Indian Journal of Science and Technology, *Vol 10(29), DOI: 10.17485/ijst/2017/v10i29/116504, August 2017*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Experimental Setup of Logs Analysis on Distributed File Systems using MapReduce

Madhavi Vaidya[1]* and Shrinivas Deshpande[2]

[1]Department of Computer Science, Vivekanand Institute of Education Society's Arts, Science and Commerce College, Mumbai – 400071, Maharashtra, India; vamadhavi@gmail.com
[2]Department of Computer Science, HVPM, Amravati – 444605, Maharashtra, India

Abstract

The computing world is undergoing a drastic change from traditional non-centralized distributed system architecture to typical parallel and pseudodistributed nodes. Such nodes are scattered across different geographic areas to a centralized cloud computing architecture where data transformation and computations are operated somewhere on any node. Data centres owned and maintained by third party or a cloud can be formed and maintained using the number of physical machines. These machines can be of different configurations or using virtual machines on a shared LAN to communicate with each other. It has been experienced that there is always a difference in performance when the MapReduce program is run on various input statements and different Distributed File System (DFS).

The use case on data generation from the Security Logs from the server machine has been taken into consideration. In our case to run this program, the mini-cloud has been configured on LAN. The outcome of analysis has been carried out using a MapReduce program, tested on the data generated from the security software, have been tested on various DFS like Hadoop, Ceph, Glusterfs and the Zfs. These DFS installed on infrastructures like Single Virtual Machine, a cluster of Virtual Machine and the minicloud. It has been noticed that MapReduce is the best technique for the logs analysis and computations.

Keywords: Ceph, Gluster, Hadoop, Logs, MapReduce

1. Introduction

Distributed File Systems have been developed to support the style of data accessing in the manner of write-once, producing high-throughput and does parallel streaming. These include the I/O subsystem such as Glusterfs, Zfs, Ceph and the Hadoop Distributed File System (HDFS).

The software framework MapReduce has been used for analysis and computation. It is an open-source implementation; Hadoop is deployed in the Cloud to generate the availability of virtualized resources and pay per-usage cost model of cloud computing. Logs are a perfect fit for analysis using MapReduce for few reasons. First, logs usually follow a certain pattern and they are unstructured in nature, however it isn't significant to use a RDBMS to handle them and may require changes to the structure of the database to compute them. Secondly, logs represent a use case where scalability not only matters but is a key to keep the system sustainable. As services grow, so does the amount of logs and the need of getting some important results out of them. In this paper the study has been performed to show how various Distributed File Systems give the output of processing the security logs generated from antivirus work on a mini cloud which has been maintained on few machines in small setup.

Instead of using the pay-per-use cloud model, we have developed a cloud on few machines which are in local area network. MapReduce is a software framework

*Author for correspondence

for parallel data intensive computations popularized by Google [recent paper]. There are several implementations available besides Google's original system, but the most popular one is Hadoop[1], developed by Apache[2].

A cloud is typically comprised of inter-connected, virtualised computers coupled with security and a programming model. From an application developer's perspective, the programming model and its performance are important criteria to select a cloud environment. A Private Cloud is a collection of virtualized physical hardware that has added services such as catalogues of software or defined platforms that a customer can control.

Distributed File System is an extension of file system which manages files and data on multiple storage devices. They provide good performance and reliability using various modern techniques.

But in case of failure or heavy load very few Distributed File Systems provide location transparency and redundancy to improve data availability. It has been noticed that some of the Distributed File Systems create bottlenecks too. Significant challenges for such a Distributed File System are extended to a large number of storage nodes and providing reasonably degraded operations when there are chances of hardware failure[3]. Shared-nothing compute clusters is a famous platform for scalable data-intensive computing. Shared-nothing compute clusters is a famous platform for scalable data-intensive computing.

Distributed File Systems have been developed to support this style of data accessing in the manner of write-once, producing high-throughput and does parallel streaming. These include the I/O subsystem such as glusterfs, zfs, Ceph and the Hadoop Distributed File System (HDFS). In this paper we have taken up the security logs analysis on Distributed File System using MapReduce algorithm. The paper's flow has been arranged in such a way that, comparative approaches of Log analysis have been elaborated in the second section. In Section 3, the elaboration on the various steps to perform analysis has been listed down. Section 4 contains the architecture and implementation issues of the said Distributed File Systems is been provided. Section 5 provides the elaboration on MapRedue implementation on DFS and experimental approaches of MapReduce implementation on mentioned Distributed File Systems and the role of MapReduce on those Distributed File Systems. The section 6 provides the discussion on the analysis and results along with the explanation on experimental approaches and finally Section 7 would be closed with the concluding remarks.

2. Comparative Approaches of Security Log Analysis

It has been mentioned in[4] that the use of processing web log file is problematic on single processor programs because of the big sizes. The authors mentioned that there is a necessity of using a parallel processing approach to work effectively on massive amount of large datasets. It has been suggested in[5], for filtering and aggregation of jobs the MapReduce method takes less time than Hash algorithm. The hash algorithm takes more time because it has to compute the hash values for both map and reduce functions and store it in a hash table. Whenever the job is executed, the hash value is retrieved from the hash table and combines it with the number of tasks it carries out and generates the result. However, the MapReduce technique doesn't need to calculate the hash value and just it execute the mapping and reducing of the data. However, as it has been suggested earlier it isn't significant to use a RDBMS to handle them and may require changes to the structure of the database to compute them as the logs are unstructured in nature.

As the quick development of internet technology and digital data in recent years is showing explosive growth, mobile devices and cloud computing environment is generating huge data logs. Traditional text software's and relational database technology has been facing bottleneck and the presented results are not up to the mark[6].

In this paper we have taken into consideration, the analysis of heavy log data generated from the security software. In order to analyse the log data, we obtain required knowledge from this log data with the help of data analysis. Log files are being produced with the software extremely quickly as and when the antivirus software is run every day on the number of machines in LAN. In a day, one data centre is able to produce ample log data. The size of the sets of data is very large. We need a parallel processing system as well as a dependable data storage mechanism to perform the analysis of such huge datasets. The Hadoop framework gives us dependable data storage with the help of Hadoop Distributed File System as well as MapReduce calculating standard that acts as a parallel operating setup over huge sets of data. A Hadoop Distributed File System splits the initial data as well as provides the initial data fragments across various computers across HDFS cluster.

This problem statement analyses the heavy data workload and following is the outcome –

- Which all users are working in LAN,
- How many number of machines are connected,
- URLs accessed by the users,
- How many times a user has accessed a specific website.

3. Various Steps to Perform Analysis on Securitylogs using Hadoop

The various steps carried to perform an analysis on Security Logs, mentioned in Table 1 using Hadoop are as follows:

3.1 Data Pre-processing

All the records to be processed are dispersed across the different Application Servers. The security logs are captured from various machines connected in LAN.

3.2 Upload

The data is a typical **b**lock size is uploaded into the HDFS specified directories through various commands.

3.3 Hadoop Processing

The data is not changed after the security log file is stored in HDFS. The system functions are divided into two categories such as batch analysis and interactive input conditions. The main purpose of batch analysis is to estimate traffic statistics and conduct multi-dimensional analysis[7].

We wrote a specific MapRedce function to count and find how many users have visited which all websites at what time and how much time.

3.4 Analysis

The count of how many users visited a specific URL in a day has been maintained.

4. Study of Architecture and Implementation Issues

4.1 Hadoop

Hadoop is an Apache open source project. Its architecture supports the storage, transformation and analysis of very large datasets. Hadoop is a massively parallel processing solution and characterized as a massive I/O platform. Massive can be a relative term which changes and dependent on various infrastructure used. Two critical components of Hadoop are the Hadoop Distributed File System (HDFS)[8] and the MapReduce algorithm[9]. Both the components allow efficient parallel computation on very large datasets. HDFS is based on Google File System i.e. GFS.

On the top of the various DFS mentioned below the Hadoop Distributed File System has been installed using plugins and MapReduce programs were executed for various use cases.

4.2 Glusterfs

Glusterfs is an open source model. Glusterfs is designed to provide a scale-out architecture for both performance and capacity and is able to scale up or down. Operation of Glusterfs is done in user space so it makes installing and upgrading Glusterfs significantly easier. Storage system servers can be added or removed on-the-fly with data automatically rebalanced across the cluster. Glusterfs locates data algorithmically on the basis of the path name and the file name, any storage system node and any client requiring read or write access to a file in a Glusterfs storage cluster performs a mathematical operation that calculates the file location[10]. Experimental work on installation and configuration of Glusterfs is found to be smooth[11].

Recent introduction of GlusterFSHadoop plugin (HDFS interface) enables Hadoop applications to take advantage of unstructured file and object data hosted

Table 1. Security logs

21 Jan 2017 (12:49:09)	HCL-PC32	CS	https://connect.facebook.net	Social Networking
21 Jan 2017 (12:49:09)	HCL-PC32	CS	platform.twitter.com/widgets.js	Social Networking
21 Jan 2017 (12:49:00)	HCL-PC32	CS	graph.facebook.com/	Social Networking
21 Jan 2017 (12:48:11)	HCL-PC32	CS	https://connect.facebook.net	Social Networking
21 Jan 2017 (12:48:11)	HCL-PC32	CS	platform.twitter.com/widgets.js	Social Networking
21 Jan 2017 (12:48:08)	E-RESOURCE6	WORKGROUP	graph.facebook.com/	Social Networking
21 Jan 2017 (12:48:02)	HCL-PC32	CS	graph.facebook.com/	Social Networking
21 Jan 2017 (12:48:01)	HCL-PC32	CS	https://api.ciuvo.com	Shopping
21 Jan 2017 (12:48:01)	E-RESOURCE6	WORKGROUP	platform.twitter.com/widgets.js	Social Networking
21 Jan 2017 (12:48:00)	E-RESOURCE6	WORKGROUP	https://connect.facebook.net	Social Networking

on Gluster volumes in place. On the Glusterfs cluster, Glusterfs 3.7.17 build is been installed. On the top of it, glusterfs-Hadoop plugin[11] has been installed. In order to configure the said plugin, we made specific type of deployment. Firstly, HadoopJobTracker and TaskTrackers are installed on Glusterfs server's trusted storage pool for a given Glusterfs volume. The JobTracker used the plugin to query the job input files in Glusterfs, they determine the file placement as well as the distribution of file replicas across the cluster. The TaskTrackers used the plugin to control a local fuse mount of the Glusterfs volume in order to access the data required for the tasks.

4.3 Zfs

The Z file system is built in 2004 by Sun Microsystems which is free and open source logical volume manager for using it in their Solaris operating system[12].

In ZFS, the basic storage unit is a "storage pool" or zpool. to a single virtual drive. The storage pool understands physical details about the various storage hardware like devices, layout, redundancy, and so on and presents itself to an Operating System as a large "data store" that can support one or more file systems[13]. It has been an experience at the time of setup, for configuring Zfs. One needs to have a robust infrastructure. Otherwise might create a bottleneck for performance. But running MapReduce on Zfs was smoothly performed[14]. This is the way by which the implementation of Hadoop Distributed File System. It has been observed that MapReduce works well on all the Distributed File Systems.

4.4 Cephfs

Ceph is distributed file system and is an object-based, open source, parallel file system[15,16] whose design is based object-based storage, which divides the traditional file system architecture into two components viz. a client component and a storage component and it increases scalability. A Ceph OSD as a process that runs on a cluster node and uses a local file system to store data objects.A CRUSH (Controlled Replication Under Scalable Hashing)[17] algorithm to compute on demand where the data should be written to or be read from.

One mon node and 3 Object Storage Device (OSD) nodes have installed on virtual machines. It has been found that during recovery mon nodes and OSDs nodes need significantly more RAM. If the user has decided to implement Ceph on Virtual Machines, one will need to ensure that those other processes leave sufficient processing power for Ceph daemons. The additional CPU-intensive processes have to be run on separate hosts[18]. After implementation, it has been observed that data storage has to be planned carefully, otherwise system gets slower. Once gets installed, achieving the active+clean states for placement groups is the final state then and there the user can start executing commands on Cephfs. As a part of this implementation, Ceph version 9.2.1 has already been installed. We configured the cephfs-Hadoop plugin on underlying distributed file system of Ceph. The ceph-hadoopplugin[18] cephfs-hadoop.jar is been configured properly. It has been observed from our implementation that Ceph and Hadoop 1.x and 2.x work satisfactorily together.

It has been studied from the literature present, which DFS will give us good performance. Most distributed storage systems do not give adequate performance with large files or sometimes few don't give the same performance with either type of files. It has been studied that Glusterfs does a decent job for both small and large objects[19]. It has been mentioned in the literature that Hadoop can also be an alternative for storing large files.

5. Mapreduce Implementation on DFS

Hadoop is an implementation of the abstract idea called "map-reduce". That is, it is a programming model for a certain type of calculation that consists of a parallel "map" step, followed by some form data gathering that is "reduce". MapReduce consists of a first "map" step, in which a master node (acting as a coordinator) divides the initial input into smaller chunks, each to be used in its own separate sub-task, and distributes them to worker nodes. Each worker performs the tasks it was given, and reports its completion to the master.

At the next "reduce" step, the master node schedules one or more tasks on workers to perform join operations on the map outputs, to combine initial problem to be solved. In the cloud environment, MapReduce relies on a shared, parallel file system at each step like Google File System[20,21].

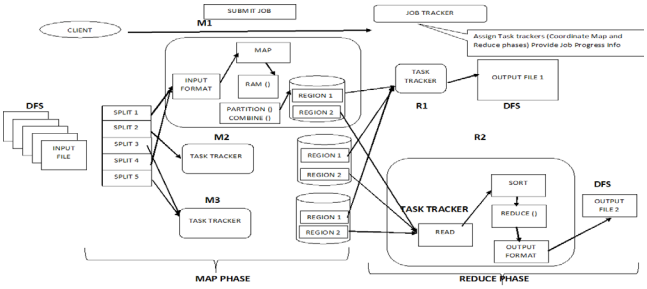

Figure 1. MapReduce architecture[6].

6. Experimental Approaches , Discussion and Analysis of DFS Implementation

6.1 Particulars of Implementation

For this article, the MapReduce algorithm was implemented on a system using:

- Off-the shelf commodity machines.
- Hadoop 1.2.1, 1 NameNode and 3 DataNodes.
- 4 GB RAM on Master node and 1 GB RAM on slave nodes.
- Eclipse IDE 3.0 .
- Cent OS7.0/Ubuntu 8.2 or above.

The security log analysis is carried on various infrastructure approaches viz. single machine as given in Figure 2, mini-cloud and the cluster of Virtual Machines as mentioned in Figures 3 and 4, which has been performed on the commodity PCs. The said Distributed File Systems named Cephfs, Hadoopfs, Glusterfs and Zfs have been installed on the given infrastructure. Memory sized used n the virtual machines is the 1 GB and for Cephfs it was 2 GB. The commodity PCs memory size is of 4 GB each. On the remaining three machines i.e. Zfs, Glusterfs and Hadoopfs the Hadoop has been installed using various plugins and it ran smoothly on these Distributed File Systems.

The analysis is carried out and the few attributes like, the MapReduce program started execution, how much total time taken for execution, CPU time taken to execute the whole process, how many records were given as an input and how much heap memory was required to execute on all types of infrastructure.

It has been observed that on single machine the total execution time taken is much more than the other three DFS where HDFS is installed for the same number of records in the input file, the reason behind it is, the resources are divided among many running processes on single machine.

But in case of Mini-cloud and the VM Cluster, the total execution time taken to execute the whole MapReduce program on Plain Hadoop is much more less than the Glusterfs and the Cephfs along with the CPU time taken. However, it has been noticed from the implementation work thatCeph was definitely more unstable than HDFS and it required lot of tweaking and fixing.

The Zfs is lightweight DFS, it has taken very less execution time, minimal CPU time and less heap memory than all the other three DFS viz. Hadoopfs, Cephfs and

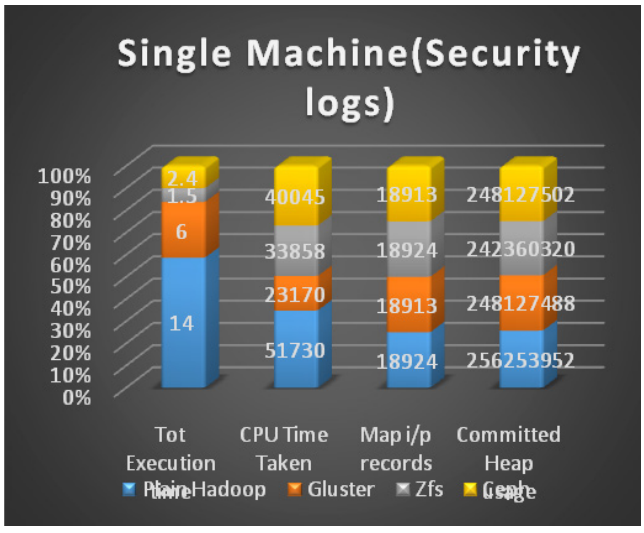

Figure 2. Security Log Analysis on Single VM.

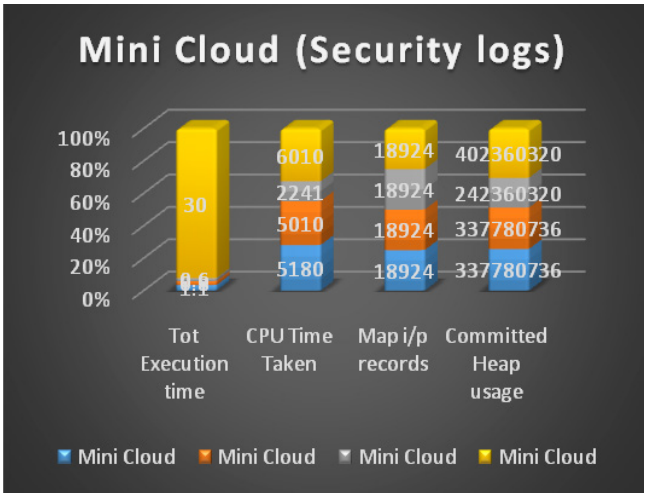

Figure 3. Security Logs on mini-cloud.

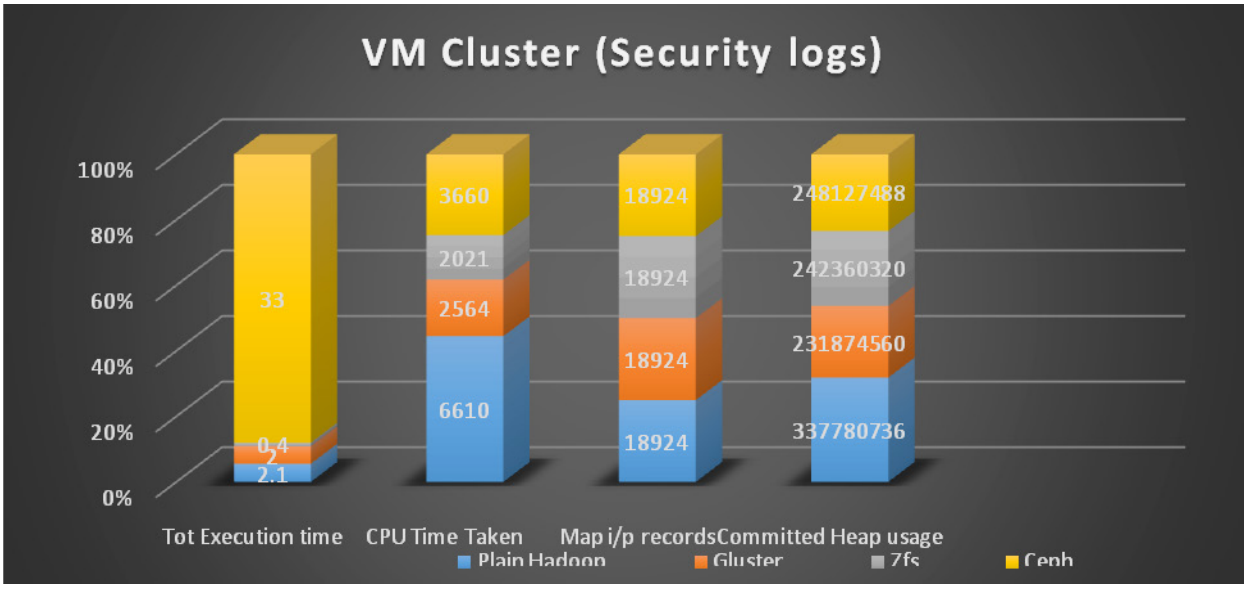

Figure 4. Processing of Security logs on VM cluster.

the Glusterfs in all the type of approaches like VM cluster, mini-cloud and the single VM machine.

7. Conclusion

It has been observed that when the log file is processed on the various infrastructure approaches like single VM, cluster of VMs and the mini-cloud then, the CPU time is always taken more on the single VM than the mini cloud. The reason behind this is all the resources are available for the running process on the mini-cloud. In case of single VM the resources like CPU time, memory and secondary storage device are distributed and divided among the number of processes running along with single VM. Overall it has been noticed that the MapReduce program run Ceph on any type of infrastructure approach took up more time on than all other DFS. The total time taken for execution and CPU time on mini-cloud was much less in all approaches of infrastructure than the other two approaches. Future work may include a comparative approach on the other types of frameworks of Hadoop by which data analysis is done satisfactorily.

8. Acknowledgement

We would like to thank the Head of the institution for granting permission to make use of the web security logs generated from the institution's server.

9. References

1. Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. Proceedings of OSDI '04: 6th Symposium on Operating System Design and Implementation; San Francisco. 2004.
2. Apache Hadoop. See website at: Crossref
3. Sacerdoti FD. Performance and fault tolerance in the store-torrent parallel file system. arXivPublications; 2010. p. 1–13. PMCid: PMC3092554.
4. Savitha K, Vijaya MS. Mining of web server logs in a distributed cluster using big data technologies. International Journal of Advanced Computer Science and Applications (IJACSA). 2014; 5(1):137–42.
5. Zaharia M, Konwinski A, Joseph AD, Katz RH, Stoica I. Improving MapReduce performance in heterogeneous environments. Osdi. 2008 Dec; 8(4):7. PMCid: PMC2627897.
6. Savant PD, Bhattacharyya D, Kim TH. Hadoop based Weblog Analysis: A Review. International Journal of Software Engineering and its Applications. 2016; 10(6):13–30. Crossref
7. Wang CH, Tsai CT, Fan CC, Yuan SM. A Hadoop based weblog analysis system. IEEE 2014 7th International Conference on Ubi-Media Computing and Workshops (UMEDIA); 2014 Jul. p. 72–7. Crossref
8. Shvachko K, Kuang H, Radia S, Chansler R. The Hadoop Distributed File System. 2010 IEEE 26th symposium on Mass Storage Systems and Technologies (MSST); 2010 May. p. 1–10. Crossref

9. Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. Communications of the ACM. 2008; 51(1):107–13. Crossref
10. Source – An Introduction to Gluster Architecture. Cloud Storage for the Modern DataCentre. 2011.
11. Source - Crossref
12. Rodeh O, Teperman A. zFS-a scalable distributed file system using object disks. Proceedings of 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies, (MSST 2003); 2003 Apr. p. 207–18.
13. Teperman A, Weit A. Improving performance of a distributed file system using OSDs and cooperative cache. IBM Journal of Research and Development. 2004.
14. Source - Crossref
15. Weil SA, Brandt SA, Miller EL, Long DD, Maltzahn C. Ceph: A scalable, high-performance distributed file system. Proceedings of the 7th Symposium on Operating Systems Design and Implementation, USENIX Association; 2006 Nov. p. 307–20.
16. Maltzahn C, Molina-Estolano E, Khurana A, Nelson AJ, Brandt SA, Weil S. (2010). Ceph as a scalable alternative to the hadoop distributed file system. login: The USENIX Magazine. 2010; 35:38–49.
17. Weil SA, Brandt SA, Miller EL, Maltzahn C. CRUSH: Controlled, scalable, decentralized placement of replicated data. Proceedings of the 2006 ACM/IEEE Conference on Supercomputing ACM; 2006 Nov. p. 122. Crossref
18. Source - Crossref
19. Glusterfs troubleshooting - Crossref
20. MapReduceTutorial - Crossref
21. Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. Communications of the ACM. 2008; 51(1):107–13. Crossref