International Journal of Computer Networks and Applications
Volume 3, Issue 1, January – February (2016)

RESEARCH ARTICLE

Distributed File Systems Implementation on an Edge Router using GlusterFS for Cloud Applications

Souryendu Das
Department of Electrical and Electronics Engineering
BITS Pilani K.K.Birla Goa Campus, Zuarinagar, Goa, India
souryendu@gmail.com

Abstract – A distributed file system (DFS) is a client/server-based application that allows clients to access and process data stored on the server as if it were on their own computer. An edge router has series of data cards, to handle data inflow and outflow which can be of the form of email headers, data packets, etc. Each of these data cards have their own file system architecture. This paper implements a distributed file system approach on all these data cards, so as to make it a centrally controlled one file system and not having parallel many file systems. Convenience of usage and file handling capacity is also looked in this paper. The servers are portrayed as the data cards of an edge router and the client access points are the router processor cards. The DFS on Edge Router is implemented using GlusterFS which is a scale-out network-attached storage file system.

Index Terms – DFS, GlusterFS, Edge Router, File-system Mount point, Cloud, Networking

1. INTRODUCTION

Client-server architectural models for communication and storage of data are increasingly becoming popular in today's world [1]. In client/server model, file server acts as a parent node which allows multiple child nodes to connect with it. It is responsible for central storage and data management so that other computers are enable to access the file under the same network. In a distributed computer network environment, a user program can access both local files, i.e., files stored on secondary storage systems directly connected to the processor, as well as remote files, i.e., files stored on secondary storage systems that are accessed by a distributed network. As the popularity of distributed computer networks has increased, the demand to sore ever increasing volumes of data as remote files has also increased. This is where distributed file system steps in making the task of storage of data easier to accomplish.

Distributed File Systems (DFS) are network file systems based on a client-server architecture which give improved scalability and performance than individual file systems present on different servers [2]. DFS has gained importance in the past years over individual file systems, as it gives a more uniform control and ease of access as a singular file system in a scalable environment escalating over many servers. It also gives advantage of slight modifications in its implementation in the forms of a replicated, distributed-replicated and striped file system which has enormous uses in cloud environments [3-4]. DFS also increases transparency and reduces the bottleneck of file transfer queue and data rates when accessing files from servers. It also ensures file security and backup as there would be multiple copies of the files in various server locations in that particular network. The files cold be stored distributed, replicated striped or any combination of these in the network locations. Early versions of DFS used File Replication Service (FRS) which provides basic file replication capability between servers. FRS identifies changed or new files, and copies the latest version of the entire file to all servers. Then there was introduction of "DFS Replication" (DFSR) which improves on FRS by only copying those parts of files which have changed, by using data compression to reduce network traffic, and by allowing administrators flexible configuration options for limiting network traffic with a customizable schedule. DFS has already been tested and used successfully in cloud environments for machine learning and data mining [5] and also is exascale computing [6]. Having DFS implemented on cloud applications such as edge router also has an advantage over multiple file systems. It makes the system more secure from eclipse attacks on overlay networks [7].

GlusterFS is an open source file system application which can be used to implement DFS. It can automatically copy files and provide file sharing service to solve the virtual machine dynamic migration bottleneck [8] and can be used as the underlying storage devices in a cloud environment. It has also found applications in networking, IP and other cloud applications [9].

In this paper I have used GlusterFS to implement a DFS over and edge router and have discussed its benefits and applications. The paper has been organized into many sections, Section 2 talks about DFS in general and Section 3 shows basic functionalities of GlusterFS. Section 4 depicts the various architectural components of a typical edge router and its functionalities. Section 5 illustrates the DFS implementation over an edge router. Section 6 discusses the applications of DFS in an edge router, and its various performance related features. Section 7 shows the results of implementing this DFS, on an edge router simulator, and then discusses the probability of implementation on an edge router chassis.

RESEARCH ARTICLE

2. DISTRIBUTED FILE SYSTEMS

DFS allows users of physically distributed computers to share data and storage resources by using a common file system over different individual file systems [10]. It provides location transparency and redundancy to improve data availability in the face of failure or heavy load by allowing shares in multiple different locations to be logically grouped under one folder, or DFS root. It achieves reliability by replicating the data across multiple servers. It ensures proper management of files and communication between clients and servers in terms of file processing downloads and uploads. DFS requirements in a network are transparency, concurrency, replication, heterogeneity, fault tolerance, consistency, security and efficiency. A typical implementation of DFS is a collection of workstations and mainframes connected by LAN, illustrated in Fig. 1. For a DFS to exist functionally a minimum of two servers and one client is required, however there isn't any upper bound to the limits here defined by DFS. The upper bound usually depends upon the quality and load handling capacity of the connectivity framework. DFS is also a key building block for cloud computing applications [11]. In such file systems, nodes simultaneously server computing and storage functions and a file is partitioned into a number of chunks allocated in distinct nodes. In networks based file systems where data can exist on various nodes in different servers, it is more convenient to have a centralized file system which can be uniformly accessed and controlled. This centralized file system would manage to store the data in the various nodes at the server locations. Thus DFS makes it more convenient to control and access files stored in a network as it eliminates the usage of multiple file systems and the control and coordination protocols employed between them. Research previously has shown that various implementations of DFS in networks, cloud, routers, etc. have improved stability of the system in overall [12-15]. DFS also ensures network transparency and high availability in a system of connected servers in a network. It dynamically resizes and redistributes stripes on the storage servers, supports varying size of stripes on the storage servers to obtain finer concurrency, granularity on accessing the data stripes [16]. This stripe management mechanism reduces I/O response time and boosts I/O data throughput significantly for applications with complicated access patterns. Architecturally DFS falls into major categories, Fully Distributed and Client-Server Model. Fully distributed means that files are distributed to all sites which leads to issues on performance and implementation complexity. In a client-server model there are two players, file servers and clients depicted in Fig. 1. File servers are dedicated sites strong files and perform storage and retrieval operation. Clients are rest of the sites which use servers to access files. DS is used in the client architectural model in this paper.

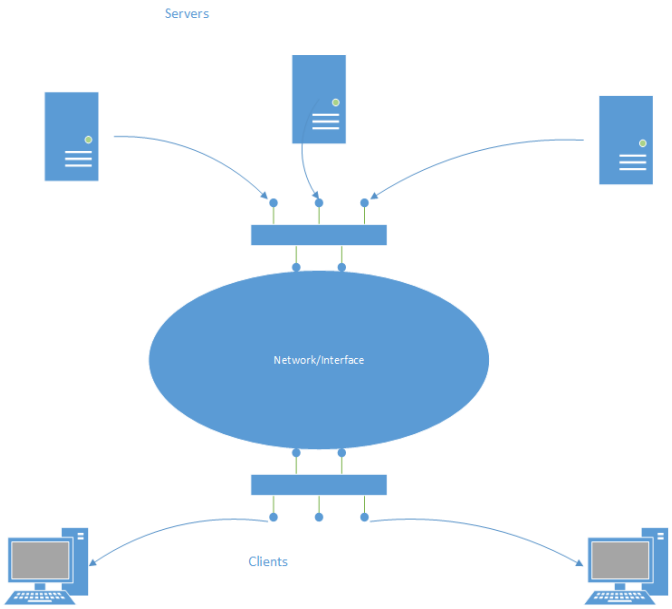

Figure 1. Distributed File Systems Architecture

3. GLUSTERFS

GlusterFS is an open source distributed file system. It has a linear scale-out and supports several petabytes and thousands of parallel connections. There is no metadata structure, it employs elastic hashing algorithm to distribute data efficiently and implements a fully distributed architecture. It also has a global namespace to support POSIX and high reliability on the grounds of data replication and data self-heal. The open source GlusterFS application comes in two packages, one is GlusterFS-server which runs on the server side and the other is GlusterFS-client which runs on the client side. The server side access is through bricks which store data in EXT3/EXT4/XFS format and the client side access file system is through TCP/NFS/SAMBA. Its design and implementation is based on a stackable modular user space. Each functional module in this design is called a translator and all such translators constitute a tree. Such a tree is depicted in Fig. 2. All translators support a common API and can be stacked on top of each other in layers [17-18]. The translator at each layer can decide to service the call, or pass it to a lower-level translator. This modular design enables translators to be composed into many unique configurations. The available translators include: a server translator, a client translator, a storage translator, and several performance translators for caching, threading, prefetching, etc. It uses hashing to distribute files among nodes. This greatly reduces the complexity of the system functions. A typical GlusterFS architecture diagram is depicted in Fig. 3 which shows a simplified version of GlusterFS implementation of DFS with two servers and one client. The two servers which are the two nodes in the figure makes it distributed and there are further two bricks inside these servers which are replicate

RESEARCH ARTICLE

copies of each other making the file system a distributed-replicated file system [19]. Bricks are physical storage units which are allocated for DFS and when a client accesses the data, it mounts a particular brick on its network device's mount point. The file system is built from a cluster of data nodes, each of which serves blocks of data over the network using a block protocol [20]. These data nodes can talk (communicate) to each other for rebalancing data distribution, to move copies around and to keep the replication of data high.

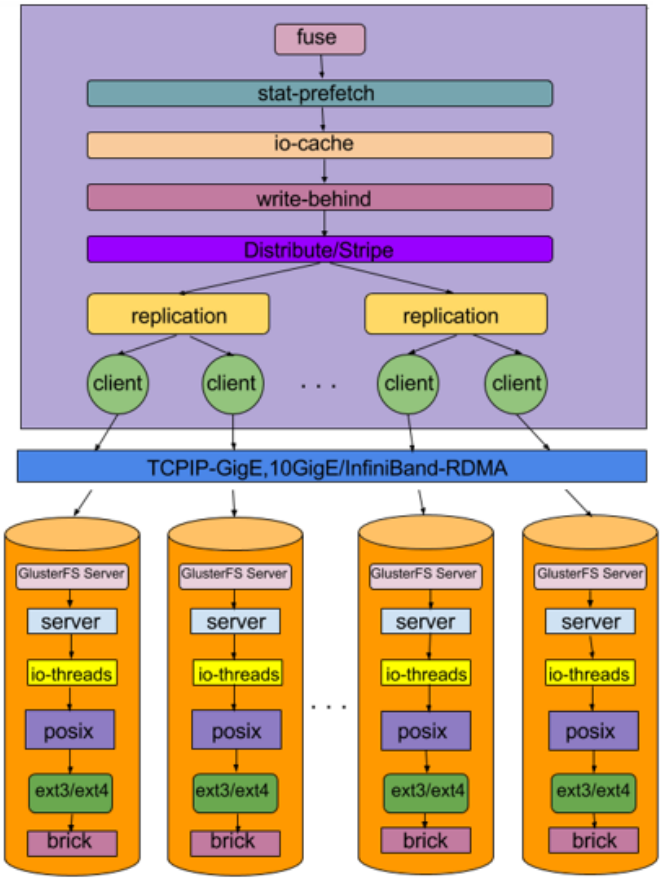

Figure 2. Tree diagram for GlusterFS on client side and server side

From Fig. 3 glusterfsd and glusterd are two daemons which run on server side and stand for gluster file system daemon and gluster daemon respectively and help in archiving the files, in providing the command line interface and executing the server side commands in the backend when a client accesses the file system which is also called a gluster volume. A distributed-replicated file system is much beneficial in this case as it gives more backup and file security in case of mismatched files as it makes a copy of the file in each of the nodes. The DFS which is proposed in this paper for Edge Router application is also of the distributed-replicated type for the obvious above mentioned reasons.

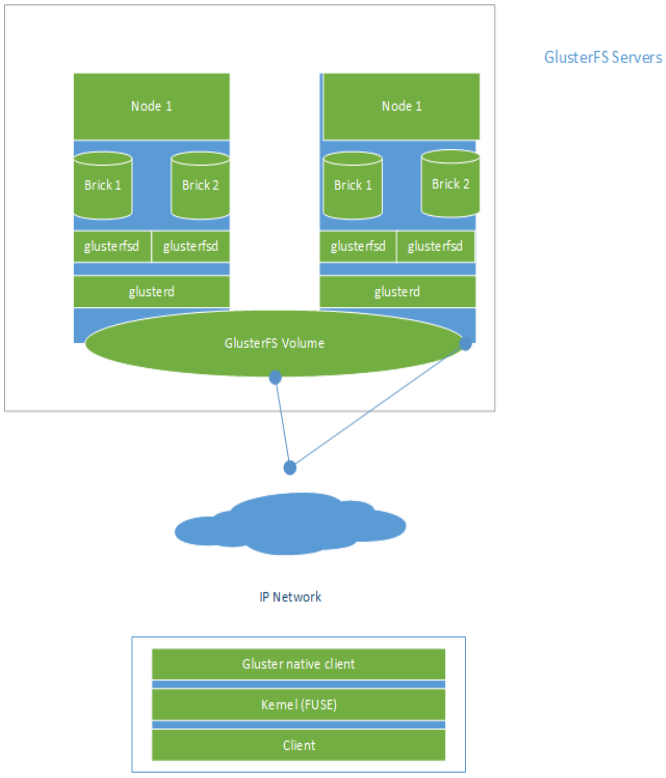

Figure 3. Architecture Diagram of a two node GlusterFS Cluster and a native client

4. EDGE ROUTER AND ITS ARCHITECTURE

An edge router routes data packets between one or more LANs and an ATM (asynchronous transfer mode) backbone network. It is also referred to as a boundary router which is in contrast to a core router, whose primary function is to forward packets to computer hosts within a network. Edge routers are also flexible for interconnection between wireless technologies and IP based networks [21]. The architecture of an edge router is not much different from core router i.e., the former is between two networks and the later works within a network. As in any other case an edge router is also built of a slotted chassis and into those slots cards are inserted. The slotted chassis arrangement helps in simplifying repairs and upgrade of components. There are three kinds of basic cards which can be inserted in a chassis, interface/line cards, route processor (RP) cards and data cards. The interface cards are the cards which actually involve themselves in routing, have the network and mask information and also have an Ethernet interface, and are commonly known as NIC (Network Interface Card). For the purpose of DFS implementation interface cards have no role to play in it. The RP card is the card which controls the functioning in a router. Just as in case of any device like computer and laptops, a router also has fans, alarms and other things which are part of its hardware and also a software which runs on the router which can be called as the operating system of the router. The RP card

RESEARCH ARTICLE

has the OS of the router and controls the hardware peripherals. The data cards is used for storage of file systems and other temporary data like cache memory and flash memory of a router, etc. The information is basically in the form of IP headers, email headers, temporary packets of data used while routing etc. which depends upon the amount of data being queued at the router gateway at that particular time. Fig. 4 shows the architecture of the slotted chassis of an edge router. The number of slots differ for different edge routers which are product specifications done by its manufacturing company. For DFS to work on an edge router, it should have atleast one RP card and two data cards. Since a router needs an interface card to communicate with the outside network, so an interface card is also required in this case. Having more RP cards or data cards is always possible and there is no maximum limit defined by DFS, however there might be hardware and network speed constraints which would be given by the manufacturer.

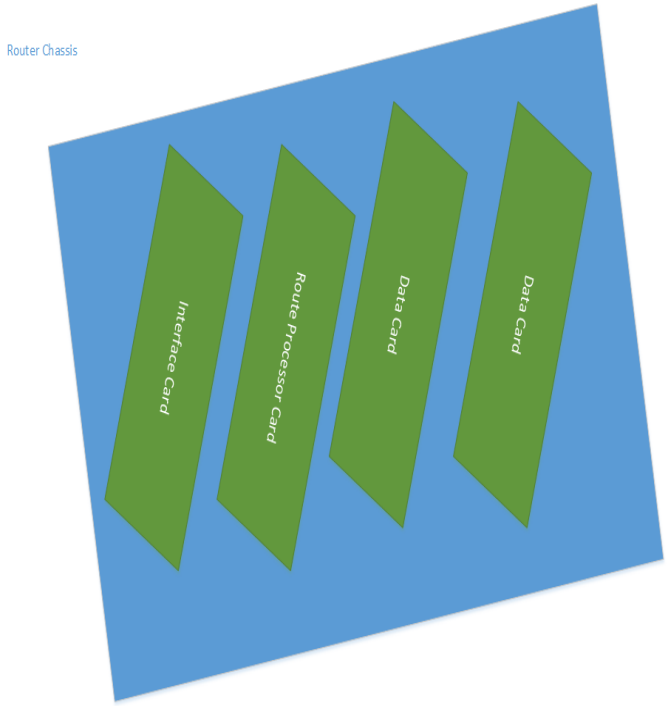

Figure 4. Slotted Chassis Arrangement of an Edge Router

5. DFS IMPLEMENTATION IN EDGE ROUTER USING GLUSTERFS

The client-server architecture described so far in DFS is implemented in the same way in an edge router. In this model the RP card acts as the client where GlusterFS-client will run and the data cards act as servers where GlusterFS-server runs having the bricks, volumes and DFS implemented over them. As described in the edge router architecture, DFS would work efficiently on a minimum of one RP card and two data cards. The data cards will also have a GlusterFS client application running inside them for client level applications running locally inside them and to handle the server access requests from the client at the RP side. The file system being implemented here is of the kind of distributed-replicated, and from Fig. 2, the distribution happen first and the replication happens at the next step. Hence, four bricks which are created are done in a two-step process, first the distribution divides it into two servers and then replication inside the servers allows two copies of the file inside each server making it four. In an edge router the distribution into two occurs into the two data cards, and for the replication inside each of the data cards, there are daughter cards. Each data card has two daughter cards in this case. Daughter cards don't physically exist as hardware, it is just a name given to a certain partitioned disk (fragment) space which is a part of the parent data card. Hence the total disk space of a daughter card is the number of daughter cards multiplied by the disk space of each of those daughter cards. Fig. 5 shows the server side working of GlusterFS on data cards. It shows the interfacing of a data card with the fabric interface and how we control it. The fabric interface is IPoF (IP over fabric), i.e., IPv6 packets are sent over a fabric interface. A similar technology to this one is VoIP (Voice over IP). CLI (Command Line Interface) is actually a backend process which handles all the service requests being raised by clients on data card and issues server responses to the clients from data card, in other words it is the control unit similar to our brain. The DFS exists as a distributed-replicated having four bricks, two each on the two data cards. When some file gets automatically or manually added, changed, modified or deleted in this file system, the commands are given at the CLI. The CLI commands are accepted by DFS Manager who has the ability to interpret these commands and execute necessary actions based on these commands. The actions are executed in the data cards through the help of DFS client handler which handles all the requests coming from the client side at RP for specific file access requests and other similar requests. The DFS client handler processes these requests and passes them on the gluster client application running on data card, which then allows mounting the file system and other specific requests to the Ethernet interface of the data card which is then passed on to the Ethernet fabric from where the client can pick up these data packets. Once the client requests have been processed and approved then, there is a gluster-client application running on data card which then actually pushes the data over the Ethernet interface which is normally a GiGe (Gigabit Ethernet) and then finally to IPoF from where the gluster client application of RP picks it up and sends it to the application running on RP to transform it to a form which an user can readily access it from.

The most interesting part of this architecture is that the client won't know the file is coming from which brick of which data card. The file may also be sent in parts from different bricks, or as a whole from one particular brick, depending upon interface availability and the traffic at each of the individual brick mount

RESEARCH ARTICLE

points. Since the whole file system is one complete DFS, the client needs to mount only one of the bricks of any of the data cards on its directory, as the other bricks are already connected to this one in a unified file system [22]. If there were individual file systems with individual bricks, then the client would have had to mount all the bricks as separate directories for accessing the files, making the process complicated and cumbersome. Since here the files can come in many parts from any of the bricks in parallel, it makes the access faster and easier at the client side. Also if any file is changed, modified, added or deleted then it needs to be done at one brick only, the corresponding and respective changes at the other brick points would be automatically reflected and handled by GlusterFS as everything is just one particular file system.

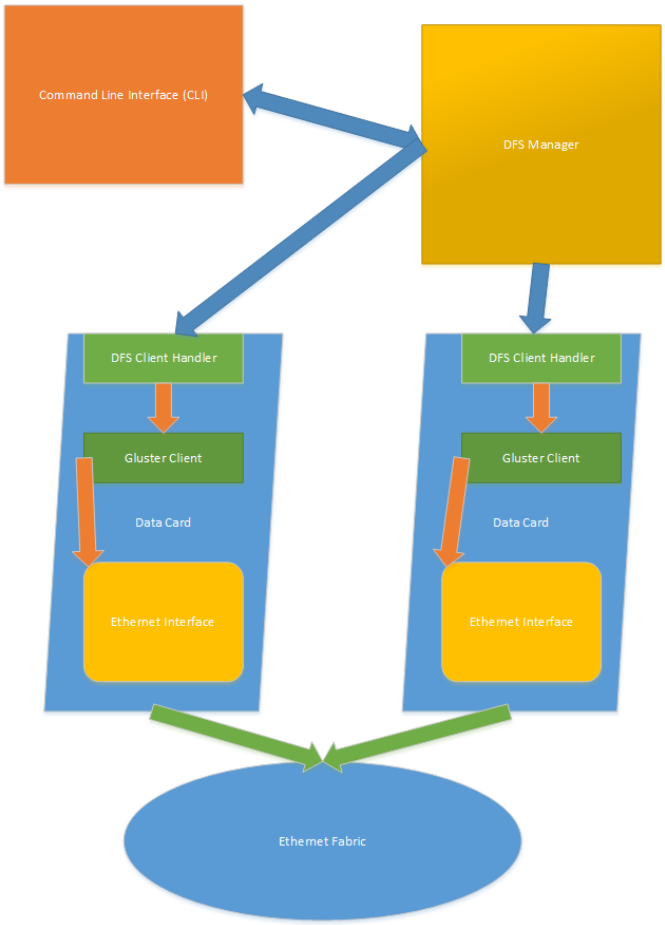

Figure 5. GlusterFS on data card

GlusterFS can't be used as it is on an edge router, as the original source code works on servers and computers with a different sort of addressing scheme. A router and its card have different addressing scheme and also extra features which are exclusively for networking. Hence GlusterFS is built (packaged) separately for edge router from its original source code, and a minimum of two such builds are required for DFS. One build is for RP card and another build is for data card. This build also accommodates some additional support for edge routers e.g., IPv6 address support. For a basic DFS this much is sufficient enough to function, however a data card can have more functions for which GlusterFS would be needed separately. These functions are for any application running separately on data card, which requires the resources of the data card. A data card has three main virtual machines running on them, application VM, platform VM, and main domain. Application VM is for running any application specific service which is discussed later. Platform VM is used for running the applications which fetch and store any data centric applications which are based on the data card. Main Domain can be regarded as the CPU of the data card of an edge router, it has the typical functionalities and CLI interpreters for any request running on a data card.

For the purpose of DFS only platform VM and main domain are of importance as DFS doesn't use any application specific processes, however any application running on data card might require GlusterFS. Hence GlusterFS needs to be packaged separately into four different packages for Platform VM, Main Domain, Application VM and one package running on RP card. This is depicted in Fig. 6. In case of an Edge Router simulator there is no Main Domain, as its running in a virtual environment and there is physically no existence of kernel.

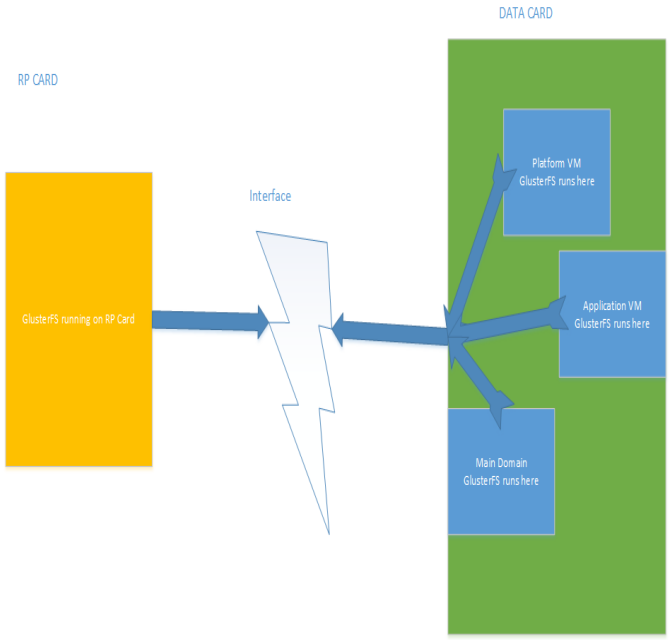

Figure 6. Gluster packages in edge router

6. APPLICATION OF DFS ON EDGE ROUTER

To understand applications of DFS on Edge Router, first a look into how edge routers deliver services to users in cloud computing scenario needs to be taken. Infrastructure as a

RESEARCH ARTICLE

service (IaaS), platform as a service (PaaS) and software as a service (SaaS) are the major three kinds of cloud computing services delivered to the users [23]. In IaaS computing resources are provided in the form of virtual machines (VMs) and a VM gives the user a view to a dedicated server. PaaS provides access to the resources in the form of application programming interface (API). Here the user doesn't have access to the system resources, and the resource allocation is done by the platform. SaaS provides application level software services from the internet. The three modes of cloud computing services along with their applications are described more clearly in Fig. 7, from where it is quite clear that DFS on edge router is applicable to the PaaS mode of cloud computing. The platform is itself the edge router, whose console output is the API which is provided to any particular user whenever he logs in to the edge router for accessing any data from the data cards. Live usage of DFS in the PaaS mode could be in analyzing network traffic itself, where the network data could be stored in the DFS and there are live examples of such applications too e.g., Service Aware Support Node (SASN). SASN is a PaaS application developed by Ericsson and it provides advanced functionality that makes it possible to identify and categorize different types of traffic. As part of a complete charging and policy control solution, the benefits include better service performance and improved network utilization. SASN uses DFS extensively for the obvious reasons mentioned above and runs on application VM of data card. Such similar applications running on edge routers are also on the phase of development if not already deployed by other such market leaders in the edge router business. For this IPv6 functionality is required which is incorporated additionally inside GlusterFS.

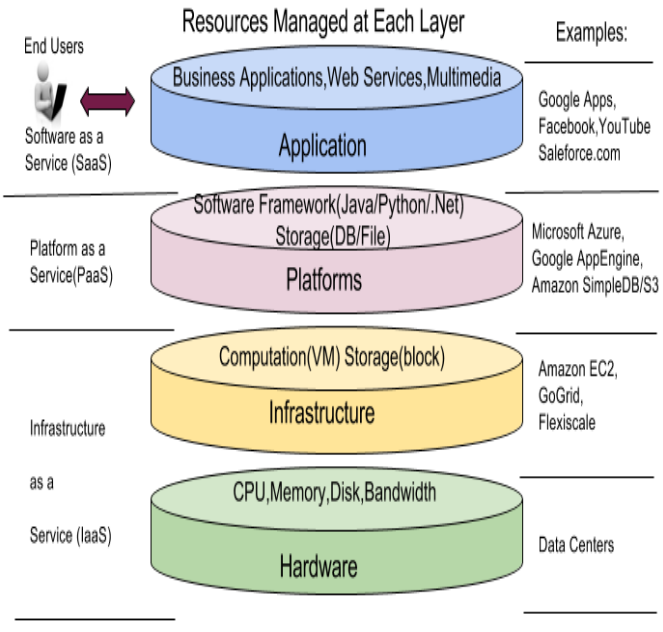

Figure 7. Services in cloud computing

DFS Manager can also be regarded as the master, and it pushes the work out to available task nodes (brick mount points at respective data cards) and it knows which node contains the data, and which other hosts are nearby. If the task cannot be hosted on the node where the data is stored, priority is given to the nodes in the same data card. In this way network traffic on the main backbone is reduced, further increasing throughput. If by chance DFS manager fails to function then after restart it can continue resuming the tasks from where it had left at the least point. Confidentiality (secure data access) is achieved by cryptographic protocols and auditability (whether security settings of applications has been tampered or not) is achieved by remote attestation techniques [20].

7. DFS SIMULATION

DFS is simulated and implemented in a step by step manner. Before one can jump onto testing DFS on an edge router, a thorough testing of GlusterFS was done on local laptop/server. This is achieved by booting up multiple Linux distributions on a virtual box where two Linux machines were made as servers and one as client and a replicated file system of our own was configured. The only issue here is that all the three Linux need to run simultaneously along with the parent windows laptop; hence RAM requirements are quite high, essentially 1GB RAM each for the three VMs and whatever is required for running the parent windows laptop. To be on safe side successful testing can be easily achieved in systems with 8GB RAM. Mount points on the servers were successfully accessed by clients and changes were made in the file system volume's bricks.

Edge router simulator software is available licensed from Cisco, Ericsson, Juniper, etc. Before DFS can be implemented on a real edge router, it needs to be tested on a simulator. A chassis is basically the physical externals of a router without any functionality. It becomes functional with the cards installed in it. The difference between a simulator and a real router chassis is that a simulator only provides the environment of the router, but there is no chassis or hardware or physical cards in a simulator. It also simulates the connections of the real world in exactly the same way how a router works but in a virtual environment. There is an assumption that is something works on simulator, then it would also work on the original and real router hardware. This statement is true to most of its extent; however certain functionalities might get affected. Hence the first step in a simulator includes creation of a chassis with the required number of cards and slots, and then starting the chassis on the simulator. The below source code shows creation of a chassis with one line/interface card, two data cards and one RP card and then inserting them into the chassis and starting the chassis. At the end there is a show command to check the status of the chassis created, so that one can verify all the cards and slots correctly. Data cards have storage disk capacities in them and in this simulation tow data cards with 50 GB capacity have

RESEARCH ARTICLE

been taken. In general the minimum size of a data card should be 10 GB i.e., 5 GB for each daughter card.

```
prepare
create chassis dfs_data_ch
create rpsw dfs_data_rp1
create data_card dfs_data_1 dfsdisk 50
create data_card dfs_data_2 dfsdisk 50
create ge-40-port  dfs_data_lc1
insert dfs_data_rp1 dfs_data_ch RP_CARD-1
insert dfs_data_1 dfs_data_ch 1
insert dfs_data_2 dfs_data_ch 2
insert dfs_data_lc1 GE_40_Port-1
start dfs_data_ch
show
```

A point to be noted is that a chassis can't be created without a line card, as no routing will happen without line card. Therefore a line card is also created in the chassis even though DFS doesn't require the usage of line cards. After the creating of chassis on simulator, the gluster packages are moved in the simulator separately for each of the components, and tested. Test results show functionalities of IPv6, aliasing, robustness and increase in stability of the system. Theoretically if there are *N* different file systems on different servers and each of their throughput capacity is *C* and file backup capacity be *B*. Then robustness of the file system is given by

$$R \propto B * C \quad (1)$$

However in case of a DFS with N servers in the same file system, the robustness is given by

$$R \propto N^2 * B * C \quad (2)$$

This is also shown by the simulator results, when tested against file redundancy, and error rates in loss of files in the file system. A factor of N^2 comes because each of the quantities, i.e., throughput capacity and file backup capacity get multiplied by a factor of *N*.

8. CONCLUSION

This paper talks about DFS implementation on an edge router using GlusterFS. It follows the client-server architecture with the various cards in the edge router being the servers and clients. DFS enhances file security, backup, ease of access and GlusterFS increases confidentiality and auditability. There are many options available for workflow storage in the cloud, and the performance of storage systems such as GlusterFS [24] is quite good. The bottleneck of reduced throughput is removed by using the GlusterFS architecture; the data card features and the stability and robustness of the system is increased. DFS finds lots of applications where data needs to be stored and accessed in an edge router and one of them being keeping a data record of network traffic and calculating tariffs according to it. Further applications of DFS would be studied, and ways to improve GlusterFS performance on edge router with other upgrades would be the course of study in further works. Also DFS would be implemented on a real chassis of an edge router, after more enhancement and betterment of performance and stability in simulator results. Applications of DFS on core routers would also be explored subsequently.

REFERENCES

[1] C.W. Zhao, J. Jegatheesan, and S. C. Loon, "Exploring IOT Application Using Raspberry Pi," in *International Journal of Computer Networks and Applications*, vol. 2(1), pp. 27–34, Feb 2015.

[2] J. H. Howard, M. L. Kazar, S. G. Menees, D. A. Nichols, M. Satyanarayan, R. N. Sidebotham, and M. J. West, "Scale and performance in a distributed file system," in *ACM Transactions on Computer Systems*, vol. 6(1), pp. 51–81, Feb 1998.

[3] O. Khan, R. Burns, J. Plank, W. Pierce, and C. Huang, "Rethining Erasure Codes for Cloud File Systems: Minimizing I/O for Recovery and Degraded Reads," in *FAST 2012*, pp. 1–14, 2012.

[4] D. Sun, G. Chang, S. Gao, L. Jin, and X. Wang, "Replication Strategy ti Increase System Availability in Cloud Computing Environments," in *Journal of Computer Science and Technology*, vol. 27(2), pp. 256–272, March 2012.

[5] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein, "Distributed Graphlah: a framework for machine learning and data mining in the cloud," in *Proceedings of the VLDB Endowment*, vol. 5(8), pp. 716–727, April 2012.

[6] D. Zhao, and I. Raicu, "Distributed File Systems for Exascale Computering," in *IEEE/ACM Supercomputing Doctoral Showcase*, pp. 1–2, 2012.

[7] A. Singh, T. W. Ngan, P. Druschel, and D. S. Wallach, "Eclipse attacks on overlay networks: Threats and Defenses," in *25th IEEE INFOCOM*, pp. 1–12, 2006.

[8] D. Xiao, C. Zhang, X. Li, "The Performance Analysis of GlusterFS in Virtual Storage," in *International Conference on Advances in Mechanical Engineering and Industrial Informatics*, pp. 199–203, 2015.

[9] M. Kumar, "Characterizing the GlusterFS distributed file system for software defined networks research," in *Proquest Dissertation and Theses*, pp. 1–43, 2015.

[10] E. Levy, and A. Silberschatz, "Distributed file systems: concepts and examples," in *ACM Computing Surveys*, vol. 22(4), pp. 321–374, Dec 1990.

[11] Hsiao, and Hung-Chang, "Load Rebalancing for Distributed File Systems in Clouds," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 25(5), pp. 951–962, 2012.

[12] B. Shao, H. Wang, and Y. Li, "Trinity: a distributed graph engine on a memory cloud," in *ACM SIGMOD International Conference on Management of Data*, pp. 505–516, 2013.

[13] L. Wang, J. Tao, R. Ranjan, H. Martin, A. Streit, J. Chen, and D. Chen, "G-Hadoop: MapReduce across distributed data centers for data-intensive computing," in *Future Generation Computer Systems*, vol. 29(3), pp. 739–750, March 2013.

[14] M. Vrable, S. Savage, and G. M. Voelkar, "BlueSky: a cloud-backed file system for the enterprise," in *10th USENIX conference on File and Storage technologies*, pp. 19–19, 2012.

[15] J. Zhang, G. Wu, X. Hu, and X. Wu, "A Distributed Cache for hadoop Distributed File System in Real-Time Cloud Services," in *ACM/IEEE 13th International Conference on Grid Computing*, pp. 12–21, 2012.

RESEARCH ARTICLE

[16] J. Liao, G. Xiao, X. Liu, and L. Zhu, "Dynamic Stripe Management Mechanism in Distributed File Systems," in *11th IFIP WG 10.3 International Conference NPC*, vol. 8707, pp. 497–509, 2014.
[17] G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B. P. Berman, and P. Maechling, "Data Sharing Options for Scientific Workflows on Amazon EC2," in *ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–9, 2010.
[18] E. Deelman, G. B. Berriman, B. P. Berman, and P. Maechling, "An Evaluation of the Cost and Performance of Scientific Workflows on Amazon EC2," in *Journal of Grid computing*, vol. 10(1), pp. 5–21, March 2012.
[19] A. Davies, and A. Orsaria, "Scale out with GlusterFS," in *Linux Journal*, vol. 2013(235), 2013.
[20] Q. Zhang, L. Cheng, R. Boutaba, "Cloud computing: state of the art and research challenges," in *Brazilian Computer Society, Journal of Internet Server Application*, vol. 1, pp. 7–18, April 2010.
[21] W. Louati, B. Jouaber, and D. Zeghlache, "Configurable software-based edge router architechture," in *Computer Communications*, vol. 28(14), pp. 1692–1699, September 2005.
[22] I. Raicu, I. T. Foster, and P. Beckman, "Making a case for disturbed file system at Exascale," in *Third International Workshop on Large-scale system and application performance*, pp. 11-18, 2011.
[23] A. Beloglazoy, S. F. Piraghaj, M. Alrokayam, and R. Buyya, "Deploying OpenStack on CentOS Using the KVM Hypervisor and GlusterFS Distributed File System," Cloudbus.org, 2012.
[24] Gluster, Inc., "GlusterFS," in *http://www.gluster.org*

Author

Souryendu Das was born in Howrah, India, in 1994. He is doing his B.E.(Hons) degree in Electrical and Electronics Engineering from BITS Pilani K. K. Birla Goa Campus and will graduate in 2016. His current research interests include antenna design, IP & Networking, Internet of Things, Wireless Communication, Cloud Computing, and network security. He has published 2 papers in the field of antenna design, and this is his 3rd publication. He works with Ericsson R&D, in the field of Cloud & IP.