جامعة المستقبل – كلية العلوم
قسم الانظمة الطبية الذكية
Intelligent Medical Systems Department

جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

Map-Reduce Techniques

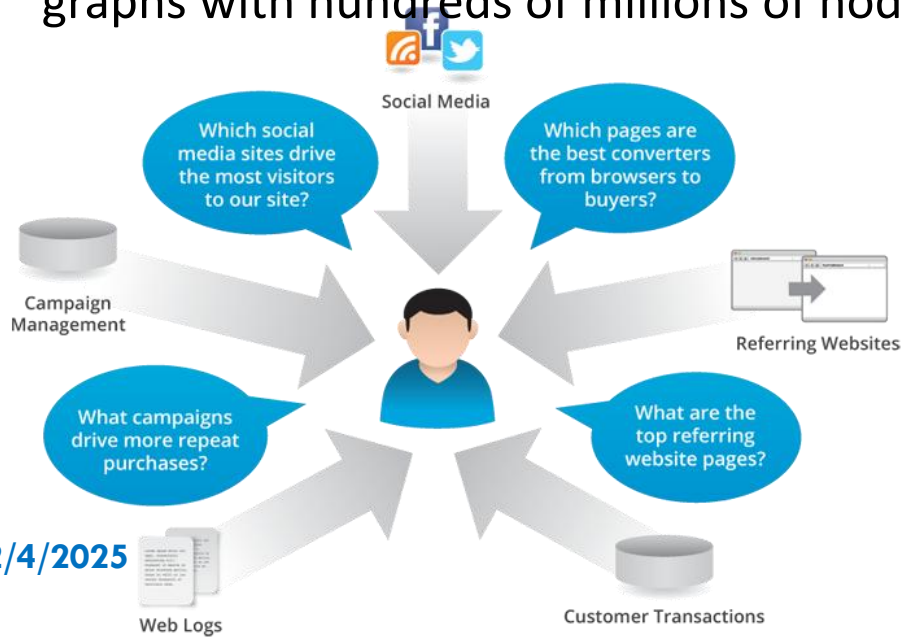
Lecture (8)

Prof. Dr. Mehdi Ebady Manaa

Map-Reduce Concepts

"big-data" analysis, require us to manage immense amounts of data quickly. In many of these applications, the data is extremely regular, and there is ample opportunity to exploit parallelism. Important examples are:

1. The ranking of Web pages by importance, which involves an iterated matrix-vector multiplication where the dimension is many billions.
2. Searches in "friends" networks at social-networking sites, which involve graphs with hundreds of millions of nodes and many billions of edges.



Map-Reduce Definition

MapReduce, the programming model for processing large data sets with a parallel and distributed algorithm on a cluster, the cornerstone of the Big Data, was invented by Google.

These programming systems are designed to get their ***parallelism*** not from a "supercomputer," but from ***"computing clusters"*** – large collections of commodity hardware, including conventional processors ("compute nodes") connected by Ethernet cables or inexpensive switches.
The software stack begins with a new form of file system, called a "**distributed file system**," which features much larger units than the disk blocks in a conventional operating system. Distributed file systems also provide replication of data or redundancy to protect against the frequent media failures that occur when data is distributed over thousands of low-cost compute nodes.

Advantages of Map-Reduce Stack Programming

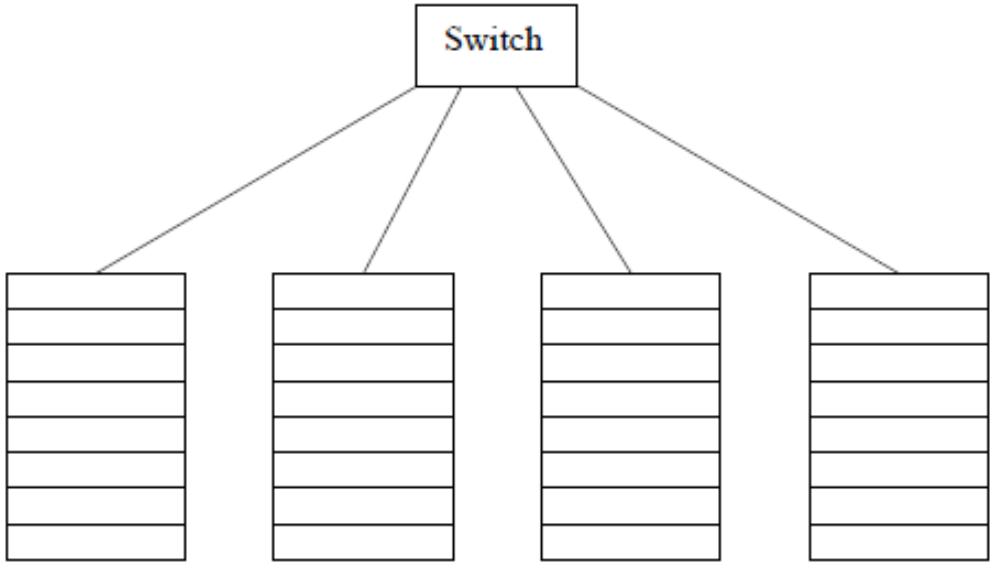
- Scalable.
- Big Data Analysis which leads to cost-effective Solution
- Simple of Implementation than Supercomputer Parallelism
- Reliability
- Fast
- Security and Authentication
- Parallel Processing by divide data into Chunks
- Availability and Resilient Nature

Distributed File System (DFS)

Most computing is done on a single processor, with its main memory, cache, and local disk (a compute node).

- In the past, applications that called for parallel processing, such as large scientific calculations, were done on special-purpose parallel computers with many processors and specialized hardware. However, the prevalence of large-scale Web services has caused more and more computing to be done on installations with thousands of compute nodes operating more or less independently. In these installations, the compute nodes are commodity hardware, which greatly reduces the cost compared with special-purpose parallel machines.
- These new computing facilities have given rise to a new generation of programming systems. These systems take advantage of the power of parallelism and 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.

Physical organization of Compute Node



Racks of compute nodes

The new parallel-computing architecture, sometimes called cluster computing, is organized as follows.

- Compute nodes are stored on racks, perhaps 8–64 on a rack.
- The nodes on a single rack are connected by a network, typically 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, but given the number of pairs of nodes that might need to communicate between racks However, there may be many more racks and many more compute nodes per rack.

Physical organization of Compute Node

Some important calculations take minutes or even hours on thousands of compute nodes. If we had to abort and restart the computation every time one component failed, then the computation might never complete successfully. The solution to this problem takes two forms:

- Files must be stored redundantly.
- Computations must be divided into tasks, such that if any one task fails to execute to completion, it can be restarted without affecting other tasks.

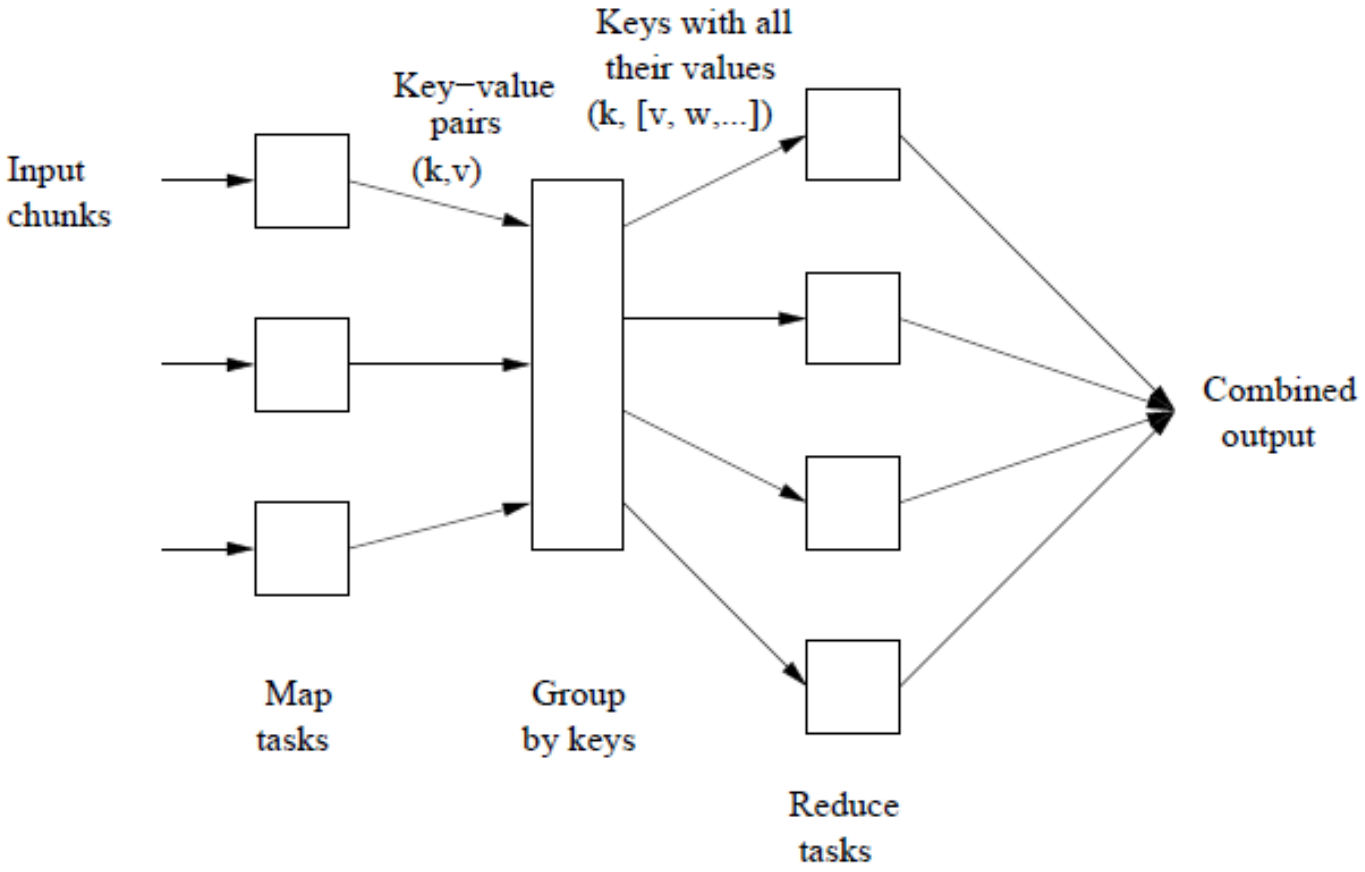
Map-Reduce

MapReduce is a style of computing that has been implemented in several systems, including Google's internal implementation (simply called MapReduce) and the popular open-source implementation Hadoop which can be obtained, along with the HDFS file system from the Apache Foundation. You can use an implementation of MapReduce to manage many large-scale computations in a way that is tolerant of hardware faults.

Map-Reduce

1. Some number of Map tasks each are given one or more chunks from a distributed file system. These Map tasks turn the chunk into a sequence of key-value pairs. The way key-value pairs are produced from the input data is determined by the code written by the user for the Map function.
2. The key-value pairs from each Map task are collected by a master controller and sorted by key. The keys are divided among all the Reduce tasks, so all key-value pairs with the same key wind up at the same Reduce task.
3. The Reduce tasks work on one key at a time, and combine all the values associated with that key in some way. The manner of combination of values is determined by the code written by the user for the Reduce function.

Map- Reduce


Keys with all
their values
(k, [v, w,...])
Key−value
pairs
(k,v)
Input
chunks
Combined
output
Map
tasks
Group
by keys
Reduce
tasks

Map- Tasks (Word Count Program)

Input	Set of data	Bus, Car, bus, car, train, car, bus, car, train, bus, TRAIN,BUS, buS, caR, CAR, car, BUS, TRAIN
Output	Convert into another set of data (Key,Value)	(Bus,1), (Car,1), (bus,1), (car,1), (train,1), (car,1), (bus,1), (car,1), (train,1), (bus,1), (TRAIN,1),(BUS,1), (buS,1), (caR,1), (CAR,1), (car,1), (BUS,1), (TRAIN,1)

Reduce Function

Input **(output of Map function)**	Set of Tuples	(Bus,1), (Car,1), (bus,1), (car,1), (train,1), (car,1), (bus,1), (car,1), (train,1), (bus,1), (TRAIN,1),(BUS,1), (buS,1), (caR,1), (CAR,1), (car,1), (BUS,1), (TRAIN,1)
Output	Converts into smaller set of tuples	(BUS,7), (CAR,7), (TRAIN,4)

Execution Steps

Workflow of MapReduce consists of 5 steps

1. **Splitting** – The splitting parameter can be anything, e.g. splitting by space, comma, semicolon, or even by a new line ('\n').
2. **Mapping** – as explained above
3. **Intermediate splitting** – the entire process in parallel on different clusters. In order to group them in "Reduce Phase" the similar KEY data should be on same cluster.
4. **Reduce** – it is nothing but mostly group by phase
5. **Combining** – The last phase where all the data (individual result set from each cluster) is combine together to form a Result

Word Counts Example

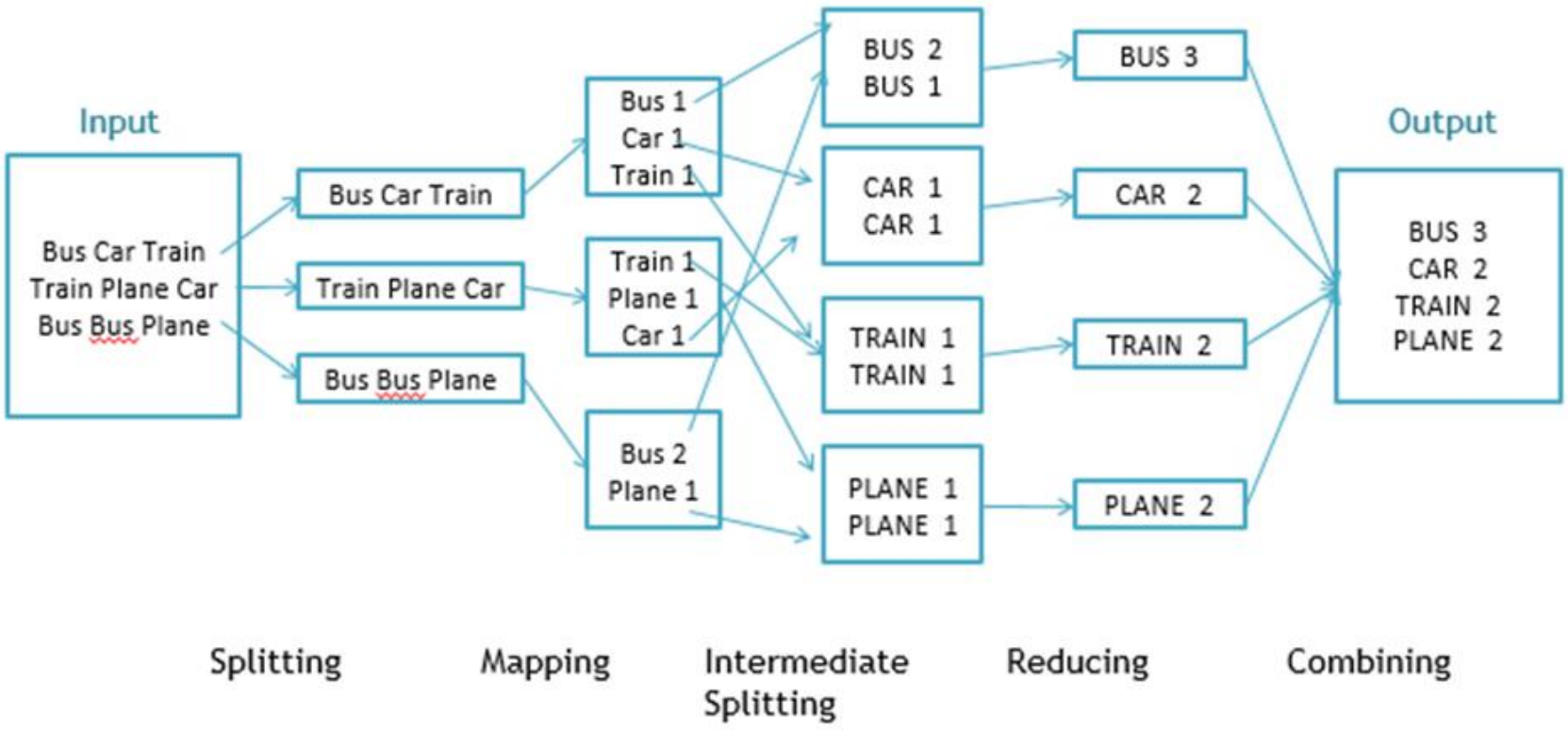


Fig. WorkFlow of MapReducing

THANK YOU