SNAP Vs Druid

A comparison of use cases

Datawarehousing

- Typical use case
  - Star schemas - Complex fact to dimension joins with thousands of columns
  - Need to choose which columns in dimension tables should be in the index
    - Example Type 1 changes should be outside the index since cost of reindexing in high

SNAP	DRUID
Full Star schema support	No Star schema support
Ability to choose columns in the index	Cannot combine non index and index cols in a query
Type 1 changes handled with Dimension context propgation	No support for Type 1 changes - Requires expensive reindexing
Designed to work seamless with Tableau/OBIEE	Does not work except for basic SQL (example dimension drag and drop on Tableau)

Partitioning

- Typical use case
  - Many use cases require partitioning by non-timestamp fields
    - A large ad tech company deals with many advertisers each with 200 million rows a month
    - Advertiser is primary filter

SNAP	DRUID
Any field can be defined as a partition	No partitioning support - Timestamp are the only means of segment level partitioning
SNAP allows the same partitioning schema as in Hadoop/Spark	Druid indexes cannot be partitioned
Partition pruning with indexing will beat Druids performance since less segments to scan	In the example of advertiser_id all partitions have to be scanned when the query has no time filter

Advanced analytics functions

- Typical B.I use cases
  - Window operations
    - Druid does not allow iterative analysis on subsets of a cube for finding moving averages, aggregations of a part of cube etc
  - Metadata driven calculations
    - In Tableau Level of Detail calculations can result in self joins to calculate a aggregate value at a specific dimension(example cohort analysis)

SNAP	DRUID
Allows for full analytics - windowing, CTE’s and mllib with Spark	Basic aggregations and counbt
Fully integrated index with Spark SQL allows for any Spark SQL query to be rewritten to use SNAP Index	No support for nested aggregations from Traditional B.I tools
Writebacks and forecasting is possible by combining SNAP with Spark in one single operation	No support for an integrated approach for planning, forecasting and other typical B.I use cases

Runtime

- Cost of operations
  - Cluster deployment
    - What does it cost to run Druid
    - How well does it integrate with existing Big Data ops
    - How does is scale?
  - Ease of support
    - What does it take to run and support Druid?
    - How many components does an IT team need to learn in addition to existing Big Data deployments?

SNAP	DRUID
Fully integrated into Spark run time - SNAP cluster is a Spark cluster	Druid is separate component that does not work well with Hadoop/ YARN based resource management
SNAP is easy to setup and can run standalone, Mesos or YARN. Can be deployed on any cloud running Hadoop or Bare metal	Can use HDFS as a deep store but has to be a separate devops and management exercise
SNAP is cost effective - Indexing/ Querying all integrated into one simple SQL operation	Indexing and Querying are separate(Indexing is a Map reduce job) and for large datasets requires Hadoop
SNAP requires minimal training for EDW teams that already know SQL and Hadoop	Druid is a learning curve for deployment and a steep one
Very robust end to end monitoring from query to query plan to run time statistics with full visibility to choke points	Very hard to debug, primitive monitoring tools and not integarted into existing Big Data ecosystem.

Comparison	SNAP	Spark+Druid	Druid
Type of analysis	Enterprise B.I/Adhoc analysis	SQL Reporting	Operational , time series reporting
Supports joins	Yes	Yes	Limited
SQL support	Spark SQL	Spark SQL	Basic
Index management	Robust Spark SQL based	Druid Based, Limited	Limited
Metadata	Rich B.I metadata	Limited	None
Type 1 change support	Yes and Optimized	Yes not optimized	None
Hybrid index/non-index	Yes	Yes	No
Tableau optimized	Yes	No	No connectivity to B.I tools
B.I Query pattern optimizations	Yes	Limited	None
Security	Integrated to Hadoop/S3/Kerberos through Spark	Not integrated. Druid indexing is done and managed by Druid	Not integrated with rest of Big Data security
Deployment	Spark Cluster	Spark Cluster + Druid Cluster	Separate stack
Datatype support for dimensions	Multiple datatypes	String+ but still limited	String only
Metric Binning, Timestamp extractions	Yes	No	No
Partitioning	Spark based - any column	Only timestamp	Only timestamp
Data science integrated	Spark based - All ML routines	Yes not optimized	None
Cost of management	Very low- Only Spark cluster	High - Spark + Druid	Medium - Druid cluster+Hadoop MR