TimeFuseDB - A temporal file system

Aniket Ray

Department of Computer Science
Tandon School of Engineering
New York University
`aniket.ray@nyu.edu`

Advisor: Dr. Kamen Yotov
Adjunct Professor
NYU Tandon School of Engineering

April 23, 2025

About Me

Professional Timeline

May, 2025	Graduate from NYU*
June 2022 - Aug 2023	Worked as a Software Engineer at *Oracle* on Cloud Infrastructure
May 2022	Graduated with a *B.Tech* degree from *NIT Durgapur*
Jun 2021 – Aug 2021	Worked as a Student Developer at *Google* MediaPipe team
2019 - 2022	Worked as an Independent Contractor for several clients

About Me

Areas of Interests

- Low-level systems
- High Performance Computing
- Operating Systems

Outline

1. Introduction
    - What is `TimeFuseDB`?
    - Content Addressable Storage
2. System Architecture & Implementation
    - Crawler
    - FUSE (File System in User Space)
3. Implementation
4. Live Demo
5. Use Cases
6. Future Works

Introduction

What is TimeFuseDB?

TimeFuseDB is a file system architecture that integrates content-addressable storage with temporal versioning capabilities through a virtual file system interface.

End Result?

We can do: `cd-<TIMESTAMP>` on the file system on a git repository and look how the file structure looked like at that very exact moment.

Content Addressable Storage

Content-addressable storage (CAS) is a way to store information so it can be retrieved based on its content, not its name or location.

- hash the file with hashing algorithms (XXH128 used here)
- store the hash
- retrieve the hash based on custom logic

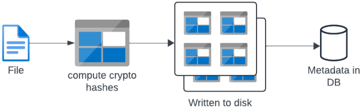


Figure: CAS

Content Addressable Storage

Advantages of CAS:

- Unique Identification of Data and data immutability
- Efficient Storage with Deduplication

Content Addressable Storage

- A notable use is in git source control system.

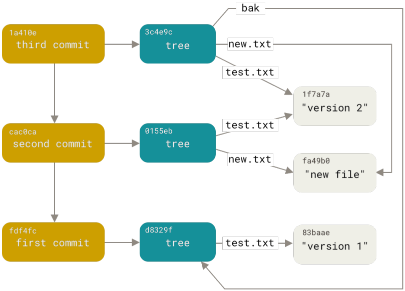


Figure: git data model

System Architecture & Implementation

Crawler

- A custom crawler designed to traverse the commit history and recursively process directory structures
- Leverages the libgit2 library for interacting with git repositories
- Uses the xxHash library for rapid hashing of files.

APIs

`getattr`, `read`, `readdir` are some APIs provided by FUSE used in this project.

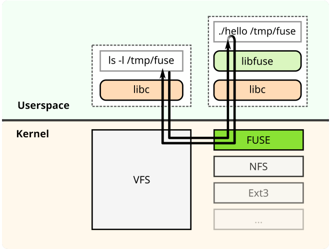
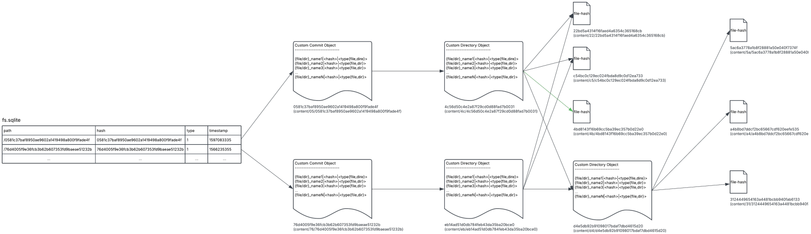


Figure: FUSE Arcitecture

Implementation



fs.sqlite

path	hash	type	timestamp
/0581c37baf8950ae9602a1419498a800f9fade4f	0581c37baf8950ae9602a1419498a800f9fade4f	1	1597083335
/76d4005f9e36fcb3b62b607353fd9baeae51232b	76d4005f9e36fcb3b62b607353fd9baeae51232b	1	1566235355
...	...	...	...

```
Custom Commit Object
--------------------------

{file/dir}_name1|<hash>|<type{file,dire}>
{file/dir}_name2|<hash>|<type{file,dir}>
{file/dir}_name3|<hash>|<type{file,dir}>
...
{file/dir}_nameN|<hash>|<type{file,dir}>
```

0581c37baf8950ae9602a1419498a800f9fade4f
(content/05/0581c37baf8950ae9602a1419498a800f9fade4f)

```
Custom Directory Object
--------------------------

{file/dir}_name1|<hash>|<type{file,dire}>
{file/dir}_name2|<hash>|<type{file,dir}>
{file/dir}_name3|<hash>|<type{file,dir}>
...
{file/dir}_nameN|<hash>|<type{file,dir}>
```

4c56d50c4e2a87f29cd0d88fad7b0031
(content/4c/4c56d50c4e2a87f29cd0d88fad7b0031)

```
Custom Commit Object
--------------------------

{file/dir}_name1|<hash>|<type{file,dire}>
{file/dir}_name2|<hash>|<type{file,dir}>
{file/dir}_name3|<hash>|<type{file,dir}>
...
{file/dir}_nameN|<hash>|<type{file,dir}>
```

76d4005f9e36fcb3b62b607353fd9baeae51232b
(content/76/76d4005f9e36fcb3b62b607353fd9baeae51232b)

```
Custom Directory Object
--------------------------

{file/dir}_name1|<hash>|<type{file,dire}>
{file/dir}_name2|<hash>|<type{file,dir}>
{file/dir}_name3|<hash>|<type{file,dir}>
...
{file/dir}_nameN|<hash>|<type{file,dir}>
```

eb14ad51d0db784feb43da35ba20bce0
(content/eb/eb14ad51d0db784feb43da35ba20bce0)

22bd5a4314f16faed4a6354c365168cb
(content/22/22bd5a4314f16faed4a6354c365168cb)

c54bc0c129ec024fbda8d9c0d12ea733
(content/c5/c54bc0c129ec024fbda8d9c0d12ea733)

4bd8143f16b69cc5ba39ec357b0d22e0
(content/4b/4bd8143f16b69cc5ba39ec357b0d22e0)

```
Custom Directory Object
--------------------------

{file/dir}_name1|<hash>|<type{file,dire}>
{file/dir}_name2|<hash>|<type{file,dir}>
{file/dir}_name3|<hash>|<type{file,dir}>
...
{file/dir}_nameN|<hash>|<type{file,dir}>
```

d4e5db92b91098017bdaf7dbd4615d20
(content/d4/d4e5db92b91098017bdaf7dbd4615d20)

5ac6a3778a1b8f28881a50e040f7374f
(content/5a/5ac6a3778a1b8f28881a50e040f

a4b8bd7ddcf2bc65667cdf620eefe535
(content/a4/a4b8bd7ddcf2bc65667cdf620e

3124449654163a4481bcbb940fab6133
(content/31/3124449654163a4481bcbb940f

Figure: Internal File Structure

Live Demo

Use Cases

- User-Friendly Temporal data Navigation
- Management of Financial Configurations
- Historical Securities Data Navigation

Future Works

- Enhancing Configurability and Usability and making it a Production-Ready, Open-Source Product
- Extending to Support Bi-temporal Data
- Serve git's internal DB

Thank You!