International Journal of Future Generation Communication and Networking
Vol. 9, No. 6 (2016), pp. 87-98
http://dx.doi.org/10.14257/ijfgcn.2016.9.6.09

An Energy-Efficiency Enhanced Mobile Storage Platform in Cloud Environments

Ning Han

HP High-Performance Networking Computing Center, Hunan Institute of Engineering
**linghan81@126.com*

Abstract

In mobile computing environments, storage service is becoming more and more important with the emerging of plenty of data-intensive applications. Although, many mobile storage systems have developed to deal with this situation, few of them take the energy-efficiency into consideration. In this paper, we presented an energy-efficient mobile storage platform, which is based on our previously developed Dependable Cloud-based Storage Platform for Mobile Computing (DCSP-MC). The key improvement of this mobile storage platform is that it takes the energy consumption of storage nodes into consideration. Extensive experiments are conducted to evaluate the performance of the proposed platform, and the results indicate that it can significantly improve the energy-efficiency of the underlying storage nodes, as well as providing desirable quality of service for mobile end-users.

Keywords*: Cloud Computing; Storage; Mobile Computing; Energy Efficiency*

1. Introduction

With the increasing performance of mobile devices, more and more data-intensive mobile applications are developed for end-users, which often involve massive data accessing, query and storage [1-2]. Although disk capacity is increasing quickly, it still can not meet the mobile user's requirements in nowadays. As a result, independent storage platform have become a promising alternative in mobile computing environments [3-5], and many commercial systems have been deployed to provide massive storage service for distributed mobile users. Unfortunately, most of those storage systems are designed for specific devices and specific data types, and their functions are generally limited to data synchronization and backup between mobile devices and the remote storage warehouse [4, 6-7]. So, they are often used as simple back-end disk with aiming to extend mobile devices' storage capability.

In the part few years, many studies have indicated that data storage service in mobile computing is of many distinctive characters from high-performance computing, which can be summarized as following: (1) Mobile users tend to organize and navigate their data via file attributes instead of traditional hierarchical directory tree [2,4]; (2) Mobile users pay more attentions on their data security, while few of them take real actions to protect their data in the local disk [8-9]; (3) Energy limitation of mobile devices has significantly effects on user's data accessing and sharing pattern [1,2,10]. In recent years, 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 [11-13].

Consequently, many researchers have realized that Cloud-based storage system is a very cost-effective for commercial applications [5, 12]. Since Cloud-based storage

ISSN: 2233-7857 IJFGCN

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. Based on the abovementioned observations, in this work we design and implement a lightweight mobile storage system, in which several innovative mechanisms are incorporated for providing better file query experience, enhanced data security, and device energy saving. This work is based on our DCSP-MC platform [5], which is designed to provide an easy-to-use file navigation service in mobile computing environments. In this work, we mainly concentrate on incorporating better data security and energy-efficiency mechanisms into our DCSP-MC platform.

The rest of this paper is organized as follows. Section 2 presents the related work. In section 3, we describe the framework and implementation details of the proposed system; in section 4, experiments are conducted to investigate the effectiveness of the proposed mechanism. Finally, Section 5 concludes the paper with a brief discussion of the future work.

2. Related Work

In the past few years, commercial cloud-based platforms have been developed with aiming to provide reliable storage services and can be friendly accessed through WAN/LAN. For example, the most famous and successful cloud-based storage system is the Amazon's Simple Storage Service [14], which applies elastic mechanism to provide unlimited storage capability for end-users so as to build various kinds of high-level applications. Another wildly-applied cloud-oriented storage platform is Microsoft's Live Mesh [15], which is designed for large-scale data synchronization between separated storage devices. Some other systems include Mozy [16], Symantec's Protection Network [17], Cumulus [18], Brackup [19] and *etc*. The above mentioned systems are designed for Internet-based computing paradigm and none of them are aiming for mobile computing environments.

In mobile computing environments, end-users often exhibit many distinguish behaviour patterns and characteristics when using filesystem. For example, they tend to organize and navigate their files via file attributes instead of traditional hierarchical directory tree and the most used attributes include content metadata, keywords, time-stamp and categorization label. To deal with such a behaviour pattern, *SmartBox* [20] and *EnsemBlue* [21] have taken this into consideration and provided file navigation function based on semantic query and several optimization mechanisms for data synchronization and replication between personal mobile devices. As the semantic query is of significant importance for quick content navigation and attracting potential users, Google Desktop [22] and Beagle [23] also extend their storage systems with extra semantic information.

On the other side, when designing mobile storage platforms an important consideration is how to enhance the data security and integration under constraint to power consumption. Many existing mobile storage systems have provided effective data sharing in mobile environment, however, their security mechanism often lead to heaving burden on handheld devices. More important, few of them have taken the limited energy of mobile devices into consideration. To deal with the data security and integration issue, Wang *et al*. proposed a flexible storage integrity auditing mechanism, which utilizes the homomorphism token and distributed erasure-coded data so as to allow users to audit the cloud storage with very lightweight communication and computation cost [24]. In [25], Mahajan *et al*. described the design and evaluation cloud storage system with aiming to minimize the unsuitable trust assumptions, and implemented a trust-enhanced cloud middleware called Depot

which provides safe guarantees to clients by using a two-layer architecture. Unfortunately, both of them have ignored the well-known fact that complicated security functionality will result in more power consumption as well as extra performance costs.

3. Framework of Storage Service in Mobile-Cloud

3.1. Architecture of DCSP-MC

In [5], we present the implementation of a lightweight mobile storage middleware, namely DCSP-MC, which is fully adoptable in various cloud platforms. In Figure 1, the overview hierarchy of DCSP-MC framework is illustrated. By applying the DCSP-MC in cloud environments, the selection of underlying filesystem is very flexible. For example, in the current implementation of our DCSP-MC, we use the open source version of GFS [26], 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 DCSP-MC server. In our DCSP-MC, it provides supports for data security accessing, data mapping, file navigation and other utilities. Generally speaking, the data accessing in DCSP-MC requires the following operations: Firstly, users send their data access query to Data Access Service, which translates the user's query into formal attribute-based query sentences; Secondly, 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.

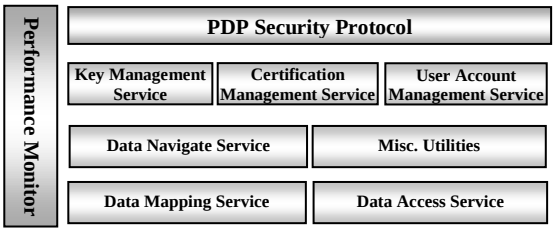


Figure 1. Overview Hierarchy of DCSP-MC Framework

DCSP-MC uses PDP security protocol [27] to interact with independent key servers and authentication servers. To provide online performance profiling service, a Performance Monitor component is implemented, which is responsible for recording the workload requests and helping to analyze the performance of other components. Also, it keeps track of the underlying storage nodes so as to manage the overall storage capacity of the system. 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.

Unlike other mobile storage systems, the goal of DCSP-MC is aiming at efficient data retrieving and convenient file navigating. So, we take most of the efforts on optimizing the efficiency of file navigation and data security when designing DCSP-MC, and details of those implementations can be seen in [5]. As a result, the energy consumption of mobile devices will be significantly increased when using DCSP-MC. For example, when using PDP security protocol, file signature and re-mapping operations will result in much extra energy consumption. So, we redesign DCSP-

MC and incorporate some other component for obtaining better tradeoffs between performance and energy-efficiency.

3.2. Energy-Efficiency Enhanced DCSP-MC

In the current implementation of DCSP-MC, the interactions and framework of the platform is shown in Figure 2. There exist four different entities including mobile client, federated cloud storage, independent security server and DCSP-MC. Mobile devices interact with DCSP-MC through a general Web service portal that deprived from our previous project VRA [28]. In order to relieve the burden of mobile devices, the data security work are shifted to independent server which interacting with DCSP-MC 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.

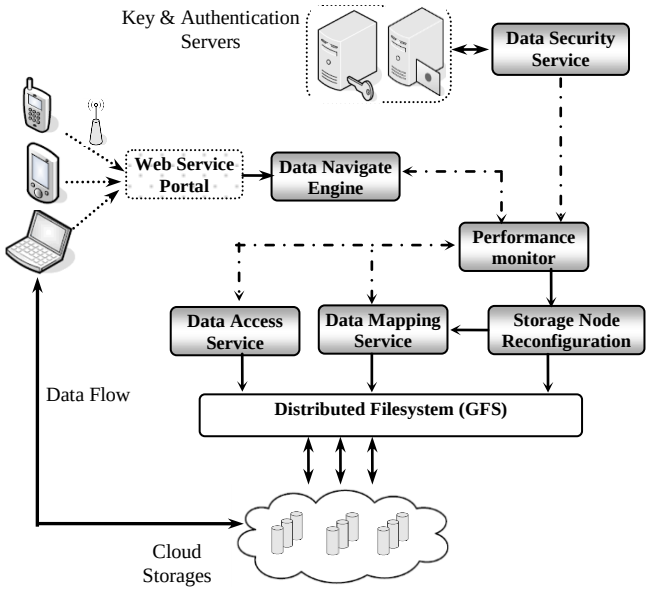


Figure 2. Interaction and Framework of the System

As shown in Figure 2, the Storage Node Reconfiguration component is responsible for scaling up/down the underlying physical devices in distributed filesystem. It executes the reconfiguration operation according to the information collected by Performance monitor. In certain conditions, it will call for Data Mapping Service to complete some work. In the above architecture, data flows are transferred between mobile devices and federated cloud storages directly instead of through DCSP-MC. It is because we do not want to put two much communication load onto DCSP-MC, or else its bandwidth bottleneck might become significantly when the number of users increases. This design also makes DCSP-MC can concentrate on its main objectives, which are data security and efficient data navigation. In the following subsections, we will present the detailed implementation of the above extensions in our DCSP-MC.

3.3. Implementation of Storage Node Reconfiguration Service

In the underlying distributed filesystem (i.e. GFS), master-slave mode is the most common designing paradigm, in which master node is responsible for storing the filesystem namespace and records changes to the metadata of filesystem and slave node are organized across multiple local filesystem. Any user's file stored in the filesystem is split into smaller chunks and distributed across the multiple storage

nodes. Each block is replicated so as to increase the data availability even in case of connectivity failure to nodes. In default setting, the replication factor of GFS is three: one replica of the block on one node in the local rack, another on a different node in the same rack, and the third on a node in some other rack. When the system is running, storage slave node periodically sends a heartbeat message to the master node so as to inform it about its current state including the block report of the blocks it stores. If heartbeat messages are not received for a period of time, the master node will assumes that the corresponding slave node is dropped by some unknown reasons and stops dispatching any other requests to this slave node.

Based on the above descriptions, we can see that energy consumption of the underlying filesystem will be increased when a lot of slave nodes join the filesystem while many of them are dropped during the execution. To address this problem, we designed an energy-aware mechanism for dispatching the storage requests to underlying filesystem. The main idea can be simple summarized as: firstly, a minimized subset of storage nodes are selected as the candidates of underlying slave nodes, which is powered on and accept user's requests like common storage systems; when the current slave node set are not enough for serving user's requests, we pick some other nodes as slave nodes and add them to the system; after every period of time (1 ~ 5 minutes), the master node try to contact those dropped slave nodes and power off them if there is no active user keeping connection with them.

To implement such an energy-aware mechanism, we must firstly monitor the underlying filesystem in real-time manner. The Performance Monitor component in DCSP-MC can monitor all the user's requests as well as other components runtime information. Therefore, we extended the Performance Monitor component by redefining its internal log file format. Secondly, an intelligent approach is required to decide when we should add more slave nodes and how many we should add. In common sense, the utilization of current slave nodes seems to be a good threshold to decide whether more slave nodes are required. For example, we can define a parameter called as $Util_{sn}$ and set it as the threshold at runtime. Unfortunately, our testing experiments show that it is difficult to set suitable $Util_{sn}$ value at runtime, because this parameter fluctuates dramatically when the number of active users is very large (i.e. >1000). In addition, abovementioned replication service will make this parameter inaccurate since we can not monitor all replications and their changes in fine-grained time scale.

Based on our testing experiments, we noticed that dynamically workload (user's storage requests) is key factor that decides how many data node we actually needs. Therefore, we present a 'request fault handler' mechanism to address this problem. It is very like interrupt handle in operation system. As all the storage requests are passed through the Data Mapping Service components (as shown in Figure 1), we insert some extra code in the originally implementation of Data Mapping Service components. More specifically, when the Data Mapping Service components occurs consecutive 'data mapping fault', the above energy-aware mechanism will be triggered. Currently, we set the number of added slave nodes as a constant parameter C_{sn} (default value is 3). It should be noted that bigger C_{sn} might result in energy wastage while small C_{sn} might result in more response time at runtime. In the future, we planning to design an adaptive mechanism to dynamically set C_{sn} according to dynamically workload characteristics.

3.4. Implementation of Data Item Migration Service

As the network that connecting storage nodes are heterogeneous, their performance not only has effects on the data accessing efficiency, but also results in different energy-efficiency in mobile devices. As mentioned previously, rearrangement of data item across storage nodes is an effective way to improve the

performance of storage system. Here, we have to deal with two contradictory problems at the same time. Firstly, balancing the workload across all storage nodes will lead to better long-term QoS performance; while concentrating more requests onto high performance storage nodes and power off some low performance storage nodes will obtain better energy-efficiency while degrading the system QoS performance. In this work, we implement a novel technique called data item migration, which takes advantage of the replication mechanism to obtain both better QoS performance as well as optimized energy-efficiency of mobile devices.

Firstly, we divide the underlying storage nodes into multiple clusters, which are categorized by their hardware performances. In each storage clusters, their performance can be considered as homogeneous. Therefore, we can set the distributed filesystem execute the replication policy as following: all the replicas from the same user's file should be stored onto those storage nodes that belong to the same cluster as possible as we can. It is clear that such a replication policy will make the user data-accessing operations most energy-efficiency if their files are stored in the storage nodes with highest performance. In addition, the energy-efficiency of those storage nodes will be improved because of the near to full utilization. On the other side, the storage node cluster needs to be rearranged after scaling up the number of nodes in the cluster in order to distribute the extra workload from over-utilized nodes to the newly added nodes. So, employ a simple heuristic for rearranging the replications in a cluster, that is, data item migration will triggered only when the underlying storage node is power on/off. More specifically, if the filesystem want to rearrange some data items in a storage node according to their predefined fault tolerance policy, those rearrangements will be suspended until it should be powered on/off. In this way, we can ensure that the fault tolerance mechanism is not be compromised, and at the same time the data item migration operation only lead to least energy consumption.

In the original version DCSP-MC system, we implement an *Aggregating I/O* mechanism to collect as much as possible user's I/O requests and send them to underlying physical storage devices in batch manner. This optimization mechanism is especially useful in the cases where the logical layout of data is different from its physical layout on disks. So, the above data item migration mechanism can be implemented by calling Aggregating I/O mechanism. The only difference is that in original DCSP-MC, aggregating I/O is passively triggered by user's data accessing operations, while in current implementation it can be aggressively performed by the system.

3.5. Implementation of Energy-aware Data Security Service

In the DCSP-MC, we implemented the data security service based on PDP security protocol. In PDP security model, the efficiency of large-size dataset operations will put heavy burden on clients. It also will result in extra energy consumption on mobile devices. More importantly, other security operation (i.e., data signature and verification) also bring about extra energy consumption. To obtain tradeoffs between QoS performance and energy-efficiency, we introduce three security levels for different kinds of user's file, including L_{low}, L_{normal}, L_{high}. As described in (Peng and Guofeng, 2013), we have already implemented a XML-based file-attributes mechanism. Therefore, it is easy to add these security levels into user's file attributes. By these security levels, mobile user's file will use different level of security services when they access their local filesystem. By default, files with L_{low} will not execution any security checking and proofing, and the files with L_{high} attribute will be operated as the original DCSP-MS. As to the L_{normal}, we leave it as a customized parameter that mobile users can set it by themselves. For example, it can be defined as using some third-part software to perform data security service.

By defining these security levels, the mobile storage system can obtain significant energy saving especially for massive multimedia data, which is frequently accessed by mobile users in their daily lives. On the other side, sensitive data will have enhanced security protection according to the user's requirements. Currently, we implemented the security by PDP protocol, however it is noteworthy that other security mechanism or technology can be applied so as to provide better QoS performance, since the security component and underlying filesystem is completely independent in our designing.

4. Experiments Evaluation and Analysis

4.1. Experimental Settings

To investigate the performance of the proposed system, massive experiments are performed on the prototype implementation. In the experiments, we mainly concentrate on the effects of the proposed mechanisms on two metrics including QoS performance and energy-efficiency. As the QoS performance, we mainly evaluate the responsive time and download/upload time; and the energy-efficiency is measured by the measurement, namely Energy Consumption per Gb Transferring (ECGT). In real-world environment, as the number of users is relatively small, in order to investigate the system performance in presence of heaving workloads, we generate various kinds of synthetic user's requests with different characteristics (i.e., file size, arrival interval and *etc*). The experiments are conducted on the cloud platform that deployed in our HP high-performance Network Center. The platform consists of 20 computing nodes (CN_1~CN_{20}) and 7 storage nodes (SN_1~SN_7) as underlying physical resources, which are virtualized by using XCP with version 1.1. To take into account the heterogeneity, we adopts various kinds of equipments that made by different vendors, and the detailed parameters of these nodes are shown in Table 1 and Table 2. The computing nodes are organized as high-performance cluster which is used for provider service for some complicated applications, i.e. online game or media sever. The storage nodes are organized by GFS filesystem.

Table 1. Hardware Parameters of Computing Nodes

Node ID	Processor Architecture	CPU Frequency	Core Voltage	Power Consumption	Memory
CN_1 ~ CN_8	Intel Xeon E5606 quad-core	4 × 2.13 GHz	0.75~ 1.35 V	Idle: > 280 W Peak: 450 W	4 × 4 GB ECC DDR3
CN_9~CN_{20}	AMD A4-3400 dual-core	2 × 2.7 GHz	0.91~1.41 V	Idle: > 120 W Peak: 200 W	2 GB DDR2 800

Table 2. Hareware Parameters of Storage Nodes

Node ID	Storage Framework	Total Capacity	Total Power Consumption	Interface Speed	I/O Cache (per disk)
SN_1 ~ SN_2	IBM Ultrastar RAID	5400 GB	Idle: >550 W Peak: 1280 W	800MB ~1.2 GB	32 MB
CN_3~CN_7	WDC iSCSI Sever	250 GB	Idle: > 180 W Peak: 250 W	200MB ~ 350MB	8 MB

4.2. Performance Evaluation on Quality of Service

In the first experiment, we mainly concentrate on the QoS of the extension of our DCSP-MC. So the basic performance measurements including Responsive Time, Upload/Download Speed are examined by extensive experiments. So, we gradually increase the number of active users from 100 to 2000, and each experiment is conducted five times, and the experimental results are shown in Figure 3.

As shown in Figure 3, we can see that the responsive time keeps below 0.5 seconds, even the number of active users is more than 2000. Such a performance is achieved by the optimized data navigation engine, since this QoS metric is of significant importance to attract more potential users in mobile computing environments. It is noteworthy that in large-scale mobile computing scenario responsive time is often limited by the bandwidth of access point. Therefore, the actual responsive time when using the system might be varied. Here, we mainly care about that whether this metric is sensitive to the number of active users. As shown in the experimental results, it only increases about 0.2 second when user number is increased from 100 to 2000.

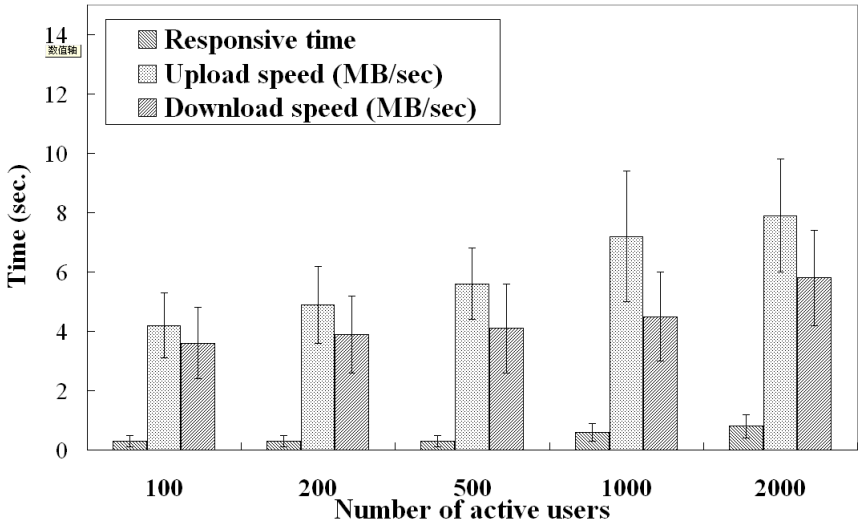


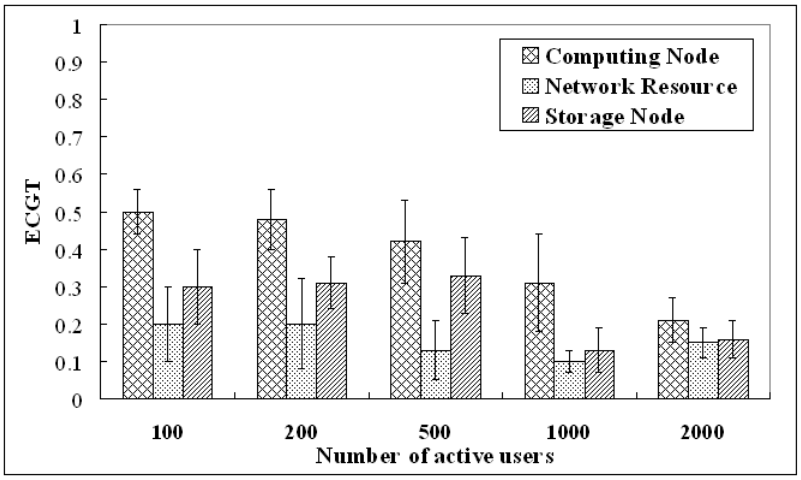
Figure 3. QoS Performance of the System

As to the upload/down speed, we can see that upload speed is slight lower than download speed when the number of active users is small (<200). However, the difference increases about 30% when the number of active users is bigger than 500. By reviewing the logs recorded by the Performance Monitor service (as shown in Figure 2), we noticed that such a difference is caused by the Data Mapping Service component. More specifically, when upload data the system often try to aggregating I/O operations together so as to improve the bandwidth utilization. The implementation of I/O aggregation mechanism can be seen in [5]. As a result, the I/O speed will be degraded when upload operations occur. However, such a performance penalty will be reduced when more and more users take participant in our system, because more concurrent I/O operations will significantly reduce the I/O latency caused by I/O aggregation mechanism. That is why when the number of active users increases from 1000 to 2000 the difference between upload speeds and download speed decreases about 5%.

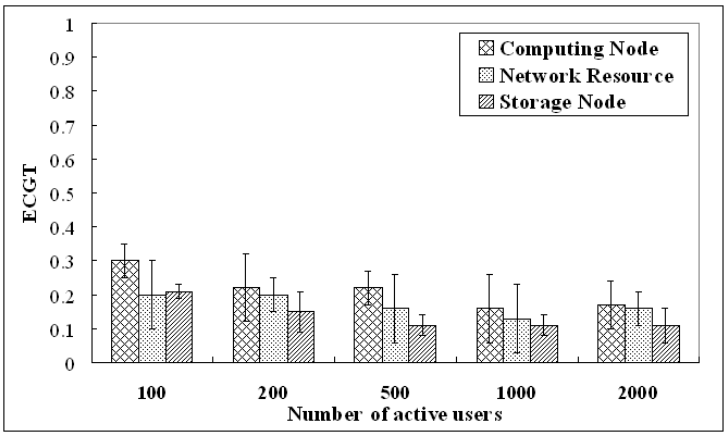
4.3. Performance Comparison on Data-accessing Energy Efficiency

When conducting experiments, the energy consumption not only involves with those storage nodes but also with the other resources that consisting the underlying cloud infrastructure (i.e., computing and network resources). In order to fully take all of them into consideration, we record the three kinds of energy separately. In this experiment, we uses three kinds of synthetic user requests as the workloads (noted as W1 ~ W3), each of them has distinctive characteristics. As to W1, all the user's requests are small size files (<10 MB) and maximal concurrent requesting number is 500; in W2, the user's requests are large-size files (>1 GB) and the maximal

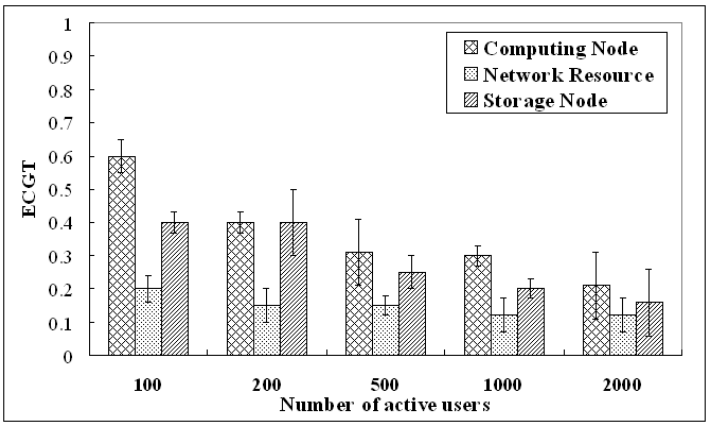
concurrent requesting number is 50; In W3, the requesting file size and concurrent requesting number are randomly distributed. The performance metric we used is Energy Consumption per Gb Transferring (ECGT), and the result is shown in Figure 4(a)~(c).



(a) W1



(b) W2



(c) W3

Figure 4. Comparison on Energy-efficiency Measurements

As shown in the experimental results, the ECGT measurements for three kinds of resources are decreased with the increasing of active users. This is because that ECGT is calculated total energy consumption dividing by data accessing volume. When the number of active users is small, many resources is working in idle states and is still consuming energy which make the resource energy-efficiency is very low). With the increasing of users, the resource utilization increases that in turn increase the energy-efficiency metric ECGT. In three kinds of resources, it is clear that the ECGT metric of network resources are relative stable comparing with other two kinds of resources. The reason is that we set all routers and switches working in constant power modes. As to the computing nodes, it is configured by DVFS mechanism which can scale up/down the processor's working frequency so as to save energy consumption when the workload is varying. In this experiment, the computing nodes are mainly executing file-forwarding and data mapping algorithm. In fact, many of computing nodes does not execute any practical jobs. As a result, we can see that the ECGT metric of computing node is very high especially when the number of active users is very small. With the increasing of active users, the experimental platform will create some virtual machines for delegating the data accessing service in distributed manner. Therefore, the computing nodes become more energy-efficient in this situation.

As to storage node, we noticed that different workload characteristics will have significantly effects on its ECGT metric. For example, as shown in Figure 4(a), when the workload is full of small size file requests, the ECGT metric is generally higher that that for W2. It is well know that extensive I/O request will result in heavy overheads on the virtualization platform, such as frequent VM switching. However, in our system this problem can be avoided by I/O aggregation mechanism. That is why the system works better for W1 than for W2. Another reason is that large-size file requests often involves longer time for executing security protection. If we can not differentiate the security requirements imposed by users, such a overhead will result in more energy consumption as well as performance penalty. That is way we implement energy-aware data security in our system. In real-world mobile environments, most of users only access small/mediate size files which means that optimization for these kinds of file accessing is more benefits that others.

5. Conclusion

In this work, we design and implement a mobile storage system which incorporates several innovative mechanisms for providing better file query experience, enhanced data security, and device energy saving. Extensive experiments are conducted to evaluate the performance of the system in terms of QoS performance as well as energy-efficiency and the results indicate that it is effective to obtain better tradeoffs between QoS performance and energy consumption. In the future, we planning incorporate some adaptive mechanisms for satisfying the other user's QoS requirements, i.e. SLA-based pricing negotiation, energy-aware load balancing and *etc*.

References

[1] Y. Fei, L. Zhong and N. K. Jha, "An energy-aware framework for dynamic software management in mobile computing systems", ACM Transactions on Embedded Computing Systems, vol. 7, no. 3, **(2008)**, pp. 1-31.
[2] N. Fernando, S. W. Loke and W. Rahayu, "Mobile cloud computing: A survey", Future Generation Computer Systems, vol. 29, no. 1, **(2013)**, pp. 84-106.
[3] M. G. Khatib and P. H. Hartel, "Optimizing MEMS-Based Storage Devices for Mobile Battery-Powered Systems", ACM Transactions on Storage, vol. 6, no. 1, **(2010)**, pp. 1-28.

[4] K. Kim, T. Xu and Y. Cai, "ELIAS: An Efficient Storage Underlay for Mobile Peer-to-Peer Systems", IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 11, **(2011)**, pp. 1851-1861.
[5] X. Peng and Y. Guofeng, "DCSP-MC: dependable cloud-based storage platform for mobile computing", International Journal Networking and Virtual Organisations, vol. 12, no. 2, **(2013)**, pp.133-148.
[6] Y. J. Zheng and A. T. S. Chan, "MobiGATE: A mobile computing middleware for the active deployment of transport services", IEEE Transactions on Software Engineering, vol. 32, no. 1, **(2006)**, pp. 35-50.
[7] C. X. Mavromoustakis, C.X. and H. D. Karatza, "Under storage constraints of epidemic backup node selection using HyMIS architecture for data replication in mobile peer-to-peer networks", Journal of Systems and Software, vol. 81, no.1, **(2008)**, pp. 100-112.
[8] G. Goth, "Mobile security issues come to the forefront", IEEE Internet Computing, vol. 16, no. 3, **(2012)**, pp. 7-9.
[9] S. Y. Wu and S. C. Chou, "Preventing information leakage in mobile applications with object-oriented access control lists and security monitor encapsulation", Journal of Information Science and Engineering, vol. 25, no. 6, **(2009)**, pp. 1921-1937.
[10] C. Li and L. Li, "Tradeoffs between energy consumption and QoS in mobile grid", Journal of Supercomputing, vol. 55, no. 3, **(2011)**, pp. 367-399.
[11] A. Ranabahu, P. Anderson and A. Sheth, "The cloud agnostic e-science analysis platform", IEEE Internet Computing, vol. 15, no. 6, **(2011)**, pp. 85-89.
[12] L. Wang, M. Kunze, J. Tao and G. V. Laszewski, "Towards building a cloud for scientific applications", Advances in Engineering Software, vol. 42, no. 9, **(2011)**, pp. 714-722.
[13] M. P. Papazoglou and W. J. V. D. Heuvel, "Blueprinting the cloud", IEEE Internet Computing, vol. 15, no. 6, **(2011)**, pp. 74-79.
[14] Amazon Simple Storage Service: http://www.amazon.com/s3/, 2015.
[15] Windows Live Mesh: http://www.mesh.com/, 2015.
[16] Mozy homepage: http://mozy.com, 2015.
[17] Symantec's Protection Network: http://www.spn.com/, 2015.
[18] Vrable, M., Savage, S., Voelker, G.M. 'Cumulus: Filesystem backup to the cloud'. ACM Transactions on Storage, Vol. 5, No. 4, pp. 1-28, 2009.
[19] Jungle Disk: http://www.jungledisk.com/, 2015.
[20] W. Zheng, P. Xu, X. Huang and N. Wu, "Design a cloud storage platform for pervasive computing environments", Cluster Computing, vol. 13, no. 2, **(2010)**, pp. 141-151.
[21] P. Daniel and F. Jason, "EnsemBlue: integrating distributed storage and consumer electronics", Proceedings of Symp. on Operating Systems Design and Implementation, Seattle, U.S., **(2006)**, pp. 221-230.
[22] Google Desktop web page: http://desktop.google.com, 2015
[23] Beagle: http://beagle-project.org, 2015
[24] C. Wang, Q. Wang, K. Ren, N. Cao and W. Lou, "Toward Secure and Dependable Storage Services in Cloud Computing", IEEE Transactions on Services Computing, vol. 5, no. 2, **(2012)**, pp. 220-232.
[25] P. Mahajan, S. Setty and S. Lee, "Depot: cloud storage with minimal trust", ACM Transactions on Computer Systems, vol. 29, no. 4, **(2011)**.
[26] G. Sanjay, G. Howard and S. Leung, "The google file system", Proceedings of ACM Symposium on Operating Systems Principles, Bolton Landing, NY, U.S., **(2003)**, pp. 29-43.
[27] G. Ateniese, R. Burns and R. Curtmola, "Provable data possession at untrusted stores", Procedings of ACM Conference on Computer and Communication Security, New York, U.S., **(2007)**, pp. 598-609.
[28] P. Xiao and Z. G. Hu, "A novel qos-based co-allocation model in computational grid", Proceedings of IEEE Global Communication Conference, New Orleans, U.S., **(2008)**, pp. 1562-1566.

Authors

Ning Han, he received his bachelor degree in Beijing University of Science and Technology, and now persuading master degree in Xiangtan University. Currently, he works in the HP High Performance Lab of Hunan Institute of Engineering as a senior networking engineer. His research interests include complex networking analysis, distributed computing, information security technology, fault-tolerance in distributed systems.