Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Deconstructing Network Attached Storage systems

Yuhui Deng [*]

EMC Research China, Beijing 100084, PR China

ARTICLE INFO

Article history:
Received 8 April 2008
Received in revised form
17 November 2008
Accepted 26 February 2009

Keywords:
Network Attached Storage
Performance bottleneck
Storage stack
Network stack
RAID

ABSTRACT

Network Attached Storage (NAS) has been gaining general acceptance, because it can be managed easily and files shared among many clients, which run different operating systems. The advent of Gigabit Ethernet and high speed transport protocols further facilitates the wide adoption of NAS. A distinct feature of NAS is that NAS involves both network I/O and file I/O. This paper analyzes the layered architecture of a typical NAS and the data flow, which travels through the layers. Several benchmarks are employed to explore the overhead involved in the layered NAS architecture and to identify system bottlenecks. The test results indicate that a Gigabit network is the system bottleneck due to the performance disparity between the storage stack and the network stack. The tests also demonstrate that the performance of NAS has lagged far behind that of the local storage subsystem, and the CPU utilization is not as high as imagined. The analysis in this paper gives three implications for the NAS, which adopts a Gigabit network: (1) The most effective method to alleviate the network bottleneck is increasing the physical network bandwidth or improving the utilization of network. For example, a more efficient network file system could boost the NAS performance. (2) It is unnecessary to employ specific hardware to increase the performance of the storage subsystem or the efficiency of the network stack because the hardware cannot contribute to the overall performance improvement. On the contrary, the hardware methods could have side effect on the throughput due to the small file accesses in NAS. (3) Adding more disk drives to an NAS when the aggregate performance reaches the saturation point can only contribute to storage capacity, but not performance. This paper aims to guide NAS designers or administrators to better understand and achieve a cost-effective NAS.

1. Introduction

With the explosive growth of the Internet and its increasingly important role in our daily lives, traffic on the Internet is increasing dramatically, more than doubling every year. Under such circumstances, the enormous amount of digital data is identified as the key drivers to escalate storage requirements including capacity, performance, and quality of service. Due to the explosive data growth, three typical storage system architectures have been adopted to satisfy the increasing storage requirements, namely Direct Attached Storage (DAS), Storage Area Network (SAN), and Network Attached Storage (NAS) (Deng et al., 2005, 2006; Gibson and Meter, 2000; Mesnier et al., 2003; Patterson et al., 1988; Riedel, 2003; Varki et al., 2004). Different implementation of each of the storage architectures has a different effect on the overall system performance and application circumstances. DAS cannot be scaled because the number and type of storage devices that can be attached to a server are limited. The server could also quickly become a major system bottleneck due to store-and-forward data copying between the server and the attached storage devices (Deng et al., 2005). The complex administration and cost are indicated by many researches as the key barriers to adopt SAN solutions. Service-oriented storage (Chuang and Sirbu, 2000) and storage grid (Deng et al., 2008a; Deng and Wang, 2007) are also emerging as new storage system architectures to tackle the large-scale, geographically distributed storage demands. However, it is still in its infancy.

NAS has recently been gaining general acceptance, because it can be managed easily and files shared among many clients that run different operating systems. It also offers some other advantages such as parallel I/O, incremental scalability and lower operating costs (Sohan and Hand, 2005). The advent of Gigabit Ethernet and high speed transport protocols further facilitates the adoption of NAS. Recently, the community is very active in designing a large storage system that involves multiple NAS nodes, while providing scalability and simplified storage management. X-NAS (Yasuda et al., 2003) is a highly scalable and distributed file system designed to virtualize multiple NAS systems into a single file system view for different kinds of clients. NAS switch (Katsurashima et al., 2003) is proposed and designed as an in-band switch between clients and NAS systems to provide a single virtual NAS system for users and

[*] Tel.: +86 10 86108216.
E-mail addresses: deng_derek@emc.com, yuhuid@hotmail.com (Y. Deng).

1084-8045/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2009.02.006

administrators. Bright and Chandy (2003) designed a clustered storage system with a unified file system image across multiple NAS nodes. The system provides simplified storage management and scalable performance. CoStore (Chen et al., 2003) is a serverless storage cluster architecture that evenly distributes system responsibilities across all collaborating NAS nodes without a separate central file manager. The architecture has the potential to provide scalable performance and storage capacity with strong reliability and availability. RISC (Deng, 2008) divides storage nodes into multiple partitions to facilitate the data access locality. Multiple spare links between any two storage nodes are employed to offer strong resilience to reduce the impact of the failures of links, switches, and storage nodes. The scalability is guaranteed by plugging in additional switches and storage nodes without reconfiguring the overall system. Two I/O-aware load-balancing methods are proposed in Qin et al. (2005a) to improve the overall performance of a cluster system running I/O intensive applications. The proposed approaches dynamically identify I/O load imbalance on nodes of a cluster, and determine whether to migrate some I/O load from overloaded nodes to other underloaded nodes to alleviate the system bottleneck. Traditional load-balancing approaches normally employ static configuration for the weights of resources. These methods cannot be adjusted automatically for the dynamic workload. A feedback control mechanism is proposed to improve overall performance of a cluster with I/O-intensive and memory-intensive workload (Qin et al., 2005b). However, the aggregate performance of a NAS cluster can be dominated by the performance of a single NAS node.

A lot of research efforts have been invested in designing and developing performance optimization methods, which may be able to improve the performance of NAS. TCP offload is proposed to offload the entire TCP/IP stack to the network adapter. The method can reduce the overheads of interrupts and TCP copy-and-checksum (Mogul, 2003; Sarkar et al., 2003). Remote Direct Memory Access (RDMA) moves data directly from the memory of one computer into the memory of another computer without involving the operating systems at both sides. Zero copy is used to directly transfer data between the application memory and network adapter. VI Architecture (Compaq, Intel, Microsoft, 1997) is a user-level memory mapped architecture. By avoiding the kernel involvement, the architecture is designed to eliminate the software overhead imposed by traditional communication models, thus achieving low latency and high bandwidth. DAFS (Magoutis et al., 2002) is a network file system that allows applications to transfer data while bypassing the potential performance bottlenecks such as operating system control, buffering, network protocol, etc. DAFS works with any interconnection that supports Virtual Interface (VI) including Fibre Channel and Ethernet. Brustoloni (1999) investigated a solution that allows data to be passed between networks and file systems without copying and without changing the existing interfaces. Sohan and Hand (2005) showed that the current NAS architecture performs poorly mainly because of multiple data copies, poor kernel resource accounting, inadequate process isolation and poor application customisation facilities. They proposed and designed a user-level NAS architecture that does all file and buffer layer processing in user space without any specific hardware support. The experimental results illustrate that this architecture produces better performance than the traditional architecture. Because the traditional NAS adopts general computer system architecture, NAS may also benefit from other generic performance enhancing methods such as intelligent scheduling, caching, and prefetching.

A distinct feature of NAS is that NAS involves both network I/O and file I/O. Gigabit Ethernet further propels the wide adoption of NAS. It seems that the performance requirements of providing storage over a network should not be a problem due to the involved Gigabit network. Modern operating systems tend to be structured in distinct, self-contained layers, with little inter-layer state or data sharing. Communication between layers uses well defined interfaces that are difficult to circumvent (Sohan and Hand, 2005). An NAS normally consists of several key layers including storage devices, logical volume manager, local disk file system, and network file system. Each layer in the system takes its cut off performance from the layer below. Each layer has an amount of overhead that reduces the performance, which is available to the next higher layer of the system (Riedel, 2003). Due to the complexity of NAS system, how to identify the system bottlenecks and employ the most effective approach to eliminate the bottlenecks and boost the performance are challenging problems.

In this paper, we discuss the system architecture of a typical NAS and isolate some key components within the NAS by tracking the data flow. We use a variety of benchmarks to explore the overhead involved in the layered NAS architecture and identify the system bottlenecks, because we only need to speed a small portion (system bottleneck) of the overall system to achieve the maximum performance gains in terms of Amdahl's law. The key contribution of this paper is to identify some cost-effective methods that can boost the performance of NAS from a large number of optimization methods.

The remainder of the paper is organized as follows. Section 2 introduces the architecture and the data flow of a typical NAS. The testbed and performance measurements are depicted in Section 3 in detail. There are some discussions of the work and the indications of future research in Section 4. Section 5 concludes the paper with remarks on the contributions of the paper.

2. System overview

The hierarchy of storage in current computer architectures is designed to take advantage of data access locality to improve overall performance. Each level of the hierarchy has higher speed, lower latency, and smaller size than lower levels. For decades, the hierarchical arrangement has suffered from significant bandwidth, latency, and cost gaps between the RAM and disk drive (Pugh, 1971). The performance gap has been widened to six orders of magnitude in 2000 and continues to widen by about 50% per year (Schlosser et al., 2000). The disk I/O subsystem is repeatedly identified as a major bottleneck to system performance in many computing systems. The bottleneck of disk I/O could significantly impact the application performance. NAS employs RAID subsystems, which adopt multiple disk drives working in parallel to achieve high performance, thus alleviating or even eliminating the I/O bottleneck. The performance of the storage subsystem scales with the number of disk drives. The performance bottleneck of a NAS could be migrated from disk I/O to other components with the increase in number of disk drives.

2.1. Architecture overview of NAS

Currently, there are two popular data sharing mechanisms including network file system and an iSCSI. The two mechanisms are fundamentally different. Network file system enables files to be shared among multiple client machines. iSCSI is a block level protocol that encapsulates SCSI commands into TCP/IP packets and transfers the data through TCP/IP network. iSCSI permits applications running on a single client machine to share remote data, but it is not directly suitable for sharing data across multiple machines (Radkov et al., 2004). An NAS is a file server that normally presents a file interface to the network by employing

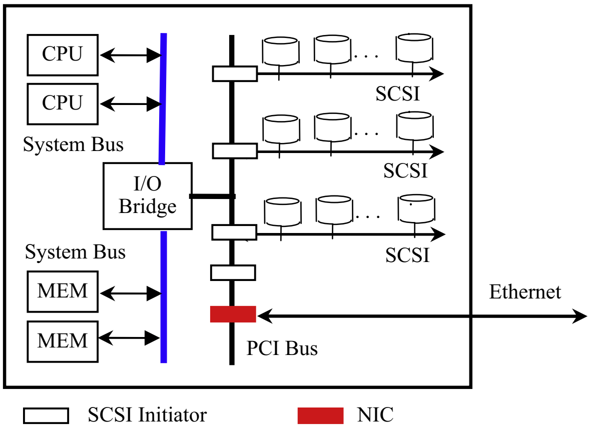

Fig. 1. Architecture of a typical Network Attached Storage system.

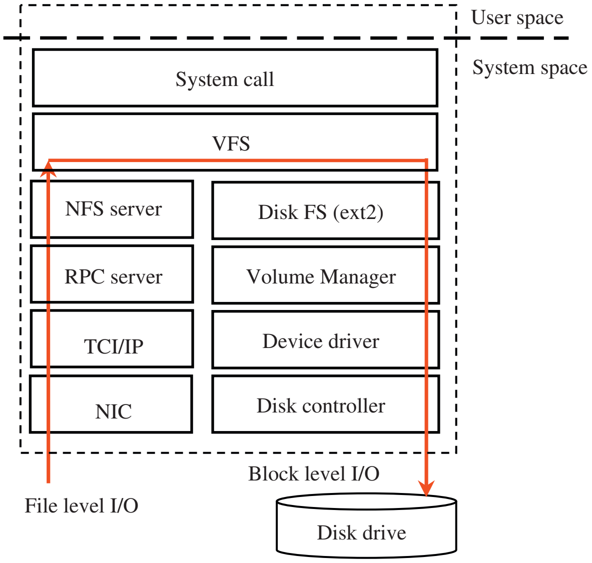

Fig. 2. Data flow of file level I/O through a typical NAS.

Windows Common Internet Files System (CIFS) or Network File System (NFS). A typical NAS connects to a LAN and provides file sharing services to many clients (Riedel, 2003). Compared with the storage system of laptops and desktops, a salient feature of the storage subsystems in a file server is that multiple high performance disk drives are involved. The multiple disk drives are normally organized as a RAID with data distributed across the disk drives to aggregate the storage capacity, performance, and reliability (Riedel, 2003).

Fig. 1 illustrates the architecture of a typical NAS and the corresponding storage subsystem. The architecture consists of CPU, main memory, multiple disk drives, etc. The I/O bridge has interfaces for the system bus and the memory bus, which are connected to CPU and memory, respectively. Multiple Small Computer System Interface (SCSI) bus adapters, which are inserted into the Peripheral Component Interconnect (PCI) slots, are used as initiators to connect the SCSI disk drives. Each SCSI adapter can connect multiple SCSI disk drives. For example, an SCSI bus with 16 bits width can connect up to 15 SCSI disk drives, which act as target devices of the adapter. The disk drives can be organized as one or multiple large logical disk drives in terms of different RAID configurations. Data accesses are provided through the Network Interface Card (NIC) to the Ethernet.

2.2. *Data flow and key components of NAS*

An NAS is a complex system composed of several components such as CPU, main memory, disk drive controllers, buses, and disk drives. It would be very complicated and unnecessary to analyze all the components which the data goes through, because it is difficult to extract the behaviours of the NAS from all the details. Varki et al. (2004) identified CPU, cache, and disk drives as the key components to model the performance of a real RAID. This is a quick way to isolate the potential problem areas with comparatively little effort. For simplicity, we have to identify the key components of an NAS. Fig. 2 shows the basic data flow of file level I/O through a NAS.

NAS is unique because it contains the characteristics of both the storage and networking subsystems. The two subsystems are correlated with each other. NAS normally adopts CIFS or NFS to provide a file interface to the network. We use NFS (Pawlowski et al., 1994) in the following analysis and tests. NFS employs a client/server model and implements a standard network protocol, which provides file sharing to the remote clients transparently.

The data flow in an NAS normally takes following process: when a client application wishes to write a file to the NAS, it calls a write ( ) system call, which invokes a request to the NAS via the Remote Procedure Calls (RPC) layer. The paramters and data will be sent to the NAS as messages through TCP/IP stack by the RPC. The messages are then transferred to the RPC server from network to the system memory cache through the NIC and the system bus of the NAS. The RPC server unpacks and passes the messages to the NFS server protocol. The NFS server protocol passes the data to the local disk file system (e.g. ext2/ext3) of NAS through the VFS. According to write through policy, the data to be written will travel via the volume manager, device driver, PCI bus, disk controller, and finally be written to the disk drives. If write back policy is chosen, the data in the memory cache will be written back to the disk drives when the data is evicted from the cache. A reply will be finially sent back to the client through the NFS, RPC, and TCP/IP protocol layers. When a read request from NAS client wishes to access data stored in the NAS, it first checks the memory cache. If the data can be found in the cache, the read request will be served and the corresponding data will be wrapped into messages and then sent to the client through TCP/IP stack by the RPC server immediately. Otherwise, the required data has to be first retrieved from the disk drives, then pass through the peripheral bus to the bus adapter (SCSI in Fig. 1), then across the PCI bus into the system memory cache through the system bus, and finally the data will be encapsulated in packages and go through the PCI bus and NIC to the client.

3. Experimental study

Section 2.2 illustrates that a data access has to travel through multiple layers within a typical NAS. In this section, we will employ some benchmarks to explore the cut off performance of the layered architecture of a real NAS.

3.1. *Testbed*

Table 1 illustrates the system configurations of a real NAS that consists of ten 400 GB SATA disk drives (HDS724040KLSA80). One disk drive is used to install the operating system. The remaining nine disk drives are adopted as data disks. Each disk drive connects to a 3Ware disk controller with a specified SATA bus.

Table 1
System configurations of a typical NAS.

CPU	Intel(R) xeon(TM) CPU3.2 GHz (dual core)
Memory	1 GB
Hard disk	SATA HDS724040KLSA80 (400 GB)
Disk controller	3Ware Inc 3ware 9xxx-series
NIC	Intel PRO/1000 MT server adapter
PCI	64 bit, 66 MHz
Linux	SUSE 9.3 (Kernel : 2.6.11.4–20a-smp)
Local disk FS	Ext3
Network FS	NFS v3.0
Logical volume manager	Linux Software RAID

Table 2
Disk characteristics of HDS724040KLSA80.

Interface	SATA (1.5 Gb/s)
Capacity	400 GB
Disks/heads	5/10
Disk cache	8 MB
Rotational speed (RPM)	7200
Media transfer rate (max)	757 Mbits/s
Interface transfer rate (max)	150 MB/s
Sustained data rate (zone 0–zone 29)	61.4–29.8 (MB/s)
Seek time (average)	8.5 ms

Table 3
Raw performance of a single SATA disk drive.

	Bandwidth (MB/s)		Through (IO/s)	
	Write	Read	Write	Read
Sequential access	50.897	60.561	5904.01	4940.77
Random access	51.505	60.549	5937.50	4940.79

The 3Ware disk controller is inserted into a 64 bit 66 MHz PCI slot. Table 2 shows the characteristics of the SATA disk drive (HDS724040KLSA80 disk specification). The hardware RAID (Denehy et al., 2004) (e.g. Adaptec 2200S hardware RAID controller) subsystem manages disk drives independently from the host and presents to the host a single logical disk. The software RAID (Denehy et al., 2004) implements the RAID functions in the kernel of operating system. Compared with hardware RAID, a software RAID is much cheaper but requires processing power from the host. We employ software RAID in the NAS testbed.

There are three main I/O performance metrics, which are normally employed to measure the storage system performance: bandwidth, throughput, and response time. Bandwidth is the amount of data processed by the system divided by data transfer time. Throughput refers to the rate at which I/O requests are served and processed. Response time is the time it takes by an I/O request from initiating the I/O operation through waiting for to receiving service. Bandwidth measures how much data can be accessed simultaneously. Throughput measures how many requests can be served within a specified time. Response time measures how fast an individual request can be stored and retrieved. Because the response time can be obtained from the throughput in terms of Little's Law, we measure the bandwidth and throughput of the NAS using different workloads.

3.2. Performance of disk drive and RAID

We employed Xdd (XDD6.5) as a benchmark to measure the performance of the SATA disk drive and the corresponding RAID subsystem. Xdd is a program that measures the performance of data transfer between host memory and disk drives. Reading data on most operating systems uses a file system buffer cache to speedup the performance. In this mode, data is read from a disk drive into a file system buffer and then copied into the Xdd I/O buffer by the processor. The written data is first copied into the file system buffer and later actually written to the disk drive. Direct I/O was adopted in our test to bypass the use of the file system buffer cache and force all read and write requests to access the disk drives. Each write operation causes the file system to allocate storage space on the disk to accommodate the data being written. These allocation operations can require additional disk drive accesses, which involve some overhead. We pre-allocated the space before the file write operations take place to minimize the effects of the space allocation process. Because we try to avoid the impact of the disk file system above the disk drives on the performance, we call the performance "raw performance" in the discussion of this section. To clear the buffer cache and guarantee that the real disk drive accesses take place, each measurement accessed a total dummy file size of 4 GB, which is four times as much as the system memory size, thus displacing any cached data. The bandwidth was measured with a file size of 4 MB, and the throughput was measured with a file size of 8 KB.

Table 3 shows the raw performance of a single SATA disk drive used in NAS. The data transfer rate of a disk drive consists of two parts. The first part is an external transfer rate adopted to measure the transfer rate between host memory and disk cache. The second part is employed to measure the transfer rate between disk cache and disk storage media. This part is called Internal Data Rate (IDR). Due to the mechanical components in the disk drive, the IDR is much lower than the external transfer rate. Table 2 indicates that the maximal media transfer rate (IDR) of the disk drive is 94.6 MB/s. Our tests show that the maximal data transfer rate is 60.6 MB/s (sequential read), which is 64.1% of the datasheet IDR. The reason is that each test request has to travel from Xdd I/O buffer to the disk drive through the device driver and disk controller, and the involved layers incur some overheads. Riedel (2003) reported that most systems produce only one-half to two-thirds of the theoretical bandwidth due to the involved overhead. Our measured bandwidth is consistent with his report.

Most of NAS systems adopt a volume manager (e.g. LVM; Logical volume manager) with RAID technology implemented in it to aggregate the storage capacity, performance, and reliability, and to export the NAS side file system to all its clients through the NFS protocol. RAID uses multiple disk drives to store and distribute data. The basic idea of RAID is dividing data into strips and spreading the strips across multiple disk drives. Different distribution policies endow the RAID subsystem with different features. RAID0 is not widely used because it does not guarantee fault tolerance, but it provides the best performance. We will explore the impact of RAID0 on the system bottleneck since it can fully leverage the hardware capability. RAID1 offers the best fault tolerance but wastes storage capacity. RAID5 guarantees fault tolerance by sacrificing some performance. Energy efficiency has long been a challenge in the storage community. From a power standpoint, one disk drive is not a problem. However, for a large-scale storage system involving hundreds or thousands of disk drives, the energy consumption and heat dissipation will quickly become a big headache (Deng et al., 2008b). RAID1 requires more disk drives than RAID5 to offer the same available storage capacity, thus consuming more energy. Therefore, it is more reasonable to investigate the impact of RAID5 on the system bottleneck. We organized the disk drives in the NAS test bed as RAID0 and RAID5 through out all the tests in this paper.

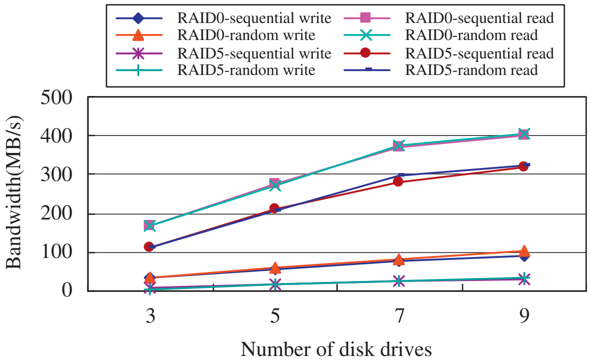

Fig. 3. Raw bandwidth with Xdd benchmark.

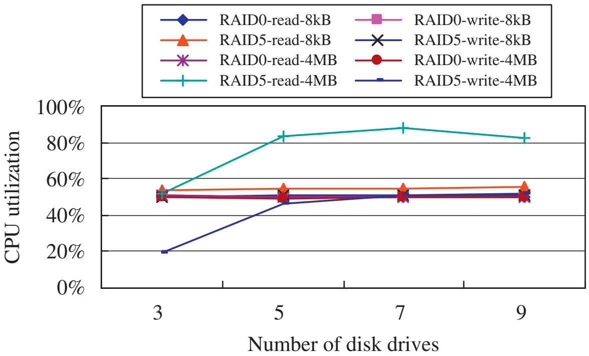

Fig. 5. Average CPU utilization of the Xdd tests.

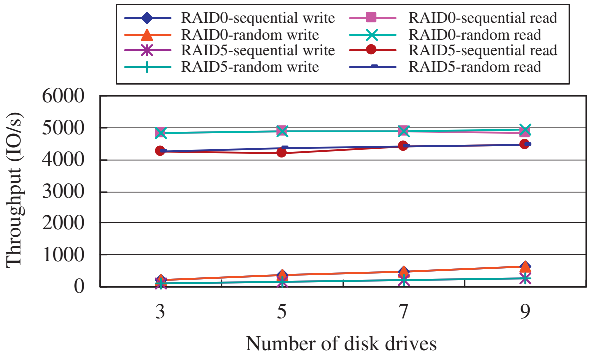

Fig. 4. Raw throughput with Xdd benchmark.

Fig. 3 depicts the raw bandwidth of the logical volume in NAS, which consists of 3,5,7,9 disk drives, respectively. Fig. 3 shows that the bandwidth of random read and sequential read of RAID0 gradually reach a saturation point with the increase in number of disk drives. The maximal bandwidth achieved is 403.12 MB/s, which is 76.35% of the 528 MB/s theoretical bandwidth provided by a 64 bit/66 MHz PCI bus. The measured highest bandwidth is reasonable because the shared PCI bus incurs some overhead such as bus arbitration, interrupt, etc. Fig. 3 also illustrates that the 64 bit/66 MHz PCI bus dominates the performance, which reaches a saturation point with seven disk drives.

Fig. 4 illustrates the throughput of NAS which is composed of 3,5,7,9 disk drives, respectively. It is interesting to observe that the highest throughput achieved is 4842.52 IO/s, which is lower than the throughput of a single disk illustrated in Table 3. Fig. 4 also depicts that the number of disk drives has negligible impact on the throughput. The test results were not expected. This is because the current generation of hardware devices have a higher CPU utilization for smaller block sizes and prove to be a latency and throughput performance bottleneck (Sarkar et al., 2003).

Figs. 3 and 4 both illustrate that the performance including bandwidth and throughput of RAID5 is much lower than that of RAID0. The reason is that the mapping mechanism of RAID5 (between the logical disk and the physical disk drives) is much more complicated than that of RAID0. For example, RAID5 has to calculate the Exclusive OR and store the value to guarantee fault tolerance. The small write penalty (Thomasian et al., 2004) of RAID5 also has a significant performance impact. The small write may require pre-reading old data, pre-reading the corresponding old parity value, writing new data, and writing new parity value. According to Figs. 3 and 4, we can also observe that the read performance is much better than write performance. Because we employed direct I/O to eliminate the performance impact of the file system buffer cache, the write operations use write through policy. However, read operations can adopt aggressive prefetching of the advanced disk drive to boost performance. Both figures show that the aggregate bandwidth and throughput of the logical volume are not improved linearly with the increase in the number of disk drives. The above performance phenomenon is also observed in the tests of the local disk file system and network file system.

The requests that go to or come from disk drives have to travel through the logical volume manager. Undoubtedly, the RAID algorithms implemented in the logical volume manager can incur some processing overhead. We used Vmstat, which adopts a sample time of 2 s to monitor the CPU utilization. The CPU utilization is governed by the amount of data and the path which the data has to go through. The average CPU utilization is illustrated in Fig. 5. It shows that the benchmark never saturates the CPU. Fig. 5 also demonstrates that the highest CPU utilization is incurred by reading 4 MB files when the NAS is configured in RAID5, which confirms the high processing overhead produced by RAID5. Therefore, at this level, we believe that the 64 bit/66 MHz PCI bus dominates the system performance, because the performance eventually reaches a saturation point with the increase in number of disk drives.

3.3. *Disk file system*

In 2000, the performance gap of processors and disk drives has been widened to six orders of magnitude and continues to widen by about 50% every year (Schlosser et al., 2000). Due to the ever increasing performance gap, the performance of a local disk file system is largely governed by the disk behaviour. In this section, we will discuss the impact of a RAID subsystem on the local disk file system of a NAS.

IOzone filesystem benchmark is a general file system benchmark used to generate and measure a variety of file operations. We employed IOzone to measure the bandwidth of write and read operations. We did not test the re-read and re-write, the reason is that the performances are normally higher than read and write due to the impact of the file system buffer cache. The accessed data set in our experiment is 4 GB, file size is 4 MB. PostMark: a new file system benchmark is a file system benchmark designed to emulate small file workloads such as e-mail and net news. It measures the number of transactions per second that a system is capable to support. A transaction is a file creation or deletion, paired with a read or an append (Thereska et al., 2006). The benchmark allows defining the number of files to be created, file size, the ratio of read to write, number of transactions, and the amount of data transfer between the client and the server. The configuration parameters used in our experiment were 50,000 files, 50,000 transactions. The file size is 10–15 KB. All other parameters were left as default.

Disk file system is designed to store and manage files on disk drives. It provides a transparent and easy way to locate and access files by hiding the details of disk drives. In a disk file system, metadata including file name, file size, timestamps, attributes, etc. is additional information associated with files. Metadata is used to facilitate the understanding, use and management of data. The read and write operations at disk file system level have two main differences with the raw read and write operations discussed in Section 3.2. Each write operation causes the disk file system to allocate storage space on the disk to accommodate the data being written. These allocation operations require additional disk drive accesses including creating file with given pathname, traversing pathname, allocating metadata, and directory entry. Each read operation has to locate the blocks of a file on disk drives in terms of metadata, then update the metadata, and finally read the required data. The tests of IOzone and Postmark involve metadata accesses, because they are file system benchmarks.

For a single disk, by using the IOzone benchmark, we obtained a write bandwidth and a read bandwidth of 34.571 and 57.314 MB/s, respectively. We achieved a throughput of 595 transactions/s of a single disk drive with Postmark. Compared with the raw bandwidth and throughput, the throughput and write bandwidth degrades significantly, and the read bandwidth has a slight reduction. The test results confirm the overhead involved in the disk file system and indicate a significant impact of the disk file system on the system throughput.

Figs. 6 and 7 depict the bandwidth and throughput of the local disk file system above a RAID subsystem. Table 4 illustrates the performance improvement with the increase in number of disk drives. Fig. 6 shows that seven disk drives saturate the system bandwidth. The throughput in Fig. 7 displays an obvious growth with the increase in number of disk drives. Fig. 8 illustrates that the CPU in the NAS is not a system bottleneck, because the highest CPU utilization is only 68%. Fig. 6 shows a similar performance trend as that of Fig. 3, we believe that the PCI bus still dominates the system bottleneck as discussed in Section 3.2. However, the highest bandwidth of local disk file system (246.12 MB/s of sequential read with seven disk drives organized as RAID0) is only 61% of the raw bandwidth (403.12 MB/s of random read with nine disk drives configured in RAID0). We believe that the performance degradation mainly attributes to the metadata operation. The reason is that although the sizes of metadata are relatively small compared to the overall size of the system, metadata operations may make up over 50% of all file system operations (Weil et al., 2004).

We employed Bonnie++ to perform the same tests because it can switch on/off the write buffer. We obtained a similar bandwidth as Fig. 6 when the write buffer was switched off. However, when the write buffer was switched on, we achieved 18.65% improvement of the maximal write bandwidth and 45.28% degradation of the maximal read bandwidth. The reason is that within a fixed size of physical memory, write buffer occupies the space of read cache. This measurement shows that a bigger memory could further improve the system performance. However, due to the bottleneck of PCI bus and the overhead of metadata accesses, the achievable performance is limited.

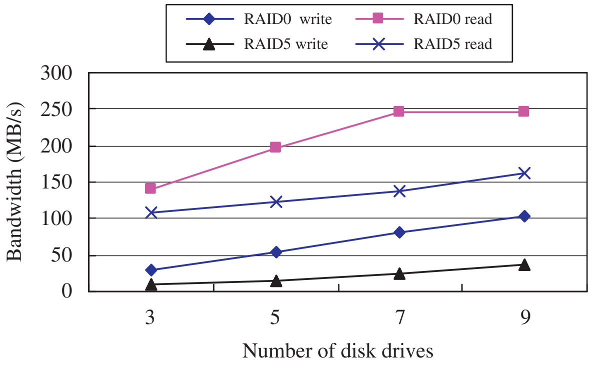

Fig. 6. Disk file system bandwidth with Iozone.

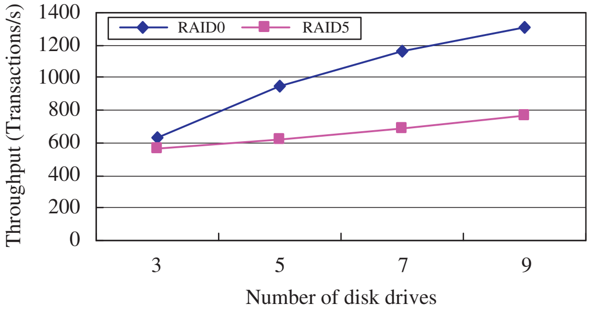

Fig. 7. Disk file system throughput with Postmark.

Table 4
Performance improvement with the increase in number of disk drives.

	3 Disks→5disks (%)	5 Disks→7disks (%)	7 Disks→9disks (%)
Bandwidth			
RAID0-write	77.29	51.69	26.43
RAID0-read	39.51	25.59	−0.47
RAID5-write	80.47	63.29	44.93
RAID5-read	13.18	11.98	18.40
Throughput			
RAID0	49.21	23.22	13.17
RAID5	11.41	11.04	10.81

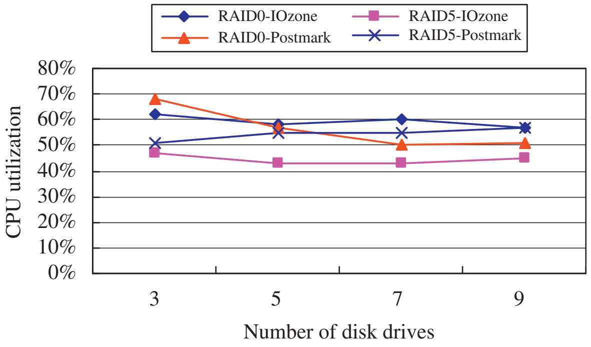

Fig. 8. Average CPU utilization of the disk file system tests.

3.4. Network and network file system

The test environment of the NFS consists of a client machine, a 1000/100 M adaptive switch (HP ProCurve Switch 2824 (J4903A)), and an NAS. The client machine is an IBM Xseries 346, which has a 2.66 GHz Intel Xeon Processor, 1 GB memory, and 1000 M network interface card. Table 1 shows the platform configurations of the NAS. By using the network benchmark Netperf 2.4.3, we achieved a 117.63 MB/s bandwidth of the baseline network performance. The benchmark also shows 7999 transactions/s within one single TCP connection (e.g. database), and 3996 transactions/s when each transaction establishes a new TCP connection (e.g. HTTP). The tests of NFS employed the same benchmarks and the same configuration parameters as that of the disk file system tests. For a single disk drive, we achieved a read bandwidth and a write bandwidth of 59.181 and 16.341 MB/s, respectively. We obtained a throughput of 41 transactions/s.

Figs. 9 and 10 illustrates the bandwidth and throughput of NFS, respectively. As explained in Section 3.2, the performance of read is better than write, and the performance of RAID0 is better than RAID5. According to Fig. 9, the highest bandwidth obtained is 104.87 MB/s, which is read test with 9 disk drives configured in RAID0. The bandwidth is only 42.6% of the highest bandwidth (246.12 MB/s, read test with seven disk drives configured in RAID0) of local disk file system. The reason is that the Gigabit Ethernet provides the maximum bandwidth of 125 MB/s. Our test achieves 83.9% theoretical bandwidth. Therefore, from the bandwidth's point of view, the network is a major system bottleneck.

Table 5 illustrates the performance improvement with the increase in number of disk drives. It shows that five disk drives have saturated the bandwidth. With more disk drives added in the NAS, the performance can only achieve limited improvement due to the network bottleneck. Fig. 11 describes the average CPU utilization of the NAS through the performance tests. Compared with Fig. 8, the CPU utilization is decreased. The reason is that the network bottleneck slows down the data traffic between disk drives and system memory, thus decreasing the CPU utilization. This test further confirms the network bottleneck.

The throughput depicted in Fig. 10 was not expected. The highest throughput is only 4.8% of that of the local disk file system. The reason is that the NFS is located at the NAS side (server side). A hit in the file system cache results in a network hop. The NFS client also employs a cache to hold both the data and the metadata. NFS v3.0 requires clients to check data consistency with the server on cached data and metadata. Therefore, to guarantee the data consistency, a read operation of data or metadata may incur a message exchange with the server even in the event of a cache hit. The write operations in NFS v3.0 support both the write through and the write back policies. The metadata write provides only write through policy (Radkov et al., 2004). PostMark is a metadata intensive file system benchmark due to its operations on a large number of small files. An analysis of the NFS v3 protocol messages exchanged between the server and the client shows that 65% of the messages are metadata related (Radkov et al., 2004). We believe that the available network bandwidth for real data transfer is occupied by a large number of metadata accesses.

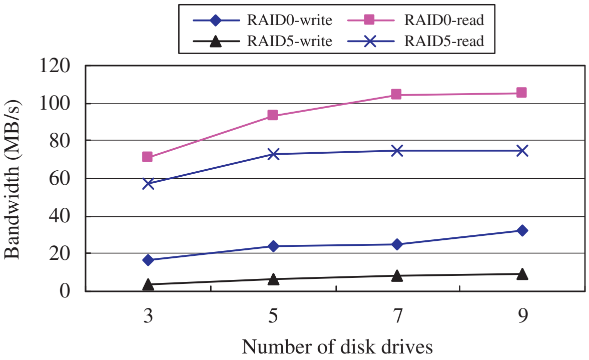

Fig. 9. Network file system bandwidth with Iozone.

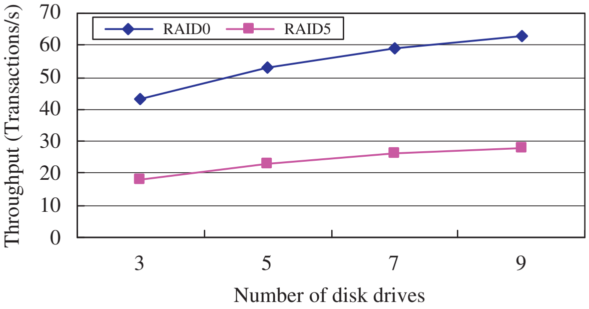

Fig. 10. Network file system throughput with Postmark.

Table 5
Performance improvement with the increase in number of disk drives.

	3 Disks→5disks (%)	5 Disks→7disks (%)	7 Disks→9disks (%)
Bandwidth			
RAID0-write	43.82	6.63	30.23
RAID0-read	30.87	12.5	0.14
RAID5-write	85.44	28.82	17.08
RAID5-read	28.20	3.07	−0.82
Throughput			
RAID0	23.26	11.32	6.78
RAID5	27.78	13.04	7.69

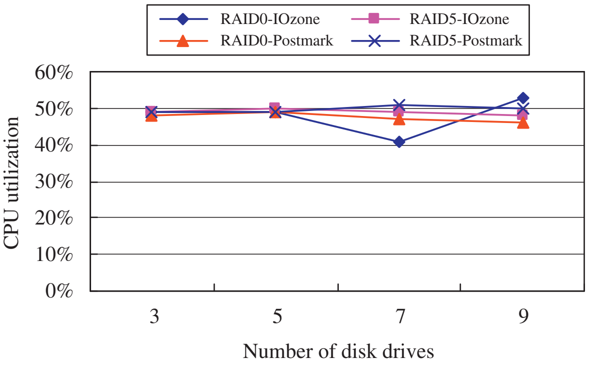

Fig. 11. CPU utilization of the NFS tests.

4. Discussions and implications

NAS is a file server that employs a network file system to provide a file access interface to a network. The NAS normally integrates RAID technology to aggregate the capacity, performance, and reliability of the storage subsystem. As illustrated in Fig. 2, the network stack and the storage stack are two independent stacks, which exchange data by accessing the local disk file system (e.g.ext2/ext3). The architecture incurs a significant overhead. For example, because the read and write operations of the NFS server are implemented by accessing the local disk file system, there is no data sharing between the network stack and the storage stack, which incurs unnecessary data copies. Another important overhead produced by the independent stacks is the multiple times of data decomposition and combination. For instance, the data packages from NFS client have to be merged at the NFS server only for re-decomposition at the logical volume manager in terms of the RAID algorithm, because the data has to be stripped and distributed across multiple disk drives to aggregate the performance and storage capacity.

As discussed in Section 1, many optimization methods that could be applied to improve the NAS performance have been proposed and developed. Basically, it can be classified into three categories in terms of Fig. 2: optimizing the network subsystem (Compaq, 1997; Magoutis et al., 2002; Mogul, 2003; Remote Direct Memory Access), optimizing the storage subsystem such as hardware RAID (Denehy et al., 2004; Deng et al., 2006), and optimizing the communication between the storage stack and the network stack (Brustoloni, 1999).

We believe that the most effective method to alleviate the network bottleneck is increasing the physical network bandwidth or improving the utilization of the network. Two typical technologies can be used to extend the physical network bandwidth. The first one is dividing NAS clients into multiple clusters, each cluster mounts and shares one exported file system through a specific network adapter of NAS. The second one is IP bonding technology, which can combine several network adapters as a virtual one, thus aggregating the physical bandwidth (HA connectivity for servers and mainframes, 2004). However, the first method has to tackle the challenge of load balance among multiple network adapters. The second method requires support from a specific switch. Wang et al. (2007) proposed to employ multiple TCP socket streams to improve the efficiency of data transfer in a Grid environment. The method is very effective to improve the network utilization when large data is transferred in a high latency environment. However, it is not suitable for the NAS because NAS is a file server. Many studies have investigated the features of the files that are located in file servers. Baker et al. (1991) reported that 80% of file accesses in file servers are less than 10 KB. Nine years later, Roselli et al. (2000) found that small files still comprised a large number of file accesses even though the number of accesses to large files had increased since the study in Baker et al. (1991). They reported that the percentage of accessed files that are under 16 KB is 88% in INS trace, 60% in RES trace, 63% in WEB trace, 24% in NT trace, and 86% in Sprite trace. Tanenbaum et al. (2006) investigated the file size distribution on UNIX systems in 1984 and 2005, respectively. Basically, the files are becoming larger. For example, the largest file recorded in 2005 was 2 GB, which is more than 4000 times larger than that in 1984. However, small files are still dominant in the systems. They reported that the files that are smaller than 8 KB is 84.97% in 1984 and 69.96% in 2005, and 99.18% of files in 1984, and 90.84% of files in 2005 are smaller than 64 K. Due to the small files stored in NAS, throughput is the most important performance requirement.

Our tests show that the highest bandwidth of NFS is 42.6% of the local disk file system, and 83.9% of the physical network bandwidth. Unfortunately, the highest throughput of NFS is only 4.8% of that of the local disk file system. The reason is that the Postmark benchmark is designed to emulate small file workloads, thus it is metadata intensive. It has been reported that 65% NFS v3 protocol messages exchanged between the server and the client are metadata related. If the metadata accesses can be reduced, the valid throughput can be improved. Radkov et al. (2004) identified three factors that impact the NFS performance of metadata intensive applications: (1) consistency check related messages, (2) synchronous metadata update messages, and (3) non-aggregated metadata updates. They also proposed some solutions to deal with the impact including using a strongly consistent read-only cache for name and attribute and employing asynchronously metadata update in an aggregated fashion through directory delegation. We conclude that a more efficient network file system could be able to boost the performance of NAS.

Chase et al. (2001) reported that the performance of a high speed network is often limited by the sending and receiving hosts, rather than by the network hardware or the TCP protocol. Therefore, the network performance can be improved by reducing host overheads through a variety of optimizations that are above or below the TCP protocol stack including interrupt coalescing, checksum offloading, and zero copy data movement by page remapping. However, the data flow of a typical NAS is different with that of an application to application communication discussed in Chase et al. (2001). The data flow in NAS does not travel through the boundary of kernel space to the application space, thus avoiding a lot of context switch and data copying (see Fig. 2). Therefore, optimizing the network stack may not be able to alleviate the network bottleneck even though it can decrease the CPU utilization. Sarkar et al. (2003) analyzed the performance behaviours of IP storage by using three approaches including software, TCP Offload Engine (TOE) and Host Bus Adapter (HBA). They reported that the hardware methods do reduce the CPU utilization for large block sizes, the hardware can itself be a performance bottleneck which hurts throughput and latency with small block sizes. The software approach achieves the best bandwidth, though the initiator CPU is saturated by the small block sizes of 0.5–8 KB. In our throughput tests at three levels, the processing power (an Intel(R) xeon (TM) 3.2 GHz (dual core) CPU) never became a system bottleneck. The tests of NFS illustrate the lowest CPU utilization due to the network bottleneck. NAS is mainly used for file sharing rather than computing intensive applications, which require higher processing power. Therefore, the hardware methods such as Compaq (1997), Magoutis et al. (2002), Mogul (2003), and Remote Direct Memory Access are unnecessary for a general NAS architecture, which has a Gigabit network to reduce the CPU utilization. On the other hand, because most of the data accesses in NAS are small file accesses, the hardware methods could have a side effect on the throughput as reported in Sarkar et al. (2003). We believe that an NAS with a Gigabit network is more cost-effective without using some specific hardware. As indicated in Sarkar et al. (2003), the hardware approaches may be well suited for high-end environments such as 10 Gigabit networks. The reason is that the communication overhead is so heavy that current CPUs may not be able to take advantage of the capacity of the network in the software approach.

Our tests indicate that the performance of NAS has lagged far behind the performance of the local storage subsystem due to the involved NFS. The reason is that there is a performance disparity between the network stack and the storage stack. From the bandwidth perspective, our experiments illustrate that the 64 bit/66 MHz PCI bus and the network are the bottlenecks of the storage stack and network stack, respectively. The tests take seven disk drives to saturate the PCI bus and five disk drives to saturate the NFS. It indicates that the network is the overall performance bottleneck of NAS. An important implication of the bottleneck analysis is that the network bottleneck gives a scalability penalty on the storage subsystem of NAS. When the aggregate bandwidth of the RAID subsystem surpasses the network bottleneck, adding more disk drives can only contribute to storage capacity, but not performance. More advanced and expensive disk drives are unnecessary, because they are not cost-effective and they do not contribute to the aggregate performance.

5. Conclusion

Based on a modern NAS, this paper did a comprehensive performance test at different layers including disk drive layer, the logical volume manager layer that employs RAID technology, local disk file system layer, and network file system layer. The test results illustrate that the PCI bus still dominates the performance of the storage subsystem and the Gigabit 1000 Mb network is the major bottleneck of the overall NAS system. Due to the bottleneck of the Gigabit network, the most effective method to alleviate the network bottleneck is increasing the physical network bandwidth or improving the utilization of the network. A more efficient network file system could be able to boost the performance of NAS. As a file server, the processing power of a generic CPU will not be a serious issue of the system performance of an NAS. Hardware methods are unnecessary for a general NAS architecture, which has a Gigabit network to reduce the CPU utilization. On the contrary, the hardware methods could have a side effect on the throughput due to the small file accesses in NAS. Software RAID is more than enough to satisfy the performance requirements of the typical applications on NAS. The increase in number of disk drives in an NAS that has a Gigabit network can only achieve limited performance improvement except for expanding the storage capacity when the network is saturated.

References

Baker M, Hartman J, Kupfer M, Shirriff K, Ousterhout J. Measurements of a distributed file system. ACM symposium on operating systems principles 1991:198–212.

Bonnie++. 〈http://www.coker.com.au/bonnie++/〉.

Bright J, Chandy J. A scalable architecture for clustered network attached storage. In: Proceedings of 20th IEEE/11th NASA Goddard conference on mass storage systems and technologies (MSST 2003), 2003, p. 196–206.

Brustoloni J. Interoperation of copy avoidance in network and file I/O. In: Proceedings of 8th annual joint conference of the IEEE computer and communications societies (INFOCOM 99), 1999, p. 534–542.

Chase J, Gallatin A, Yocum K. End system optimizations for high-speed TCP. IEEE Communications Magazine 2001;39(4):68–74.

Chen Y, Ni L, Yang M. CoStore: a storage cluster architecture using network attached storage devices. In: Proceedings of the ninth international conference on parallel and distributed systems (ICPADS 2002), 2003, p. 301–307.

Chuang J, Sirbu M. Distributed network storage service with quality-of-service guarantees. Journal of Network and Computer Applications 2000;23(3): 163–85.

Compaq, Intel, Microsoft, Virtual interface architecture specification, Version 1.0, December 1997.

Denehy T, Bent J, Popovici F, et al. Deconstructing storage arrays. In: Proceedings of the 11th international conference on Architectural support for programming languages and operating systems (ASPLOS-XI), 2004, p. 59–71.

Deng Y, Wang F, Zhang J, Feng D, Wang F, Hong J. Push the bottleneck of streaming media system from streaming media server to network. International Journal of Image and Graphics 2005;5(4):859–69.

Deng Y, Wang F, Helian N, Feng D, Zhou K. Optimal clustering size of small file access in network attached storage device. Parallel Processing Letters 2006;16(4):501–12.

Deng Y. RISC: a resilient interconnection network for scalable cluster storage systems. Journal of Systems Architecture 2008;54(1-2):70–80.

Deng Y, Wang F. A heterogeneous storage grid enabled by grid service. ACM SIGOPS operating systems review. Special Issue: File and Storage Systems 2007;41(1): 7–13.

Deng Y, Wang F, Helian N, Wu S, Liao C. Dynamic and scalable storage management architecture for grid oriented storage devices. Parallel Computing 2008a;34(1): 17–31.

Deng Y, Wang F, Helian N. EED: energy efficient disk drive architecture. Information Sciences 2008b;178(22):4403–17.

Gibson G, Meter R. Network attached storage architecture. Communications of the ACM 2000;43(11):37–45.

HA connectivity for servers and mainframes: NIC teaming and OSA/OSPF design, Cisco Systems, Inc. 2004.

HDS724040KLSA80 disk specification. 〈http://www.hitachigst.com/hdd/support/7k400/7k400.htm〉.

IOzone filesystem benchmark. 〈http://www.iozone.org〉.

Katsurashima W, Yamakawa S, Torii T, Ishikawa J, et al. NAS switch: a novel CIFS server virtualization. In: Proceedings of 20th IEEE/11th NASA Goddard conference on mass storage systems and technologies (MSST 2003), 2003, p. 82–86.

Logical volume manager. 〈http://sources.redhat.com/lvm2/〉.

Magoutis K, Addetia S, Fedorova A, et al. Structure and performance of the direct access file system. In: Proceedings of 2002 USENIX annual technical conference, 2002, p. 1–14.

Mesnier M, Ganger G, Riedel E. Object-based storage. IEEE Communications Magazine 2003;41(8):84–90.

Mogul J. TCP offload is a dumb idea whose time has come. In: Proceedings of the 9th conference on hot topics in operating systems, 2003, p. 25–30.

Netperf. 〈http://www.netperf.org/netperf/〉.

Patterson D, Gibson G, Katz R. A case for redundant arrays of inexpensive disks (RAID). In: Proceedings of the ACM SIGMOD international conference on management of data, 1988, p. 109–116.

Pawlowski B, Juszczak C, Staubach P, et al. NFS version 3 design and implementation. In: Proceedings of the summer USENIX technical conference, June 1994, p. 137–152.

PostMark: a new file system benchmark. NetApp technical report TR-3022.

Pugh E. Storage hierarchies: gaps, cliffs, and trends. IEEE Transactions on Magnetics 1971;7(4):810–4.

Qin X, Jiang H, Zhu Y, Swanson D. Improving the performance of I/O-intensive applications on clusters of workstations. Cluster Computing: The Journal of Networks, Software Tools and Applications 2005a;8(4):297–311.

Qin X, Jiang H, Zhu Y, Swanson D. A feedback control mechanism for balancing I/O- and memory-intensive applications on clusters. Scalable Computing: Practice and Experience 2005b;6(4):95–107.

Radkov P, Yin L, Goyal P, Sarkar P, Shenoy P. A performance comparison of NFS and iSCSI for IP-networked storage. In: Proceedings of the 3rd USENIX conference on file and storage technologies (FAST2004), 2004, p. 101–114.

Remote Direct Memory Access. 〈http://www.rdmaconsortium.org/home/〉.

Riedel E. Storage systems: not just a bunch of disks anymore. ACM Queue 2003; 1(4):32–41.

Roselli D, Lorch J, and Anderson T. A comparison of file systems workloads. In: Proceedings of USENIX technical annual conference, 2000, p. 41–54.

Sarkar P, Uttamchandani S, Voruganti K. Storage over IP: when does hardware support help? In: Proceedings of the 2nd USENIX conference on file and storage technologies (FAST2003), 2003, p. 231–244.

Schlosser S, Griffin J, Nagle D, Ganger G. Designing computer systems with MEMS-based storage. In: Proceedings of the 9th international conference on architectural support for programming languages and operating systems (ASPLOS), 2000, p. 1–12.

Sohan R, Hand S. A user-level approach to network attached storage. In: Proceedings of the IEEE conference on local computer networks, 2005, p. 108–114.

Tanenbaum A, Herder J, Bos H. File size distribution on UNIX systems: then and now. ACM SIGOPS Operating Systems Review 2006;40(1):100–4.

Thereska E, Abd-El-Malek M, Wylie J, Narayanan D, Ganger G. Informed data distribution selection in a self-predicting storage system. In: Proceedings of IEEE international conference on autonomic computing, 2006, p. 187–198.

Thomasian A, Han C, Fu G, Liu C. A performance evaluation tool for RAID disk arrays. In: Proceedings of the first international conference on the quantitative evaluation of systems (QEST'04), 2004, p. 8–17.

Varki E, Merchant A, Xu J, Qiu X. Issues and challenges in the performance analysis of real disk arrays. IEEE Transactions on Parallel and Distributed Systems 2004;15(6):559–74.

Vmstat. 〈http://www.linuxcommand.org/man_pages/vmstat8.html〉.

Wang F, Helian N, Wu S, Deng Y, Khare V, Parker M. GridJet: an underlying data-transporting protocol for accelerating Web communications. Computer Networks 2007;51(16):4561–73.

Weil S, Pollack K, Brandt S, Miller E. Dynamic metadata management for petabyte-scale file systems. In: Proceedings of the ACM/IEEE SC2004 conference on supercomputing (SC2004), 2004.

XDD6.5. 〈http://www.ioperformance.com/〉.

Yasuda Y, Kawarnoto S, Ebata A, Okitsu J, Higuchi T. Concept and evaluation of X-NAS: a highly scalable NAS system. In: Proceedings of 20th IEEE/11th NASA Goddard conference on mass storage systems and technologies (MSST 2003), 2003, p. 219–227.