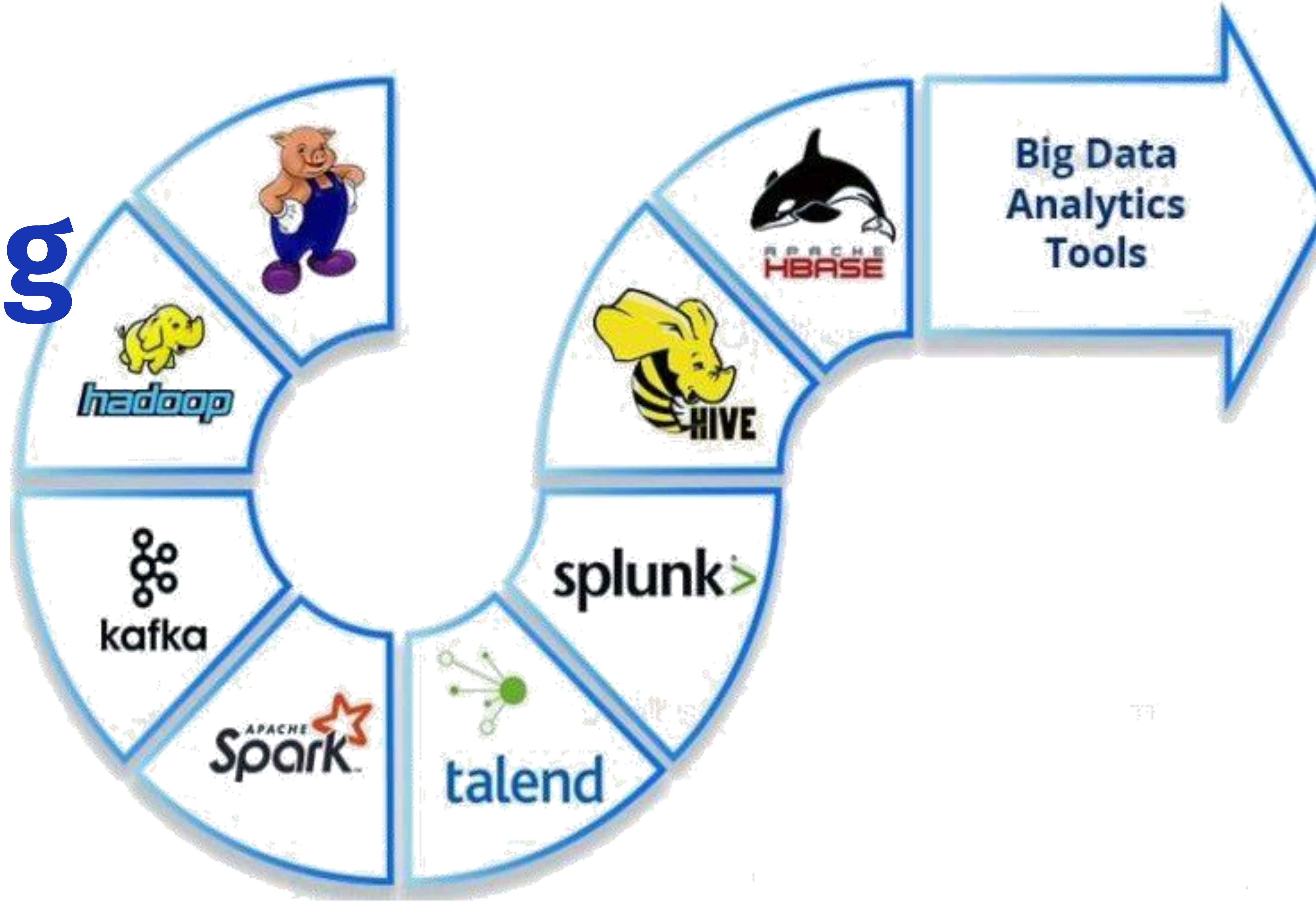

بسم الله الرحمن الرحیم

Parallel Processing in Big Data



Professor : Dr.Tadayon

Presented by : Mobarake Aftabi

Parallel Processing Systems for Big Data: A Survey

By Yunquan Zhang, *Member IEEE*, Ting Cao, *Member IEEE*, Shigang L
Liang Yuan, Haipeng Jia, and Athanasios V. Vasilakos, *Senior Mem*

ABSTRACT | The volume, variety, and velocity properties of *big data* and the valuable information it contains have motivated the investigation of many new parallel data processing systems in addition to the approaches using traditional database management systems (DBMSs). MapReduce pioneered this paradigm change and rapidly became the primary big data processing system for its simplicity, scalability, and fine-grain fault tolerance. However, compared with DBMSs, MapReduce also arouses controversy in processing efficiency, low-level abstraction, and rigid dataflow. Inspired by MapReduce, nowadays the big data systems are blooming. Some of them follow MapReduce's idea, but with more flexible models for general-purpose usage. Some absorb the advantages of DBMSs with higher abstraction. There are also specific systems for certain applications, such as machine learning and stream data processing. To explore new research opportunities and assist users in selecting suitable processing systems for specific applications, this survey paper will give a high-level overview of the existing parallel data processing systems categorized by the data input as
batch processing, stream proc
machine learning processing
projects in each category.
MapReduce system, as well as
sions on dataflow, data access
cation, and energy optimizatio
System benchmarks and open
will also be studied in this surv

KEYWORDS | Big data; machine
processing; SQL; survey

I. INTRODUCTION

The scale of petabyte data flo
social media, astronomy, and
ple, have driven the shift o
Big data refers to a collectio
not be processed using tradi
tools [72]. The storage, man
formation retrieval of big
searched and engineered by
Google's MapReduce [30]
spired new ways of thinking
gramming of large distribut
with traditional database ma
MapReduce is outstanding fo
ity, and fault-tolerance, but
and programming complexity
tion. Since the publication o
are numerous works targetin
duce. It is now the most ac
big data-processing system.
implementation of MapRed
used outside Google.

Manuscript received May 27, 2015; revised October 29, 2015 and May 19, 2016; accepted July 5, 2016. This work was supported by the National Key Research and Development Program of China under Grant [illegible], by the National Natural Science Foundation of China under Grant [illegible] ... by the National High Technology Research and Development Program of China under Grant [illegible] ... by the China Postdoctoral Science Foundation under Grant [illegible], by the Key Technology Research and Development Programs of Guangdong Province under Grant [illegible], and by the CAS Interdisciplinary Innovation Team of Efficient Space Weather Forecast Models.
Y. Zhang, T. Cao, S. Li, L. Yuan, and H. Jia are with the State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China (e-mail: [illegible]).
X. Tian is with the Advanced Computer Systems Research Center, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China (e-mail: [illegible]).
A. V. Vasilakos is with the Department of Computer Science, Electrical and Space Engineering, Lulea University of Technology, 97187 Lulea, Sweden (e-mail: [illegible]).
Digital Object Identifier: 10.1109/JPROC.2016.2591592

P

Parallel Processing with Big Data

Behrooz Parhami
Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA, USA

Synonyms

Big-data supercomputing; Computational needs of big data

Definition

Discrepancy between the explosive growth rate in data volumes and the improvement trends in processing and memory access speeds necessitates that parallel processing be applied to the handling of extremely large data sets.

Overview

Both data volumes and processing speeds have been on exponentially rising trajectories since the onset of the digital age (Denning and Lewis 2016), but the former has risen at a much higher rate than the latter. It follows that parallel processing is needed to bridge the gap. In addition to providing a higher processing capability to
deal with the requirements of lar
parallel processing has the potent
the "von Neumann bottleneck" (Ma
sometimes referred to as "the m
because of its tendency to hinde
progress of a computation, when op
be supplied to the processor at the
(McKee 2004; Wulf and McKee 1
processing algorithms and architect
1999) have been studied since th
way of improving computer system
and, more recently, as a way of c
exponential rise in performance w
the power consumption in check (G
Gepner and Kowalik 2006; Koomey

Trends in Parallel Processing

Interest in parallel processing b
1960s with the design of ILLIAC
recognized as the world's first s
(Hord 2013). The 64-processor m
built and operated by Burroughs Co
a single-instruction-stream, multipl
architecture, SIMD for short (Fly
1996), which uses a single instruct
unit, with each instruction applie
data items simultaneously. The oth
of parallel architectures is know
in which there are multiple instru
in addition to multiple data stream
architectural category has a great de
in terms of how memory is implem

2020 IEEE International Conference on Big Data (Big Data)

A Simple Low Cost Parallel Architecture for Big Data Analytics

Carlos Ordonez, University of Houston[§], USA
Sikder Tahsin Al-Amin[*], University of Houston[§], USA
Xiantian Zhou, University of Houston[§], USA

Abstract—Big Data Systems (Hadoop, DBMSs) require a complicated setup and tuning to store and process big data on a parallel cluster. This is mainly due to static partitioning when data sets are loaded or copied into the file system. Parallel processing thereafter works in a distributed manner, aiming for balanced parallel execution across nodes. Node synchronization, data redistribution and distributed caching in main memory are difficult to tune in the system. On the other hand, there exist analytical problems and algorithms, which can be computed in parallel, with minimal synchronization and fully independent computation. Moreover, some problems can be solved in one pass or few passes. In this paper, we introduce a low cost, yet useful, processing architecture in which data sets are dynamically partitioned at run-time and storage is transient. Each node processes one partition independently and partial results are gathered at the master processing node. Surprisingly, we show this architecture works well for some popular machine learning models as well as some graph algorithms. We attempt to identify which problem characteristics enable such efficient processing, and we also show the main bottleneck is the initial data set partitioning and distribution across nodes. We anticipate our architecture can benefit parallel processing in the cloud, where a dynamic number of virtual processors is decided at runtime or when the data set is analyzed for a short time.

Index Terms—Parallel architecture, Big Data, Parallel Processing.

I. Introduction

There has been a significant rising in data volumes and processing speeds for the last two decades. However, data volumes have risen at a much higher rate than the processing speeds. Though there are powerful machines with a lot of memory and disk space, it is costly and may fail when the data volume is huge. Therefore, processing and analyzing large volumes of data becomes non-feasible using a traditional serial approach. Hence, parallel processing emerges to solve the problem. Parallel processing allows a problem to be subdivided into smaller pieces that can be solved faster. Distributing the

some variants of parallel processing, it is often assumed that the same set of operations must be performed in each processing machine with shared-nothing architecture. For the output, most models send the partial output to the master node and combine the results to get the final result.

In this paper, our contributions are the following: (1) We propose a simple parallel architecture that can be used for parallel processing in big data analytics. (2) Our architecture does not depend on any external complicated file systems, rather we do the partition dynamically and run on commodity hardware. We use the file system "as is". (3) Our architecture is cheap, easy to set up, more machines can be added easily, and there is no need to maintain the partitions.

This is an outline of the rest of this article. Section 2 is a reference section. Section 3 presents our theoretical research contributions where we present our parallel architecture and how it comply with machine learning problems. Section 4 presents an experimental evaluation comparing our solution to the state of the art analytic systems. We discuss closely related work in Section 5. Conclusions and directions for future work are discussed in Section 6.

II. Preliminaries

In this section, we introduce the definitions and symbols used throughout the paper.

A. Input Data Set and Output Solution

We start by defining the input data set as D. Here, D is a matrix having n rows and a different number of columns, depending if the problem comes from machine learning or graphs. Matrix D can be either dense or sparse. We define the problem solution in a generalized manner as Θ. For machine learning problems, Θ is a model consisting of a list of matrices and associated metrics and for graphs, Θ is generally a vector

- Parallel DBMS
- Big Data
- Map Reduce
- Hadoop
- Other Techniques
- Conclusion

Parallel DBMS

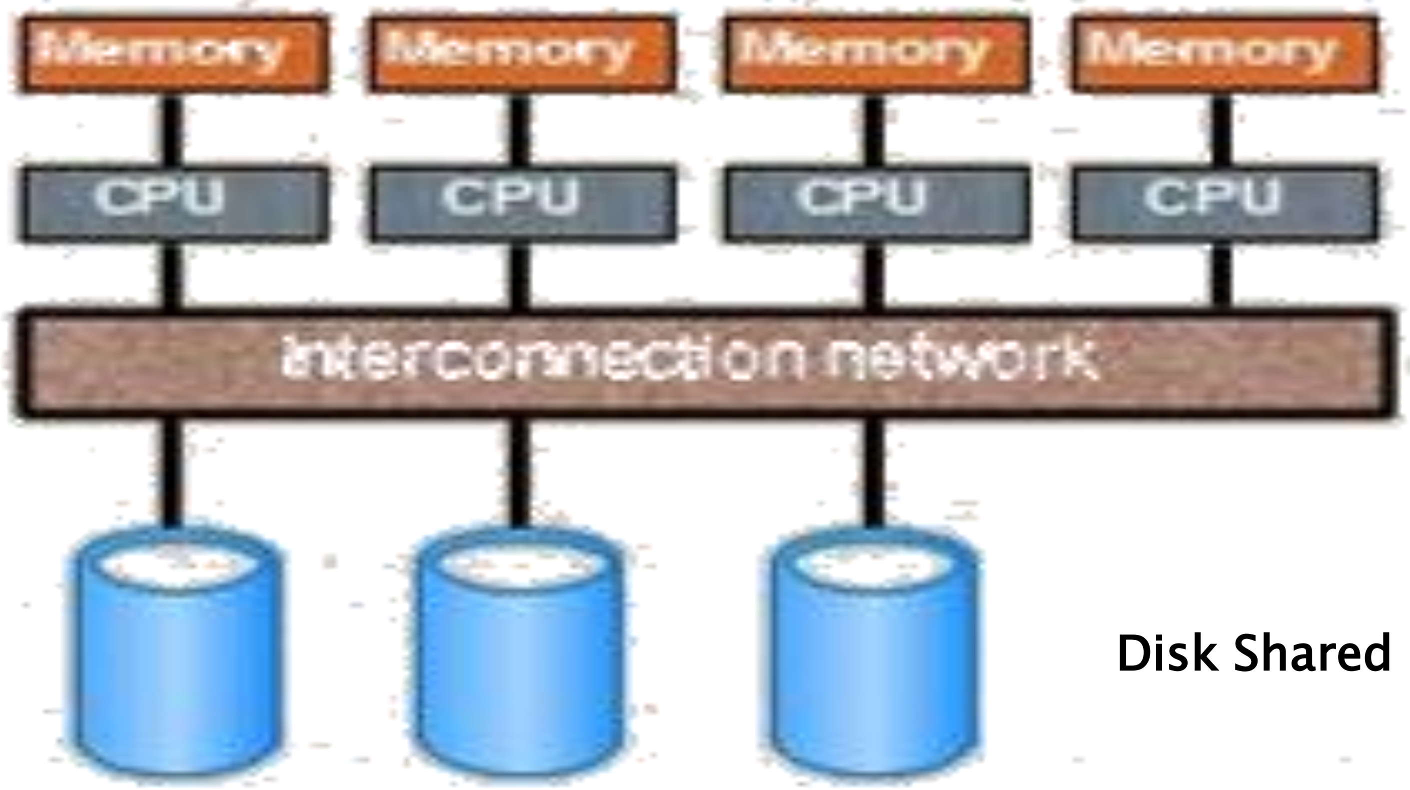
A parallel database management system seeks to improve performance through operational parallelization such as data loading, indexing, and query evaluation.

Has been able to run database systems on clusters of SN nodes since the 1980s. These systems support standard SQL spreadsheets, and as a result, the fact that data is distributed across multiple systems is obscured from the end user.

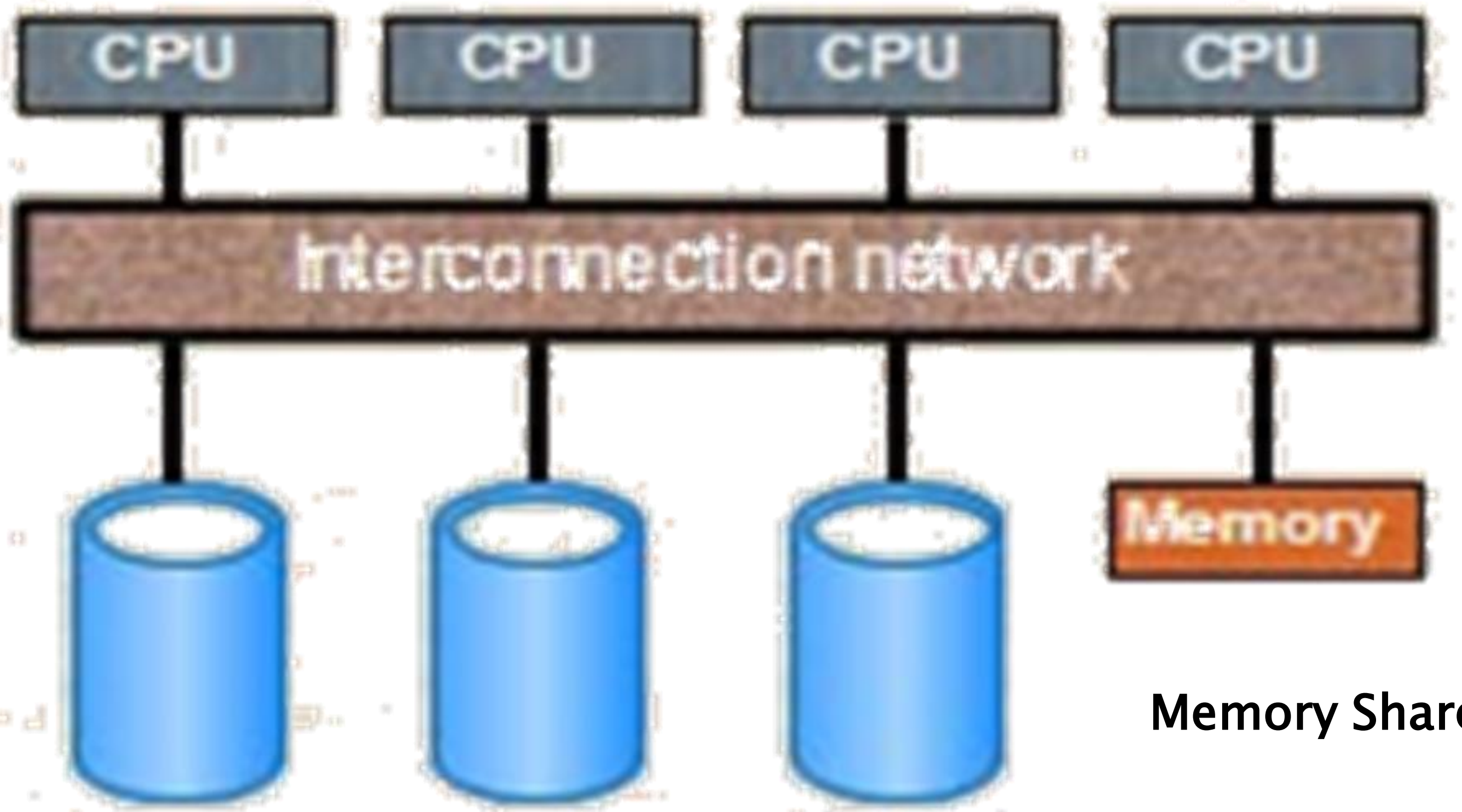
All or most of the tables are divided into nodes in a cluster

The system uses an optimizer to convert SQL statements to a query map whose execution is distributed across several nodes.

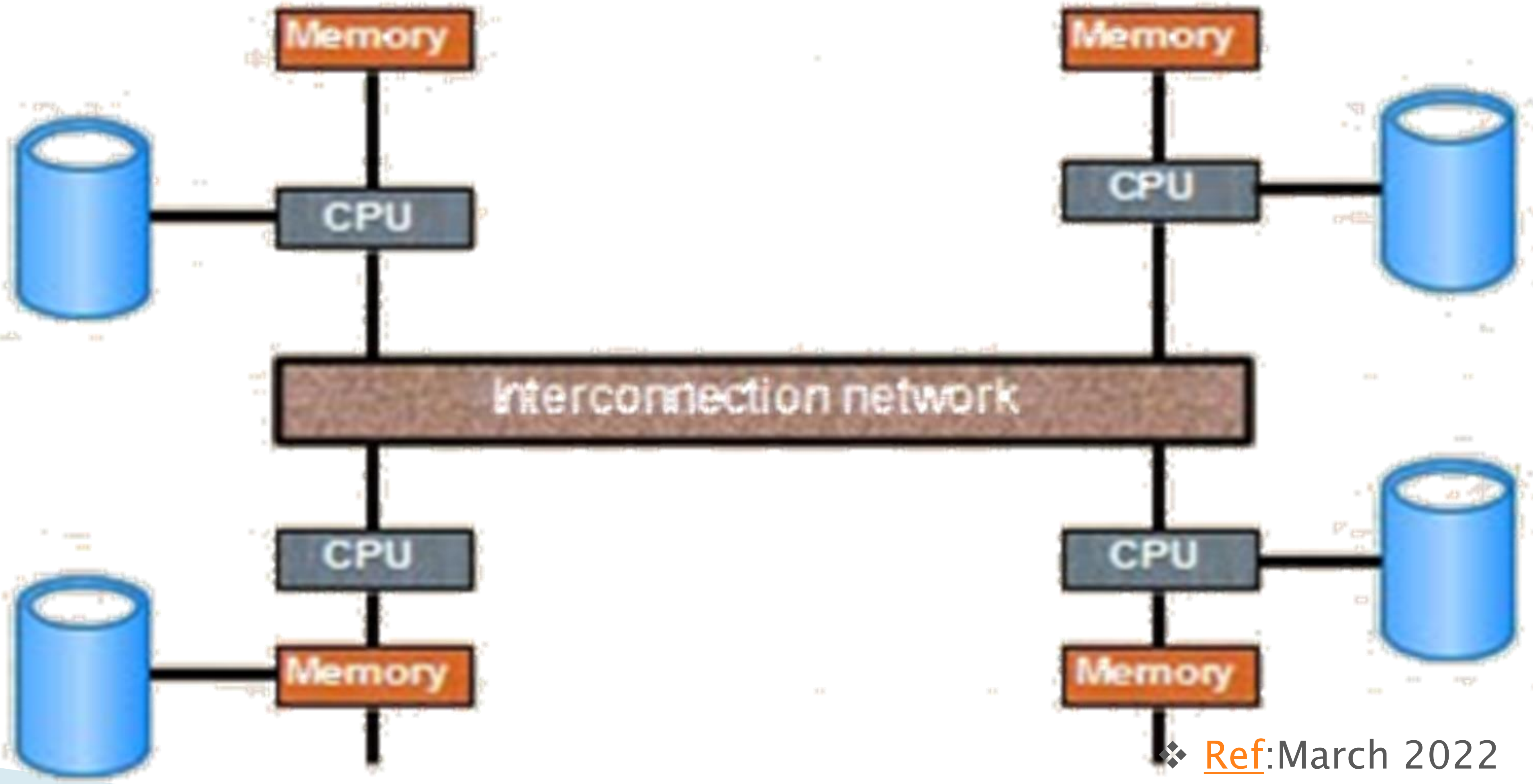
Parallel DBMS



Disk Shared



Memory Shared



Share Nothing

❖ Ref:March 2022

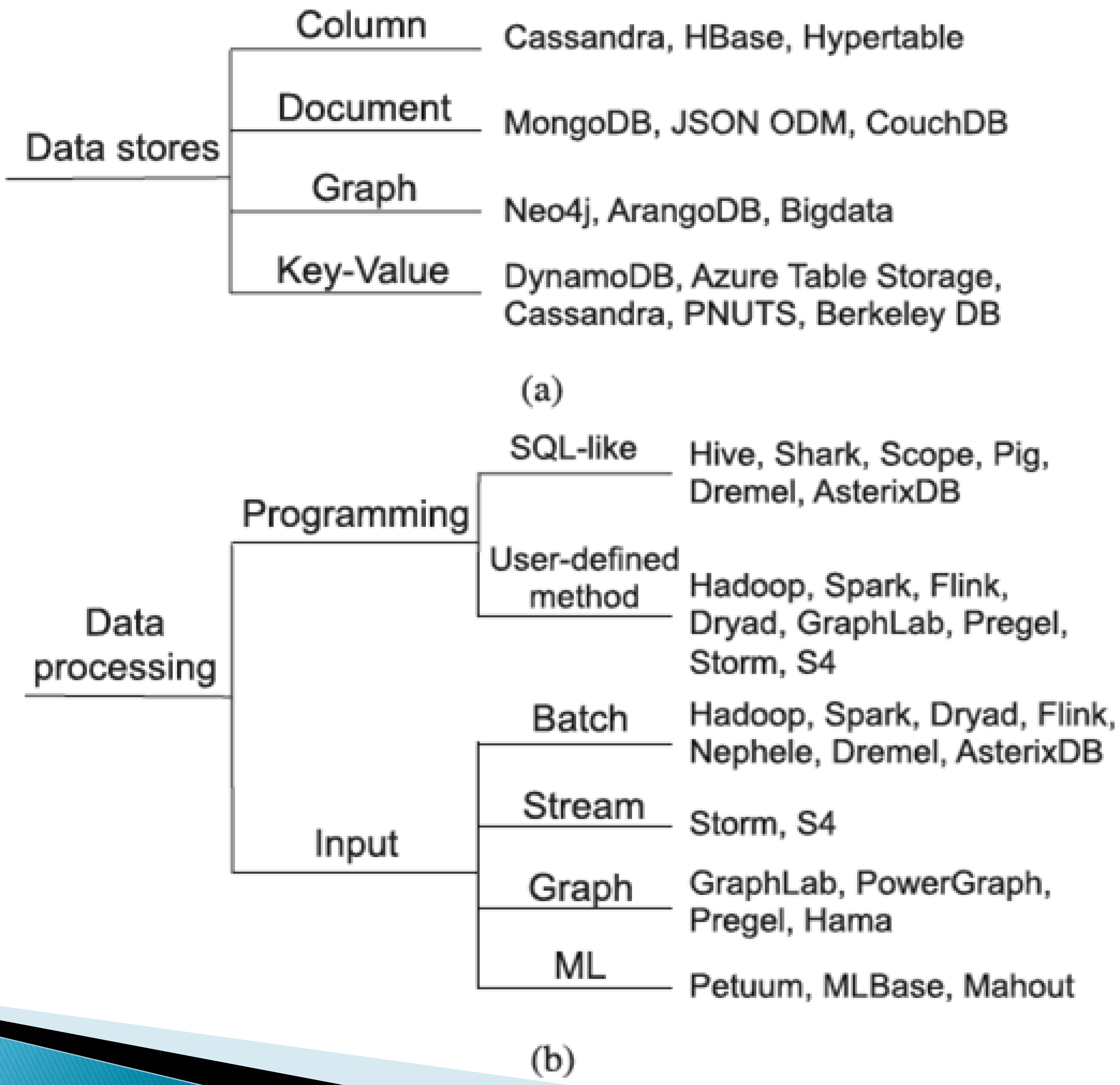
What is Big Data?(Cont.)

Author's name	Definition
Batty	Big data are massive in size and cannot fit into Excel spreadsheets comprising approximately 16 000 columns and 1 million rows.
Havens et al.	Big data cannot be loaded into local storage devices (computer memory).
Fisher et al.	Big data cannot be easily processed and managed in a straightforward manner.
The State Council of People's Republic of China	Big data have several characteristics, such as high application value, fast access speed, large volume, and multiple types.
Bayer and Laney	Big data have large volume, variety, and velocity that demand cost effectiveness and are helpful in decision making.

❖ Ref:March 2022

Characteristics of big data

❖ Ref: January 2022

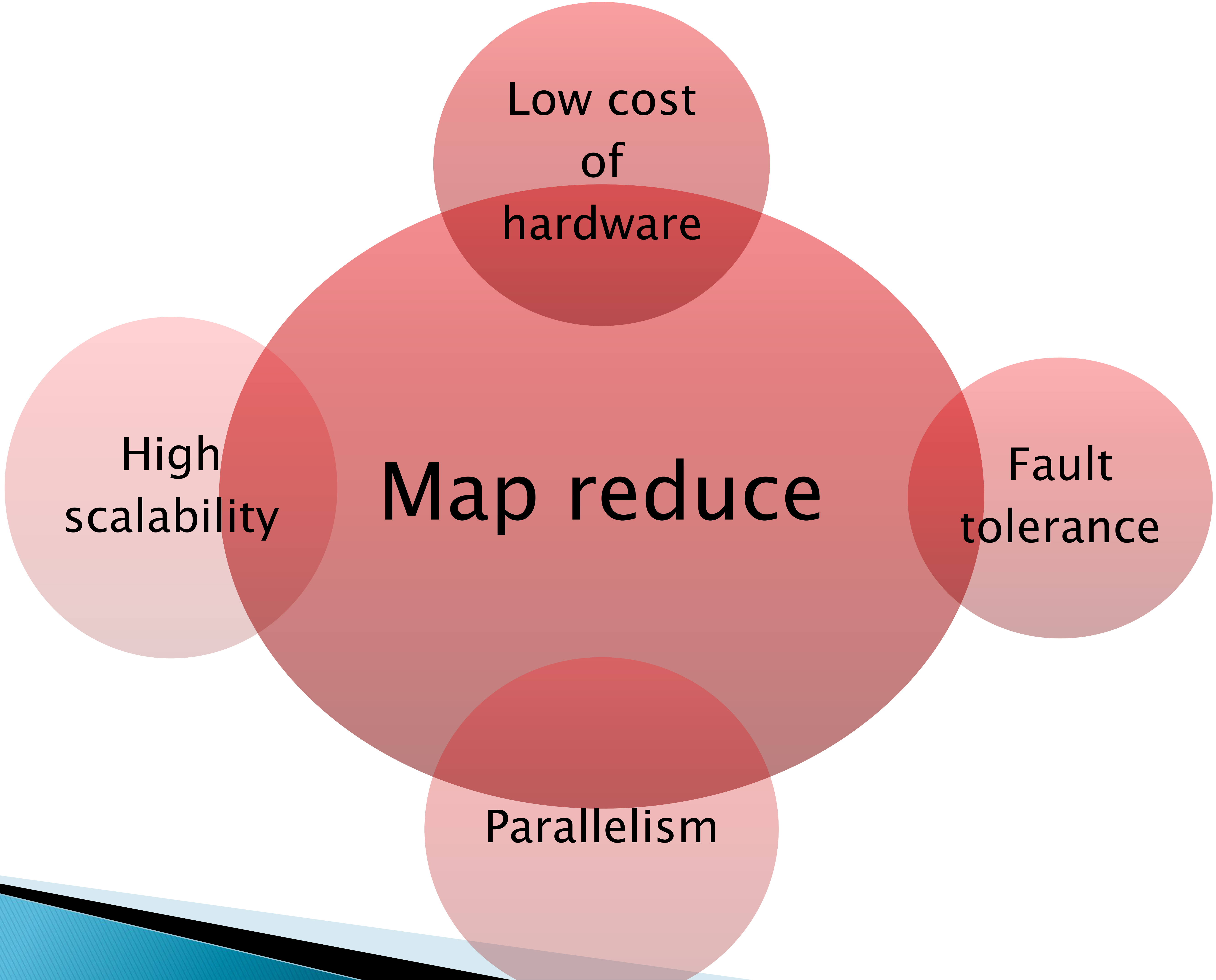

Column
Cassandra, HBase, Hypertable
Document
MongoDB, JSON ODM, CouchDB
Data stores
Graph
Neo4j, ArangoDB, Bigdata
Key-Value
DynamoDB, Azure Table Storage, Cassandra, PNUTS, Berkeley DB
(a)
SQL-like
Hive, Shark, Scope, Pig, Dremel, AsterixDB
Programming
User-defined method
Hadoop, Spark, Flink, Dryad, GraphLab, Pregel, Storm, S4
Data processing
Batch
Hadoop, Spark, Dryad, Flink, Nephele, Dremel, AsterixDB
Stream
Storm, S4
Input
Graph
GraphLab, PowerGraph, Pregel, Hama
ML
Petuum, MLBase, Mahout
(b)

❖ Ref

Map Reduce

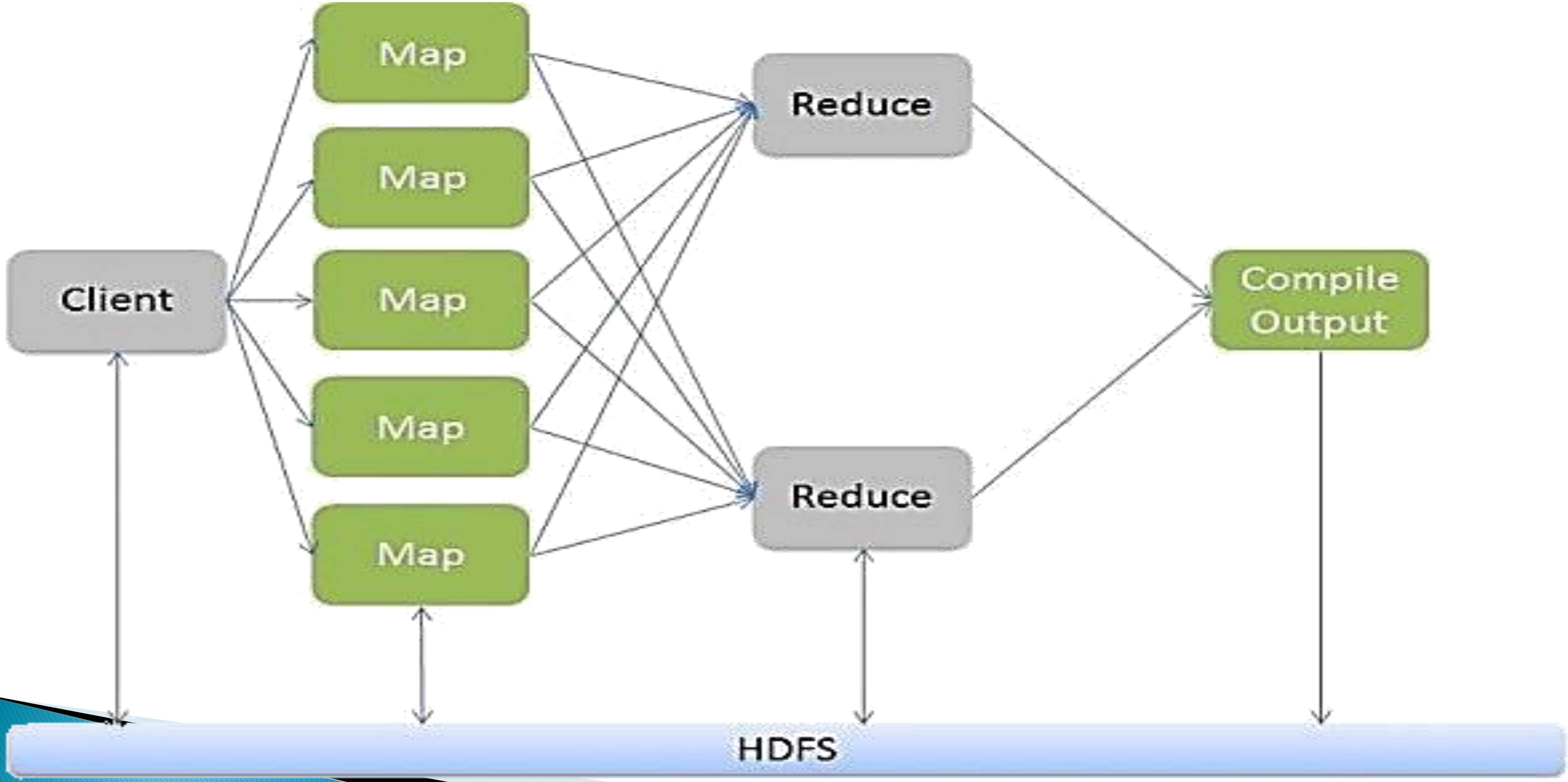
MapReduce is a parallel programming model for processing data on clusters that consists of two main phases including the mapping phase (Map) and the reduction phase (Reduce).

This framework divides the big data into subcategories, then divides them into different machines, then combines the separate processes into the final result.


Low cost of hardware
High scalability
Map reduce
Fault tolerance
Parallelism

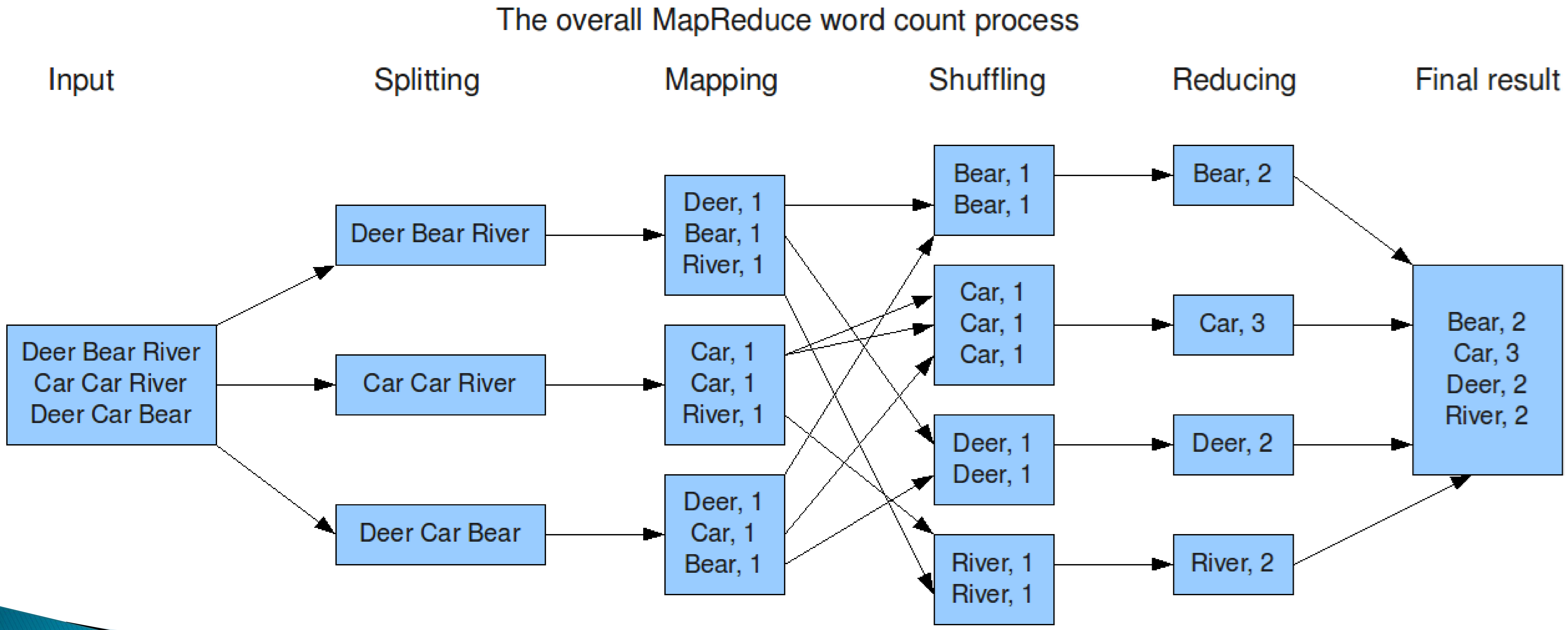
❖ Ref

Map Reduce



❖ Ref

Example



The overall MapReduce word count process
Input
Splitting
Mapping
Shuffling
Reducing
Final result
Deer Bear River
Car Car River
Deer Car Bear
Deer Bear River
Car Car River
Deer Car Bear
Deer, 1
Bear, 1
River, 1
Car, 1
Car, 1
River, 1
Deer, 1
Car, 1
Bear, 1
Bear, 1
Bear, 1
Car, 1
Car, 1
Car, 1
Deer, 1
Deer, 1
River, 1
River, 1
Bear, 2
Car, 3
Deer, 2
River, 2
Bear, 2
Car, 3
Deer, 2
River, 2

❖ Ref

Map Radius implementations

Apache Hadoop MapReduce

Google MapReduce

Amazon Elastic Mapreduce

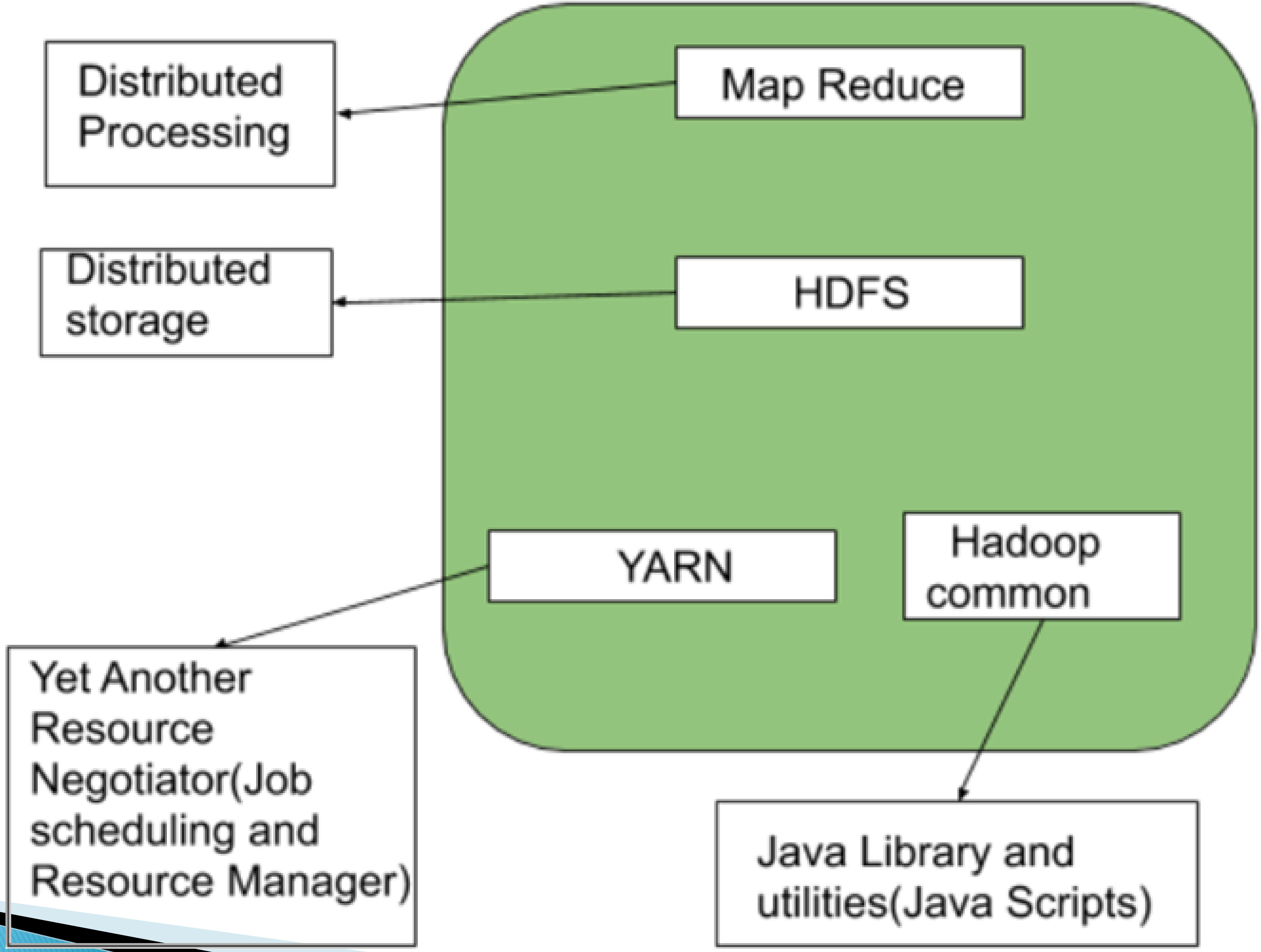
❖ Ref

Table 1 Comparison of Parallel DBMSs and MapReduce

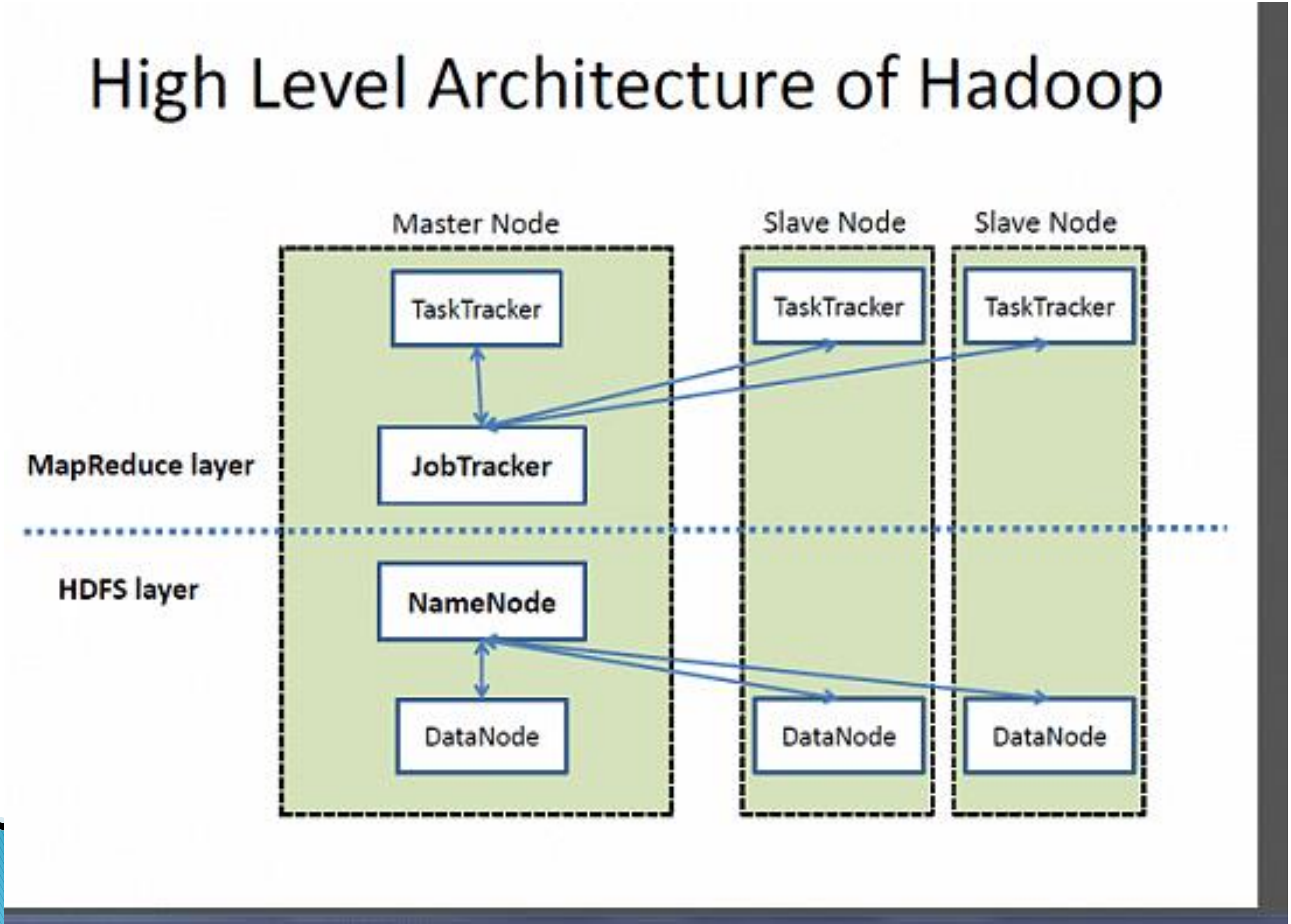
	Parallel DBMS	MapReduce
Schema	✓	Not naturally
Index	✓	Not naturally
Programming	Declarative (SQL)	Imperative (C++, Java, ...)
Optimization	Compression, Column storage, ...	Not naturally
Pre-parsing	✓	Not naturally
Flexibility	Not naturally	✓
Fault tolerance	Transaction-level	Task-level

❖ Ref


MANAGE
CLOUD
STATISTIC
RISK
HADOOP
SHARE
OPEN SOURCE
BUSINESS
INTERNET
ANALYSIS
SECURITY
STORAGE
INTERNET
HADOOP
DATA
SHARE
MANAGE
VOLUME
CLOUD
BUSINESS
ANALYTICS
SYSTEMS
HADOOP
LEARNING
COMPUTING
ANALYTICS
ENTERPRISE
PROJECTS
BIG DATA
CLOUD
MANAGE
SYSTEMS
ENTERPRISE
COMPLEX
SERVER
VOLUME
HADOOP
BIG
DATA
TREND
SOCIAL
MEDIA
SHARE
SOLVED
OPEN SOURCE
BIG
DATA
SOFTWARE
VOLUME
SOLUTIONS
DATABASE
RISK
TECHNOLOGIES
SYSTEMS
COMPUTING
SOCIAL
MEDIA
TREND
HADOOP
CLOUD
LEARNING
MARKETING
TOOLS
RISK
SYSTEMS
SOFTWARE
COMPLEX
CLOUD
BUSINESS
RISK
NETWORKS
DATABASE
SOLVED
HADOOP
SYSTEMS
COMPLEX
BIG
SERVER
DATA
SECURITY
OPEN SOURCE

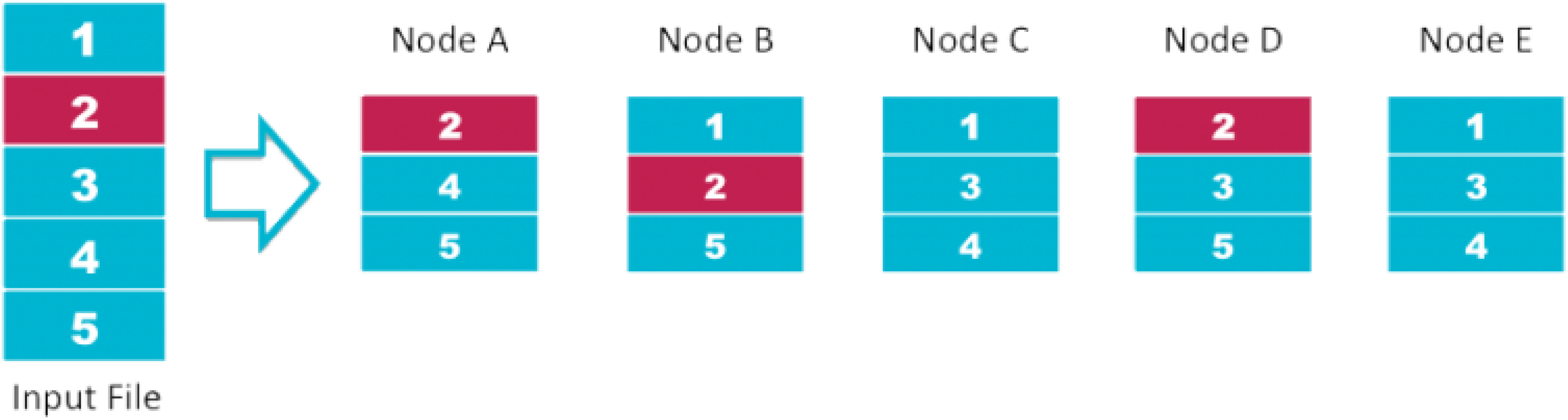

Distributed
Processing
Map Reduce
Distributed
storage
HDFS
YARN
Hadoop
common
Yet Another
Resource
Negotiator(Job
scheduling and
Resource Manager)
Java Library and
utilities(Java Scripts)

❖ Ref


High Level Architecture of Hadoop
Master Node
Slave Node
Slave Node
TaskTracker
TaskTracker
TaskTracker
JobTracker
MapReduce layer
HDFS layer
NameNode
DataNode
DataNode
DataNode

❖ Ref

HDFS Data Distribution



❖ Ref

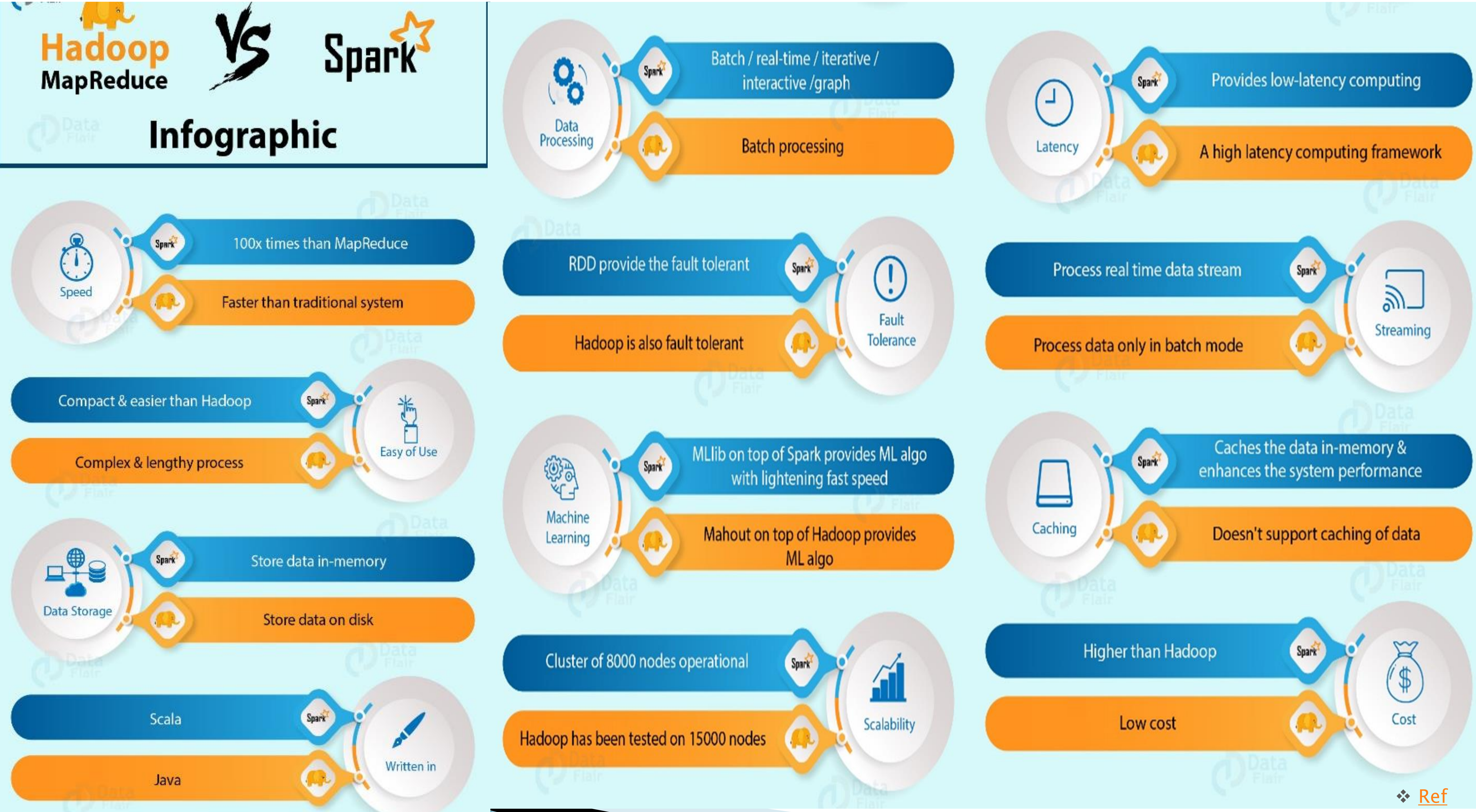
APACHE Spark™

Flink

Apache mahout

APACHE STORM™

splunk>®


Hadoop
MapReduce
VS
Spark
Infographic
Speed
100x times than MapReduce
Faster than traditional system
Easy of Use
Compact & easier than Hadoop
Complex & lengthy process
Data Storage
Store data in-memory
Store data on disk
Written in
Scala
Java
Data Processing
Batch / real-time / iterative / interactive /graph
Batch processing
Fault Tolerance
RDD provide the fault tolerant
Hadoop is also fault tolerant
Machine Learning
MLlib on top of Spark provides ML algo with lightening fast speed
Mahout on top of Hadoop provides ML algo
Scalability
Cluster of 8000 nodes operational
Hadoop has been tested on 15000 nodes
Latency
Provides low-latency computing
A high latency computing framework
Streaming
Process real time data stream
Process data only in batch mode
Caching
Caches the data in-memory & enhances the system performance
Doesn't support caching of data
Cost
Higher than Hadoop
Low cost

- Ref

Big data benchmark

Measuring and comparing big data systems and architecture	Being data-centric	Diverse and representative workloads	Covering representative software stacks	State-of-the-art techniques	Usability

Ref

Table 3 The Summary of BigDataBench. Adaptation With the Permission of Wang *et al.* [78]

Application Scenarios	Application Type	Workloads	Data Types	Data Source	Software Stacks
Micro Benchmarks	Offline Analytics	Sort	Unstructured	Text	Hadoop, Spark, MPI
		Grep			
		WordCount			
		BFS		Graph	
Basic Datastore Operations	Online Service	Read	Semi-structured	Table	Hbase, Cassandra, MongoDB, MySQL
		Write			
		Scan			
Relational Query	Realtime Analytics	Select Query	Structured	Table	Impala, MySQL, Hive, Shark
		Aggregate Query			
		Join Query			
Search Engine	Online Services	Nutch Server	Un-structured	Text	Hadoop
	Offline Analytics	Index			
		PageRank		Graph	Hadoop, Spark, MPI
Social Network	Online Services	Olio Server	Un-structured	Graph	Apache+MySQL
	Offline Analytics	Kmeans			Hadoop, Spark, MPI
		Connected Components			
E-commerce	Online Services	Rubis Server	Structured	Table	Apache+JBoss+MySQL
	Offline Analytics	Collaborative Filtering	Semi-structured	Text	Hadoop, Spark, MPI
		Naive Bayes			

❖ Ref

Conclusion

Big data-processing systems have been widely researched by academia and industry. Based on the processing paradigm, we categorize those systems into batch, stream, graph, and machine learning processing. The paper first introduced the basic framework of MapReduce and its outstanding features as well as deficiencies compared to DBMSs. According to the deficiencies, we discussed the extensions and optimizations for MapReduce platform, including support for flexible dataflows, efficient data access and communication, parameter tuning, as well as energy. We then surveyed other batch-processing systems, including general-purpose systems Dryad, Nephele/PACT and Spark. SQL-like systems involved in this paper are Hive, Shark, SCOPE, AsterixDB, and Dremel. For stream processing systems, Storm and S4 are introduced as representatives. Scalability is one of the ML algorithms bottlenecks. We then discussed how graph-centric systems like Pregel and GraphLab, and ML-centric systems like Petuum, parallelize the graph and ML model, as well as their distinctive characteristics.

1. https://ieeexplore.ieee.org/document/7547948
2. https://ieeexplore.ieee.org/document/9378386
3. https://doi.org/10.1007/978-3-319-63962-8_165-1


Thank you