

## UKUUG Linux Technical Conference 2004

*Jos van Wezel  
Forschungszentrum Karlsruhe, Germany*

### **Storage area networking and parallel file systems**

When the GridKa centre in Karlsruhe (<http://www.gridka.de>) was given the task in 2001 to build a computer cluster it most naturally chose Linux as the major operating system. The benefits are clear. Very good support on common hardware, source code available for those necessary adaptations and supported by the communities GridKa needs to serve. The major task ahead is the construction of a 1000 to 1500 node dual processor cluster including 1 PB on-line disk and 16 PB tape storage capacity. Storage sizes like these were not uncommon in the supercomputer world but can Linux handle thousands of disks, control millions of GB size files and deliver a bandwidth of over 2 – 5 GB/s. The architecture of the solution at GridKa builds around a SAN and a parallel file system. No doubt it will adapt over time to follow technology trends.

Most of the configuration, when building a cluster, is similar to what is done on a single machine. Setting the IP address and changing some startup environment are basically the same on all Linux machines. No specialized knowledge is required. This is different for large installations that implement a cluster file system. The software glue between all the cluster nodes is the cluster file system. Highly optimized for the cluster or large installation, a cluster file is not contained in the standard Linux distributions. The choice for a certain cluster file system depends on the file access pattern and the costs. There are many packages and products that implement a file system and they share similarities or have completely different approaches. What follows is a selection of cluster file system products that run on Linux.

#### **A file system is necessary**

Clusters need on-line storage that is available to every node. Programs in a cluster usually work on partitioned work areas and need to share data through a common resource like a file system. For programs running in a cluster it is most convenient to use the file system interface offered by the underlying operating system. This guarantees portability and program code stability. It is very difficult to build the storage sharing into the application.

A file system to store input and output data is normally part of each operating system and applications are programmed with this in mind. The individual Linux computers in a cluster have ext, JFS, Reiser or equivalent installed but this storage is confined to the single node. Applications that are started to run on one node may run on another the next time. They still need to see the same data. Secondly there are classes of highly parallel applications that need parallel access to the same data from many nodes at the same time. Therefore in a cluster, data needs to be available to every cluster node and access to the data has to follow the well known UNIX paradigm through `open(2)`, `read(2)`, `write(2)` and `close(2)` calls.

This is what a cluster file system offers. It gives cluster nodes global access, meaning ‘visible at every node’, as well as multiple simultaneous access to on-line storage. As such, the names “cluster file system” and “parallel file system” are interchangeable.

### **IO in blocks and in files**

Files are written by the operating system in chunks of a certain size called blocks. The Linux block size is tunable to maximal 4 kB. The file system (ext, Reiser, XFS, JFS etc) is created by the user or by the Linux installation program with a certain block size. The block size determines the size of the transfers from and to disks. Consequently small changes to a file are handled less efficiently on a file system with 4 kB blocks size. Such a file system layout is better in handling large sequential IO. The kernel operates on blocks to improve speed and does so also for the IO subsystem. Disk drivers take care of the block transfers from host to disk. The file system takes care of the allocation and administration of disk blocks.

File system data access can be block oriented or file oriented. In the landscape of cluster file systems there are solutions that use and extend the local storage and systems that implement the actual block storage themselves. The first method makes the implementation and portability a lot easier but limits throughput because it depends on the comparably heavy files system layer. The other approach is to handle the block storage and file system administration within the cluster file system. This allows optimization of data access and the addition of features for which normally a Logical Volume Manager is responsible. The system is also easier to manage because it has full data control for the complete path from disk to application.

To improve on throughput many solutions offer the possibility to read and write data in parallel to many disks. Files are striped over several disk platters or alternatively, individual files are written to different disks. The maximum IO throughput is limited by the disk transfer speed capacity and can only increase when more disks are accessed in parallel. Connections to many disks are possible via a storage area network. The data blocks travel over a dedicated fabric which is optimized for IO due to its low latency.

Alternatively one has the possibility to access the local disks on other cluster nodes. An individual node can use the network connection since an Ethernet link is already available to connect to the outside world or uses a dedicated secondary Ethernet connection for disk data transfers. The Enhanced Network Block Device (<http://enbd.sourceforge.net>) is available for Linux that allows to attach a disk over the network. But IP over Ethernet is not optimal for block oriented IO and many file systems support the use of special inter-node connecting hardware to allow them to really demonstrate their superior throughput. A dedicated network of special hardware and protocols for block IO is rather expensive. The difference in speed is reflected in the costs for interconnect hardware. Examples of this type of, also called memory attached, hardware are Myrinet, Infiniband or Fibre Channel [5]. Transporting the blocks piggy back over Ethernet is the cheapest but slowest solution.

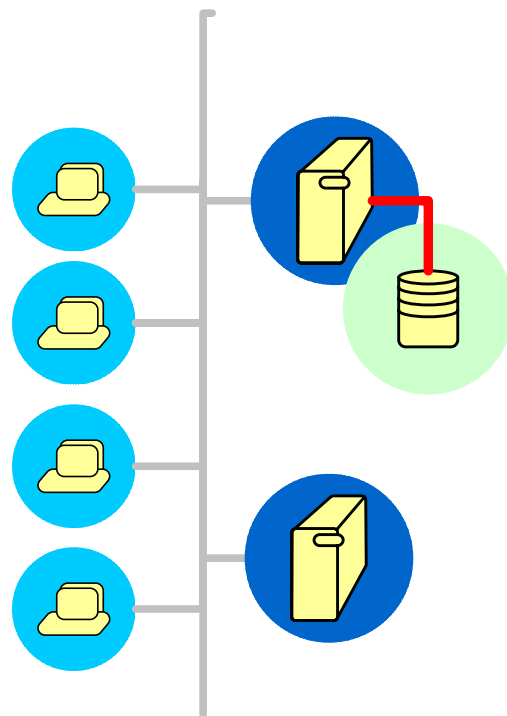
## Data about data

A recurring topic in cluster file systems is the handling of meta-data. Linux file systems store data in files in hierarchical directory trees. An information record is kept about each file and directory in I-nodes. Examples of this information are the size, creation time, type or the owner of a file or directory. This information about the data is called meta-data. Manipulations to the meta-data can be handled separately from the actual data input and output. Once the information about the data is known the data itself can be stored independently. This makes it possible to offload the meta-data handling to a dedicated meta-data server which can be optimized for the small and frequent updates. In cluster file systems the separate meta-data servers are usually also responsible for write locks on a file.

A cluster file system uses the separation to improve throughput. Read operations do not need the meta-data after the location and access rights of the content data are established. The system then directs the data transfers to a dedicated IO server. The content data can flow to possibly different servers optimized for handling large streams of continuous blocks. An IO server with high speed connections can deliver the actual data to the application more efficiently and the meta-data server can continue with other tasks. Because meta-data is many times smaller than the actual data, it can usually be completely cached. Remember that the working set in a cluster can become very large and file server caches are never large enough to store all active IO data. Having at least the meta-data cached improves the response time of the file system a lot.

## Network, direct and SAN attached storage

Well known types of file systems that can be used in a cluster are network file systems such as NFS or SMB. Especially NFS is used in many installations and mostly to connect to Network Attached Storage (NAS) servers. High throughput rates are achieved on dedicated NAS servers. Take a dual CPU, a RAID controller on a PCI card, 1 GB Ethernet interface if not on-board already, put in some SATA disks, install Linux as NFS server and your NAS box is ready.



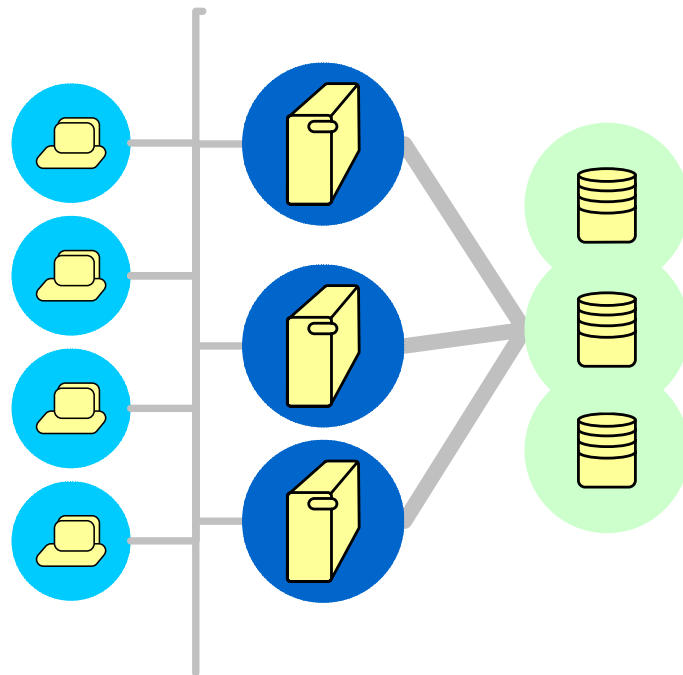
**Figure 1: NAS: storage via the IP netv**

Throughput for small files and non sequential access is rather slow because of the high network latency. Depending on the requirements regarding throughput, costs and to a lesser extend security, a network attached storage system can function very well as cluster file system. Commercial products exist that offer many interfaces and many TBytes capacity in one box.

Usually the physical storage is located next to the computer in the same housing or in near vicinity. This kind is therefore called 'direct attached storage'. The distance from computer to disk is limited by the electrical specifications of the copper based connections. Fibre Channel<sup>1</sup> has opened the possibility to connect disks and other block based storage devices at larger distance. Switches are used to connect hosts to storage devices. Switches can connect to switches to build a storage area network or

SAN. Direct attached storage connected via ATA, SCSI or Fibre Channel (FC) is used by cluster file systems to obtain a high throughput. Where SCSI or ATA is limited to one-to-one connections between host and storage, Fibre Channel is used to build switched fabrics that connect hundreds of hosts and storage devices.

The FC protocol is optimized for storage devices and has a very high throughput<sup>2</sup>, low latency and guarantees reliable end to end connections. A characteristic feature of FC host adapters is the protocol offloading which greatly reduces the interrupt load on the host when compared to Ethernet.



**Figure 2 Servers share disks via Fibre Channel in a**

### Cluster file systems and SAN

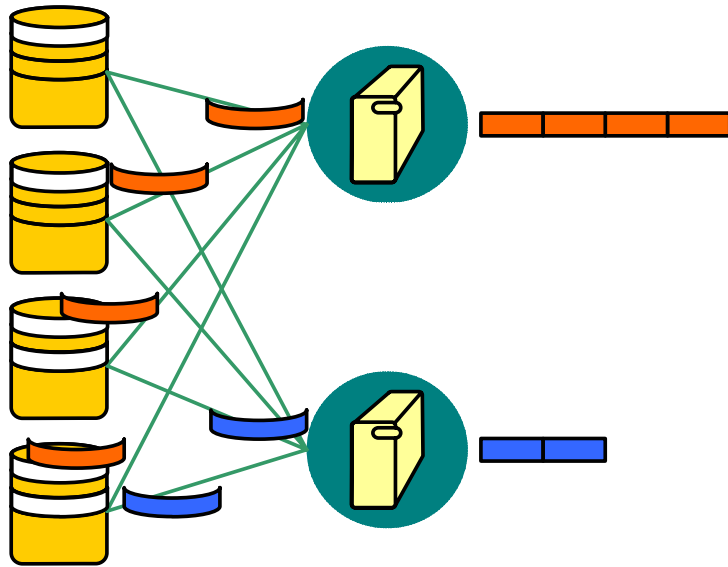
As said a cluster file system offers global simultaneous access to all nodes in the cluster. Nodes are connected to the file system via the Ethernet connection or via Fibre Channel. The amount of nodes is far too large to expect a reasonable IO throughput if no special provisions are made. The cluster file systems have the ability to spread the data over an arbitrary number of disks. This can be done block wise, in which case the file system blocks are distributed over the storage devices. The other variant is to write subsequent files to different file servers, thus spreading the load over several machines. Both methods increase the throughput because data transfers are done in parallel. The Lustre and SAN FS file systems use a combination. The General Parallel File System (GPFS) only distributes the blocks.

<sup>1</sup> Contrary to what the name suggest the Fibre Channel standard also defines a copper link.

<sup>2</sup> Fibre Channel speeds are currently at 2 Gb/s. Products for 4 Gb/s are expected for this year.

## SAN features

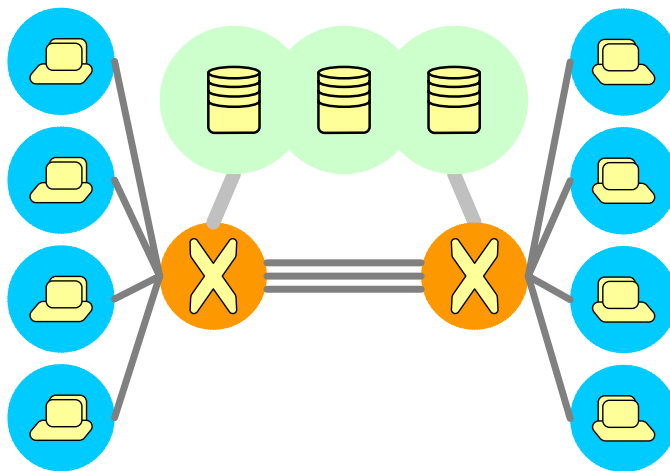
The benefit of a SAN lies in the fact that all cluster nodes are able to address all blocks of all attached disks. In the switched SAN each node can access all blocks independently which allows for some very impressive throughput because the nodes can operate in parallel. The file system takes care of addressing and administration.



**Figure 3 Parallel data access via a SAN**

The general addressability of disks in a SAN is one of the attractive features. Another is the fact that without interruption storage devices can be added or removed. In large installations this is a continuous process and the connection to servers and clients is never severed.

All mentioned file systems implement a logical volume manager (LVM), a piece of software that sits between the file system and the SAN disks. It handles the physical to logical mapping of the storage and manages the addressing at the device level. The LVM make it possible that new SAN devices are made available to the file system and can aggregate single devices into groups.



**Figure 4 Best of both. Combination of SAN and Infiniband**

performance and one needs only a single network for both IO and for inter-cluster communication.

A alternative way of accessing or starting a SAN is via an Infiniband switch. Infiniband consists of protocols and hardware to connect computers with high speed serial interconnects. An Infiniband switch moves data non-blocking at 4 GB per port with very low latency. Expansion cards connect the switch to Fibre channel thus integrating Infiniband with SAN. The existing cluster interconnects can then be used for IO too. The combination delivers a very good

## **A selection of cluster file systems**

The number of cluster parallel file systems that can make good use of SAN to increase throughput is limited. Two products are marketed by IBM: GPFS and SAN FS. There are three open source licensed products: Lustre, more or less a product of HP, GFS which is now a product of RedHat and the OpenGFS package. Some of the mentioned products are highlighted below.

### **GPFS**

The General Parallel File System is product of IBM and has a long and robust history in the cluster computer world and is since 2 years, also available on Linux. GPFS is a truly parallel file system. Data can be striped over many disks and any node can access the same file at the same time. On Linux GPFS uses several kernel modules and runs a multi-threaded daemon. Access to the data can be through direct attached storage and an IP network or via a storage area network that is shared among the GPFS nodes.

There are two possible access configurations, either via SAN or via direct attached storage. In the SAN configuration each node sees each block on all storage elements that are made available to the GPFS. The disks are shared in the SAN. Files are assembled from the blocks distributed in the SAN and are directly available on the node. The direct attached configuration relies on a high speed inter-node network which can be Ethernet or Myrinet. Blocks from local disks are shipped to other nodes over the high speed link. Files are assembled by gathering blocks over the IP network from local disks on several nodes.

Management of GPFS is very simple. Commands resemble common Unix commands and can be issued from any node. The system has the capability of adding and removing disks, rebalance data access after adding or removing disks, resize file systems, change the block size, change the number of possible I-nodes and more. It can do this on-line while the system is active. GPFS provides mirroring of metadata and data separately and allows the mirroring to be located on separate physical drives. GPFS also overcomes the maximum file system size limitation of Linux because it allows a configurable block size as large as 1 MB. Currently file systems of 18 TB are supported.

There is one node per open file for handling the metadata. All nodes can access the same file but changes to the meta-data are handled by the responsible meta-data node. All nodes can be meta-data nodes and GPFS allows access from any node in the cluster to the same file while maintaining the proper locks. The locking is distributed over the nodes accessing the file. All data is written and read in parallel and throughput scales linearly with the number of disks and nodes. Although GPFS is usually installed on the compute nodes in a cluster, GPFS file systems can be exported with NFS from dedicated servers. The compute nodes then mount the exported file systems. .

## LUSTRE

Brand new and actively developed is the LUSTRE file system. The name is a mix of the words cluster and Linux which exposes the objectives for this project. Although Lustre is marketed as product by HP the project is committed to the open source license model. Probably because Lustre is also sold as storage software product there is excellent documentation and background information available. For configuration and logging Lustre relies on the open standards LDAP and XML. Lustre is a file (object) based system.

Everything stored in Lustre is considered an object. The objects of the file system are (special) files and directories. The attributes, meta-data, of these objects such as size, creation time, symbolic link pointers or backup flags are stored on metadata servers (MDS). The meta-data is kept separate from the actual content. The MDS takes care of file creation, attribute manipulation and is responsible for the namespace handling: a file is found by asking the MDS. After opening the MDS relays the actual the IO to Object Storage Targets (OST) to takes care of the data exchange. The MDS keeps track of the data exchange in a journal. Creating and writing a file involves the creation of an I-node on the MDS which then contacts an OST to allocate storage. The allocation can be striped across several OSTs to enhance performance. An OST is implemented on common Linux file systems and drivers exist for ext3, JFS, Reiser and XFS.

Throughput achieved in some published tests is really impressive. At the moment Lustre still lack important maintenance tools to use it in a production environment. There is no file system recovery utility and there is not yet an automatic failover for the single MDS. The original approach and the development from the ground up makes Lustre a solution that could develop into a very powerful and elegant system to store files in a cluster.

## OpenGFS and GFS

OpenGFS like GPFS, also implements a journaled block based file system that provides read and write access from multiple nodes. Last year the dreaded 'pool' code was changed to allow OpenGFS (OGFS) to use any logical volume manager. ELVM is preferred. Most recently the memexp locking was replaced by the OpenDLM module. The old memexp was a single point of failure and very compute intensive. OGFS supports growing of file systems and the addition of disks (through the separate LVM). Node failures are handled by log recovery and fencing off the failed node. File systems of OGFS can be exported via AFS or NFS but are limited in size due to the Linux kernel constraints.

The recent development of OGFS makes is attractive again to consider it for use in a cluster. It has fewer features and is lacking the recovery and redundancy features of GPFS but it is free and one can certainly give it a try. Install OGFS for example to distribute read-only data trees.

The ancestor of OGFS is the GFS that is now developed and marketed by RedHat. It has similar features but is more mature and robust. RedHat has a suite of software for clusters that includes the GFS file system. See <http://sources.redhat.com/cluster>.

## Select and install

After you have selected your favored file system it is not possible to run `rpm -i` and forget about it. The software has to be tuned for the specific environment it runs. All mentioned systems offer a lot that can be customized and it can only be stressed: use this possibility. The default values are never the optimal values for your environment. Proper configuration forces you to think about data access patterns, optimal IO paths, possible bottlenecks, block sizes, strip sizes, caches usage etc. After installation of the cluster file system the fun just starts.

"They're really close and they're almost interoperable but they're just different enough to be a pain in the butt," Dr. James Gosling, CTO of Sun's Developer Products group about incompatibilities of Linux and other UNIX distributions.

## Links

1. ENBD: <http://enbd.sourceforge.net>
2. GPFS: [http://www.almaden.ibm.com/StorageSystems/file\\_systems/GPFS](http://www.almaden.ibm.com/StorageSystems/file_systems/GPFS)
3. OpenGFS: <http://www.sourceforge.net/projects/opengfs>
4. Lustre: <http://www.lustre.org>
5. Fibre Channel protocol: <http://www.fibrechannel.org>
6. The NFS protocol is standardized in RFC1094 (version 2), RFC1813 (version 3) and RFC3530 (version 4)