

Software Heritage: Our Software Commons, Forever.

challenges in storing the biggest VCS graph in history

Nicolas Dandrimont

Inria, Software Heritage

27 September 2017

Kernel Recipes — Paris, France



Software Heritage

THE GREAT LIBRARY OF SOURCE CODE



1 The Software Commons

2 Software Heritage

3 Architecture

4 Gory details

5 Community

6 Conclusion

Software source code is *special*

Harold Abelson, Structure and Interpretation of Computer Programs

“Programs must be written for people to read, and only incidentally for machines to execute.”

Quake 2 source code (excerpt)

```
float Q_rsqrt( float number )
{
    long i;
    float x2, y;
    const float threehalfs = 1.5F;

    x2 = number * 0.5F;
    y = number;
    i = * ( long * ) &y; // evil floating point bit level hacking
    i = 0x5f3759df - ( i >> 1 ); // what the fuck?
    y = * ( float * ) &i;
    y = y * ( threehalfs - ( x2 * y * y ) ); // 1st iteration
    // y = y * ( threehalfs - ( x2 * y * y ) ); // 2nd iteration, this
    // can be removed

    return y;
}
```

Net. queue in Linux (excerpt)

```
/*
 * SFB uses two B[1][n] : L x N arrays of bins (L levels, N bins per level)
 * This implementation uses L = 8 and N = 16
 * This permits us to split one 32bit hash (provided per packet by rxhash or
 * external classifier) into 8 subhashes of 4 bits.
 */
#define SFB_BUCKET_SHIFT 4
#define SFB_NUMBUCKETS (1 << SFB_BUCKET_SHIFT) /* N bins per Level */
#define SFB_BUCKET_MASK (SFB_NUMBUCKETS - 1)
#define SFB_LEVELS (32 / SFB_BUCKET_SHIFT) /* L */

/* SFB algo uses a virtual queue, named "bin" */
struct sfb_bucket {
    u16      qlen; /* length of virtual queue */
    u16      p_mark; /* marking probability */
};
```

Len Shustek, Computer History Museum

“Source code provides a view into the mind of the designer.”

Our Software Commons

Definition (Commons)

The **commons** is the cultural and natural resources accessible to all members of a society, including natural materials such as air, water, and a habitable earth. These resources are held in common, not owned privately. <https://en.wikipedia.org/wiki/Commons>

Definition (Software Commons)

The **software commons** consists of all computer software which is available at little or no cost and which can be altered and reused with few restrictions. Thus *all open source software and all free software are part of the [software] commons.* [...]

https://en.wikipedia.org/wiki/Software_Commons

Our Software Commons

Definition (Commons)

The **commons** is the cultural and natural resources accessible to all members of a society, including natural materials such as air, water, and a habitable earth. These resources are held in common, not owned privately. <https://en.wikipedia.org/wiki/Commons>

Definition (Software Commons)

The **software commons** consists of all computer software which is available at little or no cost and which can be altered and reused with few restrictions. Thus *all open source software and all free software are part of the [software] commons.* [...]

https://en.wikipedia.org/wiki/Software_Commons

Source code is *a precious part of our commons*

are we taking care of it?

Software is fragile



A word cloud centered on the slide, featuring terms related to software fragility. The words are arranged in a circular pattern, with 'damage' and 'disaster' being the largest. Other prominent words include 'malicious', 'obsolete', 'deletion', 'format', 'attack', and 'dependencies'. Smaller words like 'media', 'aging', 'tear', 'dangling', 'wear', 'corruption', 'encryption', 'reference', and 'storage' are also visible. The background of the slide features a faint world map and a decorative starburst pattern of colorful triangles on the right side.

damage
disaster
malicious
obsolete
deletion
format
attack
dependencies
media
aging
tear
dangling
wear
corruption
encryption
reference
storage

Like all digital information, FOSS is fragile

- inconsiderate and/or malicious code loss (e.g., Code Spaces)
- business-driven code loss (e.g., Gitorious, Google Code)
- for obsolete code: physical media decay (data rot)

Software is fragile



A word cloud centered on the slide, featuring terms related to software fragility. The most prominent words are 'damage' (large, brown), 'disaster' (large, purple), 'deletion' (large, blue), 'malicious' (medium, brown), 'obsolete' (medium, purple), 'attack' (medium, blue), 'format' (medium, green), 'dependencies' (small, blue), 'reference' (small, blue), 'storage' (small, brown), 'dangling' (small, brown), 'wear' (small, brown), 'corruption' (small, brown), 'encryption' (small, blue), 'aging' (small, blue), 'media' (small, brown), and 'tear' (small, blue). The words are arranged in a circular pattern around a central point.



Like all digital information, FOSS is fragile

- inconsiderate and/or malicious code loss (e.g., Code Spaces)
- business-driven code loss (e.g., Gitorious, Google Code)
- for obsolete code: physical media decay (data rot)

Where is the archive...

where we go if (a repository on) GitHub or GitLab.com goes away?



1 The Software Commons

2 Software Heritage

3 Architecture

4 Gory details

5 Community

6 Conclusion



Software Heritage

THE GREAT LIBRARY OF SOURCE CODE



Our mission

Collect, **preserve** and **share** the *source code of all the software* that is publicly available.

Past, present and future

Preserving the past, enhancing the present, preparing the future.

Our principles

Cultural Heritage



Industry



Research



Education



Software Heritage

Our principles

Cultural Heritage



Industry



Research



Education



Software Heritage

Open approach

- 100% FOSS
- transparency

In for the long haul

- replication
- non profit



1 The Software Commons

2 Software Heritage

3 Architecture

4 Gory details

5 Community

6 Conclusion

Archiving goals

Targets: VCS repositories & source code releases (e.g., tarballs)

We DO archive

- file **content** (= blobs)
- **revisions** (= commits), with full metadata
- **releases** (= tags), ditto
- where (**origin**) & when (**visit**) we found any of the above

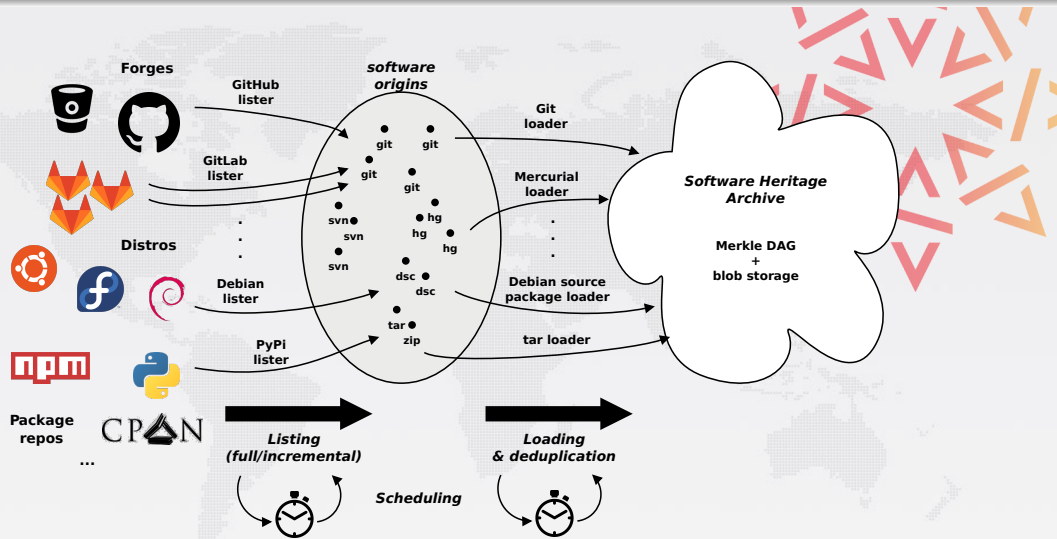
... in a VCS-/archive-agnostic **canonical data model**

We DON'T archive

- homepages, wikis
- BTS/issues/code reviews/etc.
- mailing lists

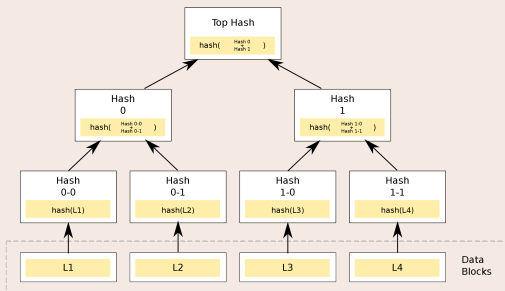
Long term vision: play our part in a *"semantic wikipedia of software"*

Data flow



Merkle trees

Merkle tree (R. C. Merkle, Crypto 1979)

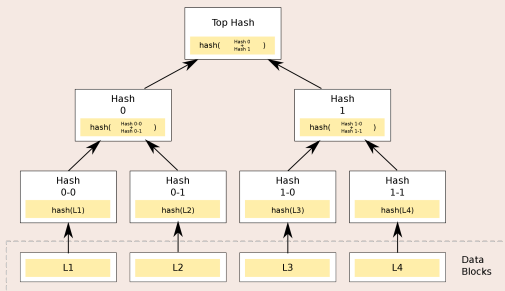


Combination of

- tree
- hash function

Merkle trees

Merkle tree (R. C. Merkle, Crypto 1979)



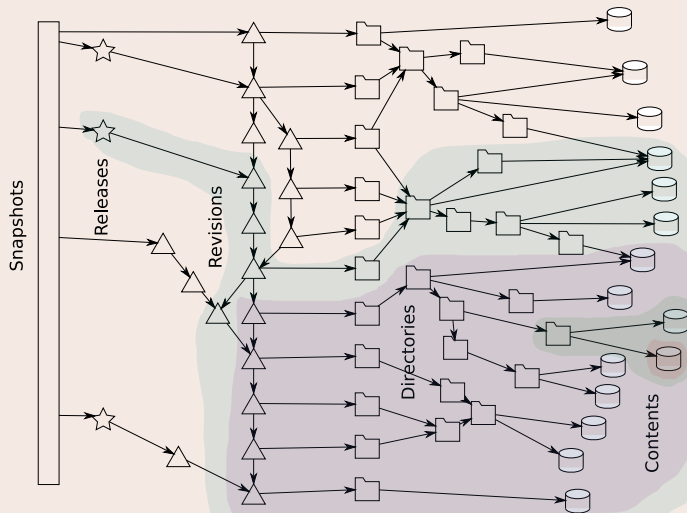
Combination of

- tree
- hash function

Classical cryptographic construction

- fast, parallel signature of large data structures
- widely used (e.g., Git, blockchains, IPFS, ...)
- built-in deduplication

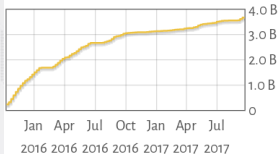
The archive: a (giant) Merkle DAG



Archive coverage

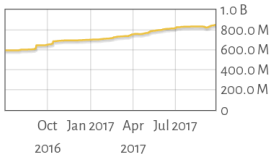
Source files

3,718,806,509



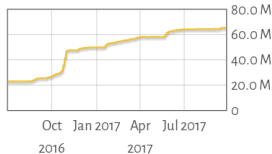
Commits

853,277,241



Projects

65,546,644



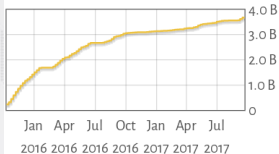
Our current sources

- GitHub
- Debian, GNU
- WIP: Gitorious, Google Code, Bitbucket

Archive coverage

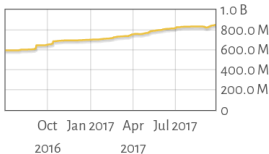
Source files

3,718,806,509



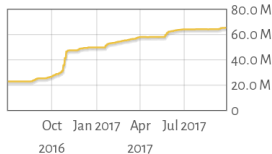
Commits

853,277,241



Projects

65,546,644



Our current sources

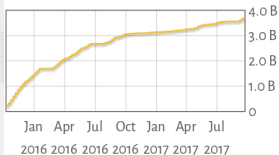
- GitHub
- Debian, GNU
- WIP: Gitorious, Google Code, Bitbucket

150 TB blobs, 5 TB database (as a graph: 7 B nodes + 60 B edges)

Archive coverage

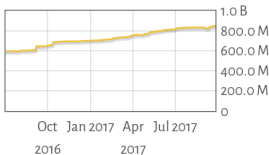
Source files

3,718,806,509



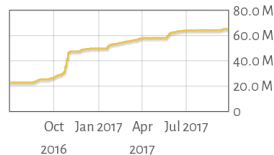
Commits

853,277,241



Projects

65,546,644



Our current sources

- GitHub
- Debian, GNU
- WIP: Gitorious, Google Code, Bitbucket

150 TB blobs, 5 TB database (as a graph: 7 B nodes + 60 B edges)

The *richest* source code archive already, ... and growing daily!



1 The Software Commons

2 Software Heritage

3 Architecture

4 Gory details

5 Community

6 Conclusion

Technology: how do you store the SWH DAG?

Problem statement

- How would you store and query a graph with 10 billion nodes and 60 billion edges?
- How would you store the contents of more than 3 billion files, 300TB of raw data?
- on a limited budget (100 000 € of hardware overall)

Technology: how do you store the SWH DAG?

Problem statement

- How would you store and query a graph with 10 billion nodes and 60 billion edges?
- How would you store the contents of more than 3 billion files, 300TB of raw data?
- on a limited budget (100 000 € of hardware overall)

Our hardware stack

- two hypervisors with 512GB RAM, 20TB SSD each, sharing access to a storage array (60 x 6TB spinning rust)
- one backup server with 48GB RAM and another storage array

Our software stack

- A RDBMS (PostgreSQL, what else?), for storage of the graph nodes and edges
- filesystems for storing the actual file contents

Metadata storage

- Python module `swh.storage`
- thin Python API over a pile of PostgreSQL functions
- motivation: keeping relational integrity for the DAG at the lowest layer

Content ("object") storage

- Python module `swh.objstorage`
- very thin object storage abstraction layer (PUT, APPEND and GET) over classic storage technologies
- separate layer for asynchronous replication and integrity management (`swh.archiver`)
- motivation: stay as technology neutral as possible for future mirrors

Object storage

Current primary deployment

- Storage on 16 sharded XFS filesystems; key = *sha1* (content), value = *gzip* (content)
- if sha1 = **abcdef01234...**, file path = / srv / storage / **a** / **ab** / **cd** / **ef** / **abcdef01234...**
- 3 directory levels deep, each level 256-wide = 16 777 216 directories (1 048 576 per partition)

It sucks.

Slow spinning storage + little RAM (48GB) + 16 million directory inodes = (very, very) bad performance

Outlook

- Moving towards a scale-out system: we need better performance and scalability.
- Can't really ask partial mirrors to commit to 100k EUR of hardware to spin up a copy of the archive.

Better object storage on plain Linux filesystems

fix our wrong assumptions about filesystems

Less directory depth, more breadth: make better use of "recent" filesystem features such as `dir_index`. Yes, directory lookups use hashtables on ext4 now!

optimize dentry storage / cache

shorter filenames (e.g. use base64/ascii85 instead of hex digits) = smaller directory entries, which means more entries in the `dir_index` hashtable.

does it work?

probably! not enough slack to test our assumptions :(



1 The Software Commons

2 Software Heritage

3 Architecture

4 Gory details

5 **Community**

6 Conclusion

You can help!

Coding

- www.softwareheritage.org/community/developers/
- forge.softwareheritage.org – **our own code**

Current development priorities

- ★★★ listers for unsupported forges, distros, pkg. managers
- ★★★ loaders for unsupported VCS, source package formats
- ★★ Web UI: eye candy wrapper around the Web API
- ★ content indexing and search

... *all* contributions equally welcome!

You can help!

Coding

- www.softwareheritage.org/community/developers/
- forge.softwareheritage.org – our own code

Current development priorities

- | | |
|-----|--|
| ★★★ | listers for unsupported forges, distros, pkg. managers |
| ★★★ | loaders for unsupported VCS, source package formats |
| ★★ | Web UI: eye candy wrapper around the Web API |
| ★ | content indexing and search |

... *all* contributions equally welcome!

Sharing the Software Heritage vision



See more

<http://www.softwareheritage.org/support/testimonials>

Sponsoring Software Heritage work



Microsoft



SOCIÉTÉ
GÉNÉRALE



Data Archiving and Networked Services
DANS

NOKIA Bell Labs



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
DIPARTIMENTO DI INFORMATICA - SCIENZA E INGEGNERIA

- 
- 1 The Software Commons
 - 2 Software Heritage
 - 3 Architecture
 - 4 Gory details
 - 5 Community
 - 6 Conclusion

Conclusion

Software Heritage is

- a *reference archive* of *all* FOSS ever written
- a unique *complement* for *development platforms*
- an international, open, nonprofit, *mutualized infrastructure*
- at the service of our community, at the service of society

References

Roberto Di Cosmo, Stefano Zacchiroli. *Software Heritage: Why and How to Preserve Software Source Code*. To appear, iPRES 2017, Kyoto, Sep 2017. Preprint: <http://deb.li/swhipres17>

Come in, we're open!

`www.softwareheritage.org` — *sponsoring, job openings*

`wiki.softwareheritage.org` — *internships, leads*

`forge.softwareheritage.org` — *our own code*

Q: how about SHA1 collisions?

```
create domain sha1 as bytea
  check (length(value) = 20);
create domain sha1_git as bytea
  check (length(value) = 20);
create domain sha256 as bytea
  check (length(value) = 32);

create table content (
  sha1      sha1 primary key,
  sha1_git  sha1_git not null,
  sha256    sha256 not null,
  length    bigint not null,
  ctime     timestampz not null default now(),
  status    content_status not null default 'visible',
  object_id bigserial
);

create unique index on content(sha1_git);
create unique index on content(sha256);
```

