CoStor, a peer-to-peer distributed backup solution

Robert Phipps

4th Year Project Report
Computer Science
School of Informatics
University of Edinburgh

2020

Abstract

This is an example of `infthesis` style. The file `skeleton.tex` generates this document and can be used to get a "skeleton" for your thesis. The abstract should summarise your report and fit in the space on the first page. You may, of course, use any other software to write your report, as long as you follow the same style. That means: producing a title page as given here, and including a table of contents and bibliography.

Acknowledgements

This project makes extensive use and builds on top of concepts explored in this paper from Paul Anderson and Le Zhang: Fast and secure laptop backups with encrypted de-duplication [1]

Table of Contents

Chapter 1

Solution overview

CoStor is designed as a turnkey solution to the problem of maintaining reliable system backups within an SME with multiple sites, such as a confederation of schools. Instead of using expensive and bandwidth intensive cloud storage services for offsite backup, CoStor is designed to hold a complete local backup on-site within the CoStor server, as well as automatically replicating backup data across a group of federated instances of the server software, ensuring that there is always at least two redundant copies of the backup datastore in two different physical locations.

To simplify networking requirements for deployment, all communication between clients and servers, both locally and between sites, makes use of standard HTTPS requests. This negates the need for complex multi-site VPNs, and simply requires a single TCP port to be forwarded to the server from the internet.

The backup datastore's metadata and directory structures are maintained inside an SQL database and can only be modified over the RESTful HTTP API, reducing attack surface compared to making use of more traditional file transfer methods like FTP, NFS or SSHFS.

All management is completed through a simple web UI, where clients can be configured for one-touch deployment and subsequently monitored, as well as where users can browse the directory trees for each backup ”snapshot” in the event that a file needs to be recovered. A full backup restore can be completed by requesting a restoration archive, whereby the server will build a complete archive of a snapshot, pulling data from its local datastore, or from other sites in the event of a remote restore.

This system makes use of Django for server-side components, a Python web framework with fantastic ORM and enforcement of best-practices.

1.1 Goals of CoStor

Given the target organisations for CoStor, there are some specific goals that need to be targeted during development.

- **Reliable backups**

  As should be very obvious, being a backup solution, CoStor needs to be able to reliably manage and maintain backups for a network. This includes protections such as an ”append-only” API for backup clients, validation of uploaded data, and mitigations against the most common reasons a backup may be called upon such as accidental deletion, user errors, hardware failure and ransomware style attacks.

- **Simple restores**

  As this system is targeted at small organisations which may not have their own full-time IT support staff, restoring from backups should be straightforward for an end-user. This is achieved through the use of a self-service web UI.

- **Robust security and audit logs**

  Backups almost always contain confidential information, so CoStor needs to be able to manage permissions on a granular user-by-user basis. It also needs to include audit logging for all operations on the system. Any offsite storage and ”data in flight” needs to be strongly encrypted to protect confidentiality.

- **Low maintenance**

  Systems tend to be forgotten about, and in the case of backups, often you only notice something hasn’t been working once you need to restore something[1], so CoStor needs to include built in maintenance and scheduled self-tests to ensure that the system is ready when the user needs it most.

- **Simple deployment**

  Again, CoStor needs to be deployable by inexperienced IT support staff without prior knowledge of network filesystems, command line interfaces or web development. This can be achieved by making use of clever packaging and deployment strategies such as Docker for server components and zero-touch installation scripts for backup clients. By making use of standard and well understood protocols such as HTTP(S) for communication between components, compatibility with most network architectures should be maintained, without the requirement for complex network share configurations, multi-site VPNs and authentication systems.

- **Centralised management and monitoring**

[1]GitLab found this to their cost in 2017 after discovering their backups hadn’t been running for some time: `https://techcrunch.com/2017/02/01/gitlab-suffers-major-backup-failure-after-data-deletion-incident/`

As this is designed to be deployed over a large number of client PCs, ensuring configuration is correct could be challenging. As such, CoStor will include configuration of backup clients from the server management panel, and all clients will pull down configuration automatically from this central repository. Backup logs will also be pushed to the server so that an administrator can monitor their entire estate from a single place.

- **Distributed and fault-tolerant file stores**

  Leaving the best to last, CoStor's standout feature will be that backups can be automatically replicated between federated instances of the CoStor server, over a zero-configuration HTTPS link. The system should be able to recover from the loss of a server without any data loss, and allow restoration of data originating from any site from any of the remaining instances of CoStor within the network.

1.2 Limitation of scope for the purposes of this project

As this project is to completed by an individual over the course of a single academic year, the scope does unfortunately have to be limited somewhat. The primary goal is to complete a ”Beta” release of the server application, with cross-site replication, web UI and ability to restore data, along with a barebones backup client to allow the system to be demonstrated.

Managing filesystem metadata and ensuring consistent snapshots is a considerable problem in its own right.

1.3 Exploration of existing solutions

There are many packages in existence which incorporate a subset of the features targeted by this project, however most either focus on the synchronisation features with some limited support for file history, and no robust backup capability, whereas others rely on cloud storage infrastructure as the backend, distributed datastore which either necessitates the use of a commercial provider such as AWS S3[2] or BackBlaze B2. Both of these greatly increase cost and introduce a reliance on a third party.

A number of existing products have been selected as they offer the closest functionality to that targeted by CoStor, and are explored here:

1.3.1 Syncthing

Syncthing is a service targeted at consumers who want a self-hosted alternative to commercial cloud storage and synchronisation services such as Dropbox, Google Drive and OneDrive. The agent software can be configured to sync any file changes between two or more devices, allowing for access anywhere, with a form of georeplication. It also requires fairly minimal network configuration, just needing a pair of ports to be opened on at least one of the nodes to allow discovery of other agents.

A very large distinction has to be made in the fact that ”Syncthing is a continuous file synchronization program”[2], in that it isn’t designed to create restorable snapshots of your data, and therefore is completely unsuitable for robust *backup* of important information.

More information is available at `https://syncthing.net` [2]

1.3.2 UrBackup

UrBackup is closer to CoStor in its goals, as is specifically built to be used as a backup system. It can auto-discover agents on the network, and begin incremental backups to its server software. It also supports full image backups of NTFS formatted drives with bare metal restore. UrBackup has many of the features targeted by CoStor, including a simple web interface for management, however it does not include any built-in support for georeplication.

More information is available at `https://www.urbackup.org/index.html` [3]

1.3.3 Hermes

Hermes is an ”open-source redundant distributed storage network”[4]. Although it doesn’t come pre-packaged with components allowing it to be used as a turnkey backup

[2] Amazon Web Services’ bucket storage solution

system, it is worth exploring as it does specifically target the geo-replication features for the purposes of backup that CoStor is looking to integrate. It promises fast, encrypted and seamless replication and sharding of data across nodes in the network, making use of LZMA compression to increase performance.

This would appear to be a very promising option to integrate as the backend storage for CoStor, however it looks like the project is very much stale, with the last commits being made in late 2014. As such, its codebase is somewhat limited in utility, given its lack of ongoing maintenance. It is also written in Go, with limited documentation, which is not a language that I am familiar enough with to begin work on reviving.

More information is available at `https://github.com/Hermes/hermes` [4]

1.3.4 Bacula

Bacula is an open-source and very mature backup framework, with tools to allow a multitude of network configurations. Unfortunately its flexibilty does result in the software being complex to configure. CoStor is targeting small organisations with limited in-house IT support capacity, so this would likely be too complex to deploy without the assistance of external contractors. Bacula is also available in an "enterprise" edition[**?**], which includes support as part of the subscription cost, however this version is both closed-source and not inconsiderably expensive.

More information is available at `https://www.bacula.org` [**?**]

1.3.5 Amanda

Amanda is another backup-specific solution, again with a "community edition" being accompanied by a commercially supported "enterprise" version of the software.

1.4 Deployment topology

The basic topology of a CoStor network would be as follows:

- **Clients** (many):

  The CoStor client software is installed on any systems which are to be backed up. It communicates over the local network to the site's local instance of the CoStor server.

- **Servers** (one per site/network):

  There should be one instance of CoStor server on the internal private network of any site that has clients to be backed up. There could be multiple instances running in one local network space, but they would operate as separate "sites" within the software.

Insert diagram of topology, and describe how data is distributed across the members of the CoStor network.

1.5 Specimen use case

Define the example setup of being used in a confederation of schools.

Chapter 2

Backup datastore implementation

Chapter 3

Client implementation and API

Chapter 4

Multi-site replication

Chapter 5

Testing and evaluation

Bibliography

[1] P. Anderson and L. Zhang, "Fast and secure laptop backups with encrypted de-duplication," in *Proceedings of the Large Installations Systems Administration (LISA) Conference*. Berkeley, CA: Usenix Association, November 2010. [Online]. Available: http://homepages.inf.ed.ac.uk/dcspaul/publications/lisa2010.pdf

[2] Syncthing website. [Online]. Available: https://syncthing.net

[3] Urbackup website. [Online]. Available: https://www.urbackup.org/index.html

[4] (2020, January) Hermes project on github. [Online]. Available: https://github.com/Hermes/hermes