International Journal of Hybrid Information Technology
Vol.8, No.6 (2015), pp.229-236
http://dx.doi.org/10.14257/ijhit.2015.8.6.22

# A Billboard Manager Based Model That Offers Dual Features Supporting Cloud Operating System And Managing Cloud Data Storage

Rajesh Bose, Sandip Roy and Debabrata Sarddar

*Ph.D. Scholar, Department of Computer Science & Engineering, University of Kalyani, West Bengal, India*
*bose.raj0028@gmal.com, SANDIPROY86@gmail.com, dsarddar@rediffmail.com*

***Abstract***

*With an increasing focus on mobility and an innate desire to access data and information efficiently and as quickly as possible, users are turning to technologies that can service their needs with the minimum of fuss and a high degree of reliability. In this paper, we have designed a proposed model that seeks to combine Billboard Manager with an open source cloud operating system known as eyeOS that, together, can help achieve the goals desired in a private cloud environment. The eyeOS is essentially a web-based operating system that allows file read/write/update operations in cloud. The architecture that we have proposed in this paper, involves a mix of the Internet, a private cloud zone, and a Billboard Manager managing an eyeOS powered layer. In this model, any user would be able to create, update, access, and delete almost any file from anywhere where there is Internet connectivity options available. The Billboard Manager would enable users to store or retrieve files across distributed storage nodes in a private cloud zone. The architecture that we have proposed is aimed at providing a thin-client architecture to any user with a PC, laptop, or a suitably-enabled mobile computing device, such that the user is able to collaborate and communicate his/her work within a private cloud zone from almost any location.*

***Keywords:*** *Billboard Manager, eyeOS, Private cloud, Internet, Distributed storage*

## 1. Introduction

Among the trending technologies powered by the Internet today, Cloud Computing has begun opening up a new era. The production and proliferation of ever more cheaper and more powerful processors have enabled to form data centers. A federation of data centers interconnected via Internet is able to support "software-as-a-service" model - a computing architecture which allows businesses and individuals alike, to rationalize operating costs and maximize profits. With ever emergence of newer hardware technologies which are able to sustain reliable network connectivity across vast regions, it becomes possible for consumers to maintain their software and data on remote data centers, thus, minimizing need to procure expensive hardware and software and maintain those in-house [1].

In this work, the primary advantage of the proposed system lies in that the same device can provide a virtual desktop environment while allowing users to place and access data in a cloud environment through secure means of connectivity. The dependency on hardware resources can be significantly reduced and the potential to efficiently utilize energy can be realized. The greatest benefit of this system, however, is that the only significant investment that needs to be made is rather closely focused at the cloud level, and that relating to Internet connectivity. With stable internet connectivity speeds, it would not be difficult for users to enjoy similar degree of response times and data security as they would enjoy directly working on their own computing devices. Regardless of what the

ISSN: 1738-9968 IJHIT

end-users use, with a standard computing device able to field modern web browsers, an user would be able to work on any of his/her file, or even create new ones, without being tied down to any particular location.

## 2. Cloud based Web Desktop and Cloud Storage

### 2.1. Cloud based Operating System

One new concept of operating environment is based on managing the processes and threads of a single or cluster of virtual machines and servers within a computing-specific environment. This is called cloud based operating system. The end user can use the light-end of the cloud OS through a web browser to access preinstalled applications and services made available. End users can use email, calendar, document and photo editor and collaborative tools like chat and social networks through this virtual operating system. Currently there are many cloud OS or web based operating environment available. For example Glide OS, JoliCloud, iSpaces cloud computer, Slive OS, Zero PC, eyeOS etc. Some of the available Cloud OS platforms are open source. We are going to use EyeOS for our purpose.

EyeOS is one of the Open Source platforms designed as cloud OS or web based operating environment. It provided all the benefits of a cloud OS, that is computing via any device, may be a full-fledged desktop or laptop computer or a tablet computer or a mobile phone. One can access applications and utilities, personal files like music and video or documents just by logging into an eyeOS server through any browser. One can upload files from any device and use word processor or pdf viewer or spreadsheet application to get the work done. It is primarily written in PHP, XML, and JavaScript and works as a platform for any applications written using eyeOS toolkit. By providing a full desktop environment through any web browser and many handy utilities it enables the end user to complete the computing task seamlessly even switching device while working is not a problem.

### 2.2. Cloud Storage Definition and it's Architecture

Cloud storage is a system that provides functions such as data storage and business access. It assembles a large number of different types of storage devices through the application software which are based on the functions of the cluster applications, grid techniques, distributed file systems, etc. Cloud storage can be simply understood as the storage in cloud computing, and also can be considered to be a cloud computing system equipped with large capacity storage. Cloud storage system architecture mainly includes storage layer, basic management layer, application interface layer and access layer.

#### 2.2.1. GFS:

**2.2.1.1. System Architecture:** Comprising a single master and a combination of multiple servers and clients as shown in the following Figure 1, a GFS cluster takes shape. Multiple servers are used to host large sections of files in blocks, or "chunks". These servers are mostly standard PC systems running Linux [2] as operating systems.

**2.2.1.2. GFS Maste**r: A GFS master incorporates the directory structure which catalogs the files and the metadata of the files listed in its directory. A single master policy is adopted by GFS such that only one master is used to render services at the same time that it takes to coordinate and synchronize between more than one masters. This translates to savings in costs. A client only has to interact with the master for the metadata, and communicates with the servers holding the "chunks", directly for data other than the metadata.

**2.2.1.3. Chunk Server:** Each fixed size chunk stored on a chunk server is sized to a default of 64M. The GFS divides files into each such sized chunk which are identified individually by uniquely identifiable 64 bit chunk handle. This globally unique identifier is assigned by the master soon after a chunk has been created. A block is replicated on three chunk servers leaving the users the option to choose different levels of replication for separate name space regions of a given file. As would be evident from Figure 1, there are four separate chunk servers with five distinct chunks labeled C0 through C4. Each of which is saved on three chunk servers.

**2.2.1.4. Client:** Every application that runs a GFS client code implements the GFS API. This API enables each application, through the GFS client code, to establish contact with the master and chunk servers to pass and/or retrieve metadata to/from the master. However, it is the group of chunk servers which handle all communication containing actual data.

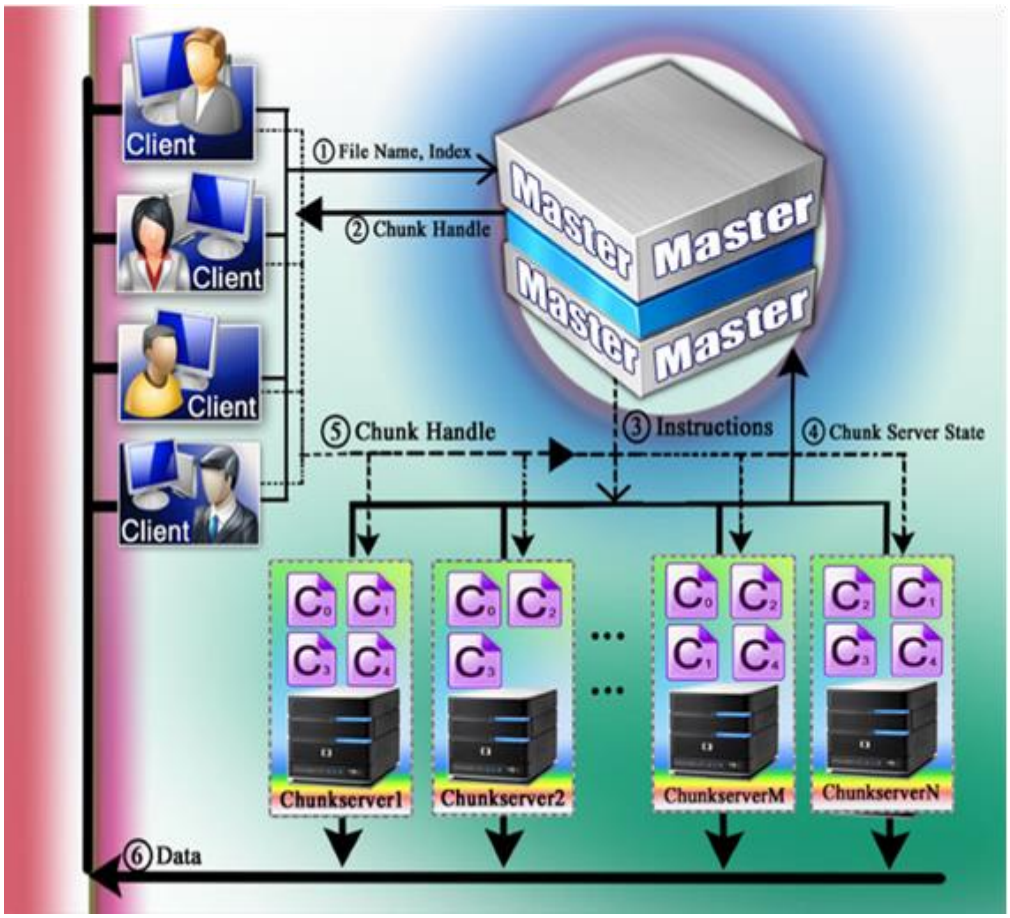



**Figure 1. GFS Architecture**

**2.2.2. Work Flow:** The thin solid lines depict the information control exchanged between the clients and the master, or between the master and the chunk servers, in Figure 1. The thick lines show the data flow between the client and the chunk servers. The dashed lines represent the control information flowing between the clients and chunk servers.

The clients, to begin with, compute a chunk index based on the structure of the files and chunk size. Following which, the file name and the corresponding chunk index is sent to the master. Next, the master sends the chunk handle and locations to clients. Subsequently in the third stage, the clients then begin to transmit chunk

handle and the range of bytes to the nearest chunk server. In the last stage, the chunk server sends the data to the client. Once the client is in receipt of the chunk locations from the master and the interaction with the master ceases. The master does not save the mapping based on the chunk server and the chunk. Instead, the master queries a chunk server at master startup or whenever the chunk server joins the cluster, for its record of chunks stored in the chunk server itself. At regular intervals, the master establishes communication with each chunk server using a heartbeat message, to transmit instructions and gather information on its state.

**2.2.3. HDFS:** Low cost and prone to failure hardware can work as a backbone of large data set distributed across clusters of computers. In this model availability is achieved by creating cluster of local computing and storage of thousands of computers called commodity hardware scaled up to serve large data set. There are open-source software projects which are developing reliable, scalable, distributed computing based on this model. Apache Hadoop is one of the open source software framework which supports this model. The basic of this model is the ability to detect and handle failure of the hardware at the application layer to assure high availability on top of the cluster of the cheap commodity machines.

Apache Hadoop framework consists of four basic modules: Hadoop common, Hadoop Distributed File System (HDFS), Hadoop YARN, and Hadoop Map Reduce. Hadoop Common contains utilities that are required by other modules. HDFS is a distributed file system written in Java. This file system stores data in commodity machines and provides high cumulative bandwidth throughout the cluster. Hadoop YARN (Yet Another Resource Negotiator) manages the computing resource in a cluster so that user's application can use them as required. Hadoop Map Reduce is a programming model that handles large data sets.

HDFS has Master/Slave architecture. The master in a HDFS Cluster is the NameNode which sits on a central server. This server manages the file system namespace and controls access to files by clients. The Data Nodes can reside in multiple computers in the cluster and work as slave of the Name Node. Name Nodes control the Data Nodes and Data Nodes run on the storage attached to it. The Client of the file system can access the name space presented by the Name Node in normal ways like opening, closing and renaming files. Inside Data Nodes, it splits files in blocks and controls the creation deletion and replication of these files blocks which is transparent to the Client. These Blocks are replicated for fault tolerance and this replication is managed by Name Nodes. Data Nodes only follows the instructions from Name Nodes. As HDFS is written in Java, it is OS independent and can run on any Java enabled OS, typically in an open source operating system like Linux.

## 3. Related Work

Recently, much of growing interest has been pursued in the context of Web operating system and cloud storage system. Some typical research projects and cases are presented in the following. A complete outline on various researches and trends in Web operating system and storage has been presented [3]. This paper introduces the concept of Web operating system and cloud storage as well as the architecture of cloud storage. The authors discuss on two fundamental technologies: distributed data store and complex event processing, and workflow description for distributed data processing [4]. The authors focus on, How EyeOS useful in Higher Education System. In this model, students are accessing their own account at anywhere [5]. It looks like a regular desktop operating system, but they can be accessed from anywhere.

## 4. Proposed Work

With a view to providing a light-weight platform to execute applications, we propose an architecture which would revolve around a Billboard Manager. The purpose of the Billboard Manager would be to host an environment which would act as an operating system as well as distribute read write operations across distributed nodes on the cloud. The Billboard Manager itself would be based off of eyeOS. This revolutionary operating system offers a cloud-based operating system environment that is powered by Apache, MySQL and PHP (AMP). It is a web-based operating system and forms the basic structure of our Billboard Manager. With the aid of this eyeOS engine, the Billboard Manager would be able to receive and act on access requests from users and carry our user authentication procedures.

In our proposed architecture, the link to our Billboard Manager's application usability would be published on a SSL device. To begin with, the user hits the SSL link. He then inserts his credentials for accessing the link and to view the remote desktop hosted by our proposed Billboard Manager. In turn, the Billboard Manager would support user actions of writing and reading files. The process of reading, displaying, managing, and then storing back again on the cloud storage system would be handled directly by Billboard Manager and its underlying eyeOS. The files are read when users log into the web-based operating system from a browser. The icon of the file is displayed in the browser supported by the web operating system.

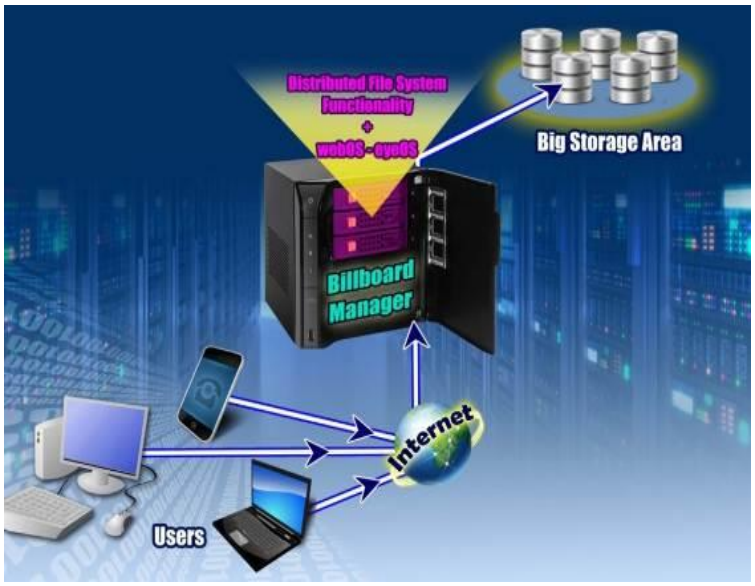
To access this file, the request is channeled to another functional part of Billboard Manager. The Billboard Manager receives the request, then locates the related information from the files, and then calculates the location of the file within the distributed storage nodes that are registered and connected to the Billboard Manager. Blocks of a file are saved separately by the nodes, and these are sent back to the Billboard Manager which then processes and joins the blocks together to form the original file that was stored in the first place. The appropriate file association is identified by the web operating system before it is started automatically for display.

To write the files, users log onto our proposed Billboard Manager using any compatible browser supporting the URL published through the SSL device. The designated web application is used to conduct necessary modifications, and to save files. The Billboard Manager, subsequently, and upon being signaled to save the file, begins uploading the file from the user's terminal. Next, the Billboard Manager identifies the storage blocks to allocate and splits the file into sections. These sections are turned over to the storage nodes registered with the Billboard Manager either singly, or in multiples as decided by it. Depending on the integrity and proper formation of the file, the Billboard Manager would complete the operation and confirm action taken. The response, success or failure, is flashed by the web-based software application for the user to take onward action.

## 5. Algorithm

1) The Billboard Manager stores data related to cloud nodes which are connected to it. The data comprises capacity of each node, IP address and shortest node distance from the Billboard Manager.
2) The Billboard Manager manages the file system name-space and further computes the mapping from files to storage nodes, and allocates storage nodes to save file blocks.
3) All the registered cloud nodes send periodic information to the Billboard Manager. The information consists of channel capacities, and storage space available.
4) The information varies from time to time and is recorded by Billboard Manager to maintain a historical data sheet from where it cans build-up index values.
5) The Billboard Manager begins be analyzing the request type, i.e., upload or download, as initiated by a user.

6) If the request type is found to be of upload, the Billboard Manager splits the file into blocks.
7) Next, the Billboard Manager allocates storage space according the file size blocks and the storage space available on its registered nodes. The split file blocks are uploaded into the appropriate registered cloud nodes by the Billboard Manager.
8) To download a file, the Billboard Manager locates the related information related to the file from its database.
9) It calculates and arrives at the information related to the location of the file within its distributed storage nodes that are connected to it.
10) Blocks of the files that were saved separately and discretely in the nodes, are now pulled in by the Billboard Manager which, subsequently, upon arrival of all the blocks, proceeds to join them to form the file that was saved.




**Figures 2. Proposed System Architecture**

## 6. Conclusion

Our proposed architecture would be able to boost productivity and extend operating efficiency to users with access to the Internet and web-browsing capabilities on their computing devices. But in our paper, we have also expounded on what more could be achieved using Billboard Manager coupled with eyeOS. While the benefits of a web-based operating system are readily apparent, what is important is that the Billboard Manager allows an efficient way of splitting and aggregating user data. By itself, the Billboard Manager acts as a layer that allows seamless integration of nodes in a private cloud environment such that the users are able to work individually or in teams to create, share and collaborate information and data without the need to use higher end-user hardware resources. Further, an user is assured of a secured method of accessing files. User validation and authentication is a key component of our proposed architecture. Our proposed model, offers a combination of security, portability and high-availability of data in a private cloud environment managed by Billboard Manager, with the added light-weight advantage of a web-based operating system in the form of eyeOS.

## References

[1] Q. Wang, C. Wang, K. Ren, W. Lou and J. Li, "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing", IEEE Transactions, vol. 22, **(2011)**, pp. 847-859.

[2] S. Ghemawat, H. Gobioff and S. T. Leung, "The Google File System", Proceedings of the 19-22nd ACM Symposium on Operating Systems Principles, New York: ACM Press, **(2003)**, pp. 29-43.
[3] K. Liu and L.-J. Dong, "Research on Cloud Data Storage Technology and Its Architecture Implementation", Procedia Engineering, vol. 29, **(2012)**, pp. 133- 137.
[4] S. Tsuchiya, Y. Sakamoto, Y. Tsuchimoto and V. Lee, "Big Data Processing in Cloud Environments", FUJITSU Sci. Tech. J., vol. 48, **(2012)**, pp. 159–168.
[5] A. P. Patil and V. S. Ahire, "Use of EyeOS in Higher Education System for Rural Area Students", IJSER, vol. 4, **(2013)**, pp. 1103-1105.

## Authors

**Rajesh Bose** is currently pursuing Ph.D from Kalyani University. He is an IT professional employed as Senior Project Engineer with Simplex Infrastructures Limited, Data Center, Kolkata. He received his degree in M.Tech. in Mobile Communication and Networking from WBUT in 2007. He received his degree in B.E. in Computer Science and Engineering from BPUT in 2004. He has also several global certifications under his belt. These are CCNA, CCNP-BCRAN, and CCA(Citrix Certified Administrator for Citrix Access Gateway 9 Enterprise Edition),CCA(Citrix Certified Administrator for Citrix Xen App 5 for Windows Server 2008).His research interests include cloud computing, wireless communication and networking.

**Sandip Roy** is currently pursuing Ph.D from University of Kalyani. He is an Assistant Professor in the Department of Information Technology, Brainware Group of Institutions, Kolkata, West Bengal, India. He has completed M.Tech in Computer Science & Engineering from HIT under WBUT in 2011. He has also done his B.Tech in Information Technology from WBUT in 2008. His main areas of research interest are Cloud Computing, Data Structure and Algorithm.

**Debabrata Sarddar**, Assistant Professor in the Department of Computer Science and Engineering, University of Kalyani, Kalyani,Nadia, West Bengal, INDIA. He has done PhD at Jadavpur University. He completed his M. Tech in Computer Science & Engineering from DAVV, Indore in 2006, and his B.E in Computer Science & Engineering from NIT, Durgapur in 2001. He has published more than 75 research papers in different journals and conferences. His research interest includes wireless and mobile system and WSN, Cloud computing.