Global Software Health

an Unified View of how our Software Commons is Doing

Stefano Zacchiroli

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

3 July 2020
SoHeal 2020
(via conf call)

Software Heritage
THE GREAT LIBRARY OF SOURCE CODE

Outline

Software Health

Software Health

Definition (Software Health)

One of the hardest research fields to search the Web for.

Proof (empirical, trivial).

Exhibit: `https://www.google.com/search?q=software+health` □

Software Health

Definition (Software Health)

One of the hardest research fields to search the Web for.

Proof (empirical, trivial).

Exhibit: `https://www.google.com/search?q=software+health` □

More seriously…

The SoHeal community has pioneered the exploration of the notion of Software Health. By now we have evidence of interest in several *dimensions* of the notion, we have *tools & techniques* that are routinely used to explore them, and we have been doing that at various *scopes*.

What are we looking at

Several dimensions have been explored thus far, e.g.:

- software evolution and "liveliness"
- quality (cf. SoHeal 2019 keynote by Jesus M. Gonzalez-Barahona)
- community
  - both static structure
  - and dynamics over time

(non-exhaustive list)

Software Health — tools & techniques

How we are exploring the topic

- classic software evolution & MSR techniques
- quantitative analysis (stats !)
- qualitative analysis
  - e.g., interviews, ethnography, Delphi method
- community metrics & their standardization (cf. CHAOSS)
- raising awareness in relevant communities: FOSS + scholars

the SoHeal workshop series!

Software Health — scope

How *far* are we looking

1. a single project
2. a set of inter-dependent projects
   - e.g., a specific framework with plugins, a software stack, etc.
   - also a community of contributors working on said projects
3. an ecosystem
   - e.g., Debian, PyPI, NPM, etc.

Software Health — scope

How *far* are we looking

1. a single project
2. a set of inter-dependent projects
   - e.g., a specific framework with plugins, a software stack, etc.
   - also a community of contributors working on said projects
3. an ecosystem
   - e.g., Debian, PyPI, NPM, etc.

Going further

- can we go further in terms of software health scope? how far?
- is there a meaningful notion of "global software health"?
- if there is, which the tools can we use to explore global software health?
- if they exist and are practical, what is the current status of global software health?

Outline

Free Software

(I know you all know this, but bear with me. I pinky promise it's gonna be useful!)

Definition (Free Software)

A program is free software if the program's users have the four *essential freedoms*:

- Freedom #0, to run the program, for any purpose
- Freedom #1, to study how the program works, and change it
- Freedom #2, to redistribute copies
- Freedom #3, to improve the program, and release improvements

by the Free Software Foundation

ChangeLog: 2-freedom version: 1986, 3-freedom: 1990; 4-freedom: early 90s

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

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

Global Software Health

Proposition #1

The full extent of our shared software commons is the ultimate scope for software health.

$$\text{global software health} = \text{software health} + \text{software commons}$$

Definition (Global Software Health (tentative))

The investigation of software health at the scale of the entire software commons.

Global Software Health

Proposition #1

The full extent of our shared software commons is the ultimate scope for software health.

global software health = software health + software commons

Definition (Global Software Health (tentative))

The investigation of software health at the scale of the entire software commons.

Proposition #2

As a starting point for global software health analysis, we need the equivalent of ancient world libraries, i.e., great libraries of software artifacts, that encompass the software commons as much as possible.

- GHTorrent
- World of Code
- Software Heritage ($\leftarrow$ my focus for the rest of this talk)

Outline

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 in a nutshell

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

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

An international, non profit initiative — built for the long term

Sharing the vision

UNESCO
United Nations Educational, Scientific and Cultural Organization

Software, GitLab, Engineering, eclipse, OW2, fsfe, nlnet foundation, SIGSOFT, République Française, INAPI, The Linux Foundation, Informatics Europe, Computer History Museum, Adullact, software freedom conservancy, Software Freedom Law Center, AdaCore, gandi.net, TDF, Free Software Foundation, creative commons, SIF, open source initiative, openinventionnetwork

www.softwareheritage.org/support/testimonials

Donors, members, sponsors

Inria — Inventeurs du monde numérique

Platinum sponsors

Microsoft, intel, Société Générale, Huawei

Gold sponsor

Ministère de l'Enseignement Supérieur, de la Recherche et de l'Innovation; openinventionnetwork; Université de Paris

Silver sponsors

CAST, République Française, GitHub, Google, Università di Pisa, vmware

Bronze sponsors

DANS, FOSSID, Nokia Bell Labs, Alma Mater Studiorum, UQAM

www.softwareheritage.org/support/sponsors

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

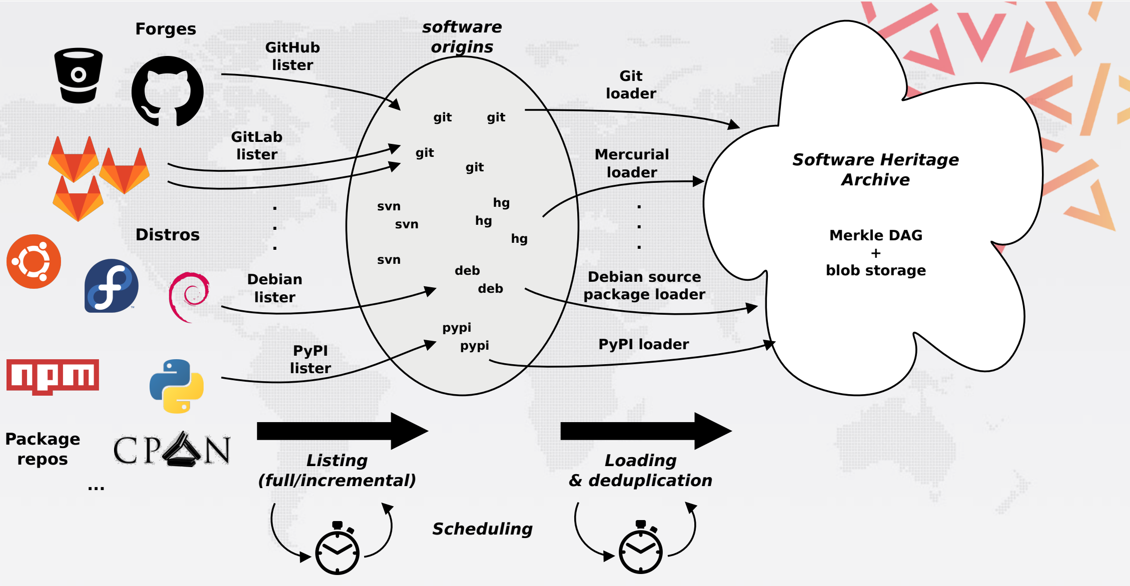
… in a VCS-/archive-agnostic canonical data model

We DON'T archive

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

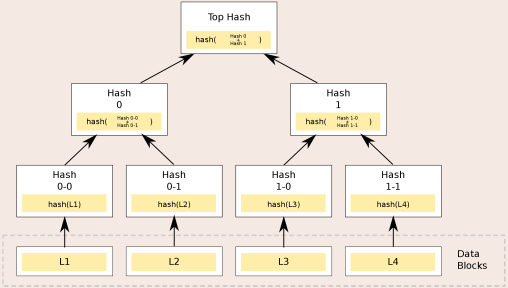
Long term vision: play our part in a *"semantic wikipedia of software"*

Data flow



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

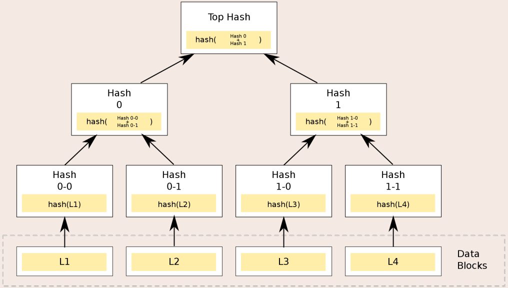
Merkle tree (R. C. Merkle, CRYPTO 1987)



Combination of

- tree
- hash function

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



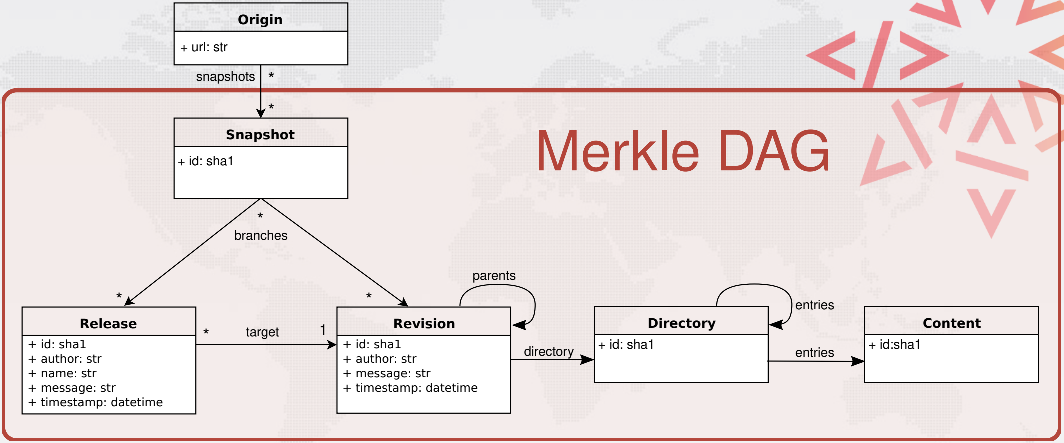
Combination of

- tree
- hash function

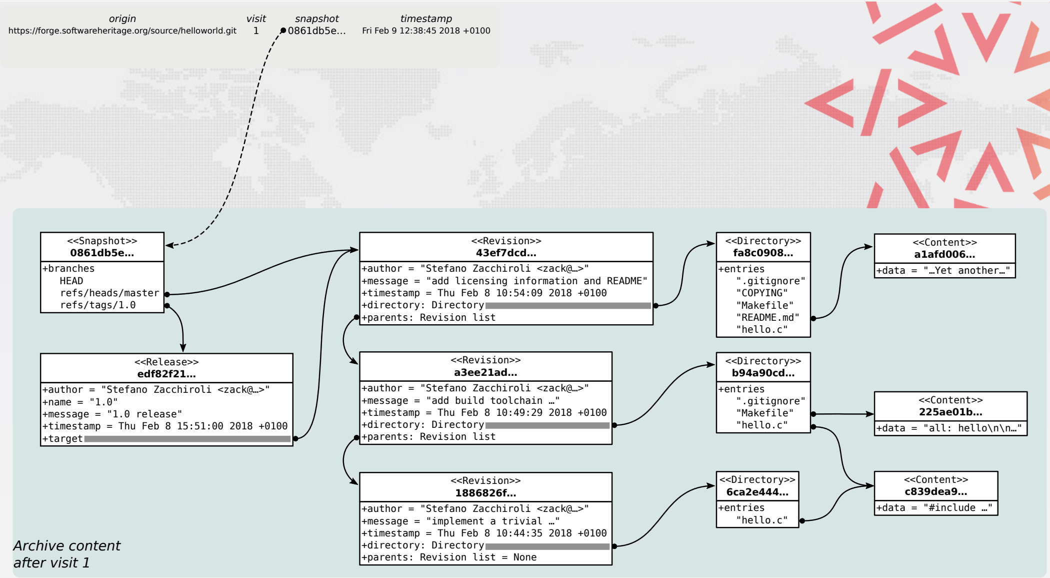
Classical cryptographic construction

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

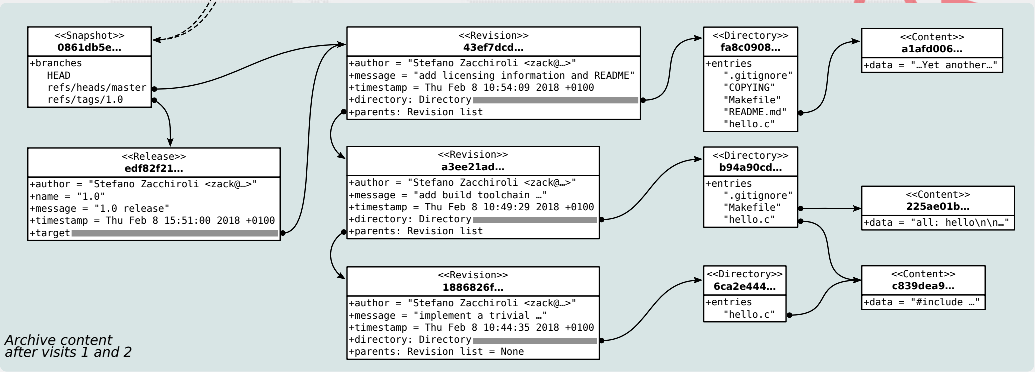
The archive: a (giant) Merkle DAG



The archive: a (giant) Merkle DAG

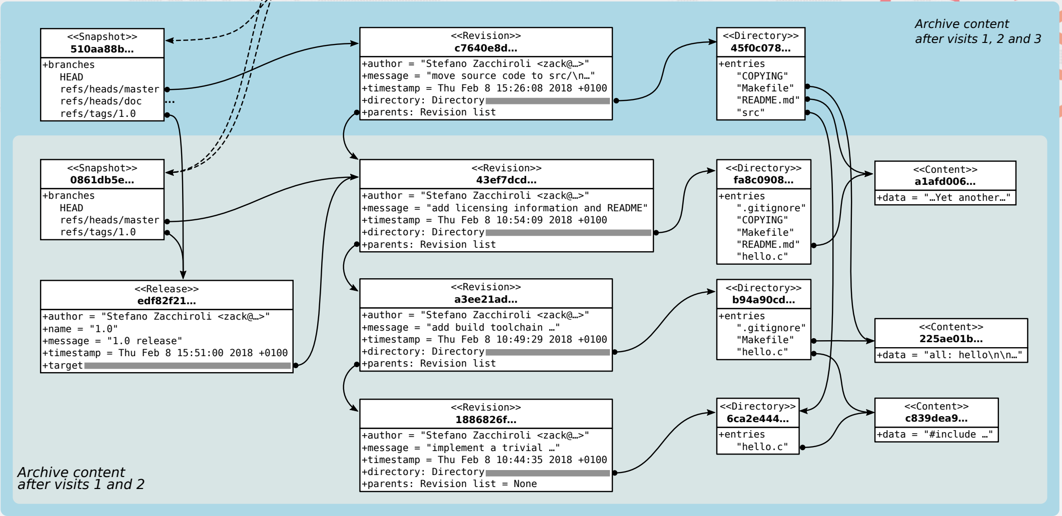


origin	*visit*	*snapshot*	*timestamp*
https://forge.softwareheritage.org/source/helloworld.git	1	0861db5e...	Fri Feb 9 12:38:45 2018 +0100
https://forge.softwareheritage.org/source/helloworld.git	2	0861db5e...	Fri Feb 9 13:29:00 2018 +0100



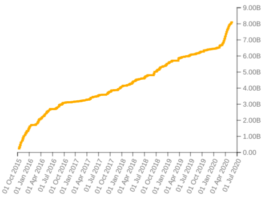
The archive: a (giant) Merkle DAG

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



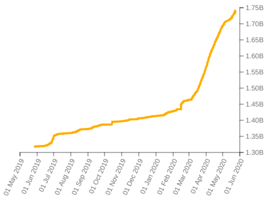
Source files

8,152,137,176



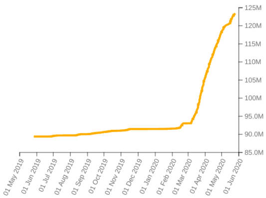
Commits

1,744,034,936



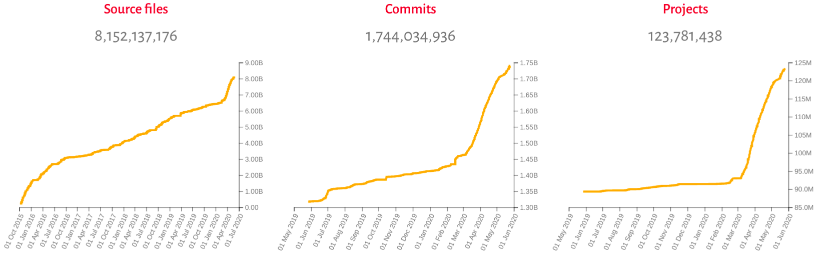
Projects

123,781,438



GitHub · GitLab · Bitbucket · Google code · GITORIOUS · Framagit

HAL archives-ouvertes.fr · debian · npm · R · GNU · Inria inventeurs du monde numérique · python Package Index



- ~400 TB (uncompressed) blobs, ~20 B nodes, ~300 B edges
- The *richest* public source code archive, … and growing daily!

Outline

Early days

- We are in the early days of full-scale explorations of the entire software commons, for both *software health* and other research or practical needs.
- We are also not yet capable of performing analyses at such scale, due to a lack of *resources* (including time!) and/or appropriate *tools* and *techniques*.

In the following I'll review some related work:

- a large-scale dataset encompassing a decent chunk of the software commons
- a technique to exploit such dataset *on a budget*
- a long-term exploration of the growth rate of the software commons

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 (soon)

Sample query — most frequent first commit words

```
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

```
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;
```

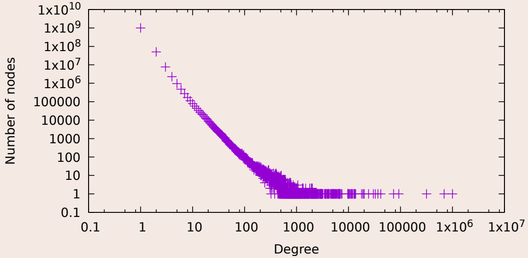
Count	Word
71 338 310	update
64 980 346	merge
56 854 372	add
44 971 954	added
33 222 056	fix

Sample query — fork and merge arities

Fork arity

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

```
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;
```

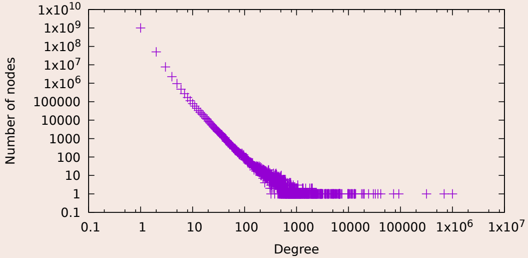


Sample query — fork and merge arities

Fork arity

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

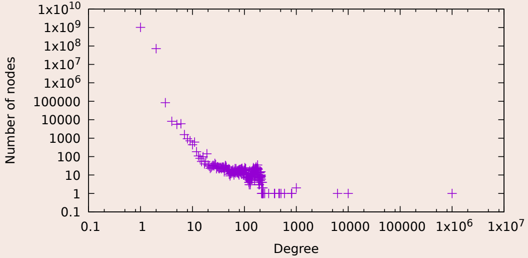
```
SELECT fork_deg, count(*) FROM (
  SELECT id, count(*) AS fork_deg
  FROM revision_history GROUP BY id) t
GROUP BY fork_deg ORDER BY fork_deg;
```



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

```
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

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
```

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)

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.

Definition (The graph of the Web)

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

Properties (1)

- Locality: pages links 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.

→ use reference compression

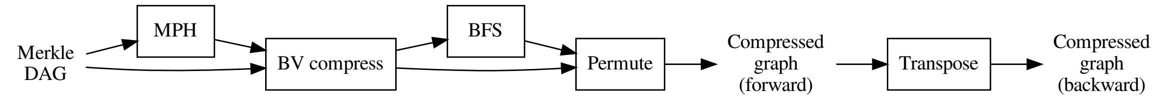
Adjacency lists

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
...	...	...	

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

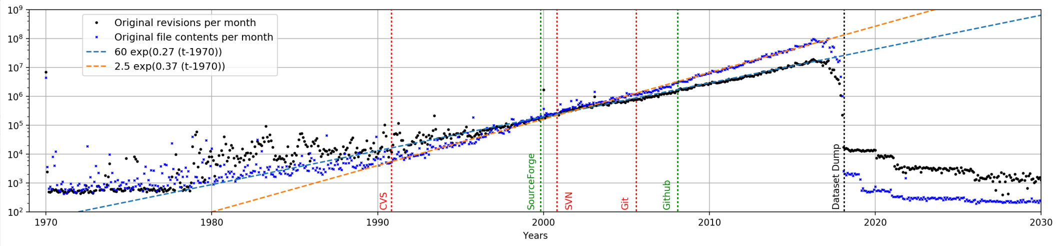
- but the graph is append-only, so…
- …based on expected graph growth rate it should be possible to pre-allocate enough free space in the integer ranges to support amortized incrementality (future work)

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

Original content growth



- 50 years of software commons history. 50 M projects, 4 B blobs, 1 B commits (Software Heritage snapshot, Feb 2018)
- *original* artifacts explored over time, after deduplication
- evidence of exponential growth: original commits doubles every 30 months; blobs every 22 months; original blobs *per commit* doubles every 7 years

Roberto Di Cosmo, Guillaume Rousseau, Stefano Zacchiroli
Software Provenance Tracking at the Scale of Public Source Code
Empirical Software Engineering, 2020

Outline

Wrapping up

- the notion of software health is shaping up nicely, with several dimensions to it and more and more established tools and techniques
- global software health, i.e., the study of software health at the scale of the full software commons is an open challenge that requires exhaustive code libraries, tools, and techniques
- Software Heritage is one such library, containing a significant span of the software commons; tools and techniques to analyze it are now badly needed
- meanwhile, the software commons seems to be doing well in terms of growth; let's dig it further to assess its health!

Contacts

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