Evaluation of Distributed File Systems

Tim-Daniel Jacobi, Jan Lingemann
Department Informatik,
Universität Hamburg

9jacobi@informatik.uni-hamburg.de,
9lingema@informatik.uni-hamburg.de

October 22, 2012

Contents

1. Introduction

Just like any other field of research climate research constantly develops towards a more specific and exact process of gathering data and calculating results. The tools for data gathering getting more precise and furthermore the process itself gets refined constantly. While this on one hand means that we can expect better understanding of how the worlds climate system works it also implies a constant growth of the amount of data to handle. Sooner or later the question will arise how all this data can be stored and managed advantageously in order to not only be able to use it for research whilst maintaining decent processing times but also use the space to store the data efficiently without wasting resources.

1.1. Objectives

In order to be able store the aforementioned amounts of data a file system is needed that can handle not only the data but also comes up with some sort of communication solution to have it running on the computer cluster. In this paper our research group will examine the possibilities of storing, managing and working with large data amounts with in a computer cluster.

1.2. Problem definition

The department of scientific computing at the University of Hamburg which is located at the DKRZ (Deutsches Klimarechenzentrum, German climate data processing centre) conducts research in the aforementioned areas. They instructed our student research group to asses, setup and benchmark various parallel distributed file systems in order to come up with at least one to run on the departments cluster.

1.2.1. Initial description

A parallel machine is a machine which is able execute applications in **real** parallelism. Current Desktop machines are little parallel machines already as they have multiple cores which allow for parallel application execution. Their high market share is the main reason why the development of parallel programs and algorithms will gain momentum in the future. Computer clusters combine multiple machines through fast networking to be able to solve larger scaled problems. Currently existing super computers easily size up to 100.000 cores. An important aspect when dealing with super computers is the efficient storage of large data amounts.

1.3. Structure of this paper

The paper starts with the problem definition you have just read. Subsequently it exposes some fundamental definitions to prepare the reader with the basics on the treated topic. Afterwards we give an overview of the market to gain an insight of what is available. After we stated the selection we will dive into the installation and benchmarking of the selected file systems. Therefore we explain the environment and software we used for the benchmarks. Following we analyse the results to finally come up with a conclusion.

2. Fundamental definitions

2.1. File system

A file system organises data expected to be saved after a program terminates by providing methods to store, retrieve and update data as well as manage the available space on the device which contains it. A file system organises data in an efficient way. A tight coupling usually exists between the operating system and the file system to extend performance. Some file systems provide mechanisms to control access to the data and metadata. Some file systems allow multiple programs to update the same file at nearly the same time.

2.2. Computer Cluster

A computer cluster consists of a set of connected computers that work together so that they can be viewed as a single system. The components of a cluster are usually connected to each other through fast local area networks like Ethernet or Infiniband where each node runs its own instance of an operating system. Clusters are usually deployed to improve performance and availability over that of a single computer, while typically being much more cost-effective than single computers of comparable speed or availability. [1]

2.3. Cluster file system

A clustered file system is a file system which is shared by being simultaneously mounted on multiple servers. There are several approaches to clustering, but most of them do not use a clustered file system. This lead to an increase of the underlying storage environment as nodes are added to the cluster. Thus it is highly recommended to use a cluster filesystem in cluster.

[1] `http://en.wikipedia.org/wiki/Computer_cluster`

2.3.1. Shared disk file system

One storage is accessed by many clients on block level

- Block Level Access
- OCFS2, GFS2, Xsan
  - Many nodes access one storage (e.g. SAN over FibreChanel)
  - Multiple Access managed by file system with shared metadata (e.g. locks, rights)

2.3.2. Distributed file system

One storage is accessed by many clients on file

- the filesystem itself is clustered
  - the data is split to several nodes
    - usually storage and meta data
- the access is clustered as well across the storage nodes
- furthermore the data is striped to increase performance

2.3.3. Distributed parallel file system

- like distributed file systems
- provide more features including
  - fast local access
- symmetric vs. asymmetric

3. Market analysis

3.1. Requirements

In order to meet the requirements of the department a suitable filesystem should be free of charge, or better licensed under GPL. Furthermore it should be future-proof, thus it should be in active development and have a long-term development plan. Since the nodes mainly run on the LTS versions of Ubuntu it would be advantageous to use a filesystem which runs under Ubuntu. Yet other file systems should be considered if their features seem to be promising.

3.2. Market overview

3.2.1. GFS (Global File System)

Global File System 2 is a shared disk file system for computer clusters based on Linux. GFS2 differs from distributed file systems because it allows all nodes to have direct and concurrent access to the same block storage. GFS2 can also be used as a local running filesystem. GFS has no disconnected operating-mode, and does not distinct between client and server. All nodes function as peers. Using GFS in a cluster requires hardware to allow access to the shared storage, and a lock manager to control access to the storage. The lock manager operates as a separate module: thus GFS2 can use the Distributed Lock Manager (DLM) for cluster configurations and the "nolock" lock manager for local filesystems. GFS2 is free software, distributed under the terms of the GNU General Public License.

3.2.2. OCFS2 (Oracle Cluster File System)

OCFS2 is a general-purpose shared-disk cluster file system for Linux capable of providing both high performance and high availability. As it provides local file system semantics, it can be used with almost all applications. Cluster-aware applications can make use of cache-coherent parallel I/Os from multiple nodes to scale out applications easily. Other applications can make use of the file system facilities to fail-over running application in the event of a node failure. The file system is currently being used in virtualisation (Oracle VM) in both the management domain, to host virtual machine images, and in the guest domain, to allow Linux guests to share a file system. It is also being used in database clusters (Oracle RAC), middleware clusters (Oracle E-Business Suite), appliances (SAP's Business Intelligence Accelerator), etc.

3.2.3. FHGFS (Fraunhofer Gesellschaft File System)

FraunhoferFS (short: FhGFS) is the high-performance parallel file system from the Fraunhofer Competence Centre for High Performance Computing. The distributed metadata architecture of FhGFS has been designed to provide the scalability and flexibility that is required to run today's most demanding HPC applications. FhGFS is provided free of charge.

FhGFS stripes file contents across multiple storage servers and distributes the file system metadata across multiple metadata servers. File system nodes can serve Infiniband and Ethernet (or any other TCP-enabled network) connections at the same time and automatically switch to a redundant connection path in case any of them fails. FhGFS requires no kernel patches (the client is a patch-less kernel module, the server components are user-space daemons), comes with graphical cluster installation tools and allows you to add more clients and servers to the running system at any given point of time. FHGFS differentiates between Clients and Servers which can be run on the same machine where it uses existing partitions, formatted with any of the standard Linux file systems, e.g. XFS or ext4.

3.2.4. PVFS2 (Parallel Virtual File System)

The Parallel Virtual File System is an open source parallel file system. PVFS is designed to provide high performance for parallel applications, where concurrent, large IO and

many file accesses are common. PVFS provides dynamic distribution of IO and metadata through avoidance of single points of data access. PVFS clients are stateless and thus allowing for failure recovery and integration with high-availability systems. PVFS is designed to support a number of access models. PVFS provides an object-based, stateless client interface, leading to optimisations for metadata operations within MPI-IO. It operates on a wide variety of systems, including IA32, IA64, Opteron, PowerPC, Alpha, and MIPS. It is easily integrated into local or shared storage configurations, and provides support for high-end networking fabrics, such as Infiniband or Myrinet.

3.2.5. GlusterFS

GlusterFS is an open source, distributed file system capable of scaling up to several petabytes and handling thousands of clients. GlusterFS clusters together storage building blocks over Infiniband RDMA or TCP/IP interconnect, aggregating disk and memory resources and managing data in a single global namespace. GlusterFS is based on a stackable user space design and can deliver exceptional performance for diverse workloads.

3.2.6. Lustre

Lustre is a parallel distributed file system, generally used for large scale cluster computing. Lustre is available under the GNU GPL and provides a high performance file system for computer clusters ranging in size from small workgroup clusters to large-scale, multi-site clusters. Lustre file systems are scalable and can support tens of thousands of client systems, tens of petabytes of storage, and hundreds of gigabytes per second of aggregate I/O throughput.

3.2.7. HAMMER

HAMMER is a high-availability 64-bit file system for DragonFly BSD. Its major features include infinite NFS-exportable snapshots, master-multislave operation, configurable history retention, fsckless-mount, and checksums to deal with data corruption. HAMMER also supports data block deduplication — identical data blocks will only be stored once on a file system.

3.2.8. Hadoop

Hadoop is a software library developed by Apache. The library is a framework that allows for the distributed processing of large data sets across clusters of computers using a simple programming model. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

3.3. Selection

After the research phase and debate on the filesystems with the department we nominated the following systems to be included in our further steps of examination.

- FHGFS
- GluserFS
- HAMMER
- Lustre

4. Installation

4.1. GlusterFS

GlusterFS is available for installation on RPM distributions and debian-based distribution as well as from source. It is possible to let GlusterFS run through InfiniBand. Yet it comes with some hardware, networking, operating system and file system requirements listed below.

4.1.1. Hardware requirements

- Processor: Intel/AMD x86 64-bit
- Disk: 8GB minimum using direct-attached-storage, RAID, Amazon EBS, and FC/Infiniband/iSCSI SAN disk backends using SATA/SAS/FC disks
- Memory: 1GB minimum

4.1.2. Networking requirements

The following are the supported networks:

- Gigabit Ethernet
- 10 Gigabit Ethernet
- InfiniBand

4.1.3. Operating system requirements

GlusterFS works with other common Linux distribution like CentOS 5.1 or higher, Ubuntu 8.04 or higher, and Fedora 11 or higher, but has not been tested extensively.

Ensure that the following packages are installed:

- Bison
- Automake/ Autoconf
- Flex
- libtool
- gcc
- Portmappper (for NFS)
- Fuse

4.1.4. File system requirements

Red Hat recommends XFS when formatting the disk sub-system. XFS supports metadata journaling, which facilitates quicker crash recovery. The XFS file system can also be defragmented and enlarged while mounted and active.

Any other POSIX compliant disk file system, such as Ext3, Ext4, ReiserFS may also work, but has not been tested widely.

4.2. Lustre

Lustre is developed for Red Hat Enterprise Linux. Since Cent OS 5.5 aims to provide a 100% binary compatible system with RHEL5 we can use it as well. Use the provided install shell-script to install Lustre on your nodes. The script will patch the kernel, install and activate it and setup a Lustre installation.

4.3. HAMMER

Fortunately HAMMER is standard filesystem of DragonFly BSD as of version 2.0. Thus to install HAMMER you only need to setup DragonFly BSD on your nodes.

4.4. FHGFS

No specific enterprise Linux distribution or other special environment is required to run FhGFS. FhGFS client and servers can even run on the same machine to enable performance increases for small clusters or networks. FhGFS requires no dedicated file system partition on the servers - It uses existing partitions, formatted with any of the standard Linux file systems, e.g. XFS or ext4. For larger networks, it is also possible to create several distinct FhGFS file system partitions with different configurations.

5. Bash-script development

5.1. Environment

5.1.1. Identify the actual environment

After we have obtained detailed information form the department of scientific computing on how the cluster works we concluded that there are three relevant topics in general. Structure, hardware and used software.

5.1.2. Structure

The structure describes how machine in the cluster are used. In this case there are distinct machines for keeping the storage and others for computational purposes. On top there is one machine which is called the master. This machine is responsible for the initiation of jobs and is accessible to users. In the scope of this paper we call it controller, due to historical work. Furthermore there are compute and storage nodes that are responsible for holding the data to store and perform computations on the data as their names suggest.

5.1.3. Hardware

The present machines in the cluster are all based on the x86_64 instruction set. The differences on the hardware side between the compute and the storage nodes are processing power, ram and disk space. Though none of these criteria is relevant for building the scripts.

5.1.4. Software

All machines run Ubuntu Server 12.04 LTS 64-bit as operation system. The cluster is managed by a variety of different applications. In first place there is SSH activated on every client for basic management. Moreover there is SLURM installed for batch processing and job distributions. Both SSH and SLURM can be used by some parallel distributed shell systems for remote management. A module manager is responsible for the script management.

The actual software system is in fact a little bit more complicated. Operating System images are rolled out and auto configured by a bootp-server which enables the user to use different operating systems.

5.2. Building the test environment

The test environment should be as close to the actual cluster to avoid compatibility errors. It should how ever also be easy to set up and to maintain. The whole test environment is build up on virtual machines on one machine. In this case we use Virtual Box to host a set of virtual machines and to maintain snapshots to be able to easily revert configuration mistakes.

5.2.1. Structure

The structure is pretty much the same as in the real cluster, but with less machines. So we have four storage nodes (nodes[]01-04]), two compute nodes (node[05-06]) and a master node called *controller*. On top of it there is a router based on for easier management of ip addresses and hostnames.

5.2.2. Hardware

Since the virtual host is a x86_64-bit machine, all virtual guests inherit this architecture. It is however recommended to have a processor with multiple cores, VT-x/AMD-V and Nested Paging. Also a lot of RAM is needed.

5.2.3. Software

We made any effort to achieve a foundational software system which is as similar as possible to the actual cluster. We used the same Ubuntu Server Version and tried to use the same software stack.

5.2.4. Desctiption

The /opt directory is shared via NFS from textitcontroller and mounted via auto-fs on the nodes. SLURM runs with different partitions. The NTP server runs on the *controller*, all nodes query PDSH as parallel distributed shell using SSH with server packages installed. All nodes posses the public key of the *controller* installed

We used a router for IP assignment and auto DNS configuration. In this case it is OpenWRT which includes DDNSmasq. This requires that the hostname is set correctly in the node and also sent to the DHCP server. In our first tests we had to tell Ubuntu 10.04 to do it. Ubuntu 12.04 is able to do it automatically.

5.3. Scripts

5.3.1. Requirements for the scripts

If it is planned to use the delivered scripts there are several requirements to take into account. They are listed as following.

- Select a proper hostname layout
  - e.g. xyz[000-099], io-node[01-06], compute-node [01-10], ...
- All hosts have to be resolvable, so the DNS structure has to be in a functional state
- Keyless SSH authentication via SSL certificates is needed, at least from controller to nodes
- The operating system has to be up to date
- Root has to be enabled and SSH accessible on all machines in the cluster

5.3.2. Structure of scripts

5.3.2.1. fs_master

This is the main script in which the SLURM commands can be run

5.3.2.2. fs_config

This script contains all the config stuff, some of these should be set by the user

5.3.2.3. fs_setup

This script installs the required packages and sets up the volume on the io-nodes, furthermore it mounts the volume on the compute-node and the *controller*

5.3.2.4. fs_cleanup

This script reverses all the steps of *fs_setup*

5.3.2.5. fs_execOn

This script contains the wrapper functions for the used parallel distributed shell

5.3.2.6. create_environment

Here are some useful chunks of code to create a managed cluster with Ubuntu 12.04, it is not a script you canshould execute but supposed to help you out with the aforementioned matters.

5.3.3. Details

The scripts are connected via the source command. Except for *fs_master* there are no commands which are not wrapped into a function. For the sake of readability and inter-

changeability we encapsulated the command for distributed parallel command execution. Also there is one function (es) to output the status on the command line.

5.3.4. fs_master

This master script is a sample script and looks a lot like a later user script. It allows you to use the function *fs_ setup* to indeed setup the distributed file system and *fs_ cleanup* to destroy it. As for the early release cycle, always use these both function in correct order and always use both.

5.3.5. fs_config

In this script there are only assignments to deliver the shared configuration to the other scripts. It is pretty straight forward. For next releases some of these configuration should be settable via the master script and the machines and their names should be auto discovered.

5.3.6. fs_setup

Here is all the logic to create a pool of potential io-nodes and to subsequently create a volume which later is mounted by the compute nodes. In the test environment only a subfolder on / is used for the storage pool. It is highly recommended to use a separate hard drive formatted with XFS for better peformance.

5.3.7. fs_cleanup

Reverses all the steps made in *fs_ setup*. This means it also destroys the Volume and the contained files. If later in *fs_ setu*p whole hard disk will be used, the partition table should be destroyed here also.

5.3.8. fs_execOn

Contains the wrapper scripts for PDSH. As described PDSH is interchangeable with SLURM or clush. In fact you can use PDSH with SLURM.

6. Testing

6.1. Software used

The tests shall be run with Parabench. Patterns for testing can be downloaded here fileop.pbl and mpiio.pbl are the ones to use. They are optimised by the department prior to testing to fit their specific needs. The test will include a variation on the memory of the nodes. To scale the amount of memory we will make use of the Memeater tool.

6.2. Test configuration

A test configuration is defined by three orthogonal parameters which include

- Amount of Memory
  - with values 1, 12 GB
- Number of server nodes
  - with values 1, 2, 5
- Number of client nodes
  - with values 1, 2, 4, 8 for 1 and 2 server nodes
  - with values 1, 2, 3, 4, 5, 10, 15 for 5 server nodes

6.3. Cluster environment

While conducting the product the cluster environment was changed by the department of scientific computing. As per request these changes needed to be reflected in the install scripts for the file systems.

6.4. Cluster benchmarking

Not conducted.

7. Post-project matters

7.1. Conslusion

The market of parallel distributed filesystem is very diverse. Each system comes up with an own idea of handling the problems aimed to solve by the department of scientific computing.

Due to the change in environment on the cluster to be used we decided to concentrate on the delivery on a script that currently installs one of the chosen filesystems but is easy to modify in order to be used with other filesystems. This way we were able to ensure the quality of the script delivered. Unfortunately we could not conduct any benchmarks due to the time consuming script modification needed which forced us to rethink the focus of the project which resulted in quality assurance of the script.

7.2. Further work

Since the result of this project provides a sophisticated foundation to conduct benchmarks on the cluster we recommend to use our achievements for further research.

7.3. Final words

During the project we had the chance to gain an insight of the work in a high performance computing department which is an comprehensive but yet interesting field. We were able to gain a lot of knowledge in a topic that we have never treated before. Parallel file systems are an exciting research topic even though they sometimes come with unresolved issues but a flourishing community which you can learn from.

A. Scripts

Please refer to the file attached to this document in order to obtain the scripts.

B. File system set up protocols

B.1. HAMMER

In order to install HAMMER the setup of DragonFly BSD on the desired machines is needed. DragonFly BSD uses HAMMER as its filesystem. Thus there is no need for a setup of the filesystem.

B.2. Lustre

Disclaimer

This guide was tested with Lustre version 1.8 under Cent OS 5.5. Other configurations may work but are not supported.

Basic description

Lustre is a parallel distributed file system often used in supercomputers due to its high performance abilities. It is highly scalable and is designed to deal even with a very large store in the PiB area and with a really high throughput. The actual data in Lustre is saved in objects that reside on node(s) which are very likely to be in a network.

Architecture

There are three different kinds of nodes in a Lustre configuration

- **MDS** - Metadata server
  - stores the namespaces metadata such as
    - filenames
    - directories
    - access permissions
    - file layout
  - on its **MDT** - metadata target
- **OSS** - object storage server
  - stores file data on one or more **OST** - object storage target
  - each OST manages one local filesystem
  - typically two to eight OSTs per OSS
  - capacity of Lustre file system is the sum of the capacity provided by the OSTs
- **Client(s)**
  - Accesses and uses the data using standard POSIX[1] semantics
  - allows concurrent and coherent read/write access

Principle of operation

The MDT, OST and client can be installed on the same node or on single nodes and then connected over a network. Their storage used for the file systems is partitioned and usually formatted as ext4 but using an enhanced version called *ldiskfs*.The OSS and MDS read, write and modify data which is held by theses nodes.
To address files Lustre uses inodes like normal unix file systems. These inodes are saved on the MDS. Other than in conventional unix file systems the inodes point to one ore more OST objects associated with a file on the OST rather than to data blocks.
Clients delegate the task of modifying a file to the OSS rather than modifying it directly on the OST. This ensures scalability in larger environments.

Supported network interconnects

[1] Portable Operating System Interface - provides an API to ensure software compatibility along different UNIX systems

- Infiniband
- TCP/IP over Ethernet (and other networks)
- Myrinet
- Quadrics
- etc.

Installation

Supported operating system, platform and interconnect

Configuration Component	Linux Platform	Architecture	Interconnect
Server	OEL[2] 5.4 RHEL[3] 5.4	x86_64	
Client	OEL 5.4 RHEL 5 SLES[4] 10,11 Scientific Linux 5 [New] Fedora 12 (2.6.31) [New]	x86_64 ia64 (RHEL) ppc64 (SLES) i686	
Server and Client			TCP/IP OFED

Since Cent OS 5.5 aims to provide a 100% binary compatible system with RHEL5 we can use it as well.

Use the provided install shell-script to install Lustre on your machine. The script will patch the kernel, install and activate it and setup a Lustre installation. Afterwards it cleans up the build folder to save space on the volume.

Configuration

As per request we will provide configuration scripts for two entities. OSS-MDS nodes combined acting as *I/O-nodes* and Lustre-Clients as *Client nodes*. On the I/O-nodes we need to partition the block device in two parts since OSS and MDS both require to run on a independent file system. Our script calculates the size of the block device and splits it evenly.

---

[2] Oracle Enterprise Linux

[3] Red Hat Enterprise Linux

[4] Suse Linux Enterprise Server

B.3. GlusterFS

Since GlusterFS was used as reference implementation of the scripts it is unnecessary to provide an additional installation guide.