DVID
Distributed, Versioned, Image-Oriented Datastore

Fly EM

William T. Katz
Steve Plaza
Goal: Create structural connectome (neurons + connections)
NEED TO PROVIDE VERSIONING

Snapshots at time of publication or sharing.
NEED DATA API

Clients include proofreading tools and automated programs.
DVID: HIGH-LEVEL API + VERSIONING
DVID: DEMO HTTP API

Start Local 3d Demo

Show Built-in Help Page
SCIENCE (CONNECTOMICS) API

Give me a 500 x 500 pixel 2d image with left corner at (30,20,10):

GET http://api/node/2a3/grayscale/raw/xy/500_500/30_20_10

Show me neuron labels larger than 100 million voxels:

GET http://api/node/2a3/bodies/sizerange/100000000

Merge some labels into label 123:

POST http://api/node/2a3/bodies/merge/123

Which neurons are connected to the neuron labeled 123?

GET http://api/node/2a3/labelgraph/neighbors/123
Embedded LevelDB
Subset of Data

Embedded LevelDB
20 TB Data
DVID DISTRIBUTED VERSIONING

- Use Directed Acyclic Graphs (DAGs) like software version control systems (e.g., git, mercurial).
- Each version "node" gets UUID at creation, not content hash. Uncommitted version = "working directory".

Ingest grayscale image volume. Ingest automatic segmentation.
DVID DISTRIBUTED VERSIONING

- Use Directed Acyclic Graphs (DAGs) like software version control systems (e.g., git, mercurial).
- Each version "node" gets UUID at creation, not content hash. Uncommited version = "working directory".

Ingest grayscale image volume. Ingest automatic segmentation.
DVID DISTRIBUTED VERSIONING

- Use Directed Acyclic Graphs (DAGs) like software version control systems (e.g., git, mercurial).
- Each version "node" gets UUID at creation, not content hash. Uncommitted version = "working directory".

Diagram:
- Ingest grayscale image volume.
- Ingest automatic segmentation.
- Proofreader A merges a false split.
DVID DISTRIBUTED VERSIONING

- Use Directed Acyclic Graphs (DAGs) like software version control systems (e.g., git, mercurial).
- Each version "node" gets UUID at creation, not content hash. Uncommitted version = "working directory".

Proofreader A merges a false split.

Ingest grayscale image volume.
Ingest automatic segmentation.
• Use Directed Acyclic Graphs (DAGs) like software version control systems (e.g., git, mercurial).
• Each version "node" gets UUID at creation, not content hash. Uncommitted version = "working directory".
DVID DISTRIBUTED VERSIONING

- Use Directed Acyclic Graphs (DAGs) like software version control systems (e.g., git, mercurial).
- Each version "node" gets UUID at creation, not content hash. Uncommitted version = "working directory".

![Diagram]

- Ingest grayscale image volume.
- Ingest automatic segmentation.
- Proofreader A merges a false split.
- Proofreader B splits a few false merges.
DVID DISTRIBUTED VERSIONING

- Use Directed Acyclic Graphs (DAGs) like software version control systems (e.g., git, mercurial).
- Each version "node" gets UUID at creation, not content hash. Uncommitted version = "working directory".

![Directed Acyclic Graph Diagram]

- 8fc4
- Ingest grayscale image volume. Ingest automatic segmentation.
- e14d
- Proofreader A merges a false split.
- ec80
- Proofreader B splits a few false merges.
- d353
- Merge edits by A and B.
DVID DISTRIBUTED VERSIONING

- Use Directed Acyclic Graphs (DAGs) like software version control systems (e.g., git, mercurial).
- Each version "node" gets UUID at creation, not content hash. Uncommitted version = "working directory".
Most data changes are at root where we ingest images and do initial segmentation.

As we "commit" each node, the size of immutable data grows.

- Large number of changes
- Small number of changes
- Ingest grayscale image volume.
- Ingest automatic segmentation.
- Proofreader B splits a few false merges.
- Merge edits by A and B.
IMMUTABLE DATA IN DAG

- Most data changes are at root where we ingest images and do initial segmentation.
- As we "commit" each node, the size of immutable data grows.

**Diagram:**
- Large number of changes
- Small number of changes
- Ingest grayscale image volume.
- Ingest automatic segmentation.
- Proofreader B splits a few false merges.
- Merge edits by A and B.
"POLYGLOT" PERSISTENCE IN DVID

- Immutable data can be stored more cheaply and have more efficient caching.
- Mutable data size is relatively small. Can use faster, more costly systems to counter distributed transactions.
PUBLISHING DATA

http://emdata.janelia.org
One-column dataset
DISTRIBUTED VERSIONING
DISTRIBUTED VERSIONING
DISTRIBUTED VERSIONING
SUMMARY OF DVID FEATURES

- DVID supports **domain-specific APIs** through easily added data types.
- Data types mainly use **ordered key-value storage** but can proxy.
- Versioning using a DAG allows provenance without necessarily storing every change.
- Versioning allows most of data to be stored in cost-effective immutable stores.
- Distribution will follow a push/pull/clone system similar to git. (In progress)
BRANCH EXPERIMENTS

Ingest grayscale image volume.

39ae → 8fc4

Segmentation Experiment 1

e14d → 8fc4

Segmentation Experiment 2

ec80 → 8fc4

Best Segmentation