Cooperative Library

By
Nathan Flores, Clark Li, and Duck Pham

1. Introduction

CL (Cooperative Library) is a new peer-to-peer storage system that provides guarantees for the efficiency, robustness, and load balance of file storage and retrieval. CL does all this with a completely decentralized architecture and efficient algorithms that easily scale to very large systems. The Cooperative Library combined with the capabilities of the Next Generation Internet can solve many of the problems corporations and institutions have when dealing with storing mass amounts of data. Current peer-to-peer systems such as Napster and Gnutella manifest the benefits of cooperative storage and serving: fault tolerance, load balance, and the ability to harness idle storage and network resources. With these benefits include a few design challenges. A peer-to-peer architecture should be symmetric and decentralized, and be able to function well with unmanaged participants. Finding requested data must be quick in a large system and servers must be able to join and leave the network frequently without affecting its robustness or efficiency. Furthermore, data load must be balanced across the available servers. While the popular peer-to-peer systems solve some of these problems, practically none solves all of them. The Cooperative Library is a new design that meets all of the challenges mentioned above.

1.1 Motivation

The Cooperative Library was built to drive demand for the Next Generation Internet, by creating a peer-to-peer file-sharing network that solves the problems most other P2P system don't. Further, it was created to meet the increasing needs for back-up and storage. IT spending has dropped since the economic slowdown and yet enterprise data is doubling every year, thus, CL attempts to fill the gap by utilizing existing IT infrastructure.

Enterprise storage represents a very large and growing market opportunity. According to IDC, the computer data storage market is expected to increase by a 23.4% CAGR from $38.3 billion in 2000 to $88 billion in 2004. The three main factors behind this dramatic growth are the enormous increase in global e-business, growing online data requirements created by the Internet, and the increasing deployment and complexity of enterprise applications. According to a study by the School of Information Management Systems at the University of California, Berkeley, the world produced approximately three exabytes of new information in 2000, the equivalent of about 500 copies of Gone With The Wind for every person on earth. This is an impressive fact, but it becomes really significant when you learn that the total amount of information created from the day man began drawing on cave walls to the year 1999 is equal to 12 exabytes of information. Furthermore, the study projects that the amount of data being generated will double every year, leading to the creation of an additional 12 exabytes by the middle of 2002.

2 Background

Conventionally, a company's storage strategy was to write a check to EMC or IBM. But why pay millions for monolithic storage devices when you've already got the

necessary storage space available? From individual power users to enterprises, today's wasted disk space can be harnessed to tomorrow's dynamically scaling storage networks. The Cooperative Library allows unused disk space within a corporation to be utilized in order to meet the increasing needs of storage. CL nodes can be setup at machines within a company's network creating an NAS (Network Attached Storage).

A CL file system exists as a set of blocks distributed over the available CL servers. CL client software interprets the stored blocks as file system data and meta-data and provides a read-only file-system interface to applications. The heart of the CL software consists of two layers, Chord and Dhash, which were researched at MIT but implemented at UC Berkeley. The Dhash (distributed hash) layer performs block fetches for the client, distributes the blocks among the servers, and maintains cached and replicated copies. Dhash uses the Chord distributed lookup algorithm to locate the servers responsible for a block.

3 Related Work

Cooperative Library was inspired by the likes of Napster, Gnutella, CFS, and OceanStore. In comparison to existing peer-to-peer file sharing systems, CL offers a simple implementation and excellent performance without sacrificing correctness.

3.1 Naming and Authentication

Like other distributed storage systems [CFS, OceanStore], CL authenticates data by naming it with public keys or content-hashes. CL implements a secure distributed read-only file system – such that, a file system in which files can be modified only by their owner, and only through complete replacement of the file. CL adds the capability to dynamically find the server currently holding the desired data, via the *Chord* location

service. This increases the robustness and the availability of CL, since changes in the set of servers are transparent to clients.

3.2 Peer-to-Peer Search

Napster and Gnutella, are arguably the most widely used P2P file systems today. They present a keyword search interface to clients, rather than retrieving uniquely identified data. Gnutella broadcasts search queries to many machines, and Napster performs searches within centralized servers. CL also provides keyword search where the query is sent to a distributed *Super File*, which contains a list of all the files in the system.

3.3 Peer-to-Peer Hash Based Systems

Like OceanStore, CL layers storage on top of an efficient distributed hash lookup algorithm. CL stores blocks, rather than whole files, and spreads blocks evenly over the available servers; this stops large files from causing an unbalanced use of storage space. CL solves the related problem of different servers having different amounts of storage space with the notion of *virtual servers*, which gives server managers control over disk space consumption. CL's block storage granularity helps it handle the load of serving popular large files, since the serving load is spread over many CL nodes along with the blocks. Such design is much more space-efficient, for large files, than whole-file caching.

OceanStore aims to build a global persistent storage utility. It provides data privacy, allows client updates, and guarantees durable storage. With this functionality comes opportunity cost: complexity. For example, OceanStore uses Byzantine agreement

protocol for conflict resolution, and a complex protocol based on Plaxton trees to implement the location service.

4 Design and Architecture

Many classic complex computer science problems such as compiler, database, operating systems, are broken down into different pieces, then tackled individually. It is said that difficult problems can be solved by imposing different layers of abstraction, CL is no exception to this rule. CL consists of three layers, namely the Chord layer, the Dhash layer, and the application layer.

4.1 Chord Layer

At the lowest level of abstraction, resides the Chord layer. Chord is a research project at MIT. We are using it for two reasons, first, Chord's ability to consistently locate a node given a hash value of a data block, or a key, secondly, the efficiency of the algorithm. It runs at O (log n) for node look up and O(log^2 n) for node join. These following sections will provide general information on the Chord concepts used in our project.

- Consistent Hashing: Each Chord node has a unique identifier, obtained by combining the IP address and the virtual node number. (Virtual node will be described in the Dhash section. Consistent hashing assigns a specific key to node by first hashing the key into the node identifier space, then finds the closest existing node with the ID on the CL network. Closeness of nodes is defined not by physical location, but by the node ID.

This algorithm is designed to allow nodes enter and leave the CL network with minimum disruption.

- Node Join/Leave: When a node joins the network, Chord assigns certain key to the new node. The selection of which key to assign to the new node is chosen by the definition of closeness of node ID. Similarly, when a node leaves the network, its keys are reassigned back to the predecessor of the node, because the predecessor will be the closest in node ID space.

- Finger table: Every virtual node contains a finger table. A finger table carries the routing information on only a part of the network. Each entry of the node is carefully chosen by the Chord algorithm, so that they are 2^i (in node ID space) apart from the previous entry, if i is the position of the node in the finger table. The special arrangement of the node guarantees a O(log n) look up time.

4.2 Dhash Layer

Building on top of Chord is the Dhash layer. It is responsible to retrieve and store data blocks reliably, even in an unreliable network environment. The Dhash layer serves as the communication between the application layer and the Chord layer.

- Data block: Each data block is associated with a key, generated by the SHA1 hash function. After the Chord layer correctly identifies the node responsible for the key, the Dhash layer sends a request to retrieve or upload the block.

- Duplicated data blocks: To induce reliability, each data block is replicated 8 times, and the 8 identical blocks are transmitted to 8 different machines for storage. A block retrieve will fail if and only if all 8 copies are unavailable. When half of the machine in CL network is down, the chance that a specific block is missing should be 1/2, since we have even distribution of blocks. Given each block is replicated 8 times, the chance that all of the 8 copies are missing is (1/2) ^ 8 = 0.00391. For the sake of the proof, let us define a reliable network as when half of the network was wiped out in a physical attack, the probability that the network can function without any failure is over 99%. If there are n block in the network, to achieve reliability, we need c copies, where (1/2)^c * (n) < .01. With reliability, there comes a price of performance degradation. The administrator of CL should carefully choose the number of copies to use in the network.

4.3 Application Layer

- User Interface: In the first version of CL, the UI was web-browser based. This required the use of servlets further needing an application server such as Tomcat running on each node. The second version of the UI is based on the Java 2 API primarily on the swing class library. We tried to build structures following the model/view/control design allowing them to be easily extendable for future additions in functionality.

- Searching: The search feature is based on a *super_file*, which is implicitly distributed throughout the network as is treated like any other file in the system. Every time a user adds a file into the system, the corresponding domain name, file name, and file description will be appended at end of the super file. When a search query is

submitted the query text is compared against all file names and keywords in the file descriptions, then a list of matches; which contain the domain name and file name are returned to the user.

5 Results

Given the 4-node network provided at the NGI office, CL was able to upload and download large files successfully between the nodes. Continuous testing of the system sometimes found that a user is unable to download files large than 150 MB, this bug is still under inspection. When shutting 2 nodes down, we were still able to obtain all the data in the system. Overall, the performance of the system was a success, given the limited time the summer had to offer. Much larger tests were applied to a similar system[1].

6 Conclusion

CL is a scalable and secure read-only file system. It provides stored data to applications through an ordinary file-system interface. Servers store blocks of data with unique identifiers. Clients retrieve blocks from the servers and interpret them as files.

CL uses the p2p Chord lookup algorithm to map blocks to servers. This mapping is done dynamically and is implicit. As a result, there is no directory information to be updated when the underlying network changes. This allows to CL to be robust and scalable. CL caches data along the lookup path for a block to achieve availability and further replicates blocks in order to achieve availability. CL achieves load balance by spreading blocks randomly over servers. It replicates a data block along the *successor*

[1] Frank Dabek et. al., "Wide-area cooperative storage with CFS", ACM SOSP 2001, Banff, October 2001

list of servers relative to the server that initially holds the block. Although CL has not been deployed on an Internet-wide scale, download performance should be as fast as standard FTP. The design of CL has provable efficiency and provably fast recovery times after failure.

7 Future Work

CL does not yet support quotas. A malicious user can insert an unlimited amount of data into the system exhausting all the available storage space. One solution is to enforce quotas on un-trusted IP addresses, an Idea we currently are pursuing.

CL cannot yet support concurrent writes. Only one user may insert new files into the system at a time under that user's domain, though an unlimited number of users can concurrently read those files. Synchronization issues are being looked into here and making the system multi-threaded.

Further issues were to explore other distributed computing models, such as combining CL with distributed computing (ex. SETI), and distributed virtual memory; this might be significant since gigabit Ethernet to memory on other machines would be faster than paging back to disk.

Other areas of work would be in the area of load testing. In order to do this, CL can be set up at different nodes within a corporate LAN and then do frequent block transfers to exploit the network's capacity and ability to handle load from varying locations.