Tuning and Optimizing Network File System Server Performance

Ghania Al Sadi
Sohar University,
General Foundation Program - Computing Program
Sohar, University Rd, 311
Sultanate of Oman

ABSTRACT
Network file system is one of distributed file systems that are used over network to provide remotely access to data on the servers. The last version of NFS has a number of features that help in increasing server performance. However, the provided features are limited to a specific type of operations on the server. Therefore, finding more options may work alongside to enhance server performances. This research is discussing the provided features in NFSv4 and activates one of the tuning options to tests NFS server performance.

General Terms
Network File System, NFS Features, Server Performance

Keywords
NFS, RPC, Performance, Referral, Replication, Delegation, Block Size

1. INTRODUCTION
Network file system (NFS) is a distributed file system protocol that was developed by Sun Microsystem Company in 1984. NFS enables users to access data remotely over network as it is stored locally on their devices. NFS uses Remote Procedure Call (RPC) technique as its base to execute requests over network and manage the communication between computers. It minimizes the need of high storage in client devices by centralizing data on servers. NFS is built into UNIX Kernel to be implemented on UNIX operating systems like Linux. However, it can be implemented on different operating systems to share data. Many organizations that use Linux servers or mix of operating systems prefer to use NFS as file system to manage data among computers over network. Sun Microsystem has developed different versions of NFS (Traeger et al. 2007). Earlier versions of NFS were mostly implemented on LAN networks to process users' requests over TCP/IP or UDP/IP protocols. NFSv2 and NFSv3 are stateless where each NFS service is assigned a port dynamically by the portmaper. Actually, these ports are not protected by firewall filters like iptables where users are not authenticated during remotely access to the server. Therefore, each NFS service needs a static port numbers to be configured by network administrators for use over firewalls that make it difficult to be used in WANs (Chen et al. 2014). However, last version of NFS designed to be used over the Internet that includes stateful protocol to provide high performance and security (Yangli et al. 2011).

NFS is considered as a transparent protocol because it enables accessing and viewing data on clients' computers through the network as they are stored locally on client's device disks. However, data are actually stored on the server with invisible structure to client (Chao et al. 2008). Moreover, NFS supports virtualization by implementing Virtual File system (VFS) that allow clients to access data transparently on different type of devices regardless of file system type used on those devices. Also, VFS delivers the required options to manage and handle various requests of files on the server simultaneously.

2. NFS SECURITY
Security is the main concern in networks utilizing NFS while it is used to share data over internet or even private networks. Security issue mostly faced by earlier versions that eliminate the use of NFS over the internet because NFS make use of RPC to transmit data over network and RPC is considered as insecure that can be only used over trusted network with firewall. In earlier versions, it was difficult to use firewall with NFS because NFS servers use multiple ports for each service while firewall can control network traffic using one defined port. This makes it difficult for clients who try to access the NFS server where they need to find the mounted server's port by contacting the server's portmapper. However, NFSv4 make use of a single network port that is well-known by the firewall to enable filtering traffic on the network. Using one port eliminate the client from contacting the portmapper to locate the mounted server. Moreover, NFSv4 enables using some cryptographic mechanisms to encrypt RPC. In Linux, NFS use Kerberos5 and SPKM-3 as security mechanism to provide authentication, privacy and integrity of data while transmitted via network (Traeger et al. 2007).

3. NFS ARCHITECTURE
The main components of NFS are NFS server and client where both can share and exchange data remotely using Remote Procedure Call (RPC). Setting NFS in a network system needs to install NFS packages on both client and server regardless of the operating system type. NFS client and server deal with TCP/IP to send and update files. Earlier versions of NFS used both TCP and UDP to transfer data where UDP mostly used with applications because it is considered faster than TCP in application side. However, UDP is unreliable protocol since it cannot guarantee that the data is delivered to its intended destination. Moreover, UDP has no control over network traffic and data flow. This issue is overcome by using TCP protocol in the last version of NFS. NFSv4 makes use of TCP/IP to transport data for more reliability where data transmission is managed and controlled in Transport Layer (Yangli et al. 2011).

Exchanging data between server and clients requires exporting shared directories from server where the clients mount data to the mounted server. Client requests are transmitted by RPC to the intended destination on the network. Data transmitted using TCP/IP, UDP/IP and Microsoft protocols that are used by both Client Redirection and Server. Figure 1 illustrates NFS architecture that is setup on UNIX systems.

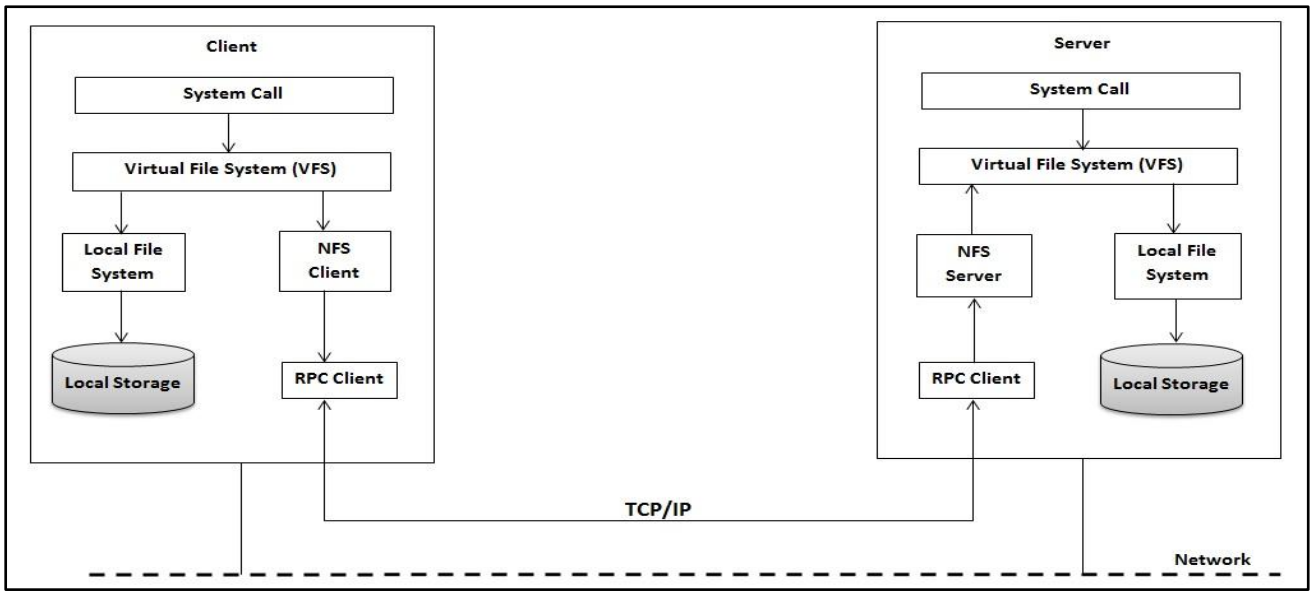

Figure 1: NFS Architecture

As mentioned before, NFS supports virtualization by implementing Virtual File system (VFS) that is setup between the file system and the client processes. Once a node request files from another node (server or client) in the network, VFS passes the requests that can be (open, access, create read files. etc.) to NFS. These requests are converted into NFS procedures that are stored in NFS RPC to decide how to deal within the NFS protocol. The obtained procedure then will be executed inside the RPC layer. RPC organizes requests and its arguments and transfer them to the intended node or destination. In destination side, RPC controls the responds and delivers it to the requester. RPC is also, responsible to ensure that all NFS users are speaking the same language to be able to understand the exchanged file's data type between users' machines using External Data Representation (XDR) layer. XDR is responsible for converting data format during exchanged over network by different type of nodes to be understood by each other. Data is converted into XDR format prior to transmitting to the intended node. Using XDR eliminates convention or interpretation issues that may be faced over network during exchanging data among heterogeneous network computers (Yangli et al. 2011).

4. NFS ARCHITECTURE

The last version of NFS provides a number of optional supported features that increase NFS performances by reducing network traffic. Most used features are delegation, referral and replication that are granted by NFS server to reduce the workload on the server and enhance fault-tolerance in case of server failures. However, these features may be beneficial for specific operations with some considerations.

4.1 Referral

Referral is a global namespace feature that provides easy distribution of data among multiple servers in the network. NFS use referral object that is created in the server namespace to state the location of the attached information. Actually, a referral server does not contain the file system requested by the client but it contains references to the requested data that is exported on another server. Therefore, it automatically redirects and passes the client's request to the server that contains the requested files system in a transparent way that appears as it is provided by the primary server (Curylo, Joltes, Nayar, Oesterlin 2005). NFS server using referral must enable replication feature to maintain information location. Figure 2 illustrate the process of redirecting client's request using referrals.

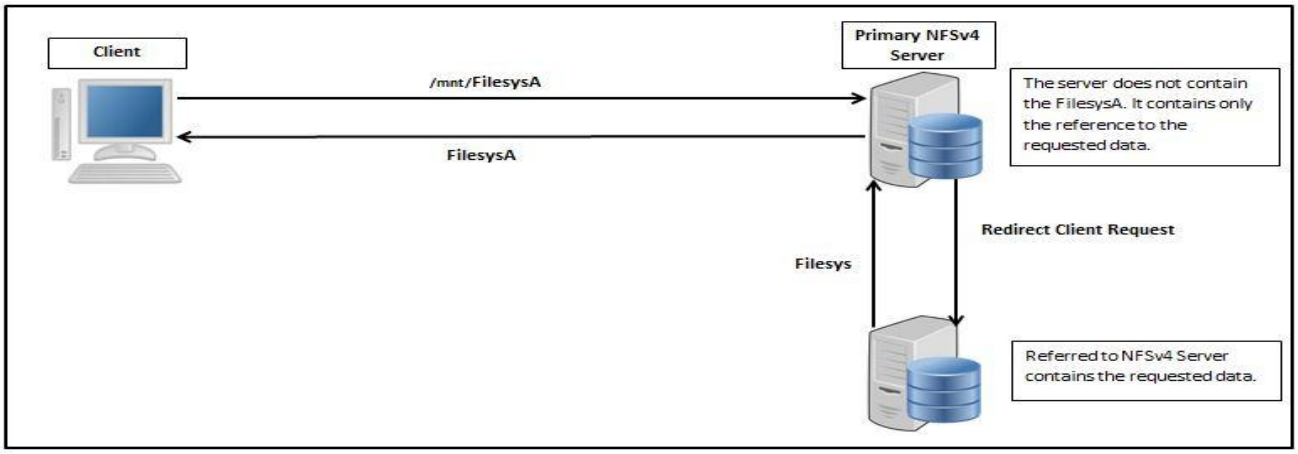

Figure 2: Referral Feature enabled on NFSv4 Server

4.2 Replication

NFS supports replication features in the servers that specify where copies of data can be available in case of server failures. Replication provides a replica of data on multiple servers to improve the availability. In case of server failure, user's requests and application are transparently switched to a working server that contains a replica of the requested data. Actually, replication improves the availability of data and enables load balancing by distributing loads among multiple servers and thus referring clients to these servers when the main server is unavailable. The server can export a file system and specify the location of the replica of the file system. While exporting the file system, the server must ensure exporting the root of the replicated file system. Also, the location of the replica must be the root of the file system and the location is the attribute of the file system, not a directory or file attribute. In this case, all mounted directories to a replicated directory must be also replicated. Therefore, replicas will be copies of the file system that enable the client to access same file from replica locations when the primary server is inaccessible. Moreover, replication must ensure the synchronization of data copied to replicas where some methods must be provided for the primary file system to ensure the consistency of data in the replica. Therefore, replication is mostly provided for read-only data to avoid inconsistency of data. However, replication depend on a number of factors like the number of clients accessing data, the number of available servers, the required redundancy level and the requirements of load balancing (Curylo, Joltes, Nayar, Oesterlin 2005).

4.3 Delegation

NFSv4 support delegation feature as an optional mechanism to reduce network traffic caused by the increase number of workload. As illustrated in Figure 3, Delegation caches requested data and attributes locally in the client device at the first server call to reduce the interaction between the server and client. Actually, delegation does not eliminate network traffic completely since the client needs to contact the server at the first activity call like open, edit or lock that will be cached by delegation on client device. Then, all later file activities like open, edit or lock can be done on client devices without the need to call the server again. Using cached files on client device, eliminate the client from finding any file's changes on the server. However, delegation can be cancelled when another client request for performing edit activity on the same file. Delegation is more beneficial with read-only files where the client is allowed to read data only. The limitation using delegation is that the recent changes of the file are not provided for the client in the cached copy (Chen et al. 2014).

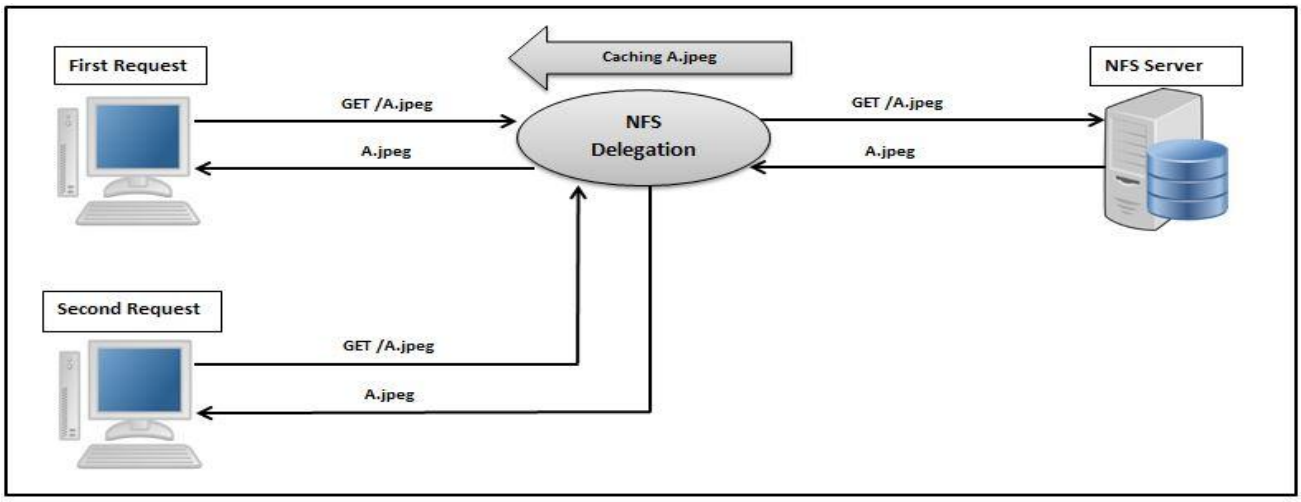

Figure 3: NFS Delegation

Actually, client does not request a delegation but the server decide whether to grant the delegation for client depending on access patterns for a requested file. Usually, client is granted a delegation in respond to file opens. Afterward, NFS server has the right to recall the delegation from client at any time using back-channel connections. Moreover, NFS server can grant read delegation for multiple users simultaneously but write delegation can be granted to one client at a time to avoid conflict of write operations on the same file. Therefore, server may not grant read delegation for any client while the file is edited by any client currently granted write delegation to avoid any potential conflict. Generally, conflict may occur when the file is accessed inconsistently with a current granted delegation. For instance, if the file is currently accessed by a write delegation and another operation occur on the file like read access, the server will recall the first write delegation where the second access is not granted a delegation. In case of conflict occurrence, the server activates callback mechanism to contact the client currently holding the delegation to send the update of the file and returns the delegation. In case of respond failure, the server revokes the delegation and report the failure as I/O errors to the application. These types of errors are recovered by closing and reopening the file (Curylo, Joltes, Nayar, Oesterlin 2005)

5. INCREASING NFS SERVER PERFORMANCES

As discussed previously in this research, NFS features may improve server performance, however these performances are limited and restricted for special type of operations. For example, delegation is more beneficial for read operations since data will be cached on client device. Server can grant read delegation for a number of clients simultaneously but only one client can be granted write delegation at a time to avoid conflict with other write operation on the same file. Therefore, workloads can be reduced only during read operations. Also, considering replication, Generally, NFS server performance based on the capability of the server to

process request of clients quickly and utilizing less CPU processing while read and write process. High performance obtained by reducing read and writes latency and reducing kernel CPU processing during read/writes. For high performance, NFS server and client need to be tuned using a number of options like changing block size setting, change MTU size, change the time the client must take to retry same request from the server, and changing other options available with NFS (Lu et al. 2009).

Increasing block size will reduce the number of IP packets exchanged in the network. On the other hand, changing Network MTU Size may be beneficial to improve server performance. Actually, MTU is the maximum transmission time unit that refers to the maximum volume of passed data in a single Ethernet. It can be changed in both client and server to meet read/write data size transmission without the division to portions because increasing the number of portion will increase the number of IP Packets to be transmitted and thus increase the work load across the server. Moreover, changing the number of client attempts can help to increase server performances by using timeo and retrans options. These options are used to set the time client use to make the next retry of the same request in case of server delay and to set the number of attempts (Seoane et al. 2012). Along with these options, increasing the number of NFS threads is considered as a simple factor of server performance where this option is done in the server to allow a high number of clients to access the server. In this research, changing block size will be used as an experiment to test NFS performances on Linux environment as discussed in next section.

5.1 Change Block Size

Data block size is one factor of increasing server performances because it specifies the amount of data that will pass between server and client. Most versions of NFS has a default value for block sizes, however the value can be changed depending on the available requirements. NFS uses RPC to exchange data over network, therefore increasing and decreasing the size of read and write in RPC packets will affect the number of IP packets that are transferred over the network (Yangli et al. 2011). Decreasing the size of read and write will increase the total number of IP packets that need to be sent over network. Technically, decreasing the total number of IP packets that are exchanged over network is more beneficial to increase network performance where the network traffic will be reduced. Therefore, increasing the number of read and write size in RPC packets will reduce the total number of IP packets transferred over network and thus increasing NFS performances (Schmuck and Haskin 2002). For example, if the 1 MB of data is divided into 32 KB of equal chunks, then the number of the transferred IP packets will be increased because 32 chunks will be transferred over network. On the other hand, if 1 MB of data is divided into 64 KB of equal chunks, then the number of IP packets will be decreased where only 16 chunks will be transferred over network that will reduce the traffic over network and the thus increase the speed over network. This example is illustrated in the below Figure 4.

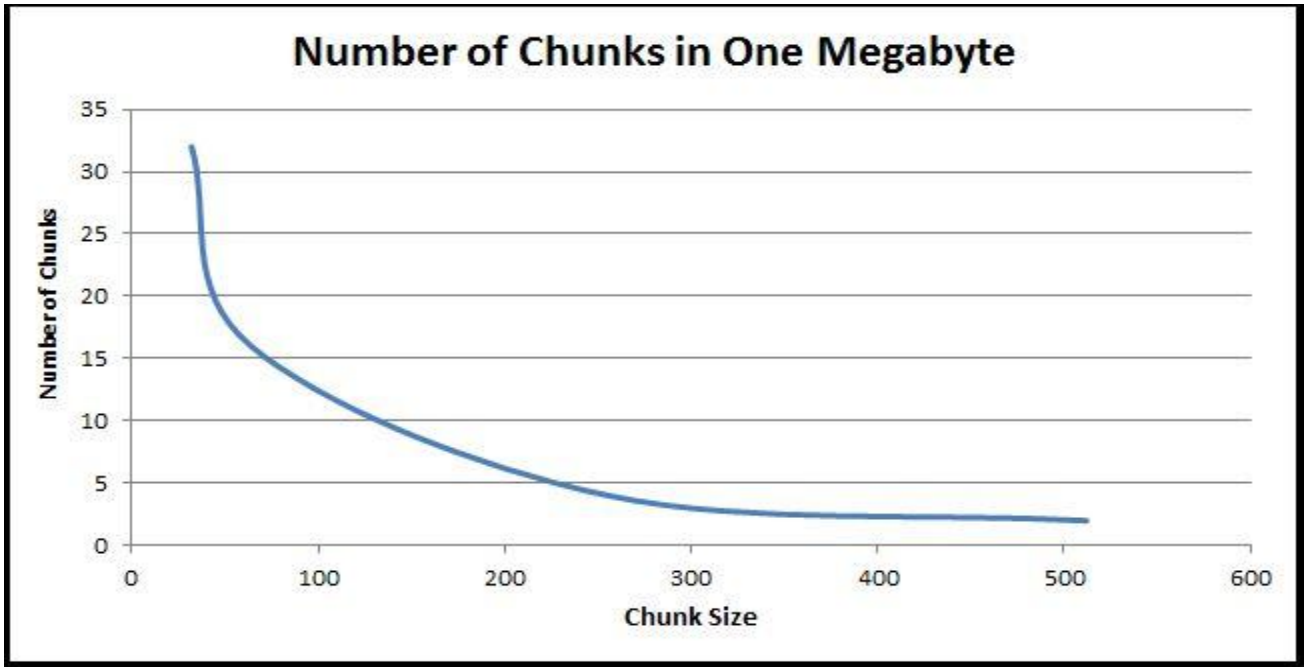

Figure 4: Increasing chunk size to decrease the number of IP Packet

Therefore, it is more reasonable for NFS to change the default value of read and write size of the NFS share mounted on client devices. As shown in Figure 5 below, the default options of data chunk size that are transferred by RPC packets during read and write is 131072 that are represented in by rsize and wsize respectively. These values are set by default in NFSv4 that may differ in other NFS versions.

```
192.168.1.100:/home/nfsassig /mnt/NFS nfs4 rw,relatime,vers=4,rsize=131072,wsize
=131072,namlen=255,hard,proto=tcp,port=0,timeo=600,retrans=2,sec=sys,clientaddr=
192.168.1.101,minorversion=0,local_lock=none,addr=192.168.1.100 0 0
```

Figure 5: The default value of rsize and wsize in RPC packets

The default values of rsize and wsize can be tuned while mounting to meet NFS network requirement by considering the network capacity and power processing and performance of both the client and the server. As shown, in Figure 6, the rsize and wsize values are increased to 200000 that will reduce the number of data chunk need to be sent by RPC packets and thus decrease the total number of IP packets that will be transferred over network.

Figure 6: Tuning rsize and wsize to reduce the total number of IP packets

However, modifying and tuning read and write size parameters must depend on the capability of the network and its components like ports of servers, clients and switches. For instance, if the server and client ports are one gigabyte and as well for network switches, then tuning these parameters to a high value is recommended to increase network performance (Wang and Hsu 2010).

6. CONCLUSION

Network File System is considered as a suitable protocol that can be used over networks to access data remotely. It provides a powerful mechanism to access data using RPC. Also, NFS provides optional features like Delegation, Referral and Replication to improve NFS server performance. However, the provided features still have some limitations since they may work with a specific type of operations only like read-only requests. Therefore, tuning NFS server using some optimization mechanisms is required to provide more performances on the server. Increasing block size of data that are mounted over NFS server can work powerfully because the number of chunks in the RPC packets will be decreased and thus the total number of IP packets will be reduces. Technically, reducing the total number of IP packets transferred over network will reduce network traffic and thus increase the network respond speed. Testing NFS server using different parameters like tuning block size, modifying MTU size may be the aim for future work. A number of tools are available to test NFS performance and speed that may be used to compare the available optimization parameters and mechanisms.

7. REFERENCES

[1] Chao H-C, Liu T-J, Chen K-H, Dow C-R. A seamless and reliable distributed network file system utilizing webspace. Web Site Evol 2008 WSE 2008 10th Int Symp. 2008;65–8.

[2] Chen M, Hildebrand D, Kuenning G, Shankaranarayana S, Tarasov V, Vasudevan AO, et al. Linux NFSv4 . 1 Performance Under a Microscope. 2014;

[3] Curylo, Joltes, Nayar, Oesterlin P. Front cover Implementing NFSv4 in the Enterprise : IBM Corp. 2005;

[4] Lu J, Du B, Zhu Y, Ren L, Li D. A High Performance Cluster File System with Standard Network File System Interface. 2009 Int Forum Inf Technol Appl [Internet]. 2009;397–400. Available from: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5231636

[5] Schmuck F, Haskin R. GPFS: A Shared-Disk File System for Large Computing Clusters. Proc First USENIX Conf File Storage Technol [Internet]. 2002;(January):231–44. Available from: http://portal.acm.org/citation.cfm?id=1083349

[6] Seoane N, Valin R, Garcia-Loureiro A, Pena TF, Zablah I. Performance of numerical simulations on the cloud. 2012;19–21.

[7] Traeger A, Thangavelu K, Zadok E. Round-Trip Privacy with NFSv4 ∗ . Security. 2007;1–6.

[8] Wang C-C, Hsu Y. Wofs: A Distributed Network File System Supporting Fast Data Insertion and Truncation. 2010 Int Work Storage Netw Archit Parallel I/Os [Internet]. 2010;43–50. Available from: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5571751

[9] Yangli, Li Y, Zheng L. Transparent encryption based on network file system filtering driver. 2011 Int Conf Electr Inf Control Eng ICEICE 2011 - Proc. 2011;6339–42.