Druid

Druid is a column-oriented,open-source, distributed data store written in Java. Druid is designed to quickly ingest massive quantities of event data, and provide low-latency queries on top of the data
Druid is an open source data store designed for OLAP queries on event data. This page is meant to provide readers with a high level overview of how Druid stores data, and the architecture of a Druid cluster. If you are acquainted with OLAP terminology, the following concepts should be familiar. Timestamp column, Dimension columns, Metric columns Druid shards are called segments and Druid always first shards data by time. In our compacted data set, we can create two segments, one for each hour of data.

Overview of Druid:

Column-oriented data store, Realtime ,streaming ,ingestion, Automatic summarization ,Ad-hoc queries , Approximate algorithms,Can keep around a lot of history,Open source.

Datastore Requirements :

Arbitrary filtering, splitting, and aggregation, Respond quickly to queries (ideally < 1 second response time), Handle huge amounts of data (up to Petabytes in total), Handle streaming data

Characteristic Use of Druids:

Druid is ideal for business intelligence/OLAP use cases that: Require interactivity ,Involve filtering, grouping, aggregating data ,Have result set input set

Realtime Ingestion Performance
Over 500,000 events/ second, average Over 2M events / second, peak ~10 – 100k events / second / core and queries are about 500ms average query latency

Data Partitioning or Sharding the Data
Shards are called “segments” in Druid. First level partition done on time, Done so for query optimization, Segments are immutable. Segments contain data stored in compressed column orientations, along with the indexes for those columns

Immutable Segments :Fundamental storage unit in Druid,No contention between reads and writes, One thread scans one segment.

Druid supports:

- Hyperloglog ,
- Theta sketches ,
- Approximate Histograms

Architecture

The Druid Cluster
Fully deployed, Druid runs as a cluster of specialized processes to support a fault tolerant architecture where data is stored redundantly, and there is no single point of failure.

Node Types : Historical Nodes, Broker Nodes, Coordinator Nodes, Real-time Processing

Real-time Nodes Stores data in write-optimized structure: on-heap hash map Converts write optimized structure -> read optimized structure Read-optimized data structure: Druid segments Can query data immediately

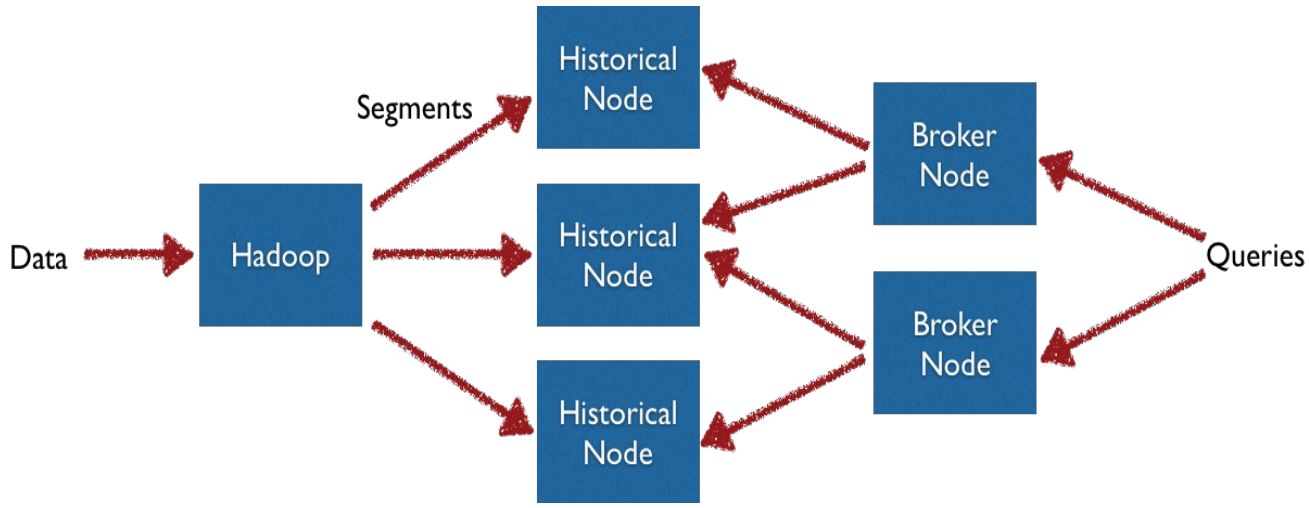
Query libraries:

- JSON over HTTP
- SQL
- R
- Javascript
- Python
- Ruby

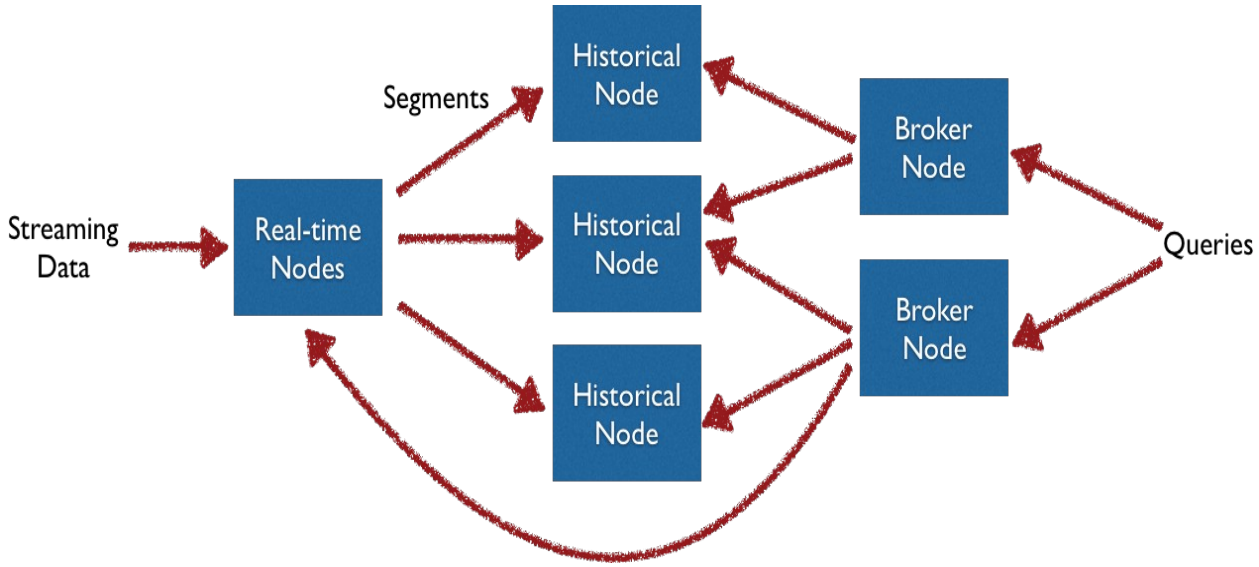
User Interfaces:

- Pivot
- Grafana
- Caravel (formerly Panoramix)

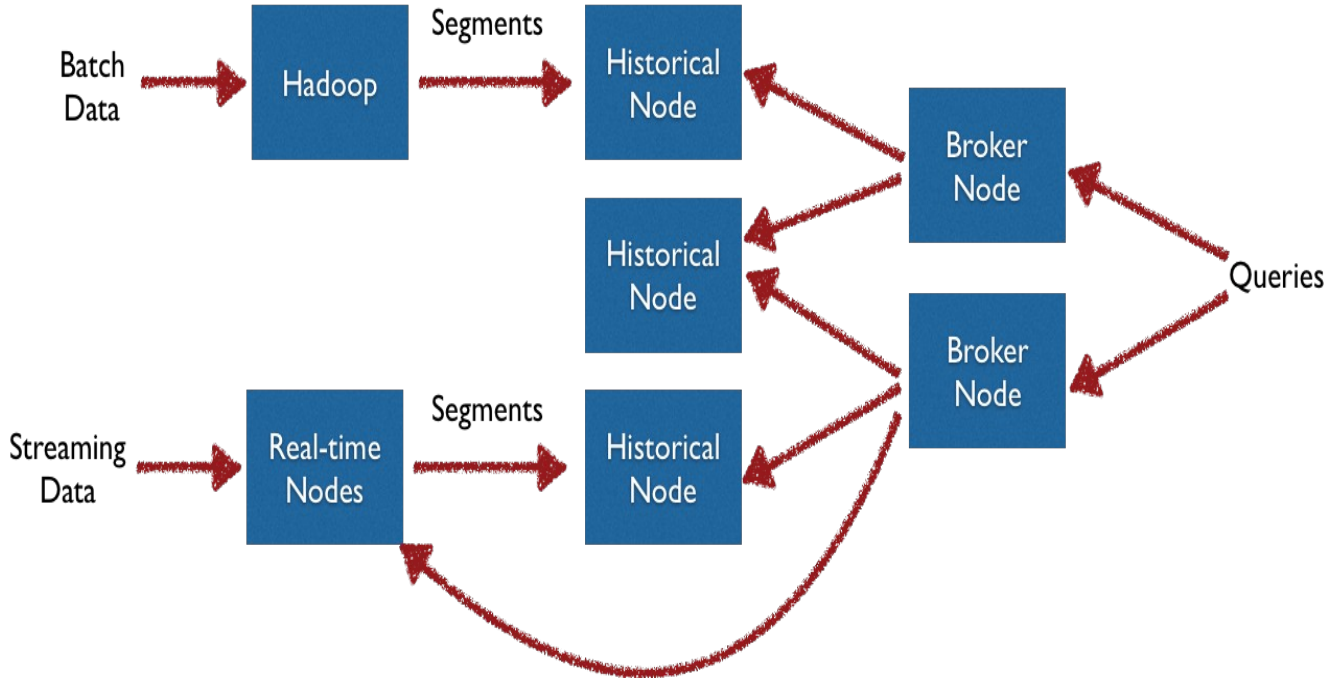
Architecture (Batch Ingestion)



Architecture (Streaming Ingestion)



Architecture (Lambda)



Druid is made for for analytic applications, Druid is good at fast OLAP queries, Druid is good at streaming ingestion

Case Study by - Varun Kashyap.K.S
Reg.No - 161046020