Big Data Processing

Tessema Mengistu

mengistu@vt.edu

- Outline
  - Introduction to Big Data
  - MapReduce Programming Model
  - The Hadoop Framework

Introduction to Big Data

- Organizations are investing heavily in data and analytics services
  - Data driven
- There is an explosion of data in the current computing landscape
  - > 65% of the world population connected to the Internet
  - Web 2.0/3.0
  - IoT
  - . . .
- This explosion coincide with the availability of powerful compute resource with relatively cheaper cost
  - Cloud Computing

Introduction to Big Data

- Data pipeline
  - An infrastructure that supports data-driven decision
  - It basically involves:
    - Collection
      - Cleansing
    - Storage and processing
    - Make decisions based on the result

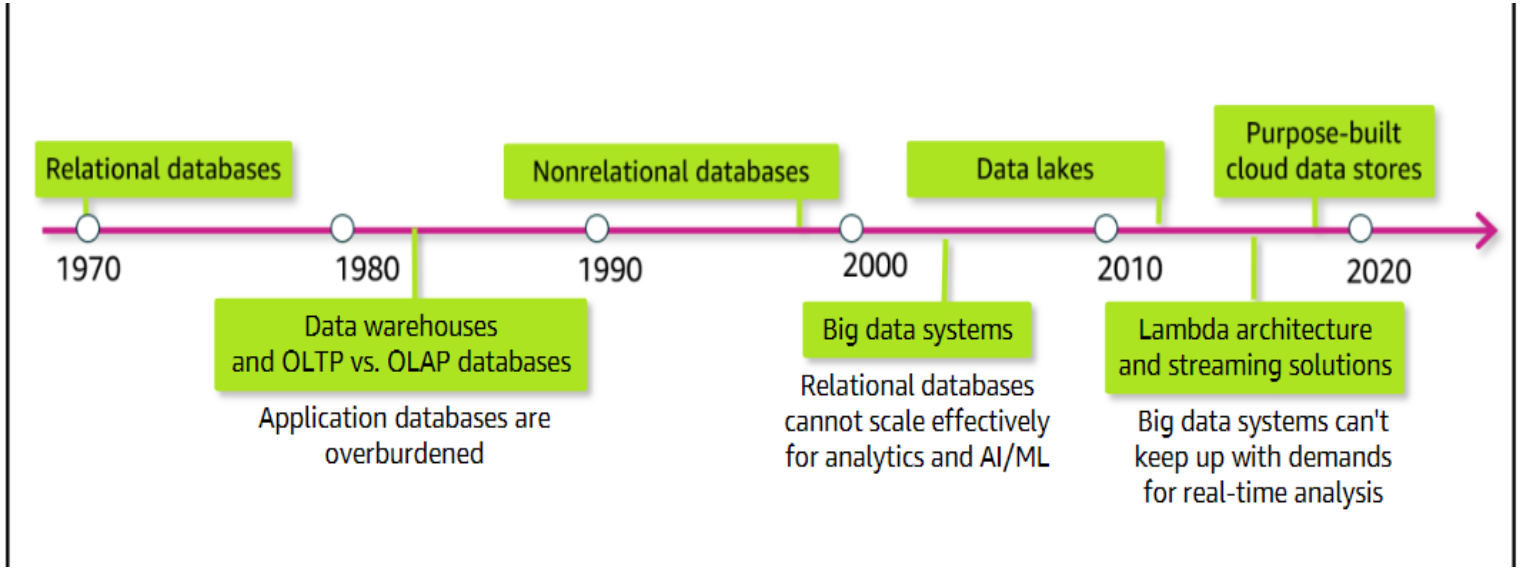
Introduction to Big Data

- Data can be generally:
  - Structured
    - Example: Relational databased
  - Semi-structured
    - Example: XML/JSON file
  - Unstructured
    - Example: collection of text files or sensor data
- The data explosion is unstructured data

Introduction to Big Data

- Organizations usually have large datasets collected from different sources
- Evolved through the years:
  - Relational DB → Data warehouses → non relational DB→ Big data → data lakes → purpose-built data stores

Introduction to Big Data



Introduction to Big Data

- Big Data
  - Extremely large and diverse collections of structured, unstructured, and semi-structured data that continues to grow exponentially over time
  - These datasets are large enough that traditional data processing software can't handle its storage and processing efficiently
    - Big data processing

Introduction to Big Data

- Big Data growth can be viewed as a three-dimensional phenomenon:
  1. Implies an increased **volume** of data
  2. Requires increased processing speed to process more data and produce more results – **velocity**
  3. Involves a diversity of data sources and data types - **variety**
- Additional three vs
  4. The accuracy or quality of the data – **veracity**
  5. The worth of the data – **value**
  6. The meaning of data changes constantly - **variability**

Introduction to Big Data

- Some of the enduring challenges posed by Big Data are:
  - Develop a scalable data infrastructures capable of responding promptly to timing constraints of an application
  - Develop effective means of accommodating diversity in data management systems
  - Support comprehensive end-to-end processing and knowledge extraction from data
  - Develop convenient interfaces for a layman involved in data collection and analysis
  - Quality of data
  - Compliance and security

Introduction to Big Data

- Big data processing can be:
  - Batch data processing
    - Involves querying large amount of data
      - Cold data - Infrequently accessed
  - Streaming data processing
    - Generated continuously by many sources
      - Example: IoT devices, sensors, financial data from trading floors, . . .

Introduction to Big Data

- Parallel Processing
  - Plays key role in big data processing
  - Different programming models:
    - Automatic parallelization
      - Parallelization made by the compiler
    - Writing of the parallel application from scratch
      - Example: MPI, OpenMP, CUDA
    - Using Parallel libraries
      - Example: Hadoop, GridGain
  - Flynn's taxonomy
  - Tasks can be:
    - Fine grained, Coarse grained, embarrassingly parallel

The MapReduce Programming Model

- MapReduce
  - Based on a very simple idea for parallel processing of data-intensive applications supporting arbitrarily divisible load sharing
  - Inspired by the **map** and the **reduce** primitives of the Lisp programming language
  - Developed by Google

The MapReduce Programming Model

- Processes:
  - split the data into blocks, assign each block to an instance/process and then run these instances in parallel - map
  - merge the partial results produced by individual instances - reduce
- The data is stored using the Google File System (GFS)

The MapReduce Programming Model

- Map function
  - The large data are traced and segregated as key/value pairs
  - The map function accepts the key/value pair and returns an intermediate key/value pair

$$map(key_1, value_1) \rightarrow List(Key_2, value_2)$$

The MapReduce Programming Model

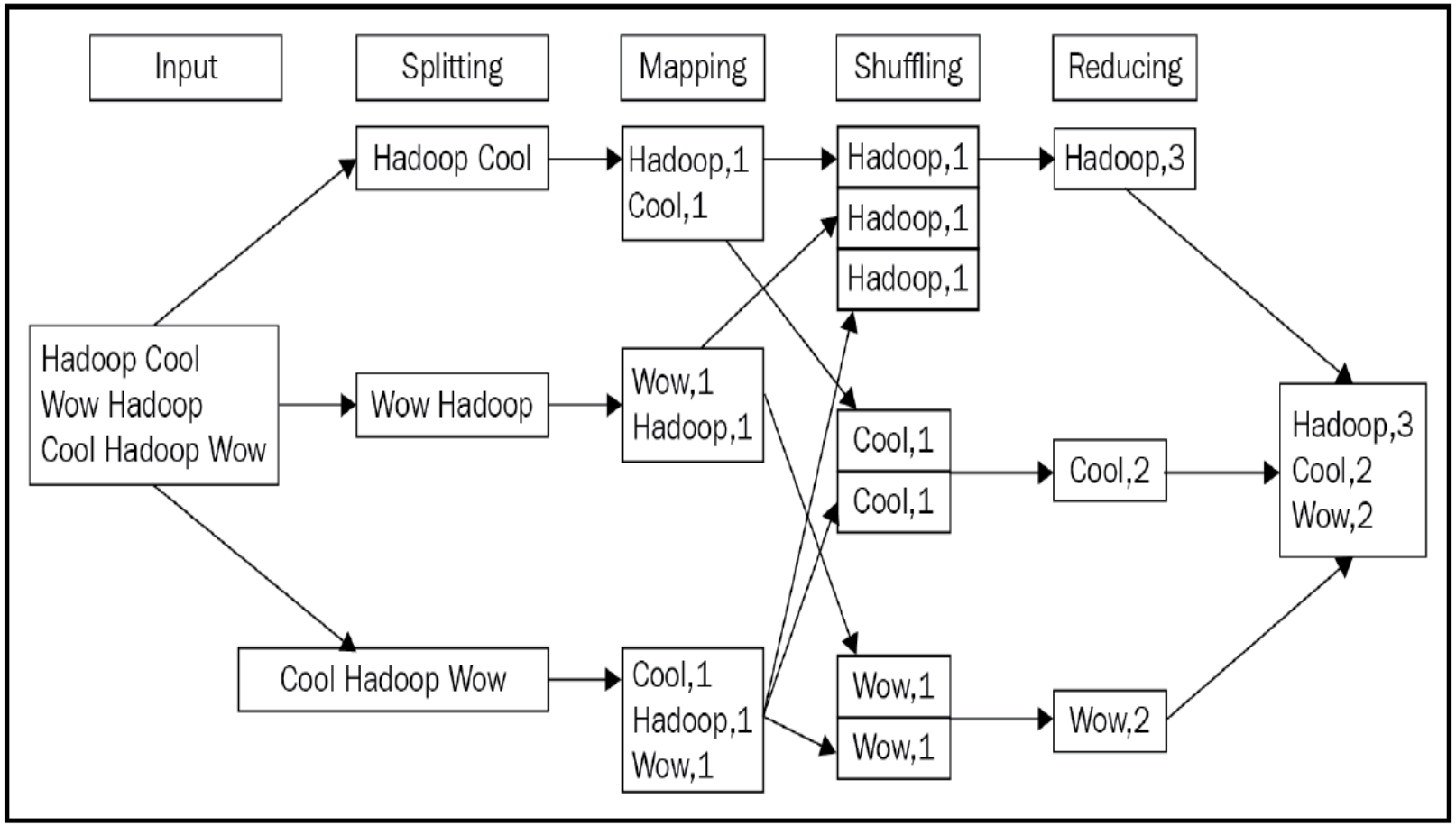
- Reduce function
  - Accepts the output of the map function (list of key/value pairs) as an input
  - A sort and merge operations are applied on key/value pairs

$$reduce(List\ (key_2, value_1) \rightarrow List(value_3)$$

The MapReduce Programming Model

```
map(String key,String value):
      // key:document name
     // value:document contents
   for each word w in value:
       EmitIntermediate(w,"1");
reduce(String key, Iterator values):
       // key:a word
       // values:a list of counts
   int result= 0;
   for each v in values:
       result+=ParseInt(v);
   Emit(AsString(result));
```


Source: https://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf



Source: Alla, Sridhar. Big Data Analytics with Hadoop 3 . . .

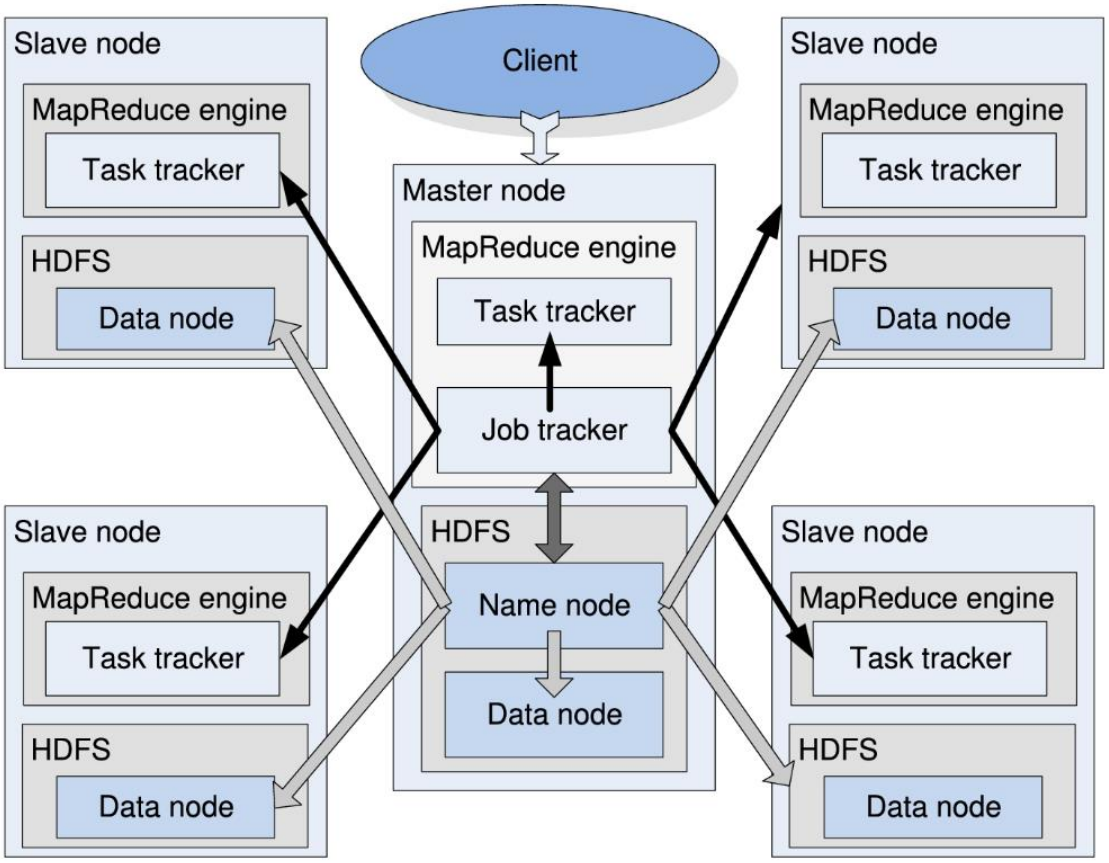
The Hadoop Framework

- Apache Hadoop
  - An open-source software framework for distributed storage and distributed processing based on the MapReduce programming model
  - Java-based
  - Supports distributed applications handling extremely large volumes of data

The Hadoop Framework

- The Apache Hadoop framework has the following modules:
  - Common
    - Contains libraries and utilities needed by all Hadoop modules
  - Distributed File System (HDFS)
    - A distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster
  - YARN – Yet Another Resource Negotiator
    - A resource-management platform responsible for managing computing resources in clusters and using them for scheduling of users' applications
  - MapReduce
    - An implementation of the MapReduce programming model

The Hadoop Framework



Source: Cloud Computing, 2nd Edition Dan C. Marinescu

The Hadoop Framework

- Distributed File System (HDFS)
  - Sits on top of the native filesystem
  - Divides the files into blocks (64 MB, 128 MB, …)
    - Distributed processing
  - HDFS has two main components:
    - The NameNode
    - The DataNode

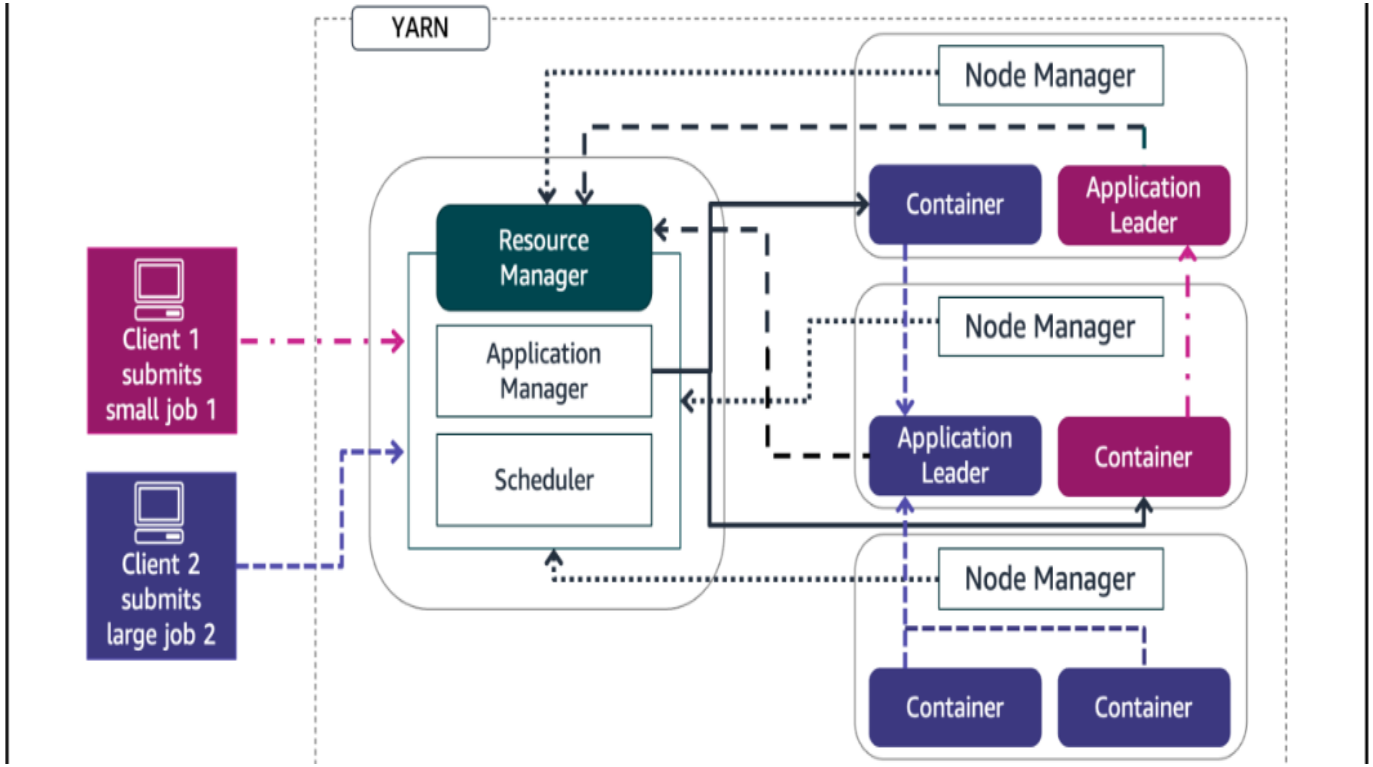
The Hadoop Framework

- The NameNode
  - Contains all the metadata of all content of the filesystem:
    - Filenames, file permissions, and the location of each block of each file
    - The most important machine in HDFS
- DataNodes
  - Connect to the NameNode and store the blocks within HDFS
  - They rely on the NameNode for all metadata information regarding the content in the filesystem

The Hadoop Framework

- YARN
  - Yet Another Resource Negotiator
  - A resource management system supplying CPU cycles, memory, and other resources needed by a single job or to a DAG of MapReduce applications
  - Has the following components:
    - Resource Manager
      - Scheduler and Application Manager components
    - Node Manager
    - Application leader
    - Containers

The Hadoop Framework



The Hadoop Framework

- Advantages
  - Scalability
  - Fault tolerance
- Limitations of Hadoop
  - Slower due to the latency of disk I/O
  - Difficult to handle interactive analytical tasks
  - Does not have many tools for:
    - Data Management and governance
    - Data quality and standardization
  - Security

The Hadoop Ecosystem

- Hive
  - An open-source system for **data-warehousing** supporting queries expressed in an SQL-like declarative language
    - Hive Query Language (Hive QL)
      - Must be implemented in the MapReduce Java API
  - Runs over Hadoop
  - Can creates a virtual table over S3 or HDFS
    - Schema-on-read

The Hadoop Ecosystem

- Presto (or PrestoDB)
  - An open source, distributed, **in-memory** SQL query engine, designed from the ground up for fast analytic queries against data of any size
  - Runs on Hadoop
  - Supports:
    - Non-relational sources
      - The Hadoop Distributed File System (HDFS) MongoDB, and HBase
    - Relational data sources such as
      - MySQL, PostgreSQL

The Hadoop Ecosystem

- Apache Spark.
  - General-purpose distributed processing framework used for big data workloads
    - Opensource **in-memory** distributed data processing
  - Has the following components:
    - Spark Core
      - Responsible for memory management, fault tolerance, scheduling and management of jobs, . . .
    - Spark SQL
      - Distributed query engine
    - Spark Graphx
      - Distributed graph processing framework run over spark
    - Spark Streaming
      - Real-time streaming analytics
    - Spark MLlib
      - A library of algorithms for ML

The Hadoop Ecosystem

- HBase
  - A non-relational or NoSQL database in the Hadoop ecosystem
  - Opensource and written in Java
  - HBase has master and region servers
  - Uses Zookeeper to provide centralized high-performance coordination between nodes or region servers
  - Compaction
    - HBase keeps track of the changes as change files and then merges them periodically
    - Can be major or minor

The Hadoop Ecosystem

- Apache Flink
  - Streaming data engine that processes real-time stream data for big data applications
  - Supports batch and streaming APIs and some SQL
- Impala
  - A query engine exploiting a shared-nothing parallel database architecture
  - Written in C++ and Java and designed to use standard Hadoop components (HDFS, HBase, Metastore, YARN)

References

- MapReduce: Simplified Data Processing on Large Clusters (2004)
- Spark: Cluster Computing with Working Sets (2010)
- Alla, Sridhar. Big Data Analytics with Hadoop 3 : Build Highly Effective Analytics Solutions to Gain Valuable Insight into Your Big Data, Packt Publishing, Limited, 2018.