Git Is For Data

Yucheng Low
CEO/Co-Founder Xetdata
Git Is For Data
Code collaboration

1st generation (versioning of individual files)
1972 – SCCS
1982 – RCS

2nd generation (centralized repository-level versioning)
1986 – CVS
1995 – Perforce
2000 – SVN

3rd generation (distributed version control model)
2005 – Git
2005 – Mercurial
2005 -- Bazaar
Code collaboration

1st generation (versioning of individual files)
1972 – SCCS
1982 – RCS

2nd generation (centralized repository-level versioning)
1986 – CVS
1995 – Perforce
2000 – SVN

3rd generation (distributed version control model)
2005 – Git
2005 – Mercurial
2005 – Bazaar
Data collaboration

Data: Any collection of bytes of value

While data storage has scaled with distributed filesystems, blobstores, etc.

Collaboration is still just

Build and share report? Run some experiments? Make a new version?

Make A Copy
Data is rarely standalone

**Code With Data**
- Games
- Interactive media
- Intelligent applications

**Data With Code**
- ML modeling
- Reports / notebooks
How solutions work for code and data today?
Git LFS

> git lfs track *.parquet
Git LFS

> git lfs track *.parquet
> git add *

User's Checkout

repo/
  README.md
  code.py
  data.parquet

What Git Stores

repo/
  README.md
  code.py
  data.parquet

Clean (dehydrate)

Pointer file

version https://git-lfs.github.com/spec/v1
oid sha256:4d7a214614ab2935c...
size 12345
Git LFS

> git clone ...

User’s Checkout

repo/
  README.md
  code.py
  data.parquet

What Git Stores

repo/
  README.md
  code.py
  data.parquet

Smudge (hydrate)

Pointer file

version https://git-lfs.github.com/spec/v1
oid sha256:4d7a214614ab2935c...
size 12345
Git LFS, Git DVC, Git Annex, etc. are architecturally similar

“All problems in computer science can be solved by another level of indirection”

• Explicit storage decisions
• Frequently, no different than a download manifest + curl

Data should be 1st class
<table>
<thead>
<tr>
<th>Xethub Web Frontend</th>
</tr>
</thead>
<tbody>
<tr>
<td>Git Server</td>
</tr>
<tr>
<td>Content Addressed Store (CAS)</td>
</tr>
</tbody>
</table>

Git Xet Extension
<table>
<thead>
<tr>
<th>Git Xet Extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xethub Web Frontend</td>
</tr>
<tr>
<td>Git Server</td>
</tr>
</tbody>
</table>
Git Xet Extension

Full Git Compatibility:

- `git add dataset/*`
- `git commit -a -m "adding large files"`
- `git push`
- [all esoteric git commands we have tested]

Uniform UX: Code and data files are treated the same.
Git Xet

Data-informed heuristic optimizes for best performing storage location of each file.

User’s Checkout
repo/
  README.md
code.py
data.parquet

Clean (dehydrate)

What Git Stores
repo/
  README.md
code.py
data.parquet

Smudge (hydrate)
Store a big binary file

Large_file.bin
Content Defined Chunking

Large_file.bin

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor

Rolling Hash
Content Defined Chunking

Large_file.bin

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor

Rolling Hash

If Hash % 16384 == 0, this is a chunk boundary

Target average 16KB chunks
Content Defined Chunking

Large_file.bin

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor

If Hash % 16384 == 0, this is a chunk boundary
Content Defined Chunking

Large_file.bin

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor

If Hash % 16384 == 0, this is a chunk boundary
Content Defined Chunking

If Hash % 16384 == 0, this is a chunk boundary

Large_file.bin

CDC procedure is robust to insertions and deletions as chunk boundaries are decided based on the data.
Content Defined Chunking

Large_file.bin

What is the target chunk size?
Small chunks ➔ better dedupe.
Large chunks ➔ more efficient to store

Can we get the best of both worlds?
Content Defined Merkle Tree

Large_file.bin

Root

H 12345

H 123

H 1
Data 1

H 2
Data 2

H 3
Data 3

H 4
Data 4

H 5
Data 5

H 6
Data 6

H 7
Data 7

H 8
Data 8

H 45

H 678
New Chunks

Large_file.bin

Root

H 12345
H 1
H 2
H 3
Data 1
Data 2
Data 3

H 123

H 45
H 4
H 5
Data 4
Data 5

H 678
H 6
H 7
H 8
Data 6
Data 7
Data 8
New Chunks

Large_file.bin

Root

H 12345

H 123

H 1

Data 1

H 2

Data 2

H 3

Data 3

H 4

Data 4

H 5

Data 5

H 6

Data 6

H 7

Data 7

H 8

Data 8

[Data 5][data 6][Data 7][Data 8]...

CAS 0x456

Up to 16MB
Smudging (hydration)

![Diagram showing data nodes and hash values](image-url)
Smudging (hydration)

Concatenate range 0x123[...] with range 0x456[...]
Merkle Tree Data Dedupe

Data dedupe with small chunks:
supporting insertions and deletions.

Low CAS overhead:
Large block sizes reduce overhead.

High data locality:
If a range in a block is required, it is likely that the rest of the block is also required.
Cord-19 Dataset Benchmark

Time evolving collection of covid-19 papers with full text, authors, abstracts and document embeddings.

- 2021-02-01 Snapshot
- 2021-02-08 Snapshot
- 2021-03-01 Snapshot
- 2022-06-06 Snapshot

```
git add *
git commit -m "2021-02-01"
git add *
git commit -m "2021-02-08"
git add *
git commit -m "2021-03-01"
git push
```

50 versions

81.5 GB
Covid Papers Dataset 50 versions

Naïve Cumulative Size:
2.45TB

LFS (Any File Level Dedupe):
545GB

Xet: 287GB
2x smaller!

Very dumb chunker. Can we tune it for different file types?
Covid Papers Dataset 50 versions

Naïve Cumulative Size:
2.45TB

LFS (Any File Level Dedupe):
545GB

Xet: 287GB

Xet + CSV Chunker:
87GB

6.3x smaller!
Xet + CSV Chunker Total Storage: 87GB
Final 2022-06-06 Snapshot Size: 81.5GB

Only 5.5GB more for all 49 historical versions.

Effective Data Dedupe can substantially improve ML dataset performance.
> git clone IncrediblyGiantRepo

You probably do not want to do this
Virtual Filesystem

Explore large datasets in seconds:

**Laion400M dataset**
54GB Parquet files of URLs and other metadata

git xet mount https://xethub.com/XetHub/Laion400M

> duckdb.query("select COUNT(*) from 'data/*.parquet'")
> 413871335
> duckdb.query("select LICENSE, count() .. group by LICENSE")
> ...

~ 1GB downloaded  2% of the dataset

As fast as local filesystem after first access as local caching is performed
Virtual Filesystem

Bridge the gap between experiment and production
   Same code works for both.
   Stream to GPU machines, no manual data partitioning

Writable Mounts (WIP)
   Virtual Filesystem that acts like a git repo
   “Dropbox with git semantics”
• **Full Git Compatibility** where Data and Code are 1st class

• **High performance dedupe architecture** enables cheap versioning and common ML dataset operations.

• **Virtual Filesystems** to enable scaling to very large repos
Many MLOps concepts are exactly DevOps concepts

Data Quality monitoring $\rightarrow$ Continuous Integration
Data Pipelines $\rightarrow$ Build Dependencies
Model Versioning $\rightarrow$ Build Artifacts

......

MLOps == DevOps + Data Scale

Demonstrate that we can scale the foundations.
Much More Work To Be Done

**Scaling further:**
Scaling designs to support > 100TB

**Writable Mounts:**
Virtual Filesystem that acts like a git repo.
“Shared filesystem with git commit semantics”

**Collaboration Patterns:**
Github has collaboration patterns for code. What are the right patterns for data?
Raised $7.5M Seed Round.

Public Beta: [https://xethub.com](https://xethub.com)

We are hiring: [careers@xethub.com](mailto:careers@xethub.com)