Advanced Disk Storage For Distributed Systems

Paper for the Advanced Operating Systems class of Spring 2002
Paper by **Narayanasamy Arun Prasath**
Email: apn3252@omega.uta.edu
University Of Texas At Arlington

Abstract

This paper discusses about advanced disk storage necessary for distributed systems. It briefly discusses the disk storage research and the various storage systems developed. Among them the Petal, distributed virtual disks developed at the systems research center, Digital Equipment Corporation and PersonalRaid, mobile storage for distributed and disconnected computers developed at Princeton University and supported in part by IBM is reviewed in detail. The paper goes ahead and describes some of the research carried out and storage systems developed by commercial organizations like IBM, Compaq and Hewlett Packard. Then we also compare the Petal and the Scotch systems as a discussion.

1. Introduction

Various distributed computing technologies have enabled greater integration of high-performance computing and storage resources. As computation power, degree of distribution and cost effective, scalable and high bandwidth communication increases; the need to supply the processing data promptly to attain acceptable performance increases. Very large online archives, traditional analog information being turned into digital representations and digitization of multimedia, present a challenge for developing distributed storage systems.

There are various storage systems developed as part of this challenge. The Swift system uses distributed disk stripping to provide high input and output data rates. It was developed to address the problem of data rate mismatches between the requirements of an application, storage devices and the interconnection medium. Autonomous Disks is a novel concept to build scalable distributed file systems. They were developed to support large volumes of data and high bandwidth in multimedia environment.

Mime, a high performance parallel storage device with strong recovery guarantees uses late binding of data to disk locations to improve performance. It was designed to avoid synchronous updates of metadata during normal operations. The Scotch parallel storage test beds were developed to explore and evaluate the five directions in RAID evolution. One of this is to extend RAID systems advantages to distributed and parallel computing environments.

Petal, an easy-to-manage distributed storage system is a collection of virtual disks. It can tolerate and recover from any single component failure, geographically be distributed to tolerate site failures, can be reconfigured to expand in performance, uniformly balances load and provides fast and efficient support for backup and recovery. Petal provides a lower level service than distributed file system and a distributed file system can be efficiently implemented on top of it.

While it is possible to build widely distributed systems, achieving low latency file access remains a significant challenge. NASD expanded to "Network Attached Secure Disks" is a system being developed to exhibit low latency by asynchronous oversight of the high-level file system. It provides cryptographic support for integrity of requests and self-storage management. The fastest unclassified supercomputers, ultra high speed networks, high-resolution visualization environments and toolkits for grid computing are tightly being integrated into an information structure called "TeraGrid".

PersonalRAID is a mobile storage for distributed and disconnected computers. In addition to these end hosts, at the heart of the personalRAID system is a highly mobile storage device, such as a 1GB IBM Microdrive. We call this the Virtual-A (VA) and it forms the main basis of the personalRAID. These are some of the systems. Commercially also many companies have ongoing research on these storage systems. Companies like IBM, Hitachi, Compaq and Hewlett Packard are some of them.

In Section 2 we explain in detail the Petal, distributed virtual disks and PersonalRAID, mobile storage for distributed and disconnected computers. In Section 3 as a discussion we compare the Petal and PersonalRAID systems and in Section 4 we briefly review the research being conducted commercially in companies like IBM, Compaq and Hewlett Packard on storage systems. We summarize the details by suggesting the better systems and the company with better research in Section 5.

2. Storage Systems

2.1 Petal

Petal virtual disks separate the client's view of storage from the physical resources that are used to implement it. These clients view the storage system as a collection of virtual disks and access the services using remote procedure calls. RPC interface is designed to maintain all state needed for the integrity of the system at the server and maintain only hints at the clients.

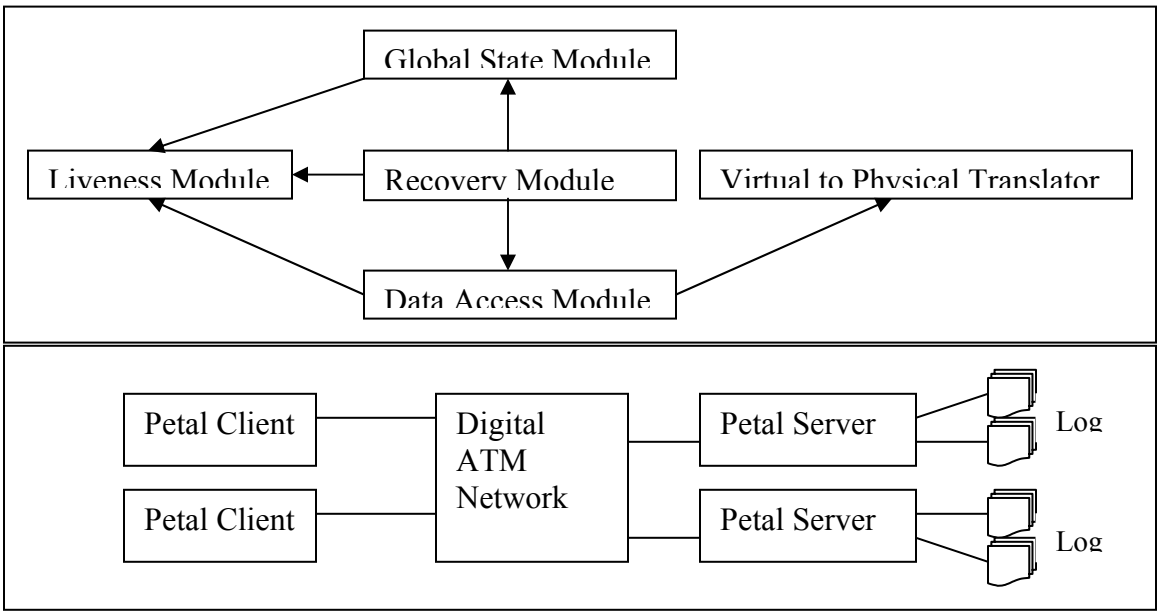

Figure above describes the software structure of Petal and the Petal prototype.

The Liveness module and the global state module handle the distributed system aspect of the Petal. The liveness module ensures the agreement of operational status on all servers and is used by other modules to guarantee continuous and consistent operation of the system. The liveness module works by exchanging messages like “I am alive” and ”You are alive” between the servers.

The global state manager is responsible for maintaining information about the current members of the storage system and the currently supported virtual disks. The global state manager is maintained based on Leslie Lamport’s algorithm for implementing distributed, replicated state machines. Since the algorithm ensures correctness in arbitrary combination of server and communication failures and recovery, it ensures fault tolerance.

The data access and recovery module controls how client data is stored and distributed in the system. A desired redundancy scheme for the virtual disk is specified when it is created. By virtual disk reconfiguration, the redundancy scheme and the other attributes can be changed transparently. The virtual to physical address translation module contains routines to translate virtual disk offsets to physical disk address.

2.1.1 Virtual to Physical Translation

This section describes how to translate virtual addresses of the form <virtual-disk-identifier, offset> to physical address of the form <server-identifier, disk-identifier, disk-offset>. There are three basic data structures: a virtual disk directory (VDir), a global map (GMap) and a physical map (PMap). The Vdir and Gmap are global data structures that are replicated and consistently updated on all servers by the global state module. The PMap is local to the particular server. The translation mechanism happens in three steps.

- The Vdir translates the client-supplied virtual disk identifier to Gmap.
- The specified Gmap determines the server responsible for translating the given offset.
- The Pmap at the specified server translates the global map identifier and the offset to a physical disk and an offset within the disk.

To minimize the cost of communication the server that performs the step 2 also performs step 3. There is one Gmap per virtual disk that specifies the tuple of servers spanned by the virtual disk and the redundancy scheme used. The Pmap is the actual data structure used to translate an offset within the virtual disk to a physical disk and an offset within that disk.

2.1.2 Backup and Incremental Reconfiguration

The mechanism for backup and recovery in Petal is called fast efficient snapshots for virtual disks. A copy of the virtual disk is created quickly and is treated like another virtual disk but it cannot be modified. The translation procedure for snapshots is slightly complicated. The Vdir is translated to a tuple of the form <global-map-identifier, epoch number>, which is used in the last step of the translation. When a new snapshot is created

then a new tuple with a later epoch number is created in the Vdir. All access to original Virtual disk is made using new epoch number which ensures that any new data written to a virtual disk will create new entries in the new epoch rather than overwriting the data in the previous epoch. Consistent snapshots require pausing the applications or else creating a crash consistent snapshot that would be left after if a disk crashed. Later by running an application-dependant recovery program we can make the snapshots consistent.

The existing virtual disks can be reconfigured to take advantage of the new resources that are added. This can be generalized for reconfiguring while removing the resources. Addition of disk to a server is handled locally and subsequent storage requests take into consideration the new disk also. A local background process runs to periodically move data among disks for load balancing. The addition of a server is a global operation involving many steps and the new server will participate in all future global operations. Finally the existing virtual disks are reconfigured to take advantage of the new server as follows.

- Create a new Gmap with the desired redundancy scheme and server mapping.
- Change all Vdir entries that refer to old Gmap to refer to new one.
- Redistribute the data to servers according to translations specified in the new Gmap and this data distribution could potentially require appropriate network and disk traffic.

The performance of the virtual dick when a server is added should increase compared to its old configuration. The reconfiguration algorithm should ensure this. There is a basic algorithm and then a refined algorithm. The refined algorithm has been implemented in this system because reconfiguration using basic algorithm can take a long time to complete and the server mapping for the entire virtual disk are changed before any data is moved which degrades performance. The basic algorithm executes the first two steps described above.

Next starting with the translations in the most recent epoch that have not yet been moved, data is moved to the new collection of servers as specified by the new global map The refined algorithm relocates only small portions of the virtual disk at a time. The basic idea is to break up its address region into three regions: old, new and fenced. Requests to old and new global maps simply use old and new Gmap's. The request to the fenced region uses the basic algorithm. Once the fenced region becomes empty then another part of the old region is relocated till the entire old region is relocated.

2.1.3 Data Access and Recovery

The dynamic load balancing eliminates bottlenecks by ensuring uniform load distribution even in the case of failure of component. The data access and recovery module are chained and declustered and hence appear as a single virtual disk to clients. There are blocks of data and two copies of the data blocks are always stored on the neighboring servers, which have common data blocks. So if there are four servers named 0,1,2 and 3 then if server 1 fails then server 0 and 2 will automatically share server 1's read load. Since server 3 is free and chained with other servers, servers 0 and 2 can offload some of their normal read load on server 3 and achieve uniform load balancing.

The choice of the load-balancing algorithm is still under active research for the Petal system. In this implementation the two copies are named as primary and secondary. Read requests can be serviced from either the primary or secondary, but the write requests should start with the primary and use the secondary only if the primary is down and this helps in avoiding deadlocks.

2.2 PersonalRAID

PersonalRAID is a mobile storage solution for distributed systems. The availability of a single coherent name space, reliability and acceptable performance are three desirable characters for such mobile storage. At the heart of the PersonalRAID is the Virtual-A (VA) a mobile storage device. Instead of totally relying on the VA, it becomes part of the PersonalRAID system because of its capacity, performance and reliability limitations. The design of the PersonalRAID should satisfy the following requirements.

- Recording should not impose excessive overhead that may interfere with normal input and output.
- During disconnection the user should not be forced to wait for a long time before the VA can be removed safely.
- During connection the user should not be made to wait for long before performing normal input and output.
- Replaying should not impose an excessive overload that might interfere with normal input and output.
- Replaying should proceed fast so that the disk space can be freed up for future input and output.

2.2.1 Design Alternatives

One of the simplest ways of design is for the system to identify at the end of the day the files and directories that were modified and then mirror only them to the mobile VA. Then after coming home it is copied from the VA to the home system. Though simple is has the disadvantage of user having to wait while the system has to copy while disconnecting and has to update again while connecting. These latencies can be intolerable. Anather way to accomplish this is to incrementally copy the files that are used as a background process. This reduces disconnection time but still the connection latency exists.

To address this the PersonalRAID has to transparently decide which device to access for the data. With this in mind, let us consider that the system while recording makes updates in the source device and logs those updates in the VA. After connection while background replaying occurs the user can access the log in the VA. This reduces both connection and disconnection latency but the file system has to be on the local system. This can be made better by having a log structure in the local file system as well. By using this log structure we expect to have efficient replay mechanisms that avoid the pitfalls.

2.2.2 Log structured organization and Log structured logical disks

A good PersonalRAID should have at least the property that the mobile device be an integral part of the system so that the users can read and write data without the connection and disconnection latencies and the transfer of data should take place in a way that avoids the intrinsic bottlenecks of disks. This can be achieved by a variant of the log structured file system. While recording the data can be copied to large memory buffers called segments, which to some extent prevent the overwritten data from reaching the VA. Fast replaying is possible as the system can transfer data with segment-sized granularity. Since live data is read large amounts of empty segments are generated on both the VA and the destination device, therefore replaying and disk cleaning become an integral process.

A PersonalRAID can be built by modifying a log structured file system. We will design a log structured logical disk to build a PersonalRAID. A logical disk behaves like a normal disc from the point of view of a file system. A log structured logical disk maps logical addresses of blocks that are written together to consecutive physical addresses. Each device is structured as a segmented log and data blocks are appended to the log as the LLD receives write requests from the file system. Each segment contains a segment summary that aids crash recovery of the in-memory logical to physical address mapping. This address mapping can be checkpointed to disk during graceful shutdown to improve recovery time.

The three main PersonalRAID data structures are segment summary, checkpoint and in-memory-map for each of which there is a one local disk version and one VA version. The version of the data structure for local disk is the same as LLD but for the VA it is augmented with a bit (b_i) per host and $b_i = 1$ if the block needs to be propagated to the host i. Each entry of the VA in-memory map is augmented with a state field of four bits ($S_{0\text{-}3}$). The state field is simply a more compact memory representation of the bitmap field.

2.2.3 Log Operations

There are various PersonalRAID operations that interact with the log on each device. During Recording a system appends newly created logical block to two logs on the source and the VA. The logical address is recorded in the segment summary along with the timestamp. The in-memory map of each device is updated to reflect the latest locations of the data block. Is a segment is set to 1 then it should be propagated to all other hosts.

A graceful shutdown forces a checkpoint to the logical-to-physical map of a device to itself during disconnection. For the local disk, we simply write the in-memory map to disk and for the VA we must read the bitmap fields using the old bitmap and state fields, and write a new checkpoint to VA. We also store the latest timestamp in the VA checkpoint, which marks the end of the current session and the beginning of the next session. A crash is a special case of a disconnection.

To recover in case of crash we maintain two checkpoints for each device. The main goal of crash recovery is to reconcile the contents of the segment summaries and checkpoints to make them consistent with each other without the need of in-memory

maps. The bits in the VA checkpoint remain fixed. On comparing the blocks latest timestamp obtained during segment summary and the timestamp at the beginning of the session, we can say if the block has been overwritten. If it is then we set to bit so that the block is propagated, otherwise we compare the timestamp of the block on a local disk against that of the same block on the VA. Until recovery is done the VA device cannot be used.

During connection the system initializes the in-memory maps by reading the checkpoints of the devices. The state fields of the VA in-memory map are calculated and set accordingly. Finally the disconnection timestamp in the VA checkpoint is read to initialize the current timestamp and in-memory map is looked at to decide which one has the most recent copy of the logical block.

Replaying is performed at the background and is integrated with disk cleaning. The system checks for live data on the VA that is yet to be propagated to the destination device. The timestamp of the propagated block is preserved and the live data read is appended to the log of the destination device. Next we check is the block needs to be propagated to other systems in the system. There are three operations that occur while cleaning. Freeing an unused block results in no cleaning. Propagating a block requires one read from the VA and one possible write to the destination disks. Retaining a block requires one read from the VA, one write to the destination disk and one write back to the VA. PersonalRAID should favor cold blocks that require less cleaning and deliver more free space.

2.2.4 Recovering from lost devices

If host disks are lost then recovering is pretty simple. The VA is taken to a surviving computer and the union of the contents of the computer and the VA gives the entire contents of the PersonalRAID, which appears to the host transparently. The more complex case is when we loose the VA device. To reconstruct this data we may have to visit all the nodes once or twice in a naïve solution. In the first tour we find the physical address at which the replica of a lost data block resides and in the second tour we copy the data onto the VA. To eliminate one of the tours we can make a copy of the VA checkpoints locally during the disconnection process. The bitmaps allow the system to identify the machines on which the latest copy of VA is stored. The price one pays for the simpler approach of using bitmaps is extra time and space. To further reduce the number of machines one must visit we can replay some updates to all as-yet-unpropagated-to hosts to eliminate these blocks from the VA or replay all blocks to one host.

2.2.5 Reconfiguration and Virtual VA's

Reconfiguring the host is pretty simple. To remove a host the system has to reformat the VA checkpoint to remove a bit from each bitmap. This checkpoint reformatting helps the system to discard data from the VA, no longer propagating it to the removed host. To further simplify, the system can simply record at the beginning of a checkpoint which bits in the bitmap are still considered to be active. Adding a host is similar like recovering from a loss of host disk.

So far we were thinking of VA to be an IBM microdrive. It can infact be backed by a file, a local disk partition, or even a network connection. These backing devices are caller virtual virtual-A or V ^{2}A. As Mobile storage devices do not have the best performance characteristics, recording to VVA can be more efficient. Still we copy the contents from VVA to a VA. A VVA that copies to a VA asynchronously at the background allows unlimited buffering.

3. Comparison

The Petal has 24 RPC calls, which are devoted to management functions like creation, deletion, making snapshots and reconfiguring a virtual disk and adding and deleting servers. Throughput is mostly limited to CPU overheads. The availability of cost effective scalable networks is the driving force behind this work. Petal uses virtual disks to hide its distributed nature from its clients. Virtual disk abstraction adds an additional overhead and can prevent application specific disk optimizations that rely carefully on placement of data. Each petal server is of the same complexity of RAID controller and has very similar hardware resource requirements. The interface is not rich and more like a file system. This is concentrated on replication based redundancy schemes like chained-declustering, even though they impose a higher capacity overhead. The performance of petal degrades gracefully as a fraction of the number of failed servers and throughput scales well with the number of servers.

The PersonalRAID is designed for single user to control a number of distributed and disconnected storage devices with VA as an integral part of it. It is a Log structured design as opposed to the Petal. VA communicates to host through various forms of connectivity. The loss of a single device doesn't result in data loss. Input and output takes place as sessions. Synchronization is achieved by replaying some of the updates, which are recorded on the VA. These systems don't receive concurrent updates, as a user cannot be at multiple sites and as the VA is mobile, being carried from one system to anather. If there are conflicting updates then a sophisticated extension of the PersonalRAID is necessary. The cost of segment cleaning arises as its log structured. The capacity constraint is not addressed properly. It can provide mechanisms such as versioning, which only higher-level systems can exploit.

4. Commercial Research in Storage Systems

4.1 IBM

There are various research projects going on in IBM related to storage systems. Some of them are the Automatic locality improving storage (ALIS) system that masks the gap between processor and disk storage. The goal of ALIS is to reorganize data to increase spatial locality to improve disk performance. Collective Intelligent Bricks (CIB) is an autonomic steward of distributed data. The goal of CIB is a modular, scalable, fault tolerant, low cost and self-managing storage system. The IceCube is a large, three-dimensional array of "intelligent bricks" that collectively provide high-performance and resilience against failures in an extremely dense package.

4.2 Compaq

The Petal system described above was developed at Compaq. Frangipani is a scalable distributed file system. The goal of this project is to provide users with a distributed and scalable file system. The idea is to build a local (vnode-level) file system that stores its data on a Petal virtual disk.

4.3 Hewlett Packard

Amongst the different storage systems developed in Hewlett Packard, A Frequent Redundant Array of Independent Disks (AFRAID) was designed to remove the small update penalty that plagued the traditional RAID 5 disk arrays. The HP AutoRAID hierarchical storage system automatically and transparently manages migration of data blocks between the two levels of hierarchy in it as access patterns change. This is suitable for a wide variety of workloads and easy to use. The Mime is another system, which is a high performance parallel storage device with strong recovery guarantees.

5. Conclusion

The principal goal of almost all the systems explained briefly and those described in detail is to have storage systems for heterogeneous environments that are easy to manage and scale gracefully in capacity and performance. But as technology advances there is a large array of disconnected storage devices. Of the two systems we have discussed in detail, the Petal handles the file system gracefully, virtualization makes it easier to allocate physical resources among many heterogeneous clients with expandability and it degrades gracefully as fraction of the number of failed servers and the throughput scales well with the number of servers and is better for multiple users without mobility in storage. The PersonalRAID system which is a device that allows user to transparently transport, replicate and access data while interacting with a number of disconnected storage devices employing distributed log-structured organization seems to be a better solution with all its limitations when mobility of storage and single user with many distributed and disconnected computers is concerned. Amongst the various companies performing research on storage systems, the research at IBM (ALIS, CIB and IceCube) and the goals they project seems better than those at Compaq (Petal and Frangipani) and Hewlett Packard (AFRAID, AutoRAID and MIME).

References

(1) J. Hennessy and D. Patterson. Computer Architecture: A Quantitative Approach. Morgan Kaufmann, 1990.

(2) Distributed Systems: Concepts and Design Second Edition Published by Addison-Wesley, 1994.

(3) S. Sobti, N. Garg, C. Zhang, X. Yu, A. Krishnamurthy, and R. Y. Wang. PersonalRAID: Mobile Storage for Distributed and Disconnected Computers. Proc. First Conference on File and Storage Technologies. January 2002.

(4) Distributed and Hierarchical Storage Systems Craig J. Patten, K.A. Hawick, J.F. Hercus and A.L. Brown. Workshop, Rutherglen Victoria, Australia, 27-29 January 1999.
(5) Petal: Distributed Virtual Disks Edward K. Lee and Chandramohan A. Thekkath. Systems Research Center Digital Equipment Corporation. Proceedings of the Seventh International Conference on Architectural Support for Programming Languages and Operating Systems 1996.
(6) The Scotch Parallel Storage Systems. Proceedings of the IEEE conference, March 5-8, 1995, San Francisco. Garth A. Gibson, Daniel Stodolsky, Fay W. Chang, William V. Courtright II, Chris G. Demetriou, Eka Ginting, Mark Holland, Qingming Ma, LeAnn Neal, R. Hugo Patterson, Jiawen Su, Rachad Youssef, Jim Zelenka.
(7) Swift: Using Distributed Disk Stripping to Provide High I/O Data Rates Luis-Felipe Cabrera Computer Science Department, IBM Almaden Research Center and Darrel D.E.Long Computer and Information Sciences, University of California, Santa Cruz.1991.
(8) Autonomous Disks: The Building Block for a Scalable Distributed File System Cuneyt Akinlar, Walid G. Aref, Ibrahim Kamel and Sarit Mukherjee Panasonic Information and Networking Technologies Laboratories 1999.
(9) Mime: A high performance parallel storage device with strong recovery guarantees. Chia Chao, Robert English, David Jacobson, Alexander Stephanov and John Wilkes HPL-CSP-92-9-rev1.1992.
(10) TeraGrid a distributed terrascale facility. http://wwwsdsc.edu/Press/01/080901_teragrid.html
(11) Network Attached Secure Disks (NASD) http://www.pdl.cmu.edu/NASD
(12) Research at Compaq on storage systems. http://www.research.compaq.com/SRC/projects/ http://research.compaq.com/SRC/articles/199811/petal/
(13) Research at Hewlett Packard on storage systems. http://www.hpl.hp.com/research/itc/csl/ssp/Panasonic
(14) Research at IBM on storage systems. http://www.research.ibm.com/compsci/storage/index.html