ISSN 1598-0170 (Print)
ISSN 2287-1136 (Online)
http://www.jksii.or.kr

클라우드 기반 프레임워크에서 유비쿼터스 워크스페이스 동기화☆

Ubiquitous Workspace Synchronization in a Cloud-based Framework

프랭크 엘리호데[1] 양 현 호[1] 이 재 완[1*]
Frank I. Elijorde Hyunho Yang Jaewan Lee

요 약

현재 서로 다른 지역에서 여러 대의 컴퓨터 장비를 가지고 파일을 접근하고 작업을 하는 것은 흔히 볼 수 있는 현상이다. 이러한 환경에서 파일의 일치성과 이동성을 이루기 위해서는 워크스페이스 동기화를 위한 효율적인 방법이 사용되어야 한다. 그러나 파일 동기화 만으로는 시간과 장소에 관계없이 작업을 재개하는 작업환경의 이동성을 보장하지는 못한다. 본 논문에서는 클라우드를 기반으로 한 사용자 워크스페이스의 접근 방안을 제공하는 유비쿼터스 동기화 기법을 제안한다. 세션 모니터링과 파일 시스템관리 방법을 결합하여 효율적인 동기화 방법을 제시하였다. 실험결과 고객요구에 대한 평균/최대 지연시간 뿐만 아니라 I/O연산의 수, CPU이용율 관점에서 우리의 연구가 Cloud Master-replica 동기화 기법보다 성능이 우수함을 보였다.

☞ 주제어 : 클라우드 스토리지, 파일 동기화 서비스, 유비쿼터스 워크스페이스

ABSTRACT

It is common among users to have multiple computing devices as well as to access their files or do work at different locations. To achieve file consistency as well as mobility in this scenario, an efficient approach for workspace synchronization should be used. However, file synchronization alone cannot guarantee the mobility of work environment which allows activities to be resumed at any place and time. This paper proposes a ubiquitous synchronization approach which provides cloud-based access to a user's workspace. Efficient synchronization is achieved by combining session monitoring with file system management. Experimental results show that the proposed mechanism outperforms Cloud Master-replica Synchronization in terms of number of I/O operations, CPU utilization, as well as the average and maximum latencies in responding to client requests.

☞ keyword : Cloud Storage, File Synchronization Service, Ubiquitous Workspace

1. Introduction

Cloud storage has gained popularity among casual and corporate users due to its convenience, cost-effectiveness, and low maintenance overhead. Currently, Amazon, Google, IBM, and Microsoft, to name a few, are the major companies pioneering in cloud computing infrastructure. Aimed for the reliable upkeep of data, cloud storage is a platform for online storage where data is stored in virtual servers at remote data centers run by third-party service providers instead of on-site dedicated servers. Today, popular applications utilizing cloud storage are various file hosting services such as Dropbox [1], iCloud [2], Skydrive [3], and Wuala [4]. Users of these applications can store and share files with others over the Internet, as well as maintain contents through file synchronization. Identical copies of the same files are thus held at two or more places, including the service provider's server and the user's devices.

Another familiar method for enabling file and workplace portability is by using desktop virtualization which allows access to an entire information system environment from a remote client device. For example, a device called "secure portable office" stores all the software required which enable tamper proof access to files, data and applications from any online computer and recreates a user's desktop [5].

1 Dept. of Information and Telecommunication Engineering, Kunsan National University, Gunsan Jeollabuk-do, 573-701, Korea
* Corresponding author (jwlee@kunsan.ac.kr)
[Received 29 August 2012, Reviewed 19 September 2012, Accepted 22 October 2012]
☆ This is the extended version of the paper published in 2012 KSII Summer Workshop. This research is partially supported by the Institute of Information and Telecommunication Technology of KNU.
☆ 본 논문은 2012년도 한국인터넷정보학회 하계학술발표대회 우수논문의 확장버전임.

In its most basic form, file synchronization can simply be thought of as file copying. But in reality, synchronization can be more complex since it has to handle several people who need access to multiple files at varying locations. This process of file synchronization is required in a number of scenarios that include synchronization of user files, as well remote back up of massive data sets. A number of works suggest methods for effective content delivery [6] as well as optimization techniques for efficient communication [7]. With considerable amount of data to be managed and processed, and expectedly, diverse working environments, an efficient file and workspace synchronization mechanism is needed.

In this paper, we propose a ubiquitous synchronization approach which provides cloud-based access to a user's workspace. By context, we define "workspace" as the current desktop session of the user comprised of file access (open, update, create, delete) and web activities (URLs). The synchronization of the user's workspace is maintained by utilizing multi-agents for efficient session monitoring and file system management. Also, to eliminate the need for frequent (if not constant) connectivity to the server, we utilize a download-once-upload-once approach which significantly reduces bandwidth requirements. Moreover, the server workload is also reduced by delegating file system monitoring tasks to the client itself; thus minimizing bottlenecks.

2. Related Work

In this section, we discuss previous works and existing technologies that provided the motivation for this research. We divide this section into Cloud Data Storage and File Synchronization Service.

2.1 Cloud Data Storage

By making data available in the cloud, it can be more easily and ubiquitously accessed, often at much lower cost, increasing its value by enabling opportunities for enhanced collaboration, integration, and analysis on a shared common platform [8]. Cloud storage [9] is a model of networked online storage where data is stored in virtualized pools of storage which are generally hosted by third parties. Physically, the resource may span across multiple servers. Cloud storage services may be accessed through a web service application programming interface, or through a web-based user interface.

One of the first milestones for cloud computing was the arrival of Salesforce.com [10] in 1999, which pioneered the concept of delivering enterprise applications via a simple website. FilesAnywhere [11] also helped pioneer cloud based storage services that also enable users to securely share files online. Both of these companies continue to offer those services today. As examples of object storage, cloud storage services like Amazon S3 [12], cloud storage products like EMC Atmos [13], and distributed storage research projects like OceanStore [14] are well-known.

In the academe, cloud data storage is also an emerging field in which a number of studies have been done. A study in [15] describes a solution that allows users to securely store data on a public cloud. In [16], they proposed HadoopRsync, which is for the synchronization from the user's devices to the cloud as well as synchronization the other way around. Another work in [17] focus on the requirement of data-intensive applications. They propose a layered view of the cloud storage architecture, which is composed of user application layer, application hosting platform layer, storage management layer and storage resources layer. In a recent study in [18], they emphasized an approach to storing personal data that would provide device transparency in which the users should see the same view of their data regardless of which device they are working on.

2.2 File Synchronization Service

File Synchronization Service ensures that computer files in two or more locations are updated and synchronized using certain rules. In the past, one well-known protocol is the algorithm used in the widely used Rsync tool for synchronization of file systems across machines [19]. The same algorithm has also been implemented in several other tools and applications. As pointed out in [20], currently there are two main approaches to the distributed synchronization, the user-controlled peer-to-peer synchronization and the cloud master-replica synchronization approach.

In the first approach, users install synchronization software such as [21] in all computers containing the data files and explicitly initiate a peer-to-peer synchronization whenever they need to synchronize their files. This is a "manual" process requiring the user to provide the network addresses of all device(s), as well as the synchronization parameters, such as direction of the synchronization, and which files to keep or overwrite. Managing synchronization for three or more file system hierarchies can be a cumbersome and error-prone process, especially for the average user without technical skills. Cox and Josephson in [22] discuss manual synchronization as a method which is highly error prone, in which users must keep track of files to determine which are up to date and which are not and manually copy most updated files on all machines.

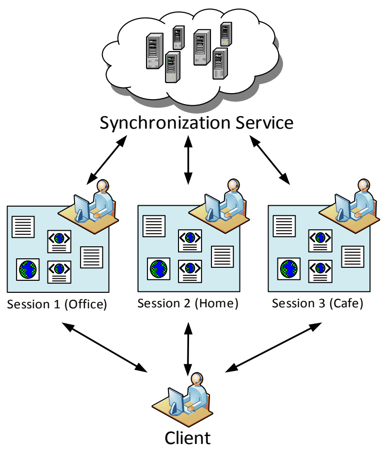
The second approach, cloud master-replica synchronization solves some of the issues by employing cloud services to automate synchronization and deal with multiple devices. In the approach implemented by many cloud-based synchronization and backup services mentioned in Section 1, a master replica containing data to be synchronized is maintained as a master copy in the cloud, to which all the user's file systems synchronize and transfer updates. As our main point of comparison, we further discuss the said approach in the next sub-section.

2.3 Cloud Master-replica Synchronization

The Cloud Master-replica Synchronization is an approach used by commercial service providers as well as the study conducted in [23]. This technique with one-way synchronization to the master replica in the cloud, aims to increase availability and reliability in case of device failure(s), and to automate synchronization management. For the purpose of comparison, we site some disadvantages of the said synchronization approach. First, the server is required to process and monitor massive number of files especially in the instance of multiple concurrent access from clients. This would mean serious workload as well as significant bandwidth requirements on the server side. Second, concurrent update/access by multiple devices from a single user is unrealistic and impractical. It is virtually impossible to have accessed and updated files in multiple devices simultaneously, making the source of file updates not a single replica, but several replicas. In this case, communication may become complex, because it involves transferring files between all devices. This requires constant and high speed network connection, and increases protocol complexity to handle ad hoc communication between devices.

3. Ubiquitous Workspace Synchronization Using the Cloud

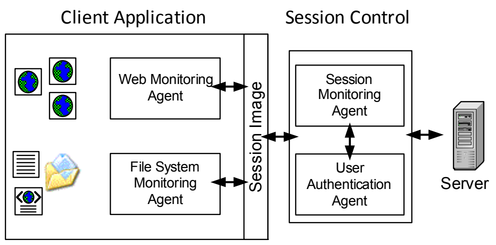
To fully describe our system, let's consider a scenario where a user has to maintain his workspace on multiple machines. On his office desktop, he could be working on multiple files or documents as well as having a hectic web browsing activity. It is likely that he needs to continue his work in another place (home, café, etc.) and this would require saving a copy of the files as well as taking a snapshot of his most recent web activity which will be stored remotely for later access. This is the problem that we would like to address. One of the challenges in maintaining consistent user data is synchronizing replicated data among a number computing devices. This approach is useful not only for backup purposes, but also for remote access, where a user needs to replicate part of a data set, modify it and then synchronize with the main data set when network connection is available. Figure 1 shows the overall view of the system.



(Figure 1) Cloud-based Ubiquitous Workspace Synchronization.

3.1 Session Initialization

In the Session Control layer, the User Authentication Agent (UAA) provides the initial interface between the user and the system. It authenticates the user credential provided by the remote client. If the authentication is failed, the UAA will send a notification to the Session Monitoring Agent (SMA) which causes the connection to be cancelled. Otherwise, SMA will establish a session request to the synchronization service. Upon successful login, a user can access all the sessions he has made which can be downloaded into his current workplace and make necessary modifications locally. The session image is used to securely store user data over the Synchronization Service. However, all operations carried out on this object are handled on the client; this prevents unauthorized access or modifications. Once the session has been downloaded, an instance of a workplace is established on the user's machine. The interaction of the agents within the session control layer is shown in Figure 2.



(Figure 2) The components of the client application.

3.2 Capturing the Session Image

Once the session has been loaded on the client's machine, an image of it will be maintained locally. By default, the entire session will be loaded as a duplicate of the workspace from the previous location. This means that the files the user is working on will be automatically opened with the corresponding application; moreover, web sessions will also be restored and loaded using the machine's default browser.

As shown in Figure 2, once the current workspace has been established, the Web Monitoring Agent (WMA) and the File System Monitoring Agent (FSMA) will come into action. The WMA keeps track of the user's web browsing activity by keeping a record of the websites accessed in the current session. Each time a new URL is loaded, it is immediately reflected to the session image. Through this the user doesn't need to keep bookmarks of the sites visited, it is all left to the WMA. On the other hand, the FSMA performs the more important task of keeping track of the file system. A session image can be thought of as a directory that contains all the files and URLs within the current workspace. Any event that pertains to file access should be captured by the FSMA in order to maintain file consistency within the current workspace.

Within the workspace file system, possible operations include open, update, create, or delete. Invoking these operations causes the FSMA to update the session image; this ensures that the current workspace is synchronized with the user's session. The status of the files within the workspace is determined by the FSMA through the following algorithm:

```
fStat {
//Let Scur and Sprev be the current
      and previous sessions
//Let f be a file
//Let fcur and fprev be the current and previous
      versions of file f

For each file fprev in Sprev and fcur in Scur,
        such that fprev = fcur do
if (fprev.name = fcur.name) and
            (fprev.hash != fcur.hash) then
            //f is Modified
            //replace f with fcur
else if (fprev.name != fcur.name) and
            (fprev.hash = fcur.hash) then
            //f is Renamed
else if fcur is not in Sprev then
            f is new file
else if fprev is not in Scur then
            f is Deleted
else
            //f is Unmodified
end if
End for
}
```

This is then used to upload only the modified parts of a session:

```
UpdateSession(S)
{
//Let S be the active session
For each file f in session S do
            if (fStat(f)=modified) then
            Upload f
elseif (fStat(f)=new) then
            Upload f
            else
            //Skip f
 End for
For each url u in Scur,
if u is not in Sprev then
u is a new url
append u to S
else if u is not in Scur then
u is Deleted
else
u is Unmodified
end if
End for
}
```

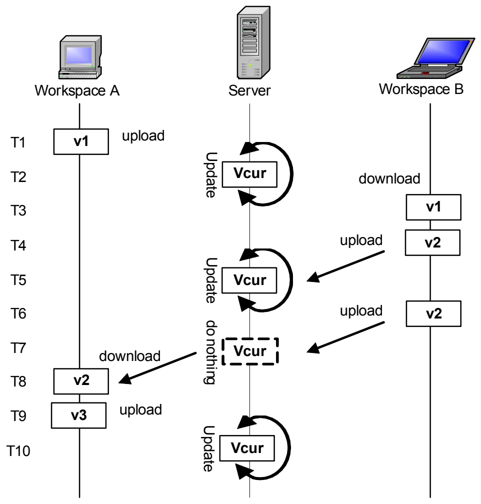
3.3 Update Synchronization

The sample scenario in Figure 3 starts with a file upload of v1 from Workspace A to the server at time T1. The newly uploaded file will become the current version called Vcur after the update at time T2. A session in Workspace B downloads Vcur and modifies v1 into v2 at time T4 followed by a successful update process to the server at time T5. At time T6, Workspace B attempts to upload v2 into the server. It was determined by the server that the current version is v2, therefore the update request will do nothing. At T8, Workspace A downloads file v2 and modified it into v3 at time T9; followed by a successful upload at time T10 upon validation by the server.

3.4 Conflict Resolution

Our approach to dealing with conflicts is to make sure that the files contained in a session are always the latest version. This is complemented by ensuring that a file uploaded by the user is always up-to-date; that is, it was derived from the most recent version. We illustrate our conflict resolution approach in Figure 4.

A sample scenario shown in the figure demonstrates a conflict resolution process. Starting at time *T1*, the first version of a file is created at *Workspace A* then uploaded to

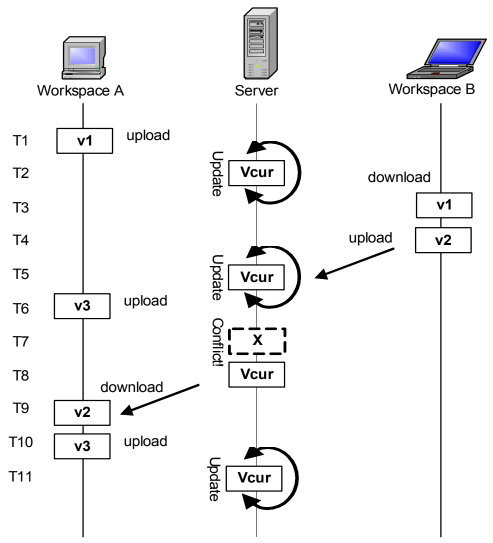


(Figure 3) A sample scenario showing the synchronization of updates: whether to copy a file version to the server or to do nothing.

the server at time *T2* denoted as *Vcur*. At time *T3*, *Workspace B* downloads the current file version and modifies it at time *T4* resulting to file *v2*. Right after the file *v1* was modified to become *v2*, it is uploaded to the server at time *T5* as the current version. Looking back at *Workspace A*, it modified its previous copy of *v1* to a different version (we call *v3*) at time *T6* and attempts to upload to it the server. As can be seen in time *T7*, such action will result to a conflict since the file modification history of *v3* tells the server that it was derived from *v1*which is older as compared to the version it currently holds; therefore the update request will be rejected. At time *T9*, *Workspace A* instead downloads the current version from the server and modifies it to become *v3* at time *T10*. As it attempts to update the current version at the server, the file *v3* will be validated by the server that it was indeed derived from the most current version that it has; therefore the update request is permitted.

3.5 Session Synchronization

On the server side, there is also a session control layer which serves as the counterpart for the client's session control mechanism. It serves as the frontmost component for



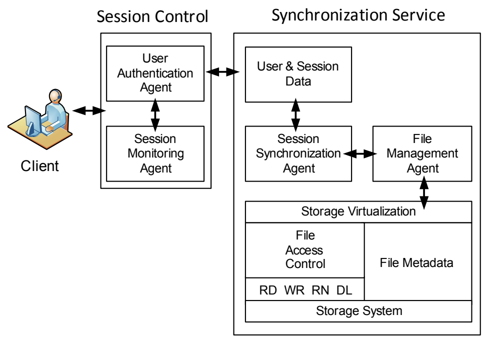
(Figure 4) Conflict resolution based on a file's modification history.

session synchronization. It provides the interface for the synchronization service which is also comprised of various components. User and session data are kept in the database which in turn provides the necessary parameters for the Session Synchronization Agent (SSA) in performing its function. The SSA is responsible for retrieving the status of the current user from the Session Control Layer. If an active session of the current user is found, it does some initialization according to the user profile and provides it with access to the File Management Agent (FMA). The SSA handles the session requests, and coordinates the required files and directories to the FMA. For each request, a copy of the session image is pulled out from the storage and served to the user. The FMA has direct access to the virtual storage of the synchronization service as it is responsible for the creation, update, removal, or swapping of the stored sessions as directed by the user. Figure 5 shows the interaction of the server-side components.

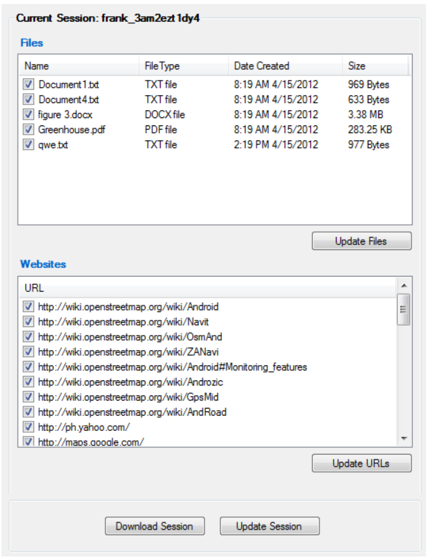
4. Implementation and Evaluation

4.1 Implementation

To realize the proposed solution, it is implemented to



(Figure 5) The components of the synchronization service.


Current Session: frank_3am2ezt1dy4

Files

Name	File Type	Date Created	Size
☑ Document1.txt	TXT file	8:19 AM 4/15/2012	969 Bytes
☑ Document4.txt	TXT file	8:19 AM 4/15/2012	633 Bytes
☑ figure 3.docx	DOCX file	8:19 AM 4/15/2012	3.38 MB
☑ Greenhouse.pdf	PDF file	8:19 AM 4/15/2012	283.25 KB
☑ qwe.txt	TXT file	2:19 PM 4/15/2012	977 Bytes

Update Files

Websites

URL
☑ http://wiki.openstreetmap.org/wiki/Android
☑ http://wiki.openstreetmap.org/wiki/Navit
☑ http://wiki.openstreetmap.org/wiki/OsmAnd
☑ http://wiki.openstreetmap.org/wiki/ZANavi
☑ http://wiki.openstreetmap.org/wiki/Android#Monitoring_features
☑ http://wiki.openstreetmap.org/wiki/Androzic
☑ http://wiki.openstreetmap.org/wiki/GpsMid
☑ http://wiki.openstreetmap.org/wiki/AndRoad
☑ http://ph.yahoo.com/
☑ http://maps.google.com/

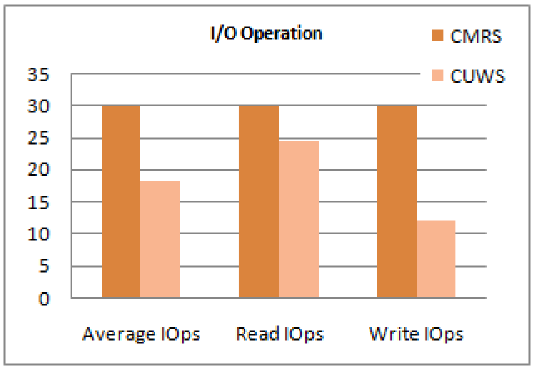
Update URLs

Download Session Update Session

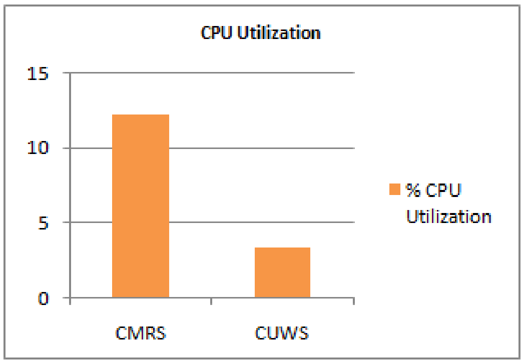
(Figure 6) Files and URLs contained in a session

work as a cloud-based client-server system. A client application is required to connect to the server in order to access and download previous sessions as well as upload new ones. The server application is responsible for authenticating and storing user profiles as well as maintaining the session synchronization.

As shown in Figure 6, selecting a session will cause the contained files and URLs to be displayed. The user can choose to either download the entire session, or just a part of it. After a session has been downloaded, this will form the current workspace of the user. Downloading a session will



(Figure 7) The I/O operations per second.



(Figure 8) CPU utilization.

initiate the files and URLs to be loaded into the workspace and automatically opened by their respective applications.

All open files will be loaded on a temporary directory which will be monitored for any file access-related events. Every time a file is modified, created, renamed, or deleted, this should be reflected in the session image.

While a session is active, the u

ser can update it by appending additional files and URLs. Any changes made to the currently loaded session image will not be reflected to the server unless an Update Session command is issued by the user. This means that all session activities occur locally within user's workspace on the client machine. As soon as the session image is uploaded to the server, the user can now choose to end the session. Once the user decides to end his session, all temporary files and directories created during the session will be deleted thus leaving no footprint of the previous workspace.

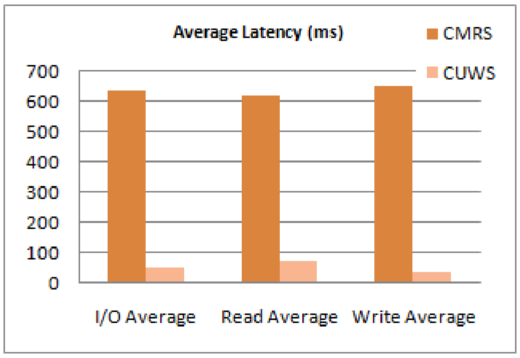
4.2 Evaluation

We evaluate the performance of our work, the Cloud-based Ubiquitous Workspace Synchronization (CUWS), by comparing it against the Cloud Master-replica Synchronization (CMRS). The experiment was conducted for a period of one hour with sessions containing files of random sizes. As for the metrics, their respective average and maximum latency in carrying out session read and write were measured. Aside from that, the I/O workloads and CPU utilization were also evaluated. We used these metrics to further emphasize that delegating all the synchronization processes to the server would significantly degrade its performance. Moreover, through a download-once-upload-once approach, less workload is delegated to the server; thereby significantly enhancing the performance of the system and improving the quality of service.

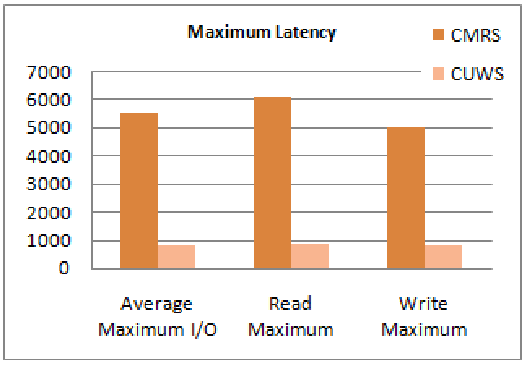
In Figure 7, the I/O operation is shown in terms of Average I/O, Read I/O, and Write I/O per second, respectively. While in Figure 8, the CPU utilization for CMRS is more than thrice as that of CUWS. As shown in the figures, simultaneous file system access considerably contributes to the overall I/O workload of the server as it handles numerous read and writes requests to the stored files. This supports our method of delegating file system management to a local replica of the session within the user's workspace.

In Figure 9, we show the average latencies of the two approaches in terms of Average I/O response time, Read response time, and Write response time. The graph shows the average time between initiation and completion of an I/O operation, averaged over the length of the test, in milliseconds. Also indicated is the average time between initiation and completion of a read and write operation. The server for CMRS greatly suffered as the number of concurrent sessions increased. As shown, there really is a huge difference between the latencies of the two methods. This can be attributed to the amount of workload that has to be handled by the server in processing I/O requests.

In the duration of our test, there exists a peak level for the parameters that we measured. As shown in Figure 10, the



(Figure 9) Average latencies in ms of I/O, Read, and Write response.



(Figure 10) Latencies in ms for Average Maximum I/O, Read Maximum, and Write Maximum.

maximum latency is the point in which the server took the longest time to respond to a client request (Session Upload/Download, I/O, Read, and Write). Similarly, the maximum latencies in terms of Read response time, Write response time, and their average are shown. At this point, the performance difference between the two approaches has greatly increased.

The results above show that delegating all the necessary synchronization processes to the server would significantly degrade its performance. The performance drawback involves I/O cost and latency concerns that arise in the Cloud Master-Replica Synchronization approach. This is the important issue addressed by our work.

5. Conclusions and Future Work

Being able to access your personal data in a ubiquitous manner is just one of the many benefits of cloud storage. However, it is worth pointing out that cloud storage systems are not specifically designed to serve as high-performance file systems but rather scalable and easy to manage storage infrastructures. This is shown in the results of our study in which a cloud-based master replica approach suffered in the event of heavy workload caused by concurrent client connections and excessive I/O requests. Our work outperformed CMRS in terms of optimal number of I/O operations, CPU utilization, as well as the average and maximum latencies in responding to client read and write requests.

Synchronization of the user's workspace was made possible by efficient session monitoring and file system management. By delegating file system monitoring tasks to the client itself, server workload is greatly reduced. As a result, bottlenecks are minimized and the need for almost constant connectivity to the server is discarded. In our future work, we intend to incorporate compression and encryption algorithms to improve the reliability and security of our system.

References

[1] "Dropbox Website", http://www.dropbox.com/
[2] "iCloud Website", http://www.icloud.com/
[3] "Skydrive Website", http://www.skydrive.com/
[4] "Wuala Website", http://www.wuala.com/
[5] "Secure Portable Office", http://www.commsbusiness.co.uk/RSS_News_Articles.cfm?NewsID=5575
[6] J. W. Byers, J. Considine, M. Mitzenmacher, and S. Rost: Informed Content Delivery Across Adaptive Overlay Networks, IEEE/ACM Transactions on Networking, vol.12, no.5 (2004) pp. 767- 780
[7] U. Irmak, S. Mihaylov, and T. Suel: Improved Single-Round Protocols For Remote Rile

Synchronization, Proc. of INFOCOM 24th Annual Joint Conference of the IEEE Computer and Communications Societies (2005) pp. 1665- 1676

[8] J. Wu, L. Ping, X. Ge, Y. Wang, and J. Fu: Cloud Storage as the Infrastructure of Cloud Computing, Intelligent Computing and Cognitive Informatics (ICICCI) International Conference (2010) pp.380-383

[9] "Cloud Storage", http://en.wikipedia.org/wiki/Cloud_storage

[10] "Salesforce Website" http://www.salesforce.com/

[11] FilesAnywhere Website", http://www.filesanywhere.com/

[12] "Amazon S3 Website", http://aws.amazon.com/s3/

[13] "EMC Atmos", http://www.emc.com/storage/atmos/atmos.htm

[14] "OceanStore", http://oceanstore.cs.berkeley.sedu/

[15] J. Zhang, X. Yu, Y. Li, and L. Lin: "HadoopRsync," Cloud and Service Computing (CSC) International Conference (2011) pp.166-173

[16] R. Koletka and A. Hutchison: An Architecture For Secure Searchable Cloud Storage, Information Security South Africa (ISSA) (2011) pp.1-7

[17] Y. Huo, H. Wang, L. Hu, and H. Yang: A Cloud Storage Architecture Model for Data-Intensive Applications, Proc. of International Conference on Computer and Management (CAMAN) (2011) pp.1-4

[18] J. Strauss, J. M. Paluska, C. Lesniewski-Laas, B. Ford, R. Morris, and F. Kaashoek: Eyo: Device-Transparent Personal Storage, Proc. of the USENIX Annual Technical Conference (2011) p.35-35

[19] A. Tridgell and P. Mackerras: The Rsync Algorithm, Technical Report TR-CS-96-05, Australian National University (1996)

[20] S. Uppoor, M.D. Flouris, and A. Bilas: Cloud-based Synchronization of Distributed File System Hierarchies, Proc. of IEEE International Conference on Cluster Computing (2010) pp.1-4

[21] B.C. Pierce and J. Vouillon: What's in Unison? A Formal Specification and Reference Implementation of a File Synchronizer, Tech. rep. MS-CIS-03-36, University of Pennsylvania (2004)

[22] R. Cox and W. Josephson: File Synchronization With Vector Time Pairs, Technical Report MIT-CSAIL-TR-2005-014 (2005)

[23] B. Xianqiang, X. Nong, S. Weisong, L. Fang, M. Huajian and Z. Hang: SyncViews: Toward Consistent User Views in Cloud-Based File Synchronization Services, Chinagrid Sixth Annual Conference (2011) pp.89-96

◐ 저 자 소 개 ◑

프랭크 엘리호데 (Frank I. Elijorde)

2003년 Western Visayas College of Science and Technology, Philippines, BS in Information Technology
2007년 Western Visayas College of Science and Technology, Philippines, MS in Computer Science
2011년~현재 Kunsan National University, South Korea, Graduate Student in Ph. D. Course
관심분야 : Distributed systems, cloud computing, data mining, ubiquitous sensor networks, RFID
E-mail : frank@kunsan.ac.kr

양 현 호 (Hyunho Yang)

1986년 광운대학교 전자공학과 졸업(학사)
1990년 광운대학교 대학원 전자공학과 졸업(석사)
2003년 광주과학기술원 정보통신공학과 졸업(박사)
1989년~1990년 삼성SDS 근무
1991년~1997년 포스데이타(주) 근무
1997년~2005년 순천청암대학 근무
2005년~현재 군산대학교 정보통신공학과 교수
관심분야 : 무선데이터통신, RFID/USN etc.
E-mail : hhyang@kunsan..ac.kr

이 재 완 (Jaewan Lee)

1984년 중앙대학교 이학사-전자계산학
1987년 중앙대학교 이학석사-전자계산학
1992년 중앙대학교 공학박사-전자계산학
1996년 3월~1998년 1월 한국학술진흥재단 전문위원
1992년~현재 군산대학교 교수
관심분야 : 분산 시스템, 운영체제, 유비쿼터스 시스템, 클라우드 컴퓨팅 등
E-mail: jwlee@kunsan.ac.kr