**Report on**

**DRUID - A Real-time Analytical Data Store**

Introduction

Druid is an open-source data store designed for sub-second queries on real-time and historical data. It is primarily used for business intelligence ([OLAP](http://en.wikipedia.org/wiki/Online_analytical_processing)) queries on event data. Druid provides low latency (real-time) data ingestion, flexible data exploration, and fast data aggregation. Existing Druid deployments have scaled to trillions of events and petabytes of data. Druid is most commonly used to power user-facing analytic applications.

Why driud?

Druid was originally designed to solve problems around ingesting and exploring large quantities of transactional events (log data).

The need for Druid was facilitated by the fact that existing open source relational Database Management Systems (RDBMS) and NoSQL key/value stores were unable to provide a low latency data ingestion and query platform for interactive applications.

In the early days of metamarkets, they faced the following challenges:

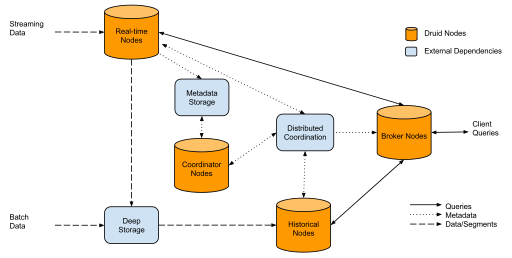
* The data store powering the dashboard needed to return queries fast enough that the data visualizations built on top of it could provide users with an interactive experience.
* The system had to be multitenantand highly available.
* They wanted to make the Business decisions in real-time and Popular open source data warehousing systems such as Hadoop were unable to provide the sub-second data ingestion latencies

There are three types of data Components.

* **Timestamp column**: We treat timestamp separately because all of our queries center around the time axis. Timestamps are faceted by varying granularities
* **Dimension columns**: They each represent an axis of the data that we’ve chosen to slice across. Dimensions are string attributes of an event, and the columns most commonly used in filtering the data.
* **Metric columns**: These represent values, usually numeric, which are derived from an aggregation operation – such as count, sum, and mean (we also run variance and higher moment calculations). Metrics are columns used in aggregations and computations.

Architecture

Druid Cluster



A Druid cluster consists of different types of nodes and each node type is designed to perform a specific set of things.

* **Real-time Nodes**

Real-time processing in Druid can currently be done using standalone realtime nodes or using the indexing service. The real-time logic is common between these two services. Real-time processing involves ingesting data, indexing the data (creating segments), and handing segments off to historical nodes. Data is queryable as soon as it is ingested by the realtime processing logic. The hand-off process is also lossless; data remains queryable throughout the entire process.

* **Historical Nodes**

Historical nodes commonly form the backbone of a Druid cluster. Historical nodes download immutable segments locally and serve queries over those segments. The nodes have a shared nothing architecture and know how to load segments, drop segments, and serve queries on segments.

* **Broker Nodes**

Broker nodes are what clients and applications query to get data from Druid. Broker nodes are responsible for scattering queries and gathering and merging results. Broker nodes know what segments live where.

* **Coordinator Nodes**

Coordinator nodes manage segments on historical nodes in a cluster. Coordinator nodes tell historical nodes to load new segments, drop old segments, and move segments to load balance.

External Dependencies

* **Metadata Storage Druid** relies on a metadata storage to store metadata about segments and configuration. Services that create segments write new entries to the metadata store and the coordinator nodes monitor the metadata store to know when new data needs to be loaded or old data needs to be dropped. The metadata store is not involved in the query path. MySQL and PostgreSQL are popular metadata stores for production, but Derby can be used for experimentation when you are running all druid nodes on a single machine.
* **Deep Storage** acts as a permanent backup of segments. Services that create segments upload segments to deep storage and historical nodes download segments from deep storage. Deep storage is not involved in the query path. S3 and HDFS are popular deep storages.
* **Druid coordinator nodes** have Zookeeper and MySQL as external dependencies.

Conclusions

Druid is designed to power high performance applications and is optimized for low query latencies.

Druid supports streaming data ingestion and is fault-tolerant.

Also helps in Arbitrary slice and dice data exploration.

Druid also gives Approximate and exact computations