# NETWORK STORAGE VIRTUALISATION AND MANAGEMENT

***BENARD ONG'ERA OSERO***
*LECTURER*
*DEPARTMENT OF COMPUTER SCIENCE*
*CHUKA UNIVERSITY*
*CHUKA*
*ongerab@yahoo.com*

ABSTRACT
*This paper presents Network Storage virtualization model (NSVM), a demonstration of how the cloud can be utilized to transfer data directly between storage and the client on the network. The concepts demonstrated in this paper indicates how network attached devices can eliminate the server as a bottleneck and thus improving both performance and scalability. The network attached devices have a capability of allowing the client to independently access data directly from the storage through the independent drive object services provided. To fully realize the full potential of this technology security challenges paused by interaction models are carefully analyzed and provide a security implementation model and thus ensuring security and data privacy. The paper will demonstrate through experimentation and graphical representation, of results obtained, how secured network devices, can perform better than the classical store and forward processes.*

## KEYWORDS
Network Attached Devices, Storage virtualization, Security.

## INTRODUCTION

The current systems cannot withstand the exponential processing requirements that they were not initially designed to handle. The client server processing that exist in most systems today are essentially characterized by store and forward processes which are slow due to the fact that the requests are processed one at a time; therefore the file server in this case acts as a bottleneck. These bottlenecks arise because a single "server" computer receives data from the storage (peripheral) network and forwards it to the client (local area) network while adding functions such as concurrency control and metadata consistency [www.pdl.cs.cmu.edu].
This paper will demonstrate how storage virtualization can achieve high performance and scalability through a virtualized model named *NSVM* (Network Storage Virtualisation Model). The model will be explained in details in the subsequent illustrations and experimentations.

### STORAGE MANAGEMENT PROBLEM DEFINITION

A good system model should exhibit high performance and high scalability to withstand exponentially increasing demands, the current systems fall short of such salient features; instead they are encumbered by low speeds, and can't scale considerably to cater for the users needs.
The above short comings are currently solved by manual techniques and administrators who, are not properly equipped to undertake such complex optimizations and configuration maneuvers. It is worth to note that human expertise is scarce and expensive making the storage management process a costly undertaking.
The figure below shows the interaction between the clients and the file server over storage area network where a file has to be downloaded to the file server before the client requesting for it.

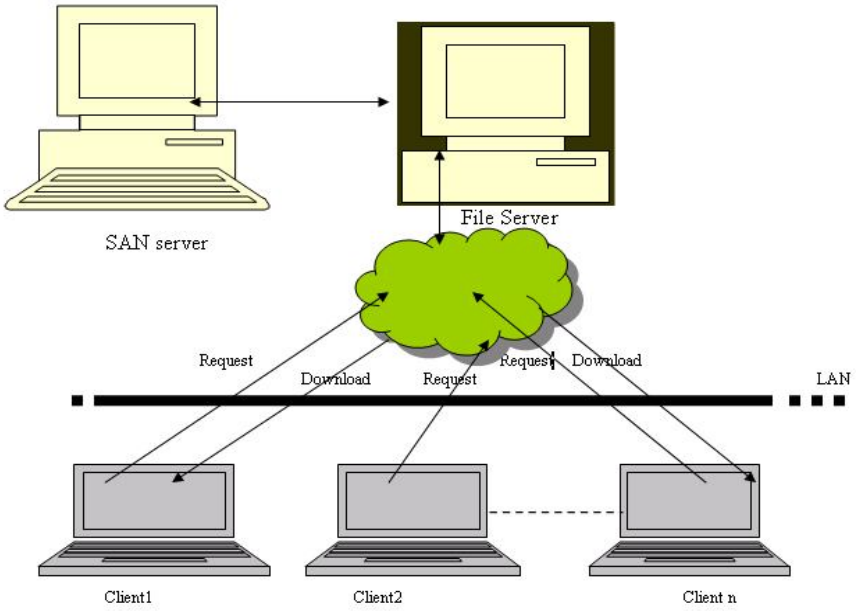


**Fig 1.0 Showing Classical store and forward processes through the file server.**

## RESEARCH OBJECTIVES

The general objective of the study was to determine the effect of virtualizing server on a network and and the effect of store and forward processes under the following sub-headings:

- Availability – To demonstrate that Parallel hosts attached to an array of shared storage devices via a switched storage manager can achieve scalable bandwidth and a shared virtual storage abstraction hence increasing availability.
- To demonstrate how security can be implemented
- To prove using experimentation to show that NASD can achieve both high performance and scalability.
- To implement a different approach of using Network attached storage.
- Design and develop a virtualized environment.

## RESEARCH HYPOTHESIS

1. Process management is faster than data managent.
2. Virtualized disks performs faster than non-virtualied disks .

## RESEARCH METHODOLOGY

1. Design of the client using java APIs.
2. Design of the server with Ruby.
3. Develop the interaction model.
4. The virtual Environment design and Implementation.
5. Network design and co-ordination.

## RELATED WORK

An experimental approach of implementing real-time processing systems in a desktop virtualization environment; demonstrating the use Java programming language, open source APIs such as jPOS (www.jpos.org) and desktop virtualization systems such as Virtual Box to realize a near real life scenario on how to implement systems coordination. It uses Artificial Intelligence techniques of coordination in a more practical way by using XML and TCP/IP data exchange to implement a coordination media, Java programs as agents or co-ordinables and ISO8583 standards as coordination rules. (Ochomo, 2011)

## VIRTUAL ENVIRONMENT ARCHTECTURE

VM-ware technology provides a fundamentally better way to manage storage resources for a virtual infrastructure, giving an organization the ability to:

- Improve storage resource utilization and flexibility.
- Reduce management overhead and increase application uptime.
- Leverage and compliment existing storage infrastructure.

The conceptual design in figure 1.1 below was implemented and tested (ZablonO, 2009).

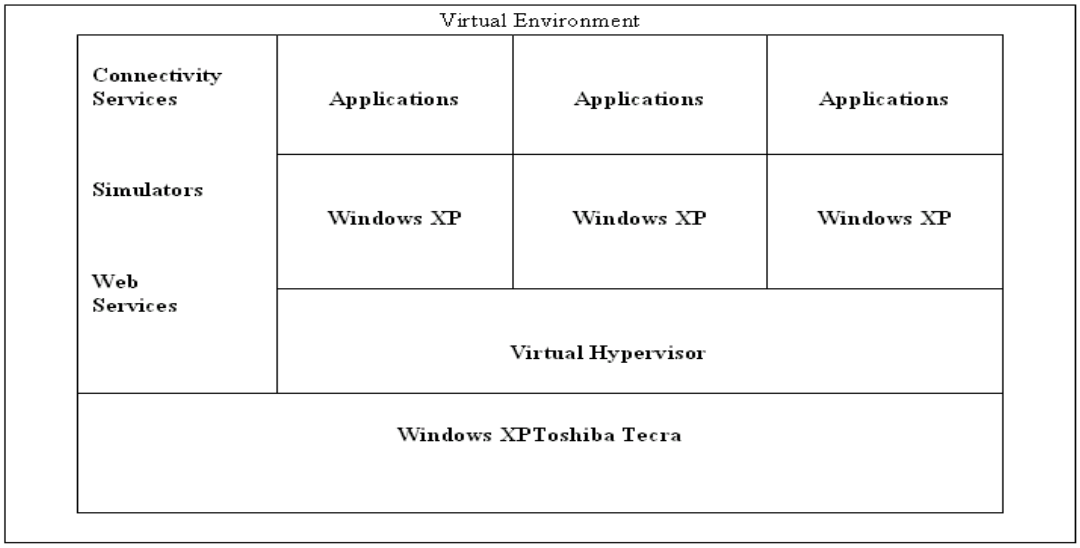


**Fig 1.1: Virtual environment design**

The V-model environment discussed in this paper was, implemented on the windows XP platform Virtual Storage model.

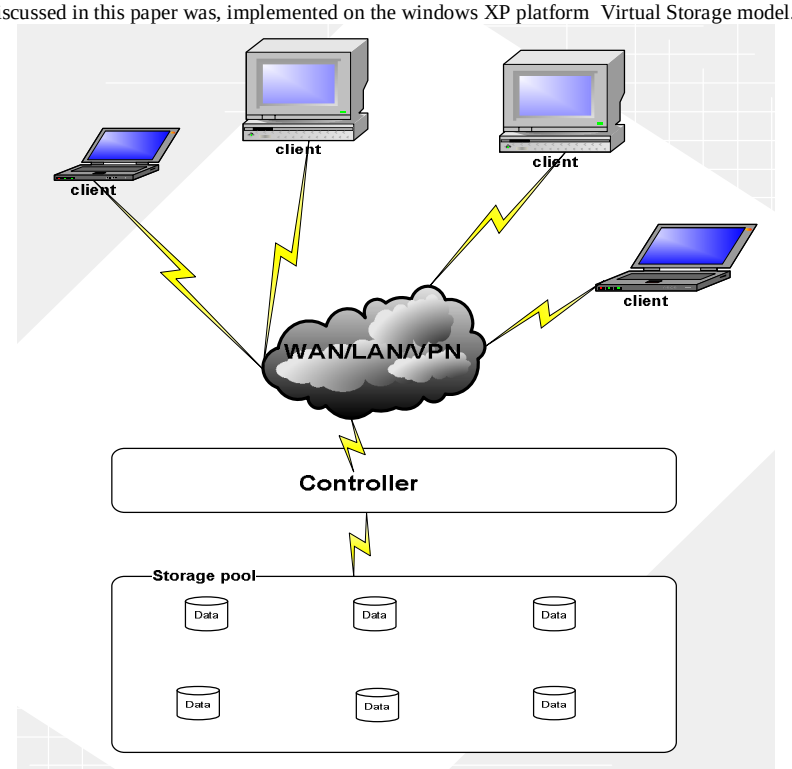


**Fig 1.2: Interaction design**

Fig 1.2 above show the interaction that take place between a clients distributed on a network and the virtual server/controller that eventually provides virtual access to the data/information repository.

**3.2 Traditional network-attached storage Architecture**

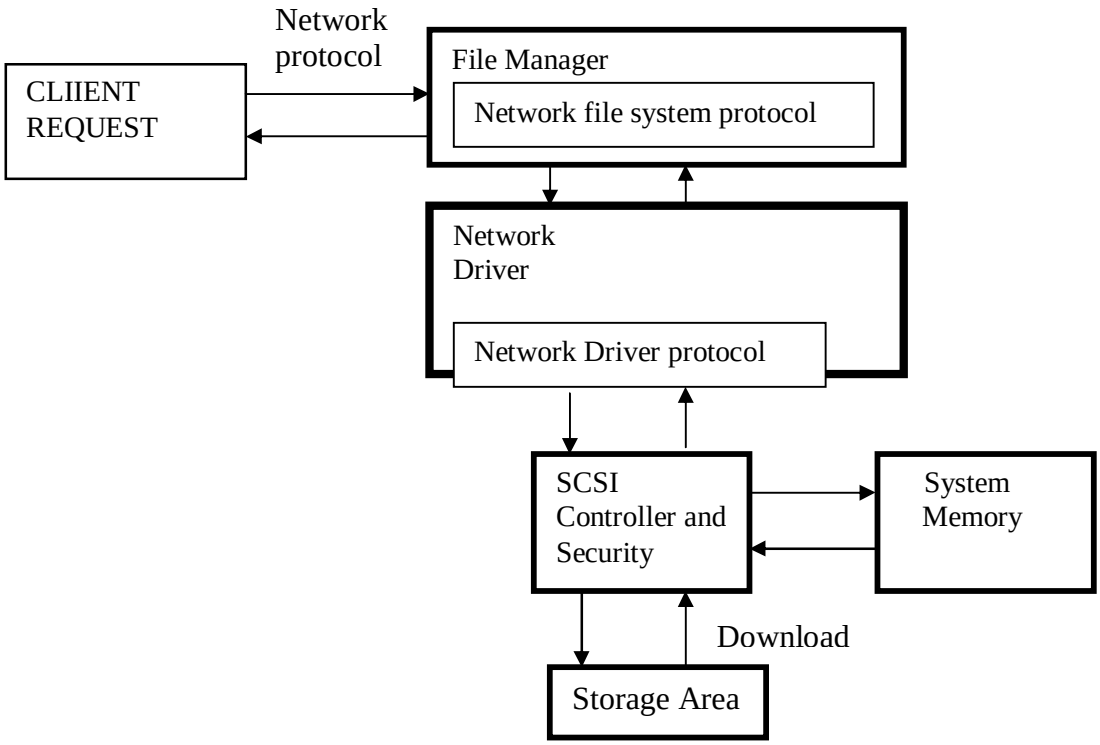


**Fig 1.3: Classical Store and forward process architecture.**

### 3.3 THE SECURE DISKS ARCHITECTURE LAYOUT.

This architecture changes the server's role from being actively involved in every request to a management role of providing high level application-specific semantics to clients. The server is eliminated from the data path and its responsibilities have changes bringing about a new functionality called the file manager. The file manager is responsible of policy definition regarding who can access the storage and other high level functions such as cache consistency and namespace management. A file manager could be the management portion of **any other application such as a database.**

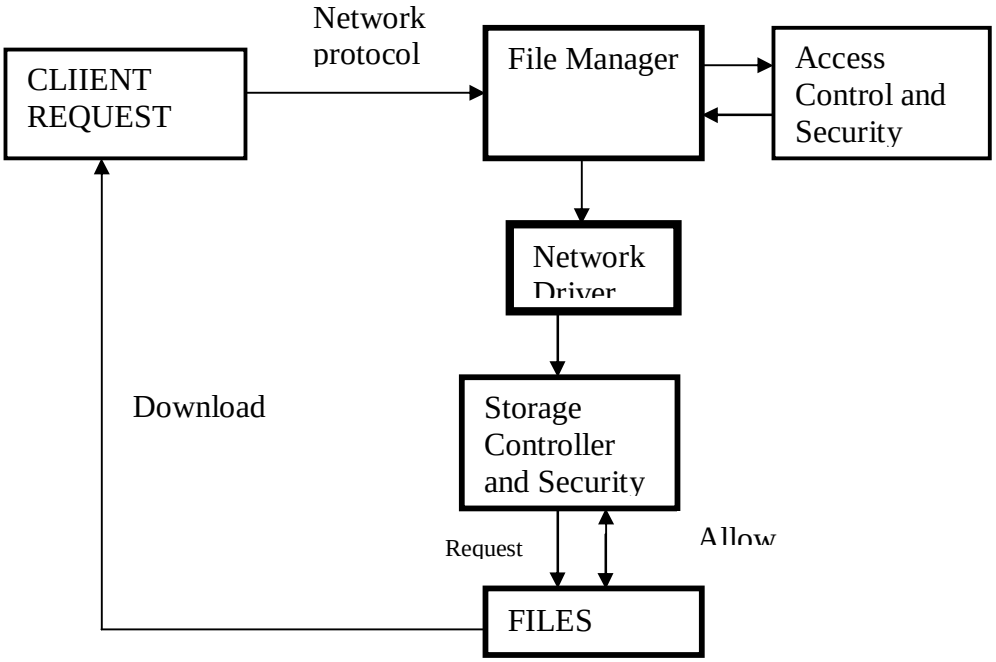


**Fig 2.0 THE VISSM STORAGE FLOW DIAGRAM**

The VISSM architecture discussed in this paper tends to separate management and application level semantics from general data movement operations. The server handles the former while a low-level storage device focuses on the latter.

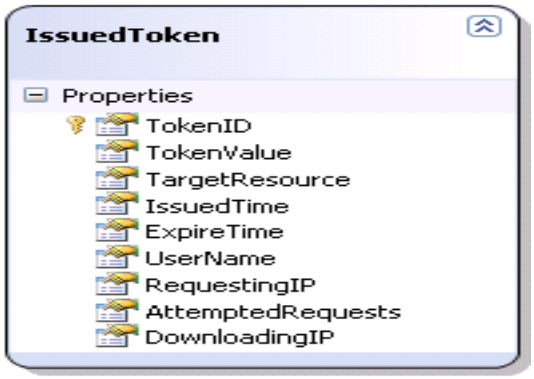


**Fig 2.1: Structure of a VISSM Token object's attributes.**

## 4.0 Security Problem.

In most distributed system environments communication cannot be complete without discussing the issue of security; as a system grows in complexity the security problem also gets more advanced, the security goal in VISSM model is to:

i. Protect the integrity of communication involving network attached storage.
ii. Deliver the scalability and aggregate bandwidth potential of the VISSM architecture.
iii. Optionally, protect the privacy of communication involving network attached storage. (Amiri, 2000)

### 4.1 Security Design

The security architecture discussed in this paper employs cryptographic capabilities issued by the file manager and checked by drives with minimal hardware support.

The separation between issuing and verifying capabilities enables the separation of file storage from file management —they may be done by machines separated by distance and with only indirect communication. Access rights control is managed through the cryptographic information stored in the capabilities.

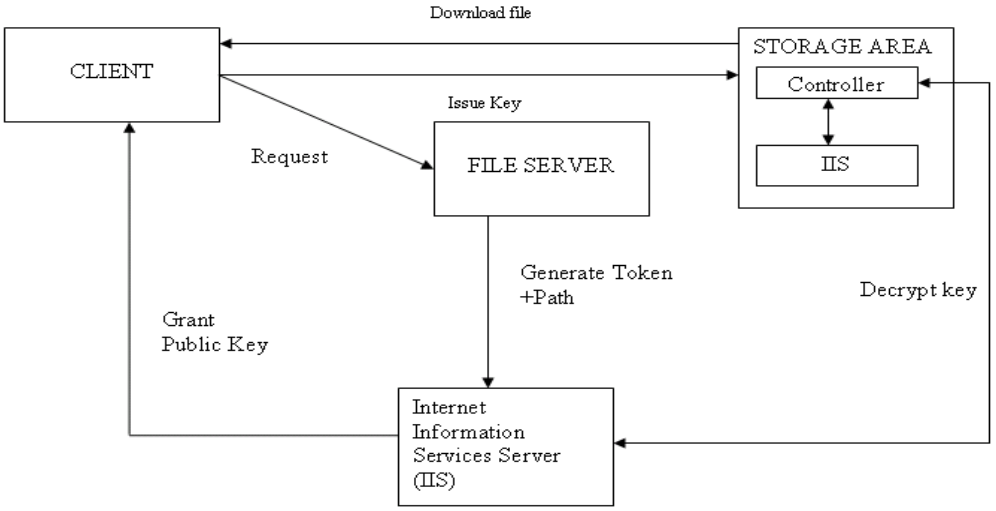


**Fig 2.3 SSL control flow diagram**

Secure sockets layers (SSL) allow encryption of the data and provide a secure communication link between the client and the server this technology is provided by the IIS security implementation framework.; for the VISSM technology to be effective two IIS servers are used, first one residing in the storage area for encrypting the data and the token and the second one residing in the client side for decrypting the token and storage path provided by the File server. Therefore the first IIS server provides public key to the server/controller generated token and path and the former provides a secret key for decryption

*The figure below illustrates a detailed design layout of a virtual secured system*

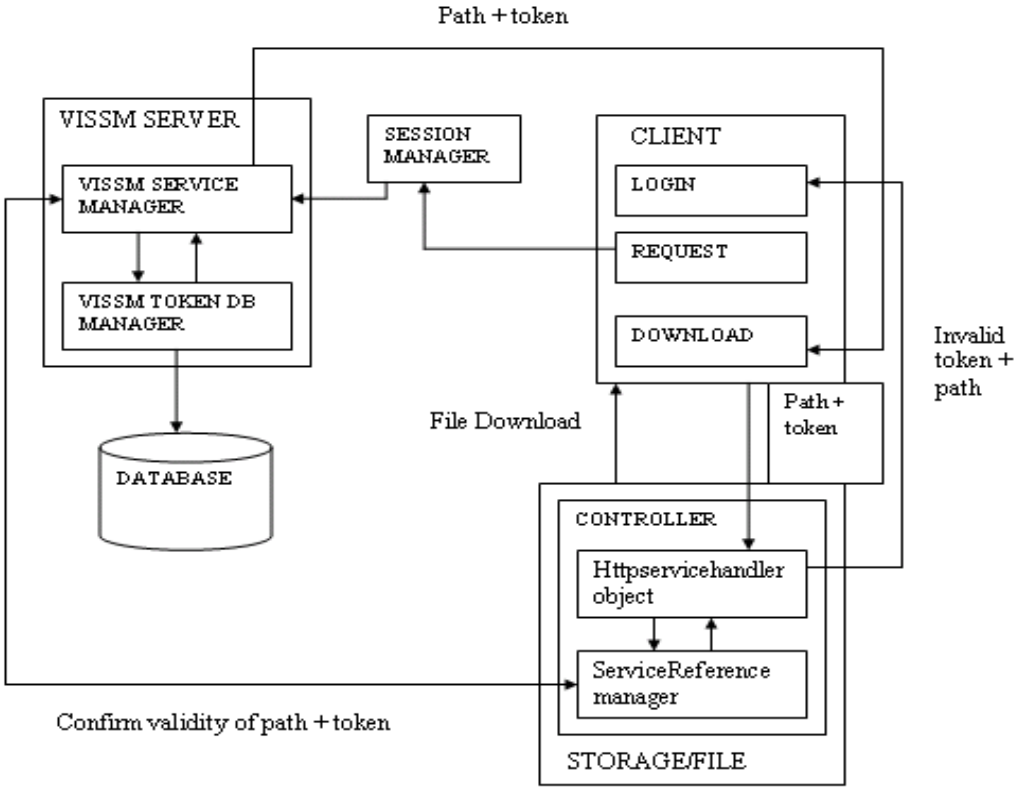


**Fig 2.4 showing a VISSM system**

Fig 2.4 above is a communication between the file server and the client. The user first log into the web server client interface and then will be prompted to select the file to be downloaded from a list of files provided if he successfully logs in. If the file requested for is valid then the request hits the server and the server generates a time-stamped time-limited token appended with the file location storage path, this is handled by the file server section called VISSM SERVICE MANAGER Object. Then on receiving the token the VISSM TOKEN MANAGER object is tasked with the responsibility of updating the database on the date and time the token was issued. Then the client receives the token and the path to the file location, as a download event leading the client to the remote controller in the file storage disks, where the client can download the file as many time as possible as long as it remains valid.
At this point the **Http-service-handler** object is invoked and receives the request capability presented by the client, on receiving this capability (the path and token it forwards) it to a SERVICE REFERENCE MANAGER object which subsequently forwards the issued token and path back to VISSM TOKEN MANAGER object whose sole work is to check the validity of the token and return VALID/INVALID to the SERVICE REFERENCE MANAGER. If the token has expired INVALID is returned and the **Http service handler** Object forces the client to login in afresh, otherwise if the token is marked as VALID then the client will be allowed to DOWNLOAD the file directly without involving the file server.

## 5.0 EXPERIMENTAL RESULTS

The VISSM file server Simulator and client were designed from scratch. The client was designed using java APIs and the VISSM file server simulator was designed using Ruby and runs on a TCP/IP protocol layer. The file server runs on top of a host computer TOSHIBA Tecra model with 1GB Ram and 60 GB hard disk drive. The storage Area was created in the Virtual Machine and then the client was directed to request the data access object defined in the storage pool in the file server the storage pool, it then gets the path to access the real file in the storage area.

The aim of the experiment is to test time it takes a particular node/client to request a file through the server, then several threads of the same file are initiated to discover the response times under increased loads and the following are some of the results capture on the excel spread sheet.
The simulator and system implementation were done separately, the system was implemented on the Toshiba tecra Laptop computer with the same specifications as the simulator, but this time the files were stored in one of the three virtual machines on the network and the File Server was residing in the main (host) computer. The rest of the virtual machines acted as clients- requesting the file(s) through the file server.

**5.1: Simulation Results**

**Table 1.0: SAF processes and VISSM performance.**

| | Experiment to show performance in the disks | | | |
|---|---|---|---|---|
| **No of users(1 file=167 KB)** | **Size of file** | **VISSM Time taken(ms)** | **SAF(s)** | **SAF(ms)** |
| 1 | 167 | 80 | 21.801 | 21801 |
| 5 | 835 | 400 | 88.727 | 88727 |
| 10 | 1670 | 701 | 143.646 | 143646 |
| 15 | 2505 | 840 | 205.996 | 205996 |
| 20 | 3340 | 972 | 259.383 | 259383 |
| 25 | 4175 | 1031 | 340.129 | 340129 |
| 30 | 5010 | 1102 | 382.92 | 382920 |
| 35 | 5845 | 1152 | 430.559 | 430559 |
| 40 | 6680 | 1262 | 475.644 | 475644 |
| 45 | 7515 | 1302 | 529.481 | 529481 |
| 50 | 8350 | 1341 | 574.906 | 574906 |
| 55 | 9185 | 1362 | 628.934 | 628934 |
| 60 | 10020 | 1471 | 682.04 | 682040 |
| 65 | 10855 | 1491 | 738.481 | 738481 |
| 70 | 11690 | 1562 | 784.287 | 784287 |
| 75 | 12525 | 1672 | 830.143 | 830143 |
| 80 | 13360 | 1672 | 879.384 | 879384 |
| 85 | 14195 | 1722 | 949.425 | 949425 |
| 90 | 15030 | 1772 | 1017.923 | 1017923 |
| 95 | 15865 | 1782 | 1065.071 | 1065071 |
| 100 | 16700 | 1793 | 1130.575 | 1130575 |

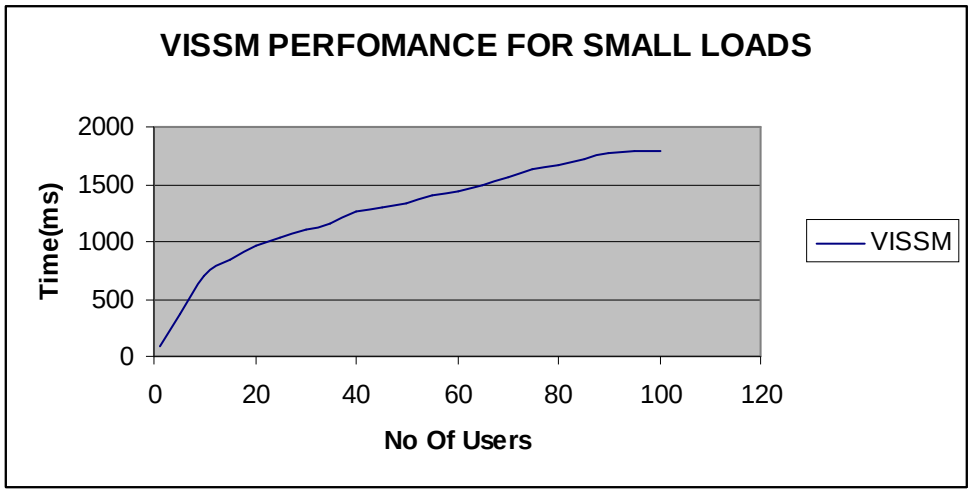


**Fig 2.5: A graph showing the Showing the classical SAF (for small load).**

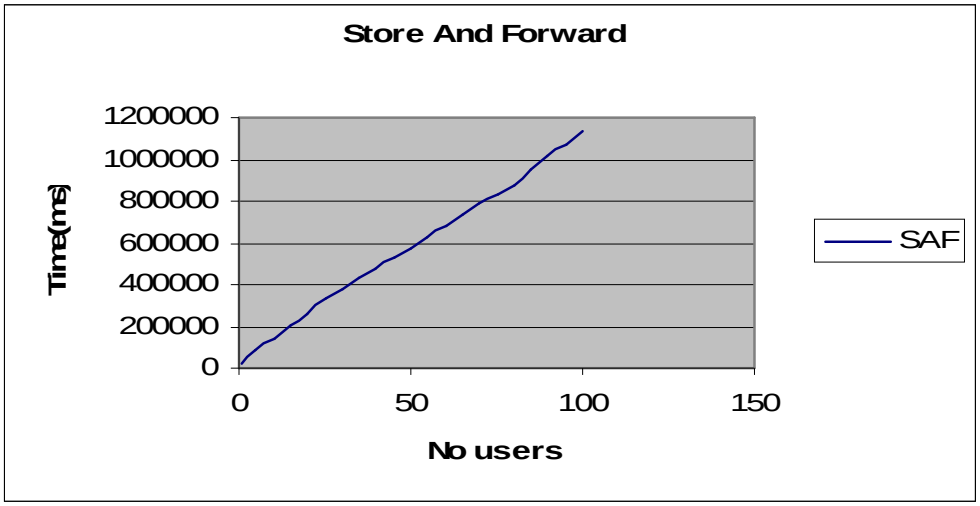


**Fig 2.6: A graph showing the VISSM performance (for smaller load capacities).**

Below shows an Experiment Showing performance of both classical SAF and VISSM under very **large load capacities.**

**Table 1.1: SAF and VISSM under very heavy load capacities**

| **Simulation for Huge file Requests for both SAF and VISSM processes** | | | | |
|---|---|---|---|---|
| **No of Users** | **Size of file** | **VISSM(ms)** | **SAF(s)** | **SAF(ms)** |
| 1000 | 167000 | 2654 | 1398.711 | 1398711 |
| 10000 | 1670000 | 17776 | 1453.42 | 1453420 |
| 20000 | 3340000 | 33959 | 1611.176 | 1611176 |
| 30000 | 5010000 | 50834 | 1806.697 | 1806697 |
| 40000 | 6680000 | 67567 | 2056.577 | 2056577 |
| 50000 | 8350000 | 83720 | 2371.69 | 2371690 |
| 60000 | 10020000 | 99554 | 2793.877 | 2793877 |
| 70000 | 11690000 | 100434 | 3292.854 | 3292854 |
| 80000 | 13360000 | 138499 | 3832.1 | 3832100 |
| 90000 | 15030000 | 160822 | 4400.537 | 4400537 |

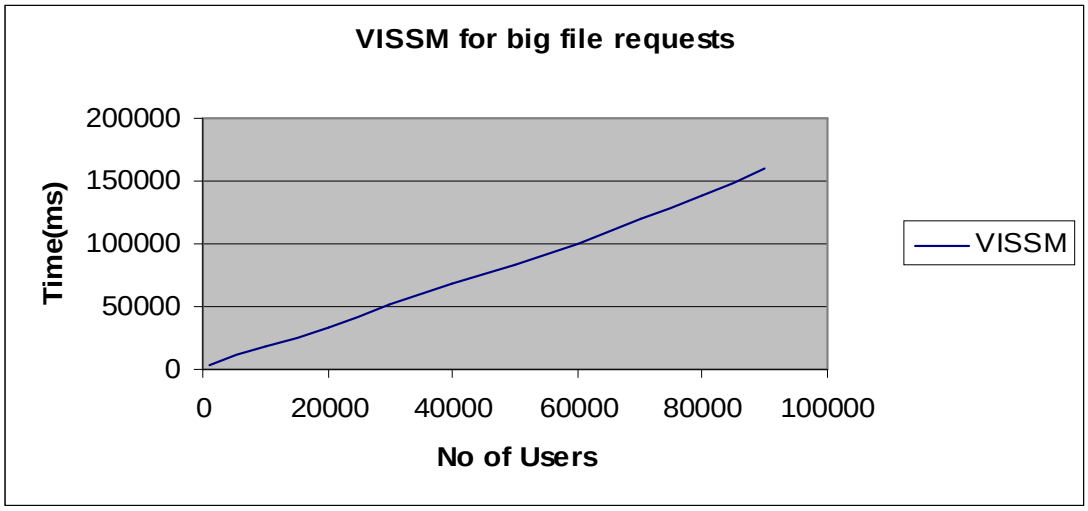


**Fig 2.7: a graph showing VISSM under very heavy load capacity.**

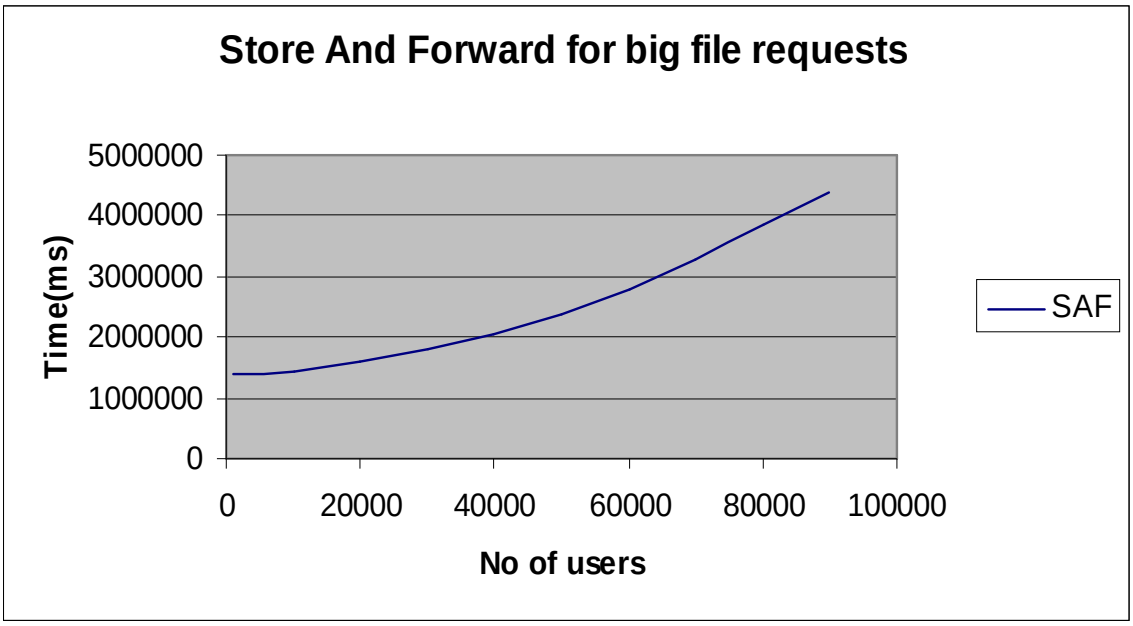


**Fig 2.8 A graph showing classical SAF under very heavy load capacity.**

**5.2 Summary of the Observations**

Comparing Fig 2.5 and 2.6 above at the face value it is evident that under same loads VISSM scales better than SAF. When the system load 20 users VISSM performance is approximately 1000 ms while at the while with the same number of users SAF performs at about 20000 ms which is actually 20 times higher than the store and forward (SAF) process. With 100 users VISSM performs at 1800 ms while at same of users SAF takes 1100000 ms which is 600 times slower than the VISSM. VISSM has a better performance than the SAF model.

It can be noted that when there are 100 users VISSM scales at 60% while SAF scales at 10%.

Fig 2.7 and 2.8 shows that VISSM has a better performance even when there are heavy user requirements. When the system load is 30000 users VISSM takes 50000 ms, at the same load capacity SAF takes 1500000 ms, which is 30 times slower than VISSM. At all these levels of load capacities VISSM performs better

Table 1.1 illustrates the comparison of SAF processes and VISSM on big load capacities; The SAF processes seem to be growing linearly with the increase in load but the VISSM time seem to be remaining constant even if the load capacity increases to large levels.

**6.0 Conclusion**

Virtualization technology, improves the performance and scalability of file server than when the server handles the processes of store and forward. Elimination of the store and forward file servers will greatly improve performance, scalability and reducing unnecessary storage overhead costs.

## 7.0 Future work

VISSM has the capability of measuring the capability of one fileserver but I could propose a further study on multiple fileservers internally integrated to measure the effect on performance with several clients. Although this may turn out to be a challenging study but it might give a new perspective on the study of network attached disks.
I also propose a further extension of the above model to include other technologies, which may include mobile agent technologies, digital cameras and other related digital technologies.
Also since security in distributed systems turns out to be the biggest challenge I could propose a further study of the security sub system of the above model to still come up with a more tamper proof security system.

**References**


1. (Amiri,2000) Scalable and manageable storage systems, CMU-CS-00-178, DEC 2000.
2. (ZablonO, 2009) *Multiple agent co-ordination in a virtualized environment*, A thesis presented at the CSI University of Nairobi.
3. *http://24x7aspnet.blogspot.com/2009/05/aspnet-application-life-cycle-overview.html (last accessed 04/07/2011.)*
4. *http://en.wikipedia.org/wiki/Virtual_appliance(last accessed 11/07/2011 ,16:40)*
5. *http://www.answers.com/topic/software-appliance(last accessed 11/07/2011 ,16:40)*
6. *http://en.wikipedia.org/wiki/Storage_virtualization( last accessed on 12/07/2011)*
7. *http://www.pdl.cs.cmu.edu/ (accessed on 12/07/2011)*
8. http://www.sanbolic.com/data_center_virtualization.htm/*(accessed on 14/07/2011)*
9. http://www.sanbolic.com/virtual_storage.htm,Shared Storage Platform for virtualization and cloud, *(accessed on 14/07/2011)*
10. Erianto Chai, et al, **Case Study on the Recovery of a Virtual Large-Scale Disk**, Dept. of Open Information Systems, Toyo University Graduate School, Japan, Springer-Verlag Berlin Heidelberg 2008