Survey of File Systems for Clusters (And Storage Options)

Contributed by Jeff Layton
Sunday, 28 August 2005
Last Updated Thursday, 13 April 2006

Plenty of Options for Plenty of Files

Welcome to Cluster Money (aka' "The Monkey") and the Cluster File Systems Column. This column will be taking a look at these new file systems, how
they work, how they fit into the HPC world, and how you can deploy them for maximum effect. It will also explore some of the details of file systems to help you understand such things as the difference between metadata and inodes. It will also discuss some of the underlying hardware that is important to file systems such as networking and the data storage devices itself.

This first column will present a brief overview of many of the file systems that are available today for clusters as well as some storage options for the file systems. This includes true parallel
file systems, new network file systems, and even a storage option for high performance storage for clusters. The list isn't extensive but is intended to wet your appetite for more information about the explosion of storage that is happening.

Storage has become a very important part of clusters. Initially people used use NFS to distributed data to all of the nodes of the cluster. However, as clusters grew to hundreds and thousands of nodes, and the demand for increased data I/O rates grew, people realized that NFS was not going to cut it. So they used somewhat kludgy systems for a while or turned to other ideas such as PVFS (Parallel Virtual File System). Recently companies have started to realize that the market for HPC file systems is larger than they thought and largely untapped. Coupling this with very high-speed, low-cost networks such as Infiniband, results in the right time for an explosion of HPC file systems. This article will present a brief overview of some of the file systems that are available today for clusters. The list isn't extensive but is intended to wet your appetite for more information about the explosion of storage that is happening.

IBRIX

{mosgoogle right}
IBRIX
is a relatively new company offering a distributed file system that presents itself as a global name space to all the clients. IBRIX'

Fusion product is a software only product takes whatever data space you designate on what ever machines you choose and creates a global, parallel, distributed file system on them. This file system, or "name space," can be mounted by clients who can share the same data with all of the other clients. In essence, each client sees that exact same data, hence the phrase, "single" or "global" name space. The key to Fusion is that the common bottlenecks in parallel global file systems have been removed. Consequently, the file systems scales almost linearly with the number of data servers (also called IO servers). This architecture allows the file system to grow to tens of Petabytes (a Petabyte is about 1,000 Terabytes or about 1,000,000 Gigabytes). It can also achieve IO (Input/Output) speeds of Tens of Gigabytes per second for large or small files.

IBRIX has automatic fail-over as well as metadata journaling to speed recover in case of a crash. Perhaps more importantly IBRIX has developed a distributed metadata capability so losing several nodes will not result in losing access to any data. This unique feature also allows parts of the name space to be taken off line for maintenance, upgrades, or even backups, while the rest of the name space stays on-line. You can also add storage space while Fusion is running and it will automatically incorporate it. It can also export the file system using NFS or CIFS (for the Windows users that haven't gotten a clue yet.

It's easy to see that Fusion could be deployed in an HPC cluster by using all of the latent space available on the compute nodes. Since most nodes come with at least something like a 40 Gig or 80 Gig had drive and the OS only takes about 2-4 GB (Gigabytes) of space, you have some extra space to do something with. Fusion allows you to combine that extra space and create a global name space for all of the nodes within the cluster. Alternatively you could choose a few nodes and load them with storage space, create a global name space using the data servers, and mount it on the client nodes at speeds much faster than traditional NFS. These clients nodes don't need a local disk so you can run them diskless.

IBRIX Fusion currently has some limitations. It isn't 64-bit (yet), and requires IP for it's networking. Also, it doesn't support SUSE on the client nodes. However, IBRIX is aware of these issues and is working to provide all of these features.

Also, IBRIX Fusion comes bundled with a number of systems. For example, Dell is shipping Fusion with some of its cluster products. Also, recently, Rackable Systems has announced an OEM agreement with IBRIX. In addition, Scali has announced a reseller arrangement with IBRIX.

Polyserve

Polyserve Inc.
has a unique product in the storage world. The Polyserve Matrix Server takes up to 16 SAN (Storage Area Network) attached servers and creates a high-performance, low-cost NAS (Network Attached Storage) system.

Polyserve takes low-cost PC Servers running Linux that are attached via a Fibre Channel (FC) network to a SAN and installs their proprietary file system. This file system is a true symmetric file system with high availability services and cluster and storage management capabilities, providing a global name space. Polyserve states that there is not central lock of metadata servers so there is no single point of failure. It provides a global name space.

The servers that are part of the Matrix Server network can then export the file system via NFS to compute nodes within a cluster. Since there are up to 16 servers in the Matrix Server, each server could NFS support for a portion of the cluster. Also, since the file system is global, if one server goes down, another server can provide NFS services to the nodes the original server was servicing.

Panasas

Panasas is one of the storage vendors contending for a part of the HPC market. Their ActiveScale Storage Cluster is a high-speed, scalable, global, storage system that uses an object based file system called ActiveScale. Panasas couples their file system with a proprietary, but commodity based, storage system termed Panasas Storage Blades. The basic storage unit consists of a 4U chassis and a number of blades that fit into the chassis with direct attached storage (hard drives). In each chassis is also a director blade that is in essence a part of the file system.

{mosgoogle left}

This file system is one of the unique features of Panasas' storage system. ActiveScale turns files into objects and then dynamically distributes the data activity across Panasas Storage Blades. The role of the director blade is to virtualize the data objects (the files) and put them onto the storage blades. This is a unique concept where the storage finds the data rather than the usual approach of the data looking for the storage.

Each chassis that is part of the Panasas Storage Cluster is called a shelf. Each shelf can hold up to two director blades and 10 storage blades, creating up to 5 TB (Terabytes) of space across the 10 storage blades (500 Gigabytes of data per blade). Each shelf also has a built-in Gigabit (GigE) switch for traffic within the chassis and for traffic within other shelves or outside the storage system. Panasas claims that their Storage Cluster system can achieve a data throughput of up to 10 Gigabytes per second.

The ActiveScale Storage Cluster is very useful for providing high-speed storage within a cluster. A typical approach for HPC clusters would have a dedicated network for parallel communication and attach each node of the cluster to a storage network where the ActiveScale Storage Cluster is attached. Then each node can communicate directly to the file system.

Terrascale

Terrascale Technologies has a software only solution for high-performance storage for clusters. Their product, TerraGrid, uses standard Linux file system tools such as md, lvm, and evms in conjunction with Linux file systems such as ext2 for a global name space. The key to TerraGrid is the use of the iSCSI protocol together with proprietary drivers and file system patches to unify the storage space across multiple servers. It supports native Linux file systems and can export the file system using NFS and CIFS (for those occasional Windows hold out).

TerraGrid is a global name space file system. It uses the md tools in Linux to aggregate the space together, presenting the file system layer with a large multi-port virtual hard disk.

In tests TerraGrid enable compute nodes can sustain 100 Mbyte/sec (Megabytes per second) of single-stream I/O (Input/Output) performance. It scales fairly linearly to hundreds of nodes until either the network or the pool of I/O servers is saturated.

Data Direct Networks

While Data Direct Networks (DDN) does not deliver a complete storage solution with a storage system and a file system, they are a major distributor in the HPC and cluster market for a robust high-speed scalable storage

system. Their S2A8500 storage system can achieve 1.5 Gbytes/sec in sustained throughput with either Fibre Channel disks or Serial ATA (SATA) disks. The company says that they can scale from a handful of disks to over 1,000 disks. This corresponds to tens of Terabytes in space up to over a Petabyte of storage. This allows the throughput to scale from 1.5 Gigabytes/sec to tens of Gigabytes/sec.

The S2A8500 is a 2U box that supports four 2 GB/sec ports or two 4U boxes with eight 2 GB/sec ports. It can support up to 20 Fibre-Channel loops supporting Fibre or SATA disks. It can accommodate up to 1,120 disks resulting in up to 130 TB using Fibre Channel disks or 250 TB using SATA disks. The controllers can be configured in a SwiftCluster configuration to achieve over 1 Petabyte in storage.

The file system built using the storage system can be exported and mounted on the compute nodes using a variety of schemes. For instance, you can use normal NFS or connect the nodes using Fibre Channel networking for the compute nodes.

GPFS

IBM has had a global parallel file system for many years. The Global Parallel File System (GPFS) started with the IBM SP systems running AIX, but has been ported to Linux for certain IBM eServer, blade server, and xServer products.

GPFS is a high-speed, parallel, distributed file system that achieves high-performance by striping data across multiple disks and multiple nodes. To further improve performance, it uses client-side caching as well as read-ahead and write-behind functions. It can use large block sizes from 16KB to 1024 KB to help improve IO throughput depending upon requirements.

To get HA capability, it can be configured using logging and replication. It can be configured for fail over at a disk-level and at the server level. This means that if you lose a node GPFS will not lose access to data nor degrade performance unacceptably. As part of this it can transparently fail-over lock servers and other core GPFS functions so that the system stays up and performance is at an acceptable level.

Lustre
{mosgoogle right}

There is a relatively new open-source file system called Lustre that has been developed for Linux clusters. It follows a model that the newest version is only available from Cluster File Systems, Inc. while the previous version is available freely from www.lustre.org. It can potentially scale to ten of thousands of nodes and hundreds of Terabytes of storage. Lustre stores data as objects, called containers, that are very similar to files, but are not part of a directory tree. The advantage to an object based file system is that allocation management of data is distributed over many nodes, avoiding a central bottleneck. Only the file system needs to know where the objects are located, not the applications, so the data can be put anywhere on the network.

As with other file systems, Lustre has a metadata component, a data component, and a client part that accesses the data. However, Lustre allows these components to be put on various machines, or a single machine (usually for home clusters or for testing). The metadata can be distributed across machines called MetaData Servers (MDS), to ensure that the failure of one machine will not cause the file system to crash. The MetaData Servers support failover as well. In Lustre 1.x, you can use up to two MDS machines while in Lustre 2.x, you can have tens or even hundreds of MDS machines.

The file system data itself, is stored as objects on the Object Storage Servers (OSS) machines. The data can be spread across the OSS machines in a round-robin fashion (striped in a RAID sense) to allow parallel data access across many nodes resulting in higher throughput. The data is also distributed to ensure that a failure of an OSS machine will not cause the lose of data. The client machines mount the Lustre file system in the same way other file systems are mounted. They communicate with the OSS machines for direct data access.

The MDS machines, OSS machines, and clients can all reside on one machine (most likely for testing or learning) or can be split among multiple machines. For example, you could make all of the nodes within a cluster OSS machines and clients so that they can see the entire file system. You could choose one or more of the nodes in the cluster to be MDS machines for fail over redundancy.

Lustre uses open network protocols to allows the components to communicate. It uses an open protocol called

Portals,
originally developed at
Sandia National Laboratories. This allows the networking portion of
Lustre to be abstracted so new networks can easily be added.
Currently it supports TCP networks (Fast Ethernet, Gigabit Ethernet),
Quadrics Elan, Myrinet GM, Scali SDP, and Infiniband. Lustre also
uses Remote Direct Memory Access (RDMA) and OS-bypass capabilities to
improve I/O performance.

Recently some performance testing was done with Lustre on a cluster at
Lawrence Livermore National Laboratories (LLNL) with 1,100 nodes.
Lustre was able to achieve a throughput for each OST of 270 MB/sec
and an average client I/O throughput of 260 MB/sec. In total, for the
1,000 clients within the cluster, it achieved an I/O rate of 11.1
GB/sec.

PVFS

One of the first parallel file systems for Linux clusters was
PVFS (Parallel Virtual File System).
The original PVFS system, now called
PVFS1, was originally developed at Clemson University. It is not
designed to be a file system for users home directories but rather
a high-speed scratch file system for the storage of
input and output data during the run of a job. It is an open-source
project that has involved several developers. There is a new version
of PVFS called PVFS2 which is a complete rewrite the code to make
PVFS more expandable for new systems and to add new features.

PVFS2 is a complete rewrite of PVFS1 focusing on improving the scalability of
PVFS, the flexibility of PVFS, and the performance of PVFS. PVFS1 has
been modified for various networking protocols beside TCP, but the
modifications were very invasive and difficult to support. PVFS2 has
abstracted the networking layer allowing new networking protocols to
be used very quickly. For example, Infiniband support was added in
about a week with only about 3,000 lines of C code. Currently PVFS2
supports TCP, Infiniband, and Myrinet. In addition, PVFS2 accomodates
heterogeneous clusters allowing x86, PowerPC, Itanium machines to all
be part of the same PVFS file system. PVFS2 also adds some management
interfaces and tools to allow easier administration.

Both PVFS1 and 2 divide the functions of the file system into two pieces,
a metadata server and data servers, and allow clients to mount PVFS as
a normal file system. The metadata server holds information about the
data in the file system, and the IO devices (IOD's) actually hold the
data. In PVFS1 the metadata is put onto one machine. In PVFS2, the
metadata can be distributed. The data is then spread across the IO
devices (IOD's) which can be the nodes within a cluster. PVFS
stores the data as files on top of an existing file systems such as
ext3 or xfs. So you can mix the file systems under a PVFS system

without worry. When data is written to PVFS it is broken into chunks of a certain size and then written to the IOD's in some fashion, usually in a round-robin fashion. The size of the chunks, what IOD nodes are used for storage, and how the chunks are written, are all configurable, allowing PVFS to be tuned for maximum performance for a given application or class of applications.

One of the features of PVFS is that one of the IOD's can go down and PVFS will still be available. PVFS will recognize that one of the IOD's is down and not write to it. Instead it will go on to the next IOD in the list. Consequently, applications that are writing will not hang or crash (as long as they are not writing to the IOD node when it crashes). However, any application that was reading data that was on the IOD will not be able to fully access all of the data since that IOD is down. If the IOD is brought back into production without the storage space being altered, the data will once again be available for use. On the other hand, if the storage space on the returned IOD has been altered, such as re-installing Linux, then the file(s) that used that IOD are corrupt and cannot be recovered. This is a design decision that the PVFS developers have made. PVFS is designed to be a temporary high-speed storage space, not a place for home directories. The intent is to use PVFS for storage of data during an application run and then move the data off of PVFS onto a permanent storage space that is backed by tape or some other type of archiving media.

There are several ways to use PVFS. The best way to take advantage of the parallel capabilities involves using an MPI-IO implementation such as ROMIO or ChaMPIon that uses PVFS directly (either PVFS1 or PVFS2). You can also use the normal Linux kernel interface but you won't get the parallel speed of PVFS. This interface allows PVFS to be mounted as a normal file system on the client nodes and for users to interact with PVFS using typical commands such as ls or mv or rm. Currently PVFS1 and PVFS2 cannot run binaries (i.e. you can't run an application stored in PVFS). In the case of PVFS1, there is also a third interface,
the PVFS library. This library has semantics very similar to the standard C library file functions. It allows the parallel performance capabilities of PVFS to be used, but requires some changes to your application.

PVFS is in use at various sites around the world. PVFS1 is in production use at several locations including the University of Utah and the Ohio State Supercomputer Center. PVFS2 in production use as well. It is being used on Orion Multi-System desktop clusters.

The performance of PVFS is heavily dependent upon the network performance. The typical ethernet networks will work well, but will limit scalability and ultimate performance. High-speed networks such as Myrinet, Quadrics, and Infiniband (IB) greatly improve scalability

and performance. In tests, the IO performance of PVFS scales linearly with the number of IOD's up to at least 128 IODs. At that point, the performance exceeds 1 Gigabyte/sec. PVFS2 has been tested with 350 IOD's, each with a simple IDE and connected via Myrinet 2000. With 100 clients, PVFS2 was able to achieve an aggregate performance of about 4.5 Gigabyte/sec in writes and almost 5 Gigabyte/sec in read performance running an MPI-IO test code.

GFS

A few years ago some researchers from the University of Minnesota formed a company called Sistina to develop open-source file system tools for Linux. They developed lvm (Logical Volume Manager) for Linux and GFS (Global File System). The goal of GFS is to provide a true global name space for clustered machines using various networking options (such as Fibre Channel or GigE) that is resilient to server or network loss.

After a period of time, Sistina took GFS and made it closed-source. Recently, Redhat has purchased Sistina and re-open-sourced GFS.
Redhat is offering GFS with commercial support as part of their Red Hat Cluster Suite for $499 for up to 8 nodes (it requires a subscription to Red Hat Enterprise Linux (RHEL) on these servers).

GFS is a global distributed file system that can run on Linux systems that are connected to a Storage Area Network (SAN). These networks can be constructed with Fibre Channel or iSCSI networks. It has distributed locking as well as distributed meta data so if a server in the file system goes down no data or access to data is lost. It has a dynamic path capability so in the event that a switch or HBA (Host Bus Adapter) goes down, it can still get to the other servers in the SAN. It has quota capability as well. GFS is flexible enough that various configurations for clusters are possible. This includes both networking and storage options as well. Redhat is claiming that GFS scales to hundreds of machines and tens of Terabytes.

iSCSI

With the advent of high-speed networks and faster processors, the ability to centralize storage and allocate it to various machines on the network has taken off. SAN systems use this approach but use expensive and proprietary Fibre Channel (FC) networks and in some cases proprietary storage media. An open initiative to replace the FC network with common IP based networks and common storage media was begun. This initiative, called iSCSI (internet SCSI), was developed by

the Internet Engineering Task Force (IETF).

{mosgoogle right}

iSCSI encapsulates SCSI commands in TCP (Transmission Control Protocol) packets and sends them to the target computer over IP (Internet Protocol) on an ethernet network. The system then processes the TCP/IP packet and processes the SCSI commands. Since SCSI is bi-directional, any results or data in response to the original request are passed back to the originating system. Thus a system can access storage over the network using standard SCSI commands. In fact, the client computer (called an initiator) does not even need a hard drive in it at all and can access storage space on the target computer using iSCSI. Using iSCSI, the storage space appears as though it's physically attached (via a block device) and a file system can be built on it.

The overall basic process for iSCSI is fairly simple. Assume that a user or an application on the initiator makes a request of the iSCSI storage space. The operating system creates the corresponding SCSI commands, encapsulates them, perhaps encrypting them, and puts a header on the packet. It then sends them over the IP network to the target. The target decrypts the packet (if encrypted) and then separates out the SCSI commands. The SCSI commands are then sent to the SCSI controller and any results of the command are returned to the original request. Since IP networks can be lossy where packets can either be dropped, or have to be resent, or arrive out of order, the iSCSI protocol has had to develop techniques to accommodate these and similar situations.

There are several desirable aspects to iSCSI. First, no new hardware is required either by the initiator (client) or the target (server). The same physical disks, network cards, and network can be used for an iSCSI network. Consequently the startup costs are much less than a SAN. Second, iSCSI can be used over wide area networks (WANs) that span multiple routers. SANs are limited to their distance based on their configuration. Also, theoretically, since iSCSI is a standard protocol, you can mix and match initiators and targets across various operating systems.

There are several Linux iSCSI projects. The most prominent is an iSCSI initiator
that was developed by Cisco and open-sourced. There are patches for 2.4 and 2.6 kernels. Many Linux distributions ship with the initiator already in the kernel. An iSCSI
target package
is also available, but only for 2.4 kernels (this package is sometimes called the Aristech target package). It allows Linux machines to be used as targets for iSCSI initiators. There is also a
project
originally developed by Intel and open-sourced.

A fork of the Ardistech iSCSI target package was made a with an eye towards porting it to the Linux 2.6 kernel and adding features to it (the original Aristech iSCSI target package has not been developed for some time). Then this project was combined with the iSCSI initiator project to develop a combined initiator and target package for Linux. This package is under very active development and fully supports the Linux 2.6 kernel series.

There is a very good HOWTO on how to use the Cisco initiator and the Ardistech target package in Linux. There is also an article on how to use iSCSI as the root disk for nodes in a cluster. This could be used to boot diskless compute nodes and provide them with an operating system located on the network.

There are several ways to use iSCSI with a cluster. A simple way would be to use a few disk-full nodes within a cluster as targets for the rest of the compute nodes in the cluster that are the initiators. The compute nodes can even be made diskless. Parts of the disk subsystem on each target node would be allocated to a compute node. A separate storage network can be utilized to increase throughput of iSCSI. The compute node can then format and mount the disk as though it were a local disk. This architecture allows the storage to be concentrated in a few nodes for easier management. More over, lvm can be used to provide space in an intelligent manner for the compute nodes so that space can be expanded.

HyperSCSI

HyperSCSI is a related protocol to iSCSI. It uses a different packet encapsulation than the TCP encapsulation of iSCSI and sends its packets over raw ethernet. HyperSCSI is being developed by researchers at the Data Storage Institute that is affiliated with the National University of Singapore and has been placed on sourceforge. The researchers have developed HyperSCSI under a GNU GPL (GNU Public License) license on Linux platforms. The developers say they have focused on developing a fast, efficient, and secure protocol that can be easily used on common, inexpensive ethernet networks.

Similar to iSCSI, HyperSCSI wraps the SCSI commands to transmit the packet to the target system over the network . However, in contrast to iSCSI, HyperSCSI uses its own packet header rather than a TCP header. This

approach promises to be more efficient because the TCP overhead has been eliminated. The target system then decodes and executes the SCSI commands. Thus, any HyperSCSI equipped system, even one without a disk or a SCSI controller, can access a HyperSCSI exported device as though they were a local device. You can even run lvm (Logical Volume Management) and RAID (Redundant Array of Inexpensive Disk) tools on these mounted devices.

The performance of HyperSCSI is also quite good. Between two systems over Gigabit Ethernet, the developers have achieved over 99% performance of a local disk using several benchmarks. The developers of HyperSCSI also claim that they can get better performance than iSCSI. For instance, they claim that they can match Fibre Channel performance with only a 21% increase in CPU utilization and 3.4 times more IRQ (Interrupt Requests) per second than Fibre Channel. To match the same Fibre Channel performance the HyperSCSI developers say that software based iSCSI requires a 33% increase in CPU utilization and 6 times more IRQs per second than Fibre Channel.

This article was originally published in ClusterWorld Magazine. It has been updated and formated for the web. If you want to read more about HPC clusters and Linux you may wish to visit Linux Magazine.

Dr. Jeff Layton hopes to someday have a 20 TB file system in his home computer. He lives in the Atlanta area
and can sometimes be found lounging at the nearby Fry's, dreaming of hardware and drinking coffee (but never during working hours).