General Parallel File System (GPFS)

The General Parallel File System (GPFS) is a high-performance clustered file system developed by IBM. It can be deployed in shared-disk or shared-nothing distributed parallel modes. It is used by many of the world's largest commercial companies, as well as some of the supercomputers on the Top 500 List. For example, GPFS was the filesystem of the ASC Purple Supercomputer which was composed of more than 12,000 processors and has 2 petabytes of total disk storage spanning more than 11,000 disks.

In common with typical cluster filesystems, GPFS provides concurrent high-speed file access to applications executing on multiple nodes of clusters. It can be used with AIX 5L clusters, Linux clusters, on Microsoft Windows Server, or a heterogeneous cluster of AIX, Linux and Windows nodes. In addition to providing filesystem storage capabilities, GPFS provides tools for management and administration of the GPFS cluster and allows for shared access to file systems from remote GPFS clusters.

GPFS has been available on IBM's AIX since 1998, on Linux since 2001, and on Windows Server since 2008, and it is offered as part of the IBM System Cluster 1350. GPFS 3.5 introduced Active File Management to enable asynchronous access and control of local and remote files, thus allowing for global file collaboration. The most recent version GPFS 4.1 introduces encryption. IBM also sells GPFS as IBM Spectrum Scale, a branding for Software-Defined Storage (SDS).

> GPFS is Scalable, highly-available, high performance file system optimized for multi-petabyte storage management

Explosions of data, transactions, and digitally-aware devices are straining IT infrastructure and operations, while storage costs and user expectations are increasing. The IBM General Parallel File System™ (GPFS™), high-performance enterprise file management, can help you move beyond simply adding storage to optimizing data management. High-performance enterprise file management using GPFS gives your business:

- Seamless capacity expansion to handle the explosive growth of digital information and improve efficiency through enterprise wide, interdepartmental information sharing
- High reliability/availability to eliminate production outages and provide disruption-free maintenance and capacity upgrades
- Performance to satisfy the most demanding applications
- Policy-driven automation to ease information life cycle management
- Extensible management and monitoring infrastructure to simplify file system administration
- Cost-effective disaster recovery and business continuity
- Optimize storage utilization, maximize return on investments

IBM General Parallel File System™ (GPFS™) currently powers many of the world's largest scientific supercomputers and commercial applications requiring high-speed access to large volumes of data such as:
1. Digital media
2. Engineering design
3. Business intelligence
4. Financial Analytics
5. Seismic data processing
6. Geographic information systems
7. Scalable file serving

GPFS provides online storage management, scalable access, and integrated information lifecycle management tools capable of managing petabytes of data and billions of files. Virtualizing your file storage space and allowing multiple systems and applications to share common pools of storage provides you the flexibility to transparently administer the infrastructure without disrupting applications, improving cost and energy efficiency, while reducing management overhead.

Massive namespace support, seamless capacity and performance scaling, along with proven reliability features and flexible architecture of GPFS helps your company foster innovation by simplifying your environment and streamlining data workflows for increased efficiency.

History

GPFS began as the Tiger Shark file system, a research project at IBM's Almaden Research Center as early as 1993. Shark was initially designed to support high throughput multimedia applications. This design turned out to be well suited to scientific computing.[3]

Another ancestor of GPFS is IBM's Vesta filesystem, developed as a research project at IBM's Thomas J. Watson Research Center between 1992-1995.[4] Vesta introduced the concept of file partitioning to accommodate the needs of parallel applications that run on high-performance multicomputers with parallel I/O subsystems. With partitioning, a file is not a sequence of bytes, but rather multiple disjoint sequences that may be accessed in parallel. The partitioning is such that it abstracts away the number and type of I/O nodes hosting the filesystem, and it allows a variety of logical partitioned views of files, regardless of the physical distribution of data within the I/O nodes. The disjoint sequences are arranged to correspond to individual processes of a parallel application, allowing for improved scalability.[5]

Vesta was commercialized as the PIOFS filesystem around 1994,[6] and was succeeded by GPFS around 1998.[7][8] The main difference between the older and newer filesystems was that GPFS replaced the specialized interface offered by Vesta/PIOFS with the standard Unix API: all the features to support high performance parallel I/O were hidden from users and implemented under the hood.[3][8] Today, GPFS is used by many of the top 500 supercomputers listed on the Top 500 Supercomputing Sites web site. Since inception GPFS has been successfully deployed for many commercial applications including digital media, grid analytics, and scalable file services.

In 2010 IBM previewed a version of GPFS that included a capability known as GPFS-SNC where SNC stands for Shared Nothing Cluster. This was officially released with GPFS 3.5 in December 2012, and is now known as GPFS-FPO [9] (File Placement Optimizer). This allows GPFS to use locally attached disks on a cluster of network connected servers rather than requiring dedicated servers with shared disks (e.g. using a SAN). GPFS-FPO is suitable for workloads with high data locality such as shared nothing database clusters like SAP HANA and DB2 DPF, and can be used as a HDFS-compatible filesystem.

Architecture

GPFS provides high performance by allowing data to be accessed over multiple computers at once. Most existing file systems are designed for a single server environment, and adding more file servers does not improve performance. GPFS provides higher input/output performance by "striping" blocks of data from individual files over multiple disks, and reading and writing these blocks in parallel. Other features provided by GPFS include high availability, support for heterogeneous clusters, disaster recovery, security, DMAPI, HSM and ILM.

According to (Schmuck and Haskin), a file that is written to the filesystem is broken up into blocks of a configured size, less than 1 megabyte each. These blocks are distributed across multiple filesystem nodes, so that a single file is fully distributed across the disk array. This results in high reading and writing speeds for a single file, as the combined bandwidth of the many physical drives is high. This makes the filesystem vulnerable to disk failures -any one disk failing would be enough to lose data. To prevent data loss, the filesystem nodes have RAID controllers — multiple copies of each block are written to the physical disks on the individual nodes. It is also possible to opt out of RAID-replicated blocks, and instead store two copies of each block on different filesystem nodes.

Other features of the filesystem include

Distributed metadata, including the directory tree. There is no single "directory controller" or "index server" in charge of the filesystem.
Efficient indexing of directory entries for very large directories. Many filesystems are limited to a small number of files in a single directory (often, 65536 or a similar small binary number). GPFS does not have such limits.
Distributed locking. This allows for full Posix filesystem semantics, including locking for exclusive file access.

Partition Aware. A failure of the network may partition the filesystem into two or more groups of nodes that can only see the nodes in their group. This can be detected through a heartbeat protocol, and when a partition occurs, the filesystem remains live for the largest partition formed. This offers a graceful degradation of the filesystem — some machines will remain working.
Filesystem maintenance can be performed online. Most of the filesystem maintenance chores (adding new disks, rebalancing data across disks) can be performed while the filesystem is live. This ensures the filesystem is available more often, so keeps the supercomputer cluster itself available for longer.
It is interesting to compare this with Hadoop's HDFS filesystem, which is designed to store similar or greater quantities of data on commodity hardware — that is, datacenters without RAID disks and a Storage Area Network (SAN).

- HDFS also breaks files up into blocks, and stores them on different filesystem nodes.
- HDFS does not expect reliable disks, so instead stores copies of the blocks on different nodes. The failure of a node containing a single copy of a block is a minor issue, dealt with by re-replicating another copy of the set of valid blocks, to bring the replication count back up to the desired number. In contrast, while GPFS supports recovery from a lost node, it is a more serious event, one that may include a higher risk of data being (temporarily) lost.
- GPFS supports full Posix filesystem semantics. HDFS and GFS do not support full Posix compliance.
- GPFS distributes its directory indices and other metadata across the filesystem. Hadoop, in contrast, keeps this on the Primary and Secondary Namenodes, large servers which must store all index information in-RAM.
- GPFS breaks files up into small blocks. Hadoop HDFS likes blocks of 64 MB or more, as this reduces the storage requirements of the Namenode. Small blocks or many small files fill up a filesystem's indices fast, so limit the filesystem's size.

Other advantages of GPFS over HDFS.

> Firstly GPFS is highly scalable file system. we have client with 18PB file system.
>
> No single point of failure because of distributed of meta data in active/active configuration
>
> POSIX file system.
>
> Policy based data ingestion
>
> Enterprise class storage solution.

GPFS has been developed years nearly decades before Map/Reduce as invented as distributed computing paradigma. GPFS by itself has not Map/Reduce capability. As is mainly aimed at HPC and the storage nodes are distinct from the compute nodes.

Therefore Map/Reduce can be done with 3rd party software (mounting GPFS on all Hadoop nodes), but it would not be very effective as all data is far away. No data locality can be used. Caches are more or less useless, etc.

In 2009, GPFS was extended to work seamlessly with Hadoop as GPFS-Shared Nothing Cluster architecture, which is now available under the name of GPFS File Placement Optimizer (FPO). FPO allows complete control over the data placements for all replicas, if applications so desires. Of course, you can easily configure to match HDFS allocation. GPFS FPO is specifically designed for Hadoop environment and is replacement for HDFS. Test have shown GPFS file system blows HDFS out of water in performance.

Check out details at http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=%2Fcom.ibm.cluster.gpfs.v3r5.gpfs200.doc%2Fbl1adv_fposettings.htm (http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=%2Fcom.ibm.cluster.gpfs.v3r5.gpfs200.doc%2Fbl1adv_fposettings.htm)

HDFS vs. Other Storage

To compare HDFS to other technologies one must first ask the question, what is HDFS good at:

- Extreme low cost per byte
  HDFS uses commodity direct attached storage and shares the cost of the network & computers it runs on with the MapReduce / compute layers of the Hadoop stack. HDFS is open source software, so that if an organization chooses, it can be used with zero licensing and support costs. This cost advantage lets organizations store and process orders of magnitude more data per dollar than tradition SAN or NAS systems, which is the price point of many of these other systems. In big data deployments, the cost of storage often determines the viability of the system.
- Very high bandwidth to support MapReduce workloads
  HDFS can deliver data into the compute infrastructure at a huge data rate, which is often a requirement of big data workloads. HDFS can easily exceed 2 gigabits per second per computer into the map-reduce layer, on a very low cost shared network. Hadoop can go much faster on higher speed networks, but 10gigE, IB, SAN and other high-end technologies double the cost of a deployed cluster. These technologies are optional for HDFS. 2+ gigabits per second per computer may not sound like a lot, but this means that today's large Hadoop clusters can easily read/write more than a terabyte of data per second continuously to the MapReduce layer.
- Rock solid data reliability
  When deploying large distributed systems like Hadoop, the laws of probability are not on your side. Things will break every day, often in new and creative ways. Devices will fail and data will be lost or subtly mutated. The design of HDFS is focused on taming this beast. It was designed from the ground up to correctly store and deliver data while under constant assault from the gremlins that huge scale out unleashes in your data center. And it does this in software, again at low cost. Smart design is the easy part; the difficult part is hardening a system in real use cases. The only way you can prove a system is reliable is to run it for years against a variety of production applications at full scale. Hadoop has been proven in thousands of different use cases and cluster sizes, from startups to Internet giants and governments.

How does the HDFS competition stack up?

This is an article about Hadoop, so I'm not going to call out the other systems by name, but I assert that all of the systems listed in the "8 ways" article don't compare well to Hadoop in one of the above dimensions. Let me list some of the failure modes:

- System not designed for Hadoop's scale
  Many systems simply don't work at Hadoop scale. They haven't been designed or proven to work with very large data or many commodity nodes. They often will not scale up to petabytes of data or thousands of nodes. If you have a small use-case and value other attributes, such as integration with existing apps in your enterprise, maybe this is a good trade-off, but something that works well in a 10 node test system may fail utterly as your system scales up. Other systems don't scale operationally or rely on non-scalable hardware. Traditional NAS storage is a simple example of this problem. A NAS can replace Hadoop in a small cluster. But as the cluster scales up, cost and bandwidth issues come to the fore.
  System that don't use commodity hardware or open source software
  Many proprietary software / non-commodity hardware solutions are well tested and great at what they were designed to do. But, these solutions cost more than free software on commodity hardware. For small projects, this may be ok, but most activities have a finite budget and a system that allows much more data to be stored and used at the same cost often becomes the obvious choice. The disruptive cost advantage of Hadoop & HDFS is fundamental to the current success and growing popularity of the platform. Many Hadoop competitors simply don't offer the same cost advantage. Vendor price lists speak for themselves in this area (where the prices are even published).
- Not designed for MapReduce's I/O patterns
  Many of these systems are not designed from the ground up for Hadoop's big sequential scans & writes. Sometimes the limitation is in hardware. Sometimes it is in software. Systems that don't organize their data for large reads cannot keep up with MapReduce's data rates. Many databases and NoSql stores are simply not optimized for pumping data into MapReduce.
- Unproven technology
  Hadoop is interesting because it is used in production at extreme scale in the most demanding big data use cases in the world. As a result thousands of issues have been identified and fixed. This represents several hundred person-centuries of software development investment. It is easy to design a novel alternative system. A paper, a prototype or even a history of success in a related domain or a small set of use cases does not prove that a system is ready to

take on Hadoop. Tellingly, along with listing some new and interesting systems, the “8 ways” article says goodbye to some systems that have previously been considered HDFS contenders by vocal advocates. I’ve got a rolodex full of folks who used to work on such systems who are now major players in the Apache Hadoop community.

It is easy to find example use cases where some other storage system is a better choice than Hadoop. But I assert that HDFS is the best system available today to do exactly what it was built for, being Hadoop’s storage system. It delivers rock solid data reliability and very high sequential read/write bandwidth, at the lowest cost possible. As a result, HDFS is, and I predict it will remain THE storage infrastructure for the vast majority of Hadoop clusters.