ApertureData

Redefining Visual Data Management

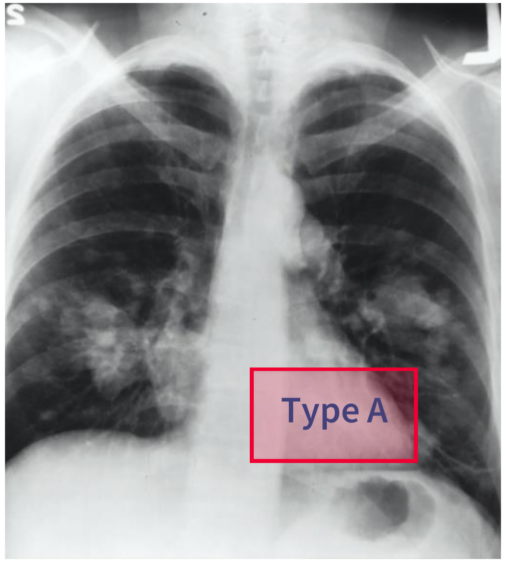
vishakha@aperturedata.io

www.aperturedata.io

Explosion in Visual Data Fueled by Data Science

Smart agriculture using drone cameras to determine right soil treatments

People identification using city cameras to determine social distancing



Machine learning on patient scans to help experts diagnose problems

What’s Special About Visual Data Now?

Large and voluminous

Individual image and video can be GB size and start at millions even for small deployments

Unique transformations

Basic operations like crop / zoom are common and can be computationally expensive

Extensive metadata

Not just about visual data itself but application context, annotations, features

Today's Visual Applications

Object stores or file systems for image / video data treated as regular blobs

CSV files or some database for metadata, not visual friendly

Stitch libraries for pre-processing before use

Manual integrations result in inefficiencies

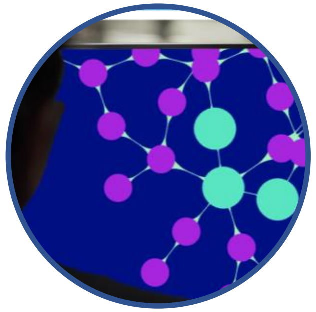
ApertureDB:
A Database for Visual Data

ApertureDB: Key Features

Visual Data

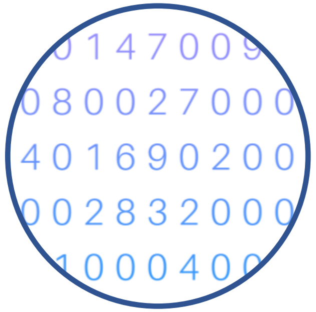
Native support for images, videos, bounding boxes, and pre-processing operations

Metadata



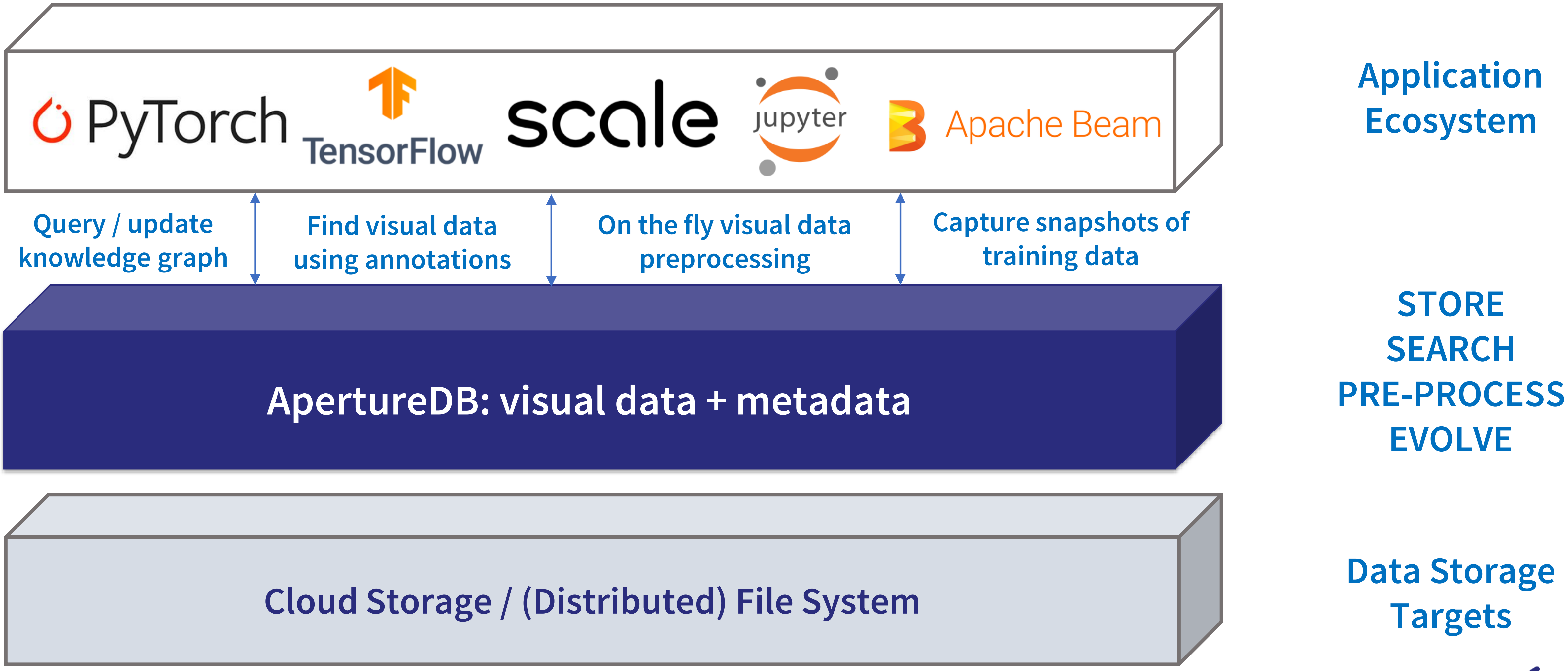
Metadata information as a knowledge graph for more advanced visual search

Similarity Search



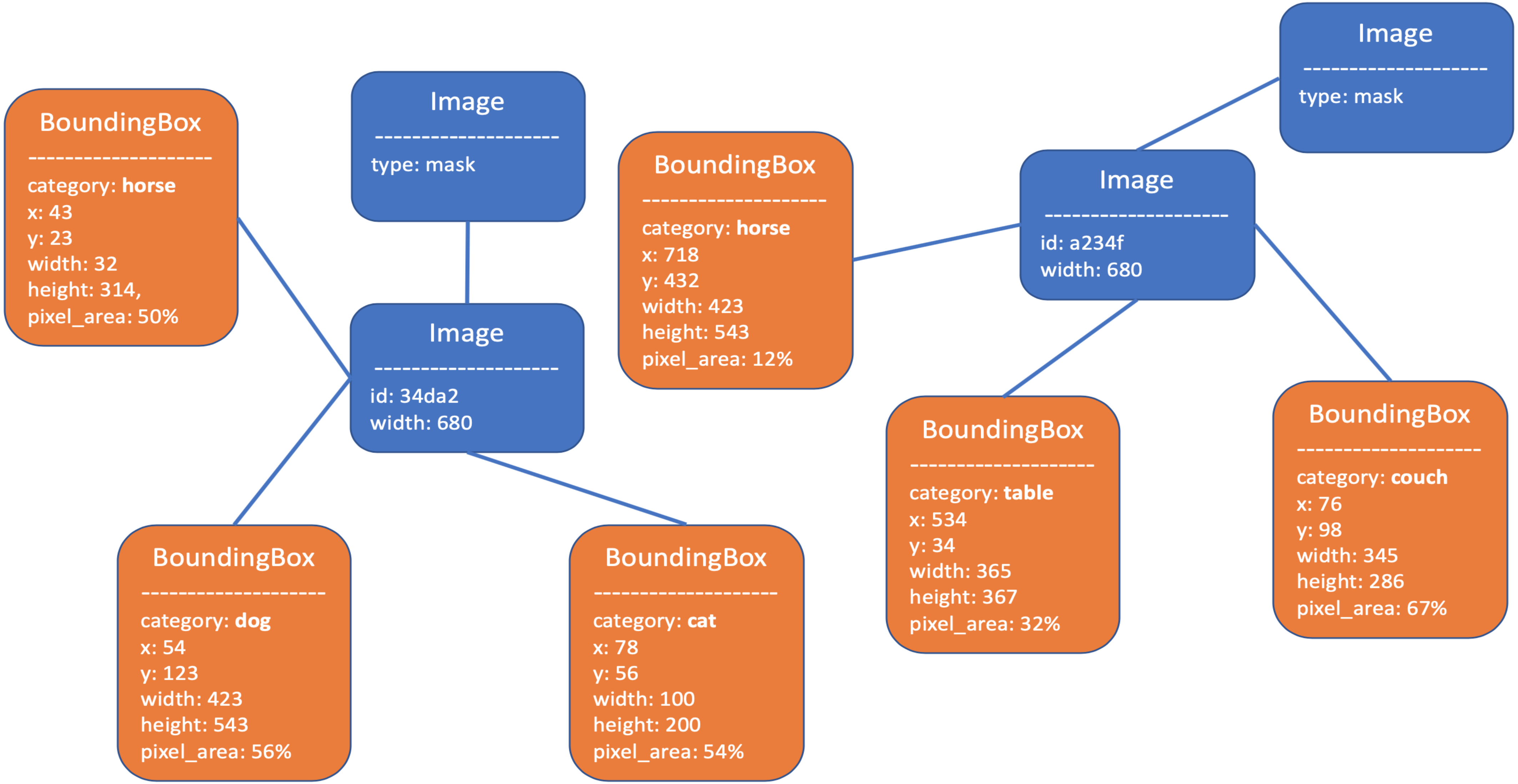
Built-in similarity matching for high-dimensional feature vectors

ApertureDB Ecosystem

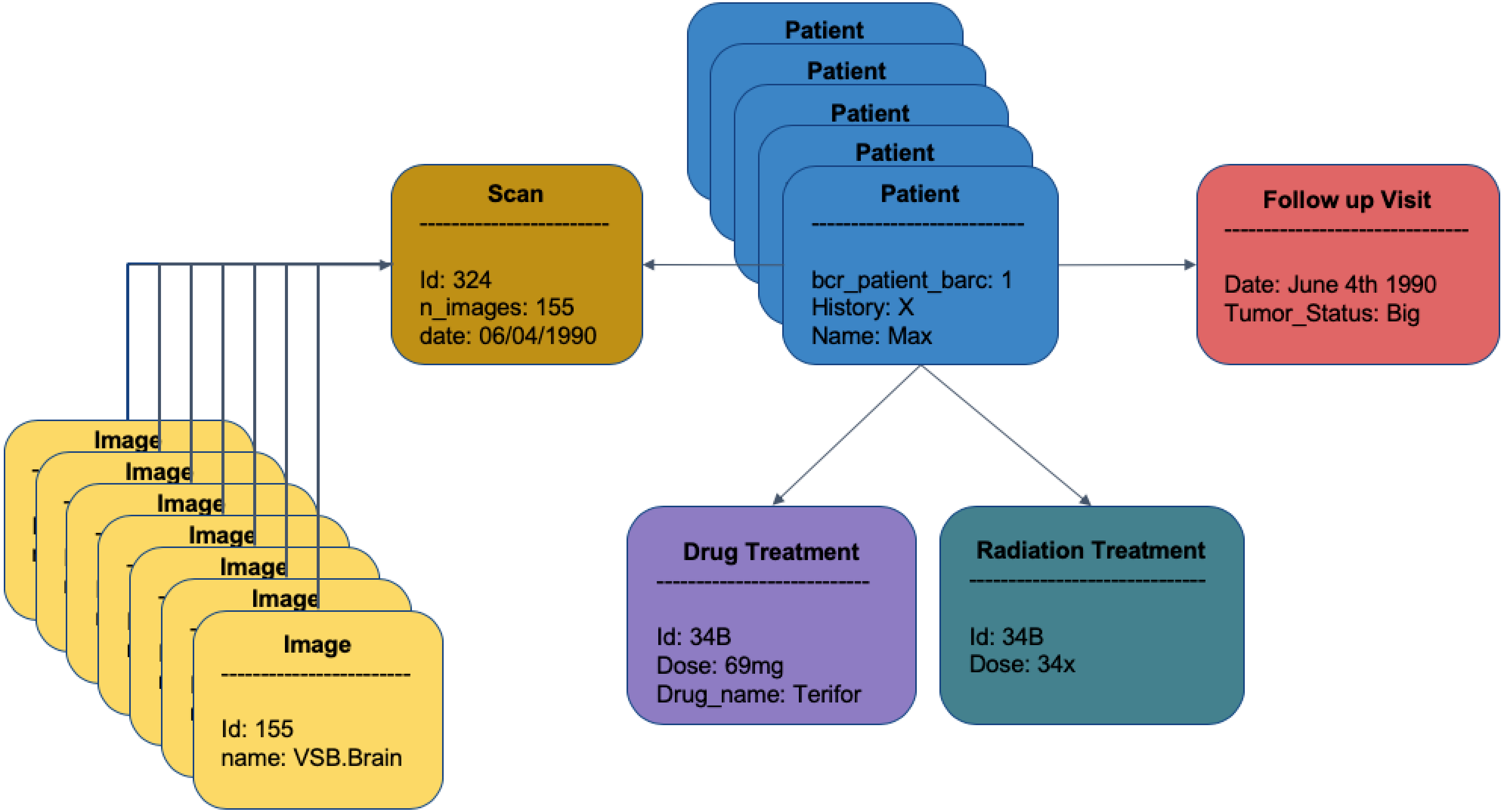


ApertureDB (VDMS) Architecture

ApertureDB (VDMS) Architecture

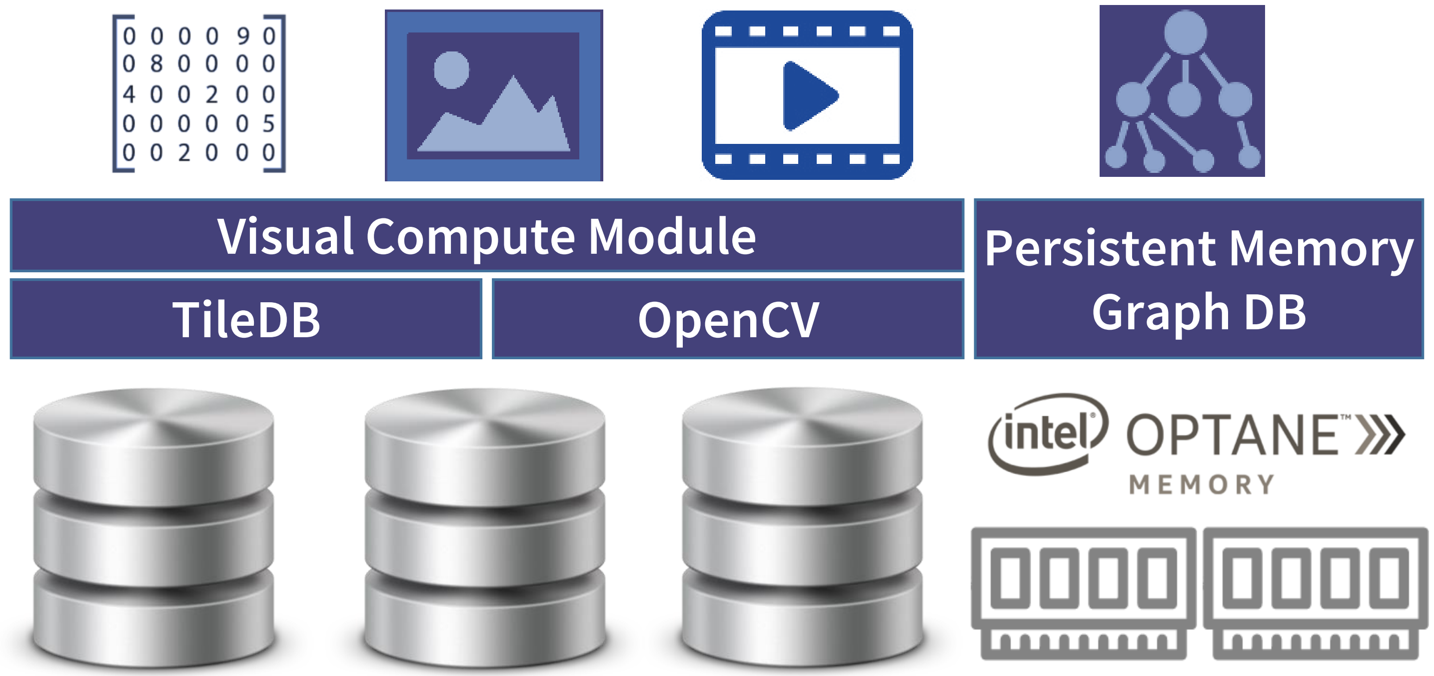


ApertureDB (VDMS) Architecture

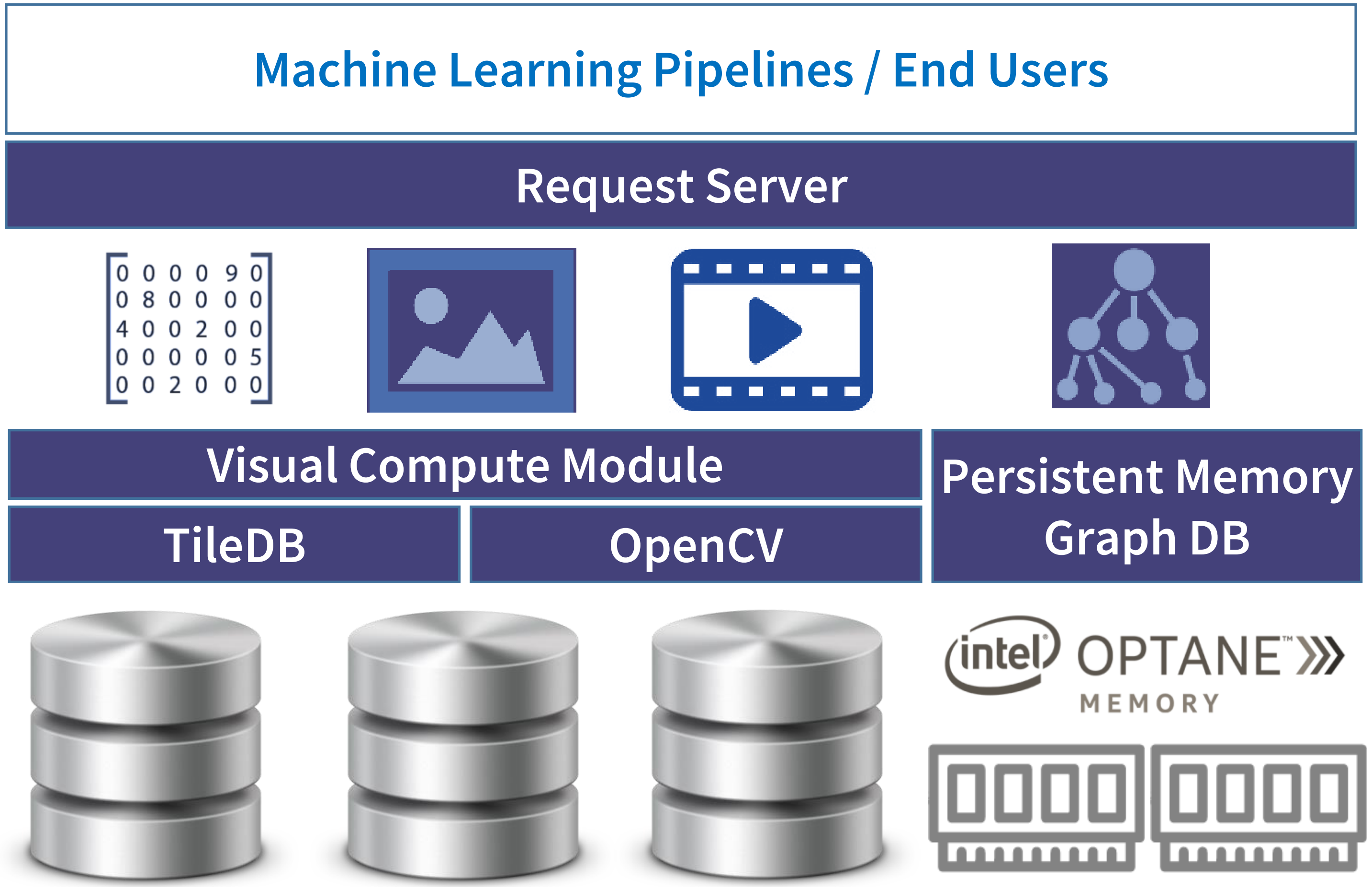


ApertureDB (VDMS) Architecture

ApertureDB (VDMS) Architecture



ApertureDB (VDMS) Architecture



Persistent Memory Graph Database

Property graph implementation targeting persistent memory (Intel)

- Traditional property graph databases plagued by disk latencies
- NVM technology (e.g. Optane) with performance close to DRAM

ACID with transactional semantics

Easy to evolve schema

Graph API, neighbor traversals, management tools

Salient Design Features

PM-aware data layout and access

- Memory-mapped large files from ext4-DAX
- Fix size node and edge objects align with cache size
- clflushopt/clwb or msync for persistence
- Custom allocators
- Lazy iteration

2 level indexes for class and properties, per node edge indexes

Undo logs

Stripe locks and variable concurrency mechanisms

Visual Compute Module

Manage actual storage and access of visual objects

- Data on file system or object store
- Implement different formats, pre-processing using OpenCV, ffmpeg when possible
- Perform common operations closer to the data

Feature vector indexing and similarity search (a la feature db)

- Various indexing techniques (trade-off accuracy and speed)
- Persistent FAISS indexes, TileDB sparse and dense indexes
- K nearest neighbors on n-dimensional feature vectors using different distance metrics
- Graph node which allows filtering with properties after KNN

Request Server and Visual-first API

jupyter 00-BrainSegmentation-UseCase Last Checkpoint: 01/23/2020 (autosaved)

Logout

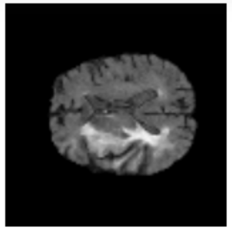
File Edit View Insert Cell Kernel Widgets Help

Trusted

Python 3

Run

Markdown



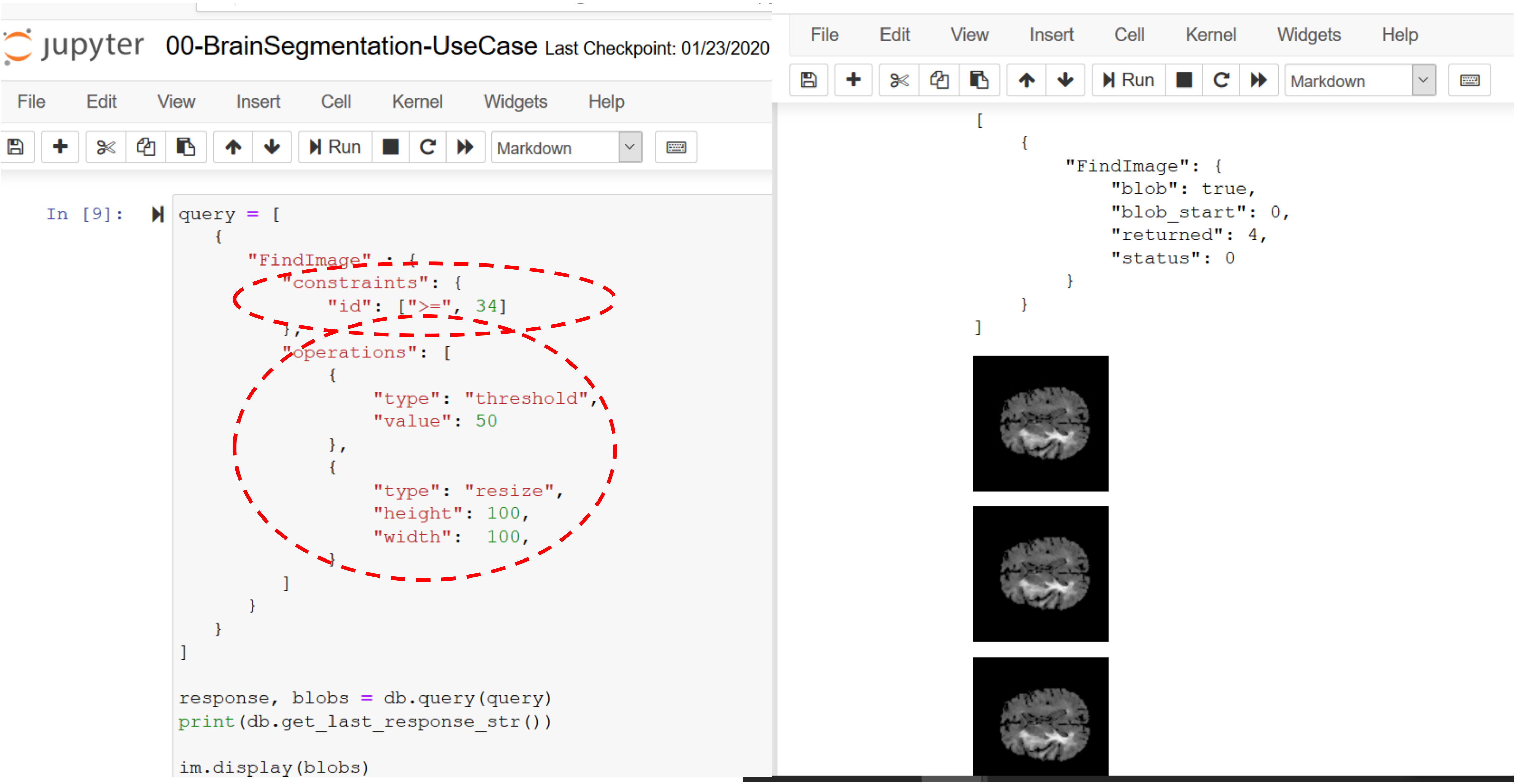
In [7]:

```
query = [
    {
       "AddImage" : {
          "properties": {
              "name": "my_favourite_image",
              "id": 39
          },
          "format":"jpg"
       }
    }
]

blobs = []
blobs.append(img_blob)

response, blobs = db.query(query, [blobs])
print(db.get_last_response_str())
```

Request Server and Visual-first API



High Performance: Comparing with Status Quo

vs.

APACHE HTTP SERVER

MySQL

Open CV

- YFCC100M (~100 million images) dataset
- Metadata-based visual search queries to find right set of images
- Up to 35x faster and 15x on average

Resource Efficient: Preprocessing Near Data

63% reduction

in data transferred over the network
using pre-processing within our API

Efficient and Scalable: Biotech AI Customer Use Case

Save months

Customer saved months of data platform engineering

time

1B metadata entities

Already loaded over 1B metadata entities and terabytes of data

Simpler ML pipeline

Fewer moving parts and simplification of a complex ML pipeline

Competitive Landscape

ML Aware

Searchable data

Complex data

Basic put / get

DIY Systems

ApertureDB

ApertureDB is unique in focusing on efficient data access for ML

Near-term Features

More integrations e.g. PyTorch data loader, cloud storage connectors

Increased reliability (data + uptime), debuggability

Advanced UI frontend beyond dataset exploration

User-defined functions

Object model for simpler pipeline interaction

More complex regions of interest e.g. polygons

Want to Learn More?

Homepage: aperturedata.io

API docs and papers: docs.aperturedata.io

Try out a demo: aperturedata.io/demo-request

Engineering blog: medium.com/aperturedata

Follow us on LinkedIn (company/aperturedata) and twitter (@aperturedata)

Intel github: github.com/IntelLabs/vdms or pmgd

ApertureData