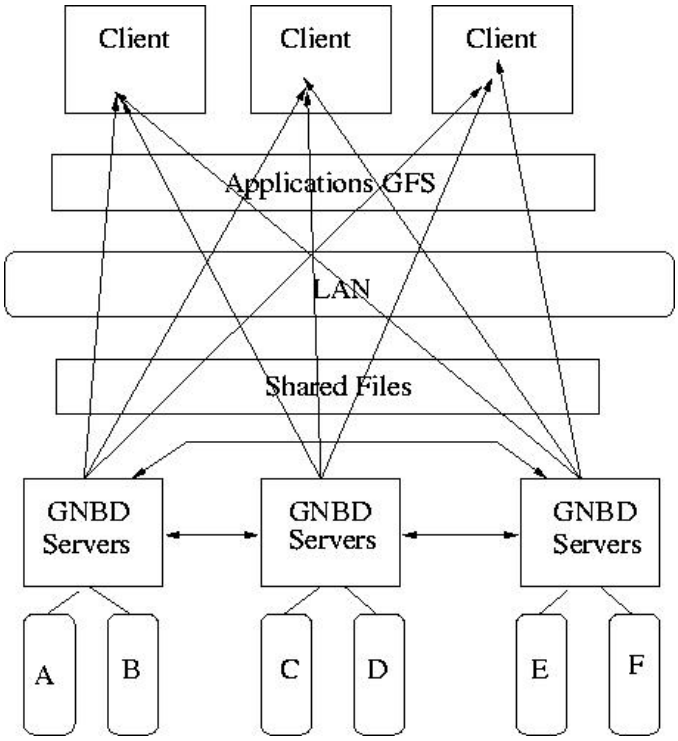

Figure 3.1: GFS and GNBD with Directly Connected Storage

8. Next, the GNBD devices need to be published overall and the logical volume need to be created upon the physical volume.

9. After all, the GFS file system can be loaded on the created logical volume. Till this point, the logical device can be mounted on a mount point locally and be further accessed.

Chapter 4

Comparison between PVFS2 and GFS

In this section, the two file systems described in the previous chapters will be compared from alternative viewpoints. Firstly, the comparison will be made from the point of the internal Working theories and the features supplied. After this analysis, the test results in this project will be discussed to compare the difference of the two file systems' performance. Also, the test settings and test operations will be supplied in this section.

4.1 Work Theory

From the point of user, both of systems hide the file storage allocation underneath a file system access interface. A client does not need to specify the physical destination of the file operation. The remote and local storage device can be directly mount on a point in the system directory hierarchy. But, on the other hand, in PVFS2 system, the storage server can also be the client within the system. Every node can act more than one role. Also, every client in PVFS2 system can mount different remote servers. In this way, the work load will be better distributed and balanced.

But, in GFS file system, a cluster logical volume manager (clvmd) is used. It is responsible for the data exchange between various servers. Also, to generate a logical volume, a volume group must be firstly made. This volume group includes all the physical volumes within the system. To create such a volume group among all the storage servers, in GFS system built up in this project, GNBD (Global Network Block Device) are exported and

imported between alternative servers. As a result, these servers can no longer create file system on the local disks which means the flexibility of the system is reduced significantly.

Since the logical volumes are used to store data in GFS system, the user will not be able to specify which servers will be utilized specially for himself. All the data from alternative users will be managed by cluster logical volume manager on diverse servers. This could generate some peak time at some moment when many clients try to access one logical volume except for the case that the client mount a special volume in the volume group.

Another important difference between PVFS2 and GFS is the locking algorithm. In the previous chapters, it has been shown that PVFS2 applies no locking method while GFS utilizes distributed lock manager to manage it. As a result, PVFS2 system becomes a stateless system. This property significantly the failure process when some client fails or crashes. When a client fails with its operations, the system does not need to carry out a complex sequence of operations to recover the system or locking re-validating. It can seamless continue to work as normal.

When a new server needs to be installed into the system, PVFS2 master server need to generate a new configuration file for it and for all the other servers also. This point can make PVFS2 system to be not such a good scalable system. On the other hand, For GFS, system, there is no master or slave, but well a cluster domain. Every node need to join the domain before it can be accessed by others. This feature enables the cluster domain to be simple to distribute and grow. The storage management is supplied by cluster logical volume manager. The new added server just needs to export his local disk to others and let clvmd to extend the volume group. In this way, the new storage device can be added and removed dynamically.

4.2 Test of Performance

To test two file systems performance, a number of IO tests have been organized on the four nodes of lisa cluster allocated for this project. After introducing how these experiments have been set up, the experiment results will be presented. Then, a short discuss will be found about some consequences found in the results.

4.2.1 Test Conditions

Four test nodes from LISA cluster at SARA (Stichting Academische Rekening Amsterdam) have been adopted in the experiments for this project. On each of this nodes, a P4 3G CPU with 2M second level cache has been installed. The memory size for every node is 2G. These four nodes have been connected with a gigabit Ethernet network.

4.2.2 Test Sets

Both PVFS2 and GFS file systems have been tested with one client and three servers and with two clients and two servers at each time. In this way, it can been probably seen what the effect is if we add more nodes to these file systems. In other words, the scalability and robustness can be shown in this way.

4.2.3 Test Program

The test program in the project to test the IO performance of the two files systems, PVFS2 and GFS, has been suggested to be Iozone. This program is a file system benchmark tool. The benchmark generates and measures a variety of file operations such as write, rewrite, read, reread, and so forth.

For the experiments in this project, the next options have been involved to test the system performance:

-a Used to select full automatic mode: Produces output that covers all tested file operations for record size of 4K to 16M for file sizes of 64K to 512K.

-i Used to specify which test to run. In the tests here, 0(=write/re-write) and 1(=read/re-read) are used

-g + a number, set maximum file size (in Kbytes) for auto mode (-a)

-R Generate Excel report. Iozone will generate an Excel compatible report to standard out.

-b + filename, Iozone will create a binary file format file in Excel compatible output of results.

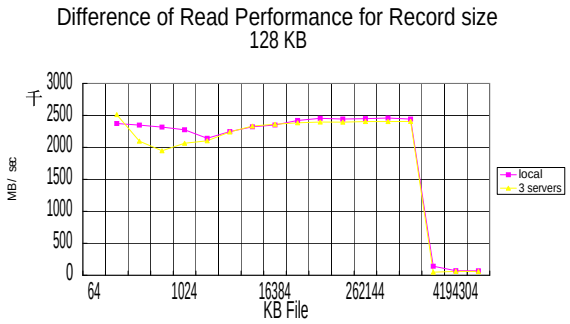

Figure 4.1: Comparison of Read performance for record size 128 KB

4.2.4 Test Results

Many experiment results have been achieved from the tests, but because of the space here, only the most important and most clear results will be presented here, more data can be found in the appendix B.

In figure 4.1, the difference of the read operations between gfs file system and local file system can be seen. This test has been carried out with 3 nodes as servers and one node as client. The X axis shows the file size and y axis shows the operations in every second. It can be seen that the read performance for both systems does not change so much when the size of file grows up gradually.

In this picture, it also shows the fact that the difference of performance between local XFS file system and GFS parallel file system is not so big when the number of clients are not so big, especially after the file size become bigger and bigger.

One important reason for this consequence is the fact that in GFS

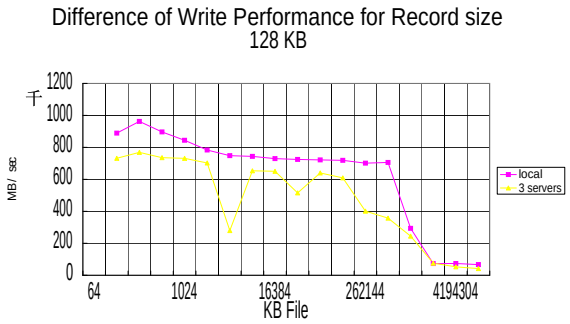

Figure 4.2: Comparison of Write performance for record size 128 KB

file system, logical volume has been generated in a striped manner which enables clients to be able to read multiple sections of files in a parallel way. Further we can still see this effect in figure 4.2 which shows the write performance difference between gfs file system and local file system.

On the figure 4.3, the performances for write operation in GFS system with 2 servers and 3 servers has been compared. Firstly, it can be obvious that the performance does not change so much when more servers are utilized in the system. It means that when more servers will generate a significant better performance for the system.

Another interesting point from figure 4.3 is that the effect of cpu cache and memory cache can also be seen. When the file size become bigger than 1MB, a "step−effect" can be detected in the performance lines for both number of servers. This consequence comes from the fact that the file size does not fit the size of cpu cache anymore and thus must be accessed from the memory each time the file be read and wrote. A more significant step can be found when the file size becomes even bigger, bigger than the memory size. As a result, the file must be accessed each time from the real

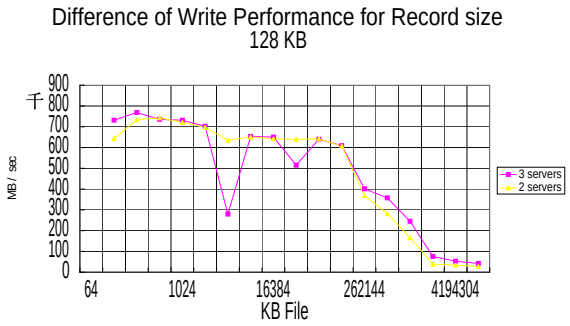

Figure 4.3: Comparison of Read performance for record size 128 KB

disk but not be able to saved in the system memory. In this way, the IO performance falls down obviously. A clearer example can be seen in the next figure 4.4.

From these two figures, the effect from client system memory can be seen clearly, but not the server system memory or cache, since in this project, the GFS file system has been built up with GNBD servers who exports his disks without caching effect. Thus, performance will also not increase with more server caches.

In figure 4.5, the write performance of PVFS2 file system can be seen when two servers are used in the system. Further in figure 4.6, the write performance of GFS file system is shown. From this two pictures, it can be convinced that when the number of clients are not so big, the performance of two file systems are very similar with each other. Also, from these two figures, the effect of buffer cache can be seen when the record size grows up.

More experiment results can be seen in the section appendix B.

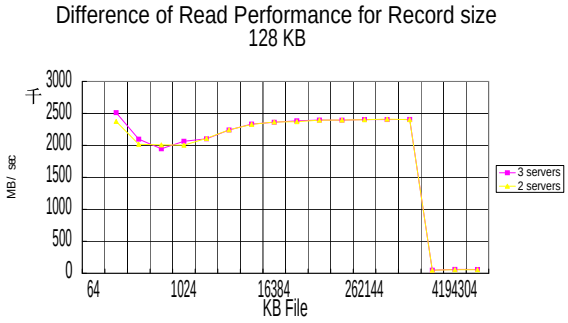

Figure 4.4: Comparison of Read performance for record size 128 KB

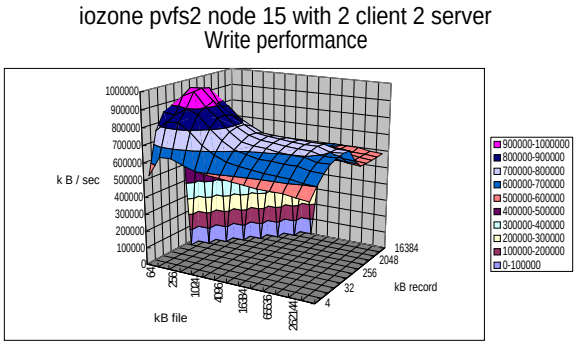

Figure 4.5: Write performance for PVFS2 file system with 2 clients 2 and servers

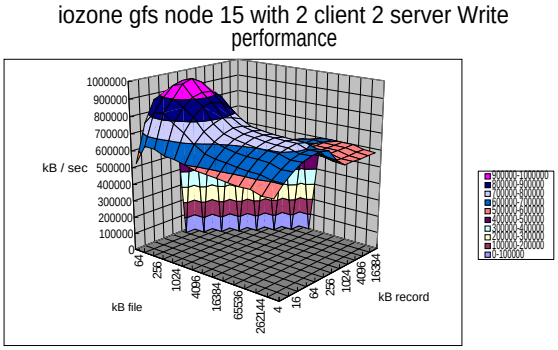

Figure 4.6: Write performance for GFS file system with 2 clients 2 and servers

Chapter 5

Future work and conclusions

In this chapter, a overview of what we have done in this project will be supplied, together with the find outs from this project. After a short conclusion for every file system, a recommendation will be offered according to the cases in SARA and a outlook for the future of parallel file system.

5.1 Project Conclusions

In this project, two file systems have been studied and tested. One is PVFS2 (Parallel Virtual File System 2), the other is GFS (Global File System) from Red Hat, Inc. Both of them can be a good choice for a parallel file system. They can offer a uniform and efficient interface for the users or clients to access the data store on the shared disk in the systems. Although there are many similar algorithms or spirits within these systems, there is still some difference which made them to be utilized in alternative environment.

5.1.1 PVFS2

PVFS2 file system has been developed for many years, since last century year 90's. This version of PVFS file system generate a more flexible shared file system, from the viewpoints of both users and administrators. Data can be located in many different ways. Internal modules are easier for developers to upgrade and improve. Management of such a file system can be executed in a simple and efficient way. Installation and configuration is also very straightforward.

But, on the other hand, there is also some shortage in PVFS2. When

the shared file system become bigger and bigger, the task of adding new servers to the system would be more difficult. The administrator of the system will need to generate a new configuration file for every server so that they can know each other and communicate with each other. Also, when a server crashes or down for some unforeseeable reason, the system must let the users wait until this server can work normally again.

5.1.2 GFS

GFS is a good choice in many environments. GFS supplies dynamic extendable shared file storage for all the clients. With the fencing domain function, the system can still maintain a health state, once the most nodes in the system are still running. Also, GFS makes it very easy to add new storage into the system. A new storage server just needs to join the domain and export its disk to others. Also, the disk in the GFS can be utilized in a more efficient manner with the support of cluster logical volume manager. The logical volume can be dynamic extended or reduced, according to the needs of user. Also, GFS can be used in multiple manners with various combination of storage device and network infrastructure.

But, every coin has two sides. GFS has also some weak points. The installation and configuration for such a system needs more knowledge about cluster domain and cluster configuration tools. When a new storage server is added to the system, the cluster logical volume manager needs to be started again to find out the added volume. Also the volume must be imported by every client and servers. For a big cluster, this would be a problem.

5.2 Future work

After comparing in this project, for the project officer from SARA, the file system PVFS2 is selected to be the one which is more proper for them. Without caching maintain algorithm and locking-module, PVFS2 can support multiple clients accessing the file storage in a parallel manner. There can be more flexibility for the users to choose their shared disks.

This system can be very proper for a large cluster when many servers and clients exists The client is very easy to join the cluster or leave the cluster, not like in GFS where if a client want to leave the domain, firstly stop all the IO to the shared disks and then stop the various daemons. And when the client tries to use the system again, a long time must be wasted on

import all the remote volumes again. What's more, PVFS2 is also simpler for installation and configuration.

Appendix A

Installation of PVFS2 and GFS file system on the local server

In this chapter, the installation steps for PVFS2 and GFS file systems on the four test nodes are introduced. It should the same procedure when more nodes need to use PVFS2 and GFS system.

A.1 Installation Steps of PVFS2

The installation of PVFS2 can be carried out in the following manner:

1. Download the PVFS2 package from *http://www.pvfs.org/pvfs2/download.html* .
2. You change to the directory where you have saved the downloaded PVFS2 package and untar the package to a directory , and then change to that directory.
3. Run the command:

   `./configure`

   , then

   `make`

   and

```
make install
```

.

4. Generate the server configuration file with command

   ```
   pvfs2-genconf /etc/pvfs2-fs.conf /etc/pvfs2-server.conf
   ```

   .

5. Then copy the file system general configuration file and the server specific configuration file to each server in the cluster.

6. Run the command:

   ```
   pvfs2-server /etc/pvfs2-fs.conf
   /etc/pvfs2-server.conf-<hostname> -f
   ```

   . This command will generate the data storage for the PVFS2 file system on this server.

7. Run the command:

   ```
   pvfs2-server /etc/pvfs2-fs.conf
   /etc/pvfs2-server.conf-<hostname>
   ```

   . This command will really start the server on this node.

8. On every client, generate a file called “

   ```
   pvfs2tab
   ```

   ” in directory

   ```
   /etc/
   ```

   .

9. Edit this file and add one line for every server like this:

   ```
   tcp://<server-host-name>:3334/pvfs2-fs<space>/mnt/pvfs2
    pvfs2 default, noauto 0 0
   ```

.

10. Further, on the clients, make a directory for PVFS2 system with command:

    ```
    mkdir /mnt/pvfs2
    ```

    .

11. To generate the PVFS2 module, first run the configuration command in the untarred directory again with option

    ```
    --with-kernel=<kernel-source-location>
    ```

    .

12. Then, type the command:

    ```
    make kmod
    ```

    , and then

    ```
    make kmod_install
    ```

    , till this point, the kernel module is generated.

13. Next, add the kernel module with command:

    ```
    insmod /<source-directory-pvfs2>/src/kernel/linux-2.6/pvfs2.ko
    ```

14. Start the PVFS2 client:

    ```
    pvfs2-client
    ```

    .

15. Mount the file system on a mount point:

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

A.2 Installation Steps of GFS

The installation of GFS can be carried out in the following manner:

1. Download and install all the package needed for GFS with command:

   ```
   apt-get install <package-name>
   ```

   in the next order:

   ```
   cman-kernel > dlm-kernel > gnbd-kernel > gfs-kernel > magma >
     iddev > ccs > cman > dlm > gnbd > gfs > fence > gulm > magma-plugins >rgmanage
   ```

   .

2. add the next module to the kernel:

   ```
   lock_harness, dm-mod, gfs, cman,
     lock_dlm, dlm
   ```

   with the next command

   ```
   modprobe <module-name>
   ```

   .

3. Make sure that the above tasks are done on every node which will be included in to the GFS system. Then run the next command on every node:

   ```
   ccsd
   ```

   start cluster configuration daemon. For this daemon, a configuration file called “

   ```
   cluster.conf
   ```

   ” need to be generated in the directory

   ```
   /etc/
   ```

. The contents for this file can be found in the man manual or internet.

4. Then on every node, run:

   ```
   cman_tool join -w
   ```

   to form the domain.

5. Then on every node, run:

   ```
   fence_tool join -w
   ```

   to generate a fence domain and start fence daemon.

6. Then on every node, run:

   ```
   clvmd
   ```

   .

7. On the servers, generate the physical volume with command:

   ```
   pvcreate <disk-name-in-/dev>
   ```

8. Start the GNBD (Global Network Block Device) server:

   ```
   gnbd_serv
   ```

9. Export the physical volume generated with command

   ```
   gnbd_export -d <physical-volume-device>
    -e <volume-name>
   ```

   .

10. Add the GNBD module to the kernel through running:

    ```
    modprobe gnbd
    ```

    .

11. Import the physical volume from other nodes by: run

```
gnbd_import -i <node-name>
```

12. Run the command:

    ```
    pvscan
    ```

    on all the nodes

13. Generate the volume group by running:

    ```
    vgcreate
      <volume-group-name> <phycial-device-local-or-remote> ...
    ```

    .

14. Run the command:

    ```
    vgchange -aly
    ```

    on all the nodes.

15. Create the logical volume on the volume group generated in the last step by running:

    ```
    lvcreate -L <size-of-logical-volume> --stripes <number-of-stripes>
       -n <name-of-logical-volume> <volume-group-name>
    ```

    .

16. Run the command:

    ```
    vgchange -aly
    ```

    on all the nodes again.

17. Run the command:

    ```
    lvscan
    ```

    to find out the logical volume on all the node.

18. Make the GFS file system on the generated logical volume by running:

```
gfs_mkfs -p
<local_protocal> -t <cluster-name-in-configuration-file>:<file-system-name>
 -j <number of journal> /dev/<volume-group-name>/<logical-volume-name>
```

.

19. Mount the file system on a point user chooses:

```
mount -t
  gfs /dev/<volume-group-name>/<logical-volume-name> <mount-point>
```

Till this point the GFS file system is installed on a client successfully.

Appendix B

Experiment result presentation

In this chapter some more experiment results are shown.

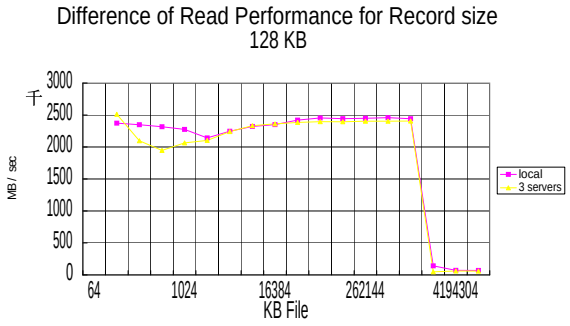

Figure B.1: Comparison of Read performance for record size 128 KB

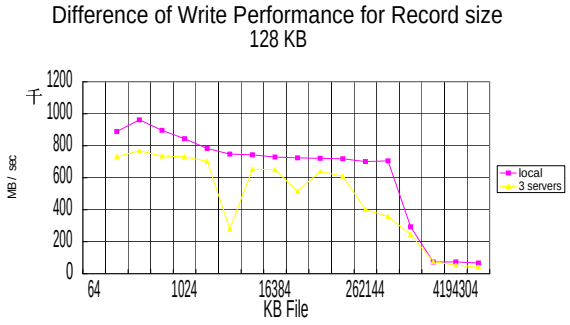

Figure B.2: Comparison of Write performance for record size 128 KB

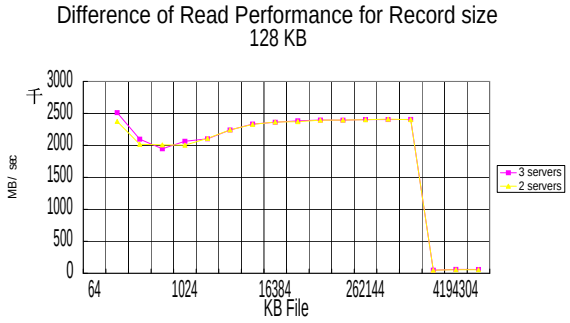

Figure B.3: Comparison of Read performance for record size 128 KB

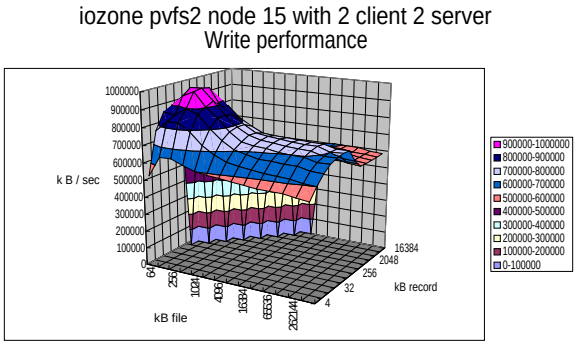

Figure B.4: Write performance for PVFS2 file system with 2 clients 2 and servers

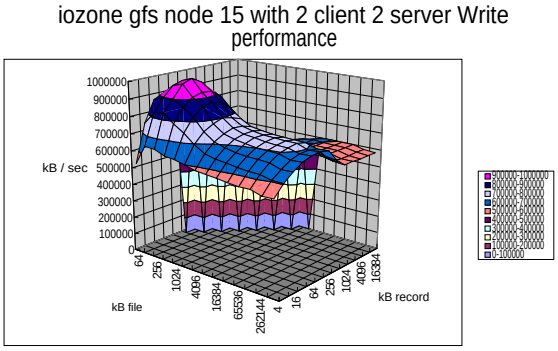

Figure B.5: Write performance for GFS file system with 2 clients 2 and servers

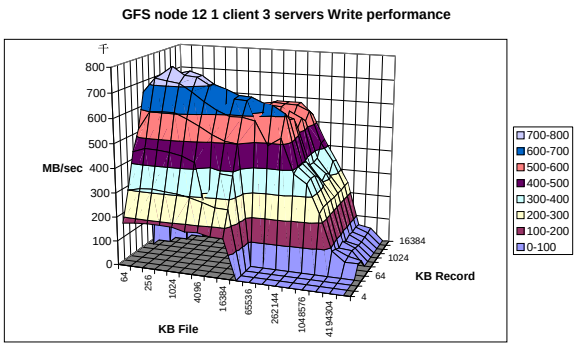

Figure B.6: Write performance for GFS file system with 1 clients 3 and servers

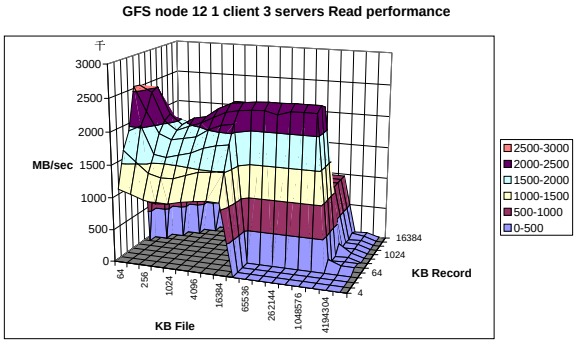

Figure B.7: Read performance for GFS file system with 1 clients 3 and servers

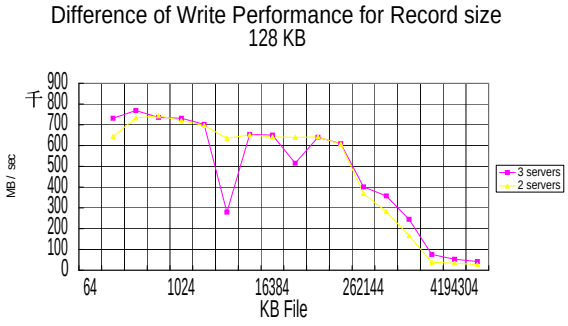

Figure B.8: Comparison of Write performance for record size 128 KB

Bibliography

[1] Parallel Virtual File System, Version 2, PVFS2 Development Team, September 2003

[2] A Quick Start Guide to PVFS2, PVFS2 Development Team, Last Updated: July 2005

[3] Red Hat GFS 6.1 Administrator's Guide, Redhat,Inc, 2005

[4] How to build the GFS software from scratch

[5] William D. Norcott, Don Capps: *Iozone Filesystem Benchmark*, http://www.iozone.org.

[6] Abhinay Ramesh Kampasi: *Design of a Cluster Logical Volume Manager*, Pune Institute of Computer Technology, University of Pune. http://abhinaykampasi.tripod.com.