Review of Limitations on Namespace Distribution for Cloud Filesystems

Genti Daci[1], Frida Gjermeni[1]

Polytechnic University of Tirana, Faculty of Information Technology, Tirana, Albania
gdaci@abcom.al, frida_gjermeni@hotmail.com

Abstract. There are many challenges today for storing, processing and transferring intensive amounts of data in a distributed, large scale environment like cloud computing systems, where Apache Hadoop is a recent, well-known platform to provide such services. Such platforms use HDSF File System organized on two key components: the Hadoop Distributed File System (HDSF) for file storage and MapReduce, a distributed processing system for intensive cloud data applications. The main features of this structure are scalability, increased fault tolerance, efficiency and high-performance for the whole system. Hadoop supports today scientific applications, like high energy physics, astronomy genetics or even meteorology. Many organizations, including Yahoo! and Facebook have successfully implemented Hadoop as their internal distributed platform. Because HDSF architecture relies on a master/slave model, where a single name-node server is responsible for managing the namespace and all metadata operations in the file system. As a result this poses a limit on its growth and ability to scale due to the amount of RAM available on the single namespace server. A bottleneck resource is another problem that derives from this internal organization. By analyzing the limitations of HDSF architecture and resource distribution challenges, this paper reviews many solutions for addressing such limitations. This is done by discussing and comparing two file systems: Ceph Distributed File System and the Scalable Distributed File System. We will discuss and evaluate their main features, strengths and weaknesses with regards to increased scalability and performance. Also we will analyse and compare from a qualitative approach the algorithms implemented in these schemes such as CRUSH, RADOS algorithm for Ceph and RA, RM algorithms for SDFS.

Keywords: Algorithm, cloud filesystems, Hadoop distributed filesystem, performance, Ceph File System, Scalable Distributed File System

1 Introduction

Apache Hadoop is a famous, suitable platform for storing, processing and transferring large amounts of data sets in a cloud environment. The compact organization

S. Markovski, M. Gusev (Editors): ICT Innovations 2012, Web Proceedings, ISSN 1857-7288

realised by Hadoop Distributed File System [2] for storing data of different complexity and variety and MapReduce [3] for processing intensive distributed data applications, give HDSF a fundamental role in adapting the changing way of storing and processing information today. Projected to run on low cost hardware, HDSF file system is highly fault tolerant due to its replication policies for recovering data blocks in case of hardware failures or data corruption. Efficiency and high performance are provided adding to replication policies the ability to access data files in a stream, write-once-read-many [2] manner. One of the main features of HDSF is its ability to scale through hundreds of nodes in cloud computing systems. This is achieved by decoupling or separating metadata from data operations since metadata operations are usually faster than the data ones. MapReduce exploits the scalability of the file system [7] offering computation movement to interested data for a certain application.

Several important fields benefit from HDSF services such as astronomy, mathematics, genetics [9] or even economy used in online commerce and fraud detection [7], etc. Also important organizations like Yahoo!, Facebook and eBay have successfully implemented Hadoop as their internal distributed platform.

HDSF file system structure organization is based on the master/slave model where only a single NameNode server (master) menages the entire namespace, metadata operations and access to data files from clients of the file system. Data files are split into data blocks that are stored in DataNodes, usually one for every node in a cluster environment. Because of speed increasing, the whole file system metadata is stored in the NameNode single server's main memory. This implies a limit on the capacity of the system to grow and scale into a larger number of nodes than those supported by the available amount of RAM present in the NameNode server. Also the presence of this single server as the only metadata source creates a bottleneck resource and a single point-of-failure if it goes down. The same scenario can happen in the case where the number of contemporary requests for metadata operations that address the namespace server, reaches the limits of its capacities. Also the network bandwidth available for exchanging data between the actors in the file system is a limiting factor regarding scalability and overall performance.

The contributions of this paper are: 1) review of Ceph distributed file system and the new Scalable Distributed File System with regards to growth and scalability limits and 2) compare their features, strengths and weaknesses 3) analyse and compare from a qualitative approach the algorithms implemented in these schemes such as CRUSH, RADOS algorithm for Ceph and RA, RM algorithms for DSFS.

The rest of the paper is organized in the following order: section 3 will describe the limitations and solutions of the existing HDSF architecture, regarding scalability improvement. In section 4, features and comparison of Ceph file system and DSFS file system will be discussed and in section. Also algorithms implemented in Ceph such as CRUSH, RADOS and those implemented in DSFS such as RA, RM will be compared from a qualitative approach.

A lot of work has been made during the last years on the field of distributed networked file systems. Andrew File System (AFS) [15] is a distributed file system that presents several advantages over traditional networked file systems, especially in the fields of security and scalability. It uses a cluster of servers to provide a homogeneous

and location transparent namespace to all the clients who want to interact with it. No matter the fact that it can handle large amounts of file requests it may not be convenient for large scale and scientific operations like those handled by HDSF.

The Networked File System (NFS) [16] is another way to share files on a network. It has the disadvantage that it oversimplifies everything by making a remote file system appear as a local one. In situations like those handled in HDFS it wouldn't be appropriate or even reliable.

Other file systems used in distributed enviroments are Frangipiani [17] and InterMezzo [18]. Frangipiani is a distributed file system where disks on multiple machines are stored and managed in a single shared pool. Due to a simple internal structure system recovery and load balancing are easily managed. In InterMezzo, local file systems serve as server storage and client cashe and make the kernel file system driver a wrapper around the local file system. These file systems do not have a good resouce allocation regarding dynamic link, storage and proccesing capacities.

Another very well-known file system is the Google File System (GFS) [6] which resembles to HDSF and has recently announced a namespace distribution even if no information for the public hasn't been provided yet.

2 Related Work

A lot of work has been made during the last years on the field of distributed networked file systems. Andrew File System (AFS) [15] is a distributed file system that presents several advantages over traditional networked file systems, especially in the fields of security and scalability. It uses a cluster of servers to provide a homogeneous and location transparent namespace to all the clients who want to interact with it. No matter the fact that it can handle large amounts of file requests it may not be convenient for large scale and scientific operations like those handled by HDSF.

The Networked File System (NFS) [16] is another way to share files on a network. It has the disadvantage that it oversimplifies everything by making a remote file system appear as a local one. In situations like those handled in HDFS it wouldn't be appropriate or even reliable.

Other file systems used in distributed enviroments are Frangipiani [17] and InterMezzo [18]. Frangipiani is a distributed file system where disks on multiple machines are stored and managed in a single shared pool. Due to a simple internal structure system recovery and load balancing are easily managed. In InterMezzo, local file systems serve as server storage and client cashe and make the kernel file system driver a wrapper around the local file system. These file systems do not have a good resouce allocation regarding dynamic link, storage and proccesing capacities.

Another very well-known file system is the Google File System (GFS) [6] which resembles to HDSF and has recently announced a namespace distribution even if no information for the public hasn't been provided yet.

3 HDFS and Scalability

The two main actors in HDSF architecture execute different operations over the file system. Operations like open close and rename of the files and directories of the namespace are executed by the NameNode server. Whereas operations like read and write, that in fact are client request operations over the data As mentioned above there are serious limitations regarding scalability, caused by the single NameNode server and its amount of RAM for namespace storage. Furthermore, from works by Shvacko *at al*. [10], 30% of the processing capacity of the name-node capacity is dedicated to internal load, such as processing block reports and heartbeat [2]. The remaining 70% of capacity processing can easily be expired by 10.000 writers on a 10,000 node HDSF cluster as calculated by Shvacko *at al*. [10]. Consequently a bottleneck and name-node saturation would happen. Any intention to increase the number of DataNode servers, files stored in the DataNodes or clients requests beyond the supporting abilities of the single namespace server, causes a single point-of-failure in the file system.

When a client requests access to a file for a read or write operation, he first contacts the namespace server for the location of the data blocks forming the desired file on the datanodes. Than the transfer to or from the DataNodes takes place, as described in Fig.1.

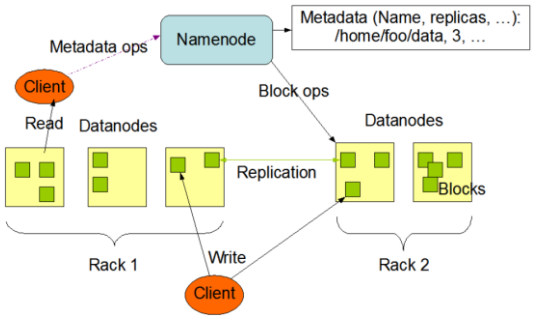

Fig. 1. HDSF architecture by [1]

Eventually the distribution of the namespace server and the decentralisation of the existing master/slave architecture represents an inevitable fact. That said, not all the approaches that attempt to introduce several namespace servers are successful. Such an example is Federation [8], where the number of clients and the amount of data stored increases, due to the presence of several NameNodes. But the static partitioning prevents from redistribution in the case when the growth of a volume reaches the NameNode capacities, so again we arrive at a lack of scalability.

No matter the system's necessity of distributing the namespace server into a cluster of NameNode servers, the costs of money and time (even years for building such a distributed environment) it takes, have to be considered. This means we have to think

about a distributed namespace rather than a distributed environment of Namenode servers. Hbase [1] is such an approach, but even here the algorithms that have to be implemented for the namespace partitioning and the issue of renaming represent hot topics of discussion.

4 Solutions Based on Distributed File Systems

In this section we analyse two file systems that address HDSF limtitations above discussed. Ceph Distributed File System and Scalable Distributed File System (SDSF) main features, strengths and weaknesses are specified regarding namespace distribution, scalability improvement, higher performance and increased fault tolerance.

4.1 Ceph [11]: How and Why

Ceph is an open-source platform [6]. How to address HDSF scalability limitations using Ceph? "Simply" by using Ceph as the file system for Hadoop platform as explained in [6].

Ceph is an object–based parallel file system, which internal organization [6] fulfills HDSF main requirements for playing the role of its file system.

One of the main strengths of Ceph is the distribution of the entire metadata operations and namespace file system into a cluster of metadata servers (MDS). The process of distribution is realised in a dynamic way thanks to the Dynamic Subtree Partitioning algorithm. Each MDS measures the popularity of metadata using counters. Any operation that affects the actual i-node increments the counters and all of it ancestors till the root. This way a weighted MDS regarding the recent local distribution, is provided. The achieved values are periodically compared and as e result appropriately sized subtrees are migrated, to keep the workload evenly distributed.

Besides workload balancing, also the ability to prevent hot spot points for metadata operation requests is achievable. Surely, the scalability and overall performance increase.

When a client wants to interact with the file system, first he requests the MDS as can be seen by Fig.2. As a response the MDS returns the i-node number, replication factor and the striping strategy of the requested file. Than the client can calculate on its own the placement of the data objects and replicas of the requested files.

OSDs (Object Storage Device) are the storage intelligent devices, that store both data and metadata, but due to RADOS (Reliable Autonomic Object Storage) [13] (also discussed in section IV) they are seen as a single logical object store by the clients and MDSs (as explined in Fig.2). RADOS represents a key component for Ceph because it provides every communication iniciatior with a cluster map containing information about OSDs, thier status and CRUSH (Controlled Replication under Scalable Hashing) [12]. CRUSH [12] resolves the problem of data distribution and location. It eliminates allocation tables by giving every party the ability to

calculate the data location.

No matter the strengths that RADOS and CRUSH offer, Ceph still remains in an experimental phase. Besides the advantages in scalability, it introduces a higher degree of complexity to the the file system, since it inserts objects, as storing entities. The abstraction that from it derives, it is not to be let out of attention.

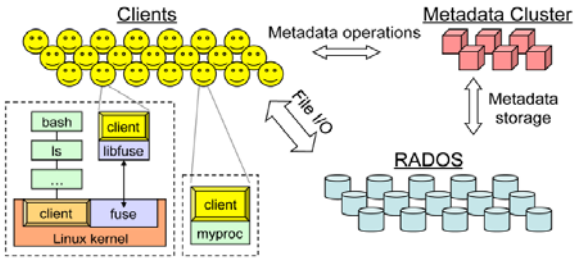

Fig. 2. Internal Architecture of Ceph by [11]

4.2 Scalable Distributed File System: Another Alternative

Scalable Distributed File System (SDSF) [4] is another distributed file system that attempts to resolve the limitations of HDSF platform. One of the main innovations of SDFS is the introduction of a front end light server (FES), as a connection point between the NameNode servers (NNS) and the client requests. The NNS handling the current request is chosen by hashing the request ID. To a certain point of view, this solution distributes metadata operations workload, but how many client requests can handle at once the FES [4]? Wouldn't this be another critical point for the file system overall performance ? No matter the fact that the presence of this front end server does not cause a resource bottleneck, since it is stateless, what about the load of a single NNS [4]? No information is provided about how many contemporary requests can a NNS handle. All the questions above raised represent fragile points that need further development or alternative solutions.

A strength point in SDFS is the introduction of a communication protocol that can achieve a better link utilisation during the operations with the file system. This is realised by the Resource Allocator (RA) and Resource Monitor (RM) algorithms, that will be further discussed in the next section. After a client connects to the FES, it is its rensponsibility to find the appropriate NNS to handle the client's request. As can be seen in Fig.3 the NNS finds the best block server (BS) after consulting the RA and the RM. The RA acts like a software router helped by the RM which gives the rmetric of every BS it is associated to. This scenario provides an increased scalability that derives from full link utilisation and rational resource allocation. By the end of this section we can conclude that namespace distribution is the key to scalability improvement regarding cloud file systems.

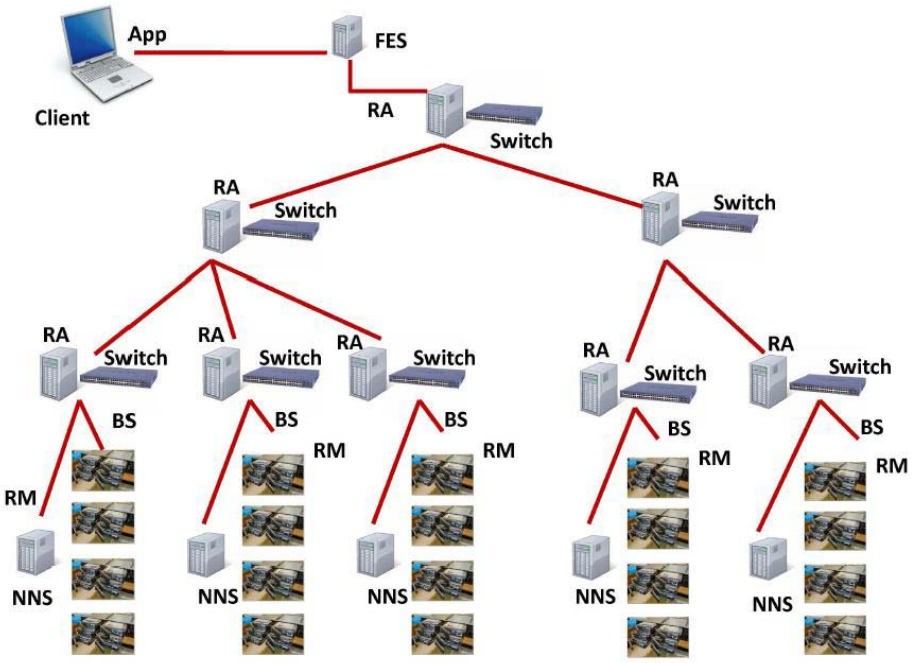

Fig. 3. Overview of SDFS by [4]

5 Comparing File Systems and Internal Algorithms

In this section we compare Ceph and SDSF file systems in their main features and their internal algorithms. While comparing we also try to give conclusions and further improvements for both file systems.

Decoupled data and metadata- Ceph and SDSF both offer separated data management from metadata. Since metadata operations are more complex and represent 30-80% of all file system operations, decoupling data from metadata operations gives the clients of the file system the possibility to a faster access on data manipulation. Therefore this derives to a performance improvement.

Distribution of Metadata Operation- Both file systems offer a distributed cluster of servers for namespace management, providing this way an increase in scalability. SDFS applies the hashing way [14] of spreading metadata workloads through the namenode servers (NNS). To some extend workload is evenly distributed for well behaved hash functions, but still hot spots can be created consisting of individual files located on a single NNS. Besides, the hashing algorithm destroys hierarchical locality and benefits that from it derive. Ceph offers a much better alternative using dynamic subtree partitionig for workload distribution in the MDS Cluster. As above explained this algorithm answers in a more effective way the dynamic needs of such cloud environments and a better chance to scale.

Data storage- Since Ceph is an object-based file system, the key logical unit of the whole file system are objects. CRUSH [6] is the algorithm that distributes data objects in a pseudo-random way through the OSDs. It allows any party to individually calculate the location of data and subtitutes this way, the process of lookup with that of calculation. Also this avoids all the overhead crewated during the lookup of the allocation tables. The abstraction of allocation tables is still present in SDSF where data is stored in the BS as data blocks directed by the NNS. In general NNS use the policies of k-local allocation or even global allocation for the storage of new data blocks. Somehow these algorithms offer a fair distribution but are not able to respond to the contiguous evolution of cloud computing systems.

Client requests- In SDSF client requests are handled at the front end light server that directs them to an appropriate NNS. No matter the fact that this server is stateless and in case of failures bringinig it up is very fast, still it becomes a critical point and compromises performance. The number of requests it can contemporary handle is a finite one. This can be a limiting factor for scalability that in fact we want to increase. On the other side Ceph offers an alternative solution provided by RADOS. RADOS enables every client with a global cluster map to direct requests. It also realises, OSDs to be seen as a single logical store. This way a critical point like the presence of the FES is avoided.

A basical advantage of SDSF over Ceph is taking into consideration full link utilisation for every operation in the file system. The RA and RM algorithms provide that, by periodically monitoring and fairly allocating resources on every request. This provides a higher data rate of exchanging information that brings to an increased performance of the whole file system.

We have to underline the fact that Ceph Distributed File system provides a higher abstraction for the system, but also introduces a more “intelligent” way to store and distribute data than SDFS. Ceph is an object-based file system, while the Scalable Distributed File System still conserves the stored data on blocks, stored in the BS[4]. This results in a higher performance and increased ability to scale.

At the end of this section we can conclude that both couples of algorithms (CRUSH, RADOS in Ceph and RA, RM in SDSF) are implemented in distributed file systems that are highly discussed today to become the file systems serving Hadoop, for scalability and performance increase. Even though there are main differences in the way they behave and the logical data partitions they operate on, a paralel comparision can be realised addressing the issues of scalability and performance.

6 Conclusions and Future Work

This paper reviewed solutions for the issue of scalability and performance increasing, regarding a well-known cloud filesystem like Hadoop Distributed File System. Because of having a single name node server as the source of all metadata operations, all stored in a finite size of RAM, scalability is seriously compromised. Two file systems were analysed and compared in their main fatures and internal algorithms to address HDSF problems: 1) Ceph Distributed File System and 2) Scalable Distributed

File System (SDFS) and their algorithms 1)CRUSH, RADOS for Ceph and 2)RA, RM for SDFS. It is for sure that to provide scalability improvement, the namespace has to be distributed in a cluster of servers where fair policies of metadata workload have to implemented. Ceph seems to answer in a better way these needs with the dynamic subtree partitioning algorithm for handling namespace distribution dynamically. Also CRUSH and RADOS algorithms handle data storage, client requests and file system opearations in a way that fulfills the contiguous evolution of cloud enviroments. Anyway the complexity is high and further developement is needed for providing full compatibily with HDSF platform.

On the other side SDSF preserves a more traditional structure with data stored in blocks and allocation tables used to retrieve them. This brings heavier workloads in the file system and therefore a lower performance. An advantage of SDSF over Ceph is this new protocol realised by the RA and RM algorithms that utilises nearly full link and results in faster operations in the file system. Implementing this new approach in Ceph should be point of further research and devlopement. A weak point in SDSF is the FES server because the amount of contemporary requests it can handle. Alternative solutions have to be provided to handle this scenario.

References

1. Apache, *The Apache HBase Book*, October 2010: http//hbase.apache.org/ book.html.
2. D. Borthakur "The Hadoop Distributed File System: Architecture and Design*"*, *The Apache Software Foundation*, 2007.
3. J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters". *In proceedings of the 6th Symposium on Operating systems Design and Implementation* (OSDI 2004), pp.137-150. USENIX Association 2004.
4. D.Fesehaye, R.Malik, K.Nahrstedt "A Scalable Distributed File System for Cloud Computing",Technical Report, March 2010.
5. M.K. McKusick and S. Quinlan, "GFS: Evolution on Fast-forward," *ACM Queue*, vol. 7, no. 7, ACM, New York, NY. August 2009.
6. C. Maltzahn, E. Molina-estolano, A. Khurana, A.J. Nelson, S. Brandt, S. Weil "Ceph as a scalable alternative to the Hadoop Distributed File System", *;login*: vol. 35, no. 4, pp.38-49, August 2010.
7. M. Olson "HADOOP: Scalable,Flexible Data Storage and Analysis", *IQT QUARTERLY/Connecting Innovation and Intelligence*, vol.1, no. 3, pp.14-18, May 2010.
8. S.Radia, S.Srinivas, "Scaling HDFS Cluster Using Namenode Federation," HDFS-1052, August 2010:https://issues.apache.org/jira/secure/attachment 12453067/high-level-design.pdf.
9. K. V. Shvachko, "Apache Hadoop The Scalability Update*", ;login*: vol.36, no.3, pp.7-13, June 2011.
10. K.V.Shvachko, "HDFS Scalability: The Limits to Growth,", *;login:*, vol.35, no.2, pp.6–16, April 2010.
11. Sage A. Weil, Scott A. Brandt, Ethan L. Miller, Darrell D.E. Long, and Carlos Maltzahn, "Ceph: A Scalable, High-Performance Distributed File System," *Proccedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI)*, Seattle, WA, November 2006, pp.307-320.

12. Sage A. Weil, Scott A. Brandt, Ethan L. Miller, and Carlos Maltzahn. "CRUSH: Controlled, Scalable, Decentralized Placement of Replicated Data," *Proc. of the 2006 ACM/IEEE Conf. on Supercomputing (SC '06)*, Tampa, FL, November 2006.
13. Sage A. Weil, Andrew Leung, Scott A. Brandt, and Carlos Maltzahn. "Rados: A Fast, Scalable, and Reliable Storage Service for Petabyte-Scale Storage Clusters," *Proceedings of the 2007 ACM Petascale Data Storage Workshop (PDSW '07)*, Reno, NV, November 2007.
14. Sage A. Weil, Kristal T. Pollack, Scott A. Brandt, and Ethan L. Miller, "Dynamic Metadata Management for Petabyte-Scale File Systems" *Proc. of the 2004 ACM/IEEE Conference on Supercomputing (SC '04)*, Pittsburgh, PA, November 2004.
15. Howard, J., Kazar, M., Menees, S., Nichols, D., Satyanarayanan,M., Sidebotham, R., and West, "M. Scale and performance in a distributed file system". ACM Transactions on Computer Systems (TOCS) 6,1 (1988), 51–81.
16. Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C.,Eisler, M., and Noveck, D. Network file system (NFS) version 4 protocol. Request for Comments 3530 (2003).
17. Thekkath, C., Mann, T., and Lee, E. "Frangipani: A scalable distributed file system", ACM SIGOPS Operating System Review 31, 5 (1997), 224–237.
18. Braam, P., Callahan, M., and Schwan, P. "The intermezzo file system. In Proceedings of the 3rd of the Perl Confer-ence,O¨ı£¡Reilly Open Source Convention, Citeseer.