ProjectDescription

Introduction

Hadoop was originally built as infrastructure for the Nutch project, which crawls the web and builds a search engine index for the crawled pages. It provides a distributed filesystem (HDFS) that can store data across thousands of servers, and a means of running work (Map/Reduce jobs) across those machines, running the work near the data.

Hadoop Map/Reduce

Programming model and execution framework

Map/Reduce is a programming paradigm that expresses a large distributed computation as a sequence of distributed operations on data sets of key/value pairs. The Hadoop Map/Reduce framework harnesses a cluster of machines and executes user defined Map/Reduce jobs across the nodes in the cluster. A Map/Reduce computation has two phases, a *map* phase and a *reduce* phase. The input to the computation is a data set of key/value pairs.

In the map phase, the framework splits the input data set into a large number of fragments and assigns each fragment to a *map task*. The framework also distributes the many map tasks across the cluster of nodes on which it operates. Each map task consumes key/value pairs from its assigned fragment and produces a set of intermediate key/value pairs. For each input key/value pair *(K,V)*, the map task invokes a user defined *map function* that transmutes the input into a different key/value pair *(K',V')*.

Following the map phase the framework sorts the intermediate data set by key and produces a set of *(K',V'*)* tuples so that all the values associated with a particular key appear together. It also partitions the set of tuples into a number of fragments equal to the number of reduce tasks.

In the reduce phase, each *reduce task* consumes the fragment of *(K',V'*)* tuples assigned to it. For each such tuple it invokes a user-defined *reduce function* that transmutes the tuple into an output key/value pair *(K,V)*. Once again, the framework distributes the many reduce tasks across the cluster of nodes and deals with shipping the appropriate fragment of intermediate data to each reduce task.

Tasks in each phase are executed in a fault-tolerant manner, if node(s) fail in the middle of a computation the tasks assigned to them are re-distributed among the remaining nodes. Having many map and reduce tasks enables good load balancing and allows failed tasks to be re-run with small runtime overhead.

Architecture

The Hadoop Map/Reduce framework has a master/slave architecture. It has a single master server or *jobtracker* and several slave servers or *tasktrackers*, one per node in the cluster. The *jobtracker* is the point of interaction between users and the framework. Users submit map/reduce jobs to the *jobtracker*, which puts them in a queue of pending jobs and executes them on a first-come/first-served basis. The *jobtracker* manages the assignment of map and reduce tasks to the *tasktrackers*. The *tasktrackers* execute tasks upon instruction from the jobtracker and also handle data motion between the map and reduce phases.

Hadoop DFS

Hadoop's Distributed File System is designed to reliably store very large files across machines in a large cluster. It is inspired by the Google File System. Hadoop DFS stores each file as a sequence of blocks, all blocks in a file except the last block are the same size. Blocks belonging to a file are replicated for fault tolerance. The block size and replication factor are configurable per file. Files in HDFS are "write once" and have strictly one writer at any time.

Architecture

Like Hadoop Map/Reduce, HDFS follows a master/slave architecture. An HDFS installation consists of a single *Namenode*, a master server that manages the filesystem namespace and regulates access to files by clients. In addition, there are a number of *Datanodes*, one per node in the cluster, which manage storage attached to the nodes that they run on. The *Namenode* makes filesystem namespace operations like opening, closing, renaming etc. of files and directories available via an RPC interface. It also determines the mapping of blocks to *Datanodes*. The *Datanodes* are responsible for serving read and write requests from filesystem clients, they also perform block creation, deletion, and replication upon instruction from the *Namenode*.

Scalability

The intent is to scale Hadoop up to handling thousand of computers. Some real Hadoop clusters are described on the PoweredBy page.