Original Research Paper

Scalability Services in Cloud Computing Using Eyeos

[1]Deivendran, P. and [2]E.R. Naganathan

[1]Manonmaniam Sundaranar University, Tirunelveli, India
[2]Department of Computer Science and Engineering, Hindustan University, India

Article history
Received: 03-12-2013
Revised: 17-03-2014
Accepted: 11-08-2014

Corresponding Author:
Deivendran, P.
Manonmaniam Sundaranar University, Tirunelveli, India
Email: deivendran77p@yahoo.com

Abstract: Cloud storage enables users to remotely store their data and benefit of the demand high quality cloud applications without the difficulty of local hardware and software management. Though the benefits are clear, such a service is also reliable to the users' physical possession of their outsourced data, which inevitably poses new security risks towards the recovery of the data in cloud. In order to address this new problem and further achieve a secure and useful cloud storage service, we propose in this study a flexible distributed storage integrity mechanism, utilizing the homomorphic token and distributed data. The proposed design allows users to check the cloud storage with very lightweight communication and computation cost. The auditing result not only ensures strong cloud storage correctness guarantee, efficiency, but also simultaneously to access data error localization, i.e., the identification of misbehaving server. Considering the cloud data are dynamic in nature, the proposed design future supports secure and efficient dynamic operations on outsourced data, including block modification, update, deletion and append. The proposed scheme is highly efficient and secure against Byzantine failure, malicious data modification attack and even server colluding attacks.

Keyword: Homomorphic, Service, Storage, Third Party Auditor, Attack

Introduction

Scalability is the ability of an application to be scaled up to meet demand through replication and the distribution of requests across a pool or farm of servers. It's the traditional load balanced model and it's an integral component of cloud computing environments. Vertical scalability is the ability of an application to scale under load; to maintain performance levels as the number of concurrent requests increases. While load balancing solutions can certainly assist in optimizing the environment in which an application needs to scale by reducing overhead that can negatively impact performance (such as TCP session management, SSL operations and compression/caching functionality) it can't solve core problems that prevent vertical scalability.

The problem is that a single database table or SQL query that is poorly constructed can destroy vertical scalability and actually increase the cost of deploying in the cloud. Because you generally pay on a resource basis, if the application isn't scaling up well it will require more resources to maintain performance levels and thus cost a lot more. Cloud computing isn't going to magically optimize code or database queries or design database table with performance in mind, that's still squarely in the hands of the developers regardless of whether or not cloud computing is used the deployment model.

XAMPP on the Web

At the beginning it is important to answer why to choose XAMPP among so many server packages available? Well, there are two strong advantages of it. First-its configuration is so easy, that even a child can do it. It particularly is minimized to unzip archive and run setup batch. Second-XAMPP is extremely

Science Publications

© 2015 Deivendran, P. and E.R. Naganathan. This open access article is distributed under a Creative Commons Attribution (CC-BY) 3.0 license.

portable (Amazon, 2009). Moving it from one directory or drive to another requires only one run of setup-xampp.bat. You can even install it on USB stick and have your private web server along with your apps go anywhere with you and to be available on any computer; you plug your USB stick. As I read other Wiki articles on how many problems people have with installing and configuring other servers or server pack I think it can be simpler than with XAMPP. After that, execute xampp-control. Exe to run any web server component (like Apache, My SQL) you need or to install it as system service. If you pass this step, you can open your browser and point it to local host to see XAMPP welcome page (Shah *et al*., 2008), which consist of some modules for checking/granting security to your web apps run under this server. If everything is double checked and all issues all solved, you may delete contents of http subfolder in you XAMPP directory (Amazon, 2008) Differences between setup version (EXE) and setup-less version (ZIP) are at least questionable (half the size for the first one) and I still can't find out how do the achieve it?) But for this tutorial and for advantages of portability we will use ZIP version.

Own Cloud Operating System with Eyes

A cloud OS simply refers to an operating system (or an interface filled with a complete suite of desktop applications) that resides on the Web and you can access to it anytime, anywhere as long as you have an Internet connection. While there are plenty of cloud OS out there that you can sign up and use for free, there might be instances where you want to have your own dedicated cloud OS. First of all, signing up a free account with third-party (Kincaid, 2009) cloud OS often means that you have limited file storage space and all your data are stored in other people's server. Next, the connection speed is dependent on the number of active users at any time. The more popular the site is, the slower it will get when you are using it. If what you want is your own dedicated Web OS (Juels *et al*., 2007) that you can use to manage your online stuff and also to provide an environment to collaborate with your colleagues/partners, then eye OS is the software for you. Eye OS is free and open source cloud OS software that you can install on your own Web server. One thing that, I like about eye OS is its small file size and ease of installation. The whole package is only 2.5 MB in size and the installation required almost zero configuration (well, there are still several steps involved) and anyone who know how to use a FTP program can get it up and running in no time.

User Classes and Characteristics:

- Third Party Auditor: Collect the data's from the user and generate tokens for that particular file for security
- Cloud Service Provider: Collect file from third party auditor generate signature like token and send cloud server

Constraints in Analysis:

- Constraints as informal text
- Constraints as operational restrictions
- Constraints integrated in existing model concepts
- Constraints as a separate concept
- Constraints implied by the model structure

Constraints in Design:

- Determination of the involved classes
- Determination of the involved objects
- Determination of the involved actions
- Determination of the require clauses
- Global actions and constraint realization

Constraints in Implementation

A hierarchical structuring of relations may result in more classes and a more complicated structure to implement. Therefore it is advisable to transform the hierarchical relation structure to a simpler structure such as a classical flat one. It is rather straightforward to transform the developed hierarchical model (Wilson, 2006) into a bipartite, flat model, consisting of classes on the one hand and flat relations on the other. Flat relations are preferred at the design level for reasons of simplicity and implementation ease. There is no identity or functionality associated with a flat relation. A flat relation corresponds with the relation concept of entity-relationship modeling and many object oriented methods.

To ensure the security and dependability for cloud data storage under the aforementioned adversary model, we aim to design efficient mechanisms for dynamic data verification and operation and achieve the following goals: Storage correctness (Arrington, 2006) to ensure users that their data are indeed stored appropriately and kept intact all the time in the cloud. Fast localization of data error: To effectively locate the malfunctioning server when data corruption has been detected. Dynamic data support (SMI, 2009) to maintain the same level of storage correctness assurance even if users modify, delete or append their data files in the cloud. Dependability (Ateniese *et al*., 2008) to enhance data availability against Byzantine failures, malicious data modification and server colluding attacks, i.e., minimizing the effect brought by data errors or server failures. Lightweight: To enable users to perform storage correctness checks with minimum overhead.

Pass Agent Architecture:

- Tunnel module
- Registration module
- Registration Server
- Service Deployment

Tunnel Module

This module is responsible for establishing the tunnel alive. The tunnel negotiation is accomplished via SSL over TCP. Once the tunnel is setup, the tunnel module can receive data from the tunnel and process accordingly before sending it to the service dispatcher.

Registration Module

In order for a SaaS application to access an on-premise service, the enterprise administrator registers the accessible on-premise services to the PASS. The registration module provides a web interface for administrators to perform this task on-demand (Arrington, 2006). The registered service will be added to the database as direct service. Meanwhile, this module also synchronizes the service registration with the synchronization, server. For security purpose, during the synchronization, the PA must present its certificate to PS over HTPPS for authentication.

PASS Architecture

As discussed in our architecture, in case the user does not have the time, feasibility or resources to perform the storage correctness verification, he can optionally delegate this task to a dependent third party auditor, making the cloud storage publicly verifiable. Third Party Auditor (TPA) Fig. 1 an optional TPA, who has expertise and capabilities that users may not have, is trusted to assess and expose risk of cloud storage services on behalf of the users upon request. Storage correctness: To ensure users that their data are indeed stored appropriately and kept intact all the time in the cloud.

Registration Server

The registration server provides two interfaces. One is a secured web interface through which administrators can manage PASS agents and services Fig. 6. The other interface is for PA's registration module to synchronize services. This interface is different from a general web interface in that it requires client's certificate by which PASS agents are authenticated. The registered service and agents will be stored in a database, in the actual implementation; a run-time copy is pushed to the routing engine for performance enhancement.

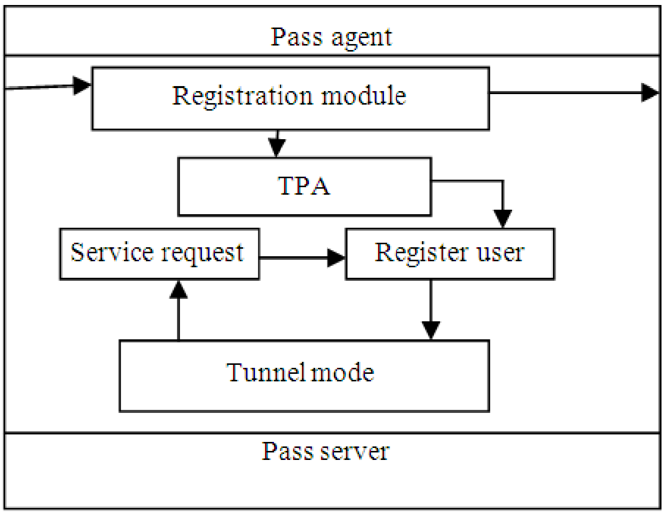


Fig. 1. PASS architecture

Service Deployment

This describes the proposed architecture to implement multi-tenancy for an SOA platform. Since the solution deals with security and implementation related complications; we describe it in terms of a concrete SOA platform Fig. 7. When a client sends a message addressed to a particular tenant's service, that request must indicate the tenant in some manner. The default approach in WSO2 Carbon is to add the tenant name to the URL as follows.

Nonfunctional Requirements

Their scheme combines spot-checking and error correcting code to ensure both possession and irretrievability of files on archive service (Juels *et al*., 2007) systems. Built on this model and constructed a random linear function based homomorphic authenticator which enables unlimited number of challenges and requires less communication overhead due to its usage of relatively small size of BLS signature. Their scheme utilized public key based homomorphism (Arrington, 2006) tags for auditing the data file Fig. 5. However, the pre-computation of the tags imposes heavy computation overhead that can be expensive for an entire file. In their subsequent work, Attendee *et al.* described a PDP scheme (Wilson, 2006) that uses only symmetric key based cryptography. This method has lower-overhead than their previous scheme and allows for block updates, deletions and appends to the stored file, which has also been supported in our work. It is not yet clear how the work can be adapted to cloud storage scenario where users (Shah *et al*., 2008) no longer have the data at local sites but still need to ensure the storage correctness efficiently in the cloud. The software may be safety-critical. If so, there are issues associated with its integrity level.

The software may not be safety-critical although it forms part of a safety-critical system. For example, software may simply log transactions. If a system must be of a high integrity level and if the software is shown to be of that integrity level, (Wang *et al*., 2009) then the hardware must be at least of the same integrity level. There is little point in producing 'perfect' code in some language if hardware and system software (in widest sense) are not reliable.

If a computer system is to run software of a high integrity level then that system should not at the same time accommodate software of a lower integrity level. Systems with different requirements for safety levels (Kincaid, 2009) must be separated. Otherwise, the highest level of integrity required must be applied to all systems in the same environment.

Cloud Architecture Design

Cloud Server (CS)

An entity, which is managed by Cloud Service Provider (CSP) to provide data storage service and has significant storage space and computation resources In order to achieve assurance of data storage correctness and data error localization simultaneously. Upon receiving challenge, each cloud server computes a short "signature" Fig. 2 over the specified blocks and returns them to the user. Fast localization of data error: To effectively locate the malfunctioning server when data corruption has been detected.

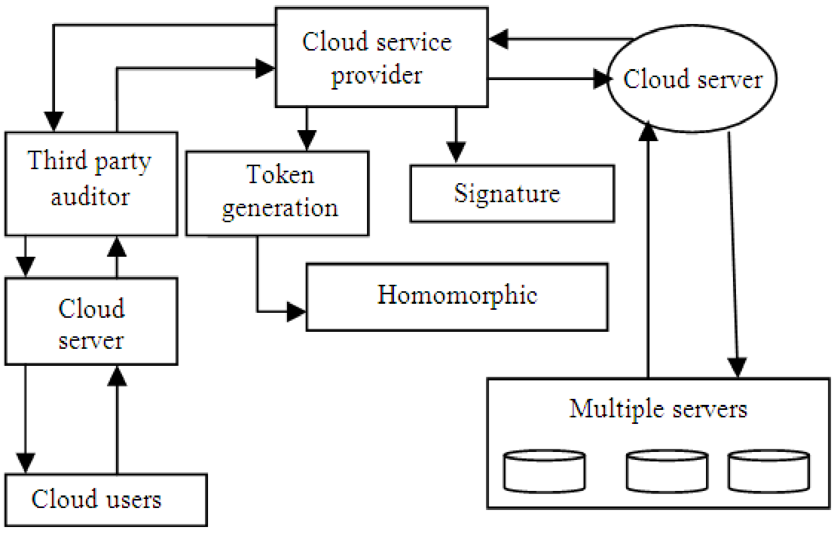


Fig. 2. Cloud Architecture user and auditor

Cloud Computing Computational Approaches

Cloud computing has computational and sociological implications. In computational terms cloud computing is described as a subset of grid computing concerned with the use of special shared computing resources (Shah *et al*., 2007). For this reason it is described as a hybrid model exploiting computer networks resources, chiefly Internet, enhancing the features of the client/server scheme. From a sociological standpoint on the other hand, by delocalizing hardware and software resources cloud computing changes the way the user (SMI, 2009) works as he/she has to interact with the "clouds" (Juels *et al*., 2007) on-line, instead of in the traditional stand-alone mode.

Cloud Server (CS): An entity, which is managed by Cloud Service Provider (CSP) to provide data storage service and has significant storage space and computation resources In order to achieve assurance of data storage correctness and data error localization simultaneously, (Ateniese *et al*., 2007) our scheme entirely relies on the pre-computed verification tokens. Later, when the user wants to make sure the storage correctness for the data in the cloud, he challenges the cloud servers with a set of randomly generated block indices. Upon receiving challenge, each cloud server computes a short "signature" (Kincaid, 2009) over the specified blocks and returns them to the user. Fast localization of data error: To effectively locate the malfunctioning server when data corruption has been detected.

```
//Public access service method creation//
Myblob(service.set-containe, x_ms_blob_Public access)
{
blob_service=Blob service (account_name = 'pytool',
ount_key='07iY9G.1r7A= =')
stoage_container_name=pyfiles'blob_service.create_con
tainer (storage_container_name)
blob_service.set_container_acl(storage_container_name,
x_ms_blob_public_access='container')
}
}

//Upload a text file and set appropriate content type//
Myblob (open s, r)
{
myblob = open(r'foo.txt', 'r').read()
blob_name='hello.txt'blob_service.put_blob(storagecont
ainer_name, _type='BlockBlob')
blob_service.set_blob_properties
(storage_container,blob_name,my_blob_content_type='t
ext/plain')}

// Upload a photo and set appropriate content type//
myblob = open(r'clouds.jpeg', 'r').read()
Myblo_upload(s,x)
{
blob_name='clouds.jpeg'blob_service.put_blob(storage_
container_name,blob_name,myblob,x_ms_blob_type='B
lockBlob')
blob_service.set_blob_properties(storage_container_nam
e,blob_name, lob_content_type='image/jpeg')
}
blobs=lob_service.list_blobs(container_name)
for blob in blobs:
print (blob.name)
print(blob.url)
```

Results

A PASS system has been implemented using eye OS/. Net services based on the architecture described (Wilson, 2006) experiments were conducted to evaluate the performance of the PASS system with regard to processing time and throughput. It is compared with the case where a reverse proxy is deployed for integration as it is approach used in SaaS integration despite the deficiencies.

Performance Comparison (RTT)

The hardware depicts the performance of PASS with regard to the average Round-Trip Time (RTT) Fig. 3, the number of simultaneous requests. In this experiment, the test client sent requests to the test server and we calculated the average round trip time over all requests (Ateniese *et al*., 2008). The test was repeated multiple times by spawning different number of threads on the same test client.

Throughput Comparison

The throughput is relatively flat with the increase of the number of threads. Note that the absolute value may not be very useful in this case as the page size is approximately 8 Kbyte Fig. 4. We are more interested in the difference between PASS and the reverse proxy under the same testing setting.

System Performance in Real Data

Two PASS were deployed in two different networks. The two test clients send requests to the test server through the different PASS and the average RTT was calculated Table1.

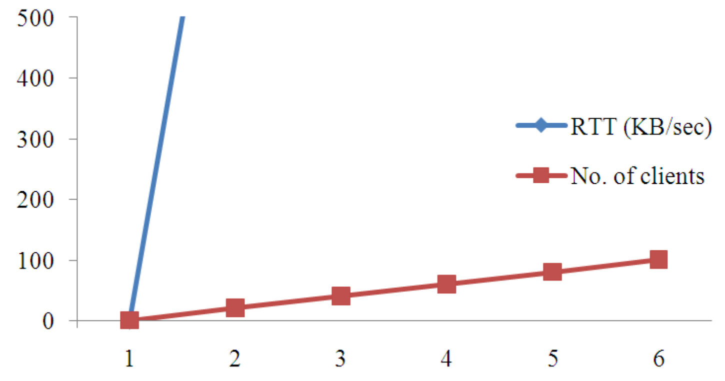


Fig. 3. RTT comparison

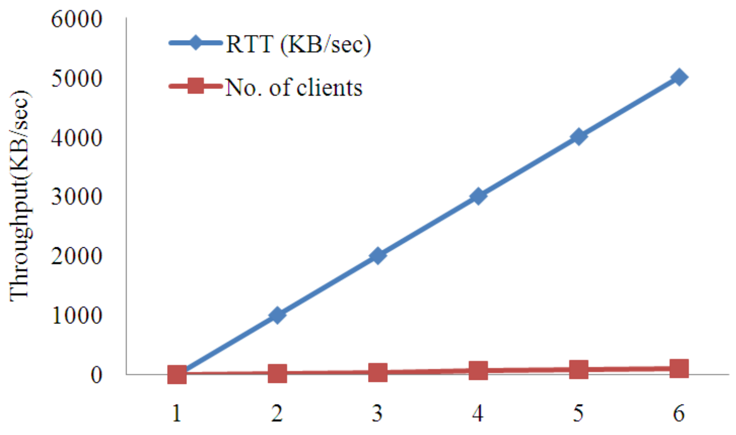
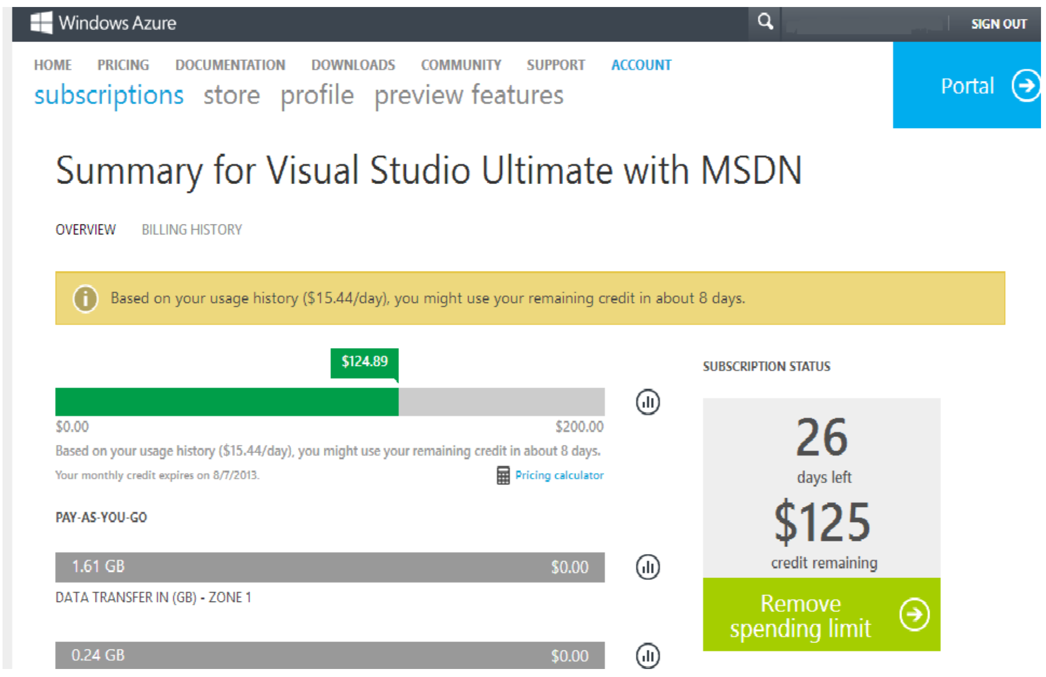


Fig. 4. Throughput comparison



Summary for Visual Studio Ultimate with MSDN

OVERVIEW BILLING HISTORY

Based on your usage history ($15.44/day), you might use your remaining credit in about 8 days.

$124.89

$0.00 $200.00

Based on your usage history ($15.44/day), you might use your remaining credit in about 8 days.

Your monthly credit expires on 8/7/2013. Pricing calculator

PAY-AS-YOU-GO

1.61 GB $0.00

DATA TRANSFER IN (GB) - ZONE 1

0.24 GB $0.00

SUBSCRIPTION STATUS

26
days left

$125
credit remaining

Remove spending limit

Fig. 5. Service registration

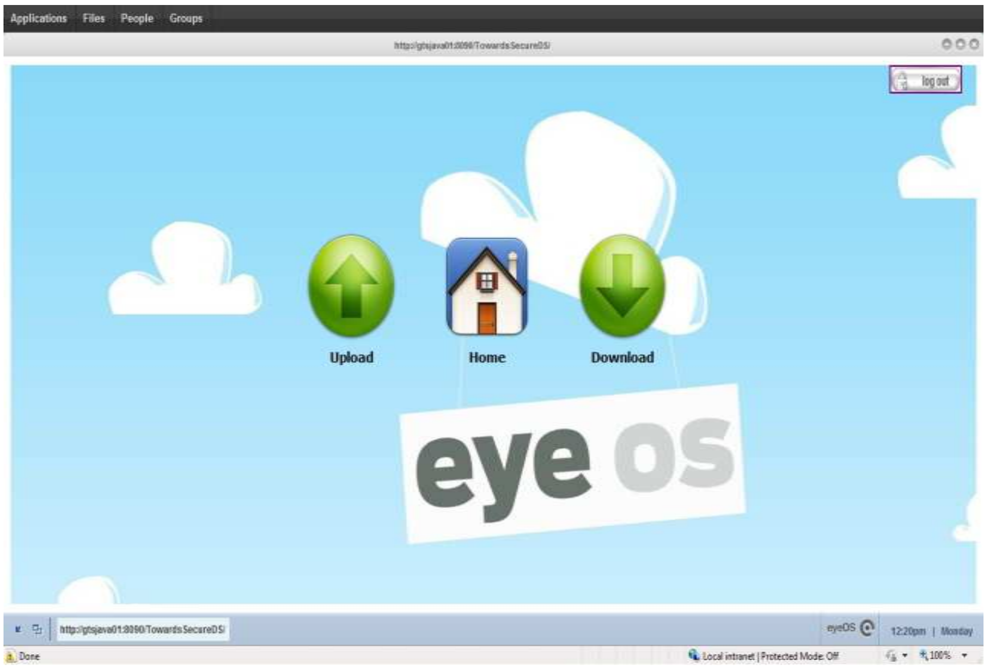
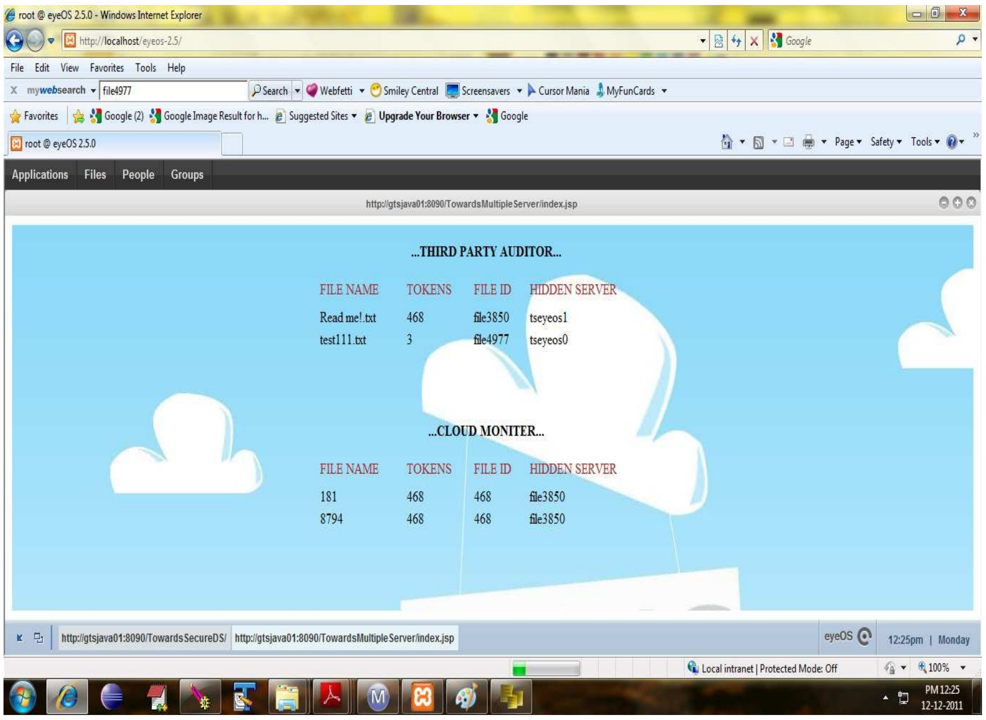


Fig. 6. Home window


root @ eyeOS 2.5.0 - Windows Internet Explorer
http://localhost/eyeos-2.5/
Applications
Files
People
Groups
http://gtsjava01:8090/TowardsMultipleServer/index.jsp
12:25pm | Monday

...THIRD PARTY AUDITOR...

FILE NAME	TOKENS	FILE ID	HIDDEN SERVER
Read me!.txt	468	file3850	tseyeos1
test111.txt	3	file4977	tseyeos0

...CLOUD MONITER...

FILE NAME	TOKENS	FILE ID	HIDDEN SERVER
181	468	468	file3850
8794	468	468	file3850

Fig. 7. Service creation data

Table 1. Performance analysis

Process	Verizon eye OS	Optimum online
Direct access	410	480
PASS	492	530
Overhead	70	50

Conclusion

User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required. We implemented and tested a working system based on PASS architecture. The experimental study shows that PASS solution is feasible.

Acknowledgement

Based on this model, we propose a scheme for assigning Web services to server nodes in the cloud in order to improve the scalability of composite services. We have presented a simulated Experimental evaluation of our scheme. We are in the process of deploying our scalability management scheme in the cloud environment and experimentally study its performance.

Author's Contributions

All authors equally contributed in this work.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

Amazon, 2008. Amazon s3 availability event. Amazon.com.

Amazon, 2009. Amazon web services (aws).

Arrington, M., 2006. Gmail disaster: Reports of mass email deletions.

Ateniese, A., R. Burns, R. Curtmola, J. Herring and L. Kissner *et al*., 2007. Provable data possession at untrusted stores. Proceedings of the 14th ACM Conference on Computer and Communications Security, Oct. 28-31, ACM, New York, pp: 598-609. DOI: 10.1145/1315245.1315318

Ateniese, G., R.D. Pietro, L.V. Mancini and G. Tsudik, 2008. Scalable and efficient provable data possession. Proceedings of the 4th international Conference on Security and Privacy in Communication Networks, Sep. 22-25, ACM, New York, DOI: 10.1145/1460877.1460889

Juels, A., S. Burton, J. Kaliski, 2007. Pors: Proofs of retrievability for large files. Proceedings of the 14th ACM Conference on Computer and Communications Security, Oct. 28-31, ACM, New York, pp: 584-597. DOI: 10.1145/1315245.1315317

Kincaid, J., 2009. Media ax/the linkup closes its doors.

Shah, M.A., M. Baker, J.C. Mogul and R. Swaminathan, 2007. Auditing to keep online storage services honest. Proceedings of the 11th USENIX Workshop on Hot Topics in Operating Systems, USENIX Association Berkeley, CA, USA, pp: 1-6.

Shah, M.R. Swaminathan and M. Baker, 2008. Privacy-preserving audit and extraction of digital contents. Cryptology e Print Archive.

SMI, 2009. Building customer trust in cloud computing with transparent security. Sun Microsystems, Inc.

Wang, C., Q. Wang, K. Ren and W. Lou, 2009. Ensuring data storage security in cloud computing. Proceedings of the 17th International Workshop on Quality of Service, Jul. 13-15. IEEE Xplore Press, Charleston, SC, pp: 1-9. DOI: 10.1109/IWQoS.2009.5201385

Wilson, S., 2006. Appengine outage. Payment Processor Breach May Be Largest Ever, 20-27.