Site-Based Cache Design ▪ SMIL 2.0 ▪ Open Source Convention

IEEE

Internet Computing

SEPTEMBER/OCTOBER 2001

Distributed Data Storage

p. 35

IEEE COMPUTER SOCIETY
http://computer.org/

IEEE

The Evolving Field of Distributed Storage

Peter N.Yianilos (guest editor)
Intermemory.net

Sumeet Sobti
Intermemory.net

Memory is a fundamental commodity of computation. Storage systems provide memory that costs far less than RAM and has greater persistence, but suffers from much lower data transfer bandwidths and higher access latencies. These attributes — cost, persistence, bandwidth, and latency — are the traditional evaluation metrics for storage systems, but the remarkable growth of communications and networking over the past few decades has complicated this simple picture.

Today the network is an integral part of the computer. Most of us routinely access Web pages whose display requires fetching data from dozens of machines around the world. The Web is the first distributed storage system to have such an immediate global impact. It illustrates the technological, economic, and cultural power of a distributed approach. However, the Web's fragility and operational semantics prevent it from addressing the storage problems of mainstream data processing. For example, error messages or suspended display is common when a network or system component fails somewhere, making a page or one of its elements inaccessible. Also, informal caching on the Web makes it hard to be certain that you're viewing current information.

A simple form of distributed file storage is widespread now, as many of us routinely and transparently access files stored somewhere else on a local area network. Between LANs and the World Wide Web lies the domain of distributed enterprise-wide storage, an area that industry is now actively developing.

In this increasingly complex and demanding world of distributed storage, we are forced to consider new metrics and issues beyond the traditional set. These include shared coherent access, availability, survivability, security, interoperability, search, caching, load balancing, and scale — the need for storage systems of truly immense proportions. Indeed, our increased appetite for storage has also engendered another design issue: the need to largely automate the now human-intensive task of managing large storage systems. Finally, an undercurrent in the flow of ideas concerns the cultural

1089-7801/01/$10.00 © 2001 IEEE

issues of privacy and anonymity in the context of distributed storage.

The builders of distributed storage systems face many architectural decisions as they work toward their targets among these metrics and issues. The most basic of these is the question, "Who's doing the work of providing storage services?"

Hierarchical approaches, including the new trend toward storage virtualization, use layers of control and abstraction to stitch together distributed and disparate storage providers into a single virtual whole. In the peer-to-peer approach, the clients themselves provide storage for everyone. There is no need for any server in the traditional sense. In the ideal case, such systems are fully symmetrical with no fixed central leader. The server-to-server approach is related to peer-to-peer, but here many servers work together in a symmetrical way to provide storage services; clients need not install any new software to consume basic storage services. This broad categorization includes much work in the established field of distributed file systems.

The enormous scale contemplated within OceanStore mandates mechanisms for automatic system maintenance.

In This Issue

This special issue presents three articles that address different approaches to and aspects of distributed storage.

The first article, "Maintenance-Free Global Data Storage" by Rhea et al., describes a proposed global-scale persistent data storage system, OceanStore, which targets high availability and survivability of data and performance in the presence of faults, attacks, and changing network conditions and server membership. The enormous scale contemplated within OceanStore mandates mechanisms for automatic system maintenance. The article discusses these mechanisms in the context of four pieces of the system: (1) an object-location and message-routing infrastructure called Tapestry, (2) a subsystem that disperses erasure-coded fragments of each data object to a diverse set of servers, (3) a subsystem that runs a Byzantine agreement protocol to serialize and authenticate updates to data objects, and (4) a set of algorithms that observe the behavior and usage of the system, and adapt the system to optimize performance.

The second article, "Managing Data Storage in the Network" by Plank et al. presents a programming abstraction called the Internet Backplane Protocol (IBP) for distributed and network applications. This protocol allows applications to control intermediate data-staging operations explicitly as data is communicated between processes. Applications that are aware of storage server locations in the network can use IBP to exploit locality and manage server resources effectively. The article illustrates some sample applications that can benefit from IBP. By design, the protocol is very simple to understand and use. It is being developed and deployed as part of the Internet-2 Distributed Storage Initiative.[1]

The IBP approach can be compared to the approach taken in OceanStore, where applications are entirely unaware of server locations and the system attempts to optimize the use of storage and network resources internally.

In the final article, "Managing Scientific Metadata," Jones et al. describe Metacat, a tool for collecting, managing, and querying heterogeneous metadata that is distributed across multiple sites, each possibly under the control of a different administration. The data syntax and semantics at each site are described using the Extensible Markup Language (XML). Metacat collects and stores these metadata documents in an SQL-compliant relational database management system. Metacat also allows replication of these metadata documents on multiple Metacat servers while maintaining consistency and allowing sites to retain local control over their own data and metadata.

A Broader View

To give a broader view of the scope of work in the distributed storage field, we offer a brief survey of several recent projects, including references for readers who want more detail. For earlier work on distributed file systems for LAN environments, see Levy and Silberschatz,[2] Anderson et al.,[3] and Thekkath, Mann, and Lee,[4] and the references therein.

Replication-Based Data Archives

Past, Intermemory, and Farsite are distributed data storage services that target high availability, survivability, performance, and scalability. Each of these services distributes data replicas or erasure-resilient fragments[5] to a diverse set of nodes to ensure availability and long-term data survivability. Sophisticated data structures and algorithms keep track of the replicas and fragments. These systems share many goals with the OceanStore and Internet-2 DSI projects.

For specialized efforts toward replication-based archival storage systems in the context of digital libraries, see Crespo and Garcia-Molina.[6]

Past. This global utility, in development at Microsoft Research, Cambridge, UK, names files with random-looking bit strings that are cryptographically tied to their content and are, therefore, immutable.[7] Replicas of files are placed on a diverse set of nodes by a fault-tolerant and self-organizing routing and location infrastructure called Pastry.[8] The set of nodes storing replicas of a file is determined pseudorandomly, which provides load-balancing guarantees and resilience against faults and attacks. When the overall storage utilization in the system is high and the balancing of data achieved by randomization is not enough, Past takes additional measures, called replica diversion and file diversion, to achieve more fine-grained load balancing.

The Past system uses smart cards to ensure system integrity. A third-party broker issues smart cards to each system node and user. The smart card contains a unique (pseudorandomly generated) ID, a unique private/public key pair, and the broker's public key; the broker signs the ID and the public key. The smart card generates and verifies various certificates during insert-file and reclaim-space operations. Each user's smart card also maintains the usage quota for the user. The use of smart cards is also a mechanism for giving users necessary credentials anonymously, as smart cards with fixed usage quotas can be sold in retail stores.

Intermemory. The Intermemory project addresses the issue of long-term preservation of information in the context of digital libraries.[9,10] The system can be envisioned as either peer-to-peer (a public intermemory) or server-to-server where libraries and institutions cooperate to create a robust storage substrate for their collective holdings.

Unlike other projects with similar availability and survivability goals, this work implements a distributed block-store substrate on which arbitrary data structures, including conventional file systems, can be built. It features two-level dispersal of erasure-resilient fragments with 1,024 nodes ultimately participating in the storage of each block. The system uses database synchronization between random pairs of Intermemory nodes to propagate metadata and data fragments efficiently and to provide an automated self-repair mechanism.

Many ideas from the original Intermemory project are now contributing to the Intermemory.net commercial project.

Farsite. Farsite is a replication-based distributed file system where clients contribute storage resources in exchange for a highly available and reliable file system service.[11] Logically, a single hierarchical file system is visible from all access points, but underneath files are replicated and distributed among the client machines. There is no central authority that administers the system. The system does not assume mutual trust among the client machines. Write access to files is controlled using digital signatures. All files are encrypted before storage using a technique called convergent encryption, which allows files with identical content to be detected even if they have different names and are encrypted using different keys. Files with identical content are coalesced to save space. Also to save space, files are compressed at write time, and decompressed when read. A distributed directory dynamically keeps track of replica and file version locations.

Peer-to-peer file-sharing systems do not guarantee long-term survivability of files.

File-Sharing and Publishing Systems

Napster, Gnutella, Mojo Nation, and Freenet are peer-to-peer file-sharing systems in which many nodes contribute storage and network resources to build a highly available pool of files. Decisions about which files to keep and evict over time are made locally by each individual node. As a result, these systems do not guarantee long-term survivability of files — especially those files that few users are interested in storing or accessing.

Freenet is one of several projects that have targeted anonymity of publication and resistance to censorship among their objectives. Publius and Free Haven are two others, summarized here. In addition, a seminal paper by Ross J. Anderson, "The Eternity Service,"[12] proposed a storage medium with properties similar to the Internet in order to resist denial-of-service attacks aimed at censorship. Two Web sites — http://www. cypherspace.org/~adam/eternity/ and http://www.kolej.mff.cuni.cz/~eternity/ — offer information on sample implementations of Anderson's ideas. A recent book edited by Andy Oram provides more information on many of these projects.[13]

Finally, the Xanadu Web site (http://www.xanadu.com) documents a long-running effort concerned with improving distributed publishing and referencing, and related intellectual property and copyright issues.

Napster. Napster (http://www.napster.com) is an MP3 file-sharing system. Users contribute storage and network resources, and their holdings are available only if they are operational. Files are searched via keywords that are matched with metadata associated with each file. Indexing and searching of files is centralized at the Napster site. Napster brokers client connections, but data transfer happens directly between clients.

Gnutella. Gnutella (http://gnutella.wego.com) is a decentralized protocol built on a self-adapting overlay infrastructure. Here files are not searched by names. Instead, each query is broadcast to a set of nodes in its raw form, and each receiving node is free to interpret it as it likes. For example, a node can try to match the query with file names or file contents, or it can process the query in any other manner. Queries and other requests are broadcast by a multihop flood algorithm that prevents loops. The result is a very high probability of finding popular files. Data transfer takes place directly between two parties.

> **…a de facto economic and cultural mandate to store and archive.**

Mojo Nation. This system (http://www.mojonation.net) splits files into multiple fragments using erasure-resilient codes and then disperses the fragments to different users' machines. The goal is not long-term durability of data, but increased bandwidth and load balancing. When accessing a file, fragments are downloaded from multiple peers in parallel, and the file is reconstructed. This allows even users with limited bandwidth to contribute resources to the system. This technology is called *swarm distribution*.

Another novel aspect of Mojo Nation is an accounting scheme that provides incentive for users to contribute resources to the system. Each peer-to-peer interaction in the system costs some credits, which are counted in terms of a digital currency called a mojo. Mojos are earned by contributing storage, network, and CPU resources to the system. The bookkeeping and the transactions are carried out with the help of a trusted third party. The accounting scheme also contributes toward achieving fine-grained load balancing. For example, if a server is heavily loaded, then the system gives rich users a higher priority, thus encouraging other users to migrate to less heavily loaded servers.

Freenet. Freenet features absolutely no centralization.[14] Files are identified with unique code names (random-looking bit strings), and are searched using these names. Freenet's goals include resistance to censorship, anonymity for users, and plausible deniability for node operators. Requests and replies are forwarded through chains of nodes. Forwarded files are cached throughout the chain. Documents are encrypted, and the keys are not known to the node operators. This gives operators some degree of deniability over the content they host.

Publius. This publishing system consists of a fixed static set of servers that host encrypted content.[15] The author of a document creates a secret key to encrypt the document. The secret key is split into n shares such that k of the shares are sufficient to retrieve the key, while less than k shares does not reveal any information. Servers are pseudorandomly chosen to store copies of the encrypted document and one key-share each.

The author-to-server communication is performed through an anonymous channel like Onion routing[16] or Freedom (http://www.freedom.net). A URL is generated that deterministically maps to a set of (at least k) servers that contain the encrypted document and the key-shares. Documents are accessed using a standard Web browser and a local Web proxy that fetches the shares and a copy of the encrypted document, reconstructs the key, and then decrypts the document using the key. The Publius system also provides mechanisms to update and delete the documents, but only the original author can perform these operations.

Free Haven. This system, developed by students at MIT and Harvard, targets anonymity, censorship resistance, and guaranteed availability of each document for a publisher-specified lifetime.[17] It includes a novel reputation system for servers and a file-trading mechanism. The system sacrifices efficiency and some convenience to achieve the goals of persistence, anonymity, and server accountability. This contrasts with systems like Freenet, which target anonymity and efficiency but do not guarantee long-term survivability.

Conclusion

The ongoing evolution of storage and network technologies has supported the rapid growth in the field of distributed storage over the past few years, but a widely felt demand for more and better storage is a significant driving force behind this growth. The demand arises from an apparent de facto economic and cultural mandate to store and archive as many bits as possible. This trend will pose interesting new challenges as storage systems are asked to store not just bits, but also the their semantics. For what use is an image that can't be seen because its format is forgotten or a program that can no longer be executed because the machine that ran it no longer exists?

Acknowledgments

We thank Joe Kilian and Dina Kravets for help in preparing this article.

References

1. M. Beck and T. Moore, "The Internet2 Distributed Storage Infrastructure Project: An Architecture for Internet Content Channels," *Computer Networks and ISDN Systems*, vol. 30, no. 22-23, 1998, pp. 2141-2148.
2. E. Levy and A. Silberschatz, "Distributed File Systems: Concepts and Examples," *ACM Computing Surveys*, vol. 22, no. 4, Dec. 1990, pp. 321-374.
3. T. Anderson et al., "Serverless Network File Systems," *Proc. 15th Symp. Operating System Principles*, ACM Press, New York, 1995, pp. 109-126.
4. C. Thekkath, T. Mann, and E. Lee, "Frangipani: A Scalable Distributed File System," *Proc. 16th ACM Symp. Operating System Principles*, ACM Press, New York, 1997, pp. 224-237.
5. M.O. Rabin, "Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance," *J. ACM*, vol. 36, no. 2, 1989, pp. 335-348.
6. A. Crespo and H. Garcia-Molina, "Archival Storage for Digital Libraries," *Proc. Third ACM Int'l Conf. Digital Libraries*, ACM Press, New York, 1998, pp. 69-78.
7. A. Rowstron and P. Druschel, "Storage Management and Caching in Past, a Large-Scale, Persistent, Peer-to-Peer Storage Utility," to be published in *Proc. 18th Symp. Operating Systems Principles*, 2001.
8. A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," submitted for publication, May 2001.
9. A.V. Goldberg and P.N. Yianilos, "Towards an Archival Intermemory," *Proc. IEEE Int'l Forum on Research and Technology Advances in Digital Libraries* (ADL 98), IEEE Computer Soc. Press, Los Alamitos, Calif., 1998, pp. 147-156.
10. Y. Chen et al., "A Prototype Implementation of Archival Intermemory," *Proc. Fourth ACM Conf. Digital Libraries* (DL 99), ACM Press, New York, 1999.
11. W.J. Bolosky et al., "Feasibility of a Serverless Distributed File System Deployed on an Existing Set of Desktop PCs," *Proc. Int'l Conf. Measurement and Modeling of Computer Systems* (SIGMetrics 2000), ACM Press, New York, 2000, pp. 34-43.
12. R.J. Anderson, "The Eternity Service," *Proc. First Int'l Conf. Theory and Application of Cryptography* (PRAGOCRYPT 96), CTU Publishing House, Prague, 1996; available online at http://www.cl.cam.ac.uk/users/rja14/eternity.html.
13. A. Oram, ed., *Peer-To-Peer: Harnessing the Power of Disruptive Technologies*, O'Reilly & Assoc., 2001.
14. I. Clarke et al., "Freenet: A Distributed Anonymous Information Storage and Retrieval System," *Proc. ICSI Workshop on Design Issues in Anonymity and Unobservability*, Int'l Computer Science Inst., 2000.
15. M. Waldman, A.D. Rubin, and L.F. Cranor, "Publius: A Robust, Tamper-Evident, Censorship-Resistant Web Publishing System," *Proc. Ninth Usenix Security Symp.*, Usenix Assoc., Berkeley, Calif., 2000, pp. 59-72.
16. D. Goldschlag, M. Reed, and P. Syverson, "Onion Routing for Anonymous and Private Internet Connections," *Comm. ACM*, vol. 42, no. 2, Feb. 1999, pp. 39-41.
17. R. Dingledine, M.J. Freedman, and D. Molnar, "The Free Haven Project: Distributed Anonymous Storage Service," *Proc. Workshop on Design Issues in Anonymity and Unobservability*, 2000.

Peter N. Yianilos leads a technology incubator, Yianilos Laboratories, in Princeton, New Jersey, where he is also chair of Netrics.com, chief technology and architecture advisor to Franklin Electronic Publishers, and president of the recently formed company, Intermemory.net. His research interests include algorithms, systems, intelligence, and electronic publication. In October 2000, he received the first annual Frankfurt eBook award for technology. Yianolos received bachelor's and master's degrees in mathematics and computer science from Emory University, and a PhD in computer science from Princeton University.

Sumeet Sobti is a graduate student in computer science at Princeton University. His research interests include file and storage systems, operating systems, and the theory of exact and approximation algorithms. Sobti earned a bachelor's degree from the Indian Institute of Technology, Kanpur, and a master's degree from the University of Washington, Seattle.

Readers may contact the authors via e-mail at pny@pnylab.com and sobti@cs.princeton.edu.