Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και
Υπολογιστών

Σχεδιασμός και Υλοποίηση ενός Φορητού Μηχανισμού Συγχρονισμού Αρχείων σε Περιβάλλον Αποθηκευτικού Νέφους

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΒΑΣΙΛΕΙΟΣ ΓΕΡΑΚΑΡΗΣ

Supervisor : Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.

Αθήνα, Αύγουστος 2015

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και
Υπολογιστών

Σχεδιασμός και Υλοποίηση ενός Φορητού Μηχανισμού Συγχρονισμού Αρχείων σε Περιβάλλον Αποθηκευτικού Νέφους

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΒΑΣΙΛΕΙΟΣ ΓΕΡΑΚΑΡΗΣ

Supervisor : Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 30η Αυγούστου 2015.

....	....	....
Νεκτάριος Κοζύρης Καθηγητής Ε.Μ.Π.	Νικόλαος Παπασπύρου Αν. Καθηγητής Ε.Μ.Π.	??? Καθηγητής Ε.Μ.Π.

Αθήνα, Αύγουστος 2015

....

Βασίλειος Γερακάρης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

-Περίληψη-

Λέξεις κλειδιά

Αποθηκευτικό Νέφος, Συγχρονισμός αρχείων

Abstract

Abstract

Key words

Cloud storage, File synchronisation

Ευχαριστίες

Η παρούσα διπλωματική εργασία σημαίνει την ολοκλήρωση ενός σημαντικού κεφαλαίου της ακαδημαϊκής μου πορείας. Θα ήθελα στο σημείο αυτό να ευχαριστήσω ορισμένους ανθρώπους που με βοήθησαν στη διαδρομή αυτή. < Ευχαριστίες >

Βασίλειος Γερακάρης,

Αθήνα, 30η Αυγούστου 2015

Η εργασία αυτή είναι επίσης διαθέσιμη ως Τεχνική Αναφορά CSD-SW-TR-42-14, Εθνικό Μετσόβιο Πολυτεχνείο, Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών, Εργαστήριο Υπολογιστικών Συστημάτων, Αύγουστος 2015.

Contents

List of Tables

List of Figures

Chapter 1

Εισαγωγή

1.1 Κίνητρο

Μπλα μπλα μπλα, μπλα μπλα, μπλα μπλα μπλα, μπλα μπλα μπλα μπλα, μπλα μπλα μπλα, μπλα μπλα μπλα, μπλα, μπλα μπλα μπλα, μπλα μπλα,

1.2 Κύρια σημεία της εργασίας

Μπλα μπλα μπλα, μπλα μπλα, μπλα μπλα, μπλα, μπλα μπλα μπλα, μπλα μπλα, μπλα, μπλα, μπλα μπλα, μπλα

1.3 Οργάνωση κειμένου

Μπλα μπλα μπλα, μπλα μπλα, μπλα μπλα μπλα, μπλα μπλα μπλα μπλα, μπλα μπλα μπλα, μπλα μπλα μπλα, μπλα, μπλα μπλα μπλα, μπλα μπλα,

1.4 Συνοπτική παρουσίαση του framework

Μπλα μπλα μπλα, μπλα μπλα, μπλα μπλα μπλα, μπλα μπλα μπλα μπλα,

1.5 Συνοπτική παρουσίαση των πειραματικών αποτελεσμάτων

Μπλα μπλα μπλα, μπλα μπλα, μπλα μπλα μπλα, μπλα μπλα μπλα μπλα

Chapter 1

Introduction

1.1 Motivation

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec euismod ante non felis condimentum efficitur. Nunc vel pretium diam.

1.2 Thesis contribution

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec euismod ante non felis condimentum efficitur. Nunc vel pretium diam.

1.3 Chapter outline

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec euismod ante non felis condimentum efficitur. Nunc vel pretium diam.

1.4 Brief description of the framework

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec euismod ante non felis condimentum efficitur. Nunc vel pretium diam.

Chapter 2

Background

2.1 Data Synchronisation

Data synchronisation is the process of establishing consistency among data from a source to a target data storage and vice versa and the continuous harmonisation of the data over time. File Synchronisation (or syncing) is the process of ensuring that files in two or more locations are updated by certain rules. In *one-way file synchronisation*, also called mirroring, updated files are copied from a 'source' location to one or more 'target' locations, but no files are copied back to the source location. In *two-way file synchronisation*, updated files are copied in both directions, usually with the purpose of keeping the two locations identical to each other

2.2 File Hosting Service

A file hosting service[14] or cloud storage service, is an Internet hosting service specifically designed to host user files. It allows users to upload files that could then be accessed over the internet from a different computer, tablet, smart phone or other networked device, by the same user or possibly by other users, after a password or other authentication is provided. File hosting services often offer file sync and sharing services, most notable consumer products being Dropbox and Google Drive.

2.3 Application programming interface

Application programming interface (API) is a set of routines, protocols, and tools for building software applications. An API expresses a software component in terms of its operations, inputs, outputs, and underlying types. An API defines functionalities that are independent of their respective implementations, which allows definitions and implementations to vary without compromising the interface. APIs often come in the form of a library that includes specifications for routines, data structures, object classes, and variables. In other cases, such as REST services, an API is simply a specification of remote calls exposed to the API consumers.

2.3.1 Representational state transfer

Representational State Transfer (REST) is a software architecture style for building scalable web services[16] REST gives a coordinated set of constraints to the design of components in a distributed hypermedia system that can lead to a more performant and maintainable architecture[17]. RESTful systems typically, but not always, communicate over the Hypertext Transfer Protocol with the same HTTP verbs (GET, POST, PUT, DELETE, etc.) which web browsers use to retrieve web pages and to send data to remote servers.

2.4 OpenStack

OpenStack[4] is a free and open source cloud operating system that controls large pools of compute, storage, and networking resources throughout a data centre. Users can manage those resources through a web-based dashboard, command-line tools, or a RESTful API. It is primarily being deployed as an infrastructure-as-a-service (IaaS). OpenStack offers support for both Object Storage and Block Storage.Object Storage is ideal for cost effective, scale-out storage. It provides a fully distributed, API-accessible storage platform that can be integrated directly into applications or used for backup, archiving and data retention. Block Storage allows block devices to be exposed and connected to compute instances for expanded storage, better performance and integration with enterprise storage platforms.

2.4.1 Object Storage - Swift

OpenStack Object Storage (Swift) is a scalable redundant storage system. Objects and files are written to multiple disk drives spread throughout servers in the data centre, with the OpenStack software responsible for ensuring data replication and integrity across the cluster. Because OpenStack uses software logic to ensure data replication and distribution across different devices, inexpensive commodity hard drives and servers can be used.

2.5 Synnefo

Synnefo[10] is an open source cloud stack, which offers Compute, Network, Image, Volume and Storage services, similar to the ones offered by OpenStack. Synnefo is written in Python and to improve third-party compatibility, it exposes the OpenStack APIs to users[11]. It is the software used for ~okeanos[9], an Infrastructure as a Service (IaaS) cloud service, provided by the Greek Research and Technology Network (GRNET) for the Greek Research and Academic Community. ~okeanos offers a virtual compute/network service called Cyclades as well as a virtual storage service, called Pithos+.

2.5.1 Pithos+

Pithos+ is the Virtual Storage service of ~okeanos, featuring cloud storage as well as file synchronisation and sharing services. Files stored in Pithos+ are accessible via the web UI or with the client software, which exists for Windows, MacOS and iOS systems. Linux users can access the files using *kamaki*, the command line client for ~okeanos resources. It is powered by the Pithos (File/Object Storage) services of synnefo.

2.6 Amazon Web Services

Amazon Web Services (AWS) is a collection of remote computing services, also called web services, that make up a cloud-computing platform offered by Amazon.The most central and well-known of these services arguably include Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Simple Storage Service (Amazon S3).

2.6.1 Amazon S3

Amazon S3[5] (Simple Storage Service) is an online file storage web service offered by Amazon Web Services. Amazon S3 provides storage through web services interfaces (REST, SOAP, and BitTorrent). It provides developers and IT teams with secure, durable, highly-scalable object storage, for a wide variety of use cases including cloud applications, content distribution, backup and archiving, disaster recovery, and big data analytics. Amazon S3 stores arbitrary objects up to 5 terabytes in size, each accompanied by up to 2 kilobytes of metadata. Objects are organised into buckets (each owned by an AWS account), and identified within each bucket by a unique, user-assigned key.

2.7 Dropbox

Dropbox offers cloud storage, file synchronisation, personal cloud, and client software. Dropbox synchronises a directory so that it appears to be the same (with the same contents) regardless of which computer is used to view it. Files placed in this folder are also accessible via the Dropbox website. Dropbox is multi-platform, and is working on all major desktop and mobile OS. Originally, both the server and client software were primarily written in Python; since 2013 Dropbox has began migrating its backend infrastructure to Go. Dropbox depends on rsync, ships the librsync binary-delta library (which is written in C) and utilises delta encoding technology. When a file in a user's Dropbox folder is changed, Dropbox only uploads the pieces of the file that are changed when synchronising, when possible. It currently uses Amazon's S3 storage system to store the files. Dropbox also provides a technology called LAN sync, which allows computers on a local area network to securely download files locally from each other instead of always hitting the central servers, improving syncing speed.

2.8 Google Drive

Google Drive is a file storage and synchronisation service created by Google. The Google Drive client communicates with Google Drive to cause updates on one side to be propagated to the other so they both normally contain the same data. Google Drive is also multi-platform, though there is no official Linux client software. The implementation and syncing algorithm underlying Google Drive are mostly unknown, due to the software being closed source.

2.9 ownCloud

ownCloud[6] is a suite of client-server software for creating file hosting services and using them. ownCloud allows synchronisation of directories, similar to the way Dropbox operates. It is a free and open-source software and is multi-platform, with clients available for all major desktop and mobile OS. The server software is written in PHP and JavaScript languages. ownCloud's desktop syncing client depends and ships with csync[7], which is a lightweight utility to synchronise files between two directories on a system or between multiple systems. The software does not currently support delta-sync (syncing only file changes).

2.10 Hash function

A hash function is any function that can be used to map digital data of arbitrary size to digital data of fixed size. The values returned by a hash function are called hash values, hash codes, hash sums, or simply hashes. Hash functions accelerate table or database lookup by detecting duplicated records in a large file. Good hash functions should satisfy certain properties. Firstly, the function must be deterministic, meaning that for a given input value it must always generate the same hash value. A good hash function should map the expected inputs as evenly as possible over its output range - this property is called uniformity. This property minimises the chance of hash collisions (pairs of inputs that are mapped to the same hash value). For hash functions used in data search, it is desirable that the output of the function has fixed size, measured in bits.

2.10.1 SHA-256

The Secure Hash Algorithm is a family of cryptographic hash functions published by the National Institute of Standards and Technology (NIST) as a U.S. Federal Information Processing Standard (FIPS). *SHA-2* is a family of two similar hash functions, with different block sizes, known as SHA-256 and SHA-512. They differ in the word size, with SHA-256 using 32-bit words where SHA-512 uses 64-bit words and have a hash value (digest) of 256 and 512 bits, respectively. They are considered to be secure and collision resistant, with SHA-256 having a collision probability of about $4.3 * 10^{-60}$ when digesting one billion (10^9) different messages.

2.10.2 xxhash

xxHash[8] is a non-cryptographic hash function designed around speed. It successfully completes the SMHasher test suite which evaluates collision, dispersion and randomness qualities of hash functions. xxHash's digests can be returned as bytes, integers or hex numbers and can be of 32 or 64 bit size.

2.11 ETag

The ETag or Entity Tag is a string identifier assigned to a resource, usually a file or block, that describe exactly one specific version of it. Whenever there is a change on the file, the ETag should be changed as well.ETags can be used for optimistic concurrency control[2], which is a method where shared data resources are being used without a transaction acquiring locks on them; before the transaction commits, it verifies that no other transaction has modified the data, otherwise it rolls back the changes. SHA-256 digests are commonly used as ETag identifiers, since the algorithm is secure and collision resistant, in contrast to MD5 digests, where collisions can be computed.

2.12 Containers

TODO: Containers

2.13 Database

A database-management system (DBMS) [15] is a collection of interrelated data and a set of programs to access those data. The collection of data, is referred to as the *database.*

2.13.1 Relational database

A relational database uses a collection of tables to represent both data and the relationships among those data. Each table represents a relation variable, has multiple columns and each column has a unique name. The columns define the attributes and each row is an instance of the variable. The rows are uniquely identified by a certain attribute, called the *primary key.*

2.13.2 Transactional database

A transactional database is one in which all changes and queries have the **ACID** [1] (Atomicity, Consistency, Isolation, Durability) set of properties. Those properties guarantee that database transactions are processed reliably even in the event of a transaction interruption.

Atomicity

The atomicity property ensures that in a transaction, a series of database operations either all occur, or nothing occurs. An atomic system must guarantee atomicity in every situation, including power failures, errors, and crashes. To the outside world, a committed transaction appears (by its effects on the database) to be indivisible (”atomic”), and an aborted transaction does not happen at all.

Consistency

The consistency property ensures that any transaction will bring the database from one valid state to another. Any data written to the database must be valid according to all defined rules, including constraints, cascades, triggers, and any combination thereof. This does not guarantee correctness of the transaction in all ways the application programmer might have wanted (that is the responsibility of application-level code) but merely that any programming errors cannot result in the violation of any defined rules.

Isolation

The isolation property ensures that the concurrent execution of transactions results in a system state that would be obtained if transactions were executed serially. Providing isolation is the main goal of concurrency control. Depending on concurrency control method, the effects of an incomplete transaction might not even be visible to another transaction.

Durability

Durability means that once a transaction has been committed, it will remain so, even in the event of power loss, crashes, or errors. In a relational database, once a group of SQL statements execute, the results need to be stored permanently (even if the database crashes immediately thereafter). To defend against power loss, transactions (or their effects) must be recorded in a non-volatile memory.

2.13.3 Structured Query Language

Structured Query Language (SQL) is a special-purpose programming language designed for managing data held in a relational database management system (RDBMS). The most important elements of the SQL language are the *Statements*, which may have a persistent effect on schemata and data, or may control transactions, program flow, connections, sessions, or diagnostics and the *Queries*, which retrieve data from the database, based on specific criteria.

2.13.4 SQLite

SQLite[3] is an open source, cross-platform RDBMS contained in a C programming library that offers a full SQL implementation. In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program and reads and writes directly to ordinary disk files. SQLite is transactional and as such, ACID-compliant. SQLite has a small code footprint and is widely used on memory and disk space constrained cases.

Chapter 3

Design & Implementation

3.1 Syncing Algorithm

3.1.1 Known algorithms

The process of synchronising two filesystem trees (henceforth called “A” and “B”) can be complex to perform correctly, if some needed information is missing. Differences in files can be detected by comparing their hash digests, which is often used as an ETag. While this is the most secure and reliable way to detect changes, computing the hash digest of a file, especially when using a cryptographically secure function) is a computationally expensive procedure that takes significant time for large files. The last modification time is a fast and cheap way to detect file changes, but since it is a property that can be manipulated by software, improper or malicious manipulation can result in failure to detect changes.

First of all, history data for those two trees are very important, as illustrated in the following example. There are generally three cases when synchronising two trees:

1. File exists on both and is identical

2. File exists on both and is different

3. File exists on A but not on B (or vice-versa)

Case 1 is easily handled, since there is no action to be taken. Without history information, the other two cases would require user input in order to be handled correctly. In case 2, the files should be merged, but cannot be done automatically - and asking regular users how to merge files is undesirable. In case 3, the file might be a newly created, or a recently deleted one and should be copied to tree B or deleted from A, respectively. While the safe choice is to assume the file is new and copy it to tree B, it is often **not** the right thing to do. It is therefore clear that without file history data the syncing algorithm makes wrong assumtions and fails to handle correctly the most common cases.

With file history present in the form of metadata, a more proper synchronisation is achievable. By checking what changes occured and when between the times T_1 and T_2, it is possible to determine the action that should be taken, based on table 3.1:

File A	File B	Action
No Change	No Change	No Action
Created (ETag = J)	Created (Etag = J)	No Action
Created (ETag = J)	Created (Etag = K)	*Merge**
Deleted	Deleted	No Action
Deleted	No Change	Delete B
Modified	No Change	Update B
Modified (ETag = J)	Modified (ETag = K)	*Merge**

Table 3.1: Syncing actions based on file states

[*] In this table, *Merge* refers to a situation where files A and B become identical. One way this can be achieved is by generating diffs and patching the files, requiring user input if a conflict arises - this is the way most Version Control Systems (VCS), like git and Mercurial, work. A different way, that requires no immediate user interaction is to accept one file version (e.g. File A) and propagating its changes to all other trees, while also renaming the conflicting files, so the conflicts can be manually merged later.

For any given time, detecting what happened between the times T_1 and T_2, is straightforward, and described in 3.2:

Time T_1	**Time T_2**	**Change**
Does not Exist	Exists	Created
Exists	Does not Exist	Deleted
Exists (ETag = J)	Exists (Etag = J)	No Change
Exists (Etag = J)	Exists (Etag = K)	Modified

Table 3.2: File change detection between two points in time

While this algorithm is significantly better than the previous one, it still has limitations, the most important of which is the failure to detect renames. This can become possible by comparing file digests, but doing so for all files in a directory or for very large files is computationally expensive and slows down the sync process. An even harder problem is the detection of a file that has been renamed and modified, hence having a different hash digest than the original one, a case most syncing algorithms fail to handle efficiently.

3.1.2 Proposed Algorithm

At this point we propose a new sychronisation algorithm, one which is efficient, fast and reliable. We assume a service that uses a central metadata server, which maintains information about each version of an uploaded object. We also assume the usage of a local state database, henceforth called **StateDB**, which locally stores the metadata of all files in the local directory, as they were during the last synchronisation with the server. More specifically, a path hash is used as a file identifier, and other important metadata saved are the file name ("path"), the inode, last modification time ("modtime"), and the file's hash digest("Etag"). Now the problem of reconcilation between the local directory replicas ("Local") and the remote server replicas ("Remote") can be now handled in three steps.

Step 1: Detect updates from Local Directory

For each file in the local directory, the necessary action can be derived from the decision tree in Figure 3.1.

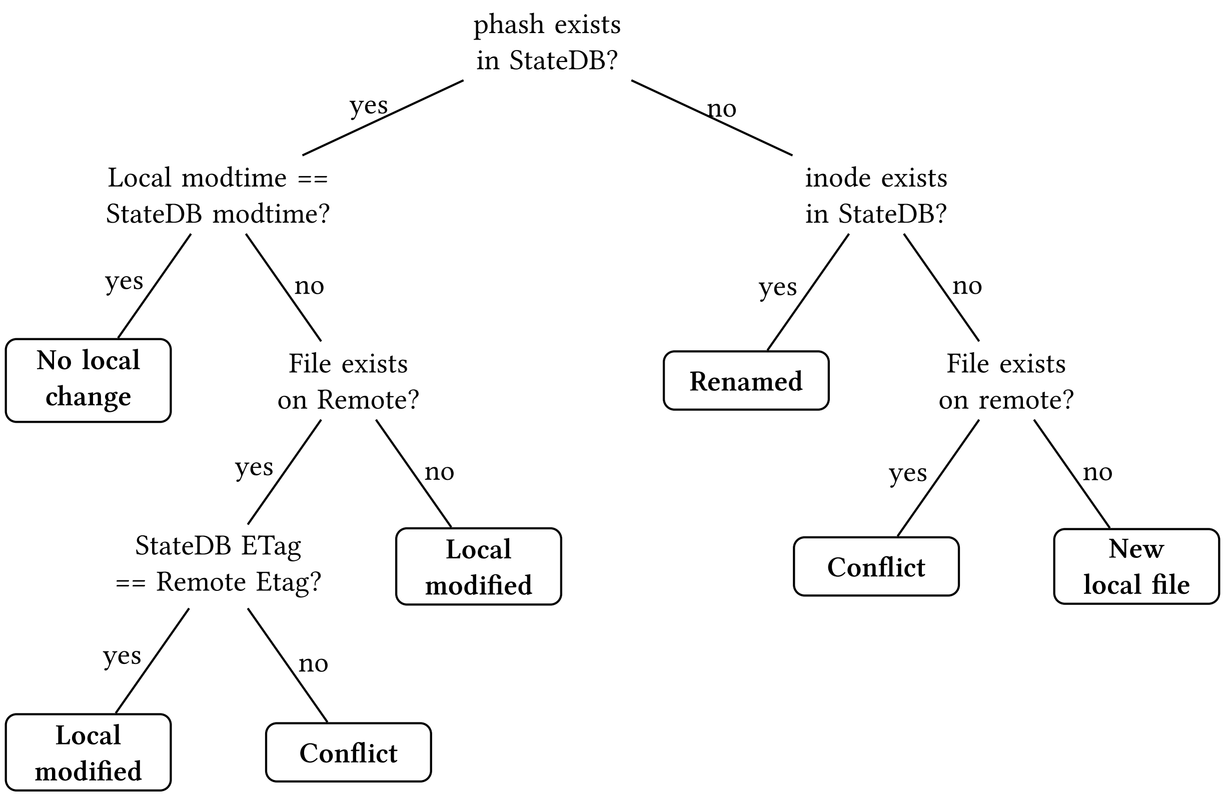
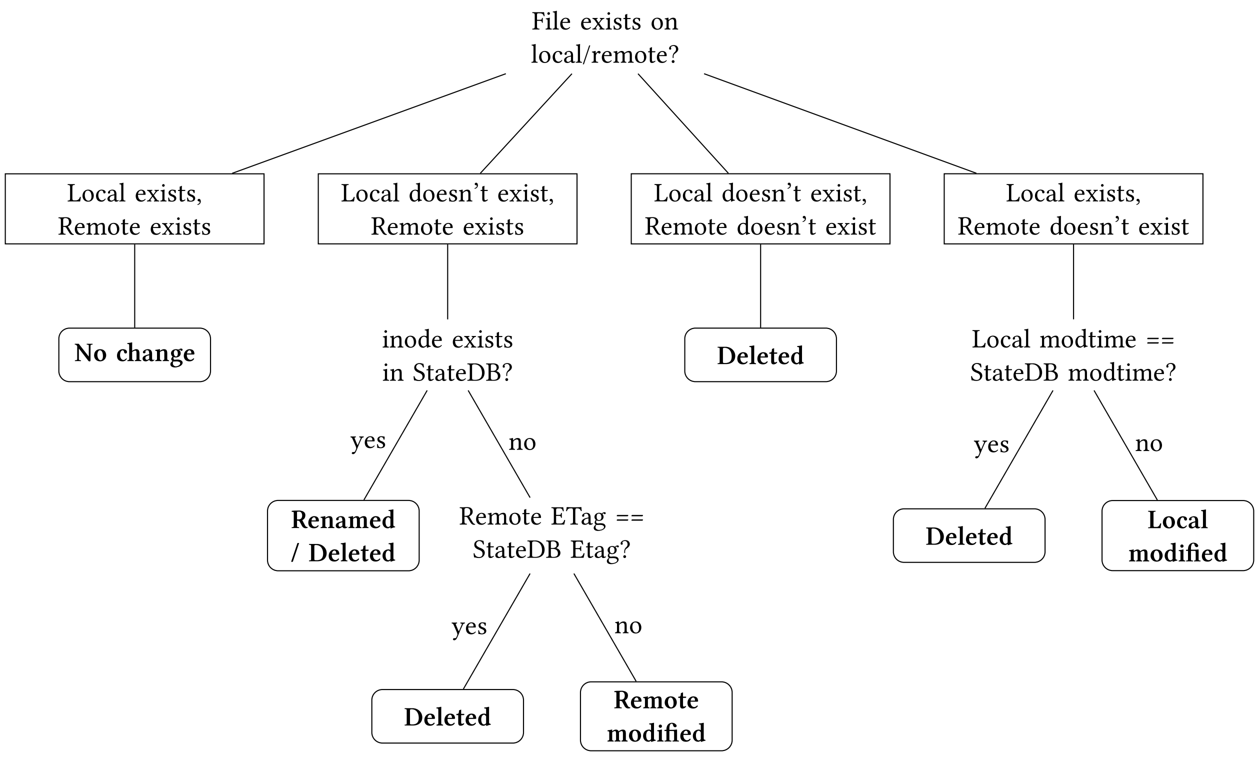


Figure 3.1: Decision tree for local directory files

Step 2: Detect updates from StateDB

For each entries in the StateDB, the necessary action can be derived from the decision tree in Figure 3.2.



File exists on local/remote?
Local exists, Remote exists
Local doesn't exist, Remote exists
Local doesn't exist, Remote doesn't exist
Local exists, Remote doesn't exist
No change
inode exists in StateDB?
Deleted
Local modtime == StateDB modtime?
yes
no
Renamed / Deleted
Remote ETag == StateDB Etag?
yes
no
Deleted
Remote modified
yes
no
Deleted
Local modified

Figure 3.2: Decision tree for StateDB entries

Step 3: Detect updates from Remote Server

TODO: Step3

Important notes on the syncing algorithm

- For update detection between Local and StateDB, we use modtime, since it is faster. For update detection between StateDB and Remote, we use the ETag, since it is the most reliable way and is readily available without hashing the file - it is stored in both cases and available with only a lookup.
- When the file is updated on Local but deleted on Remote, we decide to upload the modified file again, since it is the safe option.
- If conflict is detected, that means that the both the Local file and the Remote one have been updated since the last sync was completed. In that case we propose to rename the local one to "<filename>-conflicting_copy.<extension>" and download the remote one. Filenames ending in "-conflicting_copy" are excluded from Step 1 of the syncing algorithm.
- TODO: No Change - StateDB
- TODO: renamed/deleted

3.2 Basic Classes

3.2.1 FileStat

TODO: FileStat

Path hash algorithm selection

TODO: Phash - xxhash64

3.2.2 StateDB

TODO: StateDB

3.2.3 LocalDirectory

TODO: LocalDir

3.2.4 CloudClient

TODO: CloudClient

Chapter 4

Syncer Optimisations

4.1 Query queuing

TODO: Query queue + Benchmarks

4.1.1 Benchmarks

bench

4.2 Directory monitoring

TODO: Watchdog + benchmarks

4.2.1 Benchmarks

bench

4.3 Local block storage

TODO: Local block storage + benchmarks

4.3.1 Benchmarks

bench

Chapter 5

Comparisons with existing software

5.1 Rsync

TODO: Rsync

5.2 OneDrive

TODO: Onedrive

5.3 Dropbox

TODO: Dropbox

5.4 Google Drive

TODO: Google Drive

Chapter 6

Future Work

6.1 File Storage - FUSE

TODO: Filesystem in Userspace

6.2 Peer-to-peer syncing with direct L2 frame exchange

TODO: Copy idea from Dropbox

Bibliography

[1] T. Haerder and A. Reuter, "Principles of transaction-oriented database recovery," *ACM Comput. Surv.*, vol. 15, pp. 287–317, Dec. 1983.

[2] H. Kung and J. T. Robinson, "On optimistic methods for concurrency control," *ACM Transactions on Database Systems*, vol. 6, June 1981.

[3] "About sqlite." https://sqlite.org/about.html. [Online, accessed July 2015].

[4] "Openstack: The open source cloud operating system." https://www.openstack.org/software/. [Online, accessed July 2015].

[5] "Amazon s3." https://aws.amazon.com/s3/. [Online, accessed July 2015].

[6] "Owncloud." https://owncloud.org/faq. [Online, accessed August 2015].

[7] "csync." https://www.csync.org/about/. [Online, accessed August 2015].

[8] Y. Collet, "xxhash." https://github.com/Cyan4973/xxHash. [Online, accessed August 2015].

[9] "~okeanos." https://okeanos.grnet.gr/about/what/. [Online, accessed August 2015].

[10] "Synnefo." https://www.synnefo.org/about/. [Online, accessed August 2015].

[11] "Synnefo rest api guide." https://www.synnefo.org/docs/synnefo/latest/api-guide.html. [Online, accessed August 2015].

[12] "Wikipedia: Openstack." https://en.wikipedia.org/wiki/OpenStack. [Online, accessed July 2015].

[13] "Wikipedia: Amazon s3." https://en.wikipedia.org/wiki/Amazon_S3. [Online, accessed July 2015].

[14] "Wikipedia: File hosting service." https://en.wikipedia.org/wiki/File_hosting_service. [Online, accessed July 2015].

[15] H. K. Abraham Silberschatz and S. Sudarshan, *Database System Concepts.* McGraw-Hill Higher Education, 6th ed., 2010.

[16] R. T. Fielding and R. N. Taylor, "Principled design of the modern web architecture," in *Proceedings of the 22nd international conference on Software engineering*, ICSE '00, pp. 407–416, 2000.

[17] R. T. Fielding, *Architectural Styles and the Design of Network-based Software Architectures.* PhD thesis, University of California, Irvine, 2000.

[18] A. Tridgell, *Efficient Algorithms for Sorting and Synchronization.* PhD thesis, The Australian National University, Irvine.