Software Heritage

The Global Graph of Public Code

Stefano Zacchiroli

Université de Paris & Software Heritage — zack@upsilon.cc, @zacchiro

17 June 2021
Meetup GraphRM, Roma
(online)

Software Heritage
THE GREAT LIBRARY OF SOURCE CODE

About me

- Associate Professor, Université de Paris, on leave at Inria
- Free/Open Source Software activist (20+ years)
- Debian Developer & Former 3x Debian Project Leader
- Former Open Source Initiative (OSI) director
- Software Heritage co-founder & CTO

Outline

Software is fragile

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, Bitbucket)
- for obsolete code: physical media decay (data rot)

Software is fragile

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, Bitbucket)
- for obsolete code: physical media decay (data rot)

Where is the archive…

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

Software Heritage
THE GREAT LIBRARY OF SOURCE CODE

Collect, preserve and share *all* software source code

Preserving our heritage, enabling better software and better science for all

Software Heritage

THE GREAT LIBRARY OF SOURCE CODE

Collect, preserve and share *all* software source code

Preserving our heritage, enabling better software and better science for all

Reference catalog

find and reference all software source code

Software Heritage in a nutshell

www.softwareheritage.org

Software Heritage

THE GREAT LIBRARY OF SOURCE CODE

Collect, preserve and share *all* software source code

Preserving our heritage, enabling better software and better science for all

Reference catalog

find and reference all software source code

Universal archive

preserve all software source code

Software Heritage

THE GREAT LIBRARY OF SOURCE CODE

Collect, preserve and share *all* software source code

Preserving our heritage, enabling better software and better science for all

Reference catalog

find and reference all software source code

Universal archive

preserve all software source code

Research infrastructure

enable analysis of all software source code

Core principles

Technology

- FOSS and transparency
- replicas all the way down

Content

- intrinsic identifiers
- facts and provenance

Organization

- non-profit
- multi-stakeholder

Archiving goals

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

We DO archive

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

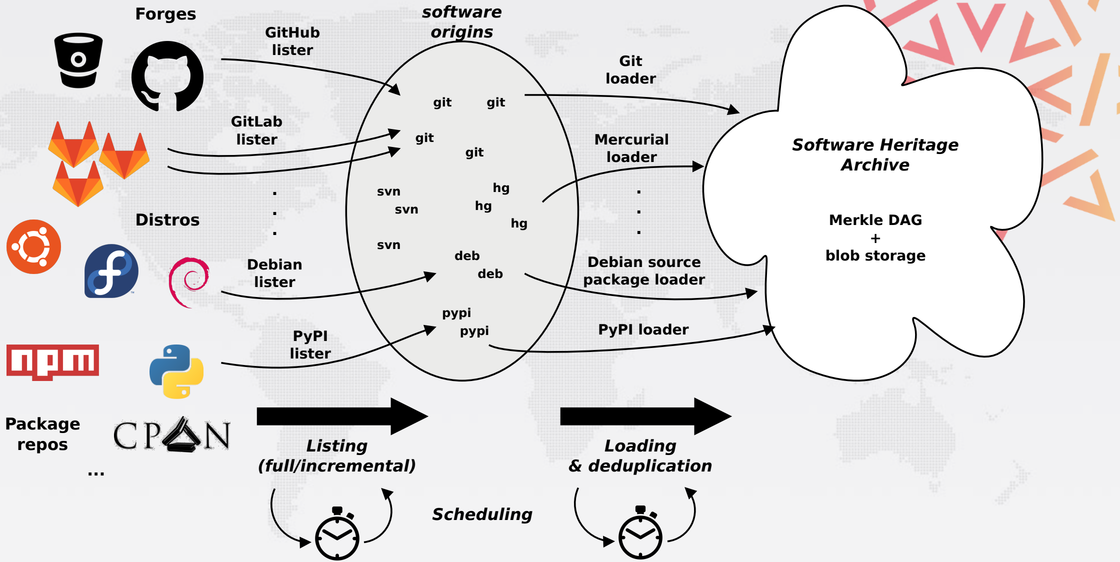
... in a VCS-/archive-agnostic canonical data model

We DON'T archive (yet)

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

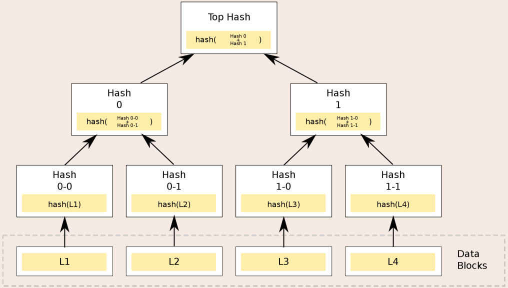
Long term vision: play our part in a *"semantic wikipedia of software"*

Data flow



Forges
GitHub lister
software origins
GitLab lister
Distros
Debian lister
PyPI lister
npm
CPAN
Package repos
...
git
git
git
git
svn
svn
svn
hg
hg
hg
deb
deb
pypi
pypi
Git loader
Mercurial loader
Debian source package loader
PyPI loader
Software Heritage Archive
Merkle DAG
+
blob storage
Listing (full/incremental)
Scheduling
Loading & deduplication

Merkle tree (R. C. Merkle, CRYPTO 1987)

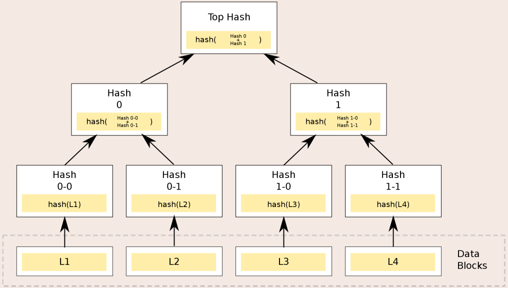


Combination of

- tree
- hash function

Merkle trees

Merkle tree (R. C. Merkle, CRYPTO 1987)



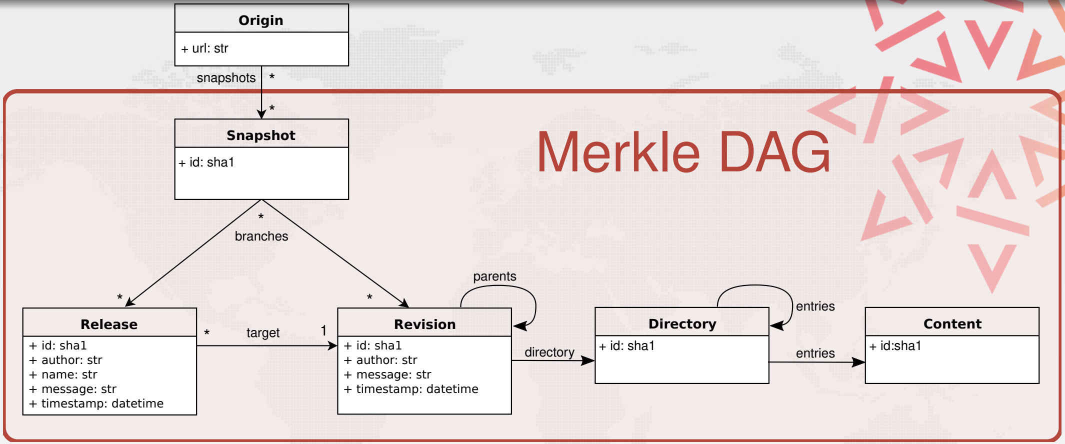
Combination of

- tree
- hash function

Classical cryptographic construction

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

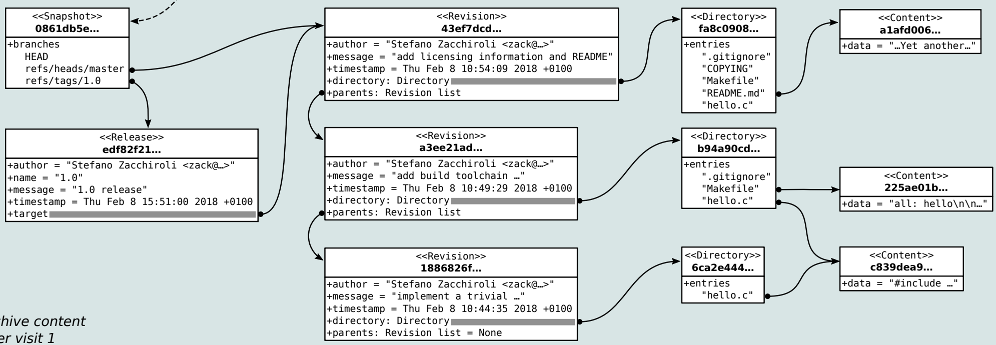
Data model



A global graph linking together fully deduplicated source code artifact (files, commits, directories, releases, etc.) to the places that distribute them (e.g., Git repositories), providing a unified view on the entire *Software Commons*.

The archive: a (giant) Merkle DAG

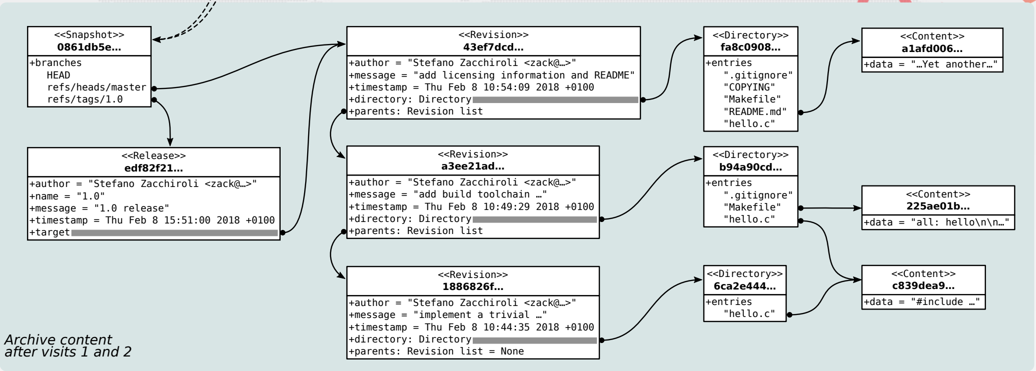
origin	visit	snapshot	timestamp
https://forge.softwareheritage.org/source/helloworld.git	1	0861db5e...	Fri Feb 9 12:38:45 2018 +0100



Archive content after visit 1

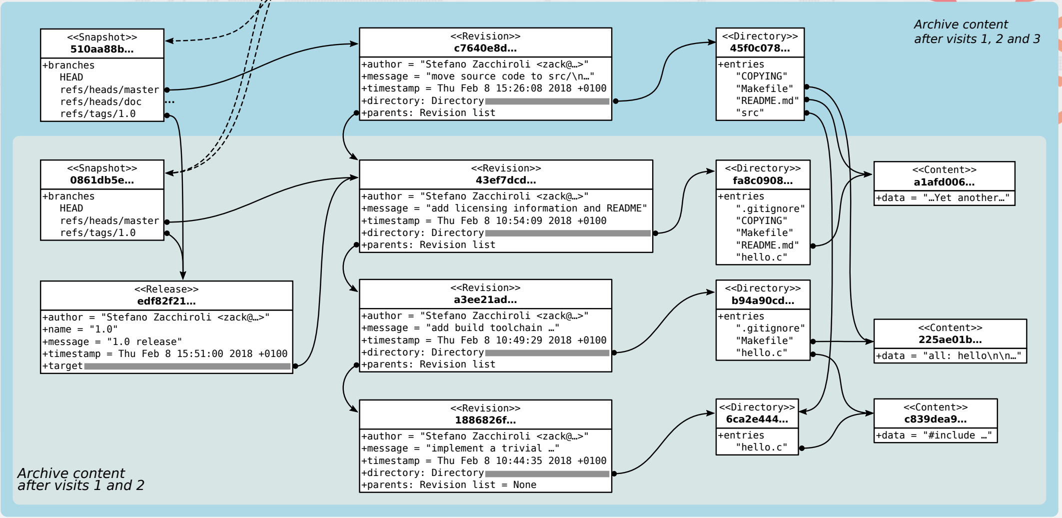
The archive: a (giant) Merkle DAG

origin	*visit*	*snapshot*	*timestamp*
https://forge.softwareheritage.org/source/helloworld.git	1	0861db5e...	Fri Feb 9 12:38:45 2018 +0100
https://forge.softwareheritage.org/source/helloworld.git	2	0861db5e...	Fri Feb 9 13:29:00 2018 +0100



The archive: a (giant) Merkle DAG

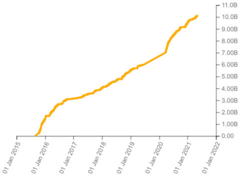
origin	visit	snapshot	timestamp
https://forge.softwareheritage.org/source/helloworld.git	1	0861db5e...	Fri Feb 9 12:38:45 2018 +0100
https://forge.softwareheritage.org/source/helloworld.git	2	0861db5e...	Fri Feb 9 13:29:00 2018 +0100
https://forge.softwareheritage.org/source/helloworld.git	3	510aa88b...	Fri Feb 9 15:52:50 2018 +0100



Archive coverage — archive.softwareheritage.org

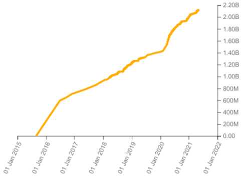
Source files

10,162,254,604



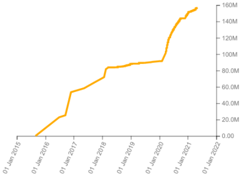
Commits

2,128,899,595



Projects

156,417,833



Directories

8,477,359,190

Authors

42,598,760

Releases

17,293,952

Bitbucket · R · debian · Framagit · GitHub · GitLab

GITORIOUS · Google code · GNU · Guix · HAL archives-ouvertes.fr · Inria

IPOL Journal · npm · NixOS · python Package Index

Archive coverage — archive.softwareheritage.org

- on disk: ~700 TB (uncompressed); as a graph ~20 B nodes, ~200 B edges
- the largest public source code archive in the world (and growing!)

Web interface

archive.softwareheritage.org

DEMO TIME !

Outline

Use cases — product needs

e.g., for `https://archive.softwareheritage.org`

Browsing

- `ls`
- `git log` (Linux kernel: 800K+ commits)

Wayback machine

- tarball
- `git bundle` (Linux kernel: 7M+ nodes)

Provenance tracking

- commit provenance (one/all contexts) note: requires backtracking
- origin provenance (one/all contexts)

Note: we therefore need both the direct Merkle DAG graph and its transposed

Use cases — research questions

For the sake of it

- local graph topology
- connected component size
  - enabling question to identify the best approach (e.g., scale-up v. scale-out) to conduct large-scale analyses
- any other emerging property

Software Engineering topics

- software provenance analysis at this scale is pretty much unexplored yet
- industry frontier: increase granularity down to the individual line of code
- replicate at this scale (famous) studies that have generally been conducted on (much) smaller version control system samples to confirm/refute their findings
- ...

How do you query the Software Heritage archive?
(on a budget)

Software Heritage Graph dataset

Use case: large scale analyses of the most comprehensive corpus on the development history of free/open source software.

Antoine Pietri, Diomidis Spinellis, Stefano Zacchiroli
The Software Heritage Graph Dataset: Public software development under one roof
MSR 2019: 16th Intl. Conf. on Mining Software Repositories. IEEE
preprint: http://deb.li/swhmsr19

Dataset

- Relational representation of the full graph as a set of tables
- Available as open data: https://doi.org/10.5281/zenodo.2583978
- Chosen as subject for the MSR 2020 Mining Challenge

Formats

- Local use: PostgreSQL dumps, or Apache Parquet files (~1 TiB each)
- Live usage: Amazon Athena (SQL-queriable), Azure Data Lake

Sample query — most frequent first commit words

```sql
SELECT COUNT(*) AS c, word FROM (
  SELECT LOWER(REGEXP_EXTRACT(FROM_UTF8(
  message), '^\w+')) AS word FROM revision)
WHERE word != ''
GROUP BY word ORDER BY COUNT(*) DESC LIMIT 5;
```

Sample query — most frequent first commit words

```sql
SELECT COUNT(*) AS c, word FROM (
  SELECT LOWER(REGEXP_EXTRACT(FROM_UTF8(
  message), '^\w+')) AS word FROM revision)
WHERE word != ''
GROUP BY word ORDER BY COUNT(*) DESC LIMIT 5;
```

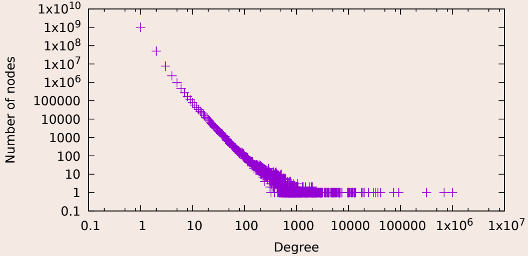
Count	Word
71 338 310	update
64 980 346	merge
56 854 372	add
44 971 954	added
33 222 056	fix

Sample query — fork and merge arities

Fork arity

i.e., how often is a commit based upon?

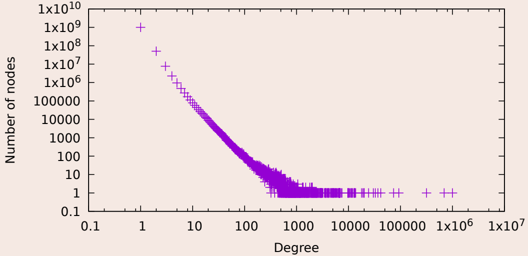
```sql
SELECT fork_deg, count(*) FROM (
  SELECT id, count(*) AS fork_deg
  FROM revision_history GROUP BY id) t
GROUP BY fork_deg ORDER BY fork_deg;
```



Fork arity

i.e., how often is a commit based upon?

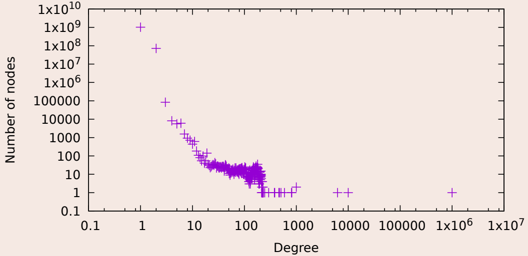
```
SELECT fork_deg, count(*) FROM (
  SELECT id, count(*) AS fork_deg
  FROM revision_history GROUP BY id) t
GROUP BY fork_deg ORDER BY fork_deg;
```



Merge arity

i.e., how large are merges?

```
SELECT merge_deg, COUNT(*) FROM (
  SELECT parent_id, COUNT(*) AS merge_deg
  FROM revision_history GROUP BY parent_id
GROUP BY deg ORDER BY deg;
```



Sample query — ratio of commits performed during weekends

```sql
WITH revision_date AS
  (SELECT FROM_UNIXTIME(date / 1000000) AS date
  FROM revision)
SELECT yearly_rev.year AS year,
  CAST(yearly_weekend_rev.number AS DOUBLE)
  / yearly_rev.number * 100.0 AS weekend_pc
FROM
  (SELECT YEAR(date) AS year, COUNT(*) AS number
  FROM revision_date
  WHERE YEAR(date) BETWEEN 1971 AND 2018
  GROUP BY  YEAR(date) ) AS yearly_rev
JOIN
  (SELECT YEAR(date) AS year, COUNT(*) AS number
  FROM revision_date
  WHERE DAY_OF_WEEK(date) >= 6
      AND YEAR(date) BETWEEN 1971 AND 2018
  GROUP BY  YEAR(date) ) AS yearly_weekend_rev
  ON yearly_rev.year = yearly_weekend_rev.year
ORDER BY  year DESC;
```

Sample query — ratio of commits performed during weekends (cont.)

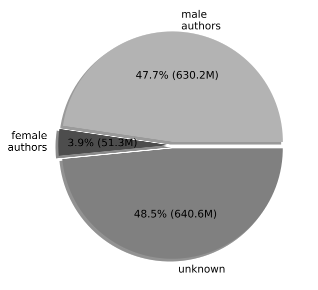
Year	Weekend	Total	Weekend percentage
2018	15130065	78539158	19.26
2017	33776451	168074276	20.09
2016	43890325	209442130	20.95
2015	35781159	166884920	21.44
2014	24591048	122341275	20.10
2013	17792778	88524430	20.09
2012	12794430	64516008	19.83
2011	9765190	48479321	20.14
2010	7766348	38561515	20.14
2009	6352253	31053219	20.45
2008	4568373	22474882	20.32
2007	3318881	16289632	20.37
2006	2597142	12224905	21.24
2005	2086697	9603804	21.72
2004	1752400	7948104	22.04
2003	1426033	6941593	20.54

Sample query — average size of the most popular file types

```sql
SELECT suffix,
  ROUND(COUNT(*) * 100 / 1e6) AS Million_files,
  ROUND(AVG(length) / 1024) AS Average_k_length
FROM
  (SELECT length, suffix
  FROM  -- File length in joinable form
    (SELECT TO_BASE64(sha1_git) AS sha1_git64, length
    FROM content ) AS content_length
  JOIN  -- Sample of files with popular suffixes
  (SELECT target64, file_suffix_sample.suffix AS suffix
  FROM  -- Popular suffixes
    (SELECT suffix FROM (
      SELECT REGEXP_EXTRACT(FROM_UTF8(name),
       '\.[^.]+$') AS suffix
    FROM directory_entry_file) AS file_suffix
    GROUP BY  suffix
    ORDER BY  COUNT(*) DESC LIMIT 20 ) AS pop_suffix
  JOIN  -- Sample of files and suffixes
    (SELECT TO_BASE64(target) AS target64,
      REGEXP_EXTRACT(FROM_UTF8(name),
        '\.[^.]+$') AS suffix
    FROM directory_entry_file TABLESAMPLE BERNOULLI(1))
    AS file_suffix_sample
  ON file_suffix_sample.suffix = pop_suffix.suffix)
```

Sample study — 50 years of gender differences in code contributions

- start from the Software Heritage graph dataset
- detect gender of author names using standard tooling (`gender-guesser`)
- analyze both authors and commits over time, bucketing by commit timestamp



total commits by author gender (left), ratio of active female commiters over time (right)

Stefano Zacchiroli
Gender Differences in Public Code Contributions: a 50-year Perspective
IEEE Softw. 38(2): 45-50 (2021)

Discussion

- one *can* query such a corpus SQL-style
- but relational representation shows its limits at this scale
  - …at least as deployed on commercial SQL offerings such as Athena
- note: (naive) sharding is ineffective, due to the pseudo-random distribution of node identifiers
- experiments with Google BigQuery are ongoing
  - (we broke it at the first import attempt…, due to very large arrays in directory entry tables)

Outline

Graph compression on the Software Heritage archive

Paolo Boldi, Antoine Pietri, Sebastiano Vigna, Stefano Zacchiroli
Ultra-Large-Scale Repository Analysis via Graph Compression
SANER 2020, 27th Intl. Conf. on Software Analysis, Evolution and Reengineering. IEEE

Research question

Is it possible to efficiently perform software development history analyses at ultra large scale (= the scale of Software Heritage archive or more), on a single, relatively cheap machine?

Idea

Apply state-of-the-art graph compression techniques from the field of Web graph / social network analysis.

Background — (Web) graph compression

Definition (The graph of the Web)

Directed graph that has Web pages as nodes and hyperlinks between them as edges.

Properties (1)

- Locality: pages link to pages whose URLs are lexicographically similar. URLs share long common prefixes.

→ use D-gap compression

Adjacency lists

Node	Outdegree	Successors
…	…	…
15	11	13,15,16,17,18,19,23,24,203,315,1034
16	10	15,16,17,22,23,24,315,316,317,3041
17	0	
18	5	13,15,16,17,50
…	…	…

D-gapped adjacency lists

Node	Outdegree	Successors
…	…	…
15	11	3,1,0,0,0,0,3,0,178,111,718
16	10	1,0,0,4,0,0,290,0,0,2723
17	0	
18	5	9,1,0,0,32
…	…	…

Background — (Web) graph compression (cont.)

Definition (The graph of the Web)

Directed graph that has Web pages as nodes and hyperlinks between them as edges.

Properties (2)

- **Similarity:** pages that are close together in lexicographic order tend to have many common successors.

$\rightarrow$ use reference compression

Adjacency lists

Node	Outd.	Successors
...	...	...
15	11	13,15,16,17,18,19,23,24,203,315,1034
16	10	15,16,17,22,23,24,315,316,317,3041
17	0	
18	5	13,15,16,17,50
...	...	...

Copy lists

Node	Ref.	Copy list	Extra nodes
...	...	...	...
15	0		13,15,16,17,18,19,23,24,203,315,1034
16	1	01110011010	22,316,317,3041
17			
18	3	11110000000	50
...	...	...	

Corpus

Nodes

Node type	N. of nodes
origins	88 M
snapshots	57 M
releases	9.9 M
revisions	1.1 B
directories	4.9 B
contents	5.5 B
Total nodes	12 B

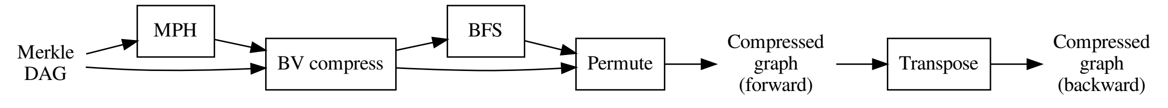
Edges

Edge type	N. of edges
origin → snapshot	195 M
snapshot → revision	616 M
snapshot → release	215 M
release → revision	9.9 M
revision → revision	1.2 B
revision → directory	1.1 B
directory → directory	48 B
directory → revisiony	482 M
directory → content	112 B
Total edges	165 B

Stats for archive snapshot 2018-09-25, from the Software Heritage graph dataset.
Growth rate: exponential, doubling every 22-30 months, cf.:

Roberto Di Cosmo, Guillaume Rousseau, Stefano Zacchiroli
Software Provenance Tracking at the Scale of Public Source Code
Empirical Software Engineering 25(4): 2930-2959 (2020)

Graph compression pipeline



- MPH: minimal perfect hash, mapping Merkle IDs to 0..N-1 integers
- BV compress: Boldi-Vigna compression (based on MPH order)
- BFS: breadth-first visit to renumber
- Permute: update BV compression according to BFS order

(Re)establishing locality

- key for good compression is a node ordering that ensures locality and similarity
- which is very much *not* the case with Merkle IDs, …but is the case *again* after BFS reordering

Compression experiment

Step	Wall time (hours)
MPH	2
BV Compress	84
BFS	19
Permute	18
Transpose	15
Total	138 (6 days)

- server equipped with 24 CPUs and 750 GB of RAM
- RAM mostly used as I/O cache for the BFS step
- *minimum* memory requirements are close to the RAM needed to load the final compressed graph in memory

Compression efficiency (space)

Forward graph	
total size	91 GiB
bits per edge	4.91
compression ratio	15.8%

Backward graph	
total size	83 GiB
bits per edge	4.49
compression ratio	14.4%

Operating cost

The structure of a full bidirectional archive graph fits in less than 200 GiB of RAM, for a hardware cost of ~300 USD.

Compression efficiency (time)

Benchmark — Full BFS visit (single thread)

Forward graph	
wall time	1h48m
throughput	1.81 M nodes/s
	(553 ns/node)

Backward graph	
wall time	3h17m
throughput	988 M nodes/s
	(1.01 µs/node)

Benchmark — Edge lookup

random sample: 1 B nodes (8.3% of entire graph); then enumeration of all successors

Forward graph	
visited edges	13.6 B
throughput	12.0 M edges/s
	(83 ns/edge)

Backward graph	
visited edges	13.6 B
throughput	9.45 M edges/s
	(106 ns/edge)

Note how edge lookup time is close to DRAM random access time (50-60 ns).

Discussion

Incrementality

compression is not incremental, due to the use of contiguous integer ranges

- but the graph is append-only, so…
- …based on expected graph growth rate it should be possible to pre-allocate enough free space in the integer ranges to support amortized incrementality (future work)

Discussion

Incrementality

compression is not incremental, due to the use of contiguous integer ranges

- but the graph is append-only, so…
- …based on expected graph growth rate it should be possible to pre-allocate enough free space in the integer ranges to support amortized incrementality (future work)

In-memory v. on-disk

the compressed in-memory graph structure has no attributes

- usual design is to exploit the 0..N-1 integer ranges to memory map node attributes to disk for efficient access
- works well for queries that does graph traversal first and "join" node attributes last; ping-pong between the two is expensive
- edge attributes are more problematic (work in progress)

Outline

Technical leads for graph geeks (selected)

A graph query language for Software Heritage

- Design and implement a graph query language suitable for Software Heritage use cases and specifities (e.g., read-only, compressed representation, split in-memory/on-disk representation, etc).
- Possibly building on-top existing graph technologies: e.g., an Apache TinkerPop backend for WebGraph
- Learn more: `https://wiki.softwareheritage.org/wiki/Graph_query_language_for_the_archive_(internship)`

Technical leads for graph geeks (selected) (cont.)

Python bindings for WebGraph (Java)

- Enable researchers and data scientists to exploit the graph in a familiar ecosystem (NumPy, Pandas, iGraph, etc.)
- Requires careful performance trade-offs to minimize jumping between the two runtimes
- Learn more: `https://wiki.softwareheritage.org/wiki/Python_bindings_for_WebGraph_(internship)`

Graph storage challenge

Can your graph database do better?

- A recent dump of the Software Heritage graph is available in Apache ORC data format from AWS S3 as an open dataset (8.4 TiB)
- Schema documentation: `https://docs.softwareheritage.org/devel/swh-dataset/graph/schema.html`
- Try it out: load it into your favorite graph database and see if it is up to the challenge. We'd love to hear about it!

```
$ aws s3 ls s3://softwareheritage/graph/2021-03-23/orc/
    PRE content/
    PRE directory/
    PRE origin/
    PRE release/
    PRE revision/
    [...]
$ aws s3 ls s3://softwareheritage/graph/2021-03-23/orc/directory/ | head -n 1
2021-04-13 06:59:21 3004055177 graph-03891ef6-18c0-46e7-88db-42056fa33282.orc
```

Outline

Wrapping up

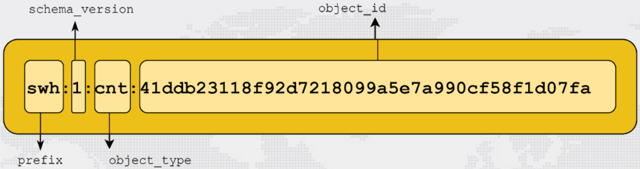
- Software Heritage archives all public source code as a huge Merkle DAG
- The Software Heritage graph offers an unified view on the software commons
- Querying and analyzing it at scale (20/200 B nodes/edges) is an open problem
- Gold mine of R&D challenges and leads for graph geeks

References

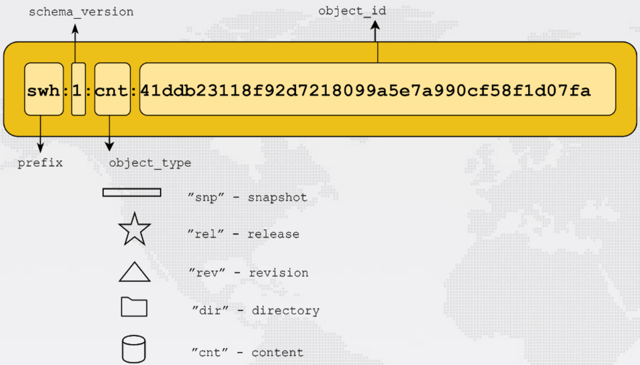
- homepage: www.softwareheritage.org
- development info: www.softwareheritage.org/community/developers

Contacts

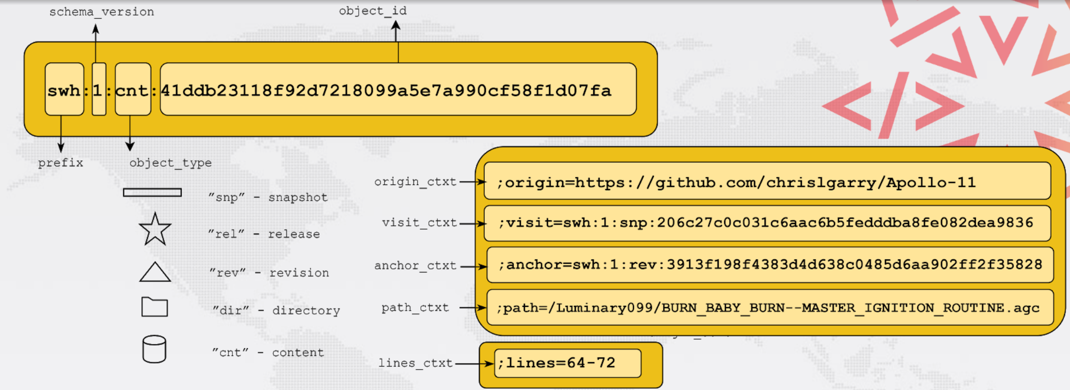
Stefano Zacchiroli / upsilon.cc / zack@upsilon.cc / @zacchiro


schema_version
object_id
swh:1:cnt:41ddb23118f92d7218099a5e7a990cf58f1d07fa
prefix
object_type

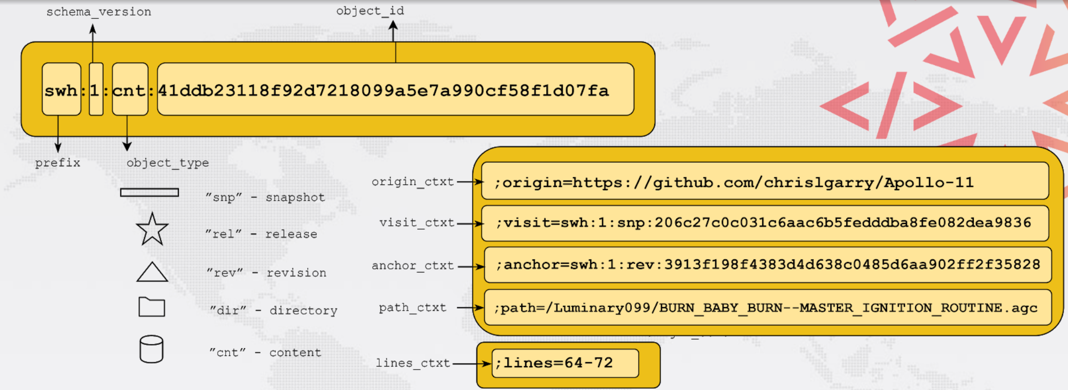
Software Heritage Identifiers (SWHIDs) (full spec)



Software Heritage Identifiers (SWHIDs) (full spec)

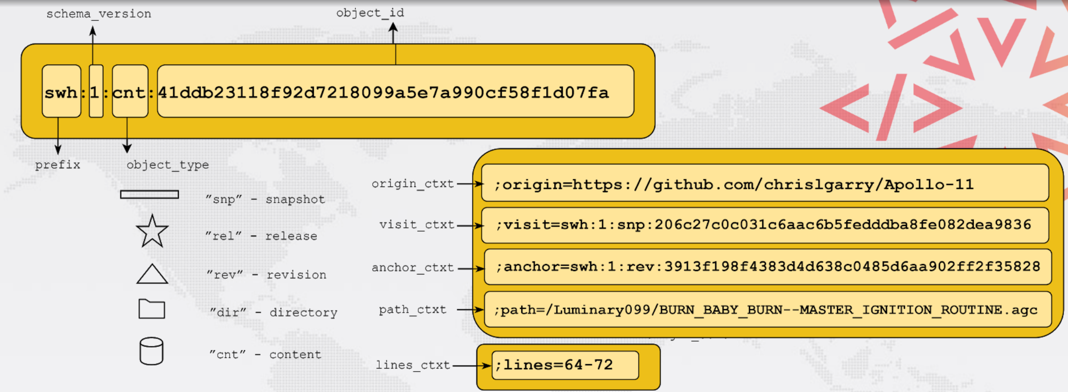


Software Heritage Identifiers (SWHIDs) (full spec)



An emerging standard

- in Linux Foundation's SPDX 2.2
- IANA-registered "`swh:`" URI prefix
- WikiData property P6138



An emerging standard

- in Linux Foundation's SPDX 2.2
- IANA-registered "swh:" URI prefix
- WikiData property P6138

Examples

- Apollo 11 AGC excerpt
- Quake III rsqrt

Web API

RESTful API to programmatically access the Software Heritage archive
`https://archive.softwareheritage.org/api/`

Features

- pointwise browsing of the archive
  - ... snapshots → revisions → directories → contents ...
- full access to the metadata of archived objects
- crawling information
  - *when have you last visited this Git repository I care about?*
  - *where were its branches/tags pointing to at the time?*

Endpoint index

`https://archive.softwareheritage.org/api/1/`

Software Heritage Filesystem (SwhFS)

The Software Heritage Filesystem (SwhFS) is a user-space POSIX filesystem that enables browsing parts of the Software Heritage archive as if it were locally available.

- code: `https://forge.softwareheritage.org/source/swh-fuse/`
- documentation: `https://docs.softwareheritage.org/devel/swh-fuse/`

Thibault Allançon, Antoine Pietri, Stefano Zacchiroli
The Software Heritage Filesystem (SwhFS): Integrating Source Code Archival with Development
ICSE 2021: The 43rd International Conference on Software Engineering
`https://arxiv.org/abs/2102.06390`