The Block Object Storage Service (bossDB): A Cloud-Native Approach for Petascale Neuroscience Discovery

Dean M. Kleissas[1], Robert Hider Jr.[1], Derek Pryor[1], Timothy Gion[1], Priya Manavalan[1], Jordan Matelsky[1], Alex Baden[2], Kunal Lillaney[2], Randal Burns[2], Denise D'Angelo[1], William Gray Roncal[1]*, Brock Wester[1]*
[1]Johns Hopkins University Applied Physics Laboratory; Laurel, Maryland. [2]Johns Hopkins University; Baltimore, Maryland
Correspondence: brock.wester@jhuapl.edu, william.gray.roncal@jhuapl.edu
* Authors contributed equally

November 10, 2017

Abstract

Large volumetric neuroimaging datasets have grown in size over the past ten years from gigabytes to terabytes, with petascale data becoming available and more common over the next few years. Current approaches to store and analyze these emerging datasets are insufficient in their ability to scale in both cost-effectiveness and performance. Additionally, enabling large-scale processing and annotation is critical as these data grow too large for manual inspection. We propose a new cloud-native managed service for large and multi-modal experiments, providing support for data ingest, storage, visualization, and sharing through a RESTful Application Programming Interface (API) and web-based user interface. Our project is open source and can be easily and cost-effectively used for a variety of modalities and applications.

1 Introduction

Mapping the brain to better understand cognitive processes and the biological basis for disease is a fundamental challenge of the 21st century that is only now emerging as a realistic endeavor, realizing the dreams of early neuroscientists such as Ramón y Cajal, who were limited to sketching brain maps in ink, one neuron at a time. Technological advances in neuroscience have exploded over the last ten years, making it almost routine to image high-resolution (submicron) brain volumes in many laboratories around the world [1, 2, 3, 4, 5].

As new tools for interrogating tissue at these resolutions advance and become more common, a centralized data archive is needed to enable the large (tera- to petascale) storage, visualization, and discovery processes, and address some of the challenges identified by the community [3, 6]. While research groups are beginning to embrace data archives, most treat the system as simply a place to deposit finalized data, with raw datasets generated and stored in a custom format and analyzed and inspected with custom software. At this scale, it is quickly becoming impossible for researchers to characterize many of the underlying properties; indeed, for petascale volumes, it is likely that most of a data volume is never viewed in detail by a human. Additionally, conventional approaches for automatically or semi-automatically reconstructing neuronal maps focus on building methods for small volumes, and scaling these tools to operate on multi-terabyte or petabyte data volumes is often either unachievable or significantly beyond the capabilities and budgets of a research group.

Large datasets are a resource incredibly rich in scientific content, which should be shared with others to best leverage the investment of time and resources and to fully exploit the value of the data. Due to the challenges in collection, storage, and analysis of terascale and petascale data volumes, few public datasets of this size are routinely shared, although many such volumes exist on local storage and a deluge of new data is forthcoming [7, 8, 9].

We considered use cases such as the first fully-automated pipelines for processing and assessing X-ray Microtomography (XRM) [10] and electron microscopy (EM) datasets [1, 2, 4] and work by many academic laboratories around the world to understand state-of-the-art approaches and their limitations. Through these efforts, we have learned that high-performance and scalable data storage is an

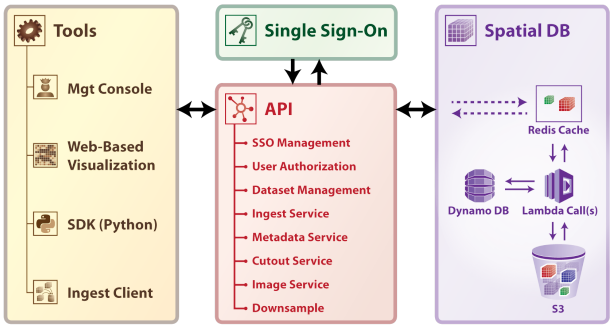

Figure 1: A high-level schematic of bossDB platform.

essential component of any modern connectomics effort. In designing our Block and Object Storage Service Database (bossDB), we researched several related efforts, including DVID [11] which excels in versioned terascale storage and CATMAID [12] which provides a mature manual annotation platform. We previously worked with NeuroData to develop ndstore [13], which originated and implemented many of the design principles necessary to store and access high-dimensional imaging datasets, including an efficient internal data representation and associated spatial indexing scheme; the Reusable Annotation Markup for Open Neuroscience (RAMON), an annotation schema for connectomics [14]; an API to remotely access services; and MATLAB and Python toolkits to facilitate usability. Based on this prior research and an understanding of the evolving requirements driven by new and maturing imaging modalities, we focused our efforts on developing a robust, cloud-native petascale datastore.

2 Methods

Towards enabling large-scale, collaborative research we developed and deployed a cloud-native data archive to support the storage, analysis, and sharing of large spatial datasets. Service-oriented architectures have continued to grow in popularity and possess many appealing properties when designing a cloud-based data archive [15]. Our solution, called the Block and Object Storage Service (bossDB), is currently deployed in the Amazon Web Services (AWS) cloud and has been architected from the ground up to leverage the capabilities of the cloud and ensure a highly-available, scalable, and cost-efficient system. bossDB was designed and implemented by a team of professional engineers, providing a robust, tested product for the research community.

2.1 Spatial Database

The spatial database is the foundation of bossDB, and uses the strengths of the cloud to efficiently store and index massive multi-dimensional image and annotation datasets (i.e. multi-channel 3D image volumes). A core concept is our managed storage hierarchy, which automatically migrates data between affordable, durable object storage (i.e. Amazon S3) and an in-memory data store (i.e. Redis), which operates as read and write cache database for faster IO performance. This allows for storage of large volumes at a low cost, while providing low latency to commonly accessed regions. We utilize AWS Lambda to perform parallel IO operations between the object store layer and memory cache layer and DynamoDB for indexing. These serverless technologies allow bossDB to rapidly and automatically scale resources during periods of heavy operation without incurring additional costs while idle.

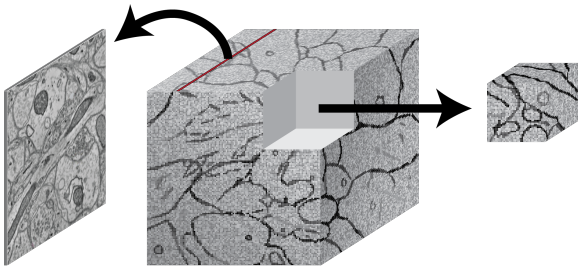

Figure 2: An illustration showing image slices (left) being composed into 3D cuboid volumes (middle). Arbitrary requests may be made to extract image regions of interest (right).

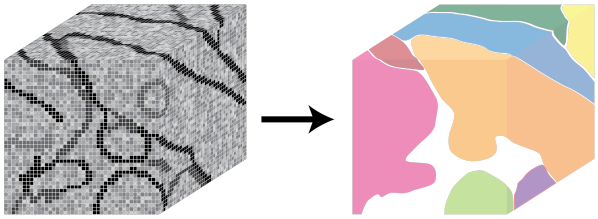

Figure 3: An illustration showing annotations, composed of voxel labels (left and center) and how a unique annotation identifier can represent a unique object in the image data (right).

The bossDB spatial database is designed to store petascale, multi-dimensional image data (i.e. multi-channel 3-dimensional image volumes, with optional time series support) and associated co-registered voxel annotations. In this context, voxel annotations are unsigned 64-bit integer labels stored in a separate channel that is in the same coordinate frame as the source image data. Each unique

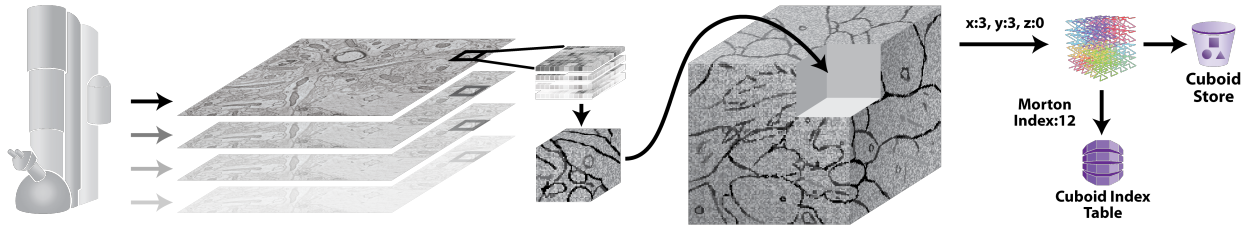

Figure 4: An illustration showing how large 2D image slices generated by an electron microscope are reformatted as cuboids, which fit into a larger 3D volume, indexed using a z-order curve.

uint64 value represents a unique object (e.g., neuron, synapse, organelle). A user can then label voxels that have some semantic meaning, typically the result of manual annotation or automated processing. The database maintains an index of annotation locations, enabling efficient spatial querying and data retrieval.

The internal representation of volumetric data is inherited from previous collaborative work with the NeuroData ndstore project [13]. Here, we operate on small cuboids, or 3D chunks of data (e.g., $512 \times 512 \times 16$ voxels), which are stored in Amazon S3 as compressed C-order arrays. Cuboids are indexed using a Morton-order space-filling curve, which maps the 3D location of each cuboid to a single dimension. In addition, annotations are indexed so `bossDB` can quickly retrieve which annotation IDs exist in an individual cuboid, and in which cuboids a unique ID exists. With these indices, all of which are stored in auto-scaling Amazon DynamoDB tables, the `bossDB` API can provide spatial querying of annotations by ID and efficient retrieval of arbitrary data volumes. The database will also render and store a resolution hierarchy through down sampling of a dataset, which is critical for visualization applications to efficiently provide low-resolution views and useful when processing large datasets.

The spatial database supports various bit-depths (i.e. uint8, uint16 image channels and uint64 annotation channels) and we continue to work with users to provide additional bit-depth and data formats as needed.

2.2 Single Sign-On (SSO) Identity Provider

A centralized and standalone authentication server provides single sign-on functionality for `bossDB` and integrated tools and applications. This allows `bossDB` to control permissions internally and operate securely, while maintaining the ability to federate with other data archives in the future.

A robust authentication and authorization system provides many important benefits, such as the ability to keep some data private while other data public and to control what actions individual users are permitted to perform. We use the open source software package Keycloak as an identity provider to manage users and roles. We created a Django OpenID Connect plugin to simplify the integration of services with the SSO provider.

Our identity provider server intentionally runs independently from the rest of `bossDB` system, forcing the `bossDB` API to authenticate just like any other SSO integrated tool or application, and making future federation with other data archives or authentication systems easy. The Keycloak server is deployed in a highly-available configuration using multiple instances across Availability Zones in an auto-scaling group that sits behind an Elastic Load Balancer.

2.3 Application Programming Interface (API)

As the primary interface to `bossDB`, the API is collection of versioned, RESTful web services. It enforces access permissions and organizes data in a logical data model for spatial and functional results. Because the API is versioned, the `bossDB` storage engine can support significant changes while still maintaining backwards compatibility with legacy applications and tools. All requests to the API are authenticated through the SSO service or via a long-lived API token, which enables tracking usage and throttling requests as needed to manage cost and ensure reliable performance (e.g., high bandwidth power user vs. a limited guest user). The services `bossDB` provides are summarized below:

2.3.1 SSO Management and User Authorization

A set of services to manage users, roles, groups, and permissions. Roles limit what actions a user can perform on the system, while permissions limit what data users can access or manipulate. Permissions are applied to `bossDB` datasets via groups, making it easy to manage and control access for both individuals and teams. Through the application of permissions, a researcher or administrator can choose to keep a dataset private, share with collaborators, or make it publicly available.

2.3.2 Dataset Management

The bossDB API organizes data into a logical hierarchy to group related data together (e.g., source image data and associated annotations, 2-photon and EM datasets from the same tissue sample). This service provides interfaces to create and manage datasets and their properties.

2.3.3 Ingest

A critical challenge when using a centralized data archive is the ingest of large datasets, as users will always locally organize and store data they generate in unique ways. The Ingest Service facilitates moving large datasets from local storage into bossDB by decoupling the upload of data to the cloud and ingesting of data into the spatial database, allowing independent scaling and failure recovery. The service provides methods to create a new ingest job, monitor the status of a job, join an upload client worker to a job, and cancel a job.

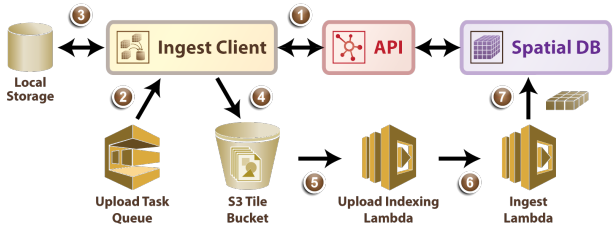

Figure 5: A diagram outlining the ingest process.

As demonstrated in Figure 5, the ingest process directly leverages AWS infrastructure, massively scaling on demand. First, using the ingest client a user uploads an ingest job configuration file to the API (1) which populates a task queue, enumerating all tiles that must be uploaded, and returns temporary AWS credentials. Next, the ingest client retrieves a task from the Upload Task Queue (2), and loads the requested local file into memory as an image tile (3). The tile data is handed back to the client which uploads it to an S3 bucket (4). Lambda and DynamoDB are used to track which tiles are successfully written to the tile bucket (5) and when enough tiles in a region have arrived to generate the bossDB cuboid data representation, a second Lambda function is triggered (6). This Ingest Lambda function then loads the specified tiles, reformats them into cuboids, inserts them into the Spatial DB S3 bucket, updates the Spatial DB cuboid index, and finally removes the temporary tiles (7). The ingest client supports both parallel and distributed operation, allowing users to maximize their network bandwidth.

2.3.4 Dataset Metadata

bossDB can store arbitrary key-value pairs linked to data model items, which is useful to track experimental metadata and provenance (e.g., voxel size, animal information, annotation algorithm used). This service provides an interface to query, create, update, and delete key-value pairs associated with a dataset.

2.3.5 Cutout

A service to interact with the Spatial Database by reading and writing arbitrary data volumes. While bossDB stores all data internally using a standardized format, the cutout service uses HTTP content negotiation to determine the the data format of a request, allowing bossDB to easily support user-specified formats when uploading or downloading data (e.g., compressed C-order blob, hdf5 file, pickled numpy array). This service enables scalable analytics by letting users access arbitrary chunks of data in parallel, perform automated processing, and write the annotation result back to bossDB. It also supports querying for the spatial properties of annotations, such as the bounding box of an annotation or identifying which annotations exist within a region.

2.3.6 Image

In addition to our volumetric cutout service, we provide an image service to meet common user needs, which retrieves a 2D slice of data from the spatial database, along one of the three orthogonal planes (i.e., XY, XZ, YZ), encoded as an image file. Again, HTTP content negotiation is used to determine the format of the response (e.g., png, jpeg). The service supports arbitrary image sizes or a fixed tile size, which is often used by visualization tools.

2.3.7 Downsample

To allow users to quickly assess, process, and interact with their data, we need to iteratively build a resolution hierarchy for each dataset by downsampling the source data. This is a workflow that is run infrequently and on-demand, and needs to scale from gigabytes to petabytes of data. We developed a serverless architecture built on AWS Step Functions to manage failures and track process state. AWS Lambda is used to perform the underlying image processing in an embarrassingly parallel, scalable fashion. This approach allows us to minimize resource costs while scaling on-demand in a fully-automated paradigm.

2.4 User Tools

To make a data archive truly useful, easy to use and well-documented user-facing tools are required. We currently offer a web-based management console, an ingest client, and a client-side Python module for programmatic interaction. We have also integrated 3rd-party web-based data visualization tools. While `bossDB` API provides a rich interface to interact with the system, user friendly tools built on top of the API are critical to increase utility and adoption by the community. We expect this tool library to grow as users build on the core `bossDB` technologies.

2.4.1 Web-based Management Console

`bossDB` has a web interface that lets users perform common actions directly in their browser (e.g., create a dataset, monitor an ingest job, share a dataset with a user). This console is the primary interface for most users and will exposes much of the API's functionality through an intuitive graphical interface. From the console, a researcher is able to manage datasets, discover new data, and launch the visualization tool.

2.4.2 Web-based Visualization

A critical capability to any data archive is the ability to easily visualize stored data. Whether inspecting ingested data, exploring a dataset, or sharing an interesting sample with a collaborator, often the most common interaction with stored data will be through visualization. We integrated a version of Neuroglancer [16] to let users visually explore data stored in `bossDB`.

2.4.3 Ingest Client

We have developed an open source ingest client in Python to manage uploading data to `bossDB`. The ingest process operates on a upload task queue which contains tasks specifying individual 2D tiles of data to upload. To deal with the unique formats and file organization methods of diverse users, the client uses a simple plug-in design to import custom snippets of code responsible for taking a task, finding the right file, and loading the data into memory, which is then uploaded by the client. The work queue design allows many copies of the client to be run distributed across compute nodes and in parallel on a single machine, drastically increasing throughput.

2.4.4 Python Software Development Kit (SDK)

To support developers and researchers who want to programmatically interact with `bossDB`, we provide a pip-installable Python library that abstracts much of the API's complexity away from the user. Data cutouts of arbitrary size can be efficiently retrieved from our archive, enabling easy integration with analytics tools. The current SDK, called `intern`, will continue to be expanded and supported to accommodate updates and additions to the existing `bossDB` system and user requests.

3 Results

3.1 Motivating Application

Many of our design requirements were motivated by the Machine Intelligent from Cortical Networks (MICrONS) Program, funded by the Intelligent Advanced Research Projects Activity (IARPA) [7]. This program seeks to rapidly advance machine learning capabilities by creating novel machine learning algorithms that use neurally-inspired architectures and mathematical abstractions of the representations, transformations, and learning rules employed by the brain. To guide the construction of these algorithms, the program centers around massive co-registered functional (e.g., two-photon calcium imaging) and structural (e.g., EM and XRM) neuroimaging experiments aimed at estimating the synapse-resolution connectome of a $1\,mm^3$ volume of cortex and using that information to constrain machine learning architectures. Our goal was to organize, store, and support the analysis of these large functional and anatomical datasets while enabling public dissemination.

3.2 Deployment

We envision that this data archive will facilitate neuroscience inquires through an extensible, scalable process, with a sample process outlined in Figure 6. Below we describe our results to date for the MICrONS datasets. Currently we serve approximately 50 regular users and 85 TB of compressed image data. date for MICrONS datasets.

3.2.1 Implementation

Figure 7 shows the architecture of bossDB. The system has two user facing services: Authentication and Web Server Endpoint, both of which sit behind AWS elastic load balancers. The system uses Keycloak servers in a high-availability configuration for single sign-on authentication. The web server endpoints use Django API, to provide access to the majority of the services in bossDB.

bossDB uses serverless computing and storage, with AWS Lambda, SQS, S3 and DynamoDB to provide all of the other services mentioned in Section 2: Ingest, Metadata, Cutout, Image and Downsample. Using serverless computing and storage for these

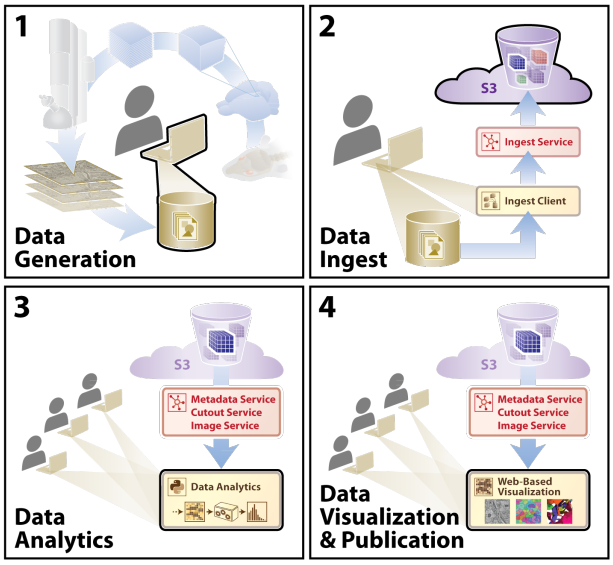

Figure 6: A diagram outlining an example user story showing utilization of the bossDB infrastructure. A typical research group collecting data for a hypothesis will move sequentially from (1)-(4). Other groups will extend these analyses using steps (3) and (4).

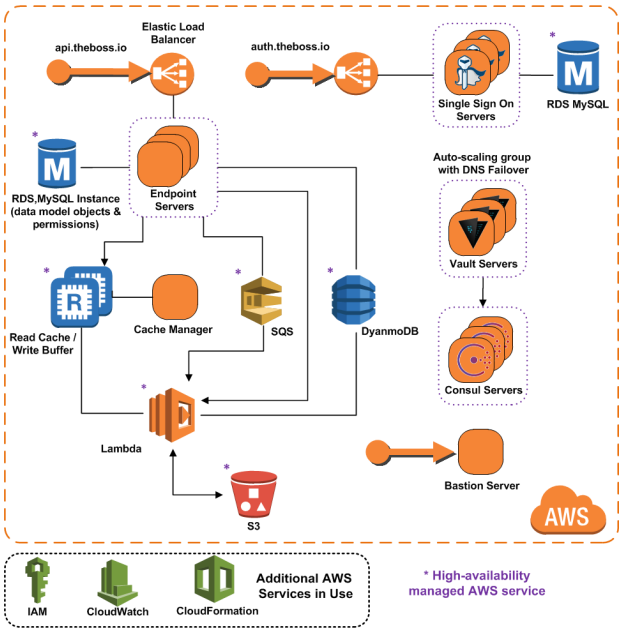

Figure 7: A high-level architecture diagram of bossDB as deployed using Amazon Web Services architecture.

components will automatically scale with demand and eliminate the need to maintain components.

BossDB is installed using the AWS CloudFormation service along with Salt and Packer to manage our infrastructure. This allows us to quickly duplicate the environment for testing and development and even change instance sizes within the new environments.

3.2.2 Data Generation

Researchers collect experimental data; stitching, alignment, and registration take part prior to upload to bossDB. Users create new resources in bossDB to identify and store their datasets, recording their experimental parameters and dataset properties (e.g., voxel dimensions, bit depth, spatial extent) prior to upload. An example screenshot from our web console is shown in Figure 8; this setup can be accomplished programmatically using `intern` as well.

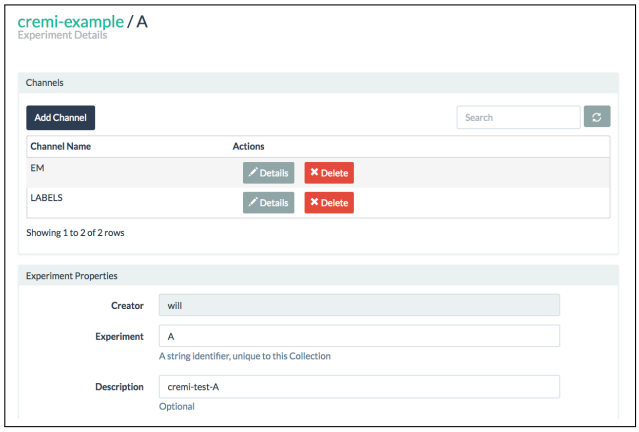

Figure 8: An example screenshot from our bossDB console.

3.2.3 Data Ingest

Once available, a researcher uploads image data via one of several methods supported by bossDB (e.g., REST API, ingest client), safely and efficiently storing data in bossDB. Large datasets can be uploaded incrementally, with data available for read as soon as it has been ingested, providing access to collaborators in minutes, not months.

The ingest client has already been used to upload many terabytes of EM and calcium imaging data, and will be used to ingest petabytes of data in the near future; many of these uploads proceed without any intervention from the developer team with the system automatically scaling to meet user's needs.

Initial testing of the ingest process reached a maximum sustained ingest throughput of 2.3 Gbps, bursting to 5 Gbps over an internet2 link from a remote user site into bossDB. The ingest client was run on two large, local servers with a total of 192 parallel threads that uploaded data at a rate of 1250 files per second. AWS Lambda scaled to roughly 1500 concurrent executing functions to handle the load.

To perform at this speed we were using approximately 47.5K lambda compute seconds per minute, 3 Endpoint servers sized with m4.2xlarge instances, a RDS database backed with a db.m4.xlarge instance,

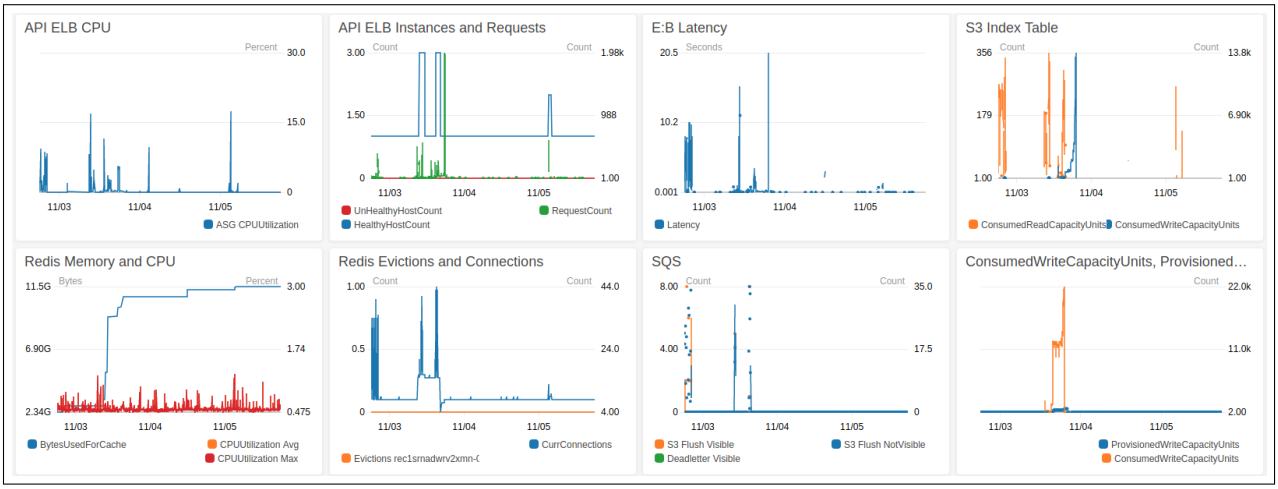

Figure 9: A CloudWatch dashboard monitoring during ingestion.

and DynamoDB table sized at 300 read / 600 write capacity. We expect to achieve sustained rates of 30 GB per minute during production.

BossDB has monitoring capability at several levels. In Figure 9 you see a snippet of our Ingest Dashboard which allows the administrator to see how much stress any one component of the system is under. Notifications will also go out if any key components fail, and when the system hits cost milestones.

3.2.4 Data Analytics

Many big data research analyses are enabled by bossDB features (e.g., standardized interfaces, arbitrary cutouts, spatial indexing), accelerating the scientific process.

```
# import intern package
from intern.remote.boss import BossRemote
from intern.resource.boss.resource import *
import numpy

# initialize BossRemote
rmt = BossRemote()

# specify data location
COLL_NAME = 'test_collection'
EXP_NAME = 'test_experiment'
CHAN_NAME = 'test_channel'

chan = ChannelResource(CHAN_NAME, COLL_NAME,
    EXP_NAME, 'image', datatype='uint16')

# specify download coordinates and resolution
x_rng = [0, 1024]
y_rng = [0, 512]
z_rng = [0, 10]
res = 0

# Download the cutout from the channel
data = rmt.get_cutout(chan, res, x_rng, y_rng,
    z_rng)
```

One common use for bossDB is acting as a backend for local data analysis pipelines. Users download chunks of data from bossDB using intern and process it to create annotation labels using humans or machines. The resulting annotation data is uploaded via a choice of methods (python API, ingest client).

3.2.5 Data Visualization and Publication

Data can be quickly visualized using applications such as NeuroGlancer (Figure 10).

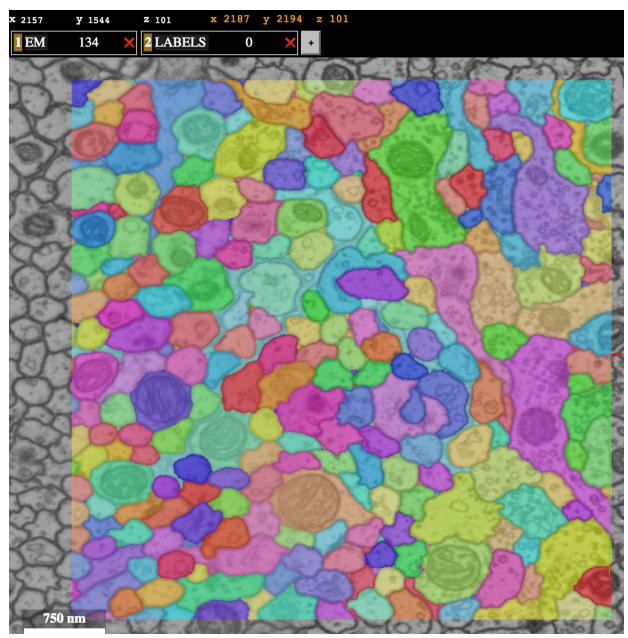

Figure 10: An example neuroglancer web visualization backended by bossDB, showing data from the recent CREMI challenge (www.cremi.org).

Data are published along with initial analysis, and made widely accessible widely through bossDB. Other research teams conduct additional analysis, extending and validating the existing scientific findings. By

providing a standardized data standard format, it is easy for researchers to apply new techniques and test additional hypotheses.

4 Discussion

Our data archive will enable scientists to easily access and process large datasets, and to scale up their approaches with minimal alterations and without needing large local storage. Because the results are anchored to a universally-accessible datastore, it is easier for others to inspect the results, improve upon them, and reproduce processing pipelines by leveraging common interfaces.

When considering a cloud-native approach, vendor lock-in is one potential concern – as we not only use the AWS cloud to deploy `bossDB`, but have integrated many of its services into the system to substantially accelerate development and performance. To minimize the development impact of expanding to an additional cloud provider or on-premise cluster, future work is needed to create a layer of abstraction between the core software and AWS services. We plan to continue to develop towards a microservices style architecture, which will decrease coupling between sub-components. This will allow `bossDB` to be able to independently scale sub-components and increase the ability to easily deploy, update, and manage services. We believe that storage engines will continue to specialize around datatypes (e.g., multidimensional image data, video data, gene sequence data) and be applicable to multiple research communities through the creation of domain-specific APIs that maintain the unique formats, organization, and needs of that community.

We expect that as the community uses our data archive, additional tools will be developed to address new researcher needs, such as a universal, robust object-level metadata system and additional visualization engines. Several new `bossDB` deployments are under development, including one by NeuroData, which serves approximately diverse collaborators in several new modalities (e.g., light microscopy, array tomography, Clarity) and adds several new tools and capabilities to the `bossDB` ecosystem.

One concern about running a cloud data archive is estimating and managing cost. `bossDB` architecture was designed to allow dynamic scaling of resources to balance cost with performance and throughput capacity. As our software stack continues to mature, we plan to further optimize our tiered storage architecture (e.g., automatic migration data between S3 Standard, Infrequent Access, and Glacier tiers). The proposed system will provide a framework that is able to trivially scale from terabytes to petabytes while maintaining a balance between cost efficiency and performance.

The standardization and scalability provided by our data archive will support a fundamental change in how researchers design and execute their experiments, and will rapidly accelerate the processing and reuse of high-quality neuroscience, most immediately for the large, petascale image and annotation volumes produced by IARPA MICrONS. No previously existing platform met the operational and scaling requirements of the program, including managing an estimated 6-10 petabytes of image and annotation data – much larger than public neuroanatomical data archives. The `bossDB` software and documentation is open source and we are eager to expand the user community, supported modalities, and features. More information, examples and support are available at bossdb.org and https://github.com/jhuapl-boss/.

References

[1] D. D. Bock, W.-C. A. Lee et al., "Network anatomy and in vivo physiology of visual cortical neurons," Nature, vol. 471, no. 7337, pp. 177–182, 2011.

[2] N. Kasthuri, K. J. Hayworth et al., "Saturated Reconstruction of a Volume of Neocortex," Cell, vol. 162, no. 3, pp. 648–661.

[3] M. Helmstaedter, "Cellular-resolution connectomics: challenges of dense neural circuit reconstruction." Nature methods, vol. 10, no. 6, pp. 501–507, jun 2013.

[4] W.-C. A. Lee, V. Bonin et al., "Anatomy and function of an excitatory network in the visual cortex," Nature, vol. 532, no. 7599, pp. 370–374, 2016.

[5] C. Dupre and R. Yuste, "Non-overlapping neural networks in hydra vulgaris," Current Biology, vol. 27, no. 8, pp. 1085 – 1097, 2017.

[6] J. W. Lichtman, H. Pfister, and N. Shavit, "The big data challenges of connectomics," Nature Neuroscience, vol. 17, no. 11, 2014.

[7] Microns: Machine intelligence from cortical networks. http://iarpa.gov/index.php/research-programs/microns. retrieved 2017-10-31.

[8] S. Mikula, "Progress Towards Mammalian Whole-Brain Cellular Connectomics," Frontiers in Neuroanatomy, vol. 10, p. 62, jun 2016.

[9] S. Takemura, C. S. Xu et al., "Synaptic circuits and their variations within different columns in the visual system of Drosophila." Proceedings of the National Academy of Sciences of the United

States of America, vol. 112, no. 44, pp. 13 711–13 716, nov 2015.

[10] E. L. Dyer, W. Gray Roncal et al., "Quantifying mesoscale neuroanatomy using X-ray microtomography," eNeuro 2017.

[11] Distributed, versioned, image-oriented dataservice. https://github.com/janelia-flyem/dvid. retrieved 2017-10-31.

[12] S. Saalfeld, A. Cardona, V. Hartenstein, and P. Tomančák, "CATMAID: collaborative annotation toolkit for massive amounts of image data," Bioinformatics, vol. 25, no. 15, pp. 1984–1986, 2009.

[13] R. Burns, W. Gray Roncal et al., "The Open Connectome Project Data Cluster: Scalable Analysis and Vision for High-Throughput Neuroscience," Proceedings of the 25th International Conference on Scientific and Statistical Database Management (SSDBM), jun 2013.

[14] W. Gray Roncal, D. M. Kleissas et al., "An Automated Images-to-Graphs Framework for High Resolution Connectomics," Frontiers in neuroinformatics (in press), pp. 1–13.

[15] J. T. Vogelstein, B. Mensh et al., "To the Cloud! A Grassroots Proposal to Accelerate Brain Science Discovery," Neuron, vol. 92, no. 3, pp. 622–627, 2016.

[16] e. a. Maitin-Shepard, Jeremy, "Neuroglancer. https://github.com/google/neuroglancer. retrieved 2017-06-10."

5 Acknowledgements

This material is based upon work supported by the Office of the Director of National Intelligence (ODNI), Intelligent Advanced Research Projects Activity (IARPA), via IARPA Contract No. 2017-17032700004-005 under the MICrONS program. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation therein.