

CHARACTERIZING THE GLUSTERFS DISTRIBUTED FILE SYSTEM FOR SOFTWARE DEFINED NETWORKS RESEARCH

BY MANONIT KUMAR

A thesis submitted to the
Graduate School—New Brunswick
Rutgers, The State University of New Jersey
in partial fulfillment of the requirements
for the degree of
Master of Science
Graduate Program in Electrical and Computer Engineering

Written under the direction of

Dr. Ivan Roderio

and approved by

New Brunswick, New Jersey

January, 2015

ABSTRACT OF THE THESIS

Characterizing the GlusterFS Distributed File System for Software Defined Networks Research

by Manonit Kumar

Thesis Director: Dr. Ivan Roderio

With the rapid rise of online resources such as those offered by Google, Dropbox and Microsoft cloud computing is playing a bigger part in everyday lives than most people can imagine. More and more services are being offered online even in academia and science terabytes of data is collected during experiments and this data has to be stored and provisioned across various geographical locations. This is achieved through distributed file systems and software defined networks is a new paradigm that provides interesting ways to improve these Distributed File systems. In order to understand the potential impact of SDN usage on distributed file systems this work has set out to characterize GlusterFS in certain configurations by emulating a realistic distributed load on the file system. Software Defined Networks as opposed to traditional networks are not static, in the sense that the system can manage data bandwidth and balance data loads according to requirements and cost. This work measures the performance in units of throughput in KB per second along with graphs of network activity from empirical executions to understand GlusterFS in more detail.

Experimental results of introducing SDN into GlusterFS yielded very good results. Significant gains in the range of 45% - 60% in the quality of service were observed during manual manipulation of the network.

Acknowledgements

First and foremost, I would like to thank my advisor Dr. Ivan Rodero for giving me the opportunity to be part of this project. His continuous support and key insights gave shape and direction to this project. The project required a lot background work to setup experiments and Dr.Rodero always pointed me in the correct direction to gain knowledge towards them.

Special thanks to Claris Castillo, Senior Computational and Networked Systems Researcher in the Renaissance Computing Institute (RENCI) at UNC-Chapel Hill, she helped me understand the network testbed ie. ExoGENI along with setting up the experiments. Claris helped me understand GlusterFS and its complications and limitation and was major force in guiding my research in the correct direction. Without Claris the completion of this project would have proved to be much more challenging than it already was.

Finally I would like to thank the good people at RENCi, UNC-Chapel Hill and all the Universities that hosted computing servers for ExoGENI for working round the clock to keep all the systems running and giving me and fellow researchers the ability to explore and innovate.

Table of Contents

Abstract	ii
Acknowledgements	iii
List of Tables	vi
List of Figures	vii
1. Introduction	1
1.1. Motivation & Goals	2
1.2. Contributions	3
1.3. Thesis Organization	3
2. Related Work	4
2.1. Literature Survey	4
3. Background	7
3.1. ExoGENI	7
3.1.1. ORCA	8
3.1.2. Flukes	8
3.2. GlusterFS File System	11
3.2.1. Gluster Concepts	12
3.2.1.1. Terms	12
3.2.1.2. Translator	13
3.2.2. GlusterFS Configurations	14
3.2.2.1. Distributed Volume	14
3.2.2.2. Replicated Volume	15

3.2.2.3. Striped	15
3.2.2.4. Distributed Striped volumes	16
3.2.2.5. Distributed Replicated volumes	16
3.2.2.6. Striped Replicated	17
3.3. Iozone File System Benchmark	18
3.4. Ganglia Network Monitoring System	19
4. Experimental Evaluation	20
4.1. Experimental Setup	20
4.2. Results & Evaluation	22
4.2.1. Workload Testing	22
4.2.2. Distributed Workload Testing	26
4.2.3. Software Defined Network (SDN) Testing	29
4.2.4. Workload Testing: Same file	30
5. Conclusion & Future Work	32
5.1. Future Work	33
References	34

List of Tables

4.1. File Size Distribution	21
4.2. Throughput Testing WVN	23
4.3. Throughput Testing WSU	24
4.4. Throughput Testing OSF	24
4.5. Bandwidth Utilization per client	26
4.6. Distributed Throughput Testing	27
4.7. Maximum Throughput per client	29
4.8. Network Manipulation: Throughput per client	30

List of Figures

3.1. Requesting a slice through flukes	9
3.2. Manifest of a requested slice	9
3.3. GlusterFS Virtual File System	11
3.4. Mounted GlusterFS File System on a Client	12
3.5. A translator logically.	13
3.6. The Posix translator	13
3.7. Example of a Brick Graph	13
3.8. GlusterFS Volume	14
3.9. Distributed GlusterFS Volume	15
3.10. Replicated GlusterFS Volume	15
3.11. Striped GlusterFS Volume	16
3.12. Distributed Striped Volume	16
3.13. Striped Replicated GlusterFS Volume	17
3.14. Iozone results	18
4.1. Implementation	20
4.2. Throughput testing on WVN	23
4.3. Throughput testing on WSU	24
4.4. Throughput testing on OSF	25
4.5. Maximum Throughput(%) per client	26
4.6. Throughput testing in parallel and isolated	27
4.7. Average throughput per user in parallel and isolated tests	28
4.8. Throughput of multiple users reading the same file	31
4.9. Network activity	31

Chapter 1

Introduction

Distributed file systems also known as network file systems have been around for quite a long time. Distributed file systems as the name suggests allows data storage across a number of servers to provide advantages such as redundancy, performance, cost etc. Distributed data storage generally follows the client/server model where servers in remote locations store data in various formats, locations and the client machines access this data exactly like they would locally.

With the advent of cloud computing which believes in economies of scale where computing services like processing, memory and storage etc. are shared among many users it is important to support this structure with a dynamic and malleable storage solution. Distributed file systems allow us to define network storage and change aspects like bandwidth and throughput according to demand and priority. This flexibility is backbone of the Infrastructure as a Service (IaaS) model. The many advantages of such file systems are already widely known, because of the pay-as-you-go model of these services new companies can get off the ground much quicker by avoiding the massive initial investments in installing data servers and focus more on differentiating their service/applications from the competition. There are also the other environmental advantages of shared resources like less wastage of power and optimize efficiency depending on demand. The adoption and growth of cloud computing [1] is what was the motivation behind this project.

Software-defined networking (SDN) is an approach that manages network services through abstraction of lower-level functionality. In contrast to traditional networks in SDN we can define the bandwidth, no. of connections, throughput dynamically. The advantage of SDN is coupled with the massive need of computing and network bandwidth in today's cloud environment where multiple users with completely different QOS requirement/needs access

data through the same network. For example a scientist running a computation heavy program using command line software doesn't need the same bandwidth as an engineer running a GUI based chip-design software, hence by using SDN we can make use of the same bandwidth to give better QOS to both. The impact of such SDN on a distributed file system is the aim of this work. The representative file system used in this work is GlusterFS.

GlusterFS is an open open source distributed file system capable of scaling to upto 72 brontobytes [2]. GlusterFS aggregates storage servers over Infiniband RDMA or TCP/IP connections, and manages them in one namespace. We have deployed GlusterFS on clusters of virtual machines on the Global Environment for Network Innovation (GENI) which is a virtual network testbed for experiments in distributed computing. All the testing and simulations were done using Iozone [12] which is an open source benchmark filesystem which generates a variety of read/write operations to emulate different categories of users on the GlusterFS client machines. This focus of my thesis is characterizing and testing GlusterFS, a distributed file system in a truly distributed environment. A lot of research has been done [4, 5, 6, 7] but all of them focus on improving speed or access times for the end user by adding memory [7] or building a layer of abstraction [6] over glusterFS without giving much details about how users were distributed among many clients all trying to access and manipulate data on the file system.

1.1 Motivation & Goals

The motivation for this work is exploring the potential impact of SDN in distributed file systems and we needed to understand the behaviour of a representing filesystem (GlusterFS) and how the networks' configuration impacts relevant metrics such as performance, resource utilization or energy consumption. We propose that through this research we want to achieve maximum throughput for accessing data on our distributed file system but we wanted to come to this solution by thoroughly testing and stressing the file system on the current network. Analysing how multiple users all distributed among different clients at separate geographical locations affect the access time for all users. The underlying focus was always understanding how GlusterFS behaves under different circumstances such as unbalanced

user loads and differences in bandwidth between multiple clients and how the file system reacts to these changes. The project is an initial exploratory work into understanding GlusterFS and the intent is to give researchers a baseline to build upon so they can make the correct choices in providing different Quality of Service as per requirements in their systems.

1.2 Contributions

One of the major contributions that this work has made is collecting empirical data on a real life testbed ie. ExoGENI. Although there is a lot of complexity involved in setting up and resource provisioning on the geographically spread out network such as the one used in this work, We focused on data that represented a real distributed environment with multiple users that are distributed logically and geographically. This work is also contributing by adding more dimensions to current research [5] by using their research and entering into the realm of Software Defined Networks and how file systems need to adapt to this demanding new area.

1.3 Thesis Organization

The remainder of this thesis is organized as follows Chapter 2 will focus on the current and related work along with motivations. Chapter 3 will define implementation of the characterization along with an explanation of the various tools and frameworks that were used for this project. Chapter 4 will define the experimental setup and assumptions (if any) that were made during the project. Finally, Chapter 5 will provide the thesis conclusions and will outline possible directions for future work.

Chapter 2

Related Work

2.1 Literature Survey

GlusterFS is a recent File System and the literature characterizing the behavior of GlusterFS is not very extensive. Existing work [5] has mainly concentrated on the characterizing the different types of configurations available in GlusterFS i.e., how is the data distributed among the GlusterFS storage servers. GlusterFS provides many different ways to distribute data across storage servers, each storage block is known as a brick and each storage server can contain one or many bricks. Files can striped (broken down) across bricks, distributed among bricks (bricks are selected using an internal hash algorithm according to filenames) or file can simply be replicated among bricks. More will be discussed about this in the implementation section. Huang et al. [5] studied the “Quality of Service” (QoS) model through measuring file write/read throughputs over different GlusterFS configurations. This work is relevant to this thesis as it provides a good baseline on what to expect while going in to experiment the different configurations but Huang et al. have not focused on understanding how this “QoS” or the throughputs of the file system as a whole change when multiple users are added to the system. They do talk about parallel and distributed applications but have merely studied how the system behaves while dealing with different file sizes sequentially, but there is no quantifiable data to illustrate that any such testing in a highly concurrent environment, which is the ultimate goal of all distributed file systems. There has been no stress testing of the system, with multiple users distributed or not, they have not given any details of the system namely, how many server nodes how many client servers did their system use etc.

Yu et al. [4] presented novel ways to seed torrents by using GlusterFS servers as a back end framework and have used Eucalyptus cloud an open source cloud computing platform developed by the University of California, to emulate a network of users, using a Bitorrent client. They have mainly concentrated on adding and removing Virtual Servers as per demand which make a good future step for this work. They do mention using a “Distributed Replicated” GlusterFS configuration without dwelling into and analysing different configurations of GlusterFS and why this specific configuration best suits their application of a highly concurrent model. The main aim of their project was not to characterize GlusterFS but they could have used a more comprehensive study of GlusterFS to better understand the back end behavior of their systems. They discuss about using three physical servers as their GlusterFS storage servers but a major drawback of their work is that while emulating a multi-user environment they do not clearly illustrate how multiple users are affecting their system. Torrents being a highly concurrent system, they have not emulated multiple users or tried to measure how multiple users affect their system.

More we dwell into related work the larger is the need to understand GlusterFS more accurately. - remove this sentence Existing work [6] have also made lot of effort building a whole virtual network layer on top of GlusterFS without addressing how to fine tune GlusterFS for their approaches. This existing work [6] has not stated how their storage is deployed which is very important because search relates mainly to metadata and hence very small file sizes but a highly concurrent environment for small files distributed systems will offer more bandwidth as the overheads involved in striped file systems may outweigh the benefits of using a striped configuration. Other work such as [7] have gained good ground in improving latency but not by fine tuning their backend storage systems but by trying to improve the current database by adding layers of storage in between the storage server and the client. Moronha et al. [7] have studied many interesting problems in multiple client file systems like **single server bandwidth drop with multiple clients, parallel I/O bandwidth from multiple servers** these are all problems we faced during this work and hence [7] makes a very good addition or next step in our work.

Finally to get a good reference to understand how to measure or benchmark our file

system Vogel et al. [14] provided a very comprehensive resource. Their work provides a lot of insight into how one can define Quality of Service (QoS). QoS can mean many things and especially in distributed media there are many dimensions like connectivity, throughput, latency and many more. In agreement with [14] we have decided to use available throughput i.e., the rate at which data can be written or read from the file system to be our QoS measure. All our experiments are hence measured in total throughput or throughput per user. The user being a process in our tests which emulates the behaviour of a user trying to access their data on our file system.

Chapter 3

Background

The implementation of this project has 3 essential requirements. The network testbed (ExoGENI), a file system (GlusterFS) and a benchmarking tool (iozone).

3.1 ExoGENI

GENI [8](Global Environment for Network Innovations) provides a virtual test bed of virtual machines for networking and distributed systems research and education. It is well suited for exploring networks at scale, thereby promoting innovations in network science, security, services and applications. GENI allows us to obtain compute resources from locations around the United States where GENI has a resources from many universities like UCD,FIU,UFL etc. Connect compute resources using Layer 2 networks in topologies best suited to their experiments. Install custom software or even custom operating systems on these compute resources as there is a whole virtual machine available to the users. Software's such as network monitoring systems were installed to study the network traffic patterns on each of the GlusterFS nodes. Control the bandwidth of network switches to test out systems and architectures as required by the user. Run their own Layer 3 and above protocols by installing protocol software in their compute resources and by providing flow controllers for their switches.

GENI provides an array of compute and storage resources at multiple universities across America. Geni provides compute resources in the form of "Virtual Machines" or "Bare Metal" Machines, attached Network Storage. Each of these machines is defined as a "Node" by GENI and we can connect all these servers together via a broadcast links that connect our virtual network together. In the next section Flukes is described which allows us provision these resources from GENI.

3.1.1 ORCA

ORCA [9] is a Control Framework to provision virtual networked systems via Secure and Distributed management of Heterogeneous Resources over Federated substrate sites and domains. It was originally developed by the NICL Lab at Duke University, and is currently being developed jointly with Duke University by the Networking Group at The Renaissance Computing Institute (RENCI).

ORCA is the control framework for the GENI testbed. ORCA is tightly integrated with OpenStack and Eucalyptus via special extensions to both for provisioning virtual machines. ORCA is also integrated with xCAT to support bare-metal node provisioning. The ORCA native interface is used by flukes to provision and monitor ORCA substrates.

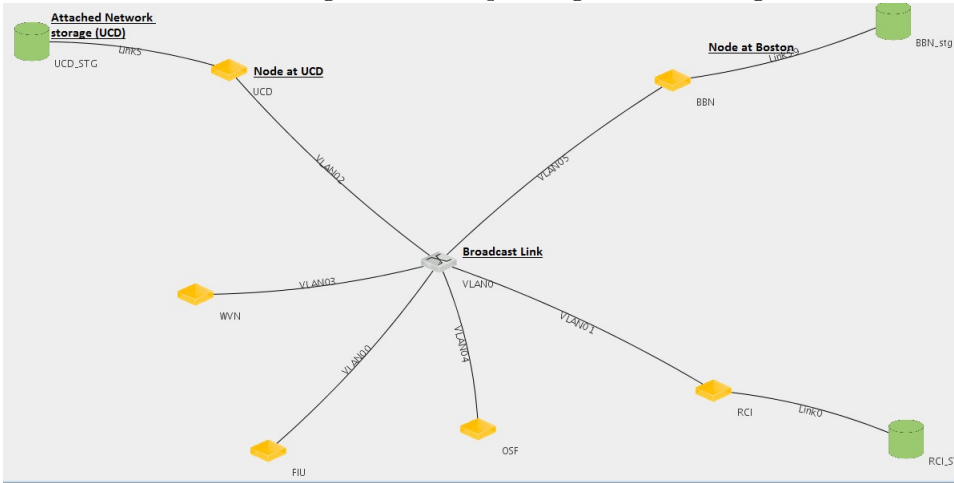
3.1.2 Flukes

Flukes [10] is Java based GUI that allows an experimenter to graphically inspect the state of ORCA substrates, create request topologies, submit requests to ORCA and inspect the returned substrate information (called manifest). The key features of flukes include

- Graphical User Interface.
- Uses native ORCA interfaces and resource descriptions for maximum flexibility.
- Allows the user to submit requests and inspect the output manifest for any failing or defective units in the network.
- Allows users to login to provisioned resources.
- Allows users to extend reservations of provisioned resources for further experiments.

Figure 3.1 shows a typical request for a slice. A “slice” is defined as a cluster of resources that include nodes, links and storage devices. Each node can chose from flavour of different operating systems like Ubuntu, CentOS, RedHat etc. Network storages can provision up to multiple 100GB of disk space which can be attached to the Nodes.

Figure 3.1: Requesting a slice through flukes



After requesting slice we get a manifest from ORCA which gives us the status of each of the provisioned resources from our slice.

Figure 3.2: Manifest of a requested slice

Resource states (start: Fri Nov 21 15:07:37 EST 2014 end: Fri Dec 12 09:44:59 EST 2014):

Resource Name	Resource State
UCD_STG	Active
wsuNet/Domain/vlan/4cd658d4-6779-4398-adb2-2b8ecbcca3...	Active
WSU	Active
FIU	Active
FIU_stg	Active
UCD	Active
ion/Domain/vlan/b10d41a7-d70c-4760-b85b-8adf4a53d41f/vl...	Active
rciNet/Domain/vlan/3c7eaeac-b585-4cf4-b15a-5bcd97e86cd/...	Active
nlr/Domain/vlan	Failed
RCI	Active
wvnNet/Domain/vlan/7583fc53-59f1-4ae7-99d4-e32ffa1984e0...	Active
ion/Domain/vlan/ef0a6b20-a0ef-4515-a87c-f8d4b9c1a0a3/vlan	Active
ion/Domain/vlan/eb59b9d3-00a3-4cf8-bf0d-5f845affba05/vlan	Active
OSF	Active
fiuNet/Domain/vlan/758559d9-5db5-4a68-982b-d0280d3be3...	Active
WVN	Active
ucdNet/Domain/vlan/abe31e29-dad8-4627-8c49-536855fbe5...	Active

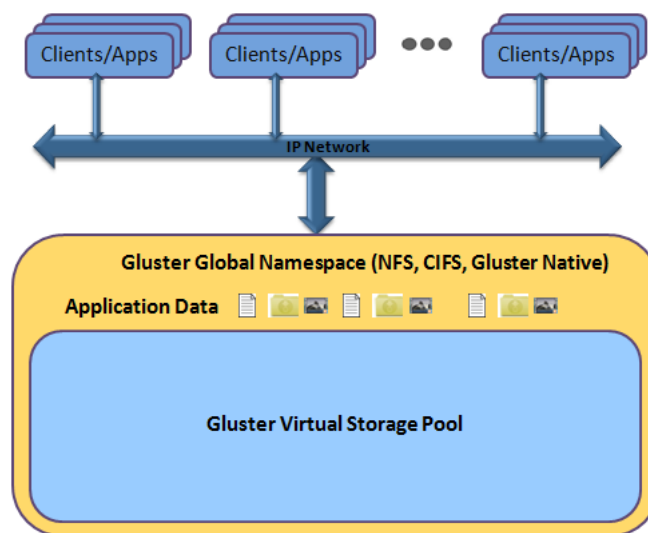
The above figure 3.1 shows a typical “slice” request in flukes and also one that was used during my research. The yellow blocks are nodes or virtual machines with a preloaded operating system. The green cylinders represent attached “network storage” which are of 100GB each in this case. The grey link is the “broadcast link” which is connecting all the nodes together. Looking at this picture it is obvious that only 3 nodes have attached storage, hence we can conclude that I used a 6 node network with 3 nodes serving as GlusterFS servers while the other 3 nodes are serving as GlusterFS servers. Figure 3.2 shows a manifest for a provisioned slice, where one can view the status of all the requested

resources.

3.2 GlusterFS File System

GlusterFS [2] is an open source, distributed file system capable of scaling beyond petabytes (actually, 72 brontobytes) 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.

Figure 3.3: GlusterFS Virtual File System



GlusterFS supports standard clients running standard application over any standard IP network. GlusterFS supports standard clients running standard applications over any standard IP network. Figure 4.1 [2], above, illustrates how users can access application data and files in a Global namespace using a variety of standard protocols. Gluster takes advantage of commodity software, users can use NFS, SMB/CIFS as shown in figure 4.1, there is also no need to fine tune the kernels GlusterFS supports almost any Operating System, of course there are some recommended ones.

3.2.1 Gluster Concepts

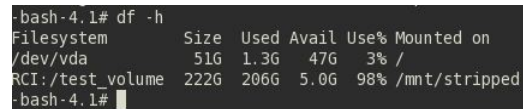
3.2.1.1 Terms

- **Brick**

The brick is the storage filesystem that has been assigned to a volume. Bricks reside on GlusterFS servers, each server can have a single or multiple bricks.

- **Client**

The machine which mounts the volume (this may also be a server). At the client GlusterFS has an API which makes the mounted drive look like a local disk area.



```
-bash-4.1# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/vda         51G   1.3G   47G   3% /
RCI:/test_volume 222G  206G   5.0G  98% /mnt/stripped
-bash-4.1#
```

Figure 3.4: Mounted GlusterFS File System on a Client

Figure 3.4 shows what mounted GlusterFS file system looks like on a client machine.

- **Server**

The machine (virtual or bare metal) which hosts the actual filesystem in which data will be stored.

- **Subvolume**

A brick after being processed by at least one translator.

- **Volume**

The final share after it passes through all the translators.

3.2.1.2 Translator

A translator connects to one or more subvolumes, and offers a subvolume connection for the users.



Figure 3.5: A translator logically.

The brick's first translator is the storage/posix translator that manages the direct filesystem interface for the rest of the translators. This posix translator directly looks at the ext3, ext4 FAT filesystem on the network storage and translates this into a common format for all other translators.

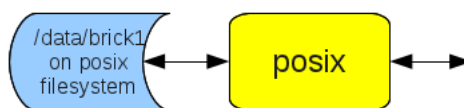


Figure 3.6: The Posix translator

The configuration of translators (since GlusterFS 3.1) is managed through the gluster command line interface (cli), so the users don't need to know in what order to graph the translators together this is done internally. Finally all the translators hooked up together to perform a complete translation from the local filesystem to the mounted network drive, is called a graph. A complete brick graph might look like this:

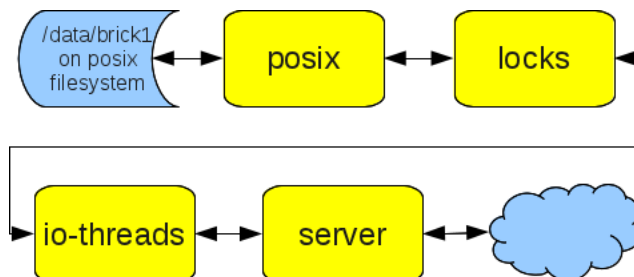


Figure 3.7: Example of a Brick Graph

3.2.2 GlusterFS Configurations

GlusterFS has 3 basic configurations [11] striped, replicated and distributed. These 3 configurations can be mixed and matched together to get better redundancy, throughput etc for different use cases.

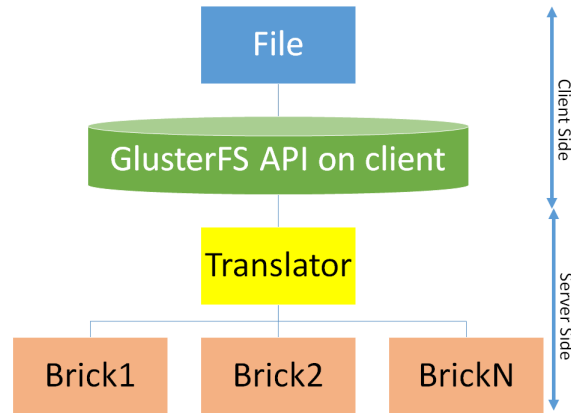


Figure 3.8: GlusterFS Volume

But before we describe all the different configurations Figure 3.7 shows a logical setup configuration of a GlusterFS setup where the GlusterFS API resides on the client server and translator decides how the files have to be striped and distributed among bricks depending upon our configurations. But as mentioned above the user is completely shielded from the translator and its activities. This mapping can although be seen in files that gluster created on the server side.

3.2.2.1 Distributed Volume

Distributed volumes distributes files among different bricks/servers depending on a hashing algorithm depending on the names of the files. These files are not processed in any way but taken as a whole and stored in one of the bricks at random. Distribution provides fast throughputs in reading and writing as there is no processing required for the file but bad redundancy because if anyone server crashes then we can lose whole chunks of data in our volume. Also if the single file is larger than a brick then the write will fail.

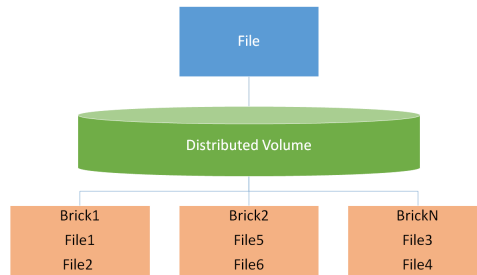


Figure 3.9: Distributed GlusterFS Volume

Distributed volumes are obviously not good for high concurrency applications where multiple clients might try to access multiple files or even the same file and be limited by the bandwidth of a single server.

3.2.2.2 Replicated Volume

Replicated simply takes a file and stores multiple replicas of it on multiple bricks as per our definition . Replication obviously is just to add data redundancy and not really for any algorithmic advantage to increase access times . Replicated volumes are usually used in addition to the other 2 types of volumes.

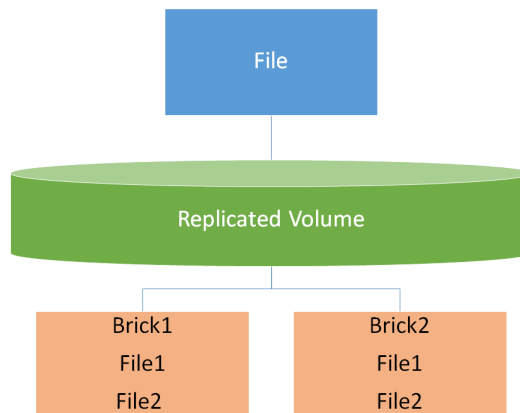


Figure 3.10: Replicated GlusterFS Volume

3.2.2.3 Striped

Striped volume stripes data across bricks in a volume. This means that the each file is split up into multiple parts and stored on different bricks. Stripe volumes do away with

much of the disadvantages of distributed volumes such as in high concurrency environments and even if the size of the file is larger than the remaining space in one brick, the file can be partitioned accordingly and stored in different bricks.

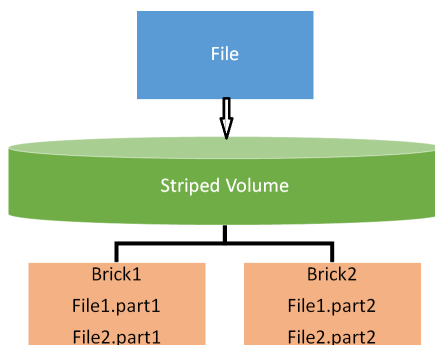


Figure 3.11: Striped GlusterFS Volume

3.2.2.4 Distributed Striped volumes

Distributed striped volumes stripe data across two or more nodes in the cluster. You should use distributed striped volumes where the requirement is to scale storage and in high concurrency environments accessing very large files is critical.

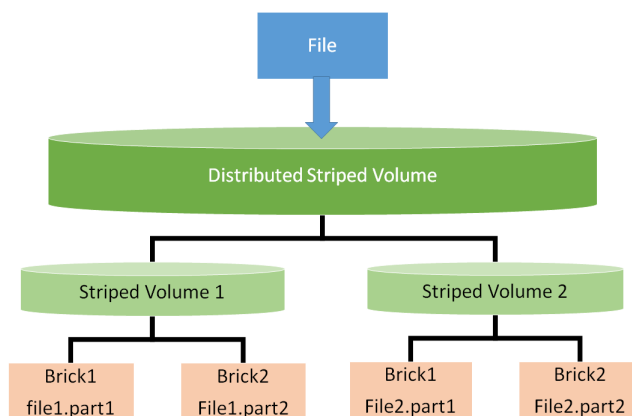


Figure 3.12: Distributed Striped Volume

3.2.2.5 Distributed Replicated volumes

Distributed replicated volumes distributes files across replicated bricks in the volume. You can use distributed replicated volumes in environments where the requirement is to scale

storage and high-reliability is critical. Distributed replicated volumes also offer improved read performance in most environments.

3.2.2.6 Striped Replicated

Striped replicated volumes stripes data across replicated bricks in the cluster. For best results, you should use striped replicated volumes in highly concurrent environments where there is parallel access of very large files and performance is critical.

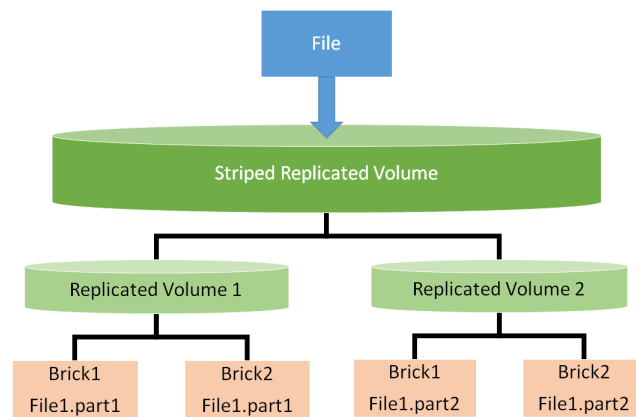


Figure 3.13: Striped Replicated GlusterFS Volume

The subtle difference between stripe-replication and distributed replication is that stripe replication will partition files and replicate the partitions of the file for added redundancy along with concurrency already present in striped volumes. On the other hand distributed-replicated drives simply replicate complete files .

3.3 Iozone File System Benchmark

IOzone [12] is a filesystem benchmark tool written in C++. The tool can be run on any linux operating system with gcc, Iozone has been ported to many machines and runs under many operating systems. Iozone is useful for performing a broad filesystem analysis of a vendors computer platform. The benchmark tests file I/O performance for the following operations: Read, write, reread, rewrite, read backwards, read strided, fread, fwrite, random read. All the I/O performance testing is done the form of throughput, Iozone measures throughput of each of the functions described above. Iozone has 2 fundamental modes of operation an auto and a manual mode. While functioning in auto mode we can provide iozone with a multitude of inputs to define file size range, types of operation etc we get a combined three dimensional throughput assessment as shown in figure 3.14.

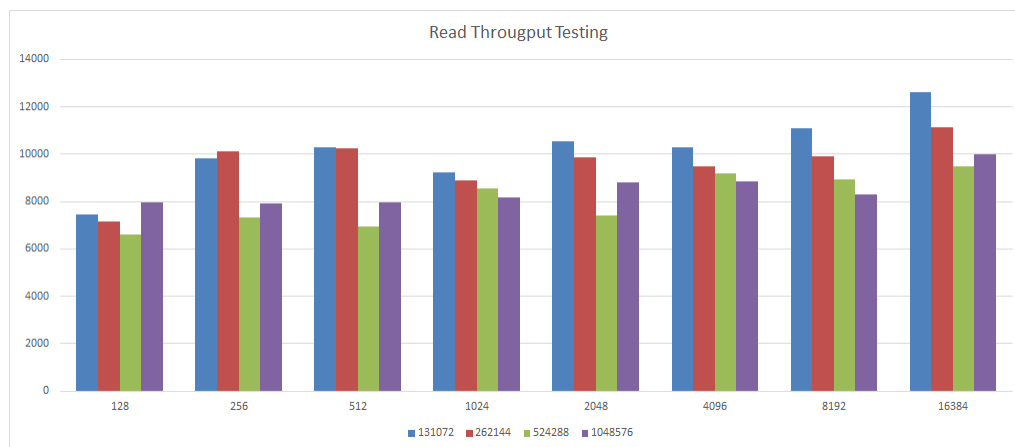


Figure 3.14: Iozone results

This graph shows a comparison in reading throughput speeds for different file sizes and each file size has been read with a different block size or page length of reading. In the manual mode operation we can do throughput testing on files previously generated and provide file names, size and precisely which test out of the multitude of available tests. Iozone testing formed platform of all experimentation, for testing in a distributed environments and emulating multi user loads on the filesystem, I had to write many scripts around the iozone executable to fine tune and adapt iozone to my needs.

3.4 Ganglia Network Monitoring System

Ganglia [15] is a scalable distributed monitoring system for high-performance computing systems such as clusters and Grids. It is based on a hierarchical design targeted at federations of clusters. It leverages widely used technologies such as XML for data representation, XDR for compact, portable data transport, and RRDtool for data storage and visualization. It uses carefully engineered data structures and algorithms to achieve very low per-node overheads and high concurrency. The implementation is robust, has been ported to an extensive set of operating systems and processor architectures, and is currently in use on thousands of clusters around the world. It has been used to link clusters across university campuses and around the world and can scale to handle clusters with 2000 nodes.

Ganglia is a BSD-licensed open-source project that grew out of the University of California, Berkeley Millennium Project which was initially funded in large part by the National Partnership for Advanced Computational Infrastructure (NPACI) and National Science Foundation RI Award EIA-9802069. NPACI is funded by the National Science Foundation and strives to advance science by creating a ubiquitous, continuous, and pervasive national computational infrastructure: the Grid. Current support comes from Planet Lab: an open platform for developing, deploying, and accessing planetary-scale services.

Chapter 4

Experimental Evaluation

4.1 Experimental Setup

The implementation of this thesis was done on a slice of six servers spread out all over USA. Figure 4.1 gives an accurate picture of the experimental setup used for this experiment. The 6 servers reside at Davis (UCD), Oakland (OSF), Miami(FIU), Chapel Hill (RCI), Detroit (WVN) and Morgantown(WSU). The 3 servers at Davis, Chapel Hill and Miami are being used as storage servers while Morgantown, Oakland and Detroit are being used as GlusterFS client servers.

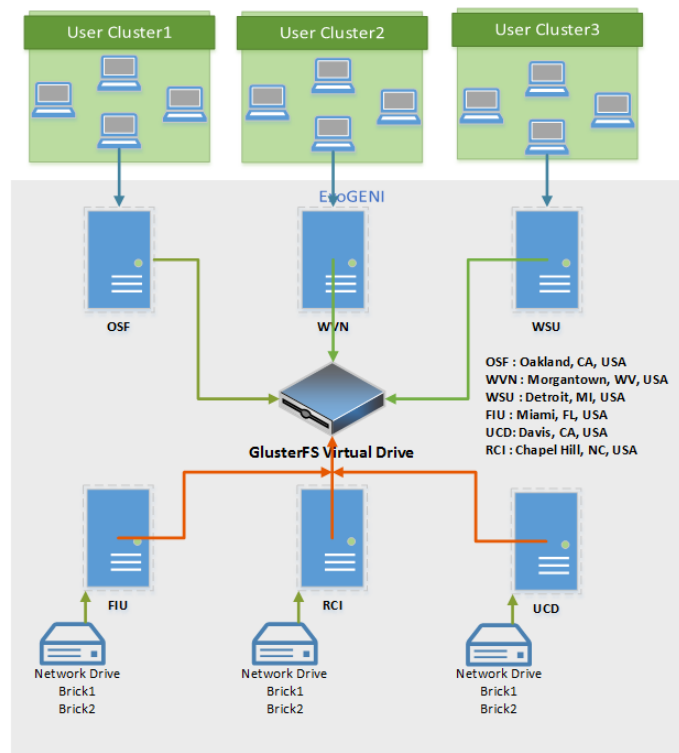


Figure 4.1: Implementation

The GlusterFS virtual network drive is created by RCI and mounted on each of the client machines. Each storage server has an allocated disk space of 75GB so the GlusterFS drive has a total of around 225GB. To test the filesystem the gluster drive was filled up to 95%. The file size distribution used in [13] was tweaked a little but the following distribution was followed :

File Size	Distribution
5MB	40%
50MB	40%
500MB	10%
5GB	10%

Table 4.1: File Size Distribution

The percentage values above reflect a percentage of the total disk available disk size, 220GB in our case. They collected file distributions from thirty-two file servers that were in use for the ASCI Linux clusters during the science runs phase. The file servers were dedicated to a small number of large-scale scientific applications, which provides a good model of data storage patterns. The experimental use case for this work was to emulate a multi user highly concurrent environment. The natural Gluster configuration for this sort of an experiment was to use a striped file system where files are partitioned into pieces and striped across multiple bricks in the volume. I will further compare the throughput of these two fundamentally different GlusterFS volumes to prove why a striped and not distributed GlusterFS volume was more suitable for my experiment.

There are three main dimensions to evaluate in this work, Workload, Network and Filesystem. Workload obviously refers to the user behaviour and types of file accesses. To test user work load an option of evaluation sequential or parallel loads is present. But for reasons of testing or even stress testing the file system there is no insight to be gained by studying sequential access to the system. Parallel workloads are more interesting to our cause because we want to see how the system behaves in a highly concurrent environment with multiple users distributed across multiple servers all trying to concurrently access their files on the filesystem. The second dimension to this study is the network. The allocated bandwidth between the storage servers and client servers, and between the storage

servers itself. There were some interesting findings on this dimension that I will present. Finally the third dimension is the file system itself, as mentioned earlier that GlusterFS offers two fundamental types of configurations, distributed and striped, we can use the replicated configuration in concurrence with these to achieve added redundancy to our system. Also mentioned earlier that striped filesystems are ideal for our use case of a highly concurrent environment but we will also study the distributed system to study and quantify its advantages/disadvantages.

4.2 Results & Evaluation

4.2.1 Workload Testing

In this thesis, we have used large files for testing workloads as all the servers used have moderately high bandwidths around 20 Mbps therefore file sizes of at least 5GB were used to allow the network activity to settle and collect more realistic data. The following three tables shows the throughput analysis done on each of client servers individually. This step was necessary to establish a baseline behavior of the different client servers in the network. Each of the client servers was tested individually with none of the the other clients competing for read/write bandwidths. The experiment setup a striped file system configuration of Section 3.2.2.3 and we simply loaded each clients with users to understand what kind of bandwidth allocation is done by GlusterFS. The key to this experiment was to study the following:

1. Bandwidth allocated to each user by GlusterFS.
2. Bandwidth allocation to users when multiple users are competing for resources at the same client.
3. Bandwidth utilization of the client at each data point.

Number of Users	Throughput (KB/sec)	Per User (KB/sec)	B/W Utilization
1	3638.19	3638.19	16.12376984
2	5769.92	2884.96	25.57119394
4	12042.24	3010.56	53.36892964
8	19288.32	2411.04	85.48218545
10	20722.33	2072.233	91.8374465
12	22564.14	1880.345	100
16	22355.36	1397.21	99.07472654

Table 4.2: Throughput Testing WVN

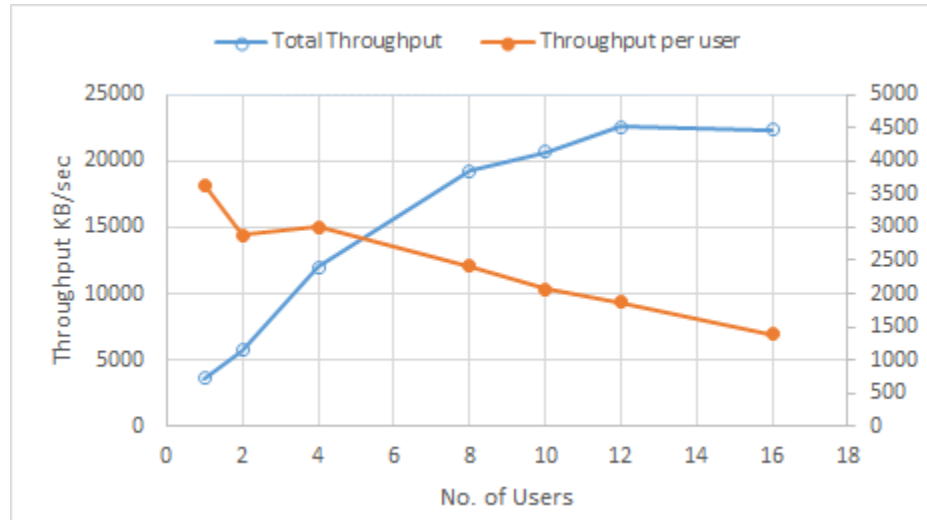


Figure 4.2: Throughput testing on WVN

Looking at the acquired data we are in a good position to answer the 3 major questions. We start testing with a single user and keep loading the client with more users. The bandwidth allocated to each user keeps degrading as the number of users keep increasing. But we notice that a maximum bandwidth is approached around 12 users per client. It is important to note that all users during testing are assumed to be active at the same time and accessing files equivalent to 5GB-10GB. Another observation to be made here is that the bandwidth utilization increases as we load the client with more number of users. This is a drawback in the file system because it should be aiming to maximize bandwidth utilization at all possible times, the same is also a drawback of the striped file system files have to be pulled from different servers and stitched together hence overheads are involved.

This makes the striped configuration a good candidate for highly concurrent use cases.

Number of Users	Throughput (KB/sec)	Per User (KB/sec)	B/W Utilization
1	2816.23	2816.23	38.26337071
2	4462.96	2231.48	85.80294886
4	6315.2	1578.8	60.63705483
8	6958.81	869.85	94.54750738
12	7191.41	599.28	97.70778194
16	7360.12	460	100

Table 4.3: Throughput Testing WSU

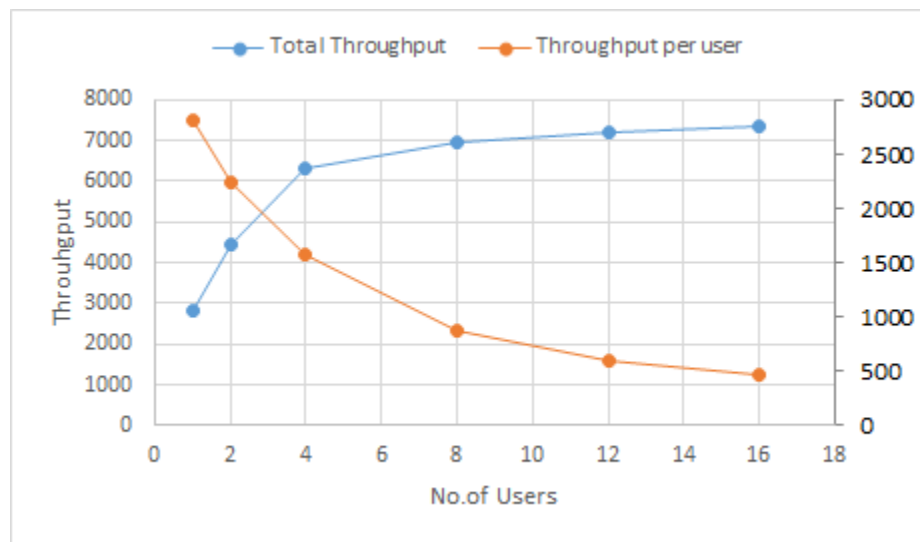


Figure 4.3: Throughput testing on WSU

Number of Users	Throughput (KB/sec)	Per User (KB/sec)	B/W utilization
1	3723.05	3723.05	14.72808786
2	5739.89	2869.945	22.7065455
4	13883.89	3470.97	54.92355778
8	24454.57	3056.821	96.74032194
10	23722.33	2372.23	93.8436391
12	25278.57	2106.5475	100
16	22355.36	1397.21	88.43601517

Table 4.4: Throughput Testing OSF

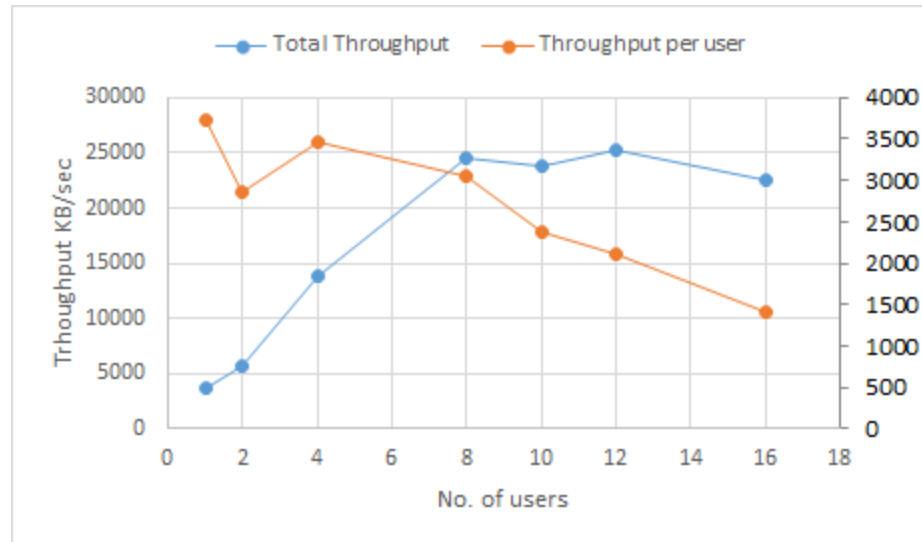


Figure 4.4: Throughput testing on OSF

All three tables resonate the same findings about bandwidth utilization and allocated bandwidth to each user on the client machines. To calculate bandwidth utilization we are going to assume that the maximum bandwidth achieved at client during testing is our true bandwidth available to each client as a total and hence we deduce the following graph, which shows the bandwidth utilization of each client server as we increase the number of users. Infact if we notice row 1 and 3 in table 4.5 you notice that by the time we measure 16 users the utilization starts falling i.e., the overheads involved in handling so many users is actually slowing the network transmission down this maybe another drawback of the striped file system.

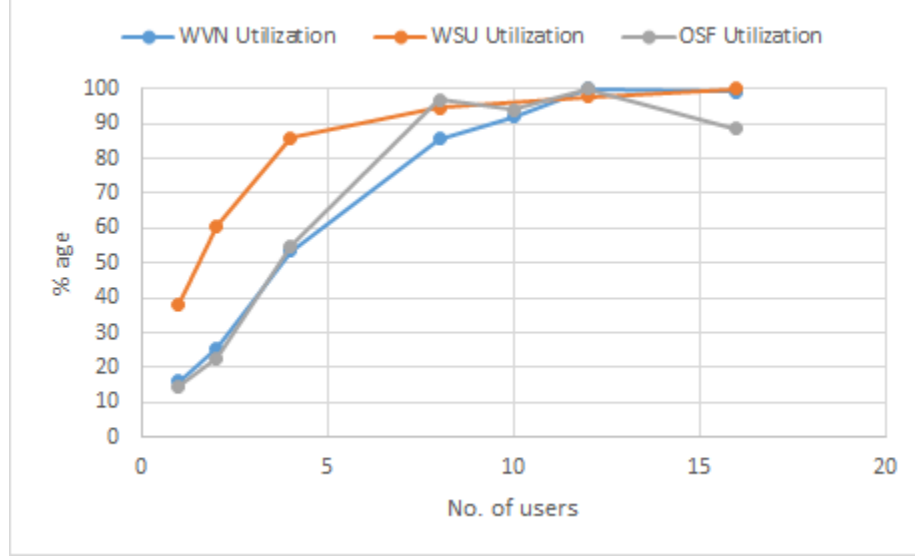


Figure 4.5: Maximum Throughput(%) per client

Number of Users	1	2	4	8	12	16
WVN Utilization	16.1237	25.5711	53.3689	85.4821	100	99.0747
WSU Utilization	38.2633	60.6370	85.8029	94.5475	97.7077	100
OSF Utilization	14.7280	22.7065	54.9235	96.7403	100	88.4360

Table 4.5: Bandwidth Utilization per client

4.2.2 Distributed Workload Testing

This section deals with the same dimension of testing ie workload testing but here we truly test a distributed user environment. The following experiment sets up multiple users on all clients simultaneously . This test was important to emulate a real multi-user environment, we want to strain the system by placing multiple requests to the filesystem not only from the same client but from different clients all at the same time. This stress testing is important in establishing a baseline to the bandwidth spread across different clients and users on that client.

No. of Users	Throughput (KB/sec)	Throughput Per User (KB/sec)	WVN (KB/sec)	OSF (KB/sec)	WSU (KB/sec)
6	16039	2673.166	5854	5886	4297
9	24361.89	2706.876	8724	9466	8724
12	29052.89	2421.074	11204	12936	6644
15	33331.69	2222.112	-	-	-
24	34854.33	1452.263	13239.58	14423.34	7191.41
30	33681.22	1122.707	13875.74	15304.91	4500.57
36	33986.3	944.0638	13198.46	14004.14	6783.7
48	37392.3	779.0062	14154.66	15192.29	8045.35

Table 4.6: Distributed Throughput Testing

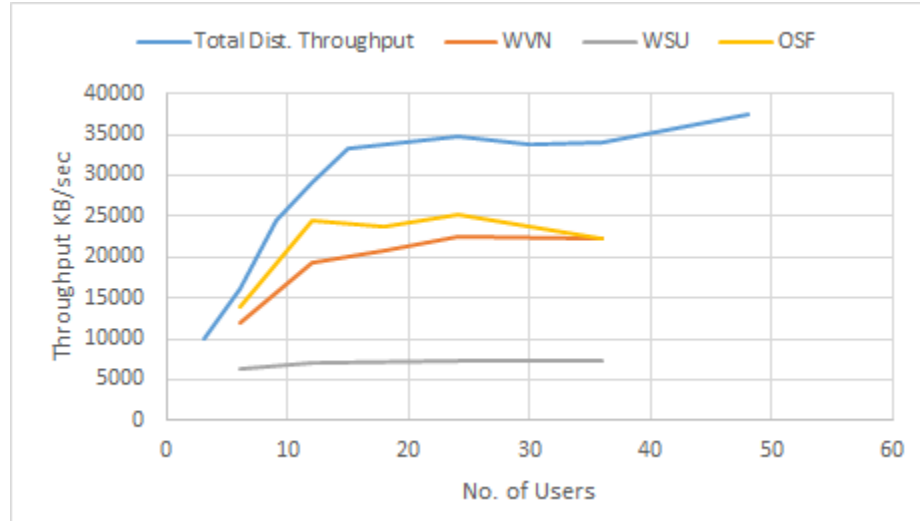


Figure 4.6: Throughput testing in parallel and isolated

The table 4.6 enumerates the same throughput testing done in the earlier sections but this experiment strains the complete network as well as the system, as mentioned earlier, the previous experiment was a simple isolated stress test at each of the client machines. This experiment gives us a good comparison of how the system behaves in a highly concurrent environment. Figure 4.6 is comparing the total throughput and throughput per user on each client during isolated and during distributed testing to better understand how what degree of advantage do we gain in high concurrency environments. It is clear from figure

4.6 that total throughput of the whole system when distributed is much higher than each of the clients in isolation . The other important observation is that throughput per user is also much higher during parallel and distributed testing. This however is very evident that multiple clients will provide more cumulative throughput than each of the clients in isolation but figure 4.7 compares the per user throughput of each of the clients compared to an average per user throughput user when we perform parallel testing.

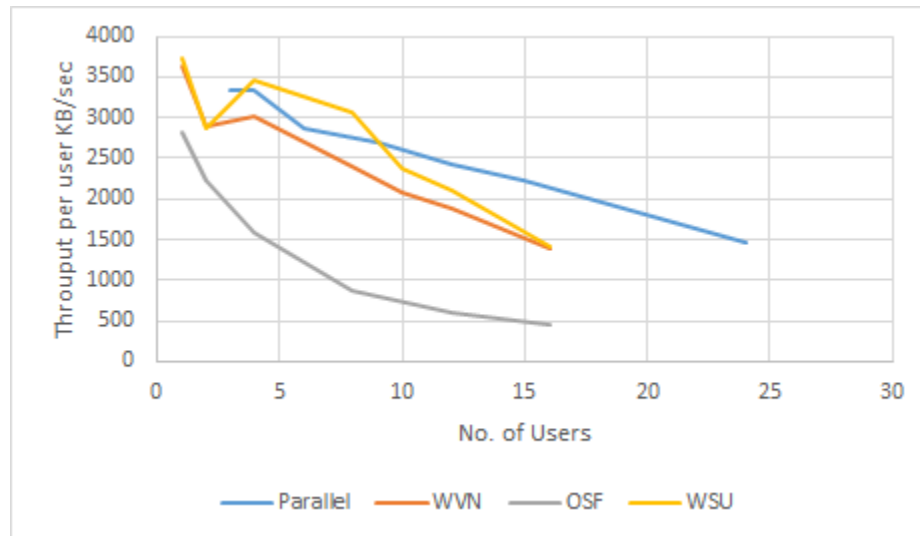


Figure 4.7: Average throughput per user in parallel and isolated tests

In Figure 4.7 although the client WSU initially has a slightly better throughput per user but add we start to increase number of users, parallel environments do have a distinct advantage in GlusterFS striped file systems.

4.2.3 Software Defined Network (SDN) Testing

This experiment is the final dimension of testing the GlusterFS file system by changing network bandwidth of client servers to understand how the GlusterFS filesystem responds. The primary aim for doing this synthetic network modification was to understand how SDN will affect the file system. We then observe whether GlusterFS takes advantage of this and diverts bandwidth to other client servers. This is where the flexibility of SDN and its impact on distributed file systems come in. The results of this experiment will determine the compatibility of GlusterFS with SDN's. To understand better the setup of this experiment let's recap the initial setup of our experiment as shown in **Figure 4.1** and in the next table, table 4.7 are enumerated the the maximum operating throughput that each client achieved during parallel isolated testing, and parallel distributed testing. The latter is expected to be slower as many more number of users were during distributed testing.

Client Server	Isolated (KB/sec)	Distributed (KB/sec)
WVN	22564.14	14154.66
WSU	7360.12	8724
OSF	25278.57	15304.91

Table 4.7: Maximum Throughput per client

An important point to note here is that each of these values is obtained when **10-12 users** are operating on each client server. This same fact is pretty visible through Figures 4.3, 4.4, and 4.5 . Now for this experiment we **limit the bandwidth of Client WSU to about 1500 KB/sec**. After setting up this limitation the effect of this limitation is checked on maximum bandwidth data points i.e., for 10 users and 12 users per client or 30 and 36 users in total. The results of this are in the following table.

	After		Before	
Client Server	10 Users	12 Users	10 Users	12 Users
Total Throughput (KB/sec)	25677.54	30199.82	33681.22	33986.3
Throughput per user (KB/sec)	855.918	838.883	1122.707	944.063
WVN (KB/sec)	19806.42	21969.32	13875.74	13198.46
WSU (KB/sec)	4740.16	6946.8	4500.57	6783.7
OSF (KB/sec)	1130.96	1446.8	15304.91	14004.14

Table 4.8: Network Manipulation: Throughput per client

Table 4.8 shows the clear distinction between the throughput “Before” and “After” synthetically manipulating the bandwidths to measure how this change affects different clients server operating throughputs. Figure 4.8 gives a more graphical description clearly showing how other clients are affected as we change/limit the bandwidth of certain parts of our network. The results are pretty conclusive and can be summarized in the following points.

1. WSU evidently as per table 4.2, is much slower than OSF, WVN and is already at near maximum operating throughput, hence we do not see much change in behavior.
2. The operating throughput for WVN increases by 42% and 66% for 30 and 36 users respectively. This is very interesting behavior which shows that we can easily divert bandwidth to different client servers by simply limiting bandwidth of certain servers.
3. This type of software defined behavior is very important for distributed filesystems in providing different levels of quality of service to different clients.

4.2.4 Workload Testing: Same file

This test although not very crucial the experiment as a whole was done on a different client server based Texas A&M University. This experiment just aims at measuring the throughput of file accesses in case we had a case of multiple users trying to access the same file. Now the drawback for this type testing is that once data is written into the local cache of the client server each user get access to their data instantaneously available to all the users. The same test was performed where multiple users are added onto the same

server to check how the throughput varies. The per user throughput remains almost the same while the total throughput keeps on climbing with each user, this result agrees with our assumption of the file being read into the local cache and then being re-read by each user. This result can be confirmed with the following plot of network activity.

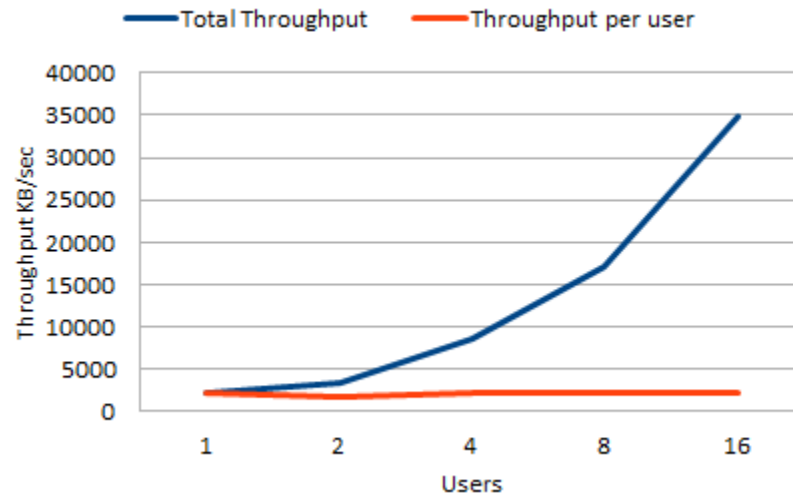


Figure 4.8: Throughput of multiple users reading the same file

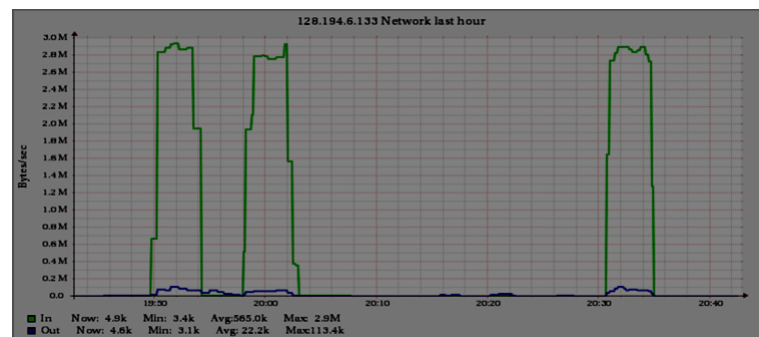


Figure 4.9: Network activity

The above figure 4.9 is plot of network throughput vs time, the the three distinct block that we see is the network activity during 2, 4 and 8 users reading the same file. From a look at the figure it visible that the network activity or the total amount of data transmitted is almost identical which conclusively proves our initial assumption and hence any further investigation down this path is trivial.

Chapter 5

Conclusion & Future Work

GlusterFS is a recent file system and there is a severe lack of research to understand this system better. This work is an initial exploratory work to venture and help towards the adoption of the exciting new open source technology and contribute to the field of distributed computing.

Based on the results obtained in this work we conclude that SDN has a large potential for balancing the different sites in distributed filesystems such as GlusterFS. Another key highlight is that while testing out GlusterFS we have to consider multiple concurrent users to get any reliable data, considering distributed file systems generally have multiple users hundreds of them at times this conclusion makes sense.

Workload testing was used as a baseline test on each client in isolation to understand how each client behaves separately and with this knowledge we can compare how the results to a more realistic distributed system with parallel file accesses to stress the system. Each client server was loaded with multiple users and each of these users were accessing files in parallel. Since we now understand that maximum throughput is not allocated to each user on network when not completely loaded, with at least 10-12 users for our configuration, and beyond these we start losing bandwidth utilization it might be more useful to maintain this balance if not too many users are demanding services at the same time. All these point to the inherent limitations of a non-distributed user environment. Distributing the users across multiple client servers was a very important experiments conducted, because this type of distributed workloads where each distributed client server was hosting multiple users emulates more closely a real life situation. In real cloud systems which are obviously much larger than our testbed, large number of users for example at a University or a research facility where a number of researchers in geographically different locations are

trying to access their data. Of course in a production environment the data access may not be as uniform as our experiments but the idea was to stress test the network and look the extreme end of the spectrum.

For the SDN testing, we synthetically changed the bandwidth of a client and tried to emulate a a software defined network where a certain quality of service is provided to a certain client server. A very important observation was made that we can divert the remaining bandwidth to other clients hence paving the way for a dynamically adjusting QOS either due to requirements of load balancing or simply and user-provider contract.

In conclusion, with this thesis we have acquired a good understanding of the GlusterFS file system and providers of federated cloud systems could use such understanding to decide what kind of QoS they would like to provide to certain users. We have many different combinations that we have studied. We can draw many conclusions based upon the different dimensions of tests that were performed during these all the tests.

5.1 Future Work

Existing work such as the provided in [4], [6] and [7] provide insight on how a better understanding of underlying storage file systems and its configurations can enhance these ideas. Wether we are deploying a novel data access architecture [4] for a specific application or we want to build an abstract layer of data distribution for client application using open source filesystems such as GlusterFS, a deeper understanding with specific benchmarks can give any researcher a better knowledge.

Future work on this project can be extended to much larger networks and increase use cases to emulate actual scientific environments where all the users may not be so uniform and equally balanced over all of the clients. Future work in the domain of SDN, using the capabilities offered by ExoGeni, and the characterization of energy consumption and how SDN can impact the energy cost due to optimization of data placement/movement and network provisioning .

References

- [1] Economist
Article 15th October,2009
<http://www.economist.com/node/14637206>
- [2] Gluster Webpage
http://www.gluster.org/documentation/About_Gluster
- [3] Iozone Webpage
<http://www.iozone.org/>
- [4] Wenlong Yu, JunWu , “The Dynamic System of BitTorrent Seed Service : Based on Eucalyptus Private Cloud”, *Proc. of the IEEE International Conference on Computer Science and Automation (CSAE)*, 2012.
- [5] W. C. Huang, C. M. Liu, and C. C. Lai, “Resource Provisioning with QoS in Cloud Storage” in *Proc. of the IEEE International Congress on Big Data 2014*, Anchorage, AK, 2009.
- [6] J. Kim, I. Kim, T. Kim, Y. I. Eom, H. Y. Kim, Y. Kim, “Design and Implementation of Networking Virtualization for Cluster File Systems”, *Proc. of the International Conference on Computational Science and Its Applications (ICCSA)*, 2009.
- [7] R. Noronha and D. K. Panda, “IMCa: A High Performance Caching Front-end for GlusterFS on InfiniBand,” *Proc. of the International Conference on Parallel Processing (ICPP)*, 2008.
- [8] Webpage Geni
<http://www.geni.net/>
- [9] ORCA Webpage
<https://geni-orca.renci.org/trac/>
- [10] Flukes Webpage
<https://geni-orca.renci.org/trac/wiki/flukes>
- [11] Gluster Community Webpage
http://www.gluster.org/community/documentation/index.php/GlusterFS_Concepts
- [12] Iozone webpage
<http://www.iozone.org>
- [13] F. Wang, Q. Xin, B. Hong, S. A. Brandt, E. L. Miller, D. D. E. Long “File System Workload Analysis For Large Scale Scientific Computing Applications”, *Proc. of the IEEE Conference on Mass Storage Systems and Technologies*, April 2004.

- [14] A. Vogel, B. Kerhervk, and G. V. Bochmann, "Distributed Multimedia and QOS: A Survey," *IEEE MutliMedia*, vol. 2, no. 2, pp. 1019, 1995.
- [15] Ganglia webpage
<http://ganglia.sourceforge.net/>