Dominique A. Heger, Fortuitous Technology, (dom@fortuitous.com), Austin, TX, 2006

An Introduction to File System Technologies in a Cluster Environment

Introduction

Traditional local file systems support a persistent name space. A local file system views devices as being locally attached, the devices are not shared, and hence there is no need in the file system design to enforce device-sharing semantics. Instead, the focus is on aggressively caching and aggregating file system operations to improve performance by economizing on the number of actual disk accesses required for each file system operation. Newer networking technologies allow multiple machines (nodes) to share storage devices. IBM' General Parallel File System (GPFS) or Red Hat' Global File System are representing distributed file system technologies that are taking a shared, network-attached storage approach. These file systems are built on the premise that a shared disk file system has to exist within the context of a cluster infrastructure, and has to provide proper error handling and recovery, as well as the best performance possible (performance, availability, and scalability features are key requirements). In a SAN attached environment, SAN clients may only manage local file system requests and act as file managers for there own I/O operations. Hence, the storage devices (the IO subsystem) serve the data directly to the clients. A cluster file system design provides transparent parallel access to storage devices while maintaining standard UNIX file system semantics. User applications only view a single logical device via the standard *open(), close(), read(), write()* and *fcntl*() primitives*.* This transparency is paramount in regards to ease of use, as well as to the portability of these file system technologies. To reiterate, a cluster file system based design differs from traditional, local file systems, by emphasizing sharing, connectivity, as well as (client side) caching. Unlike local file systems such as IBM's J2 or SGI' XFS, cluster file systems distribute file system resources (including the metadata) across the entire storage subsystem, which allows simultaneous access from multiple machines.

The goal of this article is to elaborate on the terminology's surrounding the file systems that are being used in a cluster based environment. Some of the terms actually overlap, which may result in misconceptions and confusion on the user site, an issue that is addressed in this report. Further, this document discusses the classification of some the applications that may be executed on a cluster. To illustrate cluster file system technology, the last part of this article briefly introduces IBM's GPFS file system, focusing on setup, configuration, availability opportunities, as well as by addressing some of the performance related features embedded into the GPFS framework.

Cluster File Systems and Cluster Applications

In general, cluster file systems complement the actual hardware and software cluster facilities in various ways. Traditionally, a cluster represents a group of nodes, acting as a single system. That definition may get stretched considerably though, as cluster technology today represents a dynamic field, encompassing a divers application spectrum that is continually absorbing new features. Furthermore, cluster file system technologies, whether open source or proprietary, are rapidly converging in their capabilities. Many people in the industry refer to cluster applications and the underlying file system software being used as a *unit*. More accurately though, most clusters consist of two main components, (1) the nodes, which are connected to some sort of interconnect, and (2) the cluster file system, which acts as the software layer that enables the cluster nodes to share data and work together *as a unit*. In general, cluster applications can be classified based on their varying levels of maturity and capabilities:

1. *High performance clusters*, which are also referred to as computational cluster systems. These systems are normally utilized to support very large data volumes (of computational processing). In such an environment, a parallel file system distributes the processing resources across the nodes, thereby allowing each node to access the same set of files concurrently (via concurrent *read()* and *write()* requests).

2. *High availability (HA) clusters*, which are designed for fault tolerance or redundancy purposes. As these clusters normally use one or more servers (for data processing), the servers in the cluster are able to assume processing responsibilities in case that one or more of them goes down.

3. *Load balancing clusters* distribute the workload as evenly as possible across multiple server systems, such as web or application servers, respectively.

4. *Storage cluster systems*, which are utilized between Storage Area Network (SAN) components and server systems with different operating systems. These systems provide shared access to data blocks on a common storage media.

5. *Database cluster systems*, such as Oracle's Real Application Cluster (RAC) platform, which introduce many of the cluster file system features into the application-layer itself.

Cluster File System Terminology

Cluster applications, such as the once mentioned above, provide overlapping features, and hence features of one or more of them are commonly found in a single cluster application (especially in a HA or load-balancing cluster environment). To illustrate, an Oracle RAC solution can be installed and configured over an HA cluster file system (to provide the benefits of database clustering to an actual HA cluster application). The filed of clustered file systems itself can be described as following:

1. The term *distributed file system* reflects the generic description of a client-server or network file system, respectively. In such a scenario, the user data is not locally attached to a host system. In this category, NFS and CIFS are the most common distributed file systems being used today. It has to be pointed out that in NFS, there is a *N:1* relationship between the nodes and the server system. Hence, in a general NFS environment, the NFS server represents a single point of failure.

2. The term *global file system* refers to the actual namespace, in that all the files in the environment have the same name and path when viewed from all the nodes. Therefore, a global file system allows for a simple way to share data across nodes and users that are part of different entities of an organization. To illustrate, the WWW represents a global namespace, as a unique URL works from anywhere on this planet. AFS represents an early provider of a global namespace (*/afs/cell-name*), as it is feasible to assemble AFS cells from different organizations or institutions into a single shared file system.

3. The term *SAN file system* represents a way to provide nodes with the capability to share Fibre Channel (FC) storage, a storage solution that is traditionally decomposed into private chunks that are bound to different hosts. To provide data sharing, a block-level metadata manager governs access to the different SAN devices. A SAN file system mounts storage natively on a node, but connects all the nodes to that storage, and distributes block addresses to the other nodes in the cluster. Some of the more popular SAN file systems in use today are SGI' cXFS, IBM' GPFS, or Red Hat' GFS.

4. The term *symmetric file system* describes a file system in which the client nodes also have to run the metadata manager code. In other words, all the nodes in the cluster have an understanding of the underlying disk structure. An important point that has to be considered with a symmetric file system is the overhead (or lack therefor) that the metadata management places on the client nodes, serving both itself and other nodes. A non-efficient metadata implementation has a significant impact on the compute efficiency of the client nodes, and ultimately on the aggregate cluster performance. Some of the symmetric file systems in use today include Red Hat' GFS, IBM' GPFS, or HP' CFS.

5. The term *asymmetric file system* reflects a file system solution where in the cluster, there are one or more dedicated metadata managers, which maintain the structural elements of the file system and its associated disk structure. Some of the asymmetric file systems in use include IBM' SanFS, Cluster File Systems' Lustre, or some of the more traditional file system solutions such as NFS and CIFS, respectively.

6. The term *cluster file system* represents a distributed file system that does not have a single server with a set of clients (such as NFS), but instead consists of a cluster of server nodes that all collaborate to provide (high performance) service to the client base. To the client nodes in the cluster, this is all transparent, it is just a file system solution, but the file system software itself deals with distributing the requests to the elements that comprise the storage cluster. Some of the actual cluster file systems in use are HP' Tru64 cluster or Panasas' ActiveScale storage cluster file system.

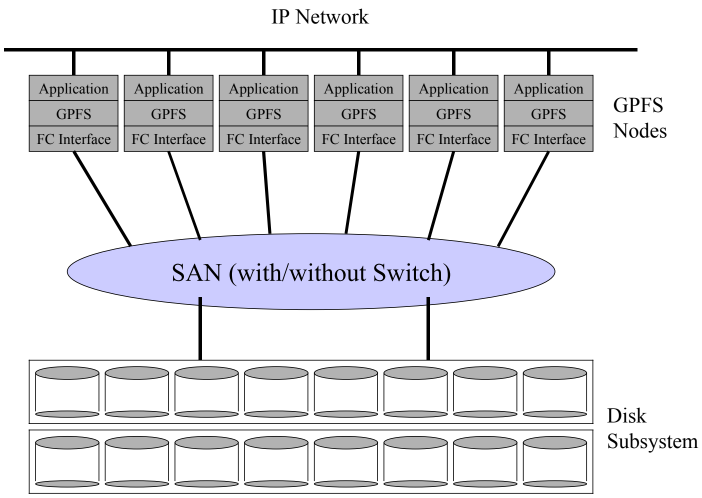
7. The term *parallel file system* describes a file system solution that supports parallel applications (such as MPI based scientific or life-science applications). In a parallel file system environment, all the nodes in the cluster may be accessing the same file (or files) at the same time, via concurrent read() and write() requests. Some parallel file systems in use today are IBM' GPFS, Cluster File Systems' Lustre, Panasas ActiveScale, or Red Hat' GFS.

It has to be pointed out that the terms discussed above overlap as well. To illustrate, a SAN file system may be symmetric or asymmetric, the environment may represent a parallel file system and/or a cluster file system based solution, respectively. As an example, Panasas' Storage Cluster and its ActiveScale File System represent a clustered, asymmetric, parallel, distributed file system solution. On the other hand, IBM's GPFS can be described as a parallel, symmetric file system solution that can be utilized in a SAN environment. There are of course many more file system solutions being used in a cluster environment today, but this article only focuses on some of the more popular once.

Introduction to IBM' GPFS

The General Parallel File System (GPFS) represents IBM's parallel file system solution. The actual file system is constructed from a collection of disks, which contain the file system data, a collection of nodes, which own and manage the file system, and a set of network connections (interconnects), which reflects the communication facility used among the nodes and the storage subsystem. In its simplest environment, the storage is connected to all the nodes by utilizing a SAN solution (see Figure 1). Figure 1 represents a Fibre Channel (FC) SAN. The nodes are connected to the storage via the SAN, and to each other by utilizing a LAN solution. Data that is being used by the applications is transferred over the SAN, whereas the control information used among the GPFS instances on the cluster is transferred via the LAN component.

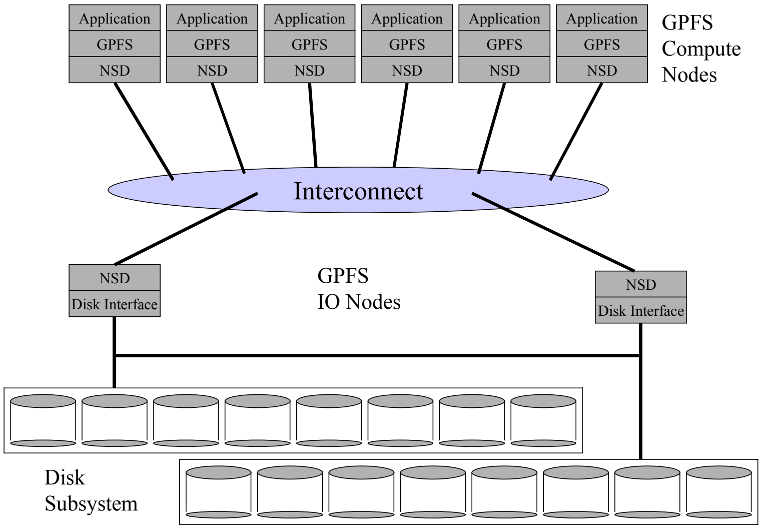
Figure 1: SAN Attached GPFS Cluster



In some cluster environments, where it is not feasible for every compute node to have direct access to the SAN, GPFS may be configured to utilize an (IBM provided) network block device capability. In an environment utilizing the IBM pSeries High Performance Switch (HPS), the capability is provided by the IBM Virtual Shared Disk (VSD) component, which is an AIX resource. In a Linux or AIX setup that utilizes any other interconnect infrastructure, GPFS uses the Network Shared Disk (NSD) capability, an actual component of GPFS. Both, VSD and NSD provide a software simulation of a SAN across either IP or an IBM proprietary network. GPFS uses NSD (and VSD if a HPS is present) to provide high-speed access to data for applications running on compute nodes that do not have a SAN attachment. Data is served to those compute nodes from a VSD or an NSD I/O server, respectively. Multiple I/O servers for each disk are feasible (actually recommended) to avoid single point of failure situations. Figure 2 reflects a GPFS Linux setup. In such a configuration, data and control information are transferred across an un-switched LAN. Such a model is only appropriate for smaller GPFS clusters. Switched LAN environments and/or bonded LAN (such as bonded Gigabit Ethernet) setups are recommended for clusters that require significant data transfer rates. Naturally, higher performance interconnects such as the IBM HPS or InfiniBand provide even better performance (if the actual workload can be driven high enough to utilize the capacity of the interconnect).

In a SAN simulation model, a subset of the cluster nodes is defined as GPFS I/O servers. The GPFS disk subsystem is only attached to the I/O servers. The NSD or VSD subsystem is responsible for the abstraction of disk data blocks across an IP-based interconnect. In such a configuration, the fact that the I/O requests are remote is transparent to the application that is issuing the file system calls. In Figure 2, a set of compute nodes is connected to a set of GPFS I/O servers via a high-speed interconnect. Each I/O server is attached to a portion of the disk subsystem. The recommendation made is that the disk subsystem is multi-tailed to the I/O servers for fail-over purposes. The choice of how many nodes to configure as I/O servers is based on the individual performance requirements and the capabilities of the storage subsystem, respectively. It has to be pointed out that in this model of storage attachment, it is not required to setup a complex SAN environment. Each storage solution is only attached to a small number of I/O servers (normally two), eliminating the necessity for (expensive) SAN switches.

Figure 2: NSD Based GPFS Cluster



The choice between a SAN attached or a SAN simulation model is performance and cost centric. The SAN simulation capabilities of GPFS provide economical solutions. In general, SAN subsystems provide the highest performance, but the cost and management complexity has to be considered when choosing such an approach. GPFS provides file data access from all the nodes in the cluster by providing a global name space for files. Applications can efficiently access files using standard UNIX file system interfaces, where GPFS supplies the data to any location in the cluster using the provided path to the storage. GPFS allows all the compute nodes that have the GPFS file system mounted to have coherent, as well as concurrent access to all of the storage (including write() sharing) resources. Some of the GPFS performance advantages are accomplished by (1) striping data across multiple disks that are attached to multiple nodes. (2) An efficient client side caching mechanism that includes read-ahead and write-behind techniques. (3) Utilizing a configurable block size, a feature that is especially important with some of the newer disk technologies where very large block sizes are paramount to aggregate storage performance. (4) Using block-level locking techniques that are based on a very sophisticated token management system that provides data consistency while allowing multiple application nodes to have concurrent access to the files.

Summary and Conclusion

This article introduced some of the terminology and features used in the world of cluster file systems. It has to be emphasized that the key components to be considered while planning for a cluster installation (and therefore choosing a cluster file system) are (1) scalability, (2) high performance, and (3) robust fail-over techniques. These features are definitely playing an increasingly important role in high performance cluster deployments, and not every cluster file system excels in all three areas. A sequel to this article will discuss and compare these three key components, focusing on the GPFS, GFS, and Lustre file systems, respectively.

References

1. Cluster File Systems, “*About the Lustre Architecture*”, Lustre File System, 2005
2. IBM Corporation, “*GPFS Primer for AIX Clusters*”, Boulder, 2005
3. OSDL, “*Cluster file system taxonomy*”, 2005
4. Red Hat, “*Red Hat Global File System*”, 2005
5. Welch Brent, “*What is a Cluster File System*”, 2005