CS-Mobile: A Cloud-based Distributed Storage Middleware for Mobile Devices

Peng Xiao[1*] and Yanping Zhang[2]

[1]*College of Computer and Information Science, Hunan Institute of Engineering*
[2]*College of Computer and Communication, Technical University of Munich*
[*]*xpeng4623@gmail.com; yanping.zhang@wzw.tum.de*

Abstract

With the development of wireless communication technology, mobile devices have played more and more important roles in the people's daily lives. However, data storage and sharing are difficult for these devices due to the data inflation and natural limitations of mobile devices, such as limited storage space and computing capability. In this paper, we present a lightweight Cloud-based storage framework, which provides an easy-to-use file navigation service for attribute-based file querying or semantic-based data retrieving. Meanwhile, it incorporates an effective mechanism for users to verify their data integrity, which can relieve much burden from mobile devices. Experimental evaluations show that the proposed framework is effective to provide flexible and reliable data sharing in mobile computing environments.

Keywords: *Cloud Computing; Mobile Storage, Data Integrity, Quality of Service*

1. Introduction

Cloud computing has emerged as a promising technology that provides large amounts of computing and storage capacity to high-performance applications with increased scalability and high availability, and reduced administration and maintenance costs [1, 2]. With the popular of "Software as a Service" (SaaS) computing architecture, large-scale storage pools have been built and deployed on more and more Cloud-based data centers by using cheaper disk components. Traditionally, these storage systems are designed for scientific research or high-performance computing. Recent years, many people have realized that Cloud-based storage system is a very cost-effective for commercial applications.

One of the most successful examples is the Cloud-based mobile storage. With the development of wireless communication technology, various kinds of mobile devices have been invented for working, studying or entertainment. These mobile devices have played more and more important roles in the people's daily lives. However, data storage and sharing is difficult for these devices due to the data inflation and natural limitations of mobile devices, such as the limited storage space and computing capability. Since the emerging Cloud storage platform can provide reliable and unlimited storage, they satisfy to the requirements of mobile computing environments very well, also it can offer a flexible and low-cost solution to meet the growth storage requirements in such environments.

However, many studies have indicated that data storage service in mobile computing environments is of many distinctive characters from that in conventional high-performance computing environment. For example, in the study of [3], Salmon, *et al.*, show that the mobile device users tend to organize and navigate their files via file

attributes instead of traditional hierarchical directory tree. The most used attributes are publisher-provider metadata, keywords, time-stamps. The inconsistence between the data access and data storage brings the users troubles in translating the semantic views to hierarchical views when operating files. In addition, the other concern about Cloud-based storage is the secure data accessing. Since mobile devices is of limited computation capability, complex security mechanism can only be carried by the remote servers in data centers, which means that "Client/Sever" based security model is more suitable in mobile storage systems.

Motivated by the above reasons, we design and implement a lightweight storage system, namely *Cloud-based Storage achieve for Mobile computing* (CS-Mobile), which is aiming to address two challenges of data storage in mobile computing environments. Firstly, CS-Mobile needs to provide an easy-to-use file navigation service for attribute-based file querying or semantic-based data retrieving; Secondly, it should implement an effective mechanism for users to verify their data integrity, which can not increase too much burden on mobile devices.

The rest of this paper is organized as follows. Section 2 presents the related work. In Section 3, we briefly describe the framework of the proposed system; In Section 4, implementation details are presented. In Section 5, experiments are conducted to investigate the effectiveness of our CS-Mobile. Finally, Section 6 concludes the paper with a brief discussion of future work.

2. Related Work

Nowadays, many big vendors have provided their Cloud-based storage solutions. For example, Amazon's *Simple Storage Service* (S3) is a popular commercial Cloud storage system, which is designed to be general purpose storage utility with simple and open standard operations for building customized applications [4]; Microsoft's *Live Mesh* is targeted at the efficient data synchronization between online storage and local devices [5]; Mozy [6] and Symantec's Protection Network (SPN) [7] focus on the data backups, which are used to provide data protection of critical files against unpredictable disasters. Other commercial distributed storage systems include Cumulus [8], Brackup [9] and Jungle Disk [10]. In general, these systems are aiming to providing reliable and extensible data backup service for users.

Recently, studies on user's behavior show that mobile device users tend to organize and navigate their files via file attributes instead of traditional hierarchical directory tree. The most used attributes are publisher-provider metadata, keywords, time-stamp. Consequently, many systems, i.e. SmartBox [11], EnsemBlue [12], FEW [13], have taken this finding into consideration and provided file navigation function based on semantic query. In addition, they also provide some optimization mechanisms for data synchronization and replication between personal mobile devices. The Semantic File System (SFS) [14] is the first concept file system which provides attribute-oriented data query. Recently, Google Desktop [15], Beagle [16] and WinFS [17] also extend their storage systems with extra semantic information, since it is critical to enlarge the number of potential users.

The above systems are all providing efficient file sharing in mobile environment, however, their security is ether too vulnerable or causing heaving burden on handheld devices. One the other side, many researchers have taken great deal of efforts on Cloud security in high-performance computing environment. For instance, Ateniese, *et al.*, [18] proposed a data possession model for ensuring possession of files on un-trusted storages, and lately they proposed a dynamic version of the prior scheme to supporting

non-blocking operation on dataset [19]. In [20], Wang, *et al.*, consider dynamic data storage in a distributed scenario, and the proposed challenge-response protocol can both determine the data correctness and locate possible errors. The above studies indicate that if an independent third part can be introduced into security mechanism, many computation costs can be removed from end-user to high-performance servers. Based on these observations, we adopt such a third-part security into our CS-Mobile so as to relieve the burden of mobile devices.

3. Framework of CS-Mobile Platform

3.1 Architecture Design

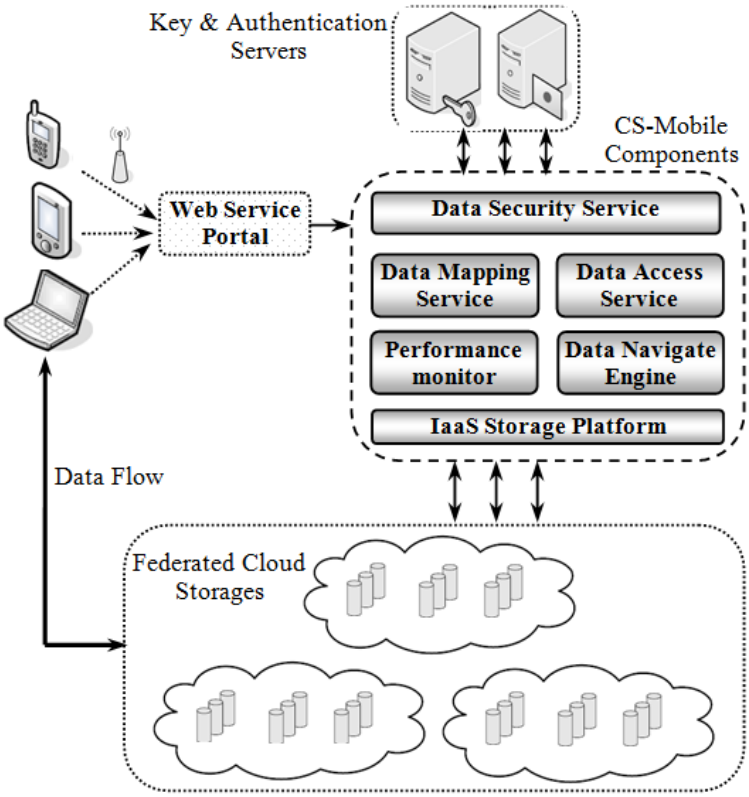


Figure 1. Architecture of Cloud-based Storage System in Mobile Computing

Unlike high-performance storage systems, the goal of CS-Mobile is designed for providing an extensible and lightweight storage system to mobile device users. In the personal mobile devices, the file is often small-size, such as MP3 music, video clips, personal photos and documents. In common, these files are created once and accessed many times after that, often called as write-once-read-many files. Therefore, efficient data retrieving and convenient file navigating are the primary objectives when designing CS-Mobile. On the other side, sharing data security is of significant importance for mobile computing, especially for those virtual communities which relies

up-to-date sharing-files to drive their works or projects [21, 22]. Therefore, we design our CS-Mobile architecture as following.

In the architecture of Cloud-based mobile storage system as shown in Figure 1, there are four different network entities including mobile client, federated cloud storage, independent security server and CS-Mobile. Mobile devices interact with CS-Mobile through a general Web service portal that deprived from our previous project VRA [23]. The key components of CS-Mobile are responsible for data security, file navigation, data mapping and etc, which will be presented in the following sections in detail. As mentioned before, mobile devices have only limited computational capabilities. In order to relieve the burden of mobile devices, the data security work are shifted to independent server which interacting with CS-Mobile component directly. At the bottom level, the storage resource layer contains heterogeneous distributed resources, including commodity computers, enterpriser storage servers and other legacy data clusters. These storages are management by an IaaS platform so as to provide unlimited storage capabilities. Currently, we choose open source version of Google File System [24] as the IaaS-level storage management middleware.

In the above architecture, data flows are transferred between mobile devices and federated cloud storages directly instead of through CS-Mobile. It is because we do not want to put two much communication load onto CS-Mobile, or else its bandwidth bottleneck might become significantly when the number of users increases. This design also makes CS-Mobile can concentrate on its main objectives, which are data security and efficient data navigation.

3.2 Hierarchical Framework of CS-Mobile Components

CS-Mobile is designed as a hierarchical framework as shown in Figure 2. The key components in CS-Mobile are illustrated by shadow rectangles. Generally speaking, there are three distinctive layers in the CS-Mobile framework.

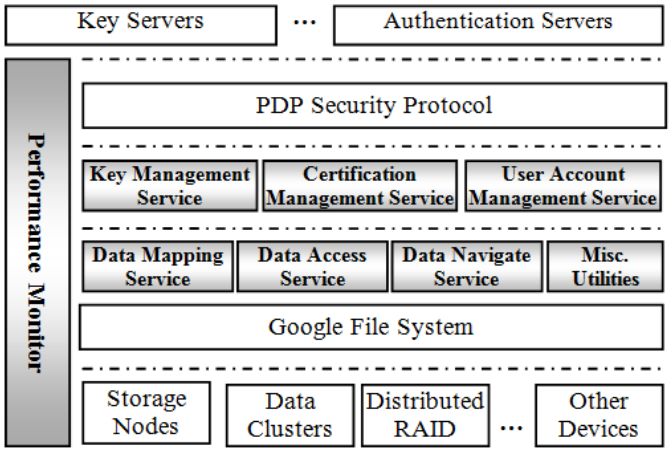


Figure 2. Hierarchical Framework of CS-Mobile

The lowest level is Cloud-based storage infrastructure which is deployed by the open source version of GFS. GFS plays the role of IaaS platform, which federates the distributed storage nodes into Cloud storage systems. Files in the system are divided into fixed-size chunks and are stored on the underlying storage servers as ordinary files. They can be accessed by the users directly in order to alleviate the burden of CS-Mobile

server. The underlying storage devices are from the network centers of five institutes in our campus. At present, they are 150 storages, 2 data clusters and 5 RAIDs. The total storage capability is up to 35 TB and the maximal throughput is about 0.7 TB per day.

The middle level consists of the key components of CS-Mobile, which provide supports for data security accessing, data mapping, file navigation and other utilities. In CS-Mobile framework, a typical data access operation involves the following steps: firstly, mobile users send their data access query to Data Access Service, which translates the user's query into formal attribute-based query sentences; then, Data Navigate Service uses these query sentences to search the GFS, and construct an attribute-namespace view tree for users; If data store or modified operation occurs, Data Mapping Service is triggered to find target storage chunks so as to direct mobile users to finish their operations. Other data access utilities are designed for data buffer, file synchronization or system testing.

The top level is an implementation of PDP security protocol [25], by which CS-Mobile can interact with independent key servers and authentication servers. The details of PDP security protocols will be presented in the next section. In order to evaluation the performance of CS-Mobile, we implement a Performance Monitor component, which can logs all the requests from mobile users and analyzes the workload of other components, also it keeps track of the underlying storage nodes. For instance, when a new storage node or device is added to our system, the Performance Monitor logs this event it with a time-stamp. If a storage node is deleted from the system, it also logs the event. For each storage node or device, Performance Monitor maintains a real-time performance table for it. Currently, we mainly use Performance Monitor to evaluate the performance of the prototype implementation. Lately, we hope it can be used to optimize the performance of our system.

4. System Implementation

4.1 Implementation of Security Data Access

In Cloud storage system, the most concern of users is the verification of data integrity at unreliable and un-trusted storage servers. For example, the storage provider, which experiences Byzantine failures occasionally, may decide to hide the data errors from the clients for their own profits. More seriously, storage providers might deliberately delete rarely accessed data belonging to clients so as to saving storage space. So, users should be provided an efficient way to check their data integrity. As mentioned before, mobile devices are of limited storage space and computational capability, therefore, such an integrity check or verification should be conducted remotely without the local copy of target files.

In the existing studies [25-28], many security solutions and models are proposed for solving the data integrity verification in distributed systems. However, most of them are not suitable for mobile computing environment, since they do not take the limitations of mobile devices into account. Among these studies, we find that PDP security protocol [25] provides a lightweight security model and can be adopted into our system if some modifications are made. Therefore, we implement our data security service based on PDP security protocol. Here, we only briefly represent our implementation and modifications.

In PDP security model, the efficiency of large-size dataset operations will put heavy burden on clients. For example, the file signature is constructed by the file index information. When a file block is deleted or added, the computation overhead for re-

construct the signature becomes unacceptable since the signatures of all the following file blocks should be recomputed with the new indexes. To deal with this problem, we do not use the index information when constructing file signatures. More specifically, considering that a data file D consists of a finite ordered set of blocks $\{m_1, m_2, \ldots, m_n\}$, our signature function is noted as $Sig(m_i)$ instead of $Sig(D, i)$. Furthermore, let M: $G \times G \rightarrow G_T$ represent a bilinear map, where G is Gap Diffie-Hellman (GDH) group and G_T is another multiplicative cyclic group [29], and let H:$\{0,1\}^* \rightarrow G$ represent a random hash function. Then, the procedure of our security protocol is shown in following steps:

(1) The user send a random generated $\{i, j\}$ to CS-Mobile with a target filename, where i is the file block index and j is the initial hash value in (0, 1].

(2) Both the public and the private key are generated by *Key Manager Service*, and the target file's signature is produced by $Sig(m_i)$. Then, proof message *ProofMsg*(*key*, *Sig*)is produced according the PDP model and sent to authentication server.

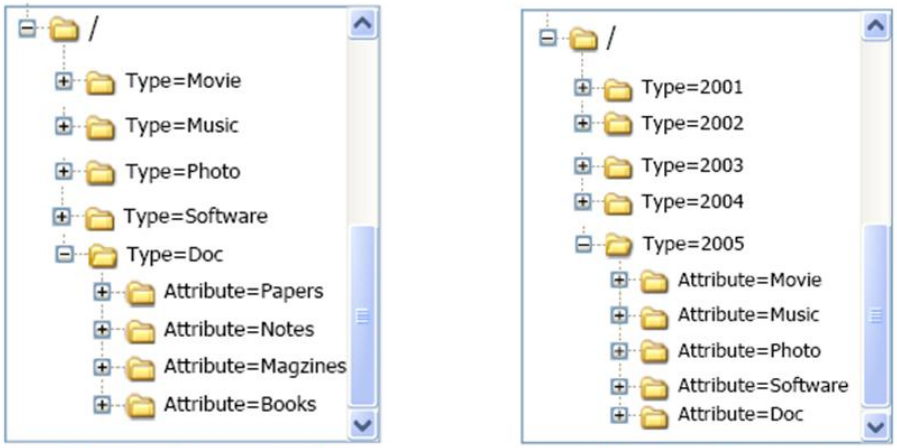
(3) Authentication server is responsible to verify the proof message by using *VerifyProof*(*pfm*, *key*, *H*(*j*)). If verifying operation fails, it return false to users; otherwise, an *Integrity Assurance* procedure will be invoked by CS-Mobile.

4.2. Implementation of Attribute-based Data Navigation

As mentioned before, attribute-based data navigation service is designed for satisfying the requirements of mobile communities. However, there are still many users used to the conventional hierarchical directory tree. So, we provide both conventional hierarchical and attribute based navigation service to meet different needs of various users. The metadata maintained in the metadata server including the mappings from files to chunks, the mappings from chunks to chunk servers and the attribute based namespaces indicating the user-friendly structure of a set of files.

To be different from hierarchical directory tree, which provides static tree view of files for the users, attribute based namespaces are dynamically generated based on the semantic queries of the users. Semantic queries can be described in terms of (attribute, value) pairs. In order to support such queries, additional mappings from attributes to files are needed. Based on the attribute mappings, files are associated to collections of attributes that can be used for dynamic generation of namespaces. For example, if a file is associated to the attributes <type=music, author=*>, the users can get the namespace containing the file by submitting a query to the metadata server, and get all the music that categorized into different directories according to the name of authors.

In order to make attribute based namespaces compatible with hierarchical directory tree, we build hierarchical namespaces based on attributes mappings automatically. In this way, attribute-based namespace can be dynamically constructed according to the query sentences from users by using the mapping tables. For example, a user's query like <type=movie,attri=action,attri=U.S,attri=2011> will be translated into a set of paths like "/movie/U.S/action/2011", "/movie/action/2011/U.S", "/movie/2011/U.S/ action". So, a predefined mapping fashion should be decided. In CS-Mobile implementation, we provide multiple predefined schemes according to the common behavior of users. At the same time, users are enabled to define their own style when navigating their files. Two examples of predefined attribute namespace mappings are shown in Figure 3.



(a) types={data_content} (b) types={data_time}

Figure 3. Examples of Attribute-Namespace Mappings

So, the file types are defined as the top-key when users navigate their files in CS-Mobile. Users are allowed to customize their file types at anytime and any where. The attributes of all files are used to construct the left part of the attributed-based namespace. In this way, users can easily define their favors, and the underlying GFS can dynamically mapping user's files to physical storage regardless of their hierarchical locations.

5. Experiment and Performance Evaluation

In this section, we present the performance evaluation of CS-Mobile when it is in real world experiments. In addition, we also conduct simulative experiments to investigate the performance of CS-Mobile when it is in presence of heavy workloads, burst workloads and traffic congests. So, the experimental results are categorized into two classes: real world performance, simulative performance.

5.1. Real-world Performance Evaluation

As shown in Figure 2, the Performance Monitor component is designed to evaluate the other components' performance. As the system is in the phrase of prototype, we incorporate many functions in this component so as to conveniently test the system. During the experiments, there are 531 registered users and about 12% of them use CS-Mobile more than 3 times per day. So we define them as active users. Firstly, we mainly care about the performance on storage capacity and efficiency. The performance results are collected from 2011-8-1 to 2011-11-30 as shown in Table 1.

Table 1. CS-Mobile Real World Performance on Data Storage

Performance Metrics	Max Value	Min Value	Mean Value
Data Upload Speed (KB/sec)	1553	791	963
Data Download Speed (KB/sec)	2273	1159	1337
Data Throughput (GB/day)	11	0.3	3.85
Response Time of Navigation (sec)	5.9	0.5	1.89
Time of Data Integrity Check (sec/GB)	0.6	0.07	0.274
Total Storage Capability (TB)	33.31	17.42	22.37
Online Active Users (/min)	71	0	5.63
Data Size of Users (GB)	29	0	1.72

The performance evaluation in Table 1 indicates some valuable results for our system. At first, we notice that data download speed is faster than upload speed by 35%. The reason is that the read speed in mobile device is faster than the write speed. As to the metric of data throughput, we notice that it is far from the designing limitation of CS-Mobile. It can be explained by the small number of active users. For example, the online active users are only 5.63/min, which is based on the statistics between 8:00 am to 12:00 pm. If we take the night into account, this value will be smaller. In a summary, the above performance results indicate that our prototype system is capable of deal with more users. In order to investigate its performance under the potential large-size of user groups, we have to conduct simulative experiments, in which we can create virtual users as many as possible, as well as the storage requirements from these virtual users.

5.2. Simulative Experiments and Evaluation

In the simulative experiment, we used 55 PCs to create virtual mobile users. These PCs are distributed in different institutes, even in different cities. All of them are running a simulation agent, which can create virtual users and send requests to the CS-Mobile server. At first, we test the data upload/download speed with different number of active users, and the results are shown in Figure 4.

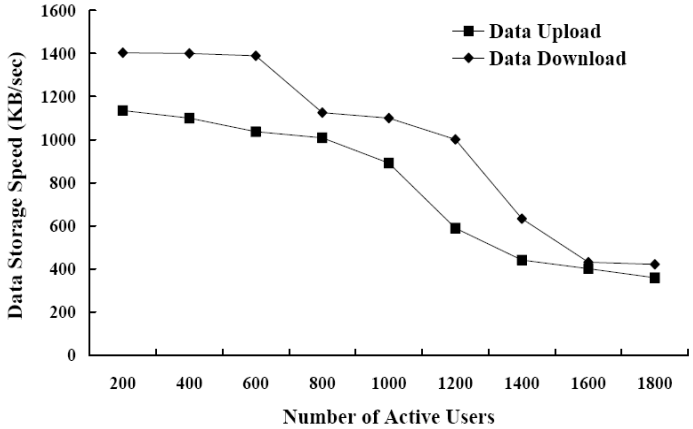


Figure 4. Data Storage Speed with Different Number of Active Users

In simulation, we set the number of active users increased from 200 to 1800. From the results in Figure 4, we can see that both upload and download speed are decreased when the number of active users increases. However, such performance decreasing is not significant when the active users are less than 800. As soon as the number is over 1000, both upload speed and download speed decrease quickly. Until the active users are over 1800, the data storage speed decreased about 75% relative to the maximal speed. It is noteworthy that such an upload/download seems to be acceptable for mobile users. So, we are confirmed that the CS-Mobile can accommodate at lest 1800 active user concurrently. By our real world experience, the percentage of active users is often less 10%. So, CS-Mobile might be able to provide service for virtual mobile communities with 20000 registered users.

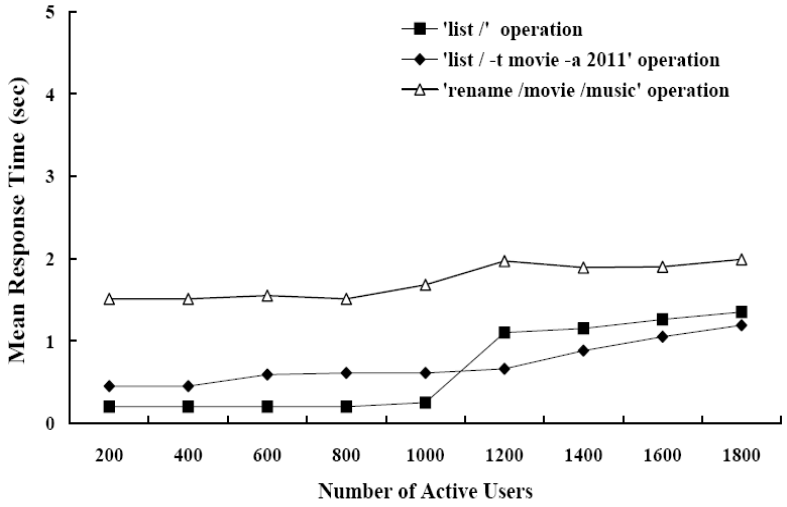


Figure 5. Response Time of File Navigation with Different Number of Active Users

We also examine the mean response time when users conduct navigation operations, and the results are shown in Figure 5. We mainly test three navigation operation: (1) "list /" is to query the list of user's root directory; (2) "list / -t move –a 2011" is to navigate all the movies that produced in 2011; (3) "rename /movie /music" is to rename a directory. The results show some interesting findings. For example, we notice that rename operation requires more time than other two navigation operations. However, its performance is not affected by the number of active users. It is because that rename operation will lead to the namespace re-mapping, which spends the extra response time. As the list operation, we notice that the response time of "list /" operation increases significantly when the number of active users is over 1000. By carefully examine the logs, we find that the frequency of "list /" operation is more than that of "list / -t move –a 2011" operation about 5 times. So, its performance is more likely to be affected by the number of active users.

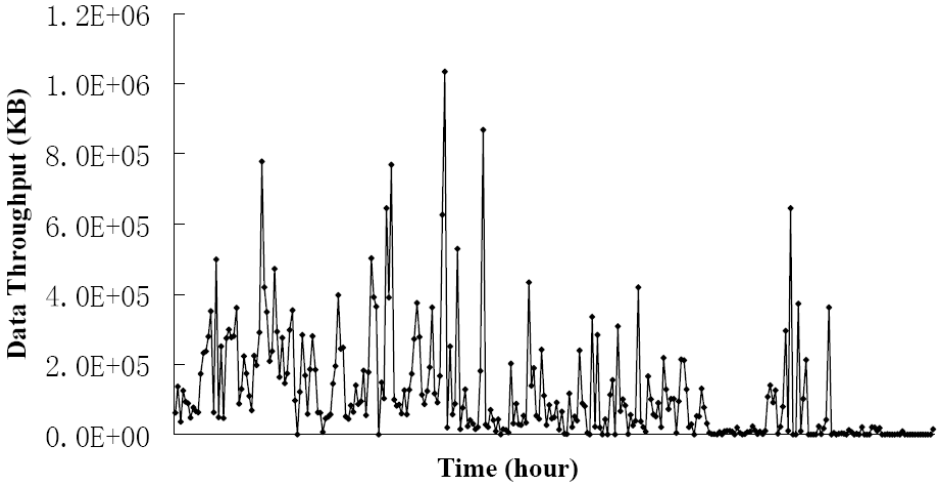


Figure 6. Real-time Throughput in Simulation with 1800 Active Users

Finally, in order to test the performance of throughput metric when system is in presence of heavy workload, we log the real-time throughput of CS-Mobile when active users are 1800, and the results are shown in Figure 6. The sampling period is an hour. As shown in Figure 6, the data throughput fluctuates dramatically. The results are decided by the behavior of users. If too many user concurrently sent data upload/down quests, the system throughput will increases in bursts. According to our designing objective, we hope that the top throughput of CS-Mobile is over 0.7 TB per day. This experimental results show that the total data throughput is about 22 GB, which is only 30% of our designing limitation.

6. Conclusion

In this paper, we present a Cloud-based storage framework for providing an easy-to-use file navigation service and lightweight data security mechanism in mobile computing environments. The implantations of the proposed framework are presented in details. Both real world performance and simulative experimental results are evaluated and analyzed. The experimental evaluations show that the proposed framework is effective to provide flexible and reliable data sharing in mobile computing environments. Based on the experimental results, we are confirmed that when the number of registered users is over 20000, our system can performance well. Currently, our system is just in prototype phase, we plan to conduct more evaluations so as to improve its performance. Also, we are going to incorporate a storage pricing service to supporting the utility-based resource providers.

Acknowledgements

This work is supported by the Provincial Science & Technology plan project of Hunan (No.2012GK3075).

References

[1] L. Youseff, M. Butrico and D. DaSilva, "Towards a Unified Ontology of Cloud Computing", Proceedings of Grid Computing Environments Workshop (GCE '08), **(2008)**, pp. 120-128.
[2] R. Prodan and S. Ostermann, "A Survey and Taxonomy of Infrastructure as a Service and Web Hosting Cloud Providers", Proceedings of International Conference on Grid Computing, **(2009)**, pp. 1-10.
[3] B. Salmon, S. W. Schlosser, L. F. Cranor and G.R. Ganger, "Perspective: Semantic data Management for the Home", Proceedings of USENIX Conference on File and Storage Technologies, **(2009)**, pp. 1-9.
[4] Amazon Simple Storage Service (S3), http://www.amazon.com/s3/.
[5] Windows Live Mesh, http://www.mesh.com/.
[6] Mozy homepage, http://mozy.com.
[7] Symantec's Protection Network, http://www.spn.com/.
[8] V. Michael, S. Stefan and M.V. Geoffrey, "Cumulus: Filesystem Backup to the Cloud", Proceedings of USENIX Conference on File and Storage Technologies, **(2009)**, pp. 1-8.
[9] B. Fitzpatrick, "Brackup", http://code.google.com/p/brackup/.
[10] Jungle disk, http://www.jungledisk.com/.
[11] W. M. Zheng, P. Z. Xu, X. M. Huang and N. Wu, "Design a Cloud Storage Platform for Pervasive Computing Environments", Cluster Computing, vol. 2, no. 13, **(2010)**, pp. 141-151.
[12] P. Daniel and F. Jason, "EnsemBlue: Integrating Distributed Storage and Consumer Electronics", Proceedings of Symposium on Operating Systems Design and Implementation, **(2006)**, pp. 221-230.
[13] N. Preguia, C. Baquero and J. L. Martins, "Few: File Management for Portable Devices", Proceedings of International Workshop on Software Support for Portable Storage, **(2005)**, pp. 110-118.
[14] K. G. David, "Semantic File Systems", Proceedings of ACM Symposium on Operating System Principles, **(1991)**, pp. 1-23.
[15] Google desktop web page, http://desktop.google.com.
[16] Beagle web page, http://beagle-project.org.
[17] M. Dahlia and T. Doug, "Concise version Vectors in WinFS", Proceedings of International Symposium on Distributed Computing, **(2005)**, pp. 87-96.
[18] G. Ateniese, "Provable Data Possession at Untrusted Stores", Proceedings of ACM Conference on Computer and Communication and Security, **(2007)**, pp. 598-609.
[19] G. Ateniese, R. D. Pietro, L. V. Mancini and G. Tsudik, "Scalable and Efficient Provable Data Possession", Proceedings of International Conference on Security and Privacy in Communication Networks, **(2008)**, pp. 1-10.
[20] C. Wang, Q. Wang, K. Ren and W. Lou, "Ensuring Data Storage Security in Cloud Computing", Proceedings of International Workshop on Quality of Service, **(2009)**, pp. 255-263.
[21] T. Hara and S. K. Madria, "Consistency Management Strategies for Data Replication in Mobile Ad Hoc Networks", IEEE Transactions on Mobile Computing, vol. 7, no. 8, **(2009)**, pp. 950-967.
[22] A. Beloglazov and R. Buyya, "Energy Efficient Allocation of Virtual Machines in Cloud Data Centers", Proceedings of IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, **(2010)**, pp. 577-578.
[23] P. Xiao and Z. G. Hu, "A Novel QoS-based Co-allocation Model in Computational Grid", Proceedings of IEEE Global Communication Conference, **(2008)**, pp. 1562-1566.
[24] G. Sanjay, G. Howard and S. Leung, "The Google File System", Proceedings of ACM Symposium on Operating Systems Principles, **(2003)**, pp. 29-43.
[25] G. Ateniese, R. Burns and R. Curtmola, "Provable Data Possession at Untrusted Stores", Proceedings of ACM Conf. Computer and Communication and Security, **(2007)**, pp. 598-609.
[26] H. Shacham and B. Waters, "Compact Proofs of Retrievability", Proceedings of International Conference on Theory and Application of Cryptology and Information Security: Advances in Cryptology, **(2008)**, pp. 90-107.
[27] E. C. Chang and J. Xu, "Remote Integrity Check with Dishonest Storage Server", Proceedings of European Symposium on Research in Computer Security, **(2008)**, pp. 223-237.
[28] T. Schwarz and E. L. Miller, "Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage", Proceedings of IEEE International Conference Distributed Computing Systems, **(2006)**, pp. 1-12.
[29] D. Boneh, B. Lynn and H. Shacham, "Short Signatures from the Weil Pairing", Proceedings of International Conference on Theory and Application of Cryptology and Information Security: Advances in Cryptology, **(2001)**, pp. 514-532.

Authors

Peng Xiao received the Ph.D degree in computer science from the Central South University in 2009. Currently, he is an associate professor in the Hunan Institute of Engineering. Also, he is the advanced network engineer in HP High-performance Network Centre in Hunan. His research interests include grid computing, distributed resource management. He is a member of ACM, IEEE, and IEEE Computer Society.

Yanping Zhang received her M.S. degree from Central South University in 2009. She is now a Ph.D candidate in the Technical University of Munich in Germany. Currently, her research interests include cloud-based large-scale application, virtual machine power model, workflow scheduling model and algorithm. She is a student member of ACM and IEEE Computer Society.