Search content via Cloud Storage System

HAYTHAM AL FEEL[1], MOHAMED KHAFAGY[2]

[1] Information system Department, Fayoum University
Egypt

[2] Computer Science Department, Fayoum University
Egypt

Abstract

With cloud computing growing in IT Enterprise. the importance of storing and searching files on the cloud increase . cloud storage is defined as a set of scalable data servers or chunk servers that provide computing and storage services to clients. Our research concern with searching in the file content throw cloud storage system Our research using ontology approach that can be store and retrieve files in the cloud based on its content to resolves the weaknesses that existed in Google File System that depends on metadata and searching only using file name Our new architecture was tested on Cloud Storage Simulator and the result shows that the new architecture has better scalability, fault tolerance and performance for searching for file content in cloud storage system.

Keywords: Ontology-Cloud-Performance-Storage File System-searching file content

1. Introduction

Cloud computing is a paradigm that changes the idea of local computers to a cloud of computers that contains server pool providing different services to many clients at the same time. In cloud computing there are multiple copies from the same application, all copies are updated regularly. Clients can share not only the software but also the hardware without being aware of the sharing methods and techniques. In addition to that cloud computing services can be varying between small to very heavy loads of applications. In addition to that the linearly scalable characteristic which is the breakdown of different workloads into pieces in different chunk servers. So we can conclude that the cloud computing refers to the application delivered via the internet (Saas) via an infrastructure as a service (Iaas) and platform as a service (Paas). This paper will focus on the storage service supplied by the cloud.

There are cloud systems have similar architecture for storage, such as GFS [1] , Elastic Cloud of Amazon[2] and Blue Cloud of IBM[3] which can be concluded in a central entity to index or handle the distributed data storage entities.

All these architectures concern in searching for files in the cloud but ignore searching in the file content. In addition to that the central server may become a bottleneck according to the regular requests to master index which can cause a single point of

failure. According to this point of failure, these architectures developed different techniques of backup and recovery to avoid system failure.

The objective of this study is to identify the weaknesses that existed in Google File System architecture [4]. In addition to that conquer the bottleneck resulting from the central master used for indexing. Reaching a new architecture for cloud storage system that replaces the master index that depends on metadata by an ontology which enhances the searching time via the master server and facilitates the searching inside the file content.

Rest of the paper is organized as follows: Section 2 introduces Google File System architecture; While Section 3 describes the architecture of the Ontology Cloud Storage System (OCSS). Section 4, presents the benchmarks used to test this work and the experimental results. Finally, Section 5 concludes this paper and suggests the future work.

2. Google File System

The first point in the cloud computing term is Google's Eric Schmidt in 2006[5]. As shown in figure 1 Google cloud infrastructure has four systems which are independent of and linked to each other. They are Google File System for distributed file storage, Map Reduce

program model for parallel Google applications [6], Chubby for distributed lock mechanism [7] and Bitable for Google large-scale distributed database [8]. Google File System is considered a distributed file system which enables the host computer to access different files allocated on other computers or servers in the same network [9].Files in Google File System divided into multi a chunk with fixed size. Each chunk is replicated three times at minimum in different chunk servers inside the network. There is a master server contains metadata in an index file contains the location of each chunk and its copies. When an application needs to access a chunk, this application will request the file from the master server. The master server will retrieve the file name associated with the location of that chunk. The application will go to the location of that chunk at the chunk server. The process of updating is slightly different than the reading process. Permissions for any modification to any chunk and agree of that modification is one of the responsibilities of the master server. The application will request a modification to a specific file from the master server. The master server will guide the application to the primary chunk server that have this file and will prevent any changes to the chunks associated to this file in another chunk server. When the modification finished and the master server agree or acknowledge of that changes. Updated versions of the file and its chunks will be replicated in the replica servers. The master server is the coordinator of all processes inside this architecture which can cause a bottleneck to the master server which can cause a single point of failure [10]

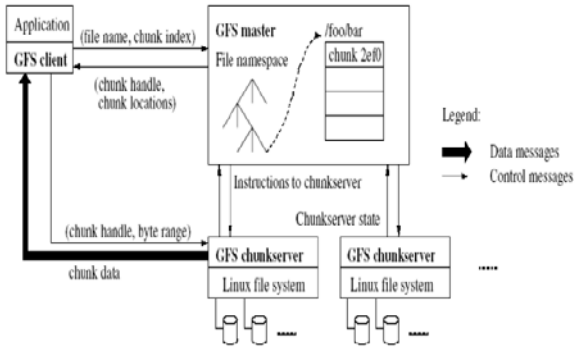

Figure 1 Architecture of Google File System[11]

3. Ontology Cloud Storage System OCSS

3.1 Ontology

Ontology is considered as one of the main components of the Semantic Web, used to represent, acquire and utilize of knowledge [12, 13] to help machines understand the meaning of content of different web resources that increase the opportunities of automated information processing [14]. Ontology provide a well defined vocabulary that define different heterogeneous data resources or files including structured, semi-structure, and unstructured files [15] enabling a new generation of applications especially that merge the idea of the Semantic Web and Cloud Computing.

There are different methods for building or reusing of ontologies such as Cyc method [17], Uschold and King's method [18] and Gruninger and Fox's method [19].

There are different ontology languages, but the reason for choosing OWL as the language for building ontologies related to documents and resources in the cloud computing in our architecture; instead of metadata such as Dublin core which is a vocabulary used to describe online content return to the expressiveness of OWL [20] especially with prosperities. [20] In addition to its ability to provide restrictions on the behavior of properties which are not available in the Resource Description Framework (RDF) and its schema. All mentioned before are not all reasons, but still the main reason for using OWL is the ability of this ontology language to be extended in the future according to the needs of clients as a W3C recommended vocabulary on February 2004 (W3C) and represent different semantic relations [21]

What we need form the ontology at this state are:

- **Identification of the resource :** describes the domain, name and subject of the resource, In addition to the author, and the keywords
- **Recourse Structure:** describes the relation between different components inside the resource.

Administration: This will have the authorization and rights to modify documents. In addition to the document's version number

The methodology used for building ontology in this paper is a result of studying different methodologies such as Cyc Method [17], Uschold and King's Method[18] and Gruninger & Fox's Method [19]. The methodology consists of five main phases. The first phase is called the “Specification Phase” which describes the goals, scope, domain and limitation of the ontology. The second phase is called the “Conceptual Phase” which is responsible for designing and organizing different classes, instances and relations. The third phase is the “Implementation Phase” which uses protégé-OWL 3.4 to implement the ontology that contains about 93 classes including chunk servers, replica, users and files classes, in addition to different properties including data types and objects properties such as the file creator, privileges, place of replica and modification dates etc.. The fourth phase is the “Reasoning

Phase" which is tested using Pellete reasoner to check class hierarchy and inconsistency. While the last phase is the "Evaluation Phase" which evaluates the ontology according to simplicity, compatibility, interoperability, versioning, lifecycle and expressivity. The ontology created in this project is tested and satisfies all the criteria of the Evaluation Phase we find sample of ontology classes in Figure 2

```
prefix xsd:     @
. <#<http://www.w3.org/2001/XMLSchema
prefix xsp:     <http://www.owl-@
. <#ontologies.com/2005/08/07/xsp.owl
. <#prefix swrl:    <http://www.w3.org/2003/11/swrl@
prefix default:  <http://www.owl-@
<#ontologies.com/Ontology1314469453.owl
. <#prefix swrlb:   <http://www.w3.org/2003/11/swrlb@
prefix protege:  @
. <#<http://protege.stanford.edu/plugins/owl/protege
prefix rdfs:    <http://www.w3.org/2000/01/rdf-@
. <#schema
prefix rdf:     <http://www.w3.org/1999/02/22-rdf-@
. <#syntax-ns
. <#prefix owl:     <http://www.w3.org/2002/07/owl@
default:ReplicaServers_2
; a       default:ReplicaServers
default:replicaHasIp
. xsd:string^^"192.162.1.1"
default:ChunkServers_1
; a       default:ChunkServers
default:chunkServerhasIp
; xsd:string^^""
default:isTheChunkServerOfFile
. default:XML_21
default:web1
; a       default:Html
default:creatingDate
; T16:47:00"^^xsd:dateTime02-08-2011"
default:first_Modification_Date
; T16:48:05"^^xsd:dateTime28-08-2011"
```

Figure 2 sample of ontology classes

3.2 Architecture

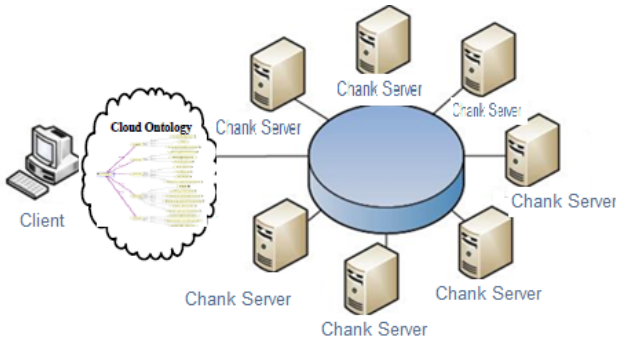

Figure 3 Cloud Storage Based on ontology

Figure 3 shows OCSS architecture that replaces the master metadata index in the cloud by an ontology which facilitates the searching inside the file content and increase the speed of search.

3.3 Read Operation

1. Client sends a request to the Ontology. Containing the logic identifier or the keywords related to the file content.
2. The location of the file which contains the matching keyword will be determined with its Replicas
3. The OCSS will select the chunk server which contains latest version number, if there are more than one candidate, the OCSS will select the nearest node by comparing the IP address of the client and the data server, then return the best address to the client.
4. When the client gets the best address, it will then send its request to the address of the chunk server which contains the data block. Now the chunk server acts as a data provider as the traditional cloud storage platform does.

3.4 Write Operation

1. Client sends a request for a data block with logic identifier and new Keyword to the Ontology.
2. The location of the file which contains the matching keyword will be determined with its Replicas
3. The Ontology updated with new information and new Keywords if there is exists
4. The OCSS will select the chunk server which contains latest version number, if there are more than one candidate, the OCSS will select the nearest node by comparing the IP address of the client and the data server, then return the best address to the client.
5. The OCSS will lock the selected chunk server and its replicas.
6. Then the write process will begin on the chunk server and its replicas.
7. After the updating of each replica, the version number also will be updated and the locked are released.

3.5 Replication

The main problem facing cloud computing is the increasing of the availability of storage system which be solved here in the OCSS by using the agent-scheduling routine replication technique [22].

4. BENCHMARK

A benchmark developed here using C#.NET to simulate and test the GFS and OCSS including the number of chunk servers, clients, operation types (read/write) and number of operations are entered as parameters to the system. Also we used OWL,RDF,XML for building Ontology.

4.1 Hardware Platform

The chunk servers run on an Intel 2.2 GHz Dual core CPU with 4GB RAM, and a Maxtor 160GB 5,400rpm disk drive. A number of 2 GHz Intel machines run the client emulation software. We must have enough client emulation machines to be sure that the clients do not become a bottleneck in any of our experiments. All machines have connected through a switched 10/100Mbps Ethernet LAN and the server connected with 10/1000 Ethernet LAN.

5. EXPERIMENTS RESULTS

The simulator was fed by various numbers of clients in order to test both response time and throughput.

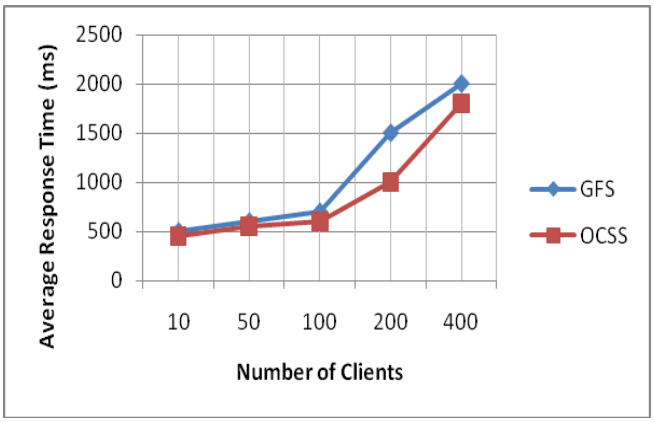

Figure 4:average response time in write operation

Figure 4 demonstrates that with increasing number of clients, GFS shows highest response time due to the bottle-neck resulted from the centralized architecture, whereas OCSS has best response time in write operation because of using ontology instead of metadata.

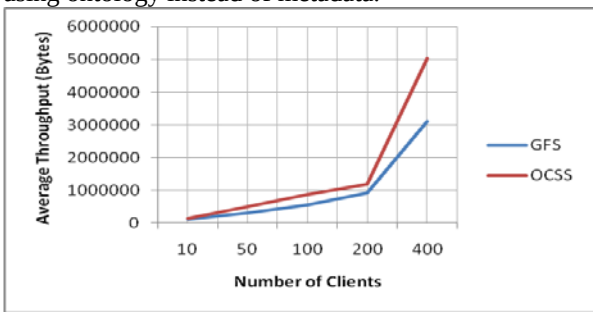

Figure 5: average throughput: during write operation

Figure 5 shows that OCSS has a higher number of write operations due to replication procedure used to maintain consistencies among replicas, compared with GFS.

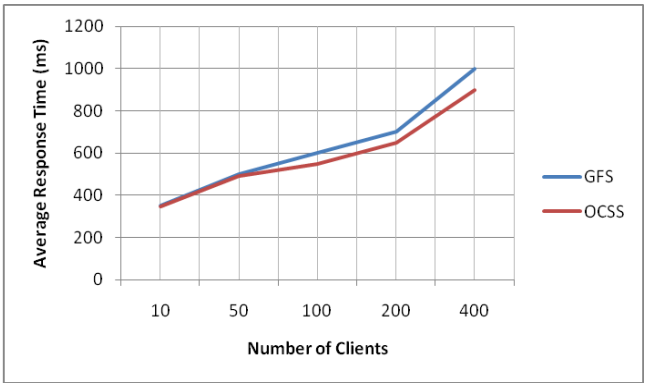

Figure 6 average response time during read operation

Figure 6 demonstrates that with increasing number of clients, OCSS shows lowest response time during read mode compared with GFS

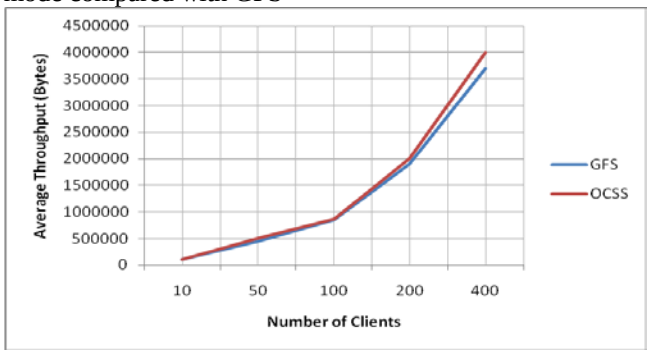

Figure 8: average throughput: during read operation

Figure 8 illustrates that all architectures show approximately similar results for the number of read because OCSS search in the file content.

6. CONCLUSIONS

The experimental results confirm that OCSS shows better results compared to GFS architecture in terms of response time and throughput with write operation according to the using of ontology instead of metadata. On other hand, in response time and throughput in read operation the results are approximately similar. But OCSS has another advantage which is the ability to search in the file content rather than GFS. Our test environment was composed of 5 servers accommodating 50 files distributed randomly and the numbers of clients were entered as a parameter ranging as 10, 50, 100, 200 & 400 where all clients each accessed 10 files applying both read/write operations.

7. REFERENCES

1. Boss G, Malladi P, Quan D, Legregni L, Hall H. Cloud computing. IBM White Paper, 2007.
2. S.G hemawat, H.Gobioff,and S'Leung. The Google file system, In proceedings of the 19th ACM podium on operating systems principles, pages 29-43,2003
3. Amazon Elastic compute cloud (URL) :http: //aws.amazon.com/ec2/, access on Jan 2011
4. IBM Blue cloud project (URL) : http:www.03.ibm.com//press/uslen/phessrelease22613. wss/, access on Jan 2011
5. Ghemawat S, Gobioff H, Leung ST. The Google file system. In: Proc. of the 19th ACM Symp. On Operating Systems Principles. New York: ACM Press, 2003. 29_43.
6. Francesco Maria Aymerich,Gianni and Simon Surcis. An approach to a cloud computing network.
7. Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. In: Proc. of the 6th Symp. On Operating System Design and Implementation. Berkeley: USENIX Association, 2004. 137_150.
8. Burrows M. The chubby lock service for loosely-coupled distributed systems. In: Proc. of the 7th USENIX Symp. On Operating Systems Design and Implementation. Berkeley: USENIX Association, 2006. PP 335-350.
9. Chang F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrows M, Chandra T, Fikes A, Gruber RE. Bigtable: A distributed storage system for structured data. In: Proc. of the 7th USENIX Symp. On Operating Systems Design and Implementation. Berkeley: USENIX Association, 2006. PP 205-218.
10. Fesehaye, Debessay,Malik,Rahul,Nahrstedt and Klara. A Scalable Distributed File System for cloud computing
11. Barroso LA, Dean J, Hölzle U. Web search for a planet: The Google cluster architecture. IEEE Micro, 2003, 23(2):22_28.
12. Li Qin and Vijayalakshmi Atluri, An ontology Guided Approach to change Detection of the Semantic Web Data
13. Amit Sheth. From Semantic Search Integration to Analytics. Dagstuhl on Seminar Interoperability and Integration, September 19-24, 2004. http://www.dagstuhl.de/04391,Materials
14. Lixin Han, Guihai Chen and Lixie. A Method of Acquiring Ontology Information from web Documents.
15. John Davies. Applications of Semantic Technology IEEE Intelligent systems, January / February 2008. www.computer.org/intelligent
16. A.Sheth, Semantic Meta Data for Enterprise Information Integration DM, Review, July 2003
17. D. Lenat and R. Guha, Building Larger Knowledge Based Systems Representation and Inference in the Cyc Project. Boston, Massachusetts: Addison-Wesley, 1990.
18. M. Uschold and M. King, "Towards a methodology for building ontologies," in Workshop on Basic Ontological Issues in Knowledge Sharing in IJCAI Montreal, Canada, 1995.
19. G.-Perez, M. F. Lopez, and O. Corcho, Ontological Engineering. London Springer Verlag Limited, 2004.
20. Yalan Yan, Jinlong Zhang, Miya. ontology Modeling for Contract: Uusing OWL to ExpressS Relations. Proceedings of the l0th IEEE International Enterprise Distributed Object Computing Conference (E Doc'ot IEEE computer society
21. http://www.w3.org/TR/owl-features/
22. E. Sarhan, A. Ghalwash, M.Khafagy, “Agent Based Replication for Scaling Back-End Databases of Dynamic Content Web Sites”, 12th WSEAS International Conference on OMPUTERS, Heraklion, Greece, 2008. PP 867-862