Storage@home: Petascale Distributed Storage

Adam L. Beberg[1], Vijay S. Pande[2]

[1]Stanford University
Computer Science Department
Stanford, CA, 94305
beberg@cs.stanford.edu

[2]Stanford University
Chemistry Department
Stanford, CA, 94305
pande@stanford.edu

Abstract

Storage@home is a distributed storage infrastructure developed to solve the problem of backing up and sharing petabytes of scientific results using a distributed model of volunteer managed hosts. Data is maintained by a mixture of replication and monitoring, with repairs done as needed. By the time of publication, the system should be out of testing, in use, and available for volunteer participation.

1. Introduction

Many researchers including ourselves are currently generating huge amounts of computational data and results that need to be stored, backed up, processed, and shared with other researchers who may have ideas for how to extract knowledge from the data. Traditional methods of storing the data to RAID, backing it up to tape, and shipping tapes by land have stopped scaling with the amounts of data being generated and money available.

For example, the Folding@home[6] distributed computing project is currently running over 210 TFLOPS (actual) and even with the aggressive discarding and compression of generated data, it has resulted in 150 terabytes (TB) of generated data. This is growing at over 2 TB per month and all of this data needs backup and some limited distribution. This leads us to use distributed computing techniques to address our storage problem.

The common distributed computing model fits the current state of the Internet with fast downloads, slow uploads, and often-disconnected hosts, and those are exactly the problems to overcome in this project. A distributed storage project is also atypical in the fact that it is not GRID or LAN-centric assuming very high available bandwidth to any given host at any time.

1-4244-0910-1/07/$20.00 c2007 IEEE.

One additional challenge is the slow broadband speeds on the commercial Internet. The most common upload speeds in North America are limited to 384 Kbps under ideal conditions. This makes using Internet hosts at 30 KB/sec 1,000 times slower then tape backup, and 10,000 times slower then a local RAID volume. (We can safely multiply broadband speeds in this paper by 20 for East Asia, as their infrastructure is more developed).

We designed an architecture for practical use, that overcomes these problems by aggregating large numbers of volunteer run hosts, each with limited bandwidth and a small donation of storage. By combining storage contributions on the order of 10 GB per computer multiplied by tens to hundreds of thousands of computers, a significant storage pool can be generated. The challenge is in determining the means to store and retrieve this data efficiently and rapidly, as well as handling data integrity via redundancy and testing. We detail our solutions to those issues below.

2. Architecture

In this section, we will focus on the policy engine, registration, and volunteer aspects of the system shown in Figure 1. These are the aspects that one would only see in a volunteer configuration. In enterprise or personal storage, hosts would be self-managed or contracted with vendors under a Service Level Agreements, leading to a different set of challenges, but managing the system is straightforward due to very low failure rates and high bandwidth.

2.1. Design Assumptions

The design assumptions result from a mixture of measured numbers from the Stanford networks, Folding@home logs, and policies resulting from experience with distributed computing.

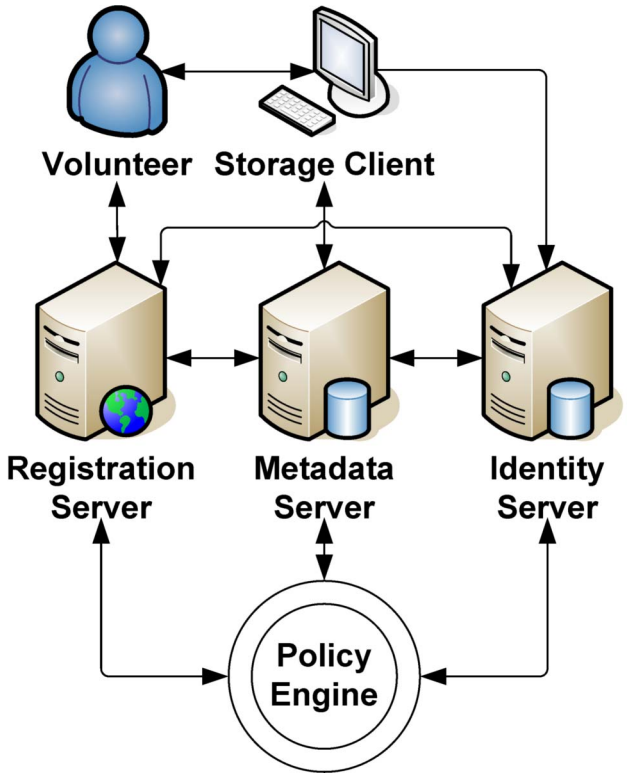

Figure 1: Storage@home Architecture

To begin, one must consider how many hosts would be volunteered. Based on previous distributed computing projects, a reasonable number of hosts to assume is on the order of 100,000 computers. This would gives one access to one petabyte of raw storage initially. Most users will have up to approximately 400Kbps of upload bandwidth, so 30 KB/s a reasonable average expectation. As ISPs typically oversubscribe, hosts near each other will have to compete for bandwidth.

Based on results from Folding@Home (unpublished results), we have found that host and volunteer churn is about 1/2% per day, rising to the 1%/day range after an influx of users from press coverage. We will cover recruiting in more depth in Section 3.1.

Files are around 100 MB each and typically represent a complete simulation run in Folding@home. This is a convenient size but not a significant parameter to performance. When dealing with the data for reading, writing, or transfer to other researchers, files are batched up or continuously queued as very large transfers. In general data is generated on the scale of weeks and months, so the need to store and retrieve data is not a surprise. The order of retrieval is also not important since each run is fairly atomic and can be processed in parallel and out of order.

A reasonable estimate of total bandwidth is 1 TB/day, in and out of the Stanford network to the commercial Internet and Internet2. This is likely to increase over time to match the increase in data generated.

One also needs to consider other challenges to the network. For example, consider natural disasters that cause the loss of all hosts on a country-wide basis, such as the December 26, 2006 earthquake near Taiwan. Internet connectivity within six countries was almost completely cutoff from the world for days. Outages due to fire, electrical loss, and storm damage are also not uncommon. Thus, one would need to consider a means to handle these correlated failures in a distributed storage system.

2.2. Implementation

The storage clients, metadata server, and identify server functions are handled by a modified version of CosmFS and CosmID[2]. The metadata server stores information about all the files stored in the system, checksums, locations, and allows for searches and status information to be run. The identity server handles the security and identity functionality, and location tracking of mobile and dynamic IP hosts.

We mainly focused on pre-configuration, hard coding authority to the Storage@home main servers, and replacing the normal user interface with the policy engine. Installers were written along with signup wrappers to make it very easy for our volunteers to install and participate without having to understand options beyond how much space we can use and tying the host to their user profile in the public statistics.

Briefly, each file is encrypted and split into four copies striped across ten hosts, giving each host an overlapped 40% of the file. This means that data is only lost if four or more of the ten hosts fail, and four or more failed hosts are adjacent in the wrapped ordering. So while some reliability is lost due to the ten hosts being involved, not all failures of 4-7 hosts result in loss. The end result is that in exchange for a 2.5x speedup in read and self repair speed, we give up only one "nine" of reliability.

Writing to the hosts is done by sending one copy of the data to a set of the 10 hosts, and letting them relay the data to complete the other 3 replicas with client-to-client transfers. Once that is done, the replicas are hashed and verified by the metadata server before the data is marked as stored.

Reading back a file can be done by reading all 10 hosts, giving us a higher read bandwidth. This is still not very much bandwidth, but since many files can be retrieved simultaneously this yields far more available bandwidth then we can possibly use.

One significant problem is that to do this, large numbers of temporary command and long lived data

connections need to be maintained to all the hosts involved in transfers at any given time. Modern operating systems are still not good at maintaining thousands of TCP connections or often even hundreds, so we will likely have to employ multiple boxes to handle all the connections. Overloading the local routers is also of some concern, but is simpler to fix by changing parameters in the routers.

The policy engine is the master of the system, and does all the planning and coordination of the storage. It pulls the information on what hosts are available from the identity server, and the data on what should be in the system from the metadata server. This allows it to trigger storage, retrieval, and transfer of data for us. It is also constantly watching the entire system ready to trigger repair operations.

A major challenge for the policy engine is the decision of where to put replicas of files so that the chances of loss are minimized. We use methods ranging from traceroute data, ISP names, clustering of the outages times, to some simple hard coded rules. One such rule is that we expect all Windows machines to reboot on the second Tuesday of each month and some fraction of them not to return until the owner intervenes. We will likely add more such rules as we discover them.

2.3 Monitoring and Self-repair

As with any storage system, the critically important aspect is not merely the size or speed of the storage pool, but how well it can handle failure and recovery operations. Even with every effort to store the data carefully on uncorrelated hosts, we will still expect 500-1000 hosts to disappear every day representing 5-10 TB of space that needs to be relocated, due to the nature of volunteer computing.

Monitoring is handled by self reporting, heartbeat functions, and polling by the identity server. Self reporting is handled during a controlled shutdown and at startup. In the case of crashes, during startup the storage client checks that the last shutdown was controlled and reports the outage since the last checkpoint. For any downtime the self reported times and the polling reported downtimes may overlap, so the downtimes are merged with min-start max-end and stored.

Each host is involved in self-repair operations for the entire system. In an average day a host will be involved in repair related transfers proportional to it's own storage, or approximately it's own storage used times the failure rate of hosts. Table 1 summarizes the expected values and overhead of the system once fully operational. The core limitation to the system is the amount of time a host is using its network connections per day, as we do not want to have hosts busy when the user wants it. We keep this

Hosts failures/day	< 1,000
MTBF of any host	86 Seconds
Repair ops/day	1,000,000
Repair ops/second	12
Hosts self-repair upload/day	100 MB
Host average upload/second	30 KB
Net use per host/day	10 x 5.5 Minutes
Average repair ops in flight	~4,000

Table 1: Summary of Self-repair Expectations

under an hour total and only 5.5 minutes at a time. Since we know the host's time zone we use that to repair from hosts that are in nighttime hours if possible. Observe that 24 hours worth of failures can all be fixed in parallel in under an hour.

Since we do not declare a host dead for 24 hours, we have much of that time to do fairly careful planning and optimizations on the locations of the new replicas. We can use all of the demographic data, clustering, and other methods to decide where to move storage. If at any time more then 2 adjacent replicas of a file are down, we can add replicas of that file immediately without any significant overhead, pruning any extras periodically.

3. Deployment

Storage@home follows a use model typical of a volunteer computing project, with an agent installed on the user's machine after they register to participate. However, much more attention needs to be paid to the user and the hosts contributing storage during the signup and maintenance processes then in other projects.

In computation-centric distributed computing, the location and management of the hosts has very little effect on the progress of the project. Slow, disconnected, or faulty (and cheating) hosts are either not heard from again, or are quickly screened out by result verification mechanisms in the servers.

In storage-centric systems, the need for long host life and the ability to decorrelate host location and failures are the most important features. This results in the need to gather accurate information during signup from the user and from automated tests. We ask the user for as much user and host demographic information as possible including location, time zone, ISP, and how much space we can use. We gather data in their bandwidth, ISP, router path to our servers, operating system, and system performance.

Unfortunately, this leads to several rather negative types of user feedback. The users most willing to help and give us significant amounts of storage are not allowed to do so due to the bandwidth constraints of their broadband

connections. Also unlike computation projects, using machines at work or school would also be quickly noticed by network administrators due to the self-repair overhead.

Since we are using the bandwidth of hosts instead of any significant computation on the hosts, this makes it possible for our Folding@home volunteer base to run both projects at full capacity. This is important as it allows us to avoid a zero-sum tradeoff to existing participants.

The current state of broadband in North America is a challenge to this type of project. Speeds are one sometimes two orders of magnitude slower then eastern-Asian options for the same price, which would allow us to use proportionally more storage per host. Currently all of the limitations of the systems are bandwidth related. Happily, this is increasing and the Internet should grow as fast as we can generate and send the data out.

The electrical usage of a host already running a computationally intense process or running as a server will not be effected by also running Storage@home as they are on all the time already. This is not true of hosts that were previously idle in hibernation when not in use, with the drives and other power-hungry components shutdown. For distributed storage, the computer need not be busy, but it must remain awake enough to answer incoming TCP connections (idle CPU, disk and display powered down). The important question here is the cost of this additional electrical use on the hosts that would be idle versus the cost of building and maintaining a data center 24/7. Since most all of our volunteers would be likely running Folding@home, and may be using their machines 24/7 for other purposes (e.g. runing a HTTPD server on their home machine), this is probably not a great increase in total energy use, or may even be a wash.

3.1. Volunteer Recruiting and Motivation

Recruiting the right type of volunteers is as important or more important than the quantity of volunteers in this case. In order for the system to perform well, we would like people to participate for at minimum 6 months. There are three main causes of users leaving a project when they are still wanted: intentionally leaving due to lack of interest, forgetting to reinstall after a hardware/OS failure, and system takeover by spyware and rootkits rendering the host useless.

Volunteers are willing to participate because the project has merit higher then the opportunity cost of volunteering, and has no real financial or commercial benefits to the sponsors. History is littered with projects that lacked one of these two aspects, so they need to be taken seriously.

The reward system for users must be chosen very carefully so that the amount of reward aligns exactly with what is most useful for the project. A central part of volunteer computing has been some means of competition. Volunteers are rewarded with points, which are made publicly viewable on the project statistics site. Volunteers also form teams to compete with other teams and to make recruiting more fun. In the case of Storage@home, we award points for giving us space, and penalize them if their hosts end up being marked as dead without them telling us first. This system will be changed as we find out what works.

However, as volunteers compete, some fraction of them will try to cheat the system and some may be hostile. This is historically a low enough percentage that it is relatively easy to detect and prevent, but it does take a large amount of time maintain. The registration server is aware of the existing Folding@home user base and statistics, but because of the intentional lack of strong identity in Folding@home, linking the two profiles is rather limited.

3.2. Policy Risk

One cannot work on the modern Internet without considering policy risks. This encompasses all the political and business reasons that may cut off communications with a great number of hosts at the same time before repairs can be made.

One example is that many companies have blocked all Internet traffic besides HTTP traffic, and that is heavily monitored and filtered. We actively discourage users from setting up Storage@home at work for this and several other reasons. A host's ISP can also change policies at any time with no notice to block some or all ports and cutoff communications. In general all peer-to-peer-like traffic is aggressively either blocked or packet-shaped today, and will be even more so in the future. The largest risk is that the ports we use get labeled peer-to-peer, which would make those ports permanently unusable to anyone on the Internet, and require us to redeploy on different ports.

We address these risks by taking measures to decorrelate storage to these parameters as well, storing replicas in different nations, states, and ISPs. Unfortunately, in the U.S. we are back to only a handful of providers, and most of our user base is in the U.S. so this is not always possible, and is as much of a risk to us as it is to every other project that uses the Internet.

3.3. The (other) Graduation Crisis

Students at universities are a common group of volunteers to distributed computing projects, and present unique challenges. Indeed, graduation is a wonderful time, unless you're running a distributed storage project.

In May/June and December (Christmas break), millions of students around the world move from high speed campus networks, to lower speed broadband connections or even to modem lines. For ~3/4 of these hosts, the move is temporary, but for those graduating it is a permanent downgrade in bandwidth, and thus in the capacity we can utilize on their computer. This is the kink in the system - high bandwidth hosts are only high bandwidth temporarily.

Educational hosts accessible by Internet2 are far faster then hosts on broadband, but they have a high degree of correlation. Many hosts at one site are far less reliable then if they are spread around and thus are less attractive in this respect.

There is also offline travel or maintenance time for most users, more common in students. The simplest solution to these problems turns out work well. Before these times of low bandwidth or travel/maintenance, we ask the users to tell us so we can flag the host as downgraded bandwidth or as in transit and not penalize them. This allows us to exempt them from the storage limit or the host-failure time limits respectively if we know about it ahead of time, since in almost all cases the data will be completely intact when they return.

The end result of this host relocation is that for now we choose to allow each machine the same maximum amount of space. This allows us to not worry about losing "superhosts" and focus on other reliability and optimization issues. This also actually helps recruitment, since people will be more likely to get their friends to run it as well to boost their team stats, since individual stats will be harder to increase.

4. Evaluation

Evaluation of the working system is primarily about reliability and thus preventing the loss of data with completely unreliable hosts.

However, there are many other metrics to consider even though most performance is limited strictly by our connection to the internet. The overhead involved in doing self-repair compared to other systems. This places stresses on volunteer systems and our servers. How well our reliability predictions compare to actual reliability is also a good metric of how well our policy engine is performing.

Scaling will also be a factor as the internet speeds up and we can store more on each node. Our servers will need to be replaced by more servers, but in all likelihood the outgoing bandwidth will scale up much slower then consumer bandwidth, server abilities, and host reliability, which means that one's institution's Internet connections remain the bottleneck. Nevertheless, we believe Storage@home will perform very well by these metrics, with caveats mentioned below.

4.1. Limitations

Using this sort of system for any sort of real-time or ad-hoc storage is not feasible. The bandwidth limitations throughout require that one plan ahead, but as stated before, this is not generally a problem for researchers. It's not difficult to plan ahead, and thus exploit the massive parallelism in the storage.

The complexity of choosing uncorrelated hosts and coordinating the repairs is non-trivial. The overhead involved in repair and verification is still minimal compared to the data bandwidth, so this will likely never become an issue. The master hosts must also be well maintained, themselves replicated with reasonable failover and recovery abilities.

5. Related Work

The number of actively or formerly deployed distributed storage systems on this scale is unfortunately limited.

SRB[1] is used to store several petabytes of federated storage in GRID systems. The storage hosts are somewhat trusted and have high reliability, so the design constraints are different. Bandwidth is measures in gigabits/second between locations, and each location has many terabytes on average. However, this is still the closest deployed system to Storage@home that we can find.

Commercial Internet backup solutions compare in many ways to Storage@home, but their cost is prohibitive to be used by a research group to store hundreds of terabytes to petabytes, and they represent a single source of failure via disaster or bankruptcy just as any off-site storage system. In the past we have backed up to Internet2 data centers, but this also has the same problems.

The Google File System[4] uses large racks of cheap drives, but relies on a high speed LAN and makes a large set of operational assumptions that do not match up with our usage. It is an example of a cluster-based system and is representative of an entire class of file systems using a metadata and data server model over the last 40 years.

Fully peer-to-peer systems research tend to focus on anonymity for large numbers of users, or the large scale distribution of copyrighted works. This results in a great deal of complexity as well as lacking the guarantees and reliability metrics we require. OceanStore[5] and PAST[3] are good representatives of this class of research.

Erasure codes are currently used in many storage and peer-to-peer systems as opposed to replication. Since the local Internet connection is a serious bottleneck in getting

the data out, straight replication allows us to send only one copy of the data and then use client-client transfers to do the replication. Verifying this has happened correctly only adds 0.1% to data overhead, where erasure coding would add 100% using a 1/2 encoding. Computationally feasible erasure code methods are also patented, and the high likelihood of submarine patents makes them untouchable in deployable systems for several more decades.

6. Conclusions and Future Work

Maintaining a centralized online petabyte would cost well over 1 million dollars for hardware alone, require twelve racks of disks, a large server room, and result in a large monthly electrical bill. Storage@home should allow us to soon access that amount of storage and maintain it with higher reliability and far less cost.

Future work includes the analysis of the performance of the system, and comparing it to our expected results in terms of reliability, and usability by a variety of researchers. We should have quite a large amount of data on machine failures, and can refine our correlation models to make them even more robust in planning where to place replicas. We also suspect we can decrease the replication factor due to the speed repairs can be made and still maintain extremely high reliability. Once established the system can be extended to other research groups at Stanford and serve to lower the cost of storage.

Acknowledgments

We would like to thank Rich Wolski for his feedback and suggestions during the preparations for this paper. We also have to thanks all the volunteers that make up Folding@home and will very shortly make up Storage@home, especially the beta testers that help us find all the moths in our relays.

References

[1] C. Baru, R. Moore, A. Rajasekar, and M. Wan, "The SDSC Storage Resource Broker", In Procs. of CASCON'98, Toronto, Canada, 1998.

[2] Cosm. http://www.mithral.com/.

[3] P. Druschel and A. Rowstron., "PAST: A large-scale, persistent peer-to-peer storage utility", In Proc. HotOS VIII, Schloss Elmau, Germany, May 2001.

[4] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, "The Google File System", In *19th Symposium on Operating Systems Principles*, pages 29–43, Lake George, New York, 2003.

[5] J. Kubiatowicz et al., "Oceanstore: An architecture for global-scale persistent storage", In *Proc. of ASPLOS*, 2000.

[6] M. R. Shirts, V. S. Pande, "Screensavers of the world unite!", Science. 290:1903-1904 (2000).