IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 16, Issue 5, Ver. VIII (Sep – Oct. 2014), PP 31-39
www.iosrjournals.org

Towards Adapting NAS Mechanism over Solid State Disks

Shaikh Muhammad Allayear[1], Jannatul Ferdaus[2], Md. Salahuddin[3] and Sung Soon Park[4]

Department of Computer Science and Engineering
East West University, Bangladesh[1,2,3] and Anyang University, Korea[4]

Abstract: *Flash memory storage such as SSDs (Solid State disks) have been gaining popularity due to its low energy consumption and durability in embedded systems and laptops. With the fast technical improvement, Solid State Disks (SSDs) are becoming an important part of the computer storage hierarchy to significantly improve performance and energy efficiency. However, due to its relatively high price and low capacity, a major system research issue to address is on how to make SSDs play their most effective roles as a high-performance storage system in cost- and performance-effective ways. In this paper, we describe a NAS prototype that eliminates most software overheads and makes Solid State Disks a high-performance storage system in cost- and performance-effective ways. NAS integrates a network adapter into SSDs, so the SSDs remove the random access time, Reduce administrative overhead while delivering scalable, reliable data storage. NAS can also give support block based I/O rather than traditional NAS system which supports only file I/O protocol such as NFS and CIFS. Our scale-out NAS solutions can boost capacity and performance while enabling flexible provisioning, no disruptive operations, easy scalability, and virtualization. As the storage and processing requirements of the file server continued to increase data security and integrity became difficult to manage in conventional NAS, this NAS can ensure the data integrity and sanctity as well.*

Keywords: *Data Storage & Data Integrity, Iscsi, NAS, Solid State Disks, Software Overhead*

I. Introduction

In the past, floppy disks with capacities in mere KB's were widely used to share data files. Over time the need for larger and larger capacity has emerged due to growing need for data to be shared across organizations. Removable storage media, such as flash disks, are capable of storing gigabytes (GB) of data have now complimented the traditional removable media disks. The emergence of non-volatile, solid-state memories (e.g., NAND flash, phase change memory, and spin-torque MRAM) changes this landscape completely by dramatically improving storage media performance. As a result, software shifts from being the least important component in overall storage system performance to being a critical bottleneck. Businesses not only need the capacity to handle huge data storage requirements, the need to share their data has made Network Attached Storage (NAS) an attractive option. NAS systems use external storage for server/hosts. Traditional NAS works at the file level, block-level storage Applications are not available on NAS. To this day, they continue to have limited network, CPU, memory, and disk I/O resources due to their single server design, which binds one network endpoint to all files in a file system.

This paper describes a new NAS architecture. Our NAS re-examines the hardware and software structure of storage systems to minimize software overheads and let the Performance of the underlying storage technology shine through. Here NAS emerged iSCSI storage with the traditional NAS. iSCSI is an Internet Protocol-based storage standard for linking data storage facilities, and presents storage to servers as disk targets[1], which, from the perspective of the application, appear to be storage attached locally to the server. NAS uses iSCSI which is designed to allow concurrent access from multiple servers. And because iSCSI presents storage space as virtual block-level devices, operating systems and applications have the ability to put their own file systems on them, which is something not possible with conventional NAS. It can support One or more disk (and possibly tape) disks that can be attached to the NAS systems to increase total capacity. With this NAS solution, we can quickly and simply store and share our music, videos, images and other files in one convenient location. What's more, we can also stream digital media to any DLNA™ CERTIFIED multimedia device located anywhere in your home – wirelessly. But it is still difficult to store multimedia data such as mpg, mp3, etc and install large software such as database engines [2][3][4][5]. To alleviate these problems and access mass storage we developed NAS, an iSCSI based NAS Cluster system, for providing the allocation of a mass storage space to each client through networks. The system offers to its users the possibility of keeping large size of data and database in a secure and safe space. It also improves NAS performance providing a very low-latency and integrates it directly into an SSD. Also it can give data protection or backup without having congestion problem in a scalable manner.

II. Motivation

Fast non-volatile memories such as SSDs are already influencing the design of storage devices in the industry. As faster memories become available the cost of software and hardware overheads on storage accesses will become more pronounced. To handle huge data storage requirements and reduce software and hardware overheads there are some devices like storage area network (SAN), direct attached storage (DAS).although SAN is most popular for block I/O based storage but it is very expensive in price. Network Attached Storage (NAS) is an attractive option because of its low price. But traditional NAS has some drawbacks we have discussed shortly into the Introduction section. Existing software stacks do not provide the low-latency access that fast SSDs require to realize their full potential. So we introduce a new architecture of NAS which can work as both block I/O and file I/O protocol. Thus it will able to reduce the software costs. Moreover it will improve the total performance of SSDs as SSDs are still not cost effective compare to its capacity. The next section describes our design for NAS that removes most of those costs to expose the full performance of SSDs to software.

III. Proposed Model

This section describes the hardware and software components of NAS. The NAS hardware adds a high-speed network interface to a PCIe-attached SSD, so SSDs can communicate directly with one another and retrieve remote data to satisfy local I/O requests. The NAS software components leverage this capability to eliminate most of the software costs.

Figure 1 illustrates the software components that are of interest for data transfer through the I/O path. The NAS metadata server will provide the I/O interface to the user-space world. The I/O system will call in metadata server use MUVFS (or generic file system) to perform their tasks, while individual file system, such as NFS, plugs their code into the MUVFS handlers.

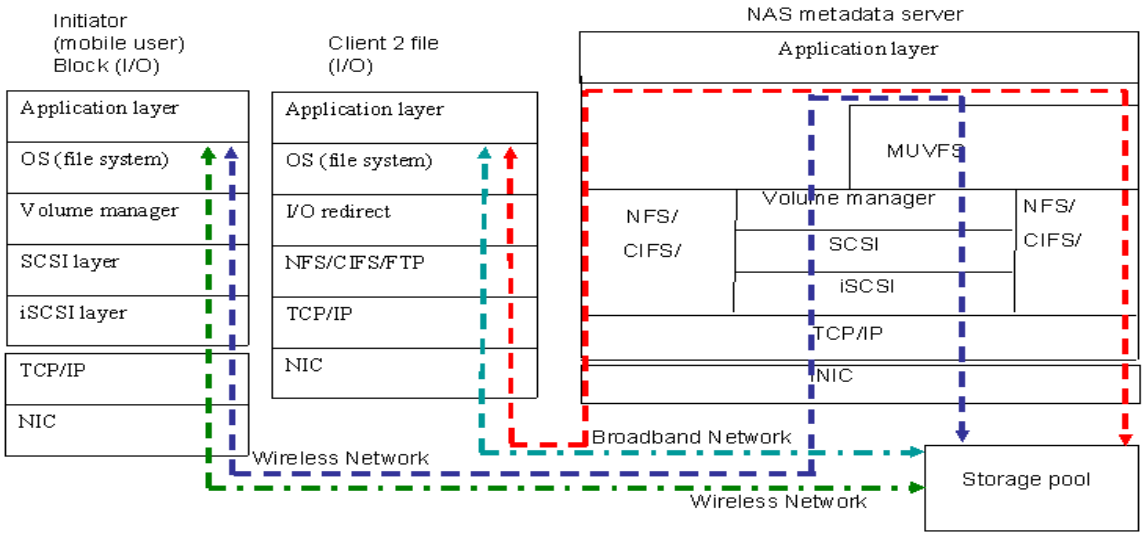

Figure 1: Software Architecture of NAS for Block I/O and File I/O Service.

When NAS offer the block I/O services, it invokes the iSCSI module, the asker is Initiator. Concrete data read/write process is as follows: (1) The block I/O commands (SCSI commands) sent by the users in Initiator are encapsulated into the IP data packets with the iSCSI device diskr and the TCP/IP Protocol stack, then transferred via the wireless network; (2) When the encapsulated packets arrive at the NAS metadata server, they are restored to the previous SCSI commands in an unpacked process, then passed to the MUVFS layer. Having been processed by the MUVFS, the previous I/O commands are packed up once again and sent to an appointed NAS via an inner network; (3) required data blocks are packed into the iSCSI protocol data units by the NAS and returned to the Initiator via a wireless Network. The way, which the appliance communicates with the NAS via a high speed IP channel, is similar to that the appliance communicates with the metadata server. When NAS offers the file I/O service, the data read/write process is almost the same as that in a usual NAS.

A. Multi User Virtualization File Systems (MUVFS)

To support Unix Client, Windows Client and iSCSI mobile appliance, we develop MUVFS system. The Multi User Virtualized File System (MUVFS) is the most important core modular in MNAS, which enables the centralized management of many NAS nodes and provides a unified file system view for clients.

Figure 2 shows the structure of MUVFS. It consists of Unfsd, UiSCSId, Samba, the management partition, an iNAS configuration table, and a virtual partition (logic volume) map table. Unfsd is a wrapper of

the global NFS daemon and provides file services for UNIX/Linux clients. Samba is used for windows clients. UiSCSId is a wrapper of the global iSCSI daemon and provides block I/O services for iSCSI clients (mobile appliances).

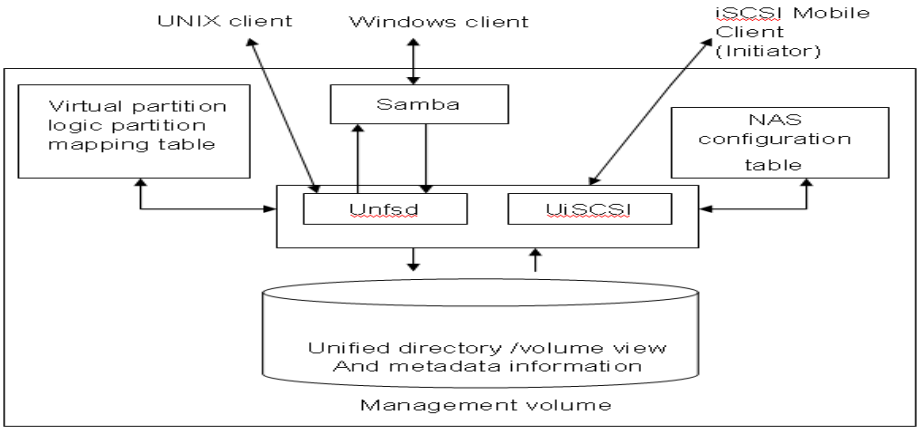

Figure 2: Structure of Multi user Virtualization File System MUVFS

The NAS configuration table contains the system information such as hostnames, independent IP of MNAS members, export points, and so on. The virtual partition (logic volume) map table specified the data partition which stores file entities, etc.

B. Adaption of NAS over SSD

To explore ways of reducing software overheads in these two architectures, we have implemented two hardware devices: A customized network interface card (the NAS NIC) and a custom SSD (the NAS SSD) that integrates both solid state storage and NAS NIC functionality. The NAS NIC exports a block device interface and forwards read and write requests over a network to a remote storage device (in our case, a NAS SSD).

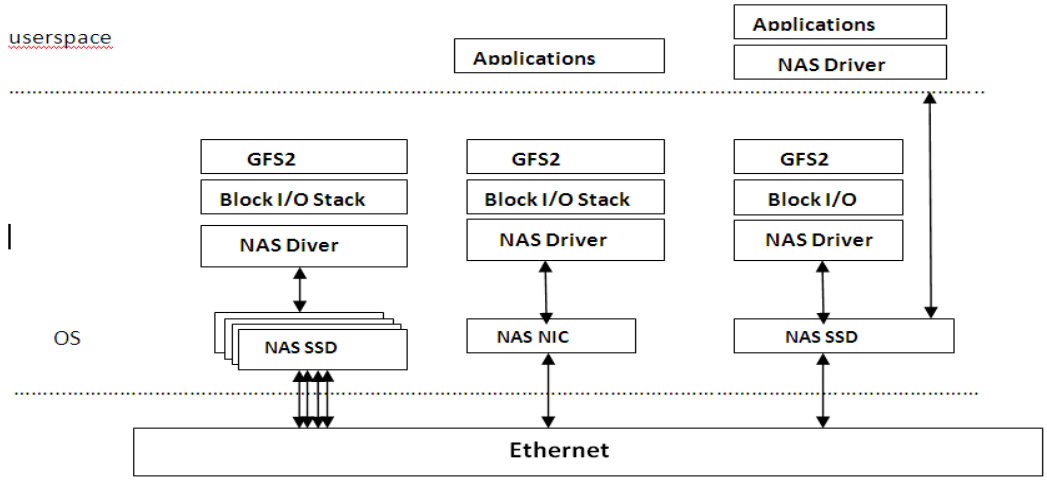

Figure 3: NAS configurations NAS supports multiple storage topologies and two software interfaces. At left, a single machine hosts multiple NAS SSDs, acting a central block server. The center machine hosts a NAS NIC that provides access to remote SSDs. The machine at right hosts a single SSD and is poised to access its local (for maximum performance) or remote data via the userspace interface.

The NAS SSD responds to remote requests without communicating with the operating system. The NAS SSD also exports a block device interface to the host system.The interface's block address space includes the SSD's local storage and the storage on the other NAS SSDs it is configured to communicate with. The NAS SSD services accesses to local storage directly and forwards requests to external storage to the appropriate NAS SSD. This organization allows NAS SSDs to communicate in a peer-to-peer fashion, with each SSD seeing the same global storage address space. NAS further reduces software latency by applying the OSbypass techniques described in [6]. These techniques allow applications to bypass the system call and file system overheads for common-case read and write operations while still enforcing file system protections. The following subsections describe the implementation of the NAS SSD and NIC in detail.

C. Implementation of NAS SSD and NIC

The NAS NIC and the NAS SSD share many aspects of their design and functionality. Below, we describe the common elements of both designs and then the SSD- and NIC-specific hardware. Then we discuss the hardware platform we used to implement them both. Common elements the left half of Figure 4 contains the common elements of the SSD and NIC designs. These implement storage like interface that the host system can access. It supports read and write operations from/to arbitrary locations and of arbitrary sizes (i.e., accesses do not need to be aligned or block-sized) in 64 bit address space. The hardware includes the host-facing PIO and DMA interfaces, the request queues, the request scoreboard, and internal buffers. These components are responsible for accepting IO requests, executing them, and notifying the host when they are complete. Communication with the host occurs over a PCIe 1.1x8 interface, which runs at 2 GB/s, full-duplex. This section also includes the virtualization and permission enforcement hardware described below (and in detail in [6]).

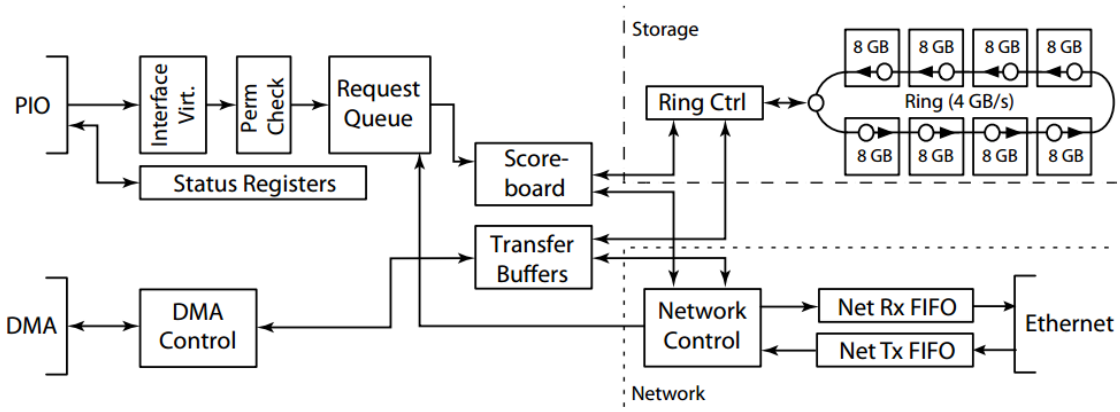

Figure 4: NAS's internal architecture The NAS NIC and SSD both expose a virtualized, storage interface to the host system via PCIe (at left). The network interface attaches to a 10 Gbit network port (bottom-right), while the NAS SSD adds 64 GB of non-volatile storage (top-left). The device services requests that arrive via either the network or host interface and forwards requests for remote storage over the network.

The storage-related portions of the design are shown in the top-right of Figure 4. The SSD contains eight high-performances, low-latency non-volatile memory controllers attached to an internal ring network. The SSD's local storage address space is striped across these controllers with a stripe size of 8 kB.

NAS's network interface communicates over a standard 10 Gbit CX4 ethernet port, providing connectivity to other NAS devices on the same network. The network interfaces plays two roles: It is a source of requests, like the PCIe link from the host, and it is also a target for data transfers, like the memory controllers. The link allows NAS to service remote requests without operating system interaction on the remote node and even allows access to storage on a remote node when that node's CPU is powered off (assuming the SSD has an independent power supply).

NAS requires a lossless interconnect. To ensure reliable delivery of 7network packets, NAS uses Ethernet but employs flow control as specified in the IEEE 802.3x standard [7]. Ethernet flow control can interact poorly with TCP/IP's own flow control and is usually disabled on data networks, but a more recent standard, 802.1qbb, extends flow control to cover multiple classes of traffic and alleviates these concerns.

Ethernet flow control provides the necessary reliability guarantees that NAS needs, but it runs the risk of introducing deadlock into the network if insufficient buffering is available. For example, deadlock can occur if an SSD must pause incoming traffic due to insufficient buffer space, but it also requires an incoming response from the network before it can clear enough buffer space to unpause the link. We have carefully designed NAS's network interface to ensure that it always handles incoming traffic without needing to send an immediate response on the network, thus breaking the conditions necessary for deadlock. We guarantee sufficient buffer space for (or can handle without buffering) all incoming small (non-payload bearing) incoming messages. Read requests allocate buffers at the source before sending the request, and a dedicated buffer is available for incoming write requests, which NAS clears without needing to send an outgoing packet.

The host-facing interface of the NAS SSD and NIC includes support for virtualization and permissions enforcement, similar to the mechanisms presented in [6][7]. These mechanisms allow applications to issue read and write requests directly to the hardware without operating system intervention while preserving file system permissions. This eliminates, for most operations, the overheads related to those software layers. The

virtualization support exposes multiple, virtual hardware interfaces to the NIC or SSD, and the operating system assigns one virtual interface to each process. This enables the process to issue and complete IO operations.
The permissions mechanism associates a permissions table with each virtual interface. The hardware checks each request that arrives via the PCIe interface against the permission table for the appropriate interface. If the interface does not have permission to perform the access, the hardware notifies the process that the request has failed.

The operating system populates the permission table on behalf of the process by consulting the file system to retrieve file layout and permission information.

Permission checks in NAS occur at the source NIC or SSD rather than at the destination. This represents a tradeoff between scalability and security. If permission checks happened at the destination, the destination SSD would need to store permission records for all remote requestors and would need to be aware of remote virtual interfaces. Checking permissions at the source allows the number of system wide permission entries and virtual interfaces to scale with the number of clients. A consequence is that NAS SSDs trust external requests, an acceptable trade-off in most clustered environments.

IV. Experimental Methodology

This section evaluates NAS's latency, bandwidth, and scalability along with other aspects of its performance. We consider throughput as performance metrics, which is the total number of application level bytes carried over an iSCSI connection divided by the total elapsed time taken by the application, as expressed in Bytes per millisecond (B/ms). We used the system timer of PDA, which is based on WinCE in order to measure throughput. Total elapsed time was measured as the time interval from the initial time when the experiment program started generating the first byte of data to the time when the last byte of data was confirmed to have been sent (received). The elapsed time therefore includes all data transfers and all read and write commands as well as responses at all levels of the protocol stack. We perform two kinds of experiments. In the first experiment, we generate an I/O stream of 5 megabytes, and then measure throughput. Our request for a 5 megabytes I/O operation is passed through the file system and the SCSI subsystem to the iSCSI initiator as a number of SCSI commands. Then the initiator sends the commands and data to the target device. At that time, we adapt various settings of the iSCSI parameters to investigate the best parameter values for performance in wireless network. Each data point plotted in every graph of this paper was calculated as the average of 5 runs with identical parameter settings. We also perform the same experiment again with a different size of I/O stream, which is a 100 megabytes I/O stream. We perform the second experiment in order to examine the effect of the characteristics of the high bit error rate within the commercial CDMA network and the variable bandwidth of the network. Our experiment program generates read I/O bursts continuously for 15 hours, and then measures throughput every 1000 seconds. The second experiment shows that the increase of the MaxRecvDataSegmentLength parameter value cannot always bring performance improvement to iSCSI-based Mobile NAS System in unstable wireless network with high bit error rate and variable bandwidth.

A. Experimental Setup:

Our experimental setup consists of a PDA and NAS Storage server connected on to Internet with CDMA 2000 1xEVD0 network. The initiator module embedded in PDA based on windows CE can transmit iSCSI command and data to the target of storage server for I/O. we set the experimental environment as table 1.

	Storage Server or Target	**PDA or Initiator**
OS	Linux Red Hat 8.0 (Kernel version 2.4.18)	Windows CE 4.0
CPU	PIII800MHz	Intel P×A270
Memory	512 MB	256 MB
NIC	1 Gbps LAN	CDMA 2000 1× EV-DO

Table1: Experimental Environment information table between Target (storage server) and Initiator (PDA).

The storage server is connected with RAID subsystem, which has the storage capacity of 1TB (Terabyte) through FC

B. Experimental Results:

Our experiment setup consists of a PDA and NAS Storage server connected on to Internet with CDMA 2000 1xEVD0 network.

Figure 5 illustrates that the increase of the Number of sectors per command causes the performance improvement in the wireless network. It is very similar to the experiment results in a wired network. However, there are the performance falling at the Number of sectors per command value of 2048 with MRDSL of 1Kbytes and 4Kbytes. At the MRDSL values of 1Kbytes and 4Kbytes, there are 1% and 15% de-creases respectively in throughput as Number of sector per command increases from 1024 to 2048. In the wireless network, the results related to the values of MRDSL from our experiment are different from those in a stable wired network. The

throughputs are better at the MRDSL values of 1Kbytes, 2Kbytes and 4Kbytes than the values of 512bytes and 8Kbytes. At the Number of sectors per command value of 2048, the throughput of 2Kbytes MRDSL is increased by 48% and 52% respectively compared with those of 512bytes and 8Kbytes MRDSL. However, when MRDSL value is among 1Kbytes, 2Kbytes and 4Kbytes, it is difficult to say which one is the best value, for different MRDSL values may outperform others at different values of Number of sectors per command. For example, MRDSL of 2Kbytes can bring better throughputs when Number of sectors per command is between 256 and 512, while MRDSL of 1Kbytes is better when Number of sectors per command is around 1024 as we can see from figure 2. Therefore, it is not always correct in wireless network to increase the value of MRDSL to get performance improvement. Correspondingly, the default value 8Kbytes of MRDSL in standard is also not suitable for a wireless network. The reason of performance dropping at the MRDSL value of 512bytes is due to extra PDUs overhead. When MRDSL is 8Kbytes, however, the reason of performance falling is due to the increase of data segments which are transmitted continuously by TCP until detecting congestion even though congestion occurred in the wireless network with narrow bandwidth. Therefore, we suggest that you should use the parameters settings of the MRDSL of 1Kbytes, 2Kbytes and 4Kbytes with the Number of sectors per command values of 1024 or 2048 when per-forming a small file read operation in CDMA network

In write operation, the Number of sectors per command is kept constant at 1024(512Kbytes) and the MaxBurstLength (MBL) varies from 512Bytes to 256Kbytes. We use the fixed Number of sectors per command, because the value of the parameter MaxBurstLength limits the total amount of all data segments of all PDUs which were requested by R2T. Figure 6 shows that the throughput is increased as MBL increases from 512Bytes to 128Kbtyes in a wireless network, after that there is a slight decrease in throughput when MBL is around 256Kbytes. The same kind of decrease in throughput happens in read operations too, which is caused by the narrow bandwidth of CDMA network. The number of iSCSI PDUs which are continuously passed to the TCP layer can be increased by increasing either the value of MaxBurstLength for write operation or Number of sectors per command for read operation, then the sender will transmit more segments until recognizing congestion in wireless network, and thus causes the performance falling of iSCSI based NAS for mobile appliances. In a write operation, the result is more obvious because the CDMA 2000 1x EV-DO network has the narrower bandwidth for upload than that for download. Therefore it is not always true that the increase of MBL value would cause the performance improvement. The MBL value of 256Kbytes in standard is not suitable too. When the MRDSL value is 2Kbytes or 4Kbytes, the throughputs are better than that is 512bytes or 1Kbytes or 8Kbytes. At the MBL of 128Kbytes, the throughputs of MRDSL of 2Kbytes are 26% and 31% increases respectively than those of MRDSL of 512bytes and 8Kbytes. However, when MRDSL is at a value of 2Kbytes or 4Kbytes, it is hard to say which one is the best value. From the figure 4 we can know that either of them may achieve better performance within a certain value range of Number of sectors per command. From these we can also see the standard value for wired network is not suitable here for a wireless network. Therefore, we suggest that you use the parameters settings of the MRDSL of 2Kbytes and 4Kbytes with the MBL of 128Kbytes when performing a small file write operation in CDMA network.

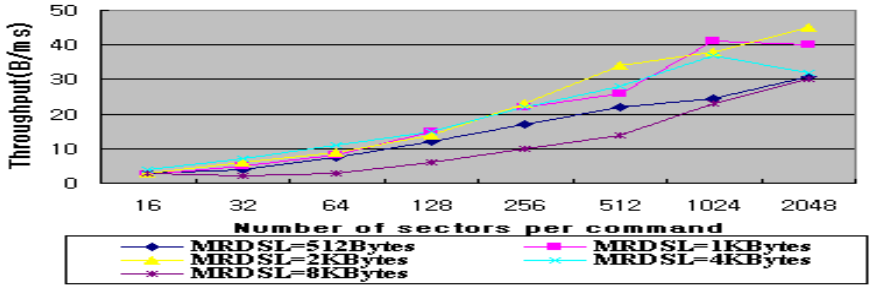

Figure 5: Effect of Number of sectors per command on throughput for 5 Mbytes read operation

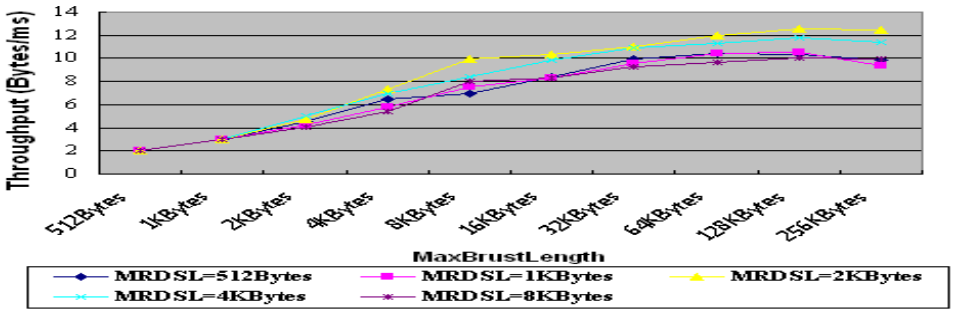

Figure 6: Effect of MaximumBrustLength (MBL) for 5 Mbytes write operation

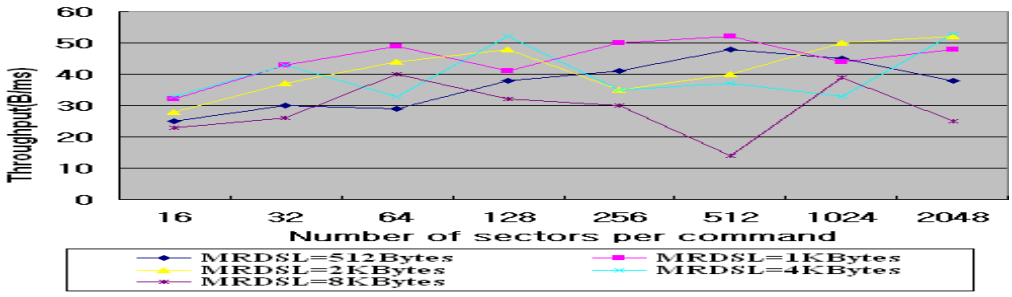

Figure 7: Effect of Number of sectors per command on throughput for 100 Mbytes read operation

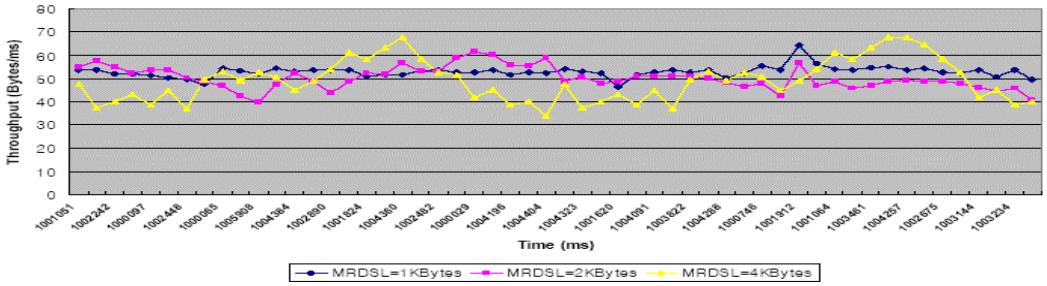

Figure 8: Effect of MRDSL on throughput about 15 hours in wireless network

Figure 5 shows the results from our experiment of 100 megabytes read operation. When MRDSL is at a value of 1Kbyte, the throughput is better than that at the value of 512bytes or 8Kbytes. When the MRDSL value is 1Kbytes, 2Kbytes and 4Kbytes, however, it is difficult to say which one is the best value. In the case of large size of I/O burst, such as 100 megabytes read operation, the best value for performance is also different from the standard value. When the MRDSL value is 2Kbytes, 4Kbytes and 8Kbytes, the performance is sharply dropped or increased with the increase of Number of sectors per command. At 8Kbytes of MRDSL value, there is a drastic decrease by 64% in throughput as Number of sectors per command increases from 64 to 512, after that there is an increase in throughput suddenly when the Number of sectors per command is around 1024. This experiment has different results from 5 megabytes I/O operation because the characteristics of a high bit error rate within the wireless channel, and a narrow and variable bandwidth of the wireless channels affect the large file read operation which was performed for a long time. Therefore, we perform the second experiment in order to examine the effect of the characteristics of CDMA network more clearly. In the experiment for continuous read I/O bursts, we measure throughput every 1000 seconds in order to observe the degree of performance increasing or decreasing with respect to 1Kbytes, 2Kbytes and 4Kbytes MRDSL parameter values respectively, and then the Number of sectors per command is kept constant at 1024(512Kbytes). Among the 1Kbytes, 2Kbytes, and 4Kbytes MRDSL, there are the fewest changes in throughput of 1Kbytes of MRDSL in figure 6. From the results, we found out that the smaller size of iSCSI PDU could be less affected by the characteristics of a high bit error rate within the wireless channel, and a narrow and variable bandwidth of the wireless channels as we explained in section 3. There-fore we suggest that you should set the MRDSL parameter value at 1Kbytes when performing a large file I/O operation while moving in CDMA network.

C. Performance Comparison between File Unit I/O and block Unit I/O:

For the comparison of file unit I/O performance and block unit I/O performance, we use CIFS as the file unit I/O remote storage system. window CE, which is OS of the mobile appliance ,basically provides CIFS.so it can communicate with SMB protocol of storage server using WNet API [10].we set up the CIFS as mobile client and MB as storage server for file unit I/O remote storage system. As a block unit I/O based – remote storage system we put our new iSCSI target module as storage server. In both cases, a number of sizes of data are requested repeatedly during the experiment since these variables of data size could have a significant effect on the system performance. We assumed that the difference of performance resulted from the protocol itself is enough to disregard if it is the same I/O approach. Because we learned that the difference of performance among the unit I/O protocol such as NBD,iSCSI is very little to be ignored in [11].

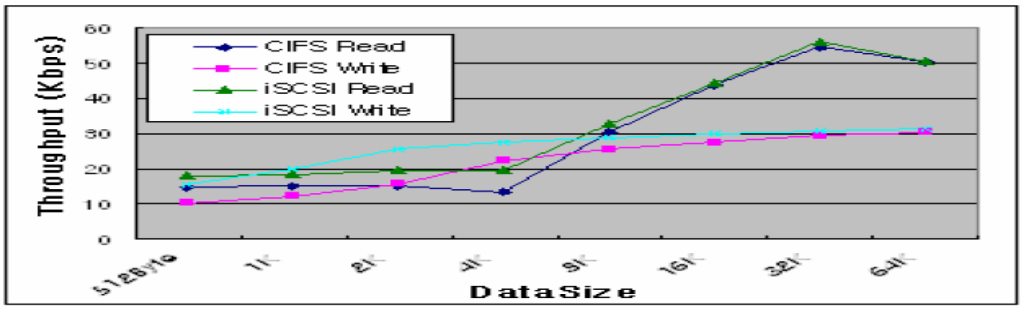

Figure 9: Throughput of iSCSI vs CIFS

For the measurement, we generate different sizes of workloads repeatedly and the ratio of read and writes is set to three to one following general storage I/O pattern. Considering the bandwidth we changed the data request size from minimum 512 bytes to maximum 64 kbytes respectively si.Fig.9 and Fig.10 show the result of measurement our system that adapts block unit I/O approach using iSCSI protocol makes the better throughput than that of the traditional remote storage system using file unit I/O approach .It also shows the better average performance about 16.4% in reads and 29.6% in write.

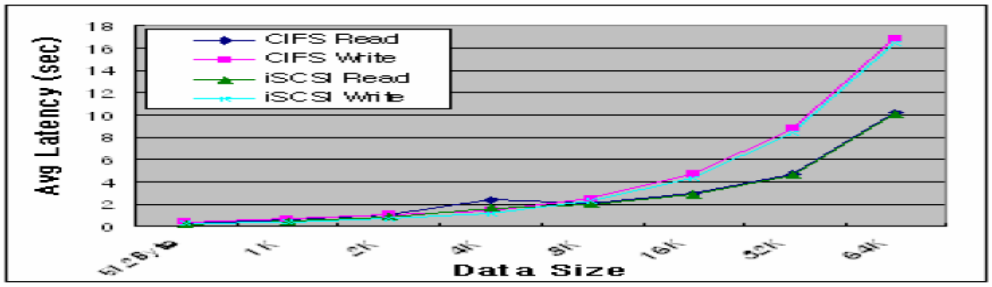

Figure 10: Avg Latency of iSCSI vs. CIFS.

In the latency test ,as it is seen from the figure 8.20,our system that uses iSCSI protocol also shows the slightly shorter response time than that of CIFS.We analyzed the reason that the result does not present distinct difference in response time is resulted from the delay in protocol conversion .

D. Performance Analysis:

To know the main factor that affects the throughput of remote storage system in mobile environment, we performed tests with various views. Most of all we cheek the CPU consumption that has been one of the major issues in IP storage network. As like the previous case of latency measurement, the CPU consumption rate of iSCSI is also lower than that of CIFS to every request size [Fig.11].

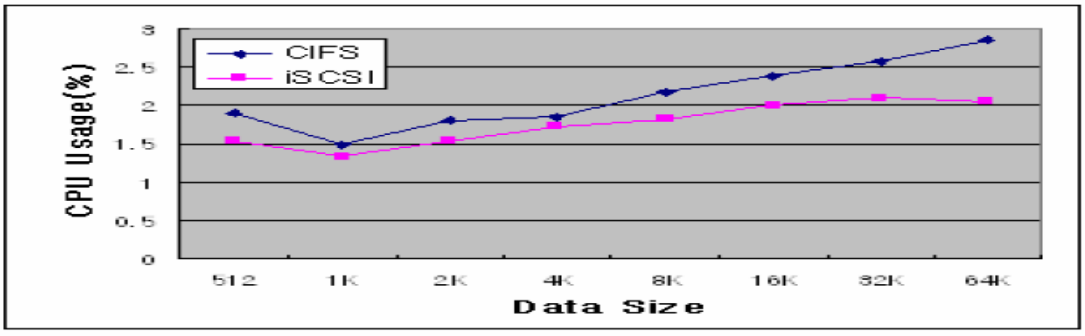

Figure 11: CPU Usage of iSSI vs. CIFS

These results show that the CPU consumption is much affected by the file system layer of file unit I/O protocol than protocol conversion overhead of block unit I/O protocol. While the file unit I/O protocol like CIFS should have loads from storage servers file processing and frequent from storage server's file system processing and frequent context switching between application level and kernel level, the block unit I/O protocol like iSCSI does not need to. The influence of protocol conversion overhead that has been regarded as the main issue in IP storage is not so serious like CDMA-2000 1x wireless module. As to the result, which we've analyzed from the measurement, block unit I/O approach is more efficient in general storage data I/O operation.

V. Conclusion

We have described a new NAS architecture that support both file I/O and Block I/O designed for solid state memories. This NAS integrates network functionality into a high-performance SSD to allow access to remote data. We also studied and analyzed iSCSI parameters defined by its standard to investigate those effects. We performed experiments to find out the best performance parameters setting with NAS system for large amount of storage facilities via wireless network. We are suggesting to use parameters settings of the MRDSL of 1Kbytes, 2Kbytes and 4Kbytes with the Number of sectors per command values of 1024 or 2048 to perform a small file read operation in CDMA network and for performing a small file write operation in CDMA network the parameters settings of the MRDSL of 2Kbytes and 4Kbytes (MBL of 128Kbytes). For large files operation, we found that the smaller size of iSCSI PDU could be less affected by the characteristics of a high bit error rate within the wireless channel, and a narrow, variable bandwidth of the wireless channels. Our proposed NAS illustrates the ongoing need to redesign computer system architectures to make the best use of fast non-volatile memories.

Acknowledgements

In this paper we adapted iSCSI protocol to achieve better optimized result of online based MapReduce mechanism.

- This iSCSI research (Grants No.2009-0075576) was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by Ministry of Education, Science and Technology.
- This iSCSI research work (Grants No. 47504) was supported by Business for Cooperative R&D between industry, Academy and Research Institute funded Korea small and Medium Business Administration 2011.
- Research Lab : Gluesys Lab, Anyang University, Korea and East West University, Bangladesh.

References

[1]. SAM-3 Information Technology – SCSI Architecture Model 3, Working Draft, T10 Project 1561-D, Revision7, 2003
[2]. S.M Allayear, S.S Park: "iSCSI MultiConnection and Error Recovery Method for Remote Storage System in Mobile Appliance". ICCSA-06, Glasgow, LNCS 3891, pp.641-650, May 2006.
[3]. D.Kim, M. Ok, M.-s. Park. An Intermediate "Target for Quick-Relay of Remote Storage to Mobile Devices". Proc. Of ICCSA, May 2005.
[4]. Sura Park, Bo-Suk Moon, Myong-Soon Park: Design, Implement and Performance Analysis of the Remote Storage System in Mobile Environment, Proc. ICITA 2004.
[5]. M. Ok, D. Kim, M.-s. Park, UbiqStor: A Remote Storage Service for Mobile Devices, The Fifth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 04) Singapore, December 2004.
[6]. A. M. Caulfield, T. I. Mollov, L. Eisner, A. De, J. Coburn,and S. Swanson. Providing Safe, User Space Access to Fast,Solid State Disks. In Proceeding of the 17th internationalconference on Architectural support for programminglanguages and operating systems, New York, NY, USA,March 2012. ACM.
[7]. https://www.google.com.bd/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CBwQFjAA&url=http%3A%2F%2Fcseweb.ucsd.edu%2F~swanson%2Fpapers%2FISCA2013QuickSan.pdf&ei=cdryU4KhMMuY1AXruoGAAQ&usg=AFQjCNEtNmsFfhof1M-J_b9gLX_Ceqw1MQ&sig2=-Ct0a6poen4yn0oPEXV6Ww&bvm=bv.73231344,d.d2k .
[8]. https://www.google.com.bd/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CBoQFjAA&url=http%3A%2F%2Fnsrc.cse.psu.edu%2Ftech_report%2FNAS-TR-0036-2006.pdf&ei=a-XyU4TxHJK1uAT11YLIBg&usg=AFQjCNFQZNzJK9YBg_-9t7Jmh7de4FCh1A&sig2=wZuMhRpm2BBNbYdQW378vQ&bvm=bv.73231344,d.c2E .
[9]. http://www.amsstorage.com/downloads/DakotaRAID-LP/WP-NAS_to_NAS_Date_Replication.pdf
[10]. MSDN,http://msdn.microsoft.com/library/.
[11]. B.Moon, S.Park, M.Park,"A Performance Analysis of CIFS, NBD, iSCSI on the Wireless Network by Using CDMA-2000 1x".
[12]. http://medialab.bme.hu/internet_docs/tech/techno15/panel2.html
[13]. http://medialab.bme.hu/internet_docs/tech/techno14/panel2.html
[14]. http://www.wwpi.com/index.php?option=com_content&id=7311:what-are-ssds-and-how-do-they-function-in-storage-systems&Itemid=2701018