Big Data Frameworks CSE6001

Prof. Ramesh Ragala

August 13, 2019

Distributed File System

- Mostly computing is done on a single processor, with its main memory, cache, and local disk → compute node.
- Applications that called for Parallel processing such as large scientific calculations were executing on special-purpose parallel computers with many processors and specialized hardware.
- The prevalence of large-scale Web services has caused more and more computing to be done on thousands of computing nodes operating more or less independently.
- Computing clusters: Large collections of commodity hardware, including conventional processors ("compute nodes") connected by Ethernet cables or inexpensive switches.
- Moores law suited → building bigger and bigger servers is no longer necessarily the best solution to large-scale problems. → An alternative that has gained popularity is to tie together many low-end/commodity machines together as a single functional distributed system.
- Distributed system → Scale-Out

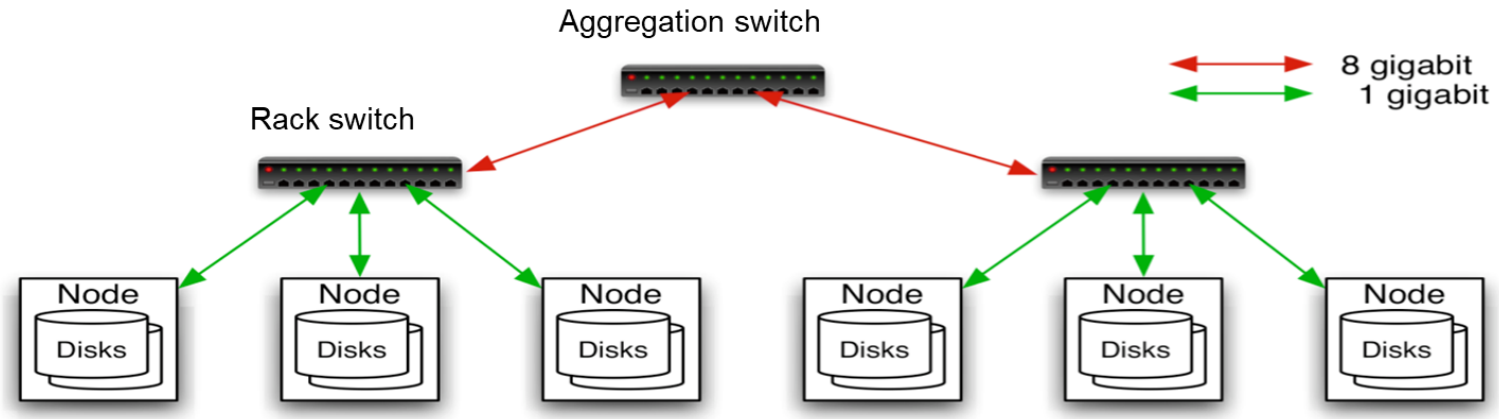
SIMPLE SCENARIO

A high-end machine with four I/O channels each having a throughput of 100 MB/sec will require three hours to read a 4 TB data set! With Hadoop, this same data set will be divided into smaller (typically 64 MB) blocks that are spread among many machines in the cluster via the Hadoop Distributed File System (HDFS). With a modest degree of replication, the cluster machines can read the data set in parallel and provide a much higher throughput. And such a cluster of commodity machines turns out to be cheaper than one high-end server

Distributed File System

- The compute nodes are **commodity hardware**, which greatly reduces the cost compared with special-purpose parallel machines.
- New computing facilities have given support to a new generation of programming systems. $\rightarrow$ the power of parallelism.
- problem: At the same time avoid the reliability problems that arise when the computing hardware consists of thousands of independent components, any of which could fail at any time.
- Design of specialized file system that have been developed to take advantage

Typical 2 Level Architecture



- It involves the use of large numbers of available computing node for parallel computing, to get the greatest amount of useful computation at low cost
- The nodes in above commodity hardware are PCs
- 30 - 40 nodes per rack
- Uplink from rack is 3-4 gigabit
- Rack-internal is 1 gigabit

- **Physical Organization of Computing Nodes:**
  - new parallel-computing architecture → sometimes called as "Cluster Computing".
  - Compute nodes are stored on racks, perhaps 864 on a rack.
  - The nodes on a single rack are connected by a network → Gigabit Ethernet
  - There can be many racks of compute nodes, and racks are connected by another level of network or a switch.
  - The bandwidth of inter-rack communication is somewhat greater than the intrarack Ethernet,
- problems:
  - Failure of Computing nodes → loss of single node
  - Failure of Interconnection networks → loss of entire rack
- Difficult to restart or abort the computation for every component failure.
- Solutions to this problem:
  - Files must be stored redundantly
  - Computation must be divided into tasks.

Distributed File System

- To exploit cluster computing, files must look and behave somewhat differently from the conventional file systems found on single computers. $\rightarrow$ distributed file system or DFS.
- DFS is a new file system, is typically as follows:
  - Files can be enormous, possibly a terabyte in size
  - Files are rarely updated
  - Files are divided into chunks and these chunks are replicated. Both the chunk size and the degree of replication can be decided by the user.
  - Meta data about the chunks are available in master node.
- There are several distributed file systems of type are used in practice
  - GFS $\rightarrow$ Google File System
  - HDFS $\rightarrow$ Hadoop Distributed File System
  - CloudStore $\rightarrow$ an open-source DFS originally developed by Kosmix.

Hadoop -Introduction

- An Open source software framework (Apache Project)
- In this Framework, users can write and run the distributed applications that process massive dataset.
- what makes it especially useful
  - Scalable: It can reliably store and process petabytes.
  - Economical: It distributes the data and processing across clusters of commonly available computers (in thousands).
  - Efficient: By distributing the data, it can process it in parallel on the nodes where the data is located. $\rightarrow$ Moving the processing to the data
  - Robust and Reliable: Hadoop is architected with the assumption of frequent hardware malfunctions. It automatically maintains multiple copies of data and automatically redeploys computing tasks based on failures.
  - Simple and Accessible: It runs on large clusters of commodity machines or on cloud computing services

Design principle of Hadoop

- **1.Leveraging commodity server hardware**
- **2.Moving the processing to the data**
  - Hadoop design principle suggests that instead of separating the processing from the data storage, move the processing to where the data is stored.
  - This allows for faster, more efficient processing of the data
  - Application queries are not required to access a remote disk in order to complete their task.
  - This improves performance and decreases complexity.
- **3.Utilizing local hard drives**
  - Hadoop performs and scales better with local hard drives and can be configured to use Just a Bunch of Disks (JBOD) rather than Redundant Array of Independent Disks (RAID).
  - By default, Hadoop creates three copies of the data and replicates the data across multiple servers and drives.
  - By using large spinning disk drives, Hadoop takes into account disk failure and resubmits existing queries without interruption.
  - Hadoop handles the mirroring or replication of the data as its written across the cluster, thereby eliminating the need for RAID

Design principle of Hadoop

- Hadoop has two main components
  - 1. HDFS
    - It is a file system
    - Used to store very large files with streaming data access patterns, running on cluster of commodity Hardware.
  - 2. MapReduce
    - MapReduce is a programming model for data processing.
    - Hadoop can run MapReduce programs written in various languages;

Comparison with Other Systems

- Due to the unique properties of Hadoop, it is apart from other systems.
- **Hadoop vs. Relational Database Management Systems**
  - Why can't we use databases with lots of disks to do large-scale analysis? Why is Hadoop needed?
  - Solution: seeking time is improving more slowly than transfer rate
  - Seeking $\rightarrow$ the process of moving the disk's head to a particular place on the disk to read or write data.
  - The seek time characterize the latency of disk operation, whereas the transfer rate corresponds to the disk's bandwidth.
  - To update a small proportion of records in a database, B-Tree works well. For updating of majority of database, B-Tree is less efficient than MapReduce. Here MapReduce uses sort or merge to rebuild the database.
  - MapReduce is a good fit for problems that need to analyze the whole dataset in a batch fashion
  - Another difference between Hadoop and an RDBMS is the amount of structure in the datasets on which they operate

Design principle of HDFS

- **Very Large Files:**
  - There are Hadoop clusters running to store petabytes of data
- **Streaming data access:**
  - HDFS idea $\rightarrow$ Processing patterns is "Write-Once, Read Many-Times"
  - A dataset is stored from source and then various analysis are performed on that dataset over period of time
  - The time to read the dataset is more important than the latency in reading the first record.
- **Commodity hardware:**
  - It is designed to run on clusters of commodity hardware $\rightarrow$ chance of node failure across the cluster is high
  - HDFS is designed to carry on working without a noticeable interruption to the user in the face of such failure

- **HDFS Limitations:**
- **Low-latency data access:**
  - Applications that require low-latency access to data, in the tens of milliseconds range.
  - HDFS is optimized for delivering a high throughput of data, so it may be expense of latency
- **Lots of small files:**
  - The metadata information stored in master node. This has limitation based on the memory of masternode. So many small files information may be stored in masternode.
  - Rule $\rightarrow$ Each file, Directory and block takes approximately 150 bytes.
- **Multiple writers, arbitrary file modifications:**
  - Files in HDFS may be written to by a single writer.
  - Writes are always made at the end of the file, in append-only fashion
  - There is no support for multiple writers or for modifications at arbitrary offsets in the file.