

Software Heritage

Preserving the Free Software Commons

Nicolas Dandrimont

Software Engineer - Software Heritage
nicolas@dandrimont.eu

13 february 2018

Demandez le Programme! - Inria Saclay



Software Heritage

THE GREAT LIBRARY OF SOURCE CODE

- 1 The Software Commons
- 2 Software Heritage
- 3 Architecture
- 4 Gory details
- 5 Come in, we're open!
- 6 Conclusion
- 7 FAQ

B_APPENDIX

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[l][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 also 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.”

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

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?



A word cloud of terms related to software fragility, including: damage, disaster, malicious, obsolete, attack, dependencies, deletion, reference, storage, dangling, wear, corruption, encryption, format, aging, media, and tear.

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)



A word cloud of terms related to software fragility, including: damage, disaster, malicious, obsolete, attack, dependencies, deletion, reference, storage, dangling, wear, corruption, encryption, format, media, aging, and tear.

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?



A wealth of software research on crucial issues...

- safety, security, test, verification, proof
- software engineering, software evolution
- big data, machine learning, empirical studies



A wealth of software research on crucial issues...

- safety, security, test, verification, proof
- software engineering, software evolution
- big data, machine learning, empirical studies

If you study the stars, you go to Atacama...

... where is the *very large telescope* of source code?

- 1 The Software Commons
- 2 Software Heritage
- 3 Architecture
- 4 Gory details
- 5 Come in, we're open!
- 6 Conclusion
- 7 FAQ



B_APPENDIX



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

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 Come in, we're open!
- 6 Conclusion
- 7 FAQ



B_APPENDIX

Archiving goals

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

We DO archive

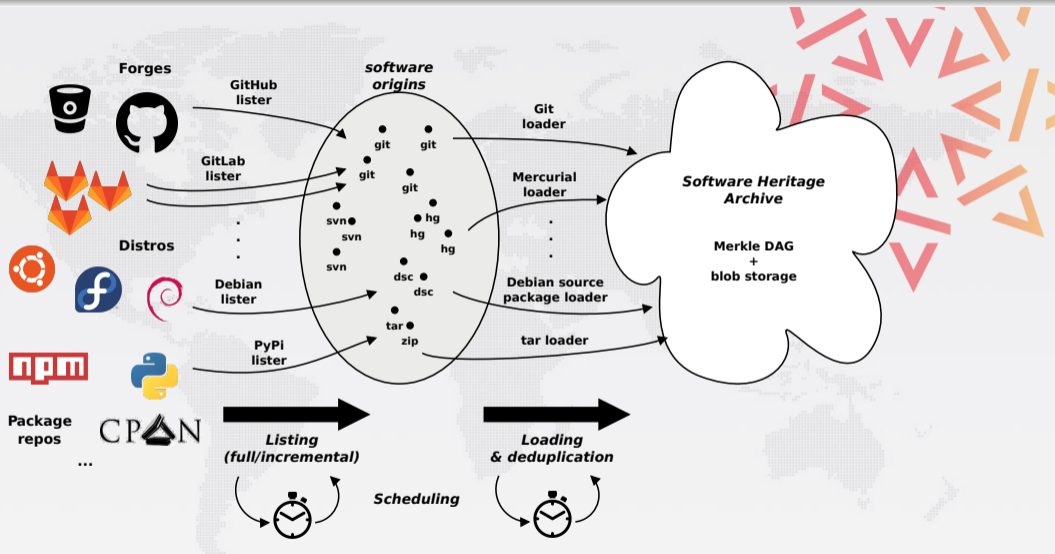
- file **content** (= blobs)
- **revisions** (= commits), with full metadata
- **releases** (= tags), ditto
- where (**origin**) & when (**visit**) we found any of the above

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

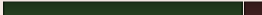
We DON'T archive

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

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



Revisions

Details	Changes	Files
SHA: 963634dca6ba5dc37e3ee426ba091092c267f9f6		
Author: Nicolas Dandrimont <nicolas@dandrimont.eu> (Thu Sep 1 14:26:13 2016)		
Committer: Nicolas Dandrimont <nicolas@dandrimont.eu> (Thu Sep 1 14:26:13 2016)		
Subject: <code>provenance.tasks: add the revision -> origin cache task</code>		
Parent: fc3a8b59ca1df424d860f2c29ab07fee4dc35d10 : <i>test...storage: property pipeline origin and cont...</i>		
<code>provenance.tasks: add the revision -> origin cache task</code>		
swh/storage/provenance/tasks.py  77		

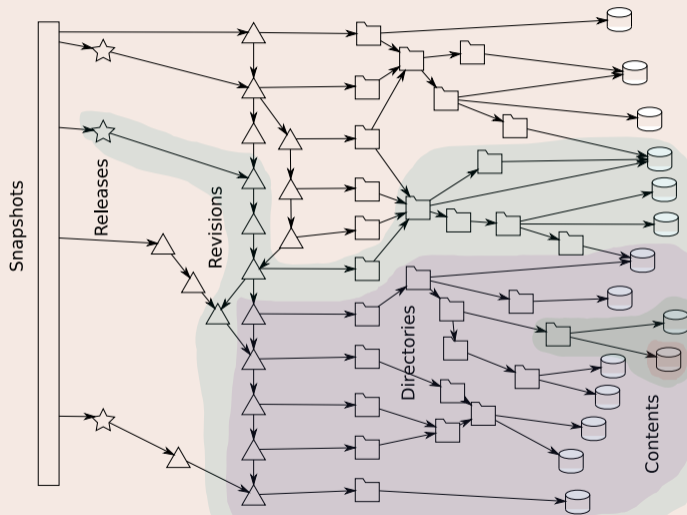


tree [515f00d44e92c65322aaa9bf3fa097c00ddb9c7d](#)
parent [fc3a8b59ca1df424d860f2c29ab07fee4dc35d10](#)
author Nicolas Dandrimont <nicolas@dandrimont.eu> 1472732773 +0200
committer Nicolas Dandrimont <nicolas@dandrimont.eu> 1472732773 +0200

`provenance.tasks: add the revision -> origin cache task`

id: [963634dca6ba5dc37e3ee426ba091092c267f9f6](#)

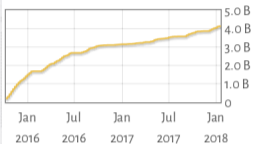
The archive: a (giant) Merkle DAG



Archive coverage

Source files

4,130,492,226



Commits

943,061,517



Projects

71,814,787



Current sources

- live: GitHub, Debian
- one-off: Gitorious, Google Code
- WIP: Bitbucket

Archive coverage



Current sources

- live: GitHub, Debian
- one-off: Gitorious, Google Code
- WIP: Bitbucket

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

Archive coverage



Current sources

- live: GitHub, Debian
- one-off: Gitorious, Google Code
- WIP: Bitbucket

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

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

3rd party

- Debian, Puppet
- PostgreSQL for metadata storage, with barman & pglogical
- Celery (RabbitMQ backend) for task scheduling
- Python3 and psycopg2 for the backend
- Flask and Bootstrap for Web stuff
- Phabricator

in house

- *ad hoc* object storage (to avoid imposing tech to mirrors)
- data model implementation, listers, loaders, scheduler
- ~60 Git repositories (~20 Python packages, ~30 Puppet modules)
- ~30 kSLOC Python / ~12 kSLOC SQL / ~4 kSLOC Puppet

in house

- 2x hypervisors with ~20 VMs
- 2x high density storage array (60 * 6TB => 300TB usable)

on Azure

- full object storage mirror
- full mirror of the database containing the graph
- workers for content indexing
- workers for download bundle preparation

at the University of Bologna

- backend storage (60TB) for the bundles available for download

classic FOSS development

- language: English
- development mailing list
<https://sympa.inria.fr/sympa/info/swh-devel>
- IRC
#swh-devel / FreeNode
- Forge
<https://forge.softwareheritage.org>
- Git, tasks, code review, etc.

for more information

<https://www.softwareheritage.org/community/developers/>

Features...

- (done) **lookup** by content hash
- **browsing**: "wayback machine" for archived code
 - (done) via Web API
 - (early access) via Web UI
- (early access) **deposit** of source code bundles directly to the archive
- (early access) **download**: `wget / git clone` from the archive
- (todo) **provenance** lookup for all archived content
- (todo) **full-text search** on all archived source code files

Features...

- (done) **lookup** by content hash
- **browsing**: "wayback machine" for archived code
 - (done) via Web API
 - (early access) via Web UI
- (early access) **deposit** of source code bundles directly to the archive
- (early access) **download**: `wget / git clone` from the archive
- (todo) **provenance** lookup for all archived content
- (todo) **full-text search** on all archived source code files

... and much more than one could possibly imagine

all the world's software development history in a single graph!

- 1 The Software Commons
- 2 Software Heritage
- 3 Architecture
- 4 Gory details**
- 5 Come in, we're open!
- 6 Conclusion
- 7 FAQ

B_APPENDIX

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 at the lowest layer

Content ("object") storage

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

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)

Secondary deployment

- Storage on Azure blob storage
- 16 storage containers, objects stored in a flat structure there

Generic model is fine

The abstraction layer is fairly simple and generic, and the implementation of the upper layers (replication, integrity checking) was a breeze.

Filesystem implementation is bad

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

Current deployment

- PostgreSQL deployed in primary/replica mode, using pg_logical for replication: different indexes on primary (tuned for writes) and replicas (tuned for reads).
- most logic done in SQL
- thin Pythonic API over the SQL functions

end goals

- proper handling of relations between objects at the lowest level
- doing fast recursive queries on the graph (e.g. find the provenance info for a content, walking up the whole graph, in one single query)

Limited resources

PostgreSQL works really well

Limited resources

PostgreSQL works really well ... until your indexes don't fit in RAM

Limited resources

PostgreSQL works really well ... until your indexes don't fit in RAM

Our recursive queries jump between different object types, and between evenly distributed hashes. Data locality doesn't exist. Caches break down.

Limited resources

PostgreSQL works really well ... until your indexes don't fit in RAM

Our recursive queries jump between different object types, and between evenly distributed hashes. Data locality doesn't exist. Caches break down.

Massive deduplication = efficient storage

Limited resources

PostgreSQL works really well ... until your indexes don't fit in RAM

Our recursive queries jump between different object types, and between evenly distributed hashes. Data locality doesn't exist. Caches break down.

Massive deduplication = efficient storage

but Massive deduplication = exponential width for recursive queries

Limited resources

PostgreSQL works really well ... until your indexes don't fit in RAM

Our recursive queries jump between different object types, and between evenly distributed hashes. Data locality doesn't exist. Caches break down.

Massive deduplication = efficient storage

but Massive deduplication = exponential width for recursive queries

Reality check

Referential integrity?

Limited resources

PostgreSQL works really well ... until your indexes don't fit in RAM

Our recursive queries jump between different object types, and between evenly distributed hashes. Data locality doesn't exist. Caches break down.

Massive deduplication = efficient storage

but Massive deduplication = exponential width for recursive queries

Reality check

Referential integrity? Real repositories downloaded from the internet are all kinds of broken.

Object storage

Our azure prototype shows that using a scale-out "cloudy" technology for our object storage works really well. Plain filesystems on spinning rust, not so much.

Object storage

Our azure prototype shows that using a scale-out "cloudy" technology for our object storage works really well. Plain filesystems on spinning rust, not so much. We have started working on a prototype Ceph infrastructure for our main copy of the archive, as our budget ramps up.

Object storage

Our azure prototype shows that using a scale-out "cloudy" technology for our object storage works really well. Plain filesystems on spinning rust, not so much. We have started working on a prototype Ceph infrastructure for our main copy of the archive, as our budget ramps up.

Metadata storage

Our initial assumption that we wanted referential integrity and built-in recursive queries was wrong.

Object storage

Our azure prototype shows that using a scale-out "cloudy" technology for our object storage works really well. Plain filesystems on spinning rust, not so much. We have started working on a prototype Ceph infrastructure for our main copy of the archive, as our budget ramps up.

Metadata storage

Our initial assumption that we wanted referential integrity and built-in recursive queries was wrong. We could probably migrate to "dumb" object storages for each type of object, with another layer to check metadata integrity regularly.

- 1 The Software Commons
- 2 Software Heritage
- 3 Architecture
- 4 Gory details
- 5 **Come in, we're open!**
- 6 Conclusion
- 7 FAQ



B_APPENDIX

Coding

- `forge.softwareheritage.org` – our own code



listers for unsupported forges, distros, pkg. managers



loaders for unsupported VCS, source package formats



Web UI: eye candy wrapper around the Web API

Coding

- `forge.softwareheritage.org` – our own code

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

Community

- ★★ spread the news, help us with long-term sustainability
- ★★★ document endangered source code

`wiki.softwareheritage.org/index.php?title=Suggestion_box`

Core team

10 people working on the project full-time, split across engineering, research, and fundraising/management topics.

Core team

10 people working on the project full-time, split across engineering, research, and fundraising/management topics.

Inria as initiator



- .fr national computer science research entity
- strong Free Software culture

Core team

10 people working on the project full-time, split across engineering, research, and fundraising/management topics.

Inria as initiator



- .fr national computer science research entity
- strong Free Software culture

Early Sponsors and Supporters

Société Générale, Microsoft, Huawei, Nokia, DANS, Univ. Bologna, ACM, Creative Commons, Eclipse, Engineering, FSF, Gandi, GitHub, IEEE, OIN, OSI, OW2, Software Freedom Conservancy, SFLC, The Document Foundation, ...

- 1 The Software Commons
- 2 Software Heritage
- 3 Architecture
- 4 Gory details
- 5 Come in, we're open!
- 6 Conclusion**
- 7 FAQ



B_APPENDIX

- It is urgent to preserve software source code; Software Heritage has took a systematic approach at it and has already assembled the largest archive to date.
- Software Heritage responds to cultural, research, and industry needs; it is a shared infrastructure that can benefit us all.
- We should collaborate and pool resources to make it so.

References

Roberto Di Cosmo, Stefano Zacchiroli. *Software Heritage: Why and How to Preserve Software Source Code*. iPRES 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*

- 1 The Software Commons
- 2 Software Heritage
- 3 Architecture
- 4 Gory details
- 5 Come in, we're open!
- 6 Conclusion
- 7 **FAQ**

B_APPENDIX

Q: do you archive *only* Free Software?

- We only crawl origins *meant* to host source code (e.g., forges)
- Most (~90%) of what we *actually* retrieve is textual content

Our goal

Archive **the entire Free Software Commons**

- Large parts of what we retrieve is *already* Free Software, today
- Most of the rest *will become* Free Software in the long term
 - e.g., at copyright expiration

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        timestamptz not null default now(),
  status        content_status not null default 'visible',
  object_id     bigserial
);

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