EE 542 Lectures 8, 9 on Sept. 18, 20, 2017

MapReduce, Hadoop and Spark Programming

Kai Hwang
University of Southern California

EE 542 Home Work #2 (6%)

for Chapters 2, 5, 8 , Due date: Oct.9, 2017

Chapter 2: Prob. 2.4, Prob.2.5, Prob.2.13 (Lectures 10 and 11)

Chapter 5: Prob. 5.1, Prob.5.5, Prob.5.7 (Lectures 12, 13)

Chapter 8: Prob. 8.3, Prob. 8.4, Prob. 8.5, Prob.8.8, Prob. 8.11, Prob. 8.14 (lectures 8, 9)

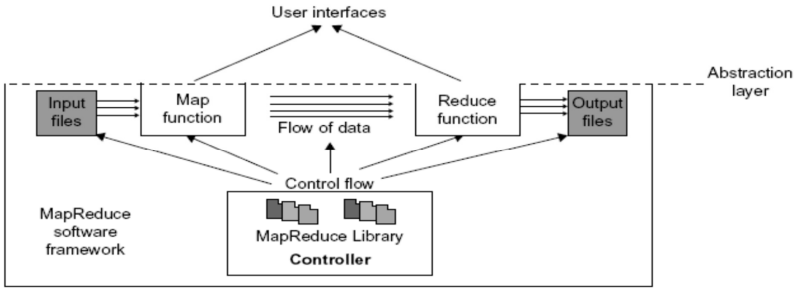
Note: You should work on those problems in Chapter 8 immediately after today's lecture. Some problems require Hadoop/Spark programming effort on the AWS cloud.

The Evolution of Scalable Parallel Computing:
From MapReduce to Hadoop and Spark in the last 10 years

- Google MapReduce Paradigm Written in C: from Search Engine to Google AppEngine
- Hadoop Library for MapReduce Programming in Java environments
- Extending Hadoop from MapReduce in Batch Processing Mode using distributed disks to Spark for In-Memory Processing in Streaming Mode over any DAG computing Paradigm

MapReduce : Scalable Data Processing on Large Clusters

- A web programming model for fast processing large datasets
- Applied in web-scale search and cloud computing applications
- Users specify a *map function* to generate intermediate key/value pairs
- Users use a *reduce function* to merge all intermediate values with the same key.



Batch Processing MapReduce :

Map: applies a programmer-supplied map function to each input data split (block)

- Runs on thousands of computers
- Provides new set of key-value pairs as intermediate values

Reduce: collapses values using another programmer-supplied function, called reduction, such as Max., Min, Average, dot product of two vectors, …….

Example : Counting the number of occurrences of each word in a large collection of documents

```
map(String key, String value):
  // key: document name
  // value: document contents
  for each word w in value:
    EmitIntermediate(w, "1");
```

The **map** function emits each word *w* plus an associated count of occurrences (just a "1" is recorded in this pseudo-code)

Example : Counting the number of occurrences of each word in a large collection of documents

```
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));
```

The **reduce** function sums together all counts emitted for a particular word

Table 8.1
Representative software libraries for big data processing on clouds

Name, Category, Language, Websites, Relevant Sections	Functionality, Applications, and User Community
Hadoop, compute engine, Java http://hadoop.apache.org/, Sec.8.2	Distributed processing of large data sets using Map-Reduce, mainly in batch processing on clouds or large clusters of servers.
Spark, compute with Java, Scala, Python, https://spark.apache.org/, Sec.8.3–8.5	General-purpose compute engine for both streaming and batch processing. Appeals to real-time applications on clouds or large websites.
HDFS, data storage, Java, C, http://hadoop.apache.org, Sec.8.2.4	A distributed file system that provides high-throughput access to application data.
Cassandra, data storage, C/C++, Java, Python, Ruby, http://cassandra.apache.org, Sec. 8.1.3	A distributed NoSQL for mission-critical data with linear scalability and proven fault tolerance on cloud infrastructure.
YARN, resource manager, Java and C, http://hadoop.apache.org, Sec.8.2.5	A new resource manager of Hadoop, which divides the JobTracker into two parts: resource management and job life-cycle management.

Mesos, resource scheduler developed by Google, http://www.google.com, Sec. 8.3	A resource scheduler developed by Google for scalable cluster computing on IaaS clouds.
Impala, query engine, Java, Python, http://www.cloudera.com, Sec. 8.1.3	Analytic database architected specifically to leverage the flexibility and scalability strengths of Hadoop.
Spark SQL, query engine, Python, Scala, Java, R, https://spark.apache.org/, Sec. 8.4	A query processing module in Spark library for structured or relational data sets data.
StormMQ, message system, Java, C++, http://stormmq.com/, sec. 8.1.3	A message queuing platform using Advanced Message Queuing Protocol (AMQP). It provides a hosted, on-premise or cloud solution for M2M apps.
Spark MLlib, Scala, Python, Java, R, https://spark.apache.org/mllib, Sec. 8.5	A machine learning module in Spark library for data analytics applications.
Mahout, data analytics, Scala, Java, http://mahout.apache.org/, Sec. 8.1	A software library for quickly creating scalable performant ML applications. It supports Hadoop and Spark platforms.
Weka, data mining, Java, Python, http://www.cs.waikato.ac.nz, Sec. 8.1	An ML software written in Java. It offers a collection of machine learning algorithms for data processing and mining tasks.
GraphX, graph processing, Scala, Python, Java, R, https://spark.apache.org/graphx, Sec. 8.5	A graph processing module in Spark library for social / media graph processing in streaming and real-time modes.

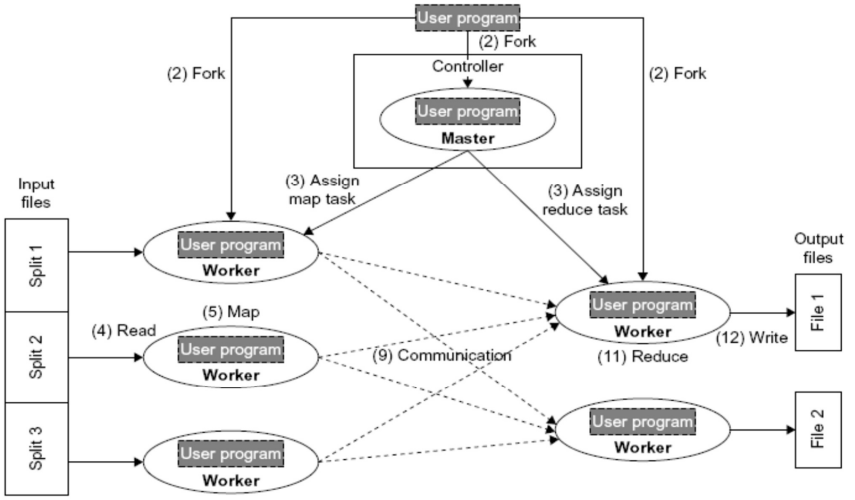


FIGURE 6.6

Control flow implementation of MapReduce.

(Courtesy of Yahoo! Pig Tutorial [54])

Linking the Map Workers and Reduce Workers by Key Matching in Partitioning Functions

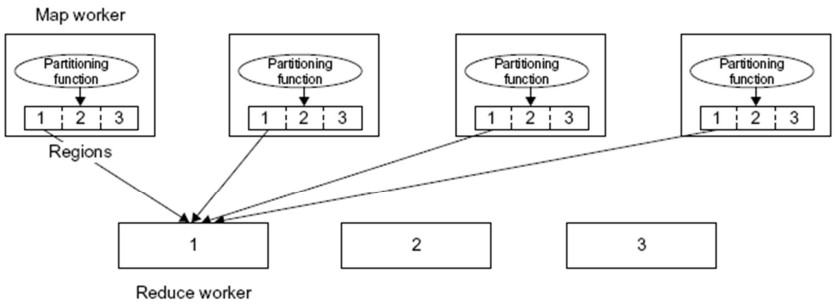
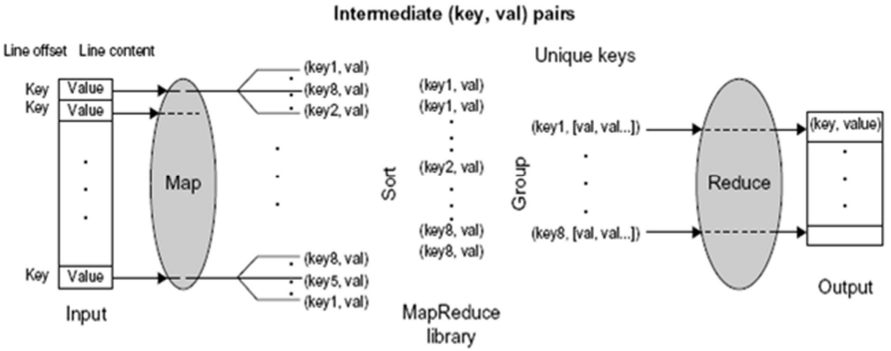


FIGURE 6.4

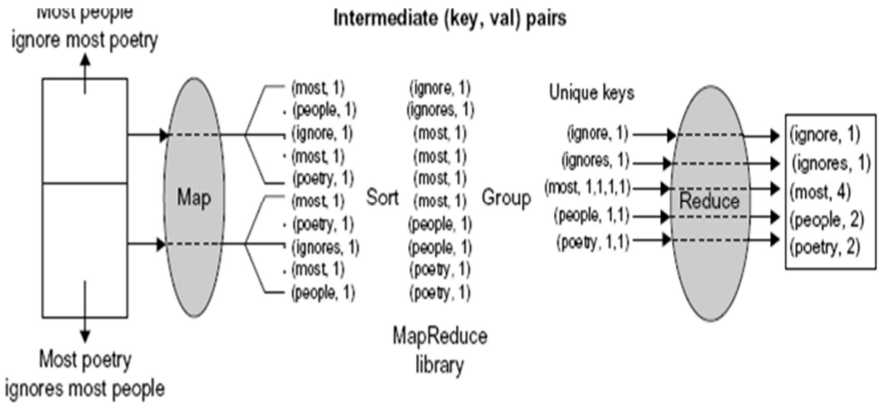
MapReduce *partitioning* function.

Logical Data Flow in 5 Processing Steps in MapReduce Process



(Key, Value) Pairs are generated by the Map function over multiple available Map Workers (VM instances). These pairs are then sorted and group based on key ordering. Different key-groups are then processed by multiple Reduce Workers in parallel.

A Word Counting Example on <Key, Count> Distribution



A Small Example

$$A \times B = \begin{bmatrix} a_{11}, & a_{12} \\ a_{21}, & a_{22} \end{bmatrix} \times \begin{bmatrix} b_{11}, & b_{12} \\ b_{21}, & b_{22} \end{bmatrix} = \begin{bmatrix} c_{11}, & c_{12} \\ c_{21}, & c_{22} \end{bmatrix} = C$$

c11 = a11 x b11 + a12 x b21 **c12 = a11 x b12 + a12 x b22**

c21 = a21 x b11 + a22 x b21 **c22 = a21 x b12 + a22 x b22**

1. **Map the first row of matrix A and entire matrix B to a Map Server 1**
2. **Map the second row of matrix A and entire matrix B to a Map Server 2**
3. **Four keys are used to identify 4 blocks of data processed**

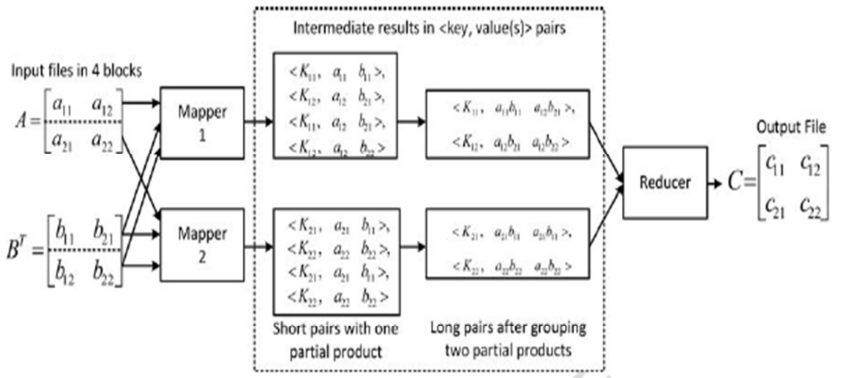
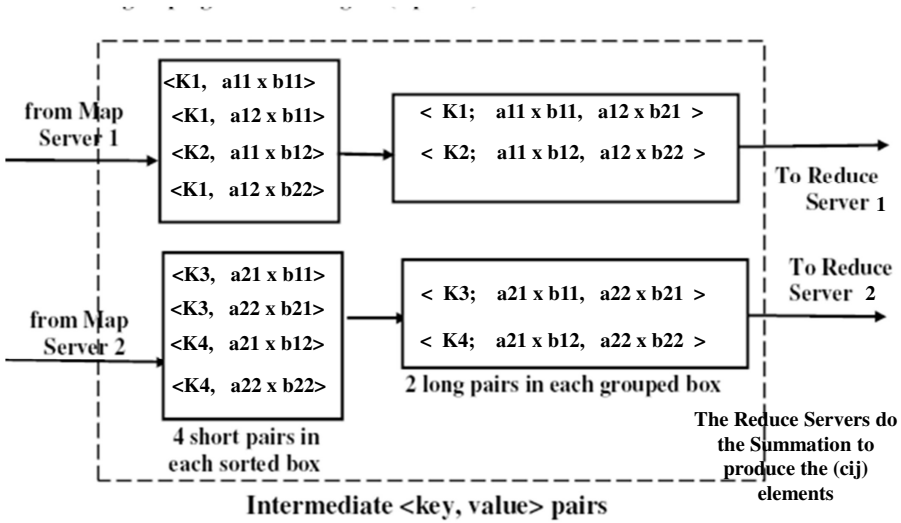


Figure 8.5
MapReduce for parallel matrix multiplication with <key, value> pairs showing as the intermediate results before and after grouping.



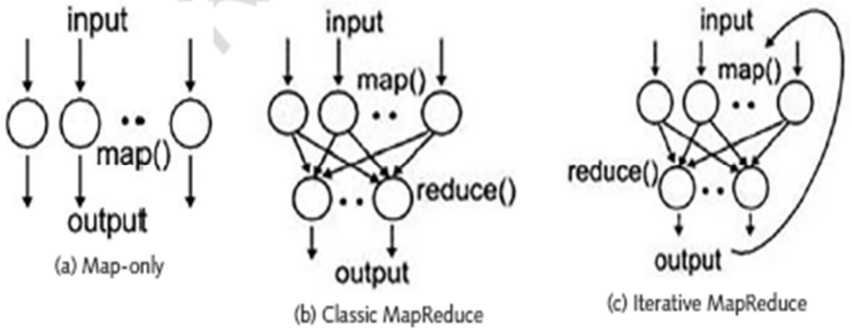


Figure 8.6
Comparison of Map-only, Classic MapReduce, and iterative MapReduce computations.

Hadoop : A software platform originally developed by a Yahoo! Group to enable users write and run applications over distributed data.

- **Scalable :** can easily scale to store and process petabytes of data in the Web space
- **Economical :** An open-source MapReduce minimizes the overheads in task spawning and massive data communication.
- **Efficient:** Processing data with high-degree of parallelism across a large number of commodity nodes
- **Reliable :** Automatically maintains multiple copies of data to facilitate redeployment of computing tasks on failures

Table 8.2
MapReduce and its variants in Hadoop and Twister

Features	Google MapReduce	Apache Hadoop	Twister
Execution Mode and Platform	Batch MapReduce on Linux cluster	Batch or real-time MapReduce on Linux cluster	Iterative MapReduce on EC@ or Linux clusters
Data Handling	GFS (Google File System)	HDFS (Hadoop Distributed File System)	Local disks and data management tools
Job Scheduling	Data locality	Data locality; rack aware; dynamic task scheduling	Data locality; static task partitions
HLL Support	Sawzall	Pig Latin	Pregel

High-Level Hadoop Master-Slave Architecture: The Yarn Scheduler, MapReduce and HDFS

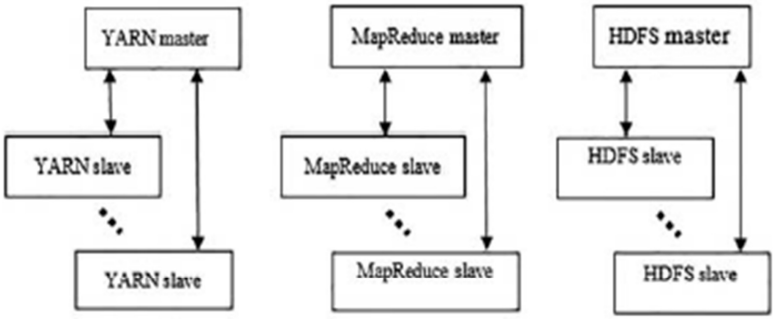
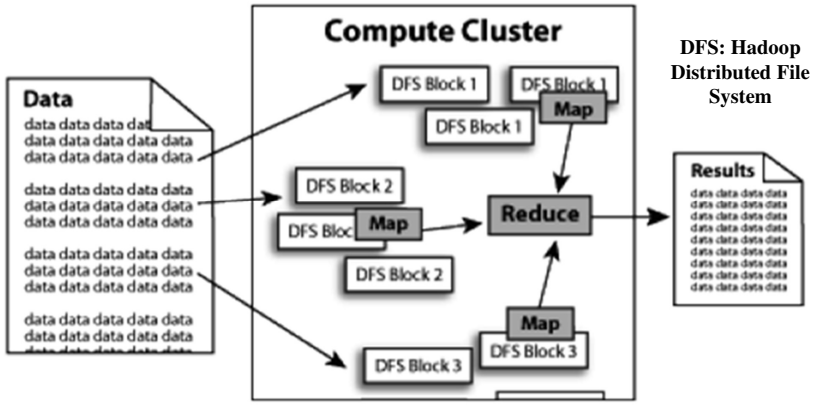


Figure 8.7
Three major functional modules in a Hadoop execution environment.

Building Blocks in Apache Hadoop

- **Hadoop Common:** The common utilities that support the other Hadoop modules
- **Hadoop Distributed File Syste, (HDFS):** A distributed file system that provides high-throughput access to application data.
- **Hadoop YARN:** A framework for job scheduling and cluster resource mangement
- **Hadoop MapReduce:** A Yarn-based system for parallel processing of large data sets

Apache Hadoop Architecture



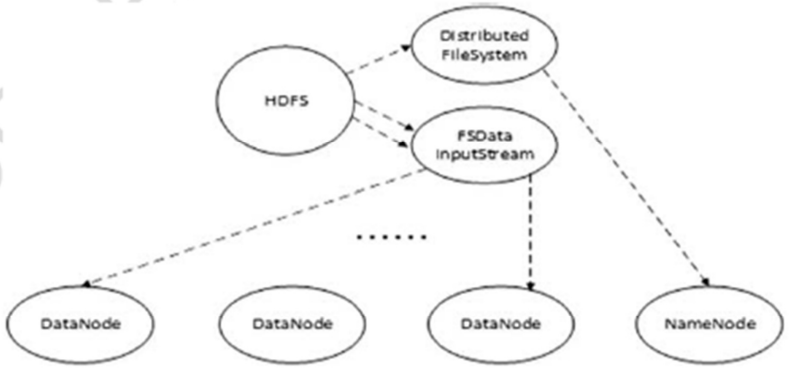


Figure 8.10
HDFS framework and its interaction with data nodes and name nodes.

An HDFS Client Communicating with the Master NameNode and Slave DataNode

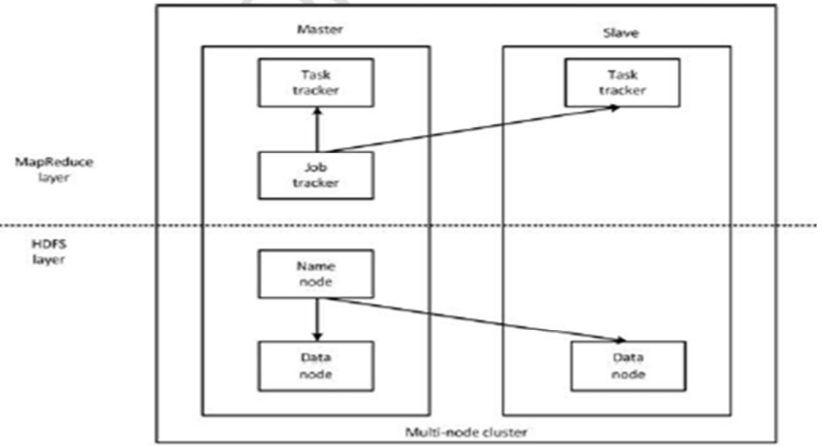


Figure 8.8
Key components in a Hadoop MapReduce engine interacting with the HDFS.

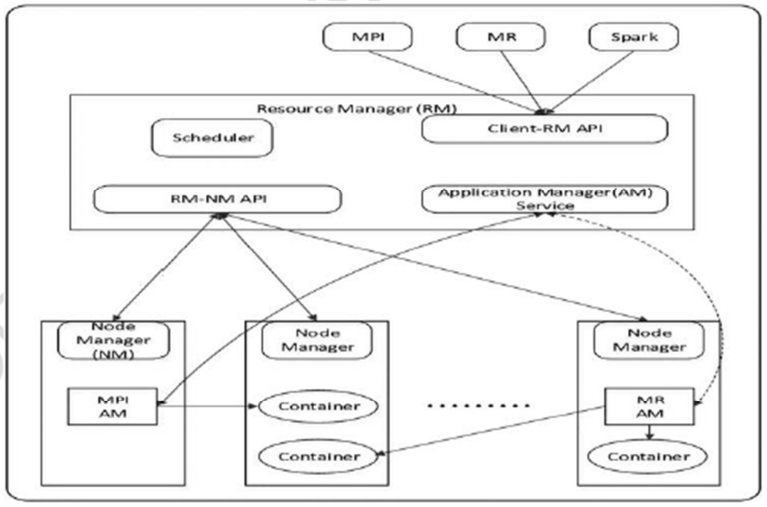


Figure 8.11
The YARN architecture for Hadoop resource management using three levels of managers to manage resources, applications, and nodes hierarchically.

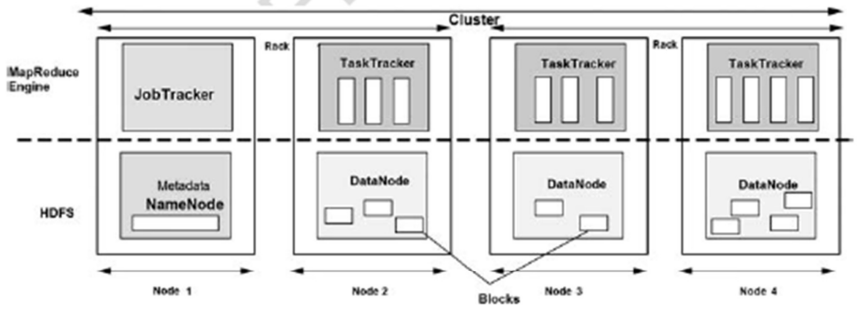


Figure 8.9
HDFS and MapReduce engine installed with, multiple nodes of a Hadoop server cluster.

Apache Hadoop Software Modules (1)
https://hadoop.apache.org/

- **Ambari : A web-based tool for provisioning, managing, and monitoring Apache Hadoop clusters. It supports HDFS, MapReduce, Hive, Hbase, Hcatelog, ZooKeeper, Pig, and Sqoop with a dashboard for viewing cluster health and diagnose their performance.**
- **Cassandra : A scalable multi-user database with no single points of failure.**
- **Chukwa: A data collection system for managing large distributed systems**
- **Hbase: A distributed database supports structured data/table storage**
- **Hive: A data warehouse for data summarization and ad hoc querying**

Hadoop Software Modules (2)
https://hadoop.apache.org/

- **Mahout: A scalable machine learning and data mining library**
- **Tez : A data-flow framework, built with YARN, which executes DAG type of tasks in batch and interactive modes**
- **ZooKeepeer : A coordination service for distributed applications.**
- **Pig: A data-flow language/execution framework for parallel computation.**
- **Spark: A expressive programming model for Hadoop data to be executed in streaming and general graph computation.**
- **Avro : A data serialization systems.**

Hadoop and Related Language and Programming Technologies

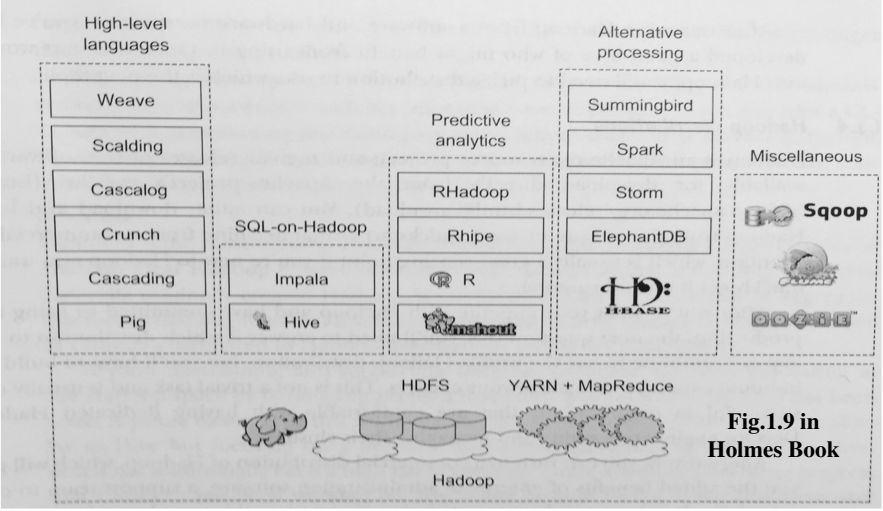


Fig.1.9 in Holmes Book

Prominent Use Cases of Hadoop

- **On February 19, 2008, Yahoo! Inc. launched what they claimed was the world's largest Hadoop production application. The Yahoo! Search Webmap is a Hadoop application that runs on a Linux cluster with more than 10,000 cores and produced data that was used in every Yahoo! web search query.**
- **There are multiple Hadoop clusters at Yahoo! and no HDFS file systems or MapReduce jobs are split across multiple data centers. Every Hadoop cluster node bootstraps the Linux image, including the Hadoop distribution. Work that the clusters perform is known to include the index calculations for the Yahoo! search engine. In June 2009, Yahoo! made the source code of its Hadoop version available to the open-source community.**
- **In 2010, Facebook claimed that they had the largest Hadoop cluster in the world with 21 PB of storage. In June 2012, they announced the data had grown to 100 PB and later that year they announced that the data was growing by roughly half a PB per day.**
- **As of 2013, Hadoop adoption had become widespread: more than half of the Fortune 50 used Hadoop.**

Hadoop hosting in the Cloud

Hadoop can be deployed in a traditional onsite datacenter as well as in the cloud. The cloud allows organizations to deploy Hadoop without the need to acquire hardware or specific setup expertise. Vendors who currently have an offer for the cloud include Microsoft, Amazon, IBM,[Google, Oracle, and CenturyLin Cloud·

On Amazon EC2/S3 services

It is possible to run Hadoop on Amazon Elastic Compute Cloud (EC2) and Amazon Simple Storage Service (S3). As an example, The New York Times used 100 Amazon EC2 instances and a Hadoop application to process 4 TB of raw image TIFF data (stored in S3) into 11 million finished PDFs in the space of 24 hours at a computation cost of about $240 (not including bandwidth).
There is support for the S3 object store in the Apache Hadoop releases, though this is below what one expects from a traditional POSIX filesystem. Specifically, operations such as rename() and delete() on directories are not atomic, and can take time proportional to the number of entries and the amount of data in them.

On Amazon Elastic MapReduce

Elastic MapReduce (EMR)[was introduced by Amazon.com in April 2009. Provisioning of the Hadoop cluster, running and terminating jobs, and handling data transfer between EC2(VM) and S3(Object Storage) are automated by Elastic MapReduce. Apache Hive, which is built on top of Hadoop for providing data warehouse services, is also offered in Elastic MapReduce. Support for using Spot Instances was later added in August 2011. Elastic MapReduce is fault-tolerant for slave failures, and it is recommended to only run the Task Instance Group on spot instances to take advantage of the lower cost while maintaining availability.

Hadoop On Microsoft Azure

- **Azure HDInsight is a service that deploys Hadoop on Microsoft Azure. HDInsight uses Hortonworks HDP and was jointly developed for HDI with Hortonworks. HDI allows programming extensions with .NET (in addition to Java).**
- **HDInsight also supports the creation of Hadoop clusters using Linux with Ubuntu. By deploying HDInsight in the cloud, organizations can spin up the number of nodes they want and only get charged for the compute and storage that is used.**
- **Hortonworks implementations can also move data from the on-premises datacenter to the cloud for backup, development/test, and bursting scenarios. It is also possible to run Cloudera or Hortonworks Hadoop clusters on Azure Virtual Machines.**

Hadoop On Google, CenturyLink Cloud (CLC) and Other Third Parties

Hadoop On Google Cloud Platform

There are multiple ways to run the Hadoop ecosystem on Google Cloud Platform ranging from self-managed to Google-managed.
Google Cloud Dataproc: a managed Spark and Hadoop service
command line tools (bdutil): a collection of shell scripts to manually create and manage Spark and Hadoop clusters
Google also offers connectors for using other Google Cloud Platform products with Hadoop, such as a Google Cloud Storage connector for using Google Cloud Storage and a Google BigQuery

CenturyLink Cloud offers Hadoop via both a managed and un-managed model. CLC also offers customers several managed Cloudera Blueprints, the newest managed service in the CenturyLink Cloud big data portfolio, which also includes Cassandra and MongoDB solutions.

Third party Hadoop distributions:

- Cloudera – using the Cloudera Director Plugin for Google Cloud Platform
- Hortonworks – using bdutil support for Hortonworks HDP
- MapR – using bdutil support for MapR

What is Apache Spark ?

- It is a cluster computing platform designed to be fast in general-purpose applications, compared with the use of MapReduce or Hadoop for just bi-parti graph computing
- On the speed side, Spark extends the MapReduce model to support interactive queries and streaming processing using in-memory computing.
- Spark offers the ability to run computations in memory, which is more efficient than MapReduce running on disks for complex applications.
- Spark is highly accessible, offering simple APIs in Python, Java, Scala, SQL, etc. Spark can run in Hadoop Clusters and access any Hadoop data source like Cassandra.

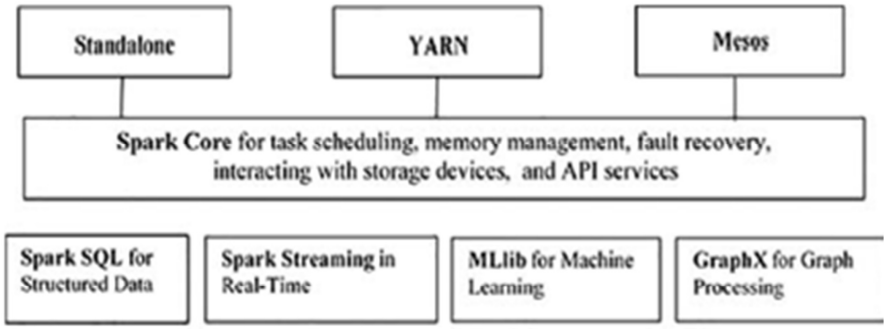


Figure 8.12
The core architecture of the Apache Spark software system.

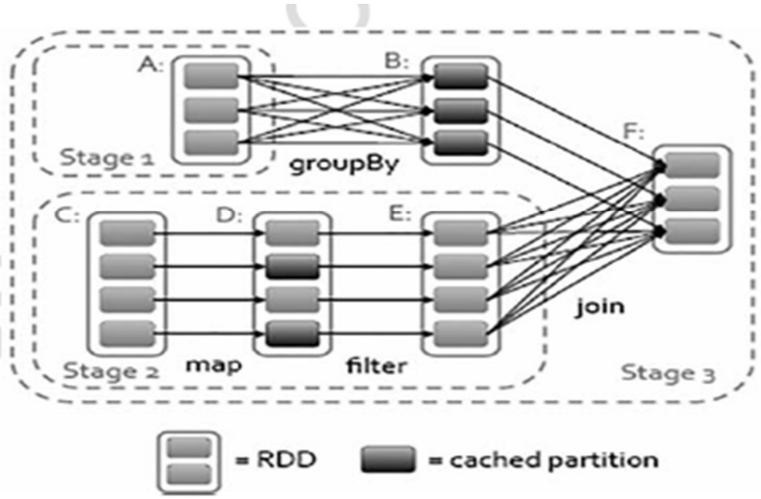


Figure 8.13
A typical task execution DAG graph, showing the scheduling of pipelined operations.

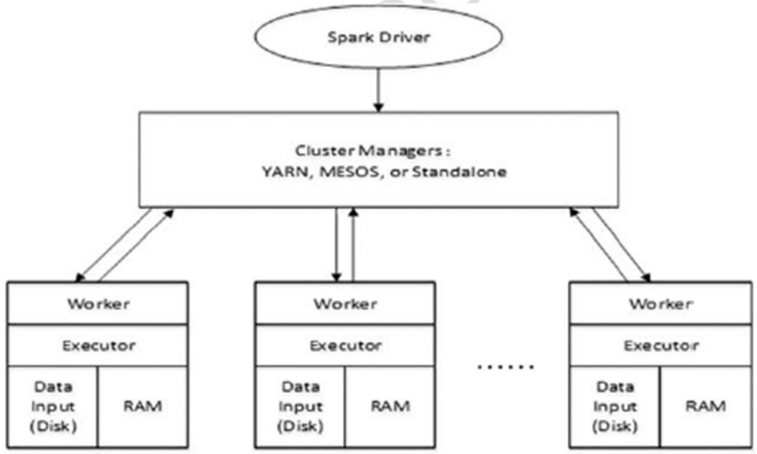


Figure 8.14
Distributed Spark execution model using a cluster of drafted workers (servers).

Table 8.4
Transformations and actions taken on the RDDs in Spark programming, where Seq(T) denotes a sequence of elements of type T (Source: Zaharia, 2016).

Transformations	*map*(*f* : T ⇒ U)	:	RDD[T] ⇒ RDD[U]
	filter(*f* : T ⇒ Bool)	:	RDD[T] ⇒ RDD[T]
	flatMap(*f* : T ⇒ Seq[U])	:	RDD[T] ⇒ RDD[U]
	sample(*fraction* : Float)	:	RDD[T] ⇒ RDD[T] (Deterministic sampling)
	groupByKey()	:	RDD[(K, V)] ⇒ RDD[(K, Seq[V])]
	reduceByKey(*f* : (V, V) ⇒ V)	:	RDD[(K, V)] ⇒ RDD[(K, V)]
	union()	:	(RDD[T], RDD[T]) ⇒ RDD[T]
	join()	:	(RDD[(K, V)], RDD[(K, W)]) ⇒ RDD[(K, (V, W))]
	cogroup()	:	(RDD[(K, V)], RDD[(K, W)]) ⇒ RDD[(K, (Seq[V], Seq[W]))]
	crossProduct()	:	(RDD[T], RDD[U]) ⇒ RDD[(T, U)]
	mapValues(*f* : V ⇒ W)	:	RDD[(K, V)] ⇒ RDD[(K, W)] (Preserves partitioning)
	sort(*c* : Comparator[K])	:	RDD[(K, V)] ⇒ RDD[(K, V)]
	partitionBy(*p* : Partitioner[K])	:	RDD[(K, V)] ⇒ RDD[(K, V)]
Actions	*count*()	:	RDD[T] ⇒ Long
	collect()	:	RDD[T] ⇒ Seq[T]
	reduce(*f* : (T, T) ⇒ T)	:	RDD[T] ⇒ T
	lookup(*k* : K)	:	RDD[(K, V)] ⇒ Seq[V] (On hash/range partitioned RDDs)
	save(*path* : String)	:	Outputs RDD to a storage system, *e.g.*, HDFS

The Core Concepts of Spark

- **Spark core contains components for**
  - **Task scheduling,**
  - **Memory management,**
  - **Fault recovery,**
  - **Interacting with storage systems,**
  - **Many APIs**
- **Spark's programming abstraction is enabled by RDD (Resilient Distributed Datasets), defined by many APIs for building and manipulating large collection of data items.**

Core Concepts of Spark (cont'd)

- **Spark SQL deal with structured data.**
- **Spark Streaming handles live streams of data.**
- **Mllib library contains common machine learning functionality.**
- **GraphX for manuipulating social network graphs.**
- **Spark's Cluster Manager can run with**
  - **Hadoop YARN**
  - **Apache Mesos**
  - **Spark's own Standalone Scheduler.**

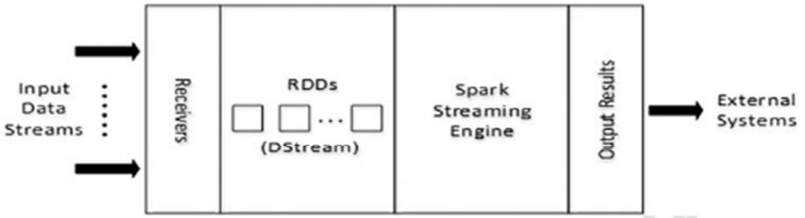


Figure 8.17
Concept of using Spark streaming engine on the DStreams.

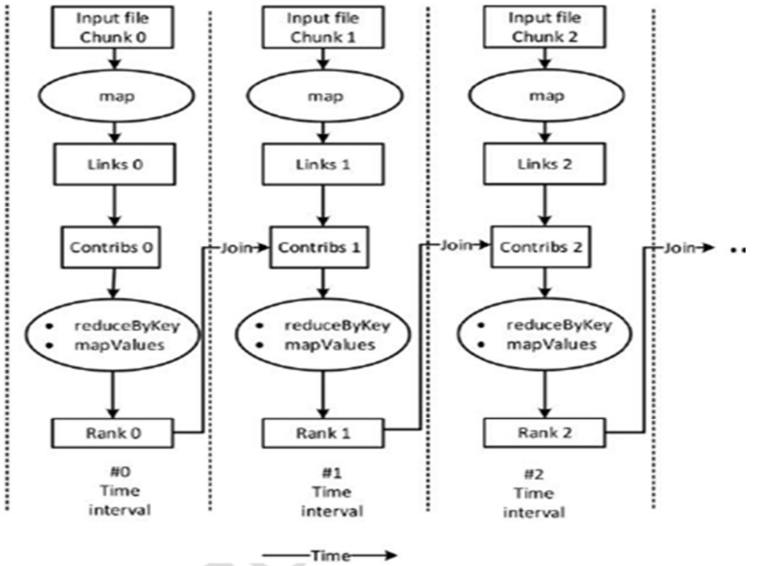


Figure 8.15
The counting process in PageRank algorithm. (Artwork courtesy of Wenhao Zhange, USC, 20

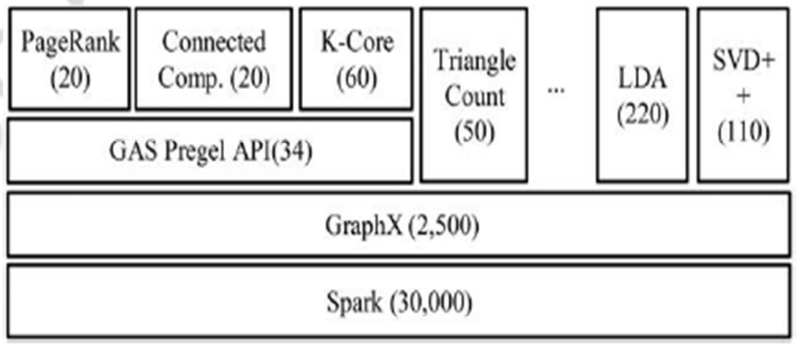


Figure 8.20
GraphX is a thin layer on top of the Spark dataflow framework.

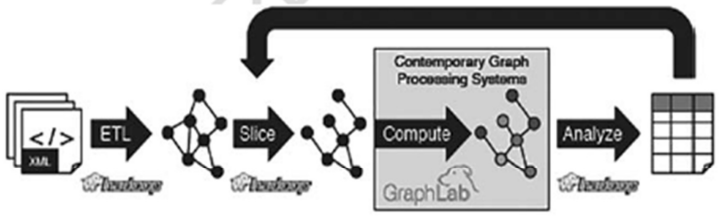
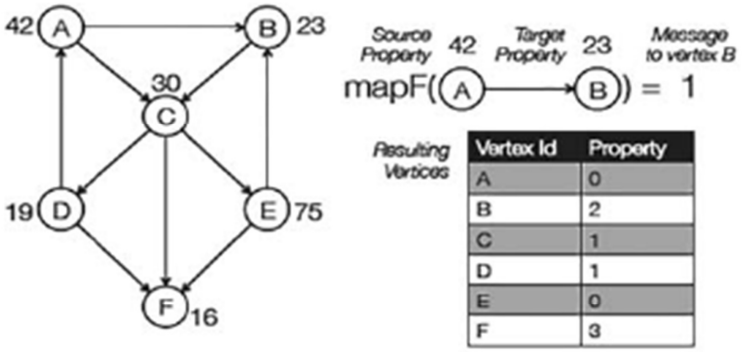


Figure 8.21
Graph analytics for pipelined processing of graph views. Source: Gonzalez et al., "GraphX: Graph Processing in a Distributed Dataflow Framework," 11th USENIX Symposium, OSDI, Bloomfield, CO, October 2014.



Vertex Id	Property
A	0
B	2
C	1
D	1
E	0
F	3

Figure 8.24
Compute the number of older followers of each vertex user.

Table 8.6
Spark TeraSort benchmark results reported in November 5, 2014

Data Features	Hadoop 100 TB	Spark, 100 TB	Spark, 1 PB
Data Size	102.5 TB	100 TB	1,000 TB
Elapsed Time	72 min	23 min	234 min
# Nodes	2,100	206	190
# Cores	50,400	6,592	6,080
# Reducers	10,000	29,000	250,000
Data Rate	1.42 TB/min	4.27 TB/min	4.27 TB/min
Rate/Node	0.67 GB/min	20.7 GB/min	22.5 GB/min

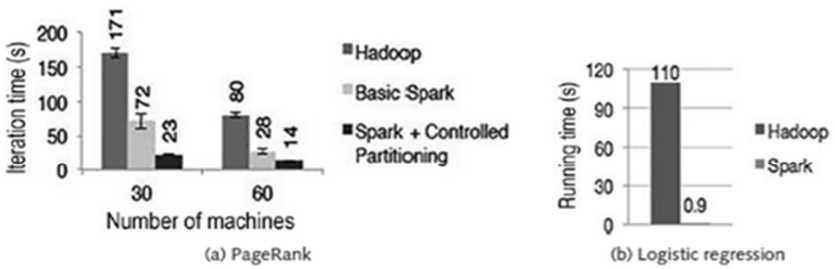


Figure 8.16
Relative performance of running PageRank and logistic regression on Hadoop and Spark separately.

Concluding Remarks :

- **To become a competent data scientist, you must master yourself with Hadoop and Spark programming skills.**
- **Both Hadoop and Spark use MapReduce engines in many existing clouds such as EMR in EC2 of AWS.**
- **For machine learning and graphic processing in streaming mode, Spark has outperformed Hadoop in some benchmark testing reports. Apache Hadoop treats Spark as its component.**
- **There is no short cut in learning these cloud programming skills unless you push yourself to use them in real-life cloud apps. Try to achieve this goal in your term project and in doing some homework problems.**