How to survive with large mono-repos in Gerrit

Luca Milanesio

Gerrit Code Review Maintainer

GerritForge

About GerritForge

Founded in the UK

HQ in London with presence in Europe (GerritForge Ltd) and the USA (GerritForge Inc.)

Committed to OpenSource and to Gerrit Code Review since 2009

Gerrit Code Review

kibana

APACHE Spark™

AVRO™

What BIG means for you?

Large mono-repo false solutions: git-lfs

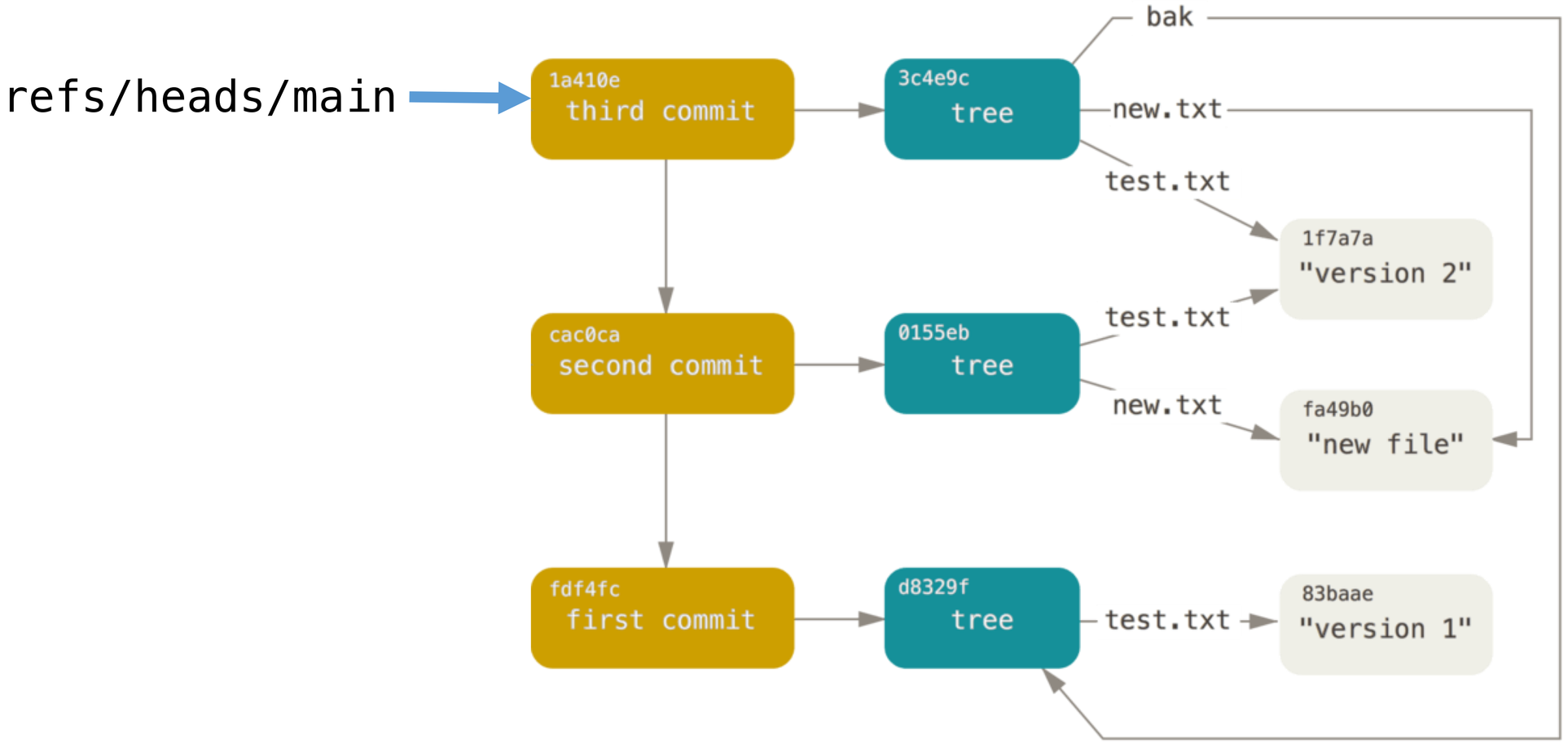
Git LFS often seen as “one-size-fits-all” solution

- Makes the repo smaller
- Reduces bandwidth utilization
- Less disk occupied locally

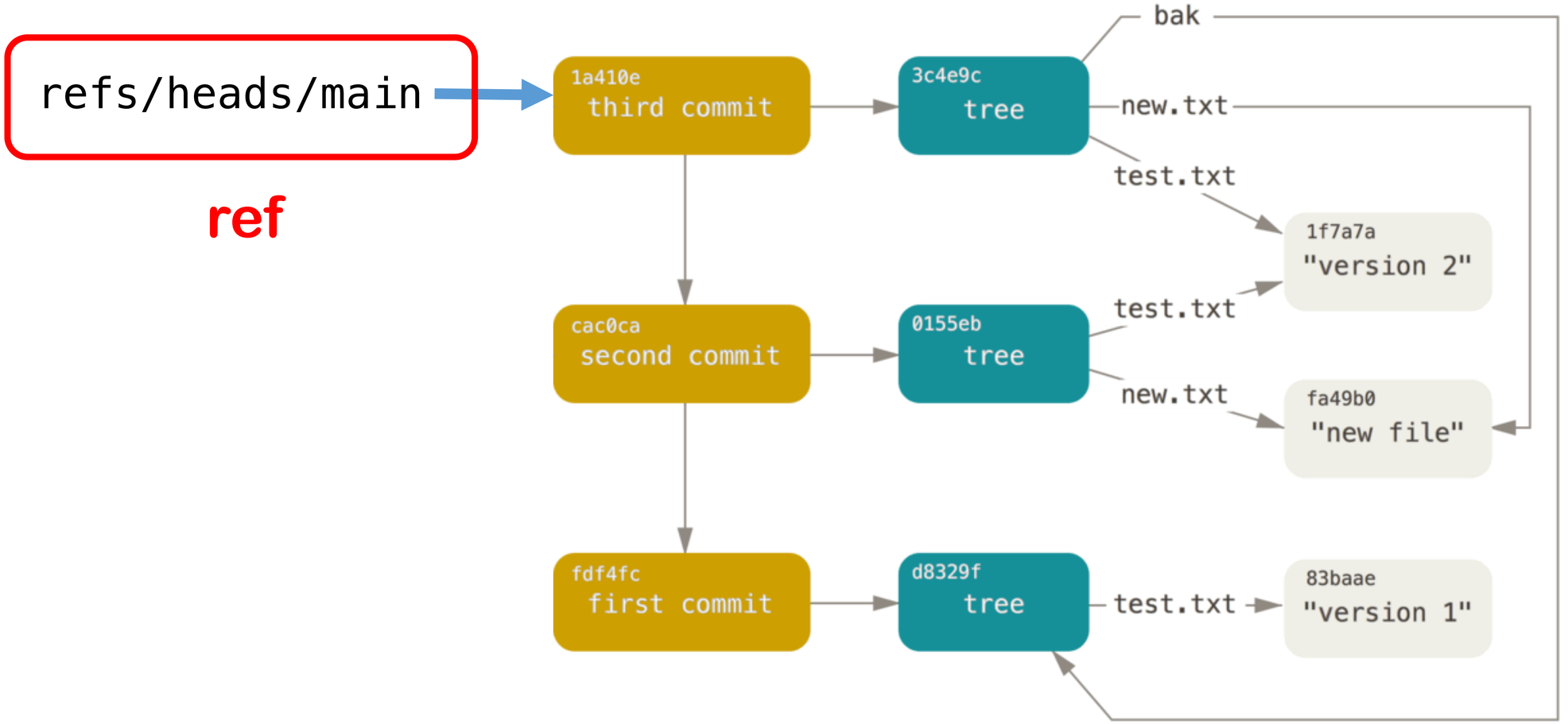
Dark-side of Git LFS

- Requires history rewrite
- Static definition of BLOB size to put onto LFS
- Does not solve large mono-repos issues in Gerrit

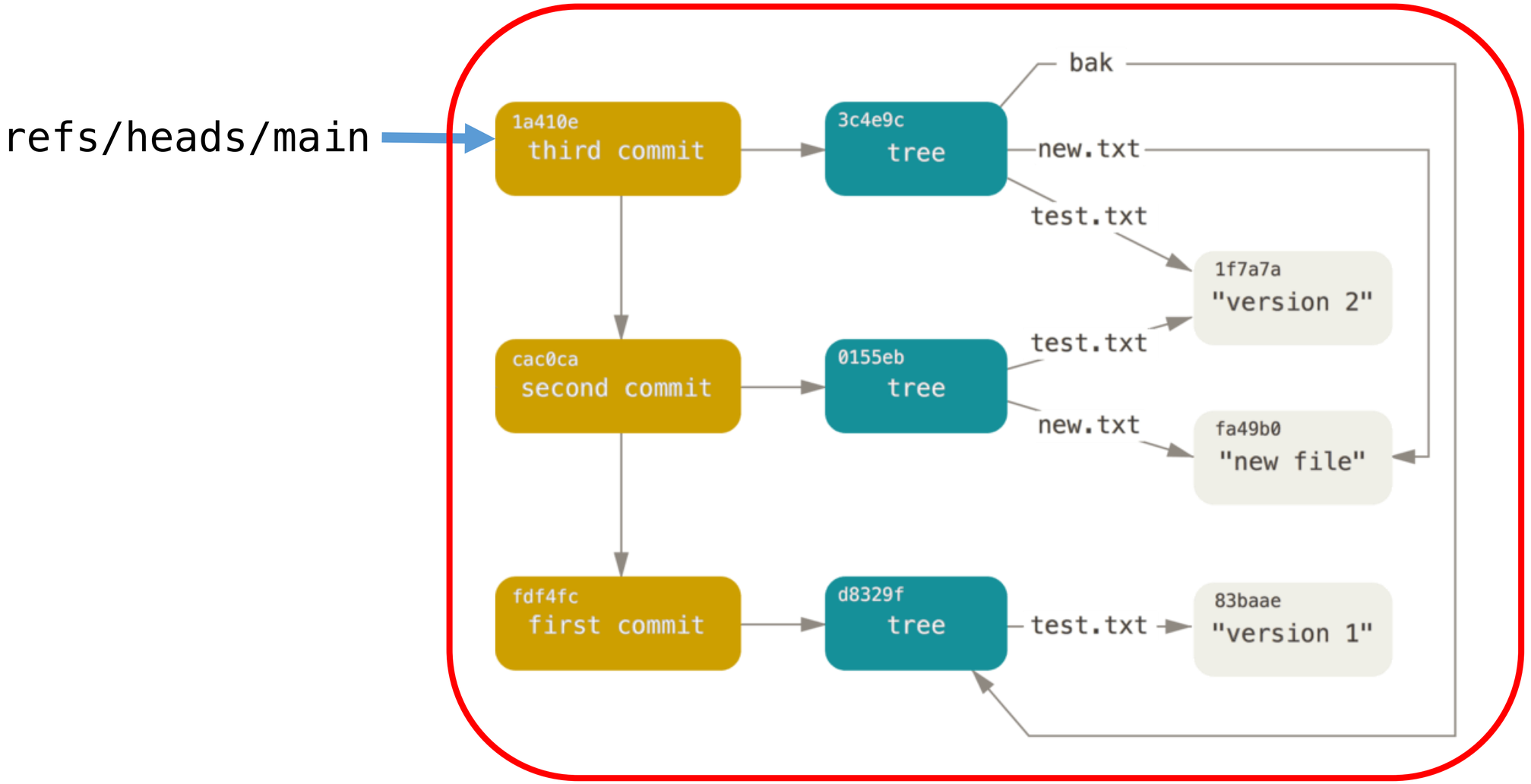
Git data model: recap



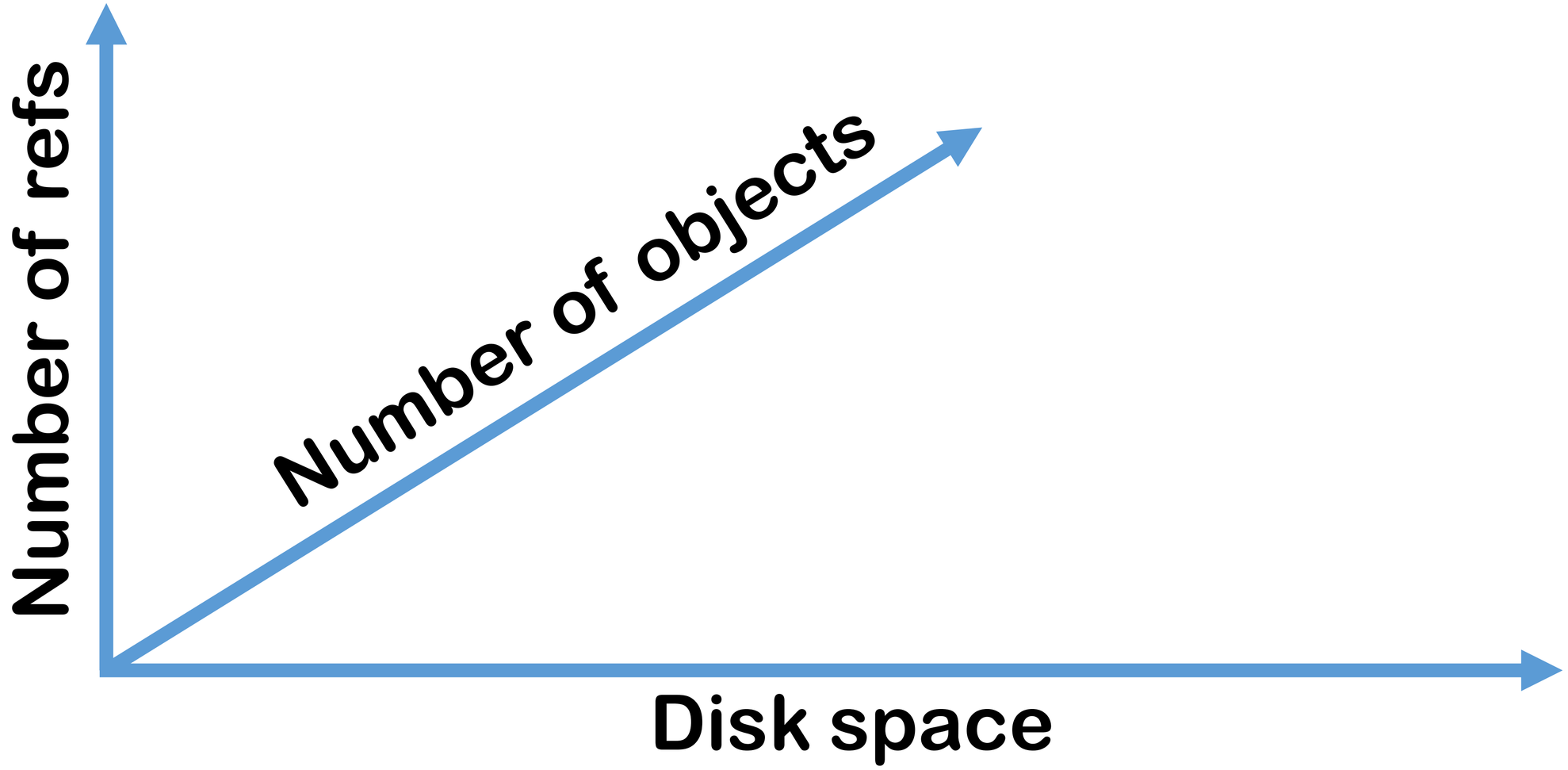
Git data model: recap



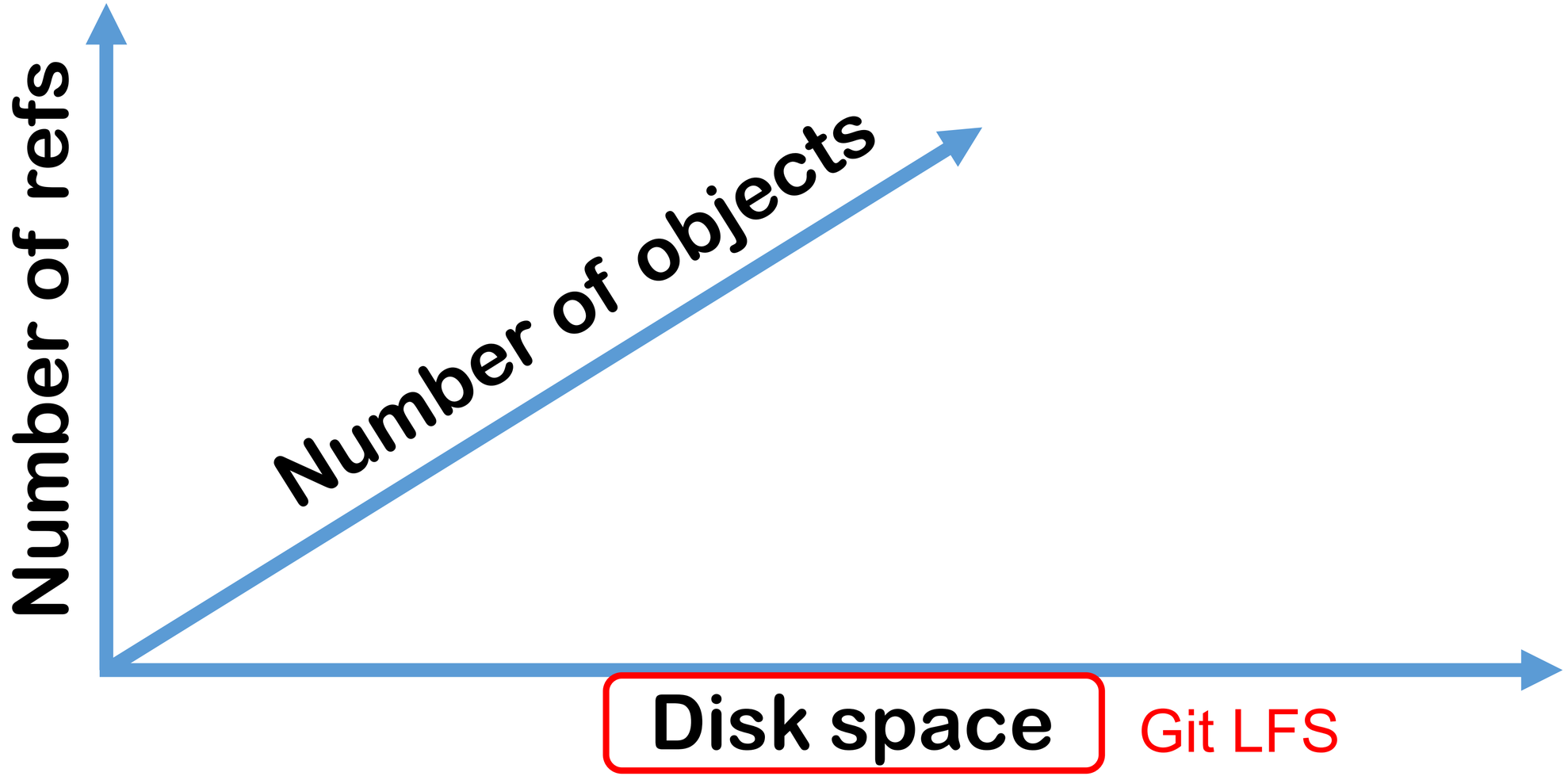
Git data model: recap



BIG repository: dimensions



BIG repository: dimensions



Measuring repo metrics: git-sizer

Developed by GitHub and available on: https://github.com/github/git-sizer

Name	Value	Level of concern
Overall repository size		
* Commits		
* Count	11.3 M	**********
* Total size	7.01 GiB	!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
* Trees		
* Count	18.1 M	**********
* Total size	60.9 GiB	!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
* Total tree entries	1.42 G	**********
* Blobs		
* Count	11.2 M	*******
* Total size	1.06 TiB	!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
* Annotated tags		
* Count	0	
* References		
* Count	2.54 M	!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Biggest objects		
* Commits		
* Maximum size [1]	6.63 MiB	!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
* Maximum parents [2]	2	
* Trees		
* Maximum entries [3]	4.39 k	****
* Blobs		
* Maximum size [4]	731 MiB	!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
History structure		
* Maximum history depth	879 k	*
* Maximum tag depth	0	
Biggest checkouts		
* Number of directories [5]	32.8 k	**********
* Maximum path depth [6]	20	**
* Maximum path length [7]	284 B	**
* Number of files [8]	385 k	*******
* Total size of files [9]	5.25 GiB	*****
* Number of symlinks [10]	363	
* Number of submodules [11]	114	*

Sample execution on the Chromium repository

Large mono-repo: Gerrit-specific issues

Very large number of refs

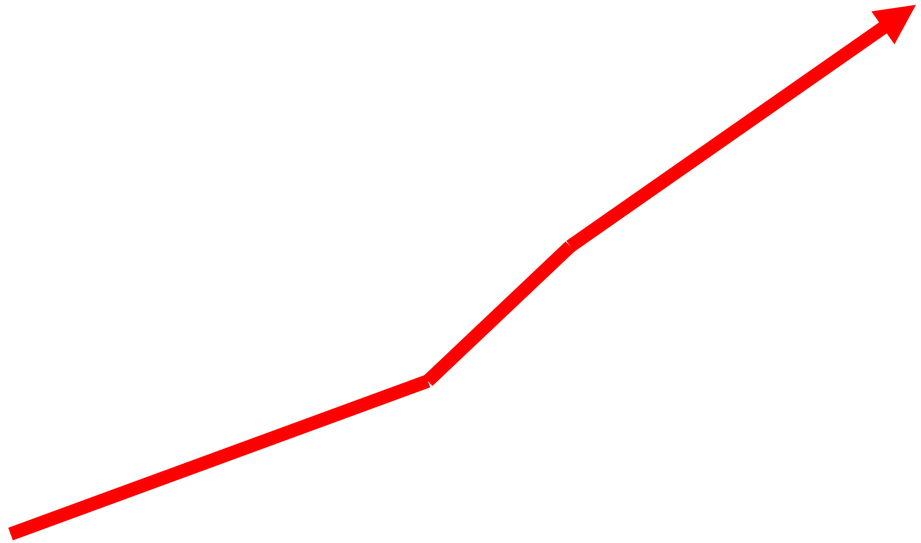
- **Each change has:**
  refs/changes/NN/NNNN/<ps>
  refs/changes/NN/NNNN/meta

- **Example of medium-size Gerrit repo: 100k changes, 10 patch-sets per change**
  100k * 11 = 1.1M refs

Large mono-repo: Gerrit-specific issues

Very old repositories = always increasing number of refs

- **Merged changes stay in the history**
- **Abandoned changes stay in the history**



Large mono-repo: Gerrit-specific issues

Mono-repo = lots of people working on the same repository

- 500 developers creating 1 change per day
- 1 year = 500 * 254 (working days) = 127k changes

What a project having 5 years of history looks like?

- 5 * 127k changes = 635k changes
- Assuming 10 patch-sets per change:
  635k * 11 ~ 7M refs !!!

Large mono-repo: example = Chromium

- > 500k changes
- > 2.5M refs
- > 42M objects
- > 60GB of disk space

Large number of refs: problems

Storage:

- Loose refs => millions of files on the filesystem
- Packed refs => single file getting bigger over time

Access times:

- Loose refs: very slow on NFS: listing of directories is notoriously problematic
- Packed refs: very slow in any condition, but NFS is a killer
- Bitmaps are not helping, as they are not computed for changes

Large number of refs: benchmarks

JGit benchmark on refs lookup and filtering:

- **Created by L.Milanesio, extended and measured by M.Sohn**
- **https://bugs.eclipse.org/bugs/show_bug.cgi?id=576165**

repo	refs	getExactRef trustFolderStat=true [µs/op]	getExactRef trustFolderStat=false [µs/op]
jgit	34,396	71.549	17,027.238
All-Users-sap02	35,129	71.924	18,886.679
gerrit	42,216	71.902	20,700.065
All-Users-sap01	94,456	70.948	65,594.445
go	179,967	72.800	101,953.274
All-Users-Eclipse	304,258	64.669	156,373.621
large-repo-sap-02	1,526,738	73.575	873,704.665
large-repo-sap-01	1,904,551	73.744	1,246,369.098
chromium	2,540,957	101.902	2,155,881.338

repo	getRefsByPrefix trustFolderStat=true [µs/op]	getRefsByPrefix trustFolderStat=false [µs/op]
jgit	1,967.296	21,699.063
All-Users-sap02	451.381	19,661.338
gerrit	2,630.773	27,255.809
All-Users-sap01	2,396.374	64,784.646
go	14,213.907	125,918.218
All-Users-Eclipse	22,297.707	200,519.193
large-repo-sap-02	106,812.654	1,018,671.689
large-repo-sap-01	142,563.930	1,233,735.363
chromium	265,694.897	2,447,755.469

Large number of refs: benchmarks

JGit benchmark on refs lookup and filtering:

- **Created by L.Milanesio, extended and measured by M.Sohn**
- **https://bugs.eclipse.org/bugs/show_bug.cgi?id=576165**

repo	refs	getExactRef trustFolderStat=true [µs/op]	getExactRef trustFolderStat=false [µs/op]
jgit	34,396	71.549	17,027.238
All-Users-sap02	35,129	71.924	18,886.679
gerrit	42,216	71.902	20,700.065
All-Users-sap01	94,456	70.948	65,594.445
go	179,967	72.800	101,953.274
All-Users-Eclipse	304,258	64.669	156,373.621
large-repo-sap-02	1,526,738	73.575	873,704.665
large-repo-sap-01	1,904,551	73.744	1,246,369.098
chromium	2,540,957	101.902	2,155,881.338

repo	getRefsByPrefix trustFolderStat=true [µs/op]	getRefsByPrefix trustFolderStat=false [µs/op]
jgit	1,967.296	21,699.063
All-Users-sap02	451.381	19,661.338
gerrit	2,630.773	27,255.809
All-Users-sap01	2,396.374	64,784.646
go	14,213.907	125,918.218
All-Users-Eclipse	22,297.707	200,519.193
large-repo-sap-02	106,812.654	1,018,671.689
large-repo-sap-01	142,563.930	1,233,735.363
chromium	265,694.897	2,447,755.469

Over 2s for a single ref lookup or filter!!!

Large number of objects: problems

Storage and bitmaps:

- Loose objects => tens millions of files on the filesystem
- Packfiles => bitmaps rely on a single packfile, which is getting bigger over time

Access times:

- Loose refs: very slow on NFS: listing of directories is notoriously problematic
- Packed refs: JGit is quite clever to be fast on NFS

Large number of objects: problems with packfiles

The struggle with packfiles:

- **Objective: one single packfile with all objects**
- **Reality: hundreds of packfiles going up and down**

Rationale:

- **Single packfile obtained by a Git GC with pruning**
- **Pruning not practical for on-line Git GC (risk of removal of in-flight pushed objects)**
- **Every Git push creates a new packfile**

Large number of objects: search-for-reuse/sizes

Digression on Git clone phases:
(https://gerrit-documentation.storage.googleapis.com/Documentation/3.4.1/logs.html#_sshd_log)

- **Refs advertisement and negotiation**
- **Bitmaps lookup optimization**
- **Search-for-reuse / search-for-sizes**
- **Counting**
- **Compressing / Writing to the wire**

Large number of objects: search-for-reuse/sizes

Digression on Git clone phases:
(https://gerrit-documentation.storage.googleapis.com/Documentation/3.4.1/logs.html#_sshd_log)

- **Refs advertisement and negotiation**
- **Bitmaps lookup optimization**
- **Search-for-reuse / search-for-sizes**
- **Counting**
- **Compressing / Writing to the wire**

This is a killer for large number of objects / packfiles

search-for-reuse/sizes: complexity

For each object, scan all packfiles looking for the best delta:

- 10M objects
- 100 packfiles
- 10M * 100 = 1BN of scans

What are YOUR problems with large mono-repos?

Large mono-repo: problems

- Disk space (server and client)
- Network bandwidth
- Long clone and fetch times
- Infinite refs negotiation
- High system load
- Timeouts
- Clucking git operations of cloned repo (logs, diffs)
- …

Large number of refs: reftable

Available since Gerrit v3.1 (JGit 5.6)
Implemented by Shawn on Google's JGit backend
Implemented for the OpenSource JGit by Han-Wen (thanks ☺)

Fully available on Gerrit v3.5 with the command to convert repos to ref-table storage

⚠ Not yet available on Git (Han-Wen is working on it)

More details:
https://www.git-scm.com/docs/reftable

Large number of refs: refs caching (WIP)

M.Sohn working on an in-memory cache for refs:
(https://git.eclipse.org/r/c/jgit/jgit/+/186205)

repo	refs	getExactRef trustFolderStat=false [µs/op]	getExactRef cached [µs/op]
jgit	34,396	17,027.238	0.646
All-Users-sap02	35,129	18,886.679	0.908
gerrit	42,216	20,700.065	0.744
All-Users-sap01	94,456	65,594.445	1.200
go	179,967	101,953.274	0.954
All-Users-Eclipse	304,258	156,373.621	1.067
large-repo-sap-02	1,526,738	873,704.665	1.800
large-repo-sap-01	1,904,551	1,246,369.098	1.912
chromium	2,540,957	2,155,881.338	1.636

repo	getRefsByPrefix trustFolderStat=false [µs/op]	getRefsByPrefix cached [µs/op]
jgit	21,699.063	35.448
All-Users-sap02	19,661.338	6.381
gerrit	27,255.809	35.070
All-Users-sap01	64,784.646	35.594
go	125,918.218	365.114
All-Users-Eclipse	200,519.193	295.683
large-repo-sap-02	1,018,671.689	1,543.187
large-repo-sap-01	1,233,735.363	1,995.160
chromium	2,447,755.469	2,462.512

Large number of refs: refs filtering

Available as libModule from Gerrit v2.16 onwards
(https://gerrit.googlesource.com/modules/git-refs-filter/)

- Mimic the “hideRefs” config in Git
  (https://git-scm.com/docs/git-config/2.32.0#Documentation/git-config.txt-transferhideRefs)

- Define a new custom permission for refs-filtering of
  - refs/changes/NN/NNNN/meta
  - closed changes
  - other custom refs patterns

Large number of refs: refs filtering – case study

Real-life case-study of refs-filter:

- **Repository with 1.5M refs**
- **Mirror clone without refs-filter: 45 mins**
- **Mirror clone with refs-filter: 3 mins (15x improvement)**

Needs enabling of the persistent change cache, otherwise the change meta-data lookup eats up all the advantages

Large number of objects: search-for-reuse deadline

Introduced by Ponch in JGit master
(https://git.eclipse.org/r/c/jgit/jgit/+/181355)

- **Define a maximum deadline for search-for-reuse to complete**
  pack.searchForReuseTimeout = 60s
- **When deadline expires, a non-optimal object representation is taken (may transfer more data)**

- **Performance improvement observed:**
  without deadline: 20 mins, with deadline: 5 mins (4x improvement)

Alternatives: repository pruning

Do you really need all the review history online?

- Two copies of the repository: full-history and pruned-history
- What to prune?
  - Auto-abandon changes (e.g. 1 month) and delete abandoned changes not resumed for a long time (e.g. 3 months)
  - Collapse merged patch-sets into latest

Q&A: excited about the future of Gerrit?

Image from: http://cypp.rutgers.edu/ru-voting/political-information/public-opinion-polls/

Wants to know more?

GerritForge

GerritForge.com/contact