Hadoop and MongoDB

What is Hadoop?

Hadoop is a software technology designed for storing and processing large volumes of data distributed across a cluster of commodity servers and commodity storage. Hadoop was initially inspired by papers published by Google outlining its approach to handling large volumes of data as it indexed the Web. With growing adoption across industry and government, Hadoop has rapidly evolved to become an adjunct to – and in some cases a replacement of – the traditional Enterprise Data Warehouse.

Many organizations are harnessing the power of Hadoop and MongoDB together to create complete big data applications:

- MongoDB powers the online, real time operational application, serving business processes and end-users, exposing analytics models created by Hadoop to operational processes
- Hadoop consumes data from MongoDB, blending it with data from other sources to generate sophisticated analytics and machine learning models. Results are loaded back to MongoDB to serve smarter and contextually-aware operational processes – i.e., delivering more relevant offers, faster identification of fraud, better prediction of failure rates from manufacturing processes.

Before exploring how users create this type of big data application, first lets dig into the architecture of Hadoop.

Hadoop Architecture

Hadoop is an open-source Apache project started in 2005 by engineers at Yahoo, based on Google's earlier research papers. Hadoop then consisted of a distributed file system, called HDFS, and a data processing and execution model called MapReduce.

The base Apache Hadoop framework consists of the following core modules:

- Hadoop Common: The common utilities that support the other Hadoop modules.

- Hadoop Distributed File System (HDFS): A distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster.
- Hadoop YARN: A resource-management platform responsible for managing compute resources in clusters and using them for scheduling of users' applications.
- Hadoop MapReduce: A programming model for large scale data processing.

In addition to these base modules, the term 'Hadoop' has evolved to also include a dozens of other independent tools and projects that can be installed on top of or alongside Hadoop to simplify access and processing of data stored in the Hadoop cluster:

- Ambari: GUI for managing and monitoring Hadoop clusters.
- Hive: Data warehouse infrastructure providing SQL-like access to data.
- Pig: Scripting language for accessing and transforming data,
- Sqoop: Managing data movement between relational databases and Hadoop.
- Flume: Service for collecting data from log files into HDFS.
- Mahout: Machine learning library.
- Tez: Data-flow programming framework, built on YARN, for batch processing and interactive queries. Used increasingly to replace MapReduce for Hive and Pig jobs.
- Spark: In-memory cluster computing framework used for fast batch processing, event streaming and interactive queries. Another potential successor to MapReduce, but not tied to Hadoop. Spark is able to use almost any filesystem or database for persistence.
- Zookeeper: A high-performance coordination service for distributed applications.
- MongoDB Connector for Hadoop: Plug-in for Hadoop that provides the ability to use MongoDB as an input source and an output destination for MapReduce, Spark, HIVE and Pig jobs.

Building on the Apache Hadoop project, a number of companies have built commercial Hadoop distributions. Leading providers include MongoDB partners Cloudera, Hortonworks and MapR. All have certified the MongoDB Connector for Hadoop with their respective distributions.

Hadoop Under the Covers

Applications submit work to Hadoop as jobs. Jobs are submitted to a Master Node in the Hadoop cluster, to a centralized process called the JobTracker. One notable aspect of Hadoop's design is that processing is moved to the data rather than data being moved to the processing. Accordingly, the JobTracker compiles jobs into parallel tasks that are distributed across the copies of data stored in HDFS. The JobTracker maintains the state of tasks and coordinates the result of the job from across the nodes in the cluster.

Hadoop determines how best to distribute work across resources in the cluster, and how to deal with potential failures in system components should they arise. A natural property of the system is that work tends to be uniformly distributed – Hadoop maintains multiple copies of the data on different nodes, and each copy of the data requests work to perform based on its own availability to perform tasks. Copies with more capacity tend to request more work to perform.

Properties of a Hadoop System

There are several architectural properties of Hadoop that help to determine the types of applications suitable for the system:

- HDFS provides a write-once-read-many, append-only access model for data.
- HDFS is optimized for sequential reads of large files (64MB or 128MB blocks by default).
- HDFS maintains multiple copies of the data for fault tolerance.
- HDFS is designed for high-throughput, rather than low-latency.
- HDFS is not schema-based; data of any type can be stored.
- Hadoop jobs define a schema for reading the data within the scope of the job.
- Hadoop does not use indexes. Data is scanned for each query.
- Hadoop jobs tend to execute over several minutes and hours

How Organizations Are Using Hadoop

Rather than supporting real-time, operational applications that need to provide fine-grained access to subsets of data, Hadoop lends itself to almost for any sort of computation that is very iterative, scanning TBs or PBs of data in a single operation, benefits from parallel processing, and is batch-oriented or interactive (i.e., 30 seconds and up response times). Organizations typically use Hadoop to generate complex analytics models or high volume data storage applications such as:

- Risk modeling
- Retrospective and predictive analytics
- Machine learning and pattern matching
- Customer segmentation and churn analysis
- ETL pipelines
- Active archives

How Organizations Are Using MongoDB with Hadoop

Users need to make analytic outputs from Hadoop available to their online, operational apps. These applications have specific access demands that cannot be met by HDFS, including:

- Millisecond latency query responsiveness.
- Random access to indexed subsets of data.
- Supporting real time expressive ad-hoc queries and aggregations against the data, making online applications smarter and contextual.
- Updating fast-changing data in real time as users interact with online applications, without having to rewrite the entire data set. Serving analytics from Hadoop to online applications and users in real time requires the integration of a highly scalable, highly flexible operational database layer.

The following table provides examples of customers using MongoDB together with Hadoop to power big data applications.