# Tools for Secured Distributed Data Processing

*- Ramanand Singh, Enterprise Architect*

## Executive Summary

In order to effectively utilize increasing amounts of available data and in light of these recent trends, a few applications have emerged that automatically distribute data and processing over many separate commodity-class servers connected via a network, and that vastly prefer sequential disk operations over random disk seeks. Unlike some distributed systems these applications do not share memory or storage, an approach called Shared-*Nothing Architecture*. They are designed to handle individual machine failures automatically with no interruption to operations.

## Accumulo

Accumulo is a highly secured, scalable, distributed, open-source database modeled after Google's BigTable design concepts. It is built to store up to trillions of data elements organized in a manner that users can perform fast lookups. Accumulo supports flexible data schemas and scales horizontally across thousands of machines. It is an ideal choice for terabyte to petabyte-scale projects.

Accumulo is a distributed application that depends on Hadoop for storage and Zookeeper for configuration purposes. But unlike Hadoop, Accumulo is designed to store data in an organized fashion so users can quickly find the data they need or incrementally add to or update a data set. The basic architecture of Accumulo is shown here in the diagram below. A brief description for each of the essential components of the architecture has been provided in the document for reference purposes.

Accumulo can be used to support a wide variety of scalable applications including storing structured or semi-structured sparse and dynamic data, building rich text search capabilities, indexing geospatial or multi-dimensional data, storing or processing large graphs, and maintaining continuously updated summaries over raw events using server-side programming mechanisms. Instances of Accumulo have been known to run on over a thousand servers, storing over a petabyte of data, and trillions key-value pairs.

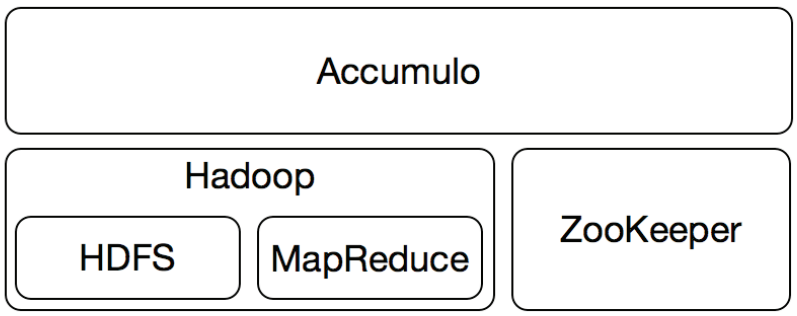


**Figure 1: Accumulo Architecture**

**Accumulo Data Model**: At the most basic level, Accumulo stores key-value pairs on disk, keeping the keys sorted at all times. This allows a user to look up the value of a particular key or range of keys very

quickly. Values are stored as byte arrays and Accumulo doesn't restrict the type or size of the values stored. Accumulo key structure is represented as shown below along with an example dataset.

| Key | | | | | Value |
|---|---|---|---|---|---|
| Row Id | Column | | | Timestamp | |
| | Family | Qualifier | Visibility | | |
| Joe Smith | Contact | Address | Billing | 201230 | 123 H street |
| Joe Smith | Contact | City | Billing | 201230 | Hagerstown |
| Joe smith | Contact | Phone | Billing | 201230 | 123-555-1212 |
| Joe Smith | Purchases | Shoes | Billing & inventory | 201231 | $79.99 |
| … | … | … | … | … | … |

## Hadoop

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly available service on top of a cluster of computers, each of which may be prone to failures.

Hadoop includes these modules:

- **Hadoop Common**: The common utilities that support the other Hadoop modules.
- **Hadoop Distributed File System**: A distributed file system that provides high-throughput access to application data.
- **Hadoop YARN**: A framework for job scheduling and cluster resource management.
- **Hadoop MapReduce**: A YARN-based system for parallel processing of large data sets.

## MapReduce

Hadoop MapReduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.

A MapReduce *job* usually splits the input data-set into independent chunks that are processed by the *map tasks* in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the *reduce tasks*. Typically both the input and the output of the job are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks.

Typically the compute nodes and the storage nodes are the same, that is, the MapReduce framework and the Hadoop Distributed File System (HDFS) are running on the same set of nodes. This configuration allows the framework to effectively schedule tasks on the nodes where data is already present, resulting in very high aggregate bandwidth across the cluster.

The MapReduce framework consists of a single master JobTracker and one slave TaskTracker per cluster-node. The master is responsible for scheduling the jobs' component tasks on the slaves, monitoring them and re-executing the failed tasks. The slaves execute the tasks as directed by the master.

## Zookeeper

Zookeeper is a highly available, highly consistent distributed application in which all the data is replicated on all the machines so that if one machine fails, clients reading from Zookeeper can quickly switch over to one of

the remaining machines. In practice Zookeeper instances tend to consist of 3 or 5 machines. Using an even number of Zookeeper hosts would provide less resilience than using one fewer machine.

Accumulo uses Zookeeper to store configuration and status information and to track changes in the cluster. Zookeeper is also used to help clients begin the process of locating the right servers for the data they seek.

It is very important to note that Accumulo stores persistent, essential information in Zookeeper. As a result, if the user deletes the Zookeeper data, related Accumulo instance will no longer run.[i]

---

[i] Author can be contacted at JavaDoctor @ gmail.com