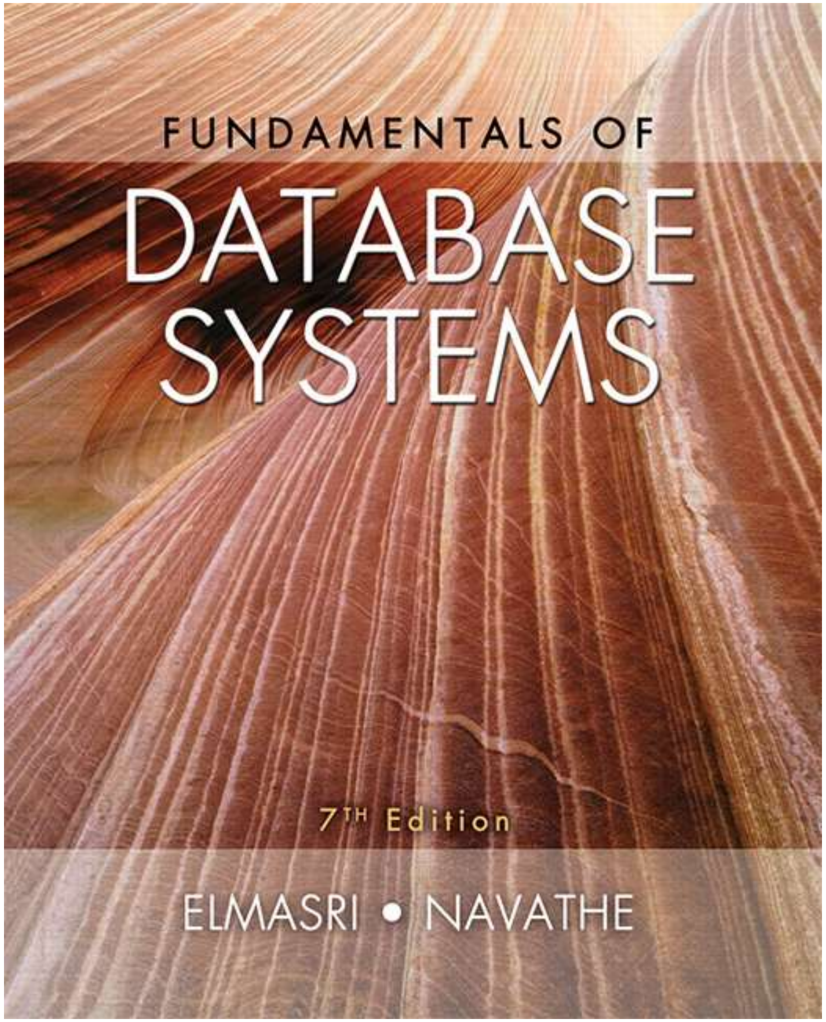
FUNDAMENTALS OF
DATABASE
SYSTEMS
7TH Edition
ELMASRI • NAVATHE

CHAPTER 25

Big Data Technologies Based on MapReduce and Hadoop

Introduction

- Phenomenal growth in data generation
  - Social media
  - Sensors
  - Communications networks and satellite imagery
  - User-specific business data
- "Big data" refers to massive amounts of data
  - Exceeds the typical reach of a DBMS
- Big data analytics

25.1 What is Big Data?

- Big data ranges from terabytes (10^{12} bytes) or petabytes (10^{15} bytes) to exobytes (10^{18} bytes)
- Volume
  - Refers to size of data managed by the system
- Velocity
  - Speed of data creation, ingestion, and processing
- Variety
  - Refers to type of data source
  - Structured, unstructured

What is Big Data? (cont'd.)

- Veracity
  - Credibility of the source
  - Suitability of data for the target audience
  - Evaluated through quality testing or credibility analysis

25.2 Introduction to MapReduce and Hadoop

- Core components of Hadoop
  - MapReduce programming paradigm
  - Hadoop Distributed File System (HDFS)
- Hadoop originated from quest for open source search engine
  - Developed by Cutting and Carafella in 2004
  - Cutting joined Yahoo in 2006
  - Yahoo spun off Hadoop-centered company in 2011
  - Tremendous growth

Introduction to MapReduce and Hadoop (cont'd.)

- MapReduce
  - Fault-tolerant implementation and runtime environment
  - Developed by Dean and Ghemawat at Google in 2004
  - Programming style: map and reduce tasks
    - Automatically parallelized and executed on large clusters of commodity hardware
  - Allows programmers to analyze very large datasets
  - Underlying data model assumed: key-value pair

The MapReduce Programming Model

- Map
  - Generic function that takes a key of type K1 and value of type V1
  - Returns a list of key-value pairs of type K2 and V2
- Reduce
  - Generic function that takes a key of type K2 and a list of values V2 and returns pairs of type (K3, V3)
- Outputs from the map function must match the input type of the reduce function

The MapReduce Programming Model (cont'd.)

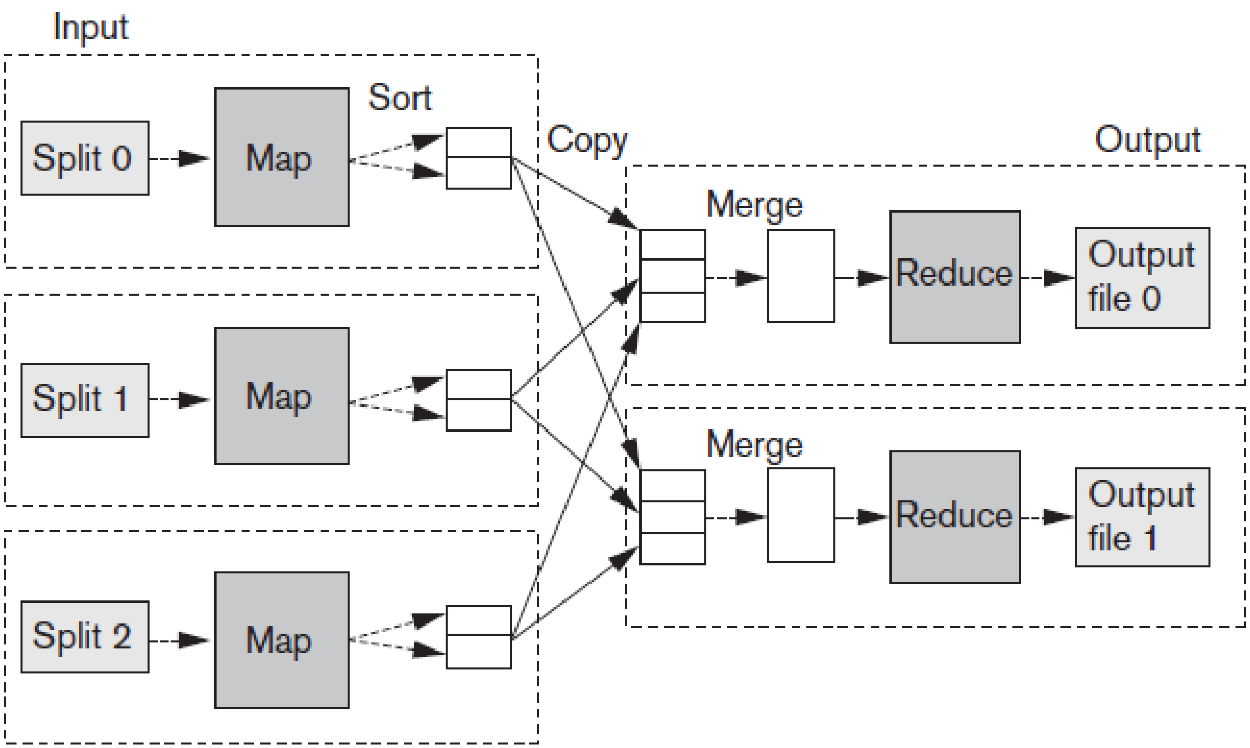

Figure 25.1 Overview of MapReduce execution (Adapted from T. White, 2012)

The MapReduce Programming Model (cont'd.)

- MapReduce example
  - Make a list of frequencies of words in a document
  - Pseudocode

```
Map (String key, String value):
    for each word w in value Emitintermediate (w, "1");

Reduce (String key, Iterator values) : // here the key is a word and values are
lists of its counts //
    Int result =0;
    For each v in values :
        result += Parseint (v);
    Emit (key, Asstring (result));
```

The MapReduce Programming Model (cont'd.)

- MapReduce example (cont'd.)
  - Actual MapReduce code

```
map[LongWritable,Text](key, value) : List[Text, LongWritable] = {
    String[] words = split(value)
    for(word : words) {
        context.out(Text(word), LongWritable(1))
    }
}
reduce[Text, Iterable[LongWritable]](key, values) : List[Text, LongWritable] = {
    LongWritable c = 0
    for( v : values) {
        c += v
    }
    context.out(key,c)
    }
```

The MapReduce Programming Model (cont'd.)

- Distributed grep
  - Looks for a given pattern in a file
  - Map function emits a line if it matches a supplied pattern
  - Reduce function is an identity function
- Reverse Web-link graph
  - Outputs (target URL, source URL) pairs for each link to a target page found in a source page

The MapReduce Programming Model (cont'd.)

- Inverted index
  - Builds an inverted index based on all words present in a document repository
  - Map function parses each document
    - Emits a sequence of (word, document_id) pairs
  - Reduce function takes all pairs for a given word and sorts them by document_id
- Job
  - Code for Map and Reduce phases, a set of artifacts, and properties

The MapReduce Programming Model (cont'd.)

- Hadoop releases
  - 1.x features
    - Continuation of the original code base
    - Additions include security, additional HDFS and MapReduce improvements
  - 2.x features
    - YARN (Yet Another Resource Navigator)
    - A new MR runtime that runs on top of YARN
    - Improved HDFS that supports federation and increased availability

25.3 Hadoop Distributed File System (HDFS)

- HDFS
  - File system component of Hadoop
  - Designed to run on a cluster of commodity hardware
  - Patterned after UNIX file system
  - Provides high-throughput access to large datasets
  - Stores metadata on NameNode server
  - Stores application data on DataNode servers
    - File content replicated on multiple DataNodes

Hadoop Distributed File System (cont'd.)

- HDFS design assumptions and goals
  - Hardware failure is the norm
  - Batch processing
  - Large datasets
  - Simple coherency model
- HDFS architecture
  - Master-slave
  - Decouples metadata from data operations
  - Replication provides reliability and high availability
  - Network traffic minimized

Hadoop Distributed File System (cont'd.)

- NameNode
  - Maintains image of the file system
    - i-nodes and corresponding block locations
  - Changes maintained in write-ahead commit log called Journal
- Secondary NameNodes
  - Checkpointing role or backup role
- DataNodes
  - Stores blocks in node's native file system
  - Periodically reports state to the NameNode

Hadoop Distributed File System (cont'd.)

- File I/O operations
  - Single-writer, multiple-reader model
  - Files cannot be updated, only appended
  - Write pipeline set up to minimize network utilization
- Block placement
  - Nodes of Hadoop cluster typically spread across many racks
    - Nodes on a rack share a switch

Hadoop Distributed File System (cont'd.)

- Replica management
  - NameNode tracks number of replicas and block location
    - Based on block reports
  - Replication priority queue contains blocks that need to be replicated
- HDFS scalability
  - Yahoo cluster achieved 14 petabytes, 4000 nodes, 15k clients, and 600 million files

The Hadoop Ecosystem

- Related projects with additional functionality
  - Pig and hive
    - Provides higher-level interface for working with Hadoop framework
  - Oozie
    - Service for scheduling and running workflows of jobs
  - Sqoop
    - Library and runtime environment for efficiently moving data between relational databases and HDFS

The Hadoop Ecosystem (cont'd.)

- Related projects with additional functionality (cont'd.)
  - HBase
    - Column-oriented key-value store that uses HDFS

25.4 MapReduce: Additional Details

- MapReduce runtime environment
  - JobTracker
    - Master process
    - Responsible for managing the life cycle of Jobs and scheduling Tasks on the cluster
  - TaskTracker
    - Slave process
    - Runs on all Worker nodes of the cluster

MapReduce: Additional Details (cont'd.)

- Overall flow of a MapReduce job
  - Job submission
  - Job initialization
  - Task assignment
  - Task execution
  - Job completion

MapReduce: Additional Details (cont'd.)

- Fault tolerance in MapReduce
  - Task failure
    - Runtime exception
    - Java virtual machine crash
    - No timely updates from the task process
  - TaskTracker failure
    - Crash or disconnection from JobTracker
    - Failed Tasks are rescheduled
  - JobTracker failure
    - Not a recoverable failure in Hadoop v1

MapReduce: Additional Details (cont'd.)

- The shuffle procedure
  - Reducers get all the rows for a given key together
  - Map phase
    - Background thread partitions buffered rows based on the number of Reducers in the job and the Partitioner
    - Rows sorted on key values
    - Comparator or Combiner may be used
  - Copy phase
  - Reduce phase

MapReduce: Additional Details (cont'd.)

- Job scheduling
  - JobTracker schedules work on cluster nodes
  - Fair Scheduler
    - Provides fast response time to small jobs in a Hadoop shared cluster
  - Capacity Scheduler
    - Geared to meet needs of large enterprise customers

MapReduce: Additional Details (cont'd.)

- Strategies for equi-joins in MapReduce environment
  - Sort-merge join
  - Map-side hash join
  - Partition join
  - Bucket joins
  - N-way map-side joins
  - Simple N-way joins

MapReduce: Additional Details (cont'd.)

- Apache Pig
  - Bridges the gap between declarative-style interfaces such as SQL, and rigid style required by MapReduce
  - Designed to solve problems such as ad hoc analyses of Web logs and clickstreams
  - Accommodates user-defined functions

MapReduce: Additional Details (cont'd.)

- Apache Hive
  - Provides a higher-level interface to Hadoop using SQL-like queries
  - Supports processing of aggregate analytical queries typical of data warehouses
  - Developed at Facebook

Hive System Architecture and Components

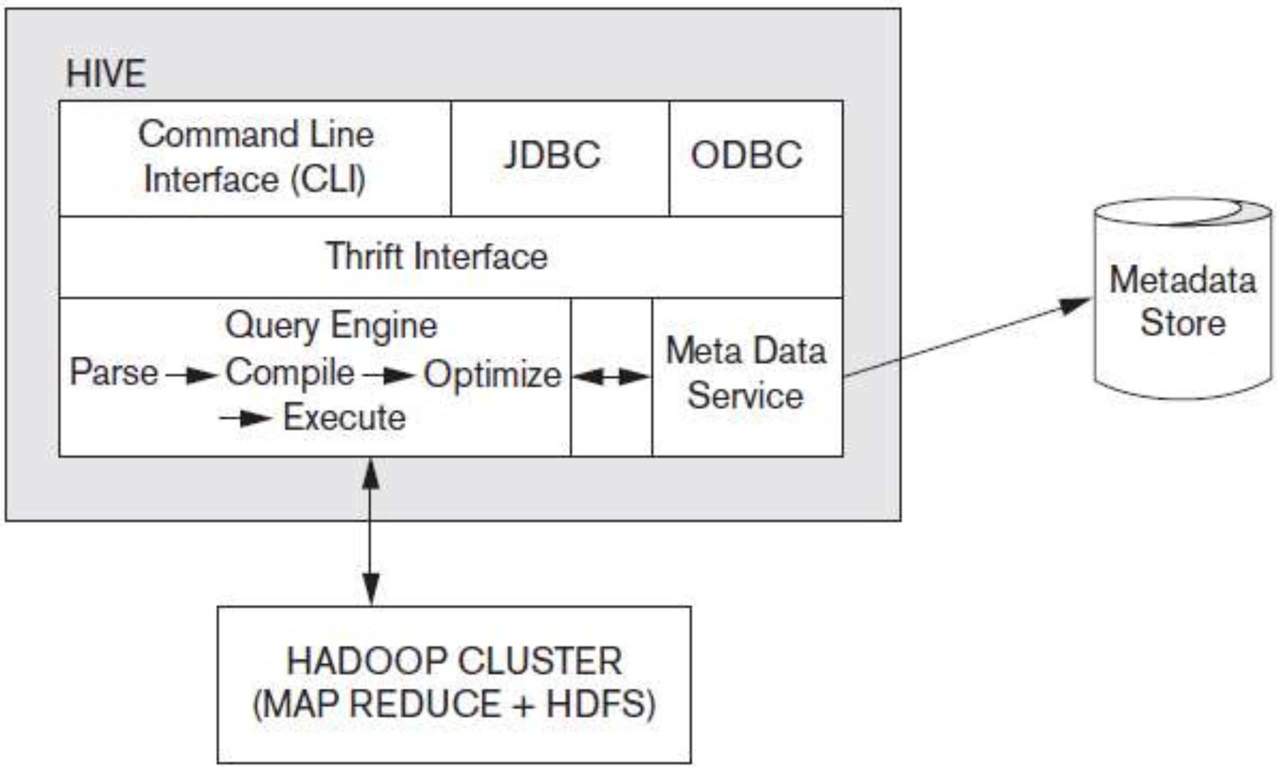

Figure 25.2 Hive system architecture and components

Advantages of the Hadoop/MapReduce Technology

- Disk seek rate a limiting factor when dealing with very large data sets
  - Limited by disk mechanical structure
- Transfer speed is an electronic feature and increasing steadily
- MapReduce processes large datasets in parallel
- MapReduce handles semistructured data and key-value datasets more easily
- Linear scalability

25.5 Hadoop v2 (Alias YARN)

- Reasons for developing Hadoop v2
  - JobTracker became a bottleneck
  - Cluster utilization less than desirable
  - Different types of applications did not fit into the MR model
  - Difficult to keep up with new open source versions of Hadoop

YARN Architecture

- Separates cluster resource management from Jobs management
- ResourceManager and NodeManager together form a platform for hosting any application on YARN
- ApplicationMasters send ResourceRequests to the ResourceManager which then responds with cluster Container leases
- NodeManager responsible for managing Containers on their nodes

Hadoop Version Schematics

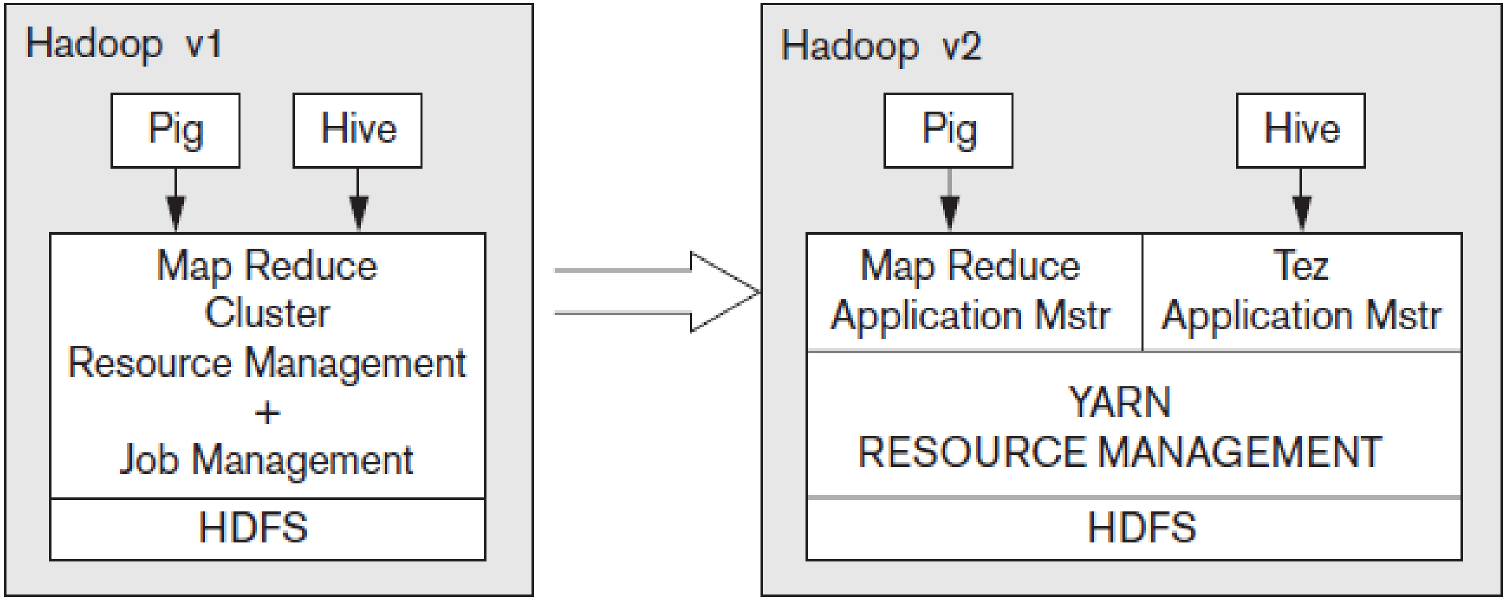

Figure 25.3 The Hadoop v1 vs. Hadoop v2 schematic

Other Frameworks on YARN

- Apache Tez
  - Extensible framework being developed at Hortonworks for building high-performance applications in YARN
- Apache Giraph
  - Open-source implementation of Google's Pregel system, a large-scale graph processing system used to calculate Page-Rank
- Hoya: HBase on YARN
  - More flexibility and improved cluster utilization

25.6 General Discussion

- Hadoop/MapReduce versus parallel RDBMS
  - 2009: performance of two approaches measured
    - Parallel database took longer to tune compared to MR
    - Performance of parallel database 3-6 times faster than MR
    - MR improvements since 2009
  - Hadoop has upfront cost advantage
    - Open source platform

General Discussion (cont'd.)

- MR able to handle semistructured datasets
- Support for unstructured data on the rise in RDBMSs
- Higher level language support
  - SQL for RDBMSs
  - Hive has incorporated SQL features in HiveQL
- Fault-tolerance: advantage of MR-based systems

General Discussion (cont'd.)

- Big data somewhat dependent on cloud technology
- Cloud model offers flexibility
  - Scaling out and scaling up
  - Distributed software and interchangeable resources
  - Unpredictable computing needs not uncommon in big data projects
  - High availability and durability

General Discussion (cont'd.)

- Data locality issues
  - Network load a concern
  - Self-configurable, locality-based data and virtual machine management framework proposed
    - Enables access of data locally
  - Caching techniques also improve performance
- Resource optimization
  - Challenge: optimize globally across all jobs in the cloud rather than per-job resource optimizations

General Discussion (cont'd.)

- YARN as a data service platform
  - Emerging trend: Hadoop as a data lake
    - Contains significant portion of enterprise data
    - Processing happens
  - Support for SQL in Hadoop is improving
- Apache Storm
  - Distributed scalable streaming engine
  - Allows users to process real-time data feeds
- Storm on YARN and SAS on YARN

General Discussion (cont'd.)

- Challenges faced by big data technologies
  - Heterogeneity of information
  - Privacy and confidentiality
  - Need for visualization and better human interfaces
  - Inconsistent and incomplete information

General Discussion (cont'd.)

- Building data solutions on Hadoop
  - May involve assembling ETL (extract, transform, load) processing, machine learning, graph processing, and/or report creation
  - Programming models and metadata not unified
    - Analytics application developers must try to integrate services into coherent solution
- Cluster a vast resource of main memory and flash storage
  - In-memory data engines
  - Spark platform from Databricks

25.7 Summary

- Big data technologies at the center of data analytics and machine learning applications
- MapReduce
- Hadoop Distributed File System
- Hadoop v2 or YARN
  - Generic data services platform
- MapReduce/Hadoop versus parallel DBMSs