A Data Platform for Machine Learning and Data Science Applications

Introduction and Overview

Craig McChesney
Christopher K. Allen
Osprey DCS

Outline

- Motivation
- Approach
- Status
- Development Plan
- Technology Drill-Down
- Installation and Deployment

Motivation

Our goal is to provide full stack support for machine learning and general data science applications. We will provide tools to:

- capture time-correlated machine data from arbitrary data sources to a central repository
- support ingestion of heterogeneous data types: scalar, array, table, structure, and image
- associate machine data with important events via "snapshot" metadata
- support post-ingestion annotation including comments, associations, and calculations
- upload derived datasets and link datasets to show provenance and relationships
- provide broad data-science oriented retrieval of heterogeneous data and annotations
- minimize time required to make data available in the repository: our performance goal is to continuously capture data for 4000 PVs each sampled at 1KHz
- provide a web application for exploring the data platform repository

Approach

- create Java server applications providing data ingestion, query, and other services such as annotation and export
- utilize a central database platform for storing time series data and metadata, and indexing data files created outside the database
- provide a cross-platform, high-performance API that supports a wide variety of programming languages and types of applications
- provide high-level libraries for building data science applications
- build a JavaScript web application for retrieving, viewing, exporting, annotating, and manipulating data in the repository

Why use an ingestion service API instead of writing directly to the database?

This is a common question, and is understandable when the focus is on ingestion performance. Using a service API facilitates:

- Changing the underlying persistence technology or database schema transparently to the clients
- Capturing data from a wide variety of devices without creating a custom capture client for each of them that exposes the details of the underlying persistence mechanism and must be kept in sync with database/schema changes
- Including time-correlated data for those devices in a single result set

The performance measured for the initial ingestion service implementation is in the same range as our benchmark for writing data directly to MongoDB, so using the service API doesn't seem to degrade performance significantly.

Status

- investigated gRPC authentication/authorization approach (9/2023)
- re-implemented ingestion service utilizing MongoDB to store time series data and metadata, exceeding baseline performance goal for scalar data (9/2023)
- simplified API specification, eliminated InfluxDB from tech stack (8/2023)
- completed performance benchmark study of technology stack alternatives (8/2023)
- built prototype web application demonstrating use of query service API from JavaScript app running in the browser (1/2022)
- built prototype ingestion and query services demonstrating use of gRPC API for capturing heterogeneous data types from arbitrary data sources (12/2022)

Road Map / Development Plan

- documentation and build process for new data platform implementation (10/2023)
- build query service that reflects API changes and new repository structure, focusing initially on scalar data (10/2023)
- add handling to ingestion and query services for arrays, tables, structures, and images (12/2023)
- add authentication/authorization mechanism
- add support for post-ingestion annotation of data
- add support for exporting data
- add support for uploading and linking datasets, data provenance
- revamp web application to reflect API changes and add new features
- deployment, prodictionization, configuration, and scaling

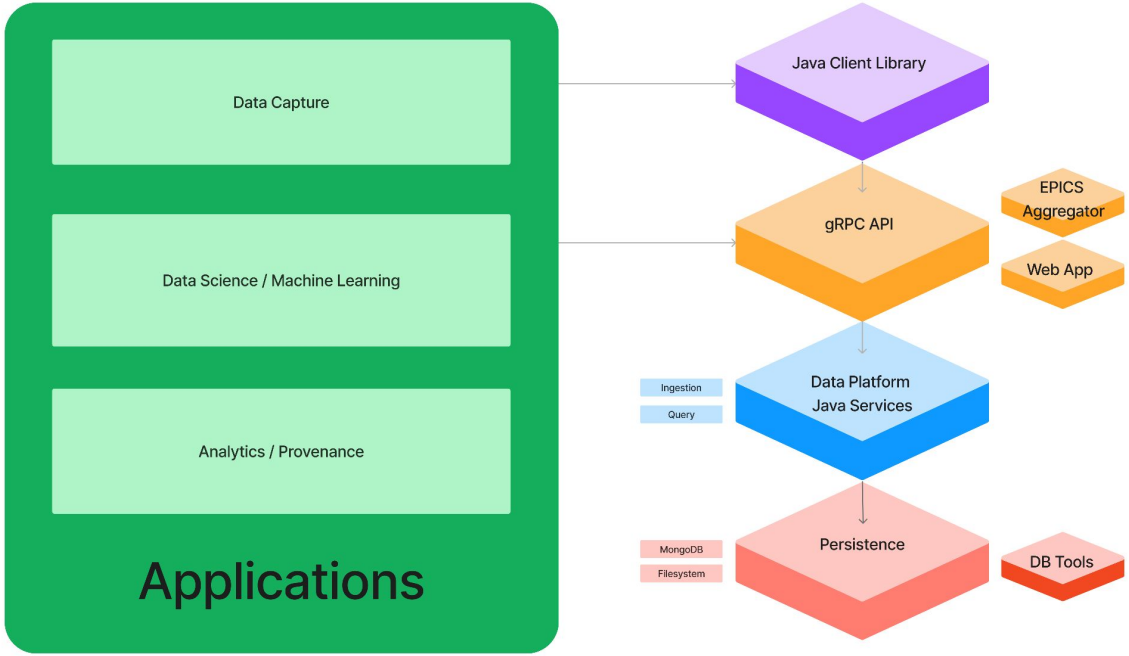
Technology Drill Down

Data Platform Technology Drill Down

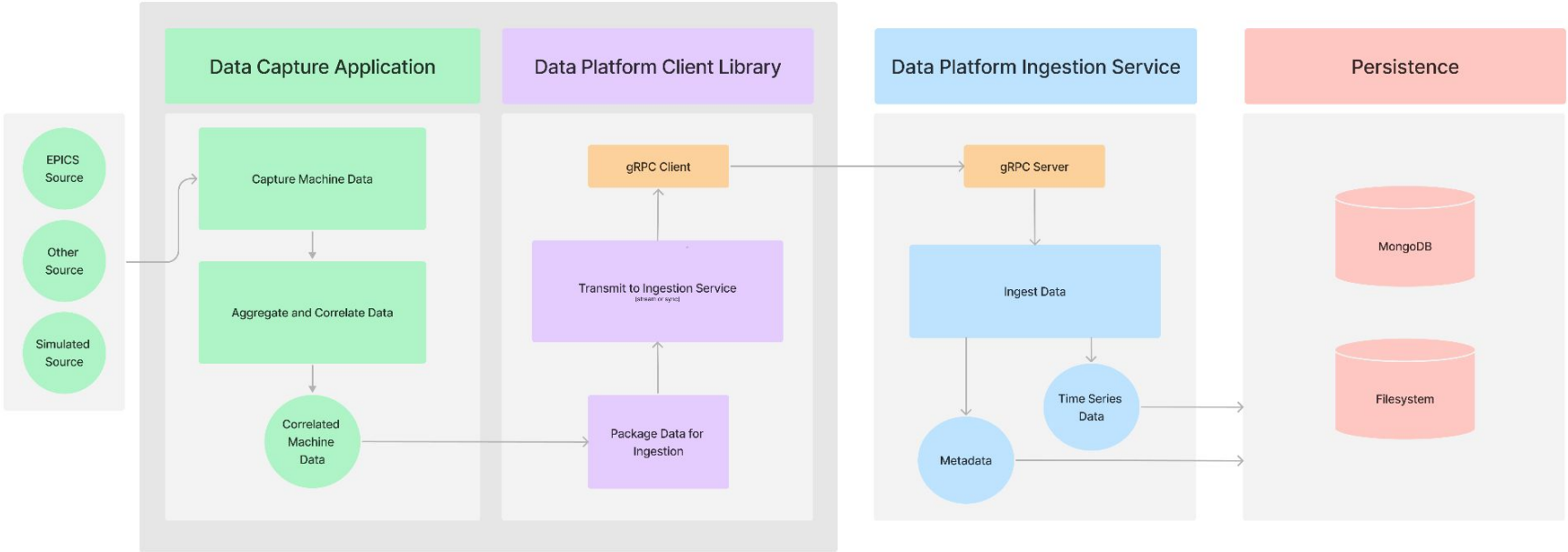
- Architecture Diagrams
  - Technology Stack
  - Data Flow
- Technology Stack Elements
  - Communication Layer
  - Service Layer
  - Persistence Layer
  - Application Layer
  - Toolbox

Data Platform Technology Stack

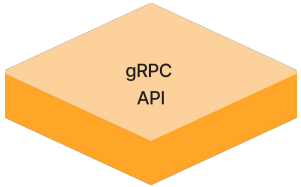
The Data Platform provides full stack support for building machine learning and data science applications.



Data Platform Ingestion Data Flow

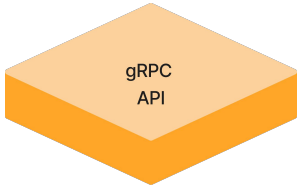


Communication Layer: gRPC



- gRPC is a high-performance, cross-platform communication framework.
- gRPC support is provided for pretty much every programming language.
- "proto" files are used to specify the API for a service along with supporting data structures.
- The protoc compiler generates programming language specific code for using the API and data structures.
- gRPC supports bidirectional streaming of data between client and server.

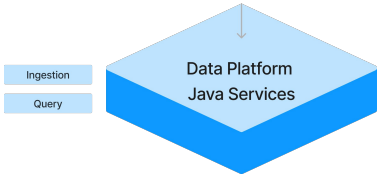
Communication Layer: gRPC Service Definition "proto" files




```
service DpIngestionService {
  rpc registerProvider (RegisterProviderRequest) returns (RegisterProviderResponse);
  rpc streamingIngestion (stream IngestionRequest) returns (stream IngestionResponse);
  rpc unaryIngestion (IngestionRequest) returns (IngestionResponse);
}
```

```
service DpQueryServiceService {
  rpc queryDataByTime (QueryDataByTimeRequest) returns (stream QueryDataResponse);
}
```

Service Layer: Ingestion and Query Services



- The Data Platform provides Java server implementations of the Ingestion and Query service APIs as defined in the gRPC "proto" files.
- The Ingestion Service uses MongoDB and the filesystem to build a repository of machine data, metadata, and annotations.
- The Query Service provides broad data-science oriented access to that repository.
- Client applications communicate with the services via the gRPC communication framework.

Service Layer: Authentication/Authorization

Assumption is that ingestion clients run behind firewall and that authentication is not required (e.g., similar to EPICS infrastructure components). Authentication is required for query clients. Initial thoughts about how we will handle authentication and authorization in the gRPC service:

- Enable server authentication and secure transport via built in gRPC support for SSL/TLS.
- Create a new "login" gRPC API method in the ingestion (and query) services. The login method will use the infrastructure LDAP service to authenticate the user credentials. It will return a JSON web token (JWT) to the caller.
- In subsequent gRPC API calls, the client will attach the JWT token as metadata to the request header.
- Investigate use of API token authentication for infrastructure applications (not tied to a specific user).

MongoDB

Persistence Layer: MongoDB

- MongoDB is a document-oriented "NoSQL" database platform.
- We are using the "Community Edition" which can be used without a licensing fee.
- JSON-like documents are used to store data.

```
{
    _id: 'pv_1001-1691609823-0',
    columnName: 'pv_1001',
    firstDate: ISODate('2023-08-09T19:37:03.000Z'),
    firstNanos: 0,
    firstSeconds: 1691609823,
    lastDate: ISODate('2023-08-09T19:37:03.999Z'),
    lastNanos: 999000000,
    lastSeconds: 1691609823,
```

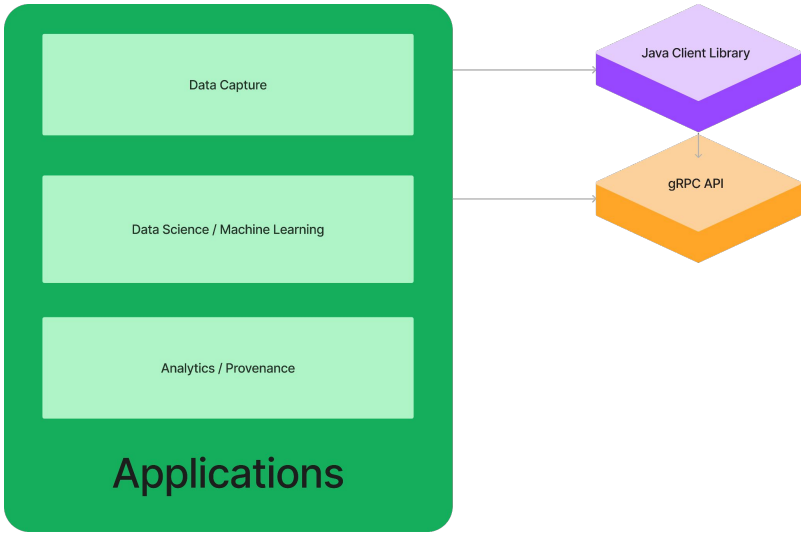
Filesystem

Persistence Layer: Filesystem

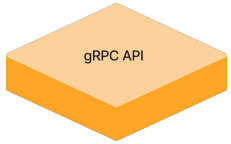
- Data files are used to store image data and possibly other complex data types.
- Elements stored in MongoDB reference those data files via indexes.
- We are evaluating the use of HDF5 files for longer term storage of infrequently accessed data.

Application Layer: Low-Level API vs. High-Level Library

Applications can be built at two levels, either using the lower-level gRPC API directly or using the higher-level Java client library.



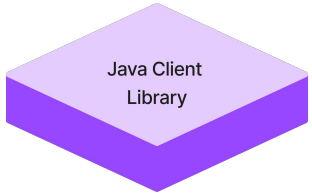
Application Layer: gRPC API Clients



- The low-level gRPC Ingestion and Query Service APIs can be used from a wide variety of programming languages to build applications.
- The "protoc" compiler generates appropriate code for using the data structures and invoking procedures defined in the service "proto" API files.

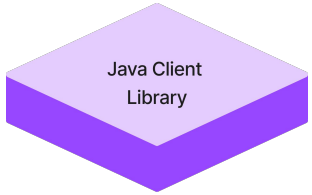
```
IngestionRequest.Builder requestBuilder = IngestionRequest.newBuilder();
Timestamp.Builder snapshotTimestampBuilder = Timestamp.newBuilder();
snapshotTimestampBuilder.setEpochSeconds(params.snapshotTimestampSeconds);
snapshotTimestampBuilder.setNanoseconds(params.snapshotTimestampNanos);
snapshotTimestampBuilder.build();
requestBuilder.setSnapshotTimestamp(snapshotTimestampBuilder);
requestBuilder.build();
IngestionResponse = blockingStub.unaryIngestion(requestBuilder);
```

Application Layer: Java Client Library



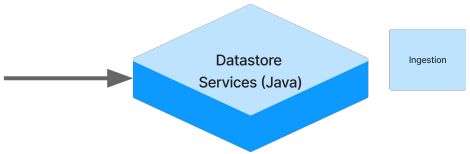
- A Java client library provides a high-level data science oriented interface to the data platform.
- It hides the gRPC API implementation from developers, allowing them to focus on the application instead of communication details.

Application Layer: Java Client Library Ingestion Example



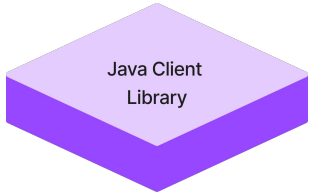
Example: Open streaming connection to Datastore Ingestion Service, create data frame for current interval, send to service.

Time	scalar PV	array PV	image PV	structure PV
2023-04-18 08:30:00.000	scalar value	array value	image value	structure value
2023-04-18 08:30:00.100	scalar value	array value	image value	structure value
2023-04-18 08:30:00.200	scalar value	array value	image value	structure value



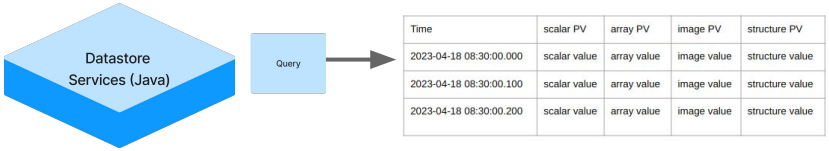

```
// Create interface to streaming ingestion service.
IIngestionStream ingStream = DsIngestionServiceFactory.connectStream();

// Open stream for PV data provider registration.
ingStream.openStream(pvProviderRegistration);

// Create DataFrame from PV table for current interval.
DataFrame dataFrameCurrentInterval = DataFrame.from(pvValueTable);

// Send data frame to ingestion service in current stream.
ingStream.streamData(dataFrameCurrentInterval);
```

Application Layer: Java Client Library Data Science Example



Objective: Retrieve PV data from Datastore Repository to train a machine learning model, or feed data to predictive ML model embedded in a control application.

Approach: Use Datastore streaming query API to retrieve latest BPM values for relevant PVs and feed them to ML model.



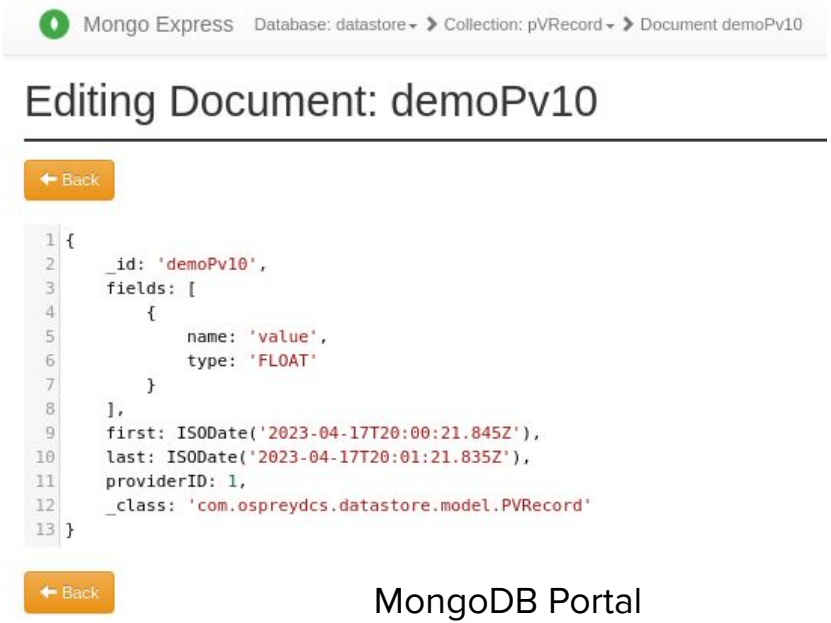

```
/ Create interface to query service.
IQueryServiceData qrySvc = DsQueryServiceFactory.connectData();

// Create query request for relevant PVs starting from now.
qryRequest = qrySvc.newRequest();
qryRequest.rangeAfter(Instant.now()); // or use "rangeBetween" to specify time range
qryRequest.selectPvs("scalar_PV", "array_PV", "image_PV", "structure_PV");

// Send query request and invoke callbackFunction with query results as they are available.
IDataTableDynamic tblResult = qrySvc.requestDataAsync(qryRequest, callbackFunction);
```

Toolbox: Feeding EPICS Data to the Ingestion Service

- There is work in progress to enhance the EPICS Aggregator so that it can feed the Data Platform Ingestion Service.
- It subscribes to EPICS data for a configured set of PVs, and pushes data to Ingestion Service using the gRPC API.

Toolbox: DB Tools for Data Access

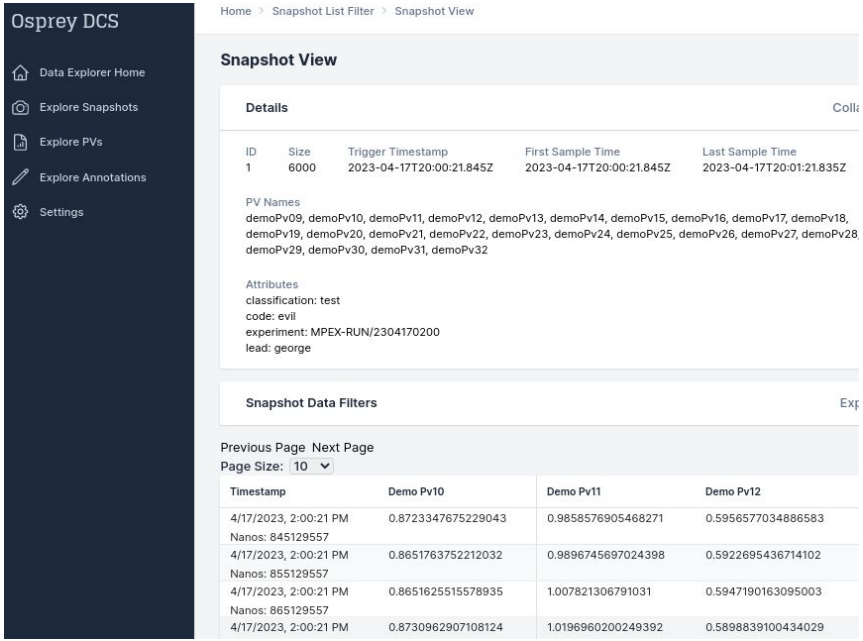
MongoDB provides a command shell and web portal for direct database access and data manipulation.



MongoDB Portal

Toolbox: Web Application

JavaScript web application uses gRPC Query API to navigate Data Platform repository.



Installation and Deployment

Data Platform Installation Process

- install MongoDB database package
- install repos for Java data platform server applications
- optionally install repos for Java client library
- optionally install repo for JavaScript web application
- install support repo for managing data platform ecosystem (building apps, managing services and applications, etc)