Distributed File Backup Across Untrustworthy Intelligent Storage Agents - Literature Review

Dominic Hauton

January 7, 2017

Abstract

An application in which users can join a large public network of storage nodes to trade their available file space. This will allow the user to store files safely on hostile computers, increasing availability and redundancy. The hosts will act as intelligent agents, and trade space when they communicate with one another.

1 Introduction

The literature review explores existing literature in the area of distributed storage across untrustworthy nodes. Distributed storage is an age old idea [source], however, with the advent of cheap high-speed internet users are now able to use monolithic cloud services to backup data. These tend to be expensive for anything but small amounts of storage and have dubious terms of service concerning the security of the data users store on them. [source]

The price of storage has also decreased at a high rate. [source] It is now common for computers to come with large amounts of hard drive space which is often not filled to capacity. The project proposed promises to swap this free hard drive space to back up other users' data in exchange for their surplus space to backup your data.

To simplify the process, the system will be able to negotiate contracts of varying complexity for space allocation on another machine, in exchange for space on itself. Unlike most distributed databases and cloud storage solutions frequent down-time will be expected across devices and contracts will have to account for this.

The challenges of such a system include a way of only copying parts of a file during updates to reduce bandwidth usage. An effective way of managing where file chunks are stored. A way to encrypt the files on the remote host and a way to authenticate that the host still holds the file.

2 History of Problem

A simple file backup can be imagined as simply copying a file to another location. In order to keep the duplicated file in sync they must be compared. The program diff solves this through finding the longest common sub-sequence between files. In order to improve performance hashing, presorting into equivalence classes, merging by binary search, and dynamic storage allocation are used. [Hunt and MacIlroy, 1976]. This allows the user to view changes and copy the file over again if required. In a networked scenario, bandwidth from source to destination is at a premium. Rsync, introduced in 1996 presents a much better solution. Instead of copying the entire file multiple times, the changed file chunks can be copied instead. [Tridgell et al., 1996]. Rsync splits the file into chunks and calculates a weak rolling checksum and strong MD4 checksum for each block. Unless there is a match, the weak checksum moves across the file to find a possible offset for matching data and confirms the offset with the strong checksum. The gap can then be requested to be copied. This results in a lot less data being copied than there would be with a diff file. [Tridgell et al., 1996] This combination of weak and strong checksums has been used across multiple distributed systems including low-bandwidth file systems [Muthitacharoen et al., 2001] and high performance transactional object stores. [Stephen et al., 2000].

Multiround Rsync improves on the rsync algorithm by allowing for more communication to lower bandwidth. Blocks of smaller and smaller sized are used to find holes in the old file in each round and the file until the minimum block size is reached and a copy occurs. [Langford, 2001] This works better than standard rsync in situations where the source file has been changed in many places distributed around the file.

Rsync requires both old and new copies of a file to exist on the host system during an update. This issue has been addressed by creating an in-place rsync (ip-rsync) that uses techniques used in delta compression and Lempel-Ziv compression to move the areas of the file around. In ip-rsync file sender sends add and move commands to the destination in an order that guarantees no files will be overwritten. [Rasch and Burns, 2003]

Git is an improvement on Rsync as it provides both version history and minimises data transfer. To keep storage simple, a copy of the whole file is stored and a reference is put into the version history. By storing old files locally operations are fast. This is also an important distinction from other version control systems, because it means the systems can operate without a centralised server. To reduce file duplication all files are referenced using their SHA-1 hash. This means you can be sure the contents of the file hasn't changed since you last read it.

Git also uses a 3 stage workflow. A working directory, where the current files are stored, a staging area and a .git directory. The staging area prepares

your next commit and then it is finally committed. When the staging is complete the change is irreversibly stored. This is a good approach that will be adopted in the final software solution. It will allow incrementally finding changed files, and assessing the need for a new version number to be saved.

The BitTorrent protocol is a mechanism for sharing files between swarms of hosts. As BitTorrent splits files into parts, users start sharing data even before they have received the full file. Each file has a SHA-1 identifier, similar to Git.

If a user wishes to download a file from the swarm, the user downloads a metadata file from the web and locates users sharing the data using a Tracker Server, Distributed Hash Table (DHT) or Peer Exchange (PEX).

A Tracker server is a centralised store of all current connected users along with how much of the file they hold. This approach is vulnerable to exploitation by authorities as all of the data about a swarm is strored on a single server and as a result cannot be used for the proposed system.

A DHT contacts other known users for information instead of a centralised server. The Mainline DHT as outlined in BEP No.5 is based on the Kademlia protocol that allows for decentralised peer discovery for a particular piece of content.

PEX is a method for two clients to share a subset of their peer lists. Coupled with DHT, PEX removes a vulnerability from the Bittorrent network by allowing fully distributed bootstrapping, tracking and peer discovery

Resilio Sync is an example of a distributed file storage system that utilises the BitTorrent protocol to automatically synchronize folders between a user's systems. It is not a cloud backup solution and not intended as a form of off-site storage. There is no distributed file system and as a result, no redundant data block algorithm adding complexity.

As Resilio Sync uses DHT to transfer data, there is no central authority to manage authentication or log data access attempts.

3 Finding External Nodes

4 Data Allocation on External Nodes

5 Distributed File System

6 Authentication

7 Untrustworthy Hosts & Encryption

8 Conclusion

References

Hunt, J. W. and MacIlroy, M. [1976], *An algorithm for differential file comparison*, Citeseer.

Langford, J. [2001], 'Multiround rsync'.

Muthitacharoen, A., Chen, B. and Mazieres, D. [2001], A low-bandwidth network file system, *in* 'ACM SIGOPS Operating Systems Review', Vol. 35, ACM, pp. 174–187.

Rasch, D. and Burns, R. C. [2003], In-place rsync: File synchronization for mobile and wireless devices., *in* 'USENIX Annual Technical Conference, FREENIX Track', Vol. 100.

Stephen, Z. H., Blackburn, S. M., Kirby, L. and Zigman, J. [2000], Platypus: Design and implementation of a flexible high performance object store., *in* 'In Proceedings of the Ninth International Workshop on Persistent Object Systems', Citeseer.

Tridgell, A., Mackerras, P. et al. [1996], 'The rsync algorithm'.