ANALYSIS AND OPTIMIZATION OF DISTRIBUTED FILE SYSTEM PERFORMANCE

Sai Sie thu Kyaw[1], Daw Moh Moh Khaing[2]

Information technology, Technological University, Taunggyi, Myanmar

Abstract

Today, file sharing is widely distributed across networks, administrators face growing problems as they try to keep users connected to the data they need. This paper proposes to implement Distributed File System DFS using network file system, serve as an NFS server it applies services for NFS which is one of the components of File Server in Windows Server 2008, and perform as an NFS client it employs services for NFS of window features in Windows Vista Ultimate. In addition, the communication between server and client uses ad hoc network to share and mount NFS resources. The performance of the NFS is analyzed by enforcing IOzone file system and it is optimized by tweaking transfer block size of mount command options. Finally, the resulted output is automatically generated with chart as an excel file.

Keyword: Ad hoc Network, Distributed File System (DFS), Network File System (NFS), Services for NFS

1.INTRODUCTION

In computing, a distributed file system is a file system that allows access to files from multiple hosts sharing via a computer network. The data can access and process it was stored on the local client machine. Distributed file system makes it possible for multiple users on multiple machines to share files and storage resources. The DFS makes to share information and files among users on a network in a controlled and authorized way [1].The proposed paper uses NFS protocol to share files over the network and to control the access lists on them.

2.AIM AND OBJECTIVES

The aim and objectives of this paper are described as follow:

- To analyze system performance in DFS
- To optimize system performance in DFS
- To understand the nature of DFS
- To understand the concept of NFS

To study benchmark tools

3.BACKGROUND THEORY

According to the way the file storage is managed, there are two distributed file system concepts:
In server client model a set of machines, known as servers a computer or device that manages the network resources provide storage for all of the files in the distributed file system. All other machines, known as clients PCs on which users run applications, must direct their file references to these machines. Servers often run on dedicated machines enabling clients to be more general and often simpler to install and support.
In peer to peer model each machine provides storage on its own attached disk, and allows others to access it remotely. A participating machine may act as both a client and a server [2].The NFS is included in the first concept and it is described in next session.

4.NETWORK FILE SYSTEM

Network File System is a distributed file system, which is developed by Sun Microsystems in 1984. NFS is allowing user on client computer to access files over a computer network like a user local storage. It is based on Sun's RPC version 2 protocol. NFSv2, published in 1985, was a 32-bit implementation of the protocol and used UDP exclusively as its transport mechanism. NFSv3, published in 1994, added TCP to its transport mechanism and was extended to 64-bit files.

The latest version, NFSv4, published in 2000, is well-suited for complex WAN deployment and fire-walled architectures, has a stronger security (public and private key) and improved multi-platform support. It adds persistent, client-side caching and support for Access Control List (ACLs).

4.1. Naming and Location

Naming is a mapping between physical and logical objects. Users work with file names representing logical data objects, while the system deals with physical objects (blocks of data) stored on disk.
Network file system makes no distinction between clients and servers. That is, a workstation may behave as a server, exporting files, and may also behave as a client, requesting file access on another workstation.
Each network file system client sees a Unix file namespace with a private root. The sub-trees that the NFS servers export are bound to this root file system, by using an extension of the Unix mount mechanism. Since the mounted sub-tree may be renamed on the client side, there is no guarantee that the shared namespace is identical at all workstations. Only previously mounted remote directories can be accessed transparently.

4.2. Caching and Replication

NFS clients cache disk blocks in the main memory (I/O buffer cache). Therefore, even if present, local disks are not used for caching. NFS uses a data caching scheme that relies on polling by the client. At the same time as a client caches a file, a timestamp is cached. This timestamp indicates the time when the file was last modified. This is used in validating the data before it is cached, always performed when a file is opened.
When a file is opened, a cache validation check is performed on its parent directory as well. If the cached timestamp coincides with the timestamp on the server, the client pulls the data. If they don't coincide, and the server's timestamp is more recent, the client machine invalidates the cached data and reattaches them on demand.
When data is placed into the cache, it is considered valid for a short length of time. During this time period the client will use the cached data without verifying its modification time with the server.
Directory caching for reading is performed in a similar way as file caching, but modification to them are performed directly on the server. File and directories don't have the same revalidation intervals. The techniques used between the server and the client may be either read-ahead or delayed-write [3].

4.3. Crash Recovery

NFS is a stateless protocol. Hence, the server does not need to keep information about which clients it is serving or which files the clients have open. A benefit of this approach is that there is no need to do state recovery after a server or client has crashed and rebooted, making the NFS crash recovery simple and normally transparent to the user program.
When a server crashes, the client re-sends the requests until a response is received (data will never be lost due to a server crash) and the server does no crash recovery at all. File operation requests are retransmitted many times without getting any response. These operations are idempotent. An idempotent operation has the same effect and returns the same output if executed consecutively [5].NFS client opens a file it will receive file handles from a server.
In this way an NFS server that becomes available after a crash, will be able to recognize which file the modified data belongs to. When a client crashes no recovery is necessary for either the server or the client.

4.4. Security

The mechanism used by NFS when performing access checks is based on the underlying Unix file protection. Each remote procedure call (RPC) request from a client sends the user's identity along with the request. The server assumes the identity and each file access while servicing the request is handled as if the user had logged in directly to the server.
In the earlier versions of NFS, mutual trust was assumed between all participating machines. The client machine determined the user's identity which was accepted, without further validation, by a server. Requests made on behalf of root were treated by the server as if they come from a non-existent user, nobody. Hence, root received the lowest privileges for remote files [2].
With more recent version of NFS higher level of security was introduced. To validate RPC requests, mutual

authentication was used. The common key needed for mutual authentication, which is obtained from information stored in a readable database. In database, stores a pair of keys suitable for public key encryption for every user and server.

One key of the pair is stored in clear, is stored encrypted with the login password of the user. Any two entities registered in the database can deduce a unique data encryption standard (DES) key for mutual authentication.

5.IOZONE

IOzone is a file system benchmark tool. It can generate and measures a variety of file operations. IOzone has been ported to many machines and runs under many operating systems. This section will cover the many different types of operations that are tested as well as all of the command line options. The benchmark tests file I/O performance for the following operations. Read, write, re-read, re-write, read backwards, read strided, fread, fwrite, random read/write, pread/pwrite variants, aio_read, aio_write, mmap.

Although this accelerates the I/O for those few applications it is also likely that the system may not perform well for other applications that were not targeted by the operating system. An example of this type of enhancement is: database. Many operating systems have tested and tuned the file system so it works well with databases.

By using IOzone to get broad file system performance coverage the buyer is much more likely to see any hot or cold spots and pick a platform and operating system that is more well balanced [7].

5.1. Building Iozone

Once obtained the source for IOzone, there have twelve files. They are

1. libasync.c (source code)
2. makefile (makefile)
3. IOzone.c (source code)
4. libbif.c (source code)
5. IOzone_msword_98.doc - documentation in word format
6. gnuplot.dem -sample gnuplot file
7. gnuplotps.dem -sample gnuplot file that generates postscript output
8. IOzone.1 - documentation in nroff format
9. read_telemetry -sample file for read telemetry file
10. write_telemetry - sample file for write telemetry file
11. run_rules.doc - run rules to get reasonable results
12. changes.txt -log of changes to IOzone since its beginning

The make-file will display a list of supported platforms. Pick the one that matches configuration and then type: make target. There is no need to have any install procedures as IOzone creates all of its files in the current working directory

5.1. Definition of Tests

Write: This test measures the performance of writing a new file. When a new file is written not only does the data need to be stored but also the overhead information for keeping track of where the data is located on the storage media. This overhead is called the metadata.

It consists of the directory information, the space allocation and any other data associated with a file that is not part of the data contained in the file. It is normal for the initial write performance to be lower than the performance of rewriting a file due to this overhead information.

Re-write: This test measures the performance of writing a file that already exists. When a file is written that already exists the work required is less as the metadata already exists. It is normal for the rewrite performance to be higher than the performance of writing a new file.

Read: This test measures the performance of reading an existing file.

Re-Read: This test measures the performance of reading a file that was recently read. It is normal for the performance to be higher as the operating system generally maintains a cache of the data for files that were recently read. This cache can be used to satisfy reads and improves the performance [7].

6.SYSTEM DESIGN

In this system, NFS client analyzes the file system performance using IOzone benchmark tool. The write test measured the performance of writing a new file in

distributed files system space. The file sizes tested were: 1 MB, 4 MB, 16 MB, 256 MB, 512 MB and 1 GB. File size 2GB is tested to get more accurate in NFS optimization. Each file was created using a record size (the amount of data written into a file during a single IO operation) of 4KB. For each file size the standard test was repeated every 5 minutes several times by using a batch script. The test's results were directed to a log file which later on was parsed by Report Generation to create the Excel file with chart.

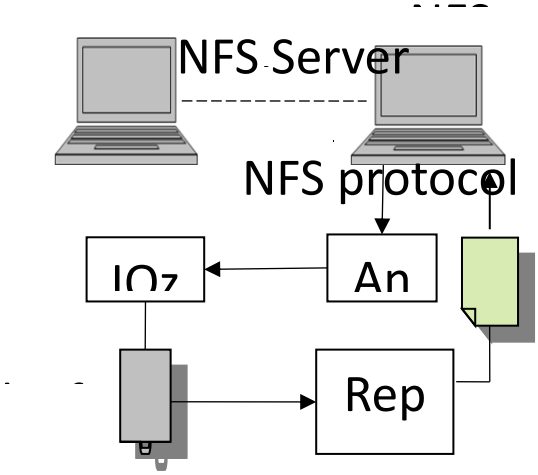

Figure 1 System design

To get the accurate result of NFS performance, need to measure repeatedly in several time with Iozone. So, use a batch script file of each file size with the following command:

```
set count=0
:while
if %count% == 7 goto end
Z:\iozone\iozone -Rac -s 1G -r 4k -i 0 -i 1 -U Z:/ >> logfile.txt
pause 5
set /a count = %count%+1
goto while
:end
```

System design is shown in Figure 1.

6.1. Optimization of System

To optimize the NFS Performance we use the best transfer size 32 KB of NFS Version 3 in both rsize and wsize where we mount. Use TCP+UDP protocol to overcome the disadvantage of TCP protocol.

6.2. Implementation Results

To implement this system,

- In Servers, we use Window Server ®2008.
- In Client, we use Window 7.

Report generator is as shown in Figure 2.

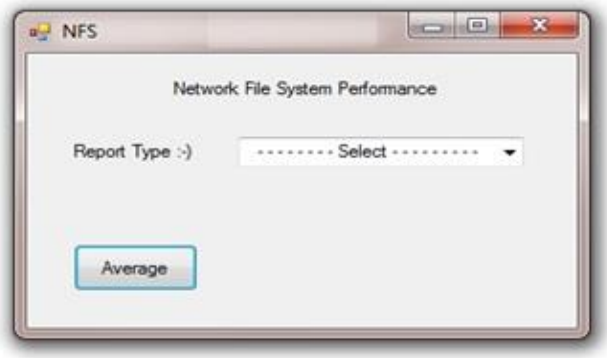

Figure 2 Report generator

The average performance of the proposed system test with record size 4k bytes is shown in Figure 3.

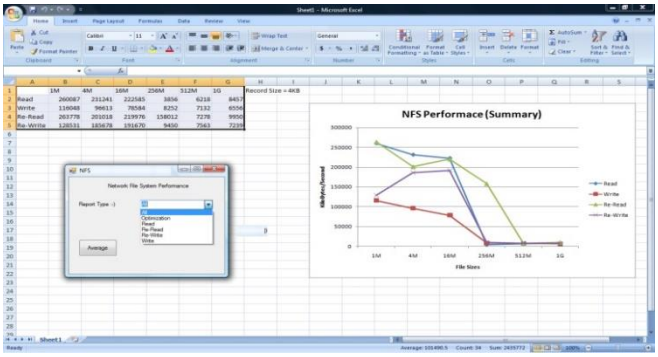

Figure 3 Network file system average performance

When the user select read and click average, the results are shown in Figure 4.

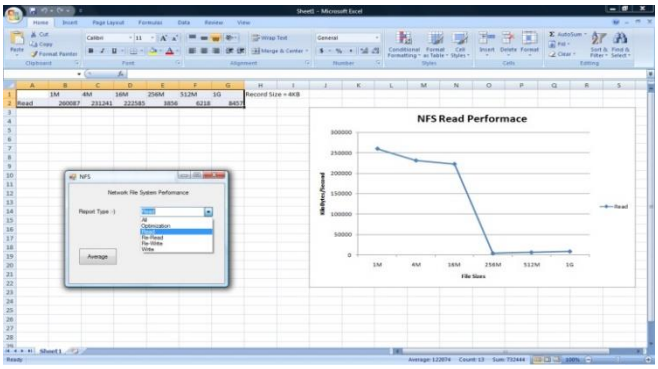

Figure 4 Network file system average read performance

The graph shows the best performance in file size 1M byte when the user select write and click average as shown in Figure 5.

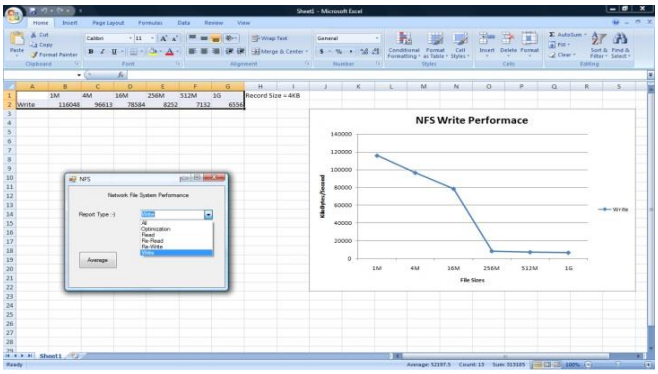

Figure 5 Network file system average writes performance

The graph decreases with increased file size 4M bytes and fall down to 256M bytes due to cache exceeded as shown in Figure 6.

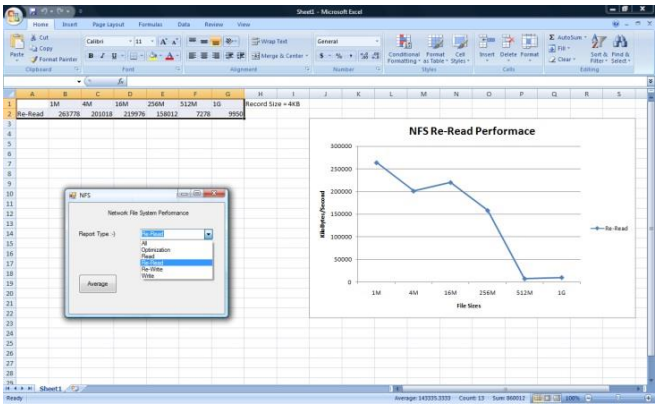

Figure 6 Network file system average re-read performance

Re-Write performance in 16M file sizes is shown in Figure 7.

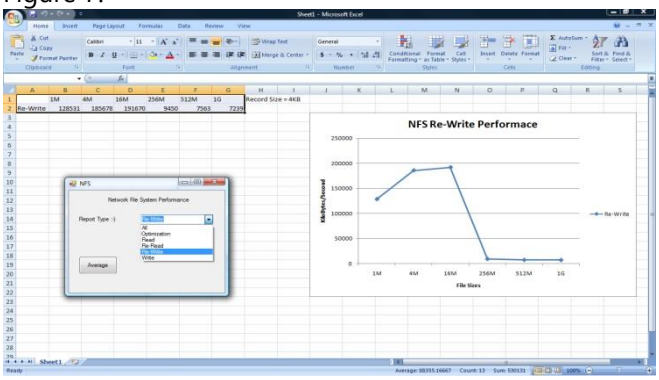

Figure 7 Network file system averages re-write performance

When the user select optimization and click average, the results are shown in Figure 8.

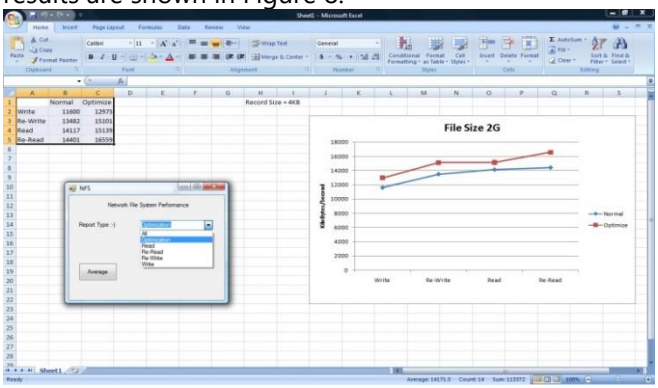

Figure 8 Optimization of system performance

7. CONCLUSION

The distributed file systems are widely used in supercomputers, clusters and data centers. Network file system provides can access the remote data. In this paper discuss about how to work network file system average performance, read performance, writes performance, re-read performance, re-write performance and optimization of system performance.

8.ACKNOWLEDGEMENT

The author wishes to express his truthfully thanks to Daw Moh Moh Khaing, Lecture , department of Information Technology , Technological University (Taunggyi), for her invaluable attitude, suggestions and encouragement for the completion of this paper. Finally, the author would like to acknowledge the very considerable contribution of his parents and all teachers from Information Technology department, Technological University Taunggyi and all those who gave up most effort on his way of living and to all of his friends and sisters, very helpful persons for his progression.

REFERENCES

[1] http://en.wikipedia.org/wiki/Distributed_file_system
[2] C. Eriksen, "Comparison of NFS, Samba, OpenAFS", OSLO University College, June 2005.
[3] Inc. 1998 Sun Microsystems, Network programming (Ascend McNair Conference), the Graduate Center of the City University of New York, 2001
[4] E. Zadok, Linux NFS and Automounter Administration: SYBEX, 2001.
[5] M. Burgess, "Analytical Network and System Administration", WILEY, 2004.
[6] William D. Norcott ,"Iozone Filesystem Benchmark".
[7] http://www.iozone.org