Analyzing the Feasibility of Building a New Mass Storage System on Distributed Resources[*]

H. Howie Huang[†‡], John F. Karpovich, and Andrew S. Grimshaw

Department of Computer Science, University of Virginia
151 Engineer's Way, P.O. Box 400740
Charlottesville, Virginia 22904, USA

SUMMARY

The average PC now contains a large and increasing amount of storage with an ever greater amount left unused. We believe there is an opportunity for organizations to harness the vast unused storage capacity on their PCs to create a very large, low cost, shared storage system. What is needed is the proper storage system architecture and software to exploit and manage the unused portions of existing PC storage devices across an organization and make it reliably accessible to users and applications. We call our vision of such a storage system Storage@desk (SD). This paper describes our first step towards the realization of Storage@desk – a study of machine and storage characteristics and usage in a model organization. We studied 729 PCs in an academic institution for 91 days, monitoring the configuration, load and usage of the major machine subsystems, i.e. disk, memory, CPU and network. To further analyze the availability characteristics of storage in an SD system, we performed a trace-driven simulation of some basic storage allocation strategies. This paper presents the results of our data collection efforts, our analysis of the data, our simulation results, and our conclusion that a Storage@desk system is indeed feasible and holds promise as a cost effective way to create massive storage systems.

KEY WORDS: storage system; distributed computing

1. INTRODUCTION

The cost of stable storage – particularly hard drives – has been dropping rapidly in terms of dollars per gigabyte and is predicted to continue to do so into the foreseeable future. The average PC now contains 100GB or more of storage capacity and will likely continue to grow. At the same time, demand for storage is also increasing rapidly. A study by IDC [1] concluded that demand for storage capacity has been growing annually by approximately 60%, and reached over 450 petabytes in Q2 2005. Anecdotally, our experience matches the results of the IDC survey. We see that users require more storage space for their applications and data and we have

[*] A preliminary version of this paper has been published in the *Proceedings of the First International Workshop on Virtualization Technology in Distributed Computing*, held in conjunction with SC06, November 17, 2006, Tampa, Florida.

[†] Correspondence to: H. Howie Huang, Department of Computer Science, University of Virginia, 151 Engineer's Way, P.O. Box 400740, Charlottesville, Virginia 22904, USA

[‡] E-mail: huang@cs.virginia.edu

also observed a sharp rise in the number of projects, both scientific and commercial, that require very large storage capacities, whether for short periods of time or on a more permanent basis. However, even though storage capacity and demand are both increasing together, storage capacity and demand are often not well matched in practice. PCs are often purchased based on the needs of an average user or market forces, while demand for storage is not at all uniform across users or projects – many require relatively little storage and others require a great deal of storage. Therefore, a significant number of PCs in an organization now contain large unused amounts of storage, even as some users or projects have unmet demand.

Unfortunately, efficiently harnessing excess storage capacity is not easy for a number of reasons. First, users demand that storage be as close to 100% reliable as possible, i.e. that there is nearly zero chance of permanent data loss. The demand for reliability forces system administrators to employ strategies to avoid data loss, most often using active data replication, fault tolerant hardware, periodic bulk data backups or some combination of these. A serious drawback to these solutions is that they are much more costly to buy, deploy and maintain than commodity storage.

Second, tools for sharing storage capacity among machines are either non-existent, cumbersome to use, or inadequate in providing reliability, availability, performance, security or load balancing storage demand with supply. Third, PCs typically have undesirable QoS properties, such as low uptime, variable performance due to locally induced load, slower disks, etc. Finally, and most importantly, there are no mechanisms for aggregating and centrally managing PC disk resources with the objective of providing specific QoS properties.

So, despite having larger and cheaper commodity devices on PCs, there still exists unmet demand for storage and inefficient use of storage infrastructure and budget. We believe that there is an opportunity to harness the unused resources on commodity PCs to create a very large virtual storage system at low cost for an organization. Such a virtual storage system – we call our vision Storage@desk (SD) – would manage exploiting the unused portions of existing PC storage devices and making it reliably and transparently accessible to users throughout the organization.

Before we can fully design and build such a virtual storage system, we must understand the PC environment and usage within real organizations. To do this we conducted a study of the characteristics of machine and storage resources in a model organization. The goals of the study were to 1) determine the feasibility of Storage@desk – does a real organization have the characteristics necessary to support a worth while distributed virtual storage system; and 2) provide data on the QoS properties of resources in a real organization – e.g. machine reliability, CPU and disk load, etc. – as input into the design of Storage@desk and our modeling of how to achieve specific QoS levels.

The remainder of the paper is organized as follows. Section 2 shows our design of Storage@desk. Section 3 describes the goals of the feasibility study. Section 4 presents the design and implementation of our study, its raw results, and our analysis. Section 5 discusses the simulation results. Section 6 examines related work and Section 7 presents our conclusion.

2. STORAGE@DESK OVERVIEW

Storage@desk aims to exploit and manage the unused portions of existing PC storage devices across an organization and make it reliably accessible to users and applications. To this end, the

Storage@desk system must achieve (at least) seven goals in order to be useful as a real world mass storage alternative.

1. *No Data Loss*. The system must ensure with a high probability that data will not be lost permanently. The standard for data loss would be that expected from a commodity RAID (Redundant Array of Independent Disks) storage solution.
2. *High Availability*. Data must be available to users when they want it with a reasonably high probability, though some occasional temporary outages are acceptable. Given that the underlying storage for the system resides on individual and likely unreliable PCs, this is one of the key challenges and one where we look to our study data for answers as to how much volatility is expected in a real system.
3. *High performance*. The system should not perform significantly worse than the current usual alternatives – notably NFS.
4. *Scalability*. The system must scale to large numbers of clients, large numbers of storage "nodes", large aggregate storage spaces, etc.
5. *Cost-efficiency*. Since there are existing solutions to buy large, reliable storage, the system must be inexpensive in hardware, software and maintenance.
6. *Flexible and Configurable*. The system must provide configuration options to tailor the system to match local policies (of disk use, CPU use, etc) and needs.
7. *Secure*. The system must be able to match the confidentiality, data integrity and authorization standards users expect. This is particularly challenging given that data will be stored on the remote machines "owned" by other users.

2.1 Architecture

The basic architecture of the Storage@desk system is driven by our goals to harness unused local disk storage on existing machines to provide a very large storage system that can be easily incorporated into existing storage systems. We have designed Storage@desk to be a block data service, organizing blocks into virtual volumes. Clients attach a Storage@desk virtual volume to their local machine much like they would attach a new local storage drive and access the resources contained in a SD system via the Internet Small Computer Systems Interface (iSCSI) protocol [2]. iSCSI is a transport protocol standard proposed by IETF which carries SCSI commands and data between components over a TCP/IP network. Because of the ubiquity of SCSI devices and drivers written to use them, exporting an iSCSI interface will make it easy for higher level services to target SD virtual volumes, exposing the large number of distributed storage resources encompassed by an SD system to client machines. Figure 1 shows a simplified version of Storage@desk architecture. Please see [3] for a more complete discussion.

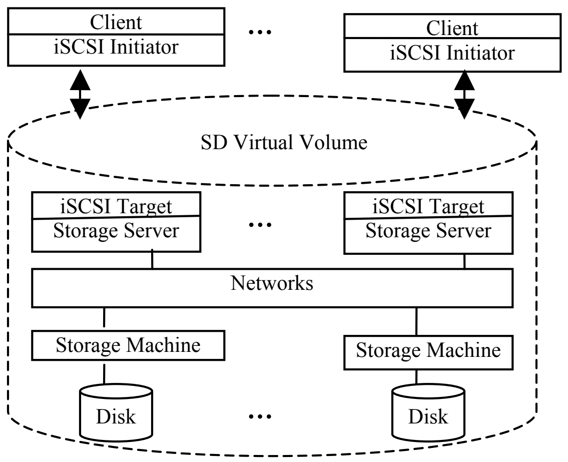


Figure 1. Storage@desk architecture.

As a demonstration of the Storage@desk system at work, we will now follow two representative requests from client to storage machine: read and write.

When a client application attempts to read a block of data from a Storage@desk configured volume, the iSCSI driver on the client's machine receives this request and, based on local configuration, directs the call to the correct storage server. After the storage server receives the request, it locates the corresponding block in the volume/block metadata. For the sake of completeness, we will assume the case where the cache on the server contains a stale (invalid) copy of the metadata. Since the server has a cached copy, it immediately calls through to the backend storage machine. However, the storage machine detects that the block in question is no longer hosted in its own environment and faults the storage server.

Upon receiving the fault from the storage machine, the storage server replaces the cached copy of the metadata in question with new up-to-date information on the volume. Once again, the original request for the volume block propagates to a storage machine (this time, the correct one). The storage machine recognizes the block in question and returns that block of data to the requesting storage server, which in turn passes the result back to the issuing client.

The write request example is similar in most details and in fact differs only in that consistency becomes an issue. Because the Storage@desk system utilizes block replication to achieve many of its QoS policies, the most common case will be that the target block for the write operation resides on more than one storage machine. This situation means that the storage server may have to implement some sort of distributed transaction algorithm (even though this is not always necessary as QoS policies may indicate that a window of incoherence is acceptable). Aside from this one minor difference, however, the operation proceeds just as it did for the read case.

The challenging part to developing SD lies in the designing an internal architecture and developing the proper algorithms to manage the underlying storage while providing certain levels of QoS and reliability to clients.

2.2 Use cases

Four scenarios are illustrative of how we imagine Storage@desk being used in large organizations: to provision persistent user storage, as a dynamic storage pool, with HPC clusters, and as a mechanism for off-site storage. One can easily imagine other uses as well.

Provisioning persistent user storage: Universities and other large enterprises often give out "individual" allocations to users that must be accessible from anywhere in the organization, and which are backed-up on a regular basis. Users expect that the data is highly persistent. Storage@desk can provide transparent access to user data from anywhere in the organization – and, by properly configuring QoS requirements, can be highly available and persistent.

Dynamic storage pool: We have observed that storage requirements vary by discipline – both in terms of amount (terabytes) and quality (temporal and reliability). High-energy physics projects for example often use huge volumes of data for short periods. When they need the storage, they need it! When they don't, it often sits relatively idle. Storage@desk can be used as a storage buffer (pool) to accommodate very large, temporary, storage demands.

HPC clusters: It is now common for organizations to have several clusters in house. For any given cluster, the total amount of spinning storage available (either on the nodes or on a file server) may be less than the sum of all of the projects that use the cluster. Using Storage@desk, administrators allocate storage volumes to each project where the actual data resides on desktops and other machines elsewhere within the organization. QoS hints allow for the data to migrate "closer" to the cluster (perhaps onto the cluster) before run-time – and migrate off the cluster when the jobs that use the data complete. Similarly, if there are multiple clusters and a meta-scheduler that places jobs on different clusters, Storage@desk can migrate the data "close" to the computation. Note that even if the data is not migrated, it can still be accessed in a transparent manner by the application (albeit possibly at a lower bandwidth). Further, in the case of read-only access, each node in the cluster can mount a volume individually, thus bypassing a central file system bottleneck.

Off-site storage: The Storage@desk architecture does not require that the storage servers be "local" to an organization. Administrators can configure a Storage@desk system to use storage from other organizations – such as other institutions or service providers connected by a metropolitan area network or wide area network. The ability to move off-site volumes that are infrequently accessed or have weak QoS requirements allows Storage@desk to effectively deal with temporary surges in demand.

3. STORAGE@DESK FEASIBILITY

Successful development of a mass storage system comprised of local machine disks relies on the characteristics of the existing IT infrastructure and on the usage patterns of its components. We need to know that real environments match our theorized characteristics, in particular whether there is enough unused storage capacity to make a Storage@desk system worth while and whether there is sufficient excess computing, storage, memory and network capacity to carry the extra loads required to manage the system. In addition, we need to gather data about the reliability and availability properties of resources in the system in order to determine what reliability and availability QoS levels can reasonably be supported.

3.1 Is there sufficient "available" unused disk storage in a typical organization?

We need to know if there is enough storage space in a typical real environment to justify proceeding with the development of the Storage@desk and to determine its potential benefits. We have posited from deduction and anecdotal evidence that there is an enormous unused resource available in a typical organization – but is this true? To provide a quick high level answer we will determine the average unused disk space on each machine and the aggregate unused disk space.

Beyond raw unused disk space, we need to know how unused disk space changes over time on individual machines. Volatility is crucial because it determines which allocation strategies are realistic (e.g. static vs. dynamic) and how much space is really available for use by a Storage@desk system. If disk usage fluctuates significantly on machines, then the system must employ either a very conservative static allocation scheme – i.e. allocate only a small percentage of average unused space - or a dynamic scheme that quickly adapts to rapid changes (and will likely have high overhead). Conversely, if unused disk space is relatively stable, the system can employ more aggressive static allocation strategies or various dynamic strategies (with the expectation of more modest overhead rates).

We define MAX_{unused} as maximum unused space, MIN_{unused} as minimum unused space, and Δ_{unused} as $(MAX_{unused} - MIN_{unused})$. Furthermore, we define $Volatility_{unused}$ as

$$Volatility_{unused} = (MAX_{unused} - MIN_{unused}) / MAX_{unused} = \Delta_{unused} / MAX_{unused}$$

By studying these metrics we hope to gain insight into the true amount of space available to an SD system.

3.2. Are there sufficient resources in the current infrastructure to handle the extra load of running a Storage@desk system?

We need to know whether there is sufficient CPU, memory, disk I/O, and network resources available to run SD software to manage the system. To study this we need to collect and analyze CPU capabilities and load, disk load, memory size and availability, and network capabilities and load for each machine.

What resources do we expect to need on local machines to construct Storage@desk? We expect SD to consume a modest amount of memory to run daemons for accessing data – but probably no more than 20 or 30 MB worth. SD daemons will also require CPU, disk I/O, and network resources to handle management and data access requests. These requirements are harder to quantify as they will depend on many factors, including the volume of requests, the capabilities of the local CPU, disk I/O and network resources themselves, and such internal details as whether there is hardware support for part of the TCP protocol stack. Nevertheless, collecting and analyzing load data for local resources is the first step in predicting whether machines have the resources necessary to support the Storage@desk system and provide at least anecdotal evidence of local resources ability to support additional component loads.

3.3 What are the failure characteristics of current resources?

Resource unavailability will be a normal part of operating on top of distributed PCs. Machines will fail, lose power, be rebooted or shutdown or lose connectivity to other machines. Since our goal is to create a system that can achieve very low data loss rates and maintain high availability, Storage@desk systems will have to employ strategies to mask failures – including replication and caching. In order to determine what levels of replication are needed to support various levels of availability and reliability, we need data on the fault characteristics of system components. For availability, we need to know how often, when, and for how long necessary components are temporarily unavailable. For reliability, we need to know how often necessary resources become permanently unavailable.

From raw data we can calculate such statistics – such as Mean Time to Failure (MTTF) and Mean Time to Recovery (MTTR) to provide insight into the average behavior of machines and the system as a whole. Using the raw data we can also look at availability of machines at any given time and detect patterns and correlated failures.

4. STUDY: IMPLEMENTATION AND RESULTS

We developed a light weight monitoring tool to gather machine statistics every 5 minutes and report them to a central database. Each snapshot contains a wide range of statistics including CPU speed and load; total, used, and available disk space; disk I/O load; used and available memory; machine uptime, etc. The monitor was designed to be very reliable – to avoid missing data – and to have minimal impact on the host machine – to avoid skewing results.

Our target environment is a collection of laboratory and classroom PCs at the University of Virginia. This was done for several reasons. First, we believe that the University's PC environment is typical of many other university environments and that it is a good candidate for the type of mass storage system that we believe we can construct. Second, the environment is convenient for us to access – being both local to our project team and having a small number of central administration points from which to distribute the monitoring software. Third, having the monitors and the statistics database both within the same organization models the most likely deployment pattern and will capture some of the networking failures we'd experience.

The study spanned 729 Windows PCs in 28 buildings, 75 rooms, and several departments operated by two administrative domains: 668 by the Department of Information Technology & Communication (ITC) and 61 by the Department of Computer Science (CS). We have been monitoring these resources continuously since September 2005 and plan to release the database to the public in the future. In this paper we present results from a three-month period from September 15, 2005 to December 15, 2005.

Before presenting our results, it is important to discuss two environmental factors. First, ITC reboots its machines daily at 4:30 AM. This means that most of the 668 ITC machines go through the same pattern at 4:30 AM: increased activity as it shuts down, then a period of inaccessibility, followed by above average activity as it restarts. Second, outages of the database server or one of its components (e.g. the web server, file server, etc.) are reflected in the raw study data. We identified several such outages by searching the data for periods where no reports were received at all.

Both factors significantly affect availability statistics for the study (e.g. MTTF, availability) in a negative – and we believe an unnecessarily pessimistic – way. The nightly reboots are a scheduled system downtime when the system is not supposed to be available. In practice, most guarantees of availability consider scheduled maintenances as exceptions and exclude planned downtimes. Furthermore, this nightly maintenance is a policy decision that can be modified (e.g. by staggering the timing or decreasing the frequency) or reversed if there were adequate incentive (such as maintaining an SD system). On the other hand, database server downtimes reflect a failure in the data collection process, not of the resources themselves. Including such database outages presents an inaccurate and overly negative picture of the availability of the underlying resources.

For these reasons, we removed the time slots effected by reboots and database outages from the study data in order to provide a more accurate and realistic picture of expected steady-state availability. Specifically, we removed the daily 4:30 AM time slot and the time slots with zero reports (and their nearest neighbor time slots). We used this adjusted data as input in Section 5 and 6. To show how this adjustment changed the data, Figure 2 shows the number of available machines during each 5 minute time slot before and after eliminating downtimes. We will revisit availability data in Section 4.4.

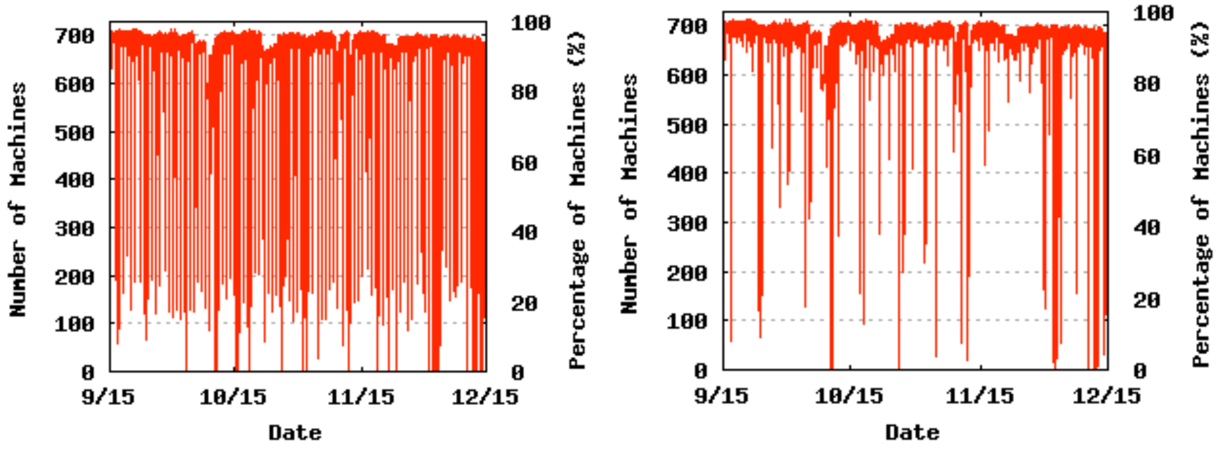


Figure 2. Number of available machines (a) before eliminating downtimes and (b) after eliminating downtimes.

4.1. Resources

Figures 3, 4 and 5 show histograms of CPU speed, disk size, and memory size of the machines in our study. In terms of CPU, disk space and memory, the 729 machines can be described - with some exceptions - by three groupings. The older ITC machines (~20%) have 1.8 – 2.4 GHz processors, 40 GB hard drives and 512 MB memory; the newer ITC machines (~70%) have 2.8 GHz processors, 80 GB drives and 1 GB memory; and the CS laboratory machines (~10%) have 3.2 GHz processors, 160 GB drives and 2 GB memory. Overall, the CPUs are all relatively modern and are dominated by the large number of 2.8 GHz machines (the average is about 2.6 GHz). For all but a few machines (~7%), memory sizes are at or above 512 MB, averaging 960 MB. Disk size averaged 70 GB ranging from a relatively modest 40 GB on 30% of the machines to 160 GB on 8% of the machines, with the majority in the middle at 80 GB.

These configurations fit the mold of middle-to-upper tier desktop machines at the time they were purchased, spanning purchases over the past few years. This seems on the surface to be a representative mix of machines for many large organizations.

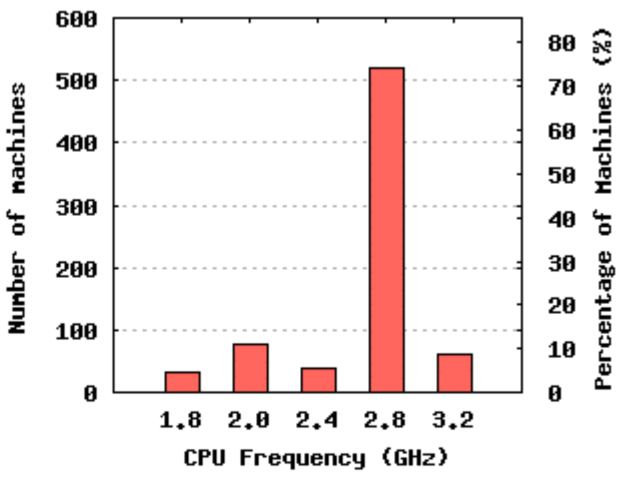


Figure 3. Distribution of CPU frequencies.

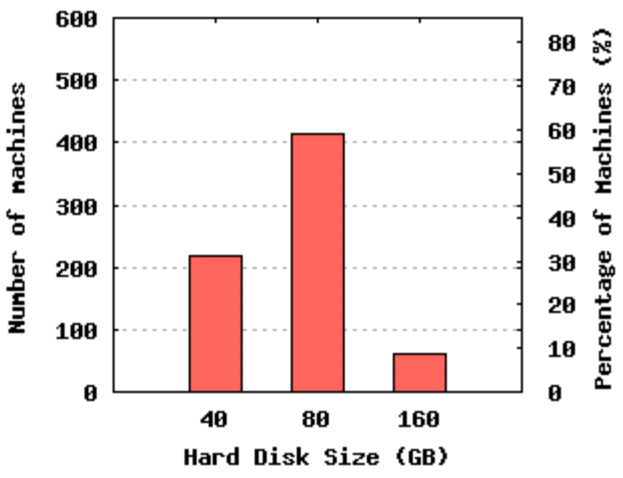


Figure 4. Distribution of hard disk sizes.

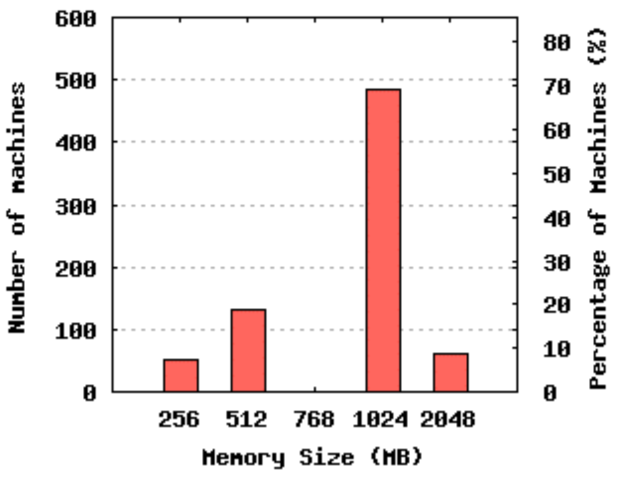


Figure 5. Distribution of memory sizes.

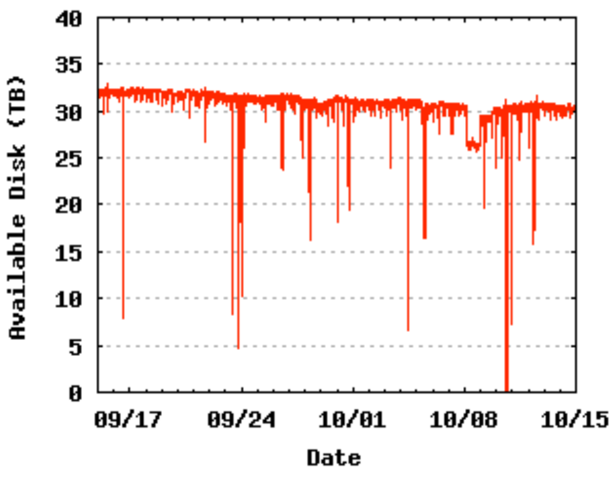
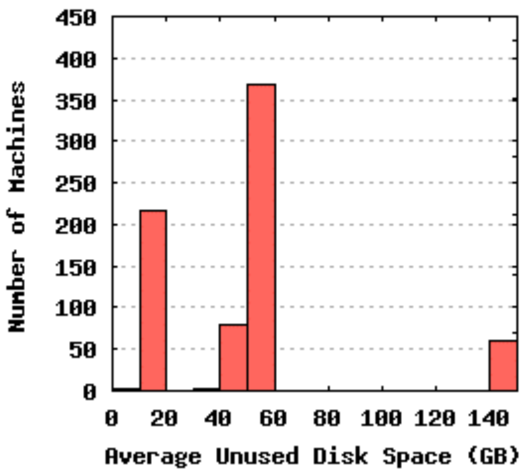


Figure 6. Aggregate unused disk space.

4.2. Disk usage and volatility

The first question that we proposed is how much unused disk space is really available. Figure 6 shows the aggregate amount of unused disk space during the first 30 days of our study – only including space from machines that reported in during the time slot. Therefore it is affected by both changes in the space used on each machine and by the availability of machines. The downward "spikes" are consistent with those in Figure 2. There is a slow linear downward trend in the available disk space. A few possibilities may have caused this trend. First, some machines failed and never were able to recover from the failures. Second, some machines were taken down for maintenance or upgrade for an extended period longer than our observation window. Third, disk spaces were gradually filled as applications were installed and users left temporary data behind. This downward trend should not be alarming, since Storage@desk intends not to take all available disk space on every machine. Rather, Storage@desk utilizes a specific percentage of disk space allowed by local users and system administrators. In conclusion, except for periods when many machines are down, on average there is very consistently 31TB of aggregate free disk space on currently available machines.

Figure 7 groups machines by the average amount of disk space they had available during the study period in 10 GB increments. As expected, the distribution of average unused disk space closely matches how much raw disk space exists on the machines. The average unused storage per machine is approximately 46 GB, skewed by the large unused space on the 61 CS machines. On average, 62% of a machine's raw disk capacity is unused, which is consistent with results of other disk usage studies [4, 5] which have reported storage utilization rates of about 50%. Figure 8 shows the minimum space available on a machine during the study period – providing a more realistic assessment of how much space a mass storage system can safely use on a machine. The results are nearly the same – the aggregate of the minimum available disk space is 32.8 TB with an average of 45 GB per machine (Figure 8 and our MIN_{unused} data do not factor in availability, while the results in Figures 6 and 7 and average unused space calculation include availability).



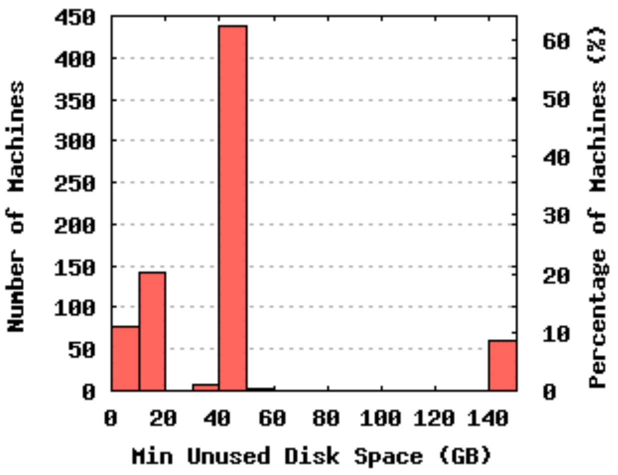


Figure 7. Average available unused disk.

Figure 8. Distribution of minimum unused disk.

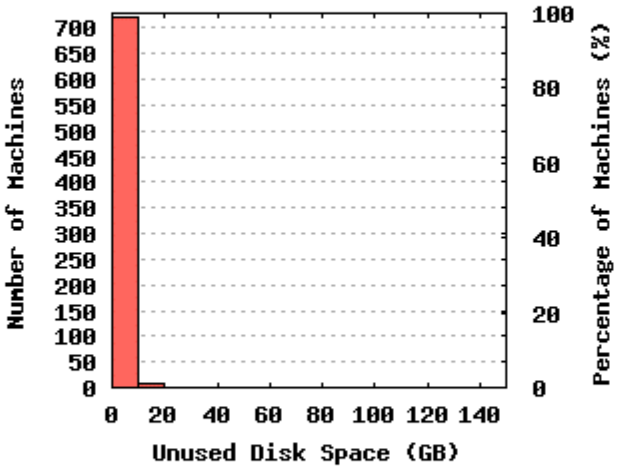


Figure 9. Distribution of Δ_{unused}.

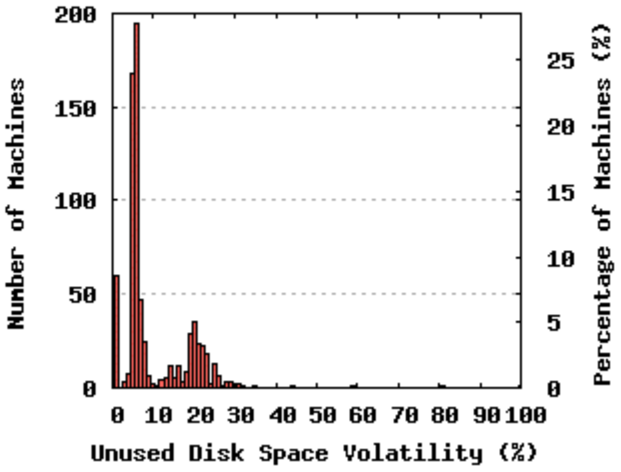


Figure 10. Distribution of $\text{Volatility}_{\text{unused}}$.

To look at the volatility of unused disk space we studied two additional metrics, Δ_{unused} and $Volatility_{unused}$. Figures 9 and 10 show the distributions for these metrics across the study machines, providing an absolute and relative measure of volatility. Both metrics show that the vast majority of machines changed their disk usage relatively little over the study period. Over the month of the study, Δ_{unused} was less than 10 GB for 721 (99%) machines and less than 20 GB for all machines, while $Volatility_{unused}$ was less than 20% for 629 (86%) machines. The only machines that registered high $Volatility_{unused}$ did so because they had little unused disk space. The low volatility bodes well for being able to safely use a large percentage of unused disk space without impacting local users.

Our conclusion is that the average availability of 31TB of unused space coupled with low volatility indicates that there is potential for Storage@desk to harness large amounts of storage in such an environment. If we further factor in that unused disk space is trending upwards and that the university has many additional machines not included in the study the potential space available is likely very significant.

4.3. System loads

To answer our second question – does the infrastructure have the capacity to support the Storage@desk system – we studied CPU load, disk I/O load, memory usage and local machine network load.

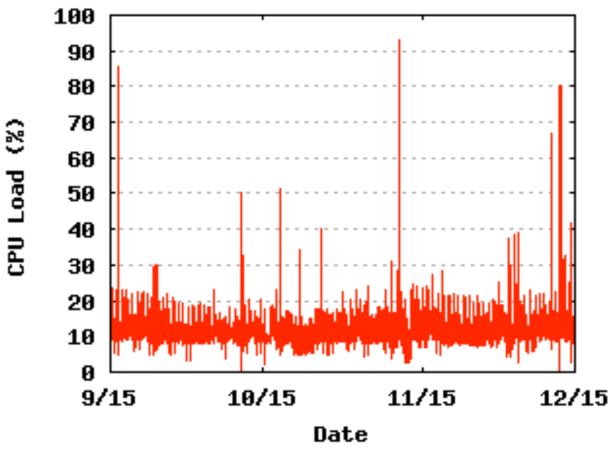


Figure 11. Average CPU load / machine.

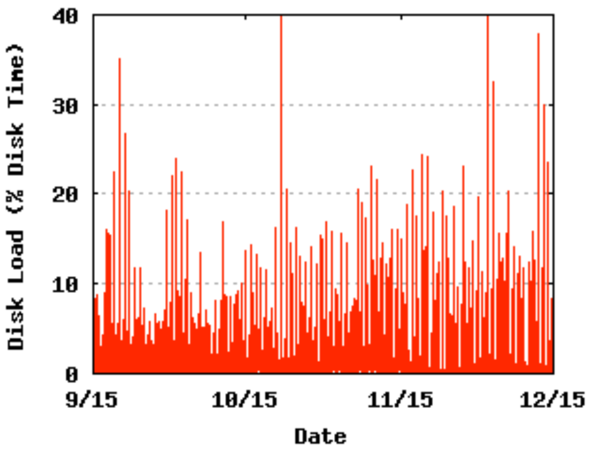


Figure 12. Average disk I/O load / machine.

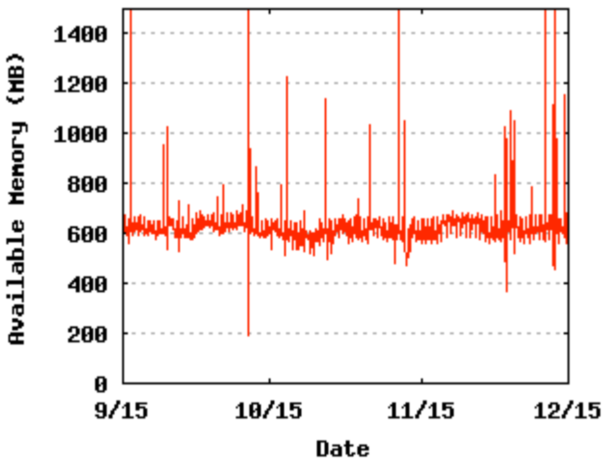


Figure 13. Average memory available / machine.

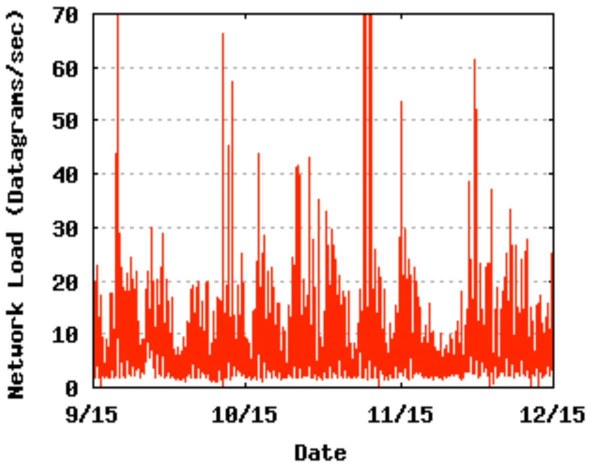


Figure 14. Average network load / machine.

As Figure 11 shows, average CPU in use hovered between 7% and 18%, with a mean of 10.28% and standard deviation of 2.80%. These are fairly modest levels of average CPU usage and volatility. While not conclusive, it does appear that there is excess CPU power available to support an SD system.

The story is similar for disk I/O load and memory available as shown in Figure 12 and 13, respectively. The disk load generally stayed low – in the range of 0.5% – 6%, with an average of 0.68% and a standard deviation of 1.42%. This is what you would expect – some bursts of higher activity interspersed between stretches of lower activity. Available memory averaged between 550 and 650 MB per machine. It appears that the study machines' disk and memory subsystems are not under stress on average and that they likely have additional capacity to support significantly more I/O operations and memory usage. The story for network load shown in Figure 14 also looks reasonably good – the average number of datagrams/sec stays mostly in the range of 3 – 20 datagrams/s with an average of 6.94 datagrams/s and a standard deviation of 5.61. If we assume all packets constitue the largest datagram size of 1500 bytes for Ethernet, then the average load is approximately 84 Kbps on average and 276 Kbps at two standard deviations out. With most network links at or above 100 Mbps, the average local bandwidth used is tiny compared to that available. The conclusion we draw from this limited information is that individual machines likely can support substantial extra network load. Due to the lack of a comprehensive look at the networking infrastructure we draw no conclusion about the capacity of the network backbone at this time. Such a study will be conducted at a later time.

4.4. System reliability and availability

Referring back to Figure 2, we see that there were approximately 700 machines available at most times, with some notable periods of significant correlated failures. We have identified some correlated failures, such as the extended dip in available machines from October 8–10, which was due to a scheduled power outage in several labs for physical plant maintenance. The remaining drops were likely caused by network or undetected database outages. Identifying that correlated failures occur is an important finding as it poses a real challenge to Storage@desk maintaining high availability.

Due to the manner in which we collect data, it is difficult to precisely pinpoint the timing of failures. We determine that a machine was unavailable by detecting that no report was recorded for a 5 minute interval. We define the length of an "up" period – the time-to-failure (TTF) – as the time elapsed between misses. Further, we define the length of a "down" period – the time-to-recovery (TTR) – as the difference between receipt timestamps of the last report before and the first report after a miss.

Figure 15 presents the distribution of availability rates for machines, where availability is defined as MTTF/(MTTF + MMTR). The average availability was 93.99%, with 82.17% of machines with greater than 95% availability and 26.89% greater than 99%. MTTF was low as expected: 17.70% of machines had a MTTF of 6 hours or less and 90.95% had a MTTF of 12 hours or less. On the other hand, MTTR was fairly short: 29.63% had a MTTR of less than 10 minutes, 77.23% had a MTTR of 30 minutes or less and 88.20% had a MTTR of less than 1 hour.

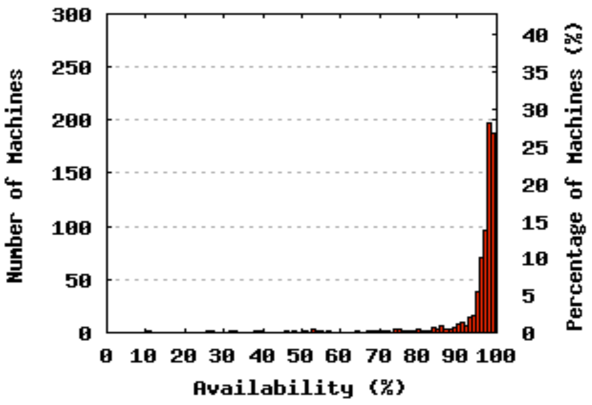


Figure 15. Distribution of machine availability.

Looking at long term reliability, several machines went down for significant periods of time – spanning days or weeks – including 10 machines that were down for at least the last two weeks of the study period. Any long term machine unavailability is cause for concern about data loss and must be dealt with by the system. If we assume that long term failures are independent, the simple strategy of keeping data on 3 unique randomly selected machines out of 729 yields an expected data loss rate of ($\binom{729}{3}\left(\frac{10}{729}\right)^3\left(1-\frac{10}{729}\right)^{726}$) = 0.73% for 10 simultaneous failures. This rate is significant and too high for most applications. So, SD systems will require more sophisticated and adaptable allocation algorithms – e.g. making extra copies when a lengthy down time is observed – in order to avoid such a high data loss rate. This is one of the significant challenges for the SD project.

5. SIMULATION

The availability data we collected – particularly the periods of correlated failures – leave in doubt the availability levels Storage@desk can deliver. Using the data from the feasibility study, we developed a trace-driven simulation to model several storage allocation schemes to determine the storage availability characteristics they exhibit. We modeled the availability of virtual storage volumes, which are fixed-sized storage allotments comprised of a number of fixed-sized data blocks (in our simulation, block sizes are all 100 MB). We use a very conservative definition of availability: a virtual volume is defined as available for a time slot only if at least one copy of *every* block it uses is available; otherwise it is unavailable.

We ran two sets of simulation cases, corresponding to reasonable numbers and sizes of virtual volumes for the first two cases presented in Section 2.2. To model modest sized individual storage allotments (small volume case), we simulated 10,000 1GB virtual volumes. To model large data pools (large volumes case), we simulated 10 1TB virtual volumes.

For each use case we simulated using two simple block allocation strategies: Random and Best Machine First (BMF). For each strategy, we statically allocated the entire volume at the start of the simulation and modeled using 3 replicas for each data block. For Random, blocks were assigned randomly from the pool of available blocks. For BMF, we used the first day's data (9/15/2005) to train the system – ranking machines in order of availability – and then assigned blocks to the most available machines first. We simulated each algorithm for both use cases using the data of the remaining 90 days (9/16/2005 – 12/15/2005). In each simulation we used a total of 30TB (10TB data x 3 copies), or 91% of the 32.8TB aggregate minimum space available. This fairly aggressive allocation of 30TB had the side effect of forcing the simulation to spread out allocations across most machines in the study – including less reliable machines and machines with little available space.

Figure 16 shows the distribution of virtual volume availability for the small volume case. Both allocation algorithms delivered fairly high availability. Random achieved 99% availability for 98.49% volumes. Unfortunately, under Random placement 1.51% volumes had 98% availability or below and as low as 92% – a completely unacceptable level (not shown in figure due to scale). This was caused by a few machines that stayed down for long periods of time during the study. With so many volumes allocated, it was likely that some volume would have a block that landed on 3 such machines. BMF performed better and achieved 99% availability for 99.6% of volumes. The rest volumes had availability from 96% to 98%.

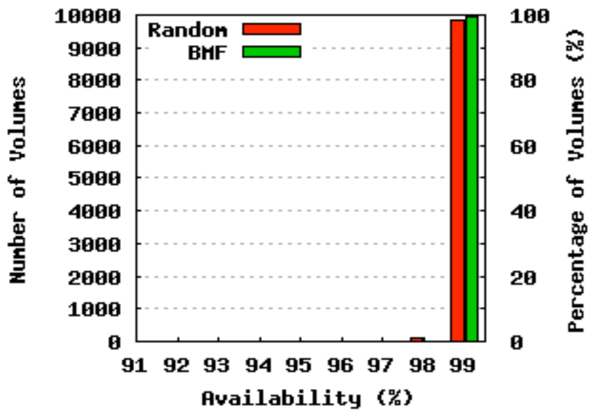
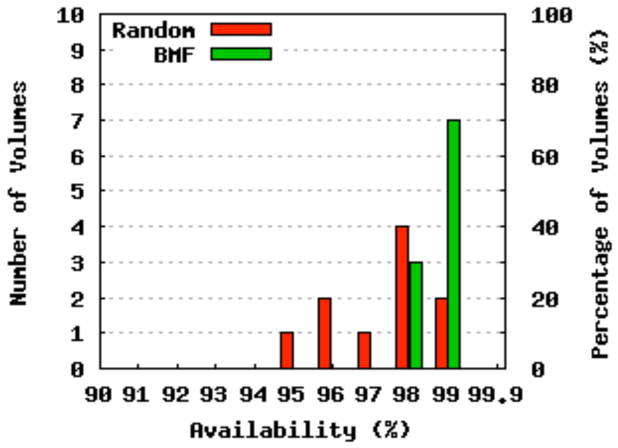


Figure 16. Random vs. BMF for small volumes.



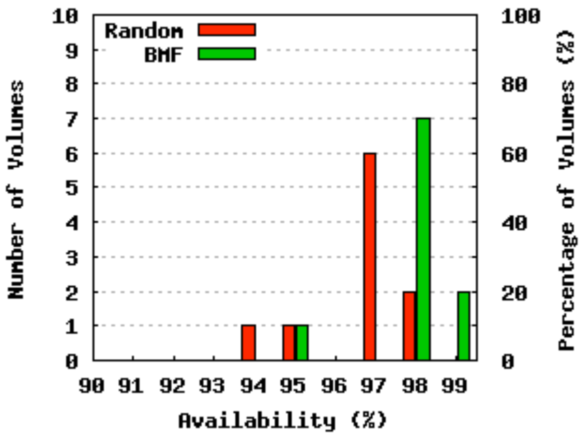


Figure 17. Random vs. BMF for large volumes. (a) volume availabilities in the first 30 days; and (b) volume availabilities in 90 days.

For the large volume case, we would expect lower availability numbers because we expect to use more blocks spread over more machines to provide 1TB of space for a volume. In Figure 17 (a), we evaluated both algorithms for the first 30 days of the simulation period. BMF achieved 99% for 7 volumes and 98% availability for 3 volumes. Random performed worse – 8 volumes did not achieve 99% and went as low as 95%. In Figure 17 (b), the simulation lasted for 90 days.

Although BMF placement still outperformed Random, BMF only allowed 2 volumes to achieve 99%, 7 volumes with 98%, and 1 volume with 95%. Random did not achieve two nines for any volume. This result is quite different from that of 30 days. It indicates that some machines experienced dramatic changes in their availability characteristics between the first 30 days and the final 61 days of the study. In fact, further investigation revealed that many of the most reliable machines during the first 30 days were taken down for an extended period for scheduled maintenance. Several simple approaches come to mind to avoid this problem. The first strategy is to make the SD system aware of scheduled maintenance so that it can migrate data off down machines. The second strategy is to exploit data on machine failure correlations to avoid putting replicas on machines with high failure correlations. The third is to use a dynamic scheme to detect down machines and create additional replicas from copies that are still running if possible.

Despite their simplicity, the two placement strategies did significantly improve availability for both use cases. This is encouraging for Storage@desk, but issues remain. First, we need to study how well BMF performs under different circumstances. Second, we need to further study long term reliability which we have theorized could be significant enough to be a problem. Third, we need to study more sophisticated approaches to achieve greater levels of availability. One approach is to adopt feedback control theory to actively monitor and manage data replicas in order to reduce the correlated faults that would happen in the system. One important property of Storag@desk system is the dynamic behaviors of distributed machines, which is difficult to predict ahead of time. We hope to model such a distributed storage system and design a closed-loop feedback controller, which can determine the appropriate number of replicas for each volume based on measured outputs. The system will then add or remove replicas for each data object accordingly. Another alternative is to use economic theory to model the storage providers and consumers in the system, namely, storage machines and clients. In a market model, storage providers compete for maximum revenues by selling the resource when the prices are right. Meanwhile, storage consumers bid in competition for scarce resource, e.g. disk space, CPU, memory, network, etc. Subject to the budget constraints, consumers purchase necessary resource at an agreed-upon price to construct virtual volumes. Market equilibrium will be achieved as consumers and providers reach a market clearing price, e.g. demand is equal to supply. We will study these approaches in future work.

6. RELATED WORK

Harnessing unused resources from desktop computers is not a new idea. Many systems, such as Condor [6], Entropia [7], or SETI@home [8], have successfully scavenged CPU cycles in the past. They make use of idle CPUs when the screensaver is on or when no-one is logged in. There have been projects that looked into scavenging unused storage resources on desktop computers. FARSITE [9, 10] is a replication-based serverless distributed file system built on Windows desktop machines. It is not intended to utilize unused disk space, although this may come as a side effect. FARSITE employs the Byzatine-fault protocol to manage file and directory metadata, which help maintain strong consistency across replicas. An NTFS interface makes FARSITE not suited for heterogeneous environments. Another resource scavenging system, FreeLoader [11], uses desktop storage to create cache/scratch space for remote scientific datasets. FreeLoader also utilizes file stripping across a set of machines in order to provider high data throughput. Both FARSITE and FreeLoader provide best-effort services to the client with no service guarantees.

In addition to NAS and SAN, several research projects have built networked storage systems. Petal [12], a distributed block-level storage system, proposes a concept called virtual disk, which can be created on demand on storage servers that manage the attached disk array. It enables user to access data via a disk-like interface. Two data distribution schemes, data striping and a replication-based scheme, are supported.

OceanStore [13] (Pond [14] as the prototype) is a global scale storage system based on a utility computing model. (Each user pays a monthly fee for a service backed up by a number of service providers.) The goal is to allow mobile devices/applications to access data anywhere, anytime. OceanStore caches data very aggressively, making a noticeable trade-off between consistency and availability. Like FARSITE, OceanStore provides strong consistency across replicas with the help of a Byzantine-fault tolerant commit protocol.

As a peer-to-peer storage system, CFS [15] stores data blocks reliably using distributed hash tables (DHash). DHash arranges blocks over a large number of servers to provide high scalability, fault tolerance and load balancing. The CFS client software can translate data blocks as files and directories and present a read-only file system interface to the clients. Ivy [16] is a read/write file system, saving data and meta-data into logs. The logs are in turn saved in DHash across the unreliable peers. A client reads a file by scanning all the logs, starting from the most recent records.

Our work differs with the above systems in the following ways. First, we attempt to build a general purpose, standard interface storage system out of underutilized storage resources on desktop machines, with no need of purchasing any new disks. With Storage@desk, a client will be able to create an NTFS or ext3 file system on a virtual volume just as what one would do on a local disk. Second, we will enhance this storage system with the support of a variety of QoS guarantees to the clients according to their QoS specifications and budgets. This is typically not available in the above systems.

Recently, Amazon Simple Storage Service (S3) [17] appears as a leader in commercial internet-scale storage system. As an online storage infrastructure, S3 relieves programmers and users from the tasks concerning data storage, data security and data availability. With a simple web service interface, a user can store unlimited number of data objects from 1 byte to 5 GB and retrieve them later, from any where on the web via a unique key. Storage@desk shares a few objectives with S3, e.g. cost-efficiency, scalability, high availability and reliability. However, while S3 is beneficial to web-based applications, internet bandwidth and latency make it undesirable for traditional applications usually running in local networks, which is the main target for Storage@desk. One of our research interests in Storage@desk is to seek good strategies for resource allocation and load balancing for high-performance and high-throughput applications.

7. CONCLUSION

In this paper we have proposed a new mass storage system that we call Storage@desk. As a software system on top of already deployed computers, Storage@desk creates a virtual storage pool by aggregating free disk space from existing workstations that can be accessed by user across an organization with the goal of providing an inexpensive storage solution that more efficiently utilizes current hardware and IT budget. To evaluate the feasibility of constructing such a system, we collected a wide range of hardware, load, and availability statistics from existing PCs in a large study organization for 30 days. In this paper we present and analyze the results of our study. Overall, our conclusion is that the existing desktop infrastructure of our study environment appears capable of supporting a virtual storage system.

We have begun to develop a prototype of the Storage@desk system.

REFERENCES

1. IDC. *External Disk Storage Systems Growth Continues on the Strength of Network Storage Systems, IDC Finds*. 2005 [cited September 2005]; Available from: http://www.idc.com/getdoc.jsp?containerId=prUS00228205.
2. IETF. *Internet Small Computer Systems Interface (iSCSI)*. 2004 April 2004 [cited September 2005]; Available from: http://www.ietf.org/rfc/rfc3720.txt.
3. H. Howie Huang, *Storage@desk: A New Mass Storage System with Quality of Service Guarantees for Large Organizations, Ph.D. proposal*, in *Department of Computer Science*. April 2006, University of Virginia.
4. Werner Vogels. *File System Usage in Windows NT 4.0*. in *SOSP*. 1999.
5. John R. Douceur and William J. Bolosky. *A Large-Scale Study of File-System Contents*. in *Proceedings of the 1999 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*. Atlanta, Georgia. 1999.
6. Michael Litzkow, Miron Livny and Matt Mutka. *Condor - A Hunter of Idle Workstations*. in *Proceedings of the 8th International Conference of Distributed Computing Systems*. San Jose, California. 1988.
7. Andrew A. Chien, Brad Calder, Stephen Elbert and Karan Bhatia, *Entropia: architecture and performance of an enterprise desktop grid system.* Journal of Parallel Distributed Computing, 2003. **63(5)**: p. 597-610.
8. SETI@Home. [cited; Available from: http://setiathome.ssl.berkeley.edu/.
9. William J. Bolosky, John R. Douceur, David Ely and Marvin Theimer. *Feasibility of a Serverless Distributed File System Deployed on an Existing Set of Desktop PCs*. in *Proceedings of the international conference on Measurement and modeling of computer systems*. Santa Clara, California. 2000.
10. Atul Adya, William J. Bolosky, Miguel Castro, Gerald Cermak, Ronnie Chaiken, John R. Douceur, Jon Howell, Jacob R. Lorch, Marvin Theimer and Roger P. Wattenhofer. *FARSITE: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment*. in *Proceedings of the 5th symposium on Operating Systems Design and Implementation*. Boston, Massachusetts. 2002.
11. Sudharshan S. Vazhkudai, Xiaosong Ma, Vincent W. Freeh, Jonathan W. Strickland, Nandan Tammineedi and Stephen L. Scott. *FreeLoader:Scavenging Desktop Storage Resources for Scientific Data*. in *Supercomputing 2005 (SC'05): Int'l Conference on High Performance Computing, Networking and Storage*. Seattle, Washington. 2005.
12. Edward K. Lee and Chandramohan A. Thekkath, *Petal: distributed virtual disks.* SIGPLAN Notices, 1996. **31**(9): p. 84-92.
13. John Kubiatowicz, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, Chris Wells and Ben Zhao. *OceanStore: An Architecture for Global-Scale Persistent Storage*. in *Proceedings of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000)*. Cambridge, Massachusetts. November 2000.
14. Sean Rhea, Patrick Eaton, Dennis Geels, Hakim Weatherspoon, Ben Zhao and John Kubiatowicz. *Pond: the OceanStore Prototype*. in *USENIX Conference on File and Storage Technologies (FAST '03)*. San Francisco, California. 2003.
15. Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris and Ion Stoica. *Wide-area cooperative storage with CFS*. in *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)*. Chateau Lake Louise, Banff, Canada. October 2001.
16. Athicha Muthitacharoen, Robert Morris, Thomer M. Gil and Benjie Chen. *Ivy: A Read/Write Peer-to-Peer File System*. in *Proceedings of 5th Symposium on Operating Systems Design and Implementation*. Boston, Massachusetts. 2002.
17. *Amazon Simple Storage Service (Amazon S3)*. [cited November 2006]; Available from: http://aws.amazon.com/s3.