Overview of Apache Kudu

Apache Kudu is an open source storage engine designed to handle very large amounts of data—terabytes or even petabytes. Kudu stores data directly on the file system of the machine it's running on. In other words, it's *not* built on top of an underlying distributed file system, as some storage systems are. (Apache HBase, for example, is built on top of HDFS.)

Kudu structures data as tables much like those in a relational database, but with the ability to scale to support more data and higher throughput than traditional RDBMSs. This means it provides excellent performance for analytical queries through mechanisms like SQL, while also offering great performance for workloads common in online databases, such as random access and lookups and updates by key.

Many storage systems excel at one or the other of these types of access patterns. Kudu excels at both, making applications simpler to develop and more scalable.

Features

In order to provide performant storage for applications that use a variety of access patterns, Kudu offers high throughput for scans through large portions of a table's data, as required by analytical use cases. It also has low latency to support quick access to individual rows or groups of rows.

Another key Kudu feature is its support for atomic transactions, much like traditional databases. This means that when you write data to a row, if any part of the operation fails, the entire operation fails, so you don't end up with data in an inconsistent state. Kudu supports atomic transactions for single-row operations, but not for multi-row or multi-table transactions.

Kudu Architecture

Kudu runs on a cluster of machines, which can range in number from just a few machines to thousands. These machines collectively store and manage Kudu's data. (Note that the Kudu cluster is distinct from the Hadoop cluster that includes HDFS, although hosts can be part of both.) The cluster architecture is designed for fault tolerance and high availability—meaning that the system will always be able to respond to client requests to read and write data. Kudu clusters are also easily scaled by adding more nodes to the cluster as your storage and throughput needs increase.

Kudu's architecture is based on *tablets,* each containing a subset of the data in a table. Instead of storing the data in a table in a single location, Kudu distributes the table's tablets to multiple machines in the cluster. Those machines store the tablet's data on their local disks and respond to client requests to read and write table data. Tables must have primary keys, and the table data is distributed using hash or range partitioning (and possibly both), with columnar storage. (See the reading on Parquet files in Week 3 for more on columnar formats.)

Kudu and Impala

Apache Impala is tightly integrated with Kudu to enable data analysts, data scientists, and developers to access data in Kudu using common SQL syntax. Kudu supports most Impala SQL commands, which you can use to create and modify Kudu tables; insert, modify, and delete data; and issue queries. The details are beyond the scope of this course.

For More Information

The following offer more information on Apache Kudu:

- Apache Kudu Overview (on the official website)
- Introduction to Apache Kudu (a training course available for purchase from Cloudera)
- Kudu: Storage for Fast Analytics on Fast Data

- Documentation for using Impala with Kudu
- Impala Guide: Using Impala to Query Kudu Tables