MetaFS - Indexed Metadata based Filesystem

Rene K. Mueller spiritdude@gmail.com

December 5, 2015 (PREVIEW)

Abstract

Ordinary computer file systems, as introduced in the late 1950's[1], structure data according to two main types: folders and files. In order to make data available at the file system level, the filename or folder has to be descriptive and well organized according to a certain schema.

Further, the actual content of a file is not known to the file system, as file formats can only be read and interpreted by third party programs, like a text editor, a document viewer, an image viewer, or a video player.

MetaFS is a file system which does not have hierarchical constraints[10] as do traditional file systems, but it has multiple or flexible views on the files/items based on free definable metadata. The metadata can either be added manually, or, has been automatically extracted by handlers thus making the common metadata of text, image, audio, and video available as well. Additionally all items have a globally unique identifier `uid`, which means that an item can be shared globally and yet kept in sync.

MetaFS has been implemented on top of UNIX[2] file system[3].

1. Introduction

The ordinary computer file system contains the notion of folders and files, whereas folders can contain folders and files again. The problem arises when to order or sort files according to a folder structure or schema, one has to make a decision:

```
2015/
   Client A/
   Client B/
2016/
   Client A/
   Client B/
```

or

```
Client A/
   2015/
   2016/
Client B/
   2015/
   2016/
```

The choice to make is based on the constraints of the folder & file notion - only two keys can be used, and the order matters.

To open these constraints a file[5] has to have more qualifiers or keys, hence free definable metadata.

2. Free Definable Metadata

The basis of MetaFS is that all files or items can have free definable metadata; and for the sake of simplicity the JSON (JavaScript Object Notion)[4] is adapted which allows hierarchical metadata:

```
{
   name: "AA.txt",
   comment: "this is a test",
   deeper: {
      sample: "a bit deeper"
   },
   tags: [ 'example', 'simple' ],
   size: 12,
   uid: "83a414d816448c6337d2920c..",
}
```

2.1. Manually Entered Metadata

The user can manually enter metadata such as tags, keywords, ratings, etc, to any item.

2.2. Automatic Created Metadata

Additionally handlers analyze common file formats of text (.txt, .doc, .pdf, .odt etc), images (.png, .jpg, .gif), audio (.m4a, .mp3, .ogg), video (.mp4, .ogv, .webm, .mkv) and extract metadata:

- text: word, unique words, lines, pages, excerpt, etc.
- image: color type (black & white, monochrome, limited color, full color), color theme, width, height, EXIF data (GPS coordinates) etc.
- audio: sampling rate, duration, song title, artist, album, etc.
- video: duration, title, author, width, height, etc.

3. Indexed Metadata

In order to provide flexible views instantly, all metadata is indexed using B-tree[7]/LSM-tree[8] storage engines; this provides fairly fast answers to queries (match, inequality, range) within sub 100ms range.

4. Flexible Views

Once the metadata is fully indexed one can query according to it quickly and provide a flexible view on the items:

4.1. UNIX FS View

As a proof of concept of MetaFS the "hierarchical file system" or "UNIX FS" view has been achieved using the following keys:

- `type`: undefined or 'folder'
- `name`: filename (string)
- `uid`: uid of item (string)
- `parent`: uid of parent item (string)
- `mtime`: modification time (float)
- `ctime`: creation time (float)
- `atime`: access time (float)

whereas the root (top-level) of the file system is referenced as `parent: 0`.

4.2. Timeline View

The timeline is shown via `ctime` or `mtime` sort view, and this view is obtained as instantly as the "UNIX FS" view.

4.3. Various Views

All metadata is available to sort and view the dataset of items:

- sort by size
- sort by MIME type
- sort by unique words (applies just to texts)
- sort by unique colors (applies just to images)
- sort by geographical location (applies to GPS tagged items like photos)
- etc.

In this sense, an item can be viewed on all the angles (keys) which one has specified.

5. Time of Origin (`otime`)

Amongst the known time stamps in UNIX `ctime`: creation time, `mtime`: modification time, `atime`: access time, an additional time is introduced:

`otime` origin time; when the data came to be (media independent):

- text: when the text was originally written (even before it was digitized)
- photo: when the photo was taken, this is often the same as `mtime` when taken by an electronic camera

`otime` is also the only time stamp which covers the actual time of the data, inside or outside of the file system; whereas `ctime`, `mtime`, and `atime` just focus on the time data was handled **in** the file system.

`otime` has been widely neglected and often mistaken by `mtime` (modification time), e.g. once a photo was copied from one media to another and `mtime` was neglected; depending on the program and the file-browser, the date/time the photo was taken could easily be lost as `mtime` was carelessly updated to present time of copying.

By introducing `otime` it ensures semantically that it really means the date/time the data came to be, and `mtime` truly means the time the data was modified in the file system.

6. Unique Identifier (`uid`)

The `uid` as used in MetaFS is globally unique, which allows the identification of files beyond the local file system, and, thus, the cooperating parties can also sync the updates among themselves.

7. Hash Digest (`hash`) & MIME Type (`mime`)

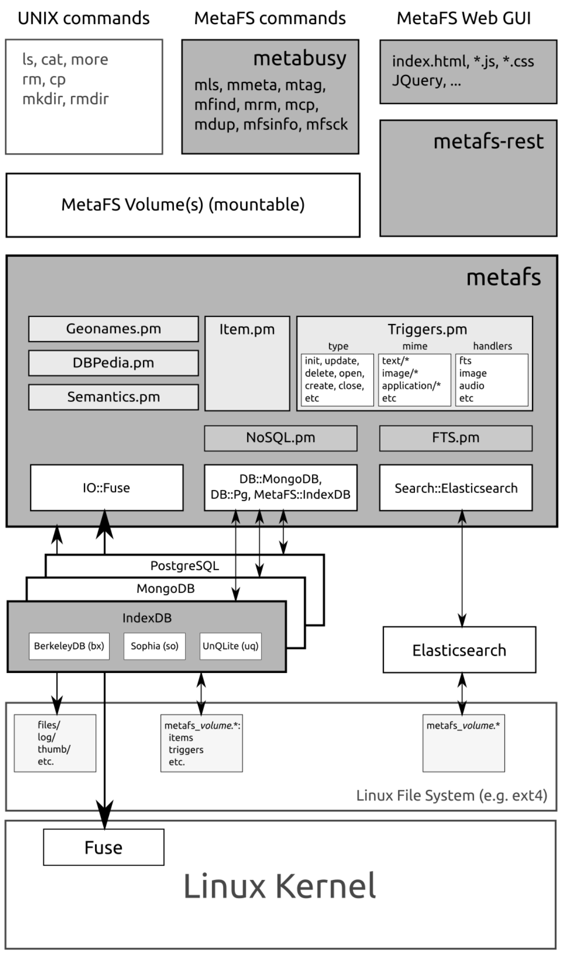
By default, all the files/items content is hash digested[9] (e.g. SHA256) which helps in determining data integrity and in finding duplicates.

Further the MIME type[12] is determined by the actual content (e.g. magic number or probing the first 512 bytes) and only in case of uncertainty is the filename extension (e.g. ".txt") considered.

8. Full Text Index & Search

Full text index & search (FTS) is also available and by default all texts are fully indexed, and so textual content can be looked up instantly as well.

9. Implementation



MetaFS has been implemented on top of the LINUX file system Ext4 using FUSE (Filesytem in Userspace)[6] and a custom NoSQL database called MetaFS::IndexDB which indexes all keys by default, while having a low memory footprint and providing predictable response time of queries.

The full text search part is implemented using Elasticsearch[11].

A fine-grained handler execution facility has been implemented to execute handlers based on MIME type and file system operation events like (CREATE, OPEN, CLOSE, UPDATE, etc), either synchronous (without queue), asynchronous (queued) and additionally prioritized with a certain `nice` level.

The semantic layer as illustrated will be described in another paper.

10. Conclusion

MetaFS as presented provides an in-depth access to the data for the human beyond the simple folder & file notion and makes it more available and comprehendible.

The traditional folder & file notion is usable for the core of the operating system, whereas a MetaFS-based filesystem is more suitable for the user files (e.g. `/home/user` home directory).

References

[1] Barnard III, G. A.; Fein, L. (1958). "Organization and Retrieval of Records Generated in a Large-Scale Engineering Project". Proceedings of the Eastern Joint Computer Conference: 5963.

[2] Ritchie, Thompson (1974), "The UNIX Time-Sharing System"

[3] McKusick, Joy, Leffler, Fabry (1984), "A Fast File System for UNIX"

[4] JSON Definition `http://json.org`

[5] Seltzer, Murphy (2006): "Hierarchical File Systems are Dead"

[6] Linux FUSE (Filesystem in Userspace) `http://fuse.sourceforge.net/`

[7] Bayer, McCreight (1972), "Organization and maintenance of large ordered indexes" (B-tree)

[8] P. O'Neil, Cheng, Gawlick, E. O'Neil (1996), "The log-structured merge-tree (LSM-tree)"

[9] Descriptions of SHA-256, SHA-384, and SHA-512: `http://www.iwar.org.uk/comsec/resources/cipher/sha256-384-512.pdf`

[10] Olson (1993), "The Design and Implementation of the Inversion File System" `http://db.cs.berkeley.edu/papers/S2K-93-28.pdf`

[11] Elasticsearch `http://elasticsearch.org`

[12] IANA: Media Types (MIME Types) `http://www.iana.org/assignments/media-types/media-types.xhtml`