The Data Platform: An Independent System for Management of Heterogeneous, Time-Series Data to Enable Data Science Applications

C. McChesney, C. K. Allen, M. Davidsaver, B. Dalesio, Osprey DCS, Ocean City, Maryland, USA

Abstract

The Data Platform is a fully independent system for management and retrieval of heterogeneous, time-series data required for machine learning and general data science applications deployed at large particle accelerator facili-ties. It is an independent subsystem within the larger Machine Learning Data Platform (MLDP) which provides full-stack support for such facilities and applications. The Data Platform maintains the heterogeneous data archive along with all associated metadata and post-acquisition user annotations. It also facilitates all interactions between data scientists and the data archive; thus it directly sup-ports all back-end data science use cases. Accelerator facil-ities include thousands of data sources sampled at high frequencies, so ingestion performance is a key requirement and the current challenge. We describe the operation, architecture, performance, and development status of the Data Platform (DP).

Data Platform Installation and Deployment

The DP has a formal installation system available at: https://github.com/osprey-dcs/data-platform which includes instructions and additional documentation. The repository access is soon to be public as we secure the API, however, interested parties can contact Osprey DCS to obtain development releases now.

PROJECT BACKGROUND

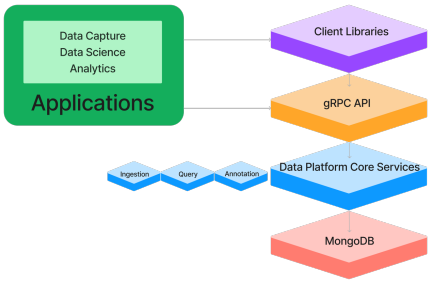
Several objectives were stated at project initiation, for both performance and operation.

1. Provide an Applications Programming Interface (API) for ingestion of heterogeneous, time-series data including scalar values, arrays, structures, and images.
2. Handle data rates expected for an experimental research facility such as a particle accelerator (minimum. performance requirement is 4,000 scalar data sources sampled at 1 KHz, i.e., 4 million samples per second).
3. Provide an API for retrieval of ingested heterogeneous data.
4. Provide an API for exploring metadata available in the archive.
5. Provide mechanisms for adding post-ingestion annotations to the archive and performing queries over those annotations.

DATA PLATFORM TECHNICAL COMPONENTS

Visible to clients are 2 primary technical components.

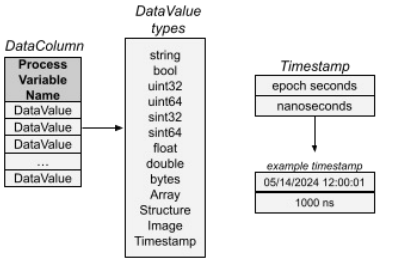
1. The *DP API* is built with the gRPC high-performance remote procedure call (RPC) framework. The API may be used directly by external applications, using gRPC's many programming language bindings. Additionally, applications may use a custom Java Client Library for higher level functionality.
2. The *DP Core Services* are implemented as Java server applications. There are currently 3 independent services: Ingestion, Query, and Annotation. All services utilize the MongoDB document-oriented database for persistence.



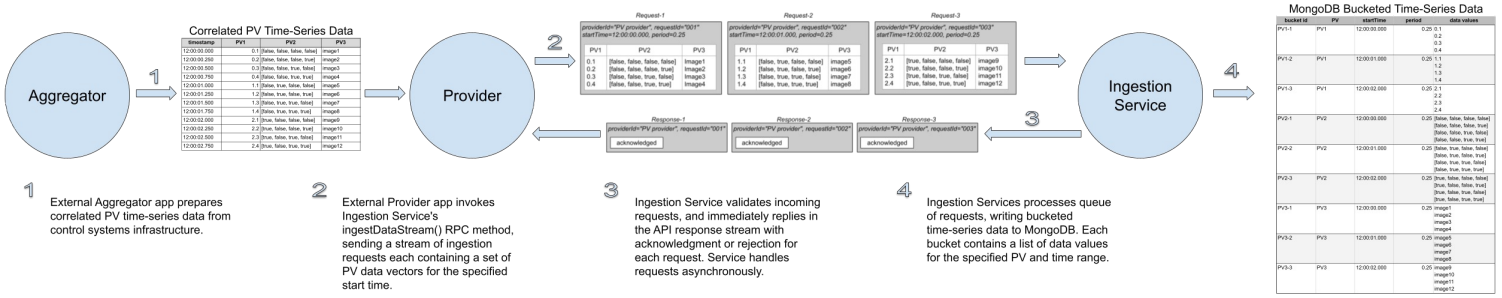
DP API DATA MODEL

The gRPC API defines the data model for the Data Platform. Key elements include Process Variables, Data Vectors, Heterogeneous Data Values, and Time.

- The core data element of the DP is the *process variable (PV)*. These are processes within the facility being monitored or controlled. Process variables are sampled within the control system to create correlated, time-series data.
- The Ingestion and Query Service APIs for handling data work with *vectors of PV measurements*. In the APIs this aspect is reflected with data type "*DataColumn*", which includes a PV name and list of measurements.
- The API supports *heterogeneous data values* in column vectors through the use of the API type "DataValue", which allows a variety of data types for PV measurements including numerous scalar types, as well as multi-dimensional arrays, structures, and images. Both Ingestion and Query Service APIs use this data type to provide a streamlined method interface.
- *Time* is represented with the "Timestamp" data type. It contains fields for 1) seconds since the epoch start, and 2) nanosecond offset from the epoch second. API time methods use either the data type "TimestampList" to send an explicit list of timestamps, or a "SamplingClock" message that includes the parameters of a uniform sampling clock (i.e., start time, sample period, sample count).



DP INGESTION SERVICE



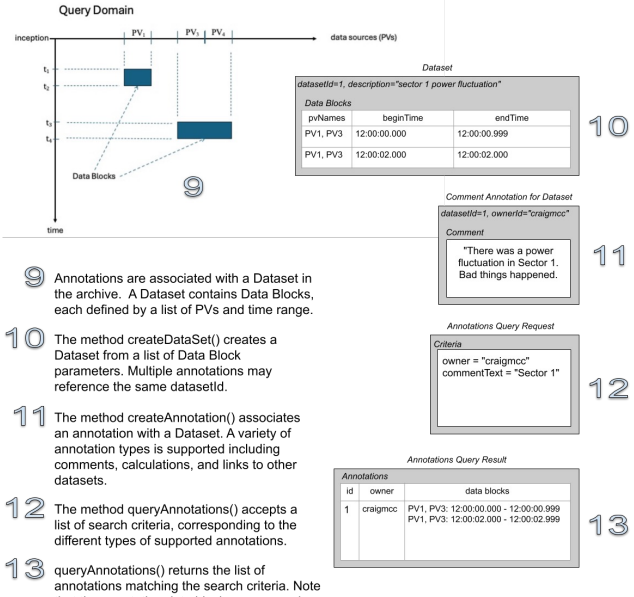
1 External Aggregator app prepares correlated PV time-series data from control systems infrastructure.

2 External Provider app invokes Ingestion Service's ingestDataStream() RPC method, sending a stream of ingestion requests each containing a set of PV data vectors for the specified start time.

3 Ingestion Service validates incoming requests, and immediately replies in the API response stream with acknowledgment or rejection for each request. Service handles requests asynchronously.

4 Ingestion Services processes queue of requests, writing bucketed time-series data to MongoDB. Each bucket contains a list of data values for the specified PV and time range.

DP ANNOTATION SERVICE



9 Annotations are associated with a Dataset in the archive. A Dataset contains Data Blocks, each defined by a list of PVs and time range.

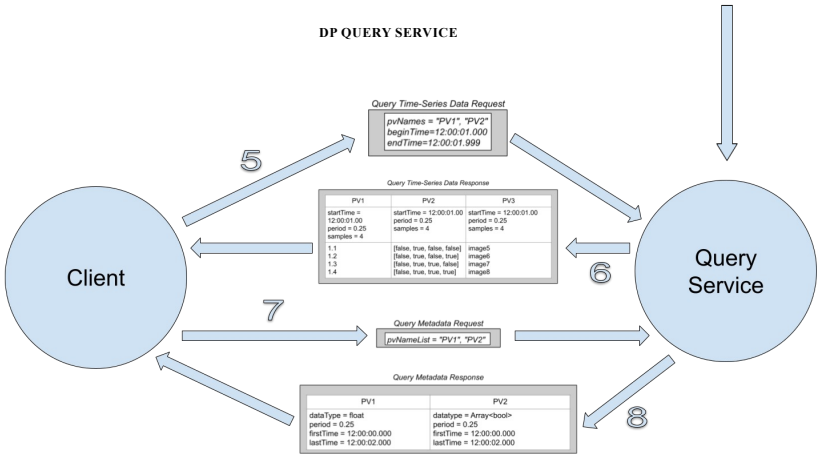
10 The method createDataSet() creates a Dataset from a list of Data Block parameters. Multiple annotations may reference the same datasetId.

11 The method createAnnotation() associates an annotation with a Dataset. A variety of annotation types is supported including comments, calculations, and links to other datasets.

12 The method queryAnnotations() accepts a list of search criteria, corresponding to the different types of supported annotations.

13 queryAnnotations() returns the list of annotations matching the search criteria. Note that the annotation data blocks correspond directly to the domain of the Query Service's time-series data query methods.

DP QUERY SERVICE



5 Client sends time-series data query request via queryData() RPC method to retrieve measurements for specified list of PVs and time range. Additional methods offer streaming and tabular formatted results.

6 Query Service responds with a set of data buckets matching query parameters, each including PV name, startTime, sample period, number of samples, and list of PV data values.

7 Client sends metadata query request via queryMetadata() RPC method to learn about data sources available in the archive, specifying list of PV names or regex pattern.

8 Query Service responds with metadata for each PV matching the query parameters, including data type, sample period, and first and last sample times for each.