Hadoop In Action

# Intro

Hadoop is an open source framework for writing and running distributed applications that process large amounts of data.

**Features**

**Accessible**—Hadoop runs on large clusters of commodity machines or on cloud computing services such as Amazon’s Elastic Compute Cloud (EC2 ).

**Robust**—Because it is intended to run on commodity hardware, Hadoop is architected with the assumption of frequent hardware malfunctions. It can gracefully handle most such failures.

**Scalable**—Hadoop scales linearly to handle larger data by adding more nodes to the cluster.

**Simple**—Hadoop allows users to quickly write efficient parallel code.

# Differences with other distributed systems

Unlike other distributed systems like SETI@Home that requires repeated transmissions of data between clients and servers, Hadoop focus on moving code to data.

The client sends only the MapReduce programs to be executed which are generally small. Data is broken up and distributed across the Hadoop Cluster and as much as possible computation takes place on the same machine where the piece of data resides.

# Differences with SQL

|  |  |
| --- | --- |
| **SQL** | **Hadoop** |
| For Structured data | Unstructured data |
| Scaling UP | Scaling out is easy |
| Relational Tables | Key/Value pairs. |
| Declarative queries- You query data by stating the result you want and let the database engine figure out how to derive it | Functional programming - specify the actual steps in processing the data, which is more analogous to an execution plan for a SQL engine . Under SQL you have query statements; under MapReduce you have scripts and codes. MapReduce allows you to process data in a more general fashion than SQL queries. For example, you can build complex statistical models from your data or reformat your image data. SQL is not well designed for such tasks. |
| Online transactions | Offline batch processing - best used as a write-once, read-many-times type of data store. In this aspect it’s similar to data warehouses in the SQL world. |
|  |  |

## MapReduce Intro

MapReduce programs are executed in two main phases, called mapping and reducing .Each phase is defined by a data processing function, and these functions are called mapper and reducer, respectively. In the mapping phase, MapReduce takes the input data and feeds each data element to the mapper. In the reducing phase, the reducer processes all the outputs from the mapper and arrives at a final result.

In simple terms, the mapper is meant to filter and transform the input into something that the reducer can aggregate over.

Partitioning and shuffling are common design patterns that go along with mapping and reducing. Unlike mapping and reducing, though, partitioning and shuffling are generic functionalities that are not too dependent on the particular data processing application.

In order for mapping, reducing, partitioning, and shuffling to seamlessly work together, we need to agree on a common structure for the data being processed. It should be flexible and powerful enough to handle most of the targeted data processing applications. MapReduce uses lists and (key/value) pairs as its main data primitives. The keys and values are often integers or strings but can also be dummy values to be ignored or complex object types. The map and reduce functions must obey the following constraint on the types of keys and values.

|  |  |  |
| --- | --- | --- |
|  | **Input** | **Output** |
| map | <k1,v1> | List(k2,v2) |
| reduce | <k2,list(v2)> | List(<k3,v3>) |





## Building Blocks of Hadoop

On a fully configured server “running Hadoop” means running a set of daemons, or resident programs, on different servers in our network. These daemons have specific roles; some exists only on one server some on multiple servers. These includes

1. NameNode
2. DataNode
3. Secondary Name Node
4. Job Tracker
5. Task Tracker

### Name Node

Hadoop employs a master/slave architecture for both distributed storage and distributed computation. The distributed storage system is called the Hadoop File System , or HDFS. The NameNode is the master of HDFS that directs the slave DataNode daemons to perform the low-level I/O tasks.

The NameNode is the bookkeeper of HDFS; it keeps track of how your files are broken down into file blocks, which nodes store those blocks, and the overall health of the distributed filesystem.

The function of the NameNode is memory and I/O intensive. As such, the server hosting the NameNode typically doesn’t store any user data or perform any computations for a MapReduce program to lower the workload on the machine. This means that the NameNode server doesn’t double as a DataNode or a TaskTracker.

There is unfortunately a negative aspect to the importance of the NameNode—it’s a single point of failure of your Hadoop cluster.

### Data Node

Each slave machine in your cluster will host a DataNode daemon to perform the grunt work of the distributed filesystem—reading and writing HDFS blocks to actual files on the local filesystem.

When you want to read or write a HDFS file, the file is broken into blocks and the NameNode will tell your client which DataNode each block resides in.

Your client communicates directly with the DataNode daemons to process the local files corresponding to the blocks. Furthermore, a DataNode may communicate with other DataNodes to replicate its data blocks for redundancy.

DataNodes are constantly reporting to the NameNode. Upon initialization, each of the DataNodes informs the NameNode of the blocks it’s currently storing. After this mapping is complete, the DataNodes continually poll the NameNode to provide information regarding local changes as well as receive instructions to create, move, or delete blocks from the local disk.

