# Pravega OLAP

Integration of Pravega and online analytical processing (OLAP) database with Java

# Requirements and Specifications

Dell Technologies



Jose Robles, Nate Tsige, Boxiang Lin

9/30/2022

# Contents

# Project Description

## I. Introduction

Dell Technologies currently takes charge of an open-source project that is known as Pravega. This is an infrastructure that serves as a storage system that implements data streams to store/serve data [1]. These data streams are made up of sections which contain events. These are sets of bytes in a stream that represent some sort of data – Pravega is effective at storing/ingesting these due to its data streams being consistent, durable, elastic, and append-only [2].

Pravega stores ingested data from many different sources in a row-oriented manner which allows for all data points relating to one object to be stored in the same data block. This is beneficial for queries needing to read and manipulate an entire object, but it is slow to analyze large amounts of data. This is an issue because when we want to process events via big data analytics queries, efficiency is poor due to the row-oriented structure of Pravega. A column-oriented processing engine in which columns store similar data points for distinct objects within a block would allow for a quicker analysis of data points, as well as the compression of columns which is efficient for storing lots of data. Without ingesting Pravega events into a proper big data analytics engine, queries against the events are very slow and not feasible for a system storing as much data as Pravega.

## II. Background and Related Work

Businesses are always looking for ways optimize their efficacy and to profit. One way to do that is to analyze data from the data source. In the early days, businesses were having difficulties as they navigated data due to the intense on-the-fly processing needed which resulted in the rise of OLAP databases. OLAPs pre-possess the data obtained from the source and store them. The processed data is instantly available for analysis.

One popular distributed event store and stream-processing platform is Apache Kafka. Kafka is a message-oriented middleware. While Kafka is great for transaction and event streams it lacks many futures that are necessary for modern data-intensive applications. Dell's Pravega further enhances programming models like Kafka and provides a cloud-native streaming infrastructure that enables a wider scope of applications by providing additional features like long-term retention, durability, auto-scale, and ingestion of large data to name a few [3].

However, Pravega is not an analytic engine hence it cannot process the data it ingests. The primary objective is to create plug-ins for either Apache Druid or Apache Pinot to enable integration to perform analysis of events from Pravega stream. This will allow users to make big data analytic queries on data that is passing through their stream.

In order to successfully provide a solution to the problem, the team will need some background knowledge of the problem space. A comprehensive understanding of Pravega and Apache Druid or Pinot is essential to solving the problem. In addition, the team will need to have a fundamental knowledge of data organization such as row-oriented and column-oriented databases, as well as experience developing in Java and familiarity with the SQL language.

## III. Project Overview

Compared to similar streaming storage systems like Apache Kafka and Apache Pulsar, Pravega provides a more extended data retention period, auto-scaling of partition, and more [3]. However, since Pravega is a storage engine and not an analytics engine, streams don't get analyzed inside Pravega. Online analytical processing (OLAP) database is a software that enables users to quickly, consistently, and interactively observe information from all aspects to gain a deep understanding of data [4]. Hence, integrating Pravega with the OLAP database will allow users to perform log-based analysis against the stream data they stored in Pravega and therefore strengthen Pravega's ecological system, establishing a bridge between the storage engine to the analytics engine.

Integration of a storage engine and analytics engine for big data streaming is powerful and so beneficial as the needs for the Industrial Internet of Things, Internet of Vehicles, and real-time fraud risk control are developing rapidly. For technology to provide better services and customer experiences, we need applications to respond quickly to customer needs while still learning and adapting to changing behavior patterns. To be able to achieve that, an excellent streaming storage engine like Pravega and a great streaming analytics engine like Apache Druid are needed.

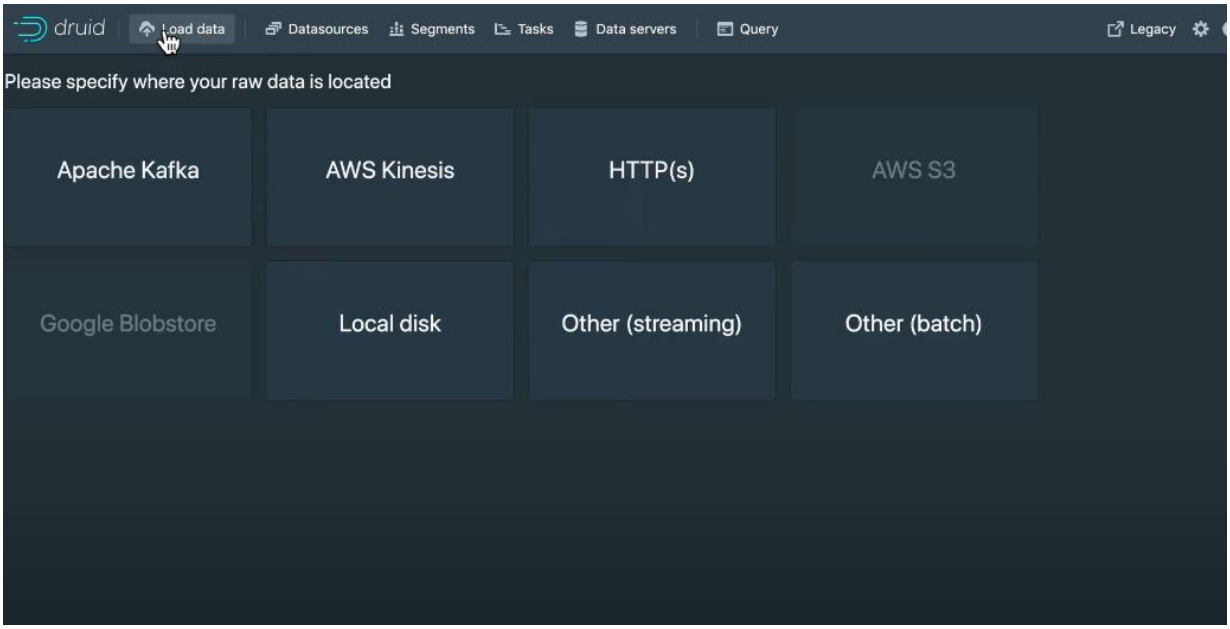
To integrate Pravega with Apache Druid, we will need to develop a plugin for the existing Apache Druid database system. This plugin will serve as a connector that first locates the source stream data in Pravega, then retrieves the stream data and transposes the data from row-oriented format to column-oriented format, and finally ingest them into the Apache Druid database. Since the Apache Druid and its provided client APIs are written in Java. We are expected to develop this plugin primarily using Java. We will also write some SQLs for data transposition and modify some configuration settings for Pravega connection.

We will develop this plugin project in Agile mode that cycles through processes of planning, executing, and evaluating. We are planning to have each sprint last for 3 weeks and each user story or use case to be tracked on GitHub Issue.

We will employ test-driven development (TDD) by having black-box test cases written before actual implementation and then finalizing the test cases after the actual implementation [5]. We might have functional test cases where we will try to validate a small amount of sample data by writing scripts to insert a few rows of sample data into Pravega and test the functionality of the plugin.

We will first develop the plugin in our local environment and version controlled by Git to our course remote repository. Once all test cases and validation are passed, we will then request a pull request to submit our plugin into Apache Druid GitHub repository.

Below is the screenshot of Apache Druid. As you can see on the “Load data” page there are plugins for Apache Kafka and AWS Kinesis stream storage system. We will develop a plugin for Pravega here.




## IV. Client and Stakeholder Identification and Preferences

The primary client and stakeholder for this project is Dell Technologies, as they lead the development of the open-source distributed storage system Pravega. Stakeholders also include the JNB team comprised of WSU students that are working to develop this project, as well as the Pravega team at Dell and our Dell software engineer mentors.

The primary need of the Pravega team is for a plugin to be created that allows for an OLAP database to be integrated with Pravega such that the automatic ingestion of Pravega data streams can be enabled to be sent to an OLAP database. A stretch goal that has been highlighted is to produce plugins for both Apache Druid and Pinot (OLAP databases), but the primary goal is to produce just one plugin.

Other stake holders include companies that utilize Pravega to ingest data. Two companies that are public about their adoption of Pravega are Link Labs and Wheels up. Companies such as these will benefit from the new Pravega plugin that will allow for much more efficient big data analytic queries through the integration of an OLAP database.

# Requirements and Specifications section

## I. Introduction

Dell Technologies takes charge of an open-source project known as Pravega. This is an infrastructure that serves as a storage system that implements data streams to store/serve data [1]. These data streams are made up of sections which contain events. These are sets of bytes in a stream that represent some sort of data – Pravega is effective at storing/ingesting these due to its data streams being consistent, durable, elastic, and append-only [2].

Pravega stores ingested data from many different sources in a row-oriented manner which allows for all data points relating to one object to be stored in the same data block. This is beneficial for queries needing to read and manipulate an entire object, but it is slow to analyze large amounts of data. This is an issue because when we want to process events via big data analytics queries, efficiency is poor due to the row-oriented structure of Pravega. A column-oriented processing engine in which columns store similar data points for distinct objects within a block would allow for a quicker analysis of data points, as well as for the compression of columns which is space efficient. Without ingesting Pravega events into a proper big data analytics engine, queries against the events are very slow and not feasible for a system storing as much data as Pravega.

## II. System Requirements Specification

The overall goal for the integration of Pravega and an OLAP database is so that the automatic ingestion of data from Pravega can be enabled to be sent to a Apache Druid (OLAP). With this, we would want a user to be able to perform log-based analytics against the “events” in the data streams. The integration of the OLAP database with Pravega as the processing engine will allow for very efficient big data analytics queries due to its column-oriented structure. Apache Pino is another OLAP database that is available for integration with Pravega. Currently this is a stretch goal, and the team will be focusing on integrating Druid with Pravega first.

This integration has several use cases in which users would use the plugin in a variety of different scenarios in order to make big data analytics queries against the events stored in Pravega’s data streams. We assume that the client in these use cases has already integrated their own system with Pravega so that Pravega can automatically ingest information from their system (this data is the one being queried). This plugin also has several features (functional and non-functional) that will be highlighted below.

### II.1. Use Cases

**Story:** A large retail corporation WalmartABC operates a chain of hypermarkets all around the world. The IT department of WalmartABC built an app that keeps track of product sales in different regions. There are tons of sales data produced every second, WalmartABC stores those data in Pravega with real-time insertion. WalmartABC wants to know what kinds of products are most popular in different parameters to prevent product shortages. To learn the usage patterns, they need to do some big data analytics and machine learning. However, since

queries are slow in Pravega, they will ingest part of the diagnostic data into Apache Druid or Apache Pinot and perform the big data analytics there.
**Source**: Senior Software Engineer from DELL provides the idea of the story.

**Story:** A large electrical power system protection corporation SchweitzerABC Engineering Laboratories operates power protection services all around the United States. The software IT department is now building a cloud visualization application "SynchrowaveABC" to visualize the real-time state of their protection equipment at different stations, improve understanding of system events and expedite root cause analysis with high-resolution time-series data. SynchrowaveABC brings synchrophasor data and relays event reports together into one place so engineers can analyze both the high-level system impact of an event and the detailed oscillography data.

They set up a connection for real-time data to be stored in Pravega. They planned to support different filter interfaces to visualize the state of equipment for the past 30 days. They ingest the real-time data of the past 30 days into Apache Druid or Apache Pinot. The app SynchrowaveABC will then be able to retrieve the data from Apache Druid or Apache Pinot for different filtering and display it with signal charts, tables, reports, etc.
**Source**: Senior Software Engineer from DELL provides the idea of the story.

**Story:** A large E-commerce corporation, Bmazon provides online shopping services all around the world. Each month, there are more than 197 million people visiting the website and purchasing items.

To better serve customers, they need to have a good recommendation system that learns each user's usage pattern and displays items highly matched to each individual user on the front page. They have the data of each individual browse and purchase history stored in Pravega with a real-time connection.

Now they have designed a few Machine Learning algorithms to train those data in different parameters. They must ingest part of the data from Pravega to Apache Druid or Apache Pinot. Since their search and filter capabilities enable rapid, easy drill-downs of users along any set of attributes.
**Source**: Principal Software Engineer from DELL provides the idea of the story.

## II.2. Functional Requirements

### III.2.1 Automatic ingestion data stream into Distributed OLAP

**Automatic ingestion**: Our plug-in should enable automatic ingestion of data stream into an OLAP database. This future may vary based on which distributed OLAP is ingesting the stream. In our case, we are integrating Pravega with Druid or Pinot. Druid and Pinot have two streaming ingestion methods (Kafka and Kinesis). Both Kafka and Kinesis provide Exactly-once guarantees a future discussed in the transaction future.

**Source**: This requirement was requested by Dell in the original abstract of the project and is the core of the project. This future is useful for anyone doing log-based data analytic against the event in their stream.

**Priority**: Priority Level 0: Essential and required functionality.

### III.2.2. Checkpoints for the plugin

**Checkpoints**: The plug-in must support checkpoints that keep track of logs during a transaction. This checkpoint is used to verify and validate modified data in real-time. The checkpoints will also be useful to create backup and recovery prior to any application of data modification in the database. This will allow us to resume from clean or unclean shutdown using the recovery system to return to the checkpoint state.

**Source**: This requirement was requested by a senior principal engineer at Dell and Pravega.io. The requirement is necessary for clients that may encounter a failure during a transaction allowing them to easily default back to the checkpoint state.

**Priority**: Priority Level 0: Essential and required functionality.

### III.2.3. Transactions

**Transactions:** The plug-in must support transaction such that when a client says commit, the whole data is appended to the log. Transactions are all or nothing / exactly once semantic. This means an event is delivered and processed exactly once.

**Source:** This requirement was requested by a senior principal engineer at Dell and Pravega.io. This future is useful for anyone doing log-based data analytic against the event in their stream.

**Priority**: Priority Level 0: Essential and required functionality.

### III.2.4. Schema Registry

**Schema Registry**: Given the data stored in the Pravega stream is unstructured to minimize payload size Pravega has a stand-alone schema registry future that contains the data type definition for events within the stream. The plugin must support this future.

**Source**: This requirement was requested by a senior principal engineer at Dell and Pravega.io. This future is necessary for anyone running big data query on Pravega. Pravega receives raw bytes and delivers raw bytes. This future is intended to help clients who want deserialize the raw bytes which requires knowing the data type definition.

**Priority**: Priority Level 0: Essential and required functionality.

### III.2.5 Operation on data

**Search**: This future will take an input of data request and returns the result from the database.

**Source**: Chief data and analytics Engineer at Dell and Pravega.io. This function will help clients do the easiest data query task which is to search.

**Priority**: Priority Level 0: Essential and required functionality.

### III.2.6 Proactive Creation of Tables within Apache Druid

**Table Creation**: Pravega data streams store events of certain data types – streams will vary in the data in which they contain. With this, we want to store data streams to certain tables within the OLAP database. When we encounter a data stream of x data type, we should not store it

within an existing table for data streams of y data. The database should expect to see streams of different types and create new tables to store them accordingly.

**Source**: This requirement was suggested by a Dell engineer. This would be good to have to make the OLAP database efficient, it is important, but it is not the highest priority.

**Priority**: Priority Level 1: Important feature, not highest priority

### II.3. Non- Functional Requirements

**Easy to Use:** The plugin should be easy to use. Its user interface should be user friendly and self-explanatory. Users should be able to easily make queries against certain Pravega data streams for processing within the OLAP database.

**Reliability:** The plugin is expected to ingest all data selected without any loss. We aim to maintain system integrity by ensuring that no data is lost during the transfer of data streams from Pravega to the OLAP database.

**Maintainable:** The plugin will be available as open source on Apache Druid GitHub Repository, it should be maintainable to the public.

**Cross Platform:** The plugin should be working on different operating systems.

**Code Comment:** Every method in the Plugin should be properly commented.

**Usage Manual:** There should be a usage manual provided along with the Plugin in production to allow users to clone the repository and make working queries against Pravega streams.

**Extensible:** The plugin should be extensible so future developers can easily add additional features to it without changing the existing structure of the code.

**Performance:** The plugin should be efficient in ingesting data from Pravega to Apache Druid.

**Storage:** Apache Druid should be able to store ingested data from Pravega with immediate concerns of running out of storage.

**Security:** Plugin must be secure to ensure that data is not tampered with in transit from Pravega to Apache Druid.

**Scalability:** The system should be able to be scaled to allow for different systems to be integrated with Pravega.

## III. System Evolution

The delivery of the plugin should be functional, bug free, and versatile in the sense that it is able to adapt to unforeseen changes in requirements. Assumptions must be made for potential changes that we may anticipate during development. With this list of possible changes/risks, the team can quickly adapt to the change and refer to this section in order to decipher what the next steps are to handle the scope/requirement change.

**Deployment**: Currently, the Dell team is aiming for us to deploy our plugin via Kubernetes. We are assuming the efficiency of Kubernetes will be sufficient for our project to be deployed and perform as expected. If change occurs, whether that be a software change of Kubernetes, or a change in the client's requirements, we propose deploying instead using AWS.

**Development (Java):** This project will be completed using Java8 development. This is the most recent version of Java, and we are assuming that Pravega and the OLAP database are compatible with it. We should anticipate potential compatibility issues with Java8 and Pravega or Apache Druid or Pino. Should this occur, we aim to instead develop with an older version of Java7 in hopes of eliminating compatibility issues.

**OLAP Database**: This project allows for our team to select either Apache Druid or Pino as our processing engine for the Pravega streams. As of now we are undecided on which we will be integrating with Pravega, but we are leaning towards Druid. We should anticipate the possibility of Druid having less functionalities or being less efficient than Apache Pino. If we integrate Druid and find our big data analytics queries performing poorly (assuming everything was integrated correctly), we should be able to adapt quickly and integrate Apache Pino with Pravega instead.

**Testing:** Once the integration is complete, we will need to test the processing engine by creating sample queries that we can make to fetch and process information within the Pravega streams. If sample queries prove to be inefficient for testing this plugin, or if we are unable to fetch data from Pravega in a testing environment, then we will use Debezium, another data source/platform so that we can test our sample queries on Debezium data sources.

**Integration of both Apache Druid and Pino:** Primary goal of this project is to just integrate one OLAP database with Pravega. We should anticipate the possibility of finishing the Druid integration early and also integrating/creating a plugin for Apache Pino to process data from Pravega.

# Glossary

Auto scaling: A Pravega concept that allows the number of Stream Segments in a stream to change over time, based on scaling policy.
Big Data – Refers to data sets that are too large or complex to be dealt with by traditional data processing software
Checkpoints - A kind of Event that signals all readers within a Reader Group to persist their state.
Data streams – method of organizing data, has a data source and a destination. Comprised of stream segments which contain events
Events – Set of bytes contained within stream segments to represent some data point
OLAP – Online analytical processing database
Pravega – Open-source storage system implementing streams for storing/serving continuous and unbounded data.
Query – Refers to a select query which retrieves data from a database and an action query that applies operations on the data.
Transaction – A collection of stream write operations that are applied atomically to the stream.

# References

[1] “Pravega Concepts” *Concepts - Exploring Pravega*. [Online]. Available: https://cncf.pravega.io/docs/nightly/pravega-concepts/#introduction. [Accessed: 23-Sep-2022].

[2] “Pravega Overview” *Exploring Pravega*. [Online]. Available: https://cncf.pravega.io/docs/v0.11.0/. [Accessed: 23-Sep-2022].

[3] Pravega, “A reliable stream storage system,” *Pravega*, 21-Mar-2022. [Online]. Available: https://cncf.pravega.io/. [Accessed: 23-Sep-2022].

[4] Foundation, A., 2022. Druid | Database for modern analytics applications. [online] Druid.apache.org. Available at: <https://druid.apache.org> [Accessed 23 September 2022].

[5] Hamilton, T., 2022. What is Test Driven Development (TDD)? Tutorial with Example. [online] Guru99. Available at: <https://www.guru99.com/test-driven-development.html> [Accessed 23 September 2022].