Hadoop and MapReduce

Chapter 8

Outline

Data

- Before the era of the Internet: Data → store and process in row and column, i.e.: Structure date
  - Data are limited
  - Single processor can be used
- As the Internet arise (around 2000)
  - Type of Data change: Text, image, video (later)
    - Rise of **Big Data**
- Problem: how to store and process Big data

Data-intensive computing

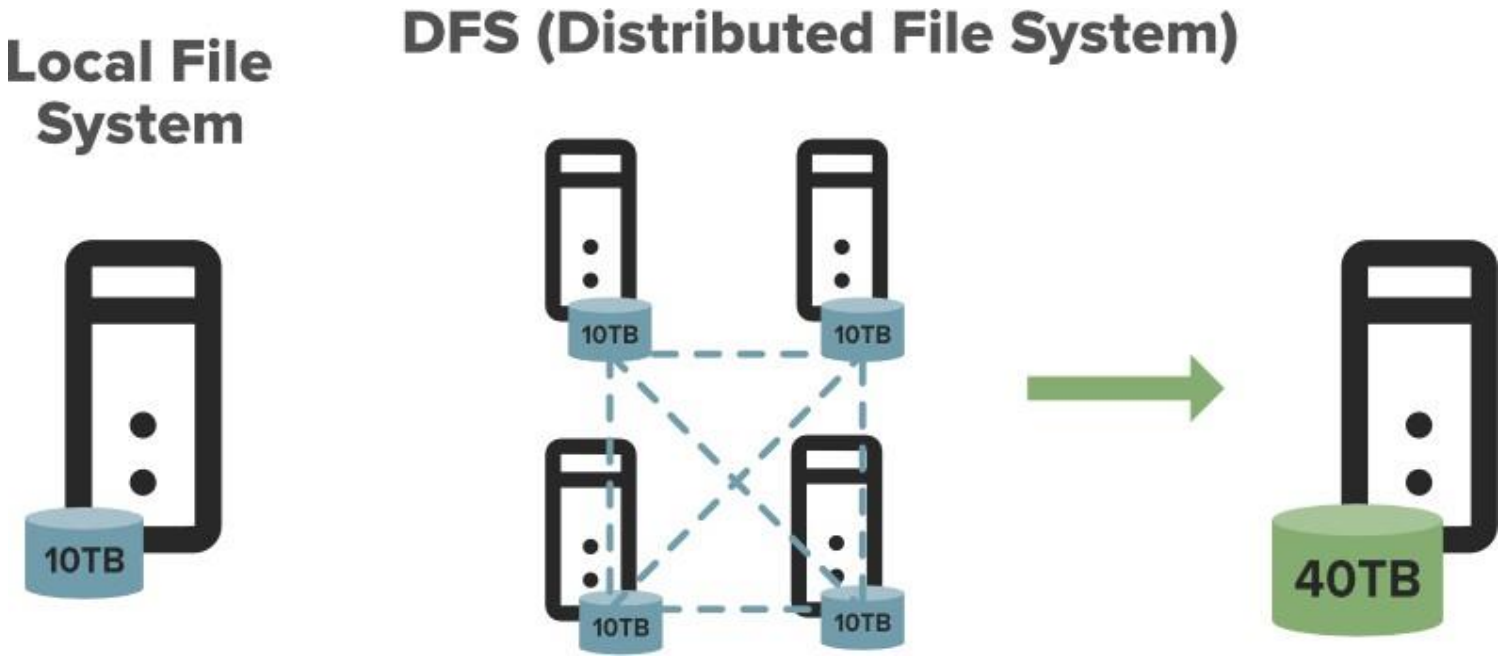
- Data-intensive computing is concerned with the production, manipulation, and analysis of large-scale data in the range of hundreds of megabytes (MB) to petabytes (PB) and beyond
- Large datasets (Big data) have mostly been the domain of scientific computing
- Relational databases and desktop statistics/visualization packages become ineffective for that amount of information, instead requiring "massively parallel software running on tens, hundreds, or even thousands of servers
- Cloud technologies support data-intensive computing in several ways:
  - By providing a large amount of compute instances on demand, which can be used to process and analyze large datasets in parallel
  - By providing a storage system optimized for keeping large blocks of data and other distributed data store architectures

Hadoop

- open-source framework that allows to store and process big data in a distributed environment across clusters of computers using simple programming models
- Instead of using one large computer to store and process the data
  - Hadoop allows clustering multiple computers to analyze massive datasets in parallel more quickly
- consists of two main modules:
  - Hadoop Distributed File System (HDFS)
    - A distributed file system that runs on standard or low-end hardware
    - HDFS provides better data throughput than traditional file systems, in addition to high fault tolerance and native support of large datasets
  - MapReduce
    - A framework that helps programs do the parallel computation on data
    - map task takes input data and converts it into a dataset that can be computed in key value pairs
    - The output of the map task is consumed by reduce tasks to aggregate output and provide the desired result

Distributed File System (DFS)

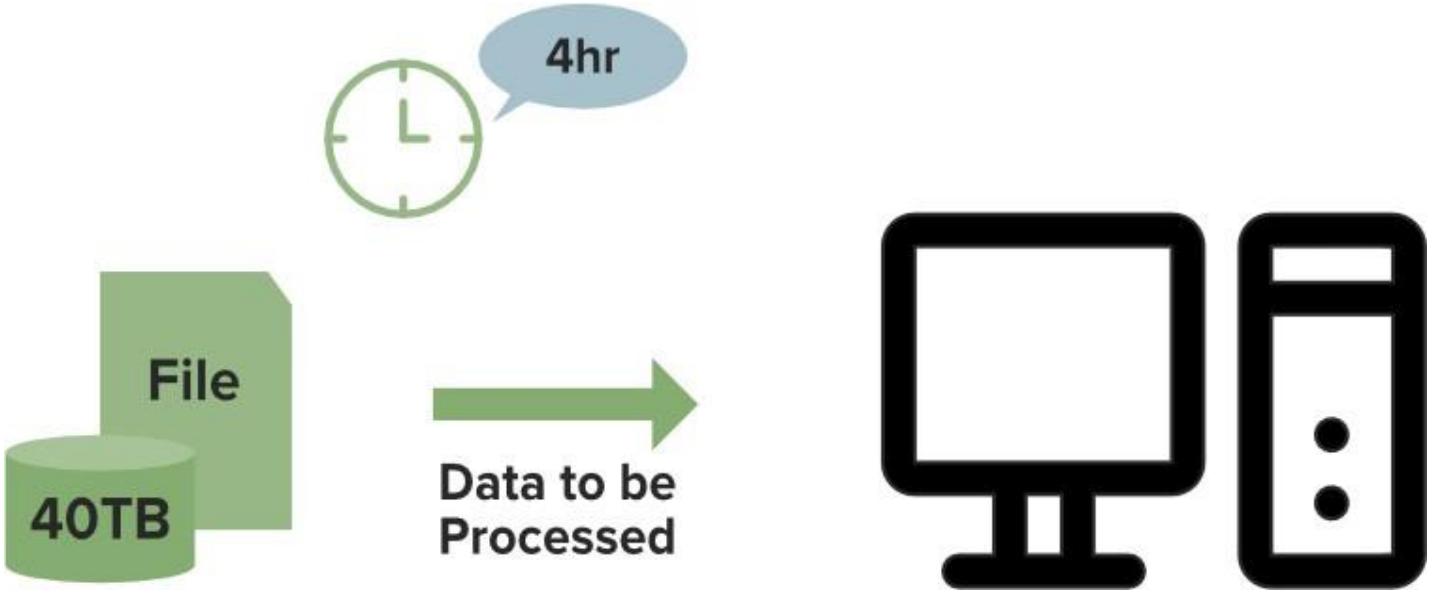
- concept of storing the file in multiple nodes in a distributed manner
- suppose you have a DFS comprised of 4 different machines each of size 10TB, i.e, 40TB



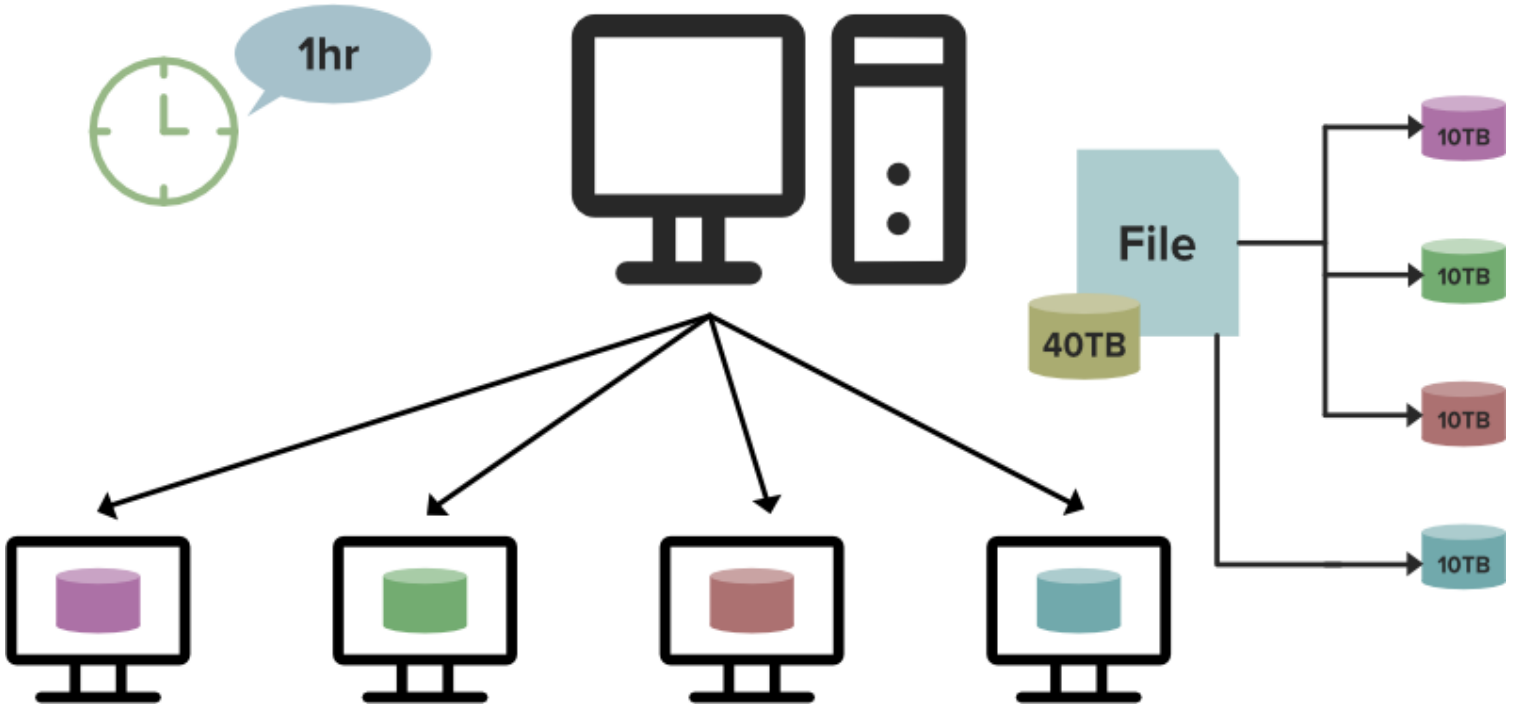
Distributed File System (DFS)

- Suppose a file of size 40TB to process
- On a single machine, it will take suppose 4hrs to process it completely
- What if you use a DFS(Distributed File System)
  - In that case, the File of size 40TB is distributed among the 4 nodes in a cluster each node stores the 10TB of file
  - These nodes are working simultaneously it will take only 1 Hour to completely process it which is Fastest
    - i.e., DFS is needed

Local File System Processing:



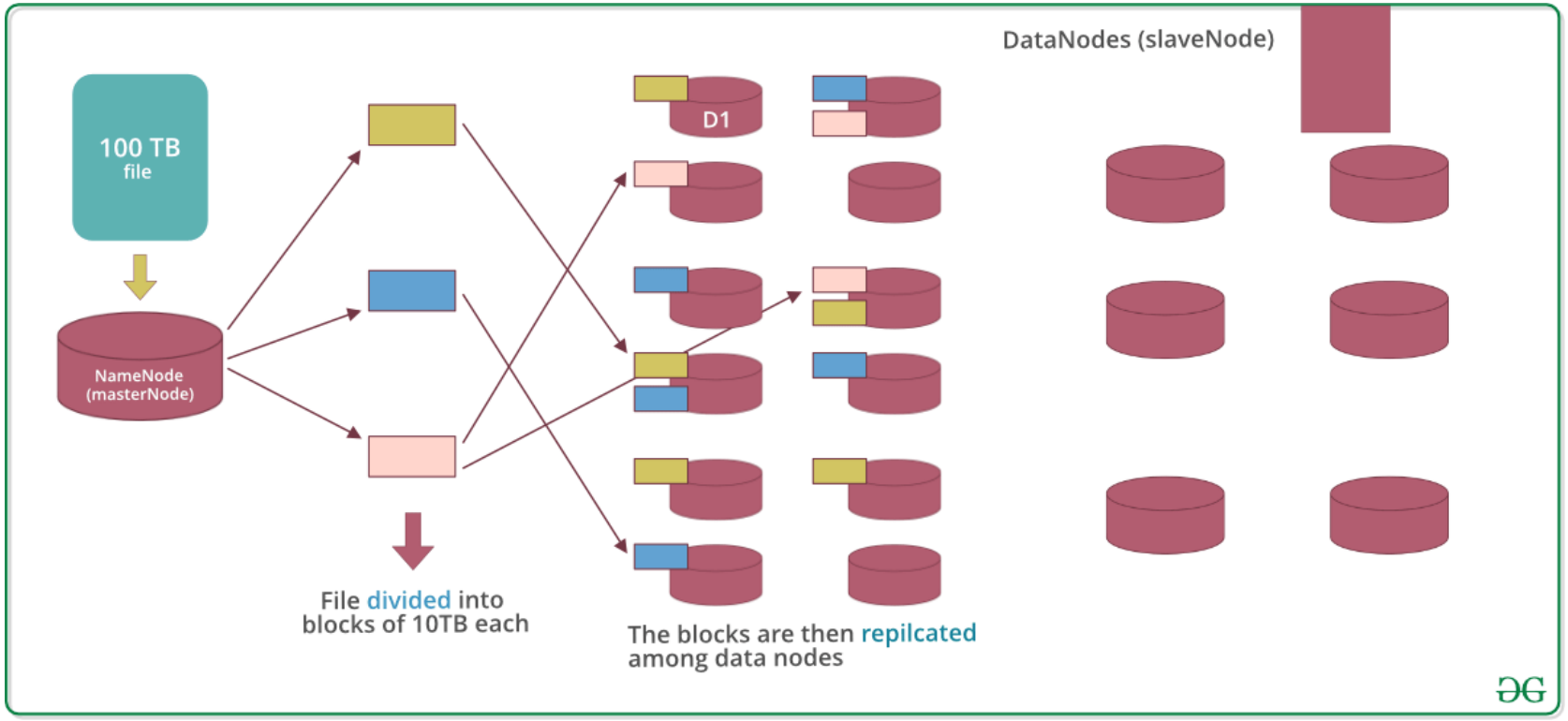
Distributed File System Processing:



Hadoop Distributed File System (HDFS)

- Data is distributed over several machines and replicated to ensure their durability to failure and high availability to parallel application in a cluster
- Cost effective as it uses low-end hardware
- HDFS is designed in such a way that it stores the data in a large chunk of blocks rather than storing small data blocks
  - size of each data block is 128MB in size
- In HDFS cluster: Master-slave nodes typically forms
  1. NameNode (Master Node)
     - Manages all the slave nodes and assign work to them
     - Executes filesystem namespace operations like opening, closing, renaming files and directories
     - Store metadata (data about data) like file path, the number of blocks, block Ids. etc.
  2. DataNode (Slave Node)
     - who does the actual work like reading, writing, processing etc.

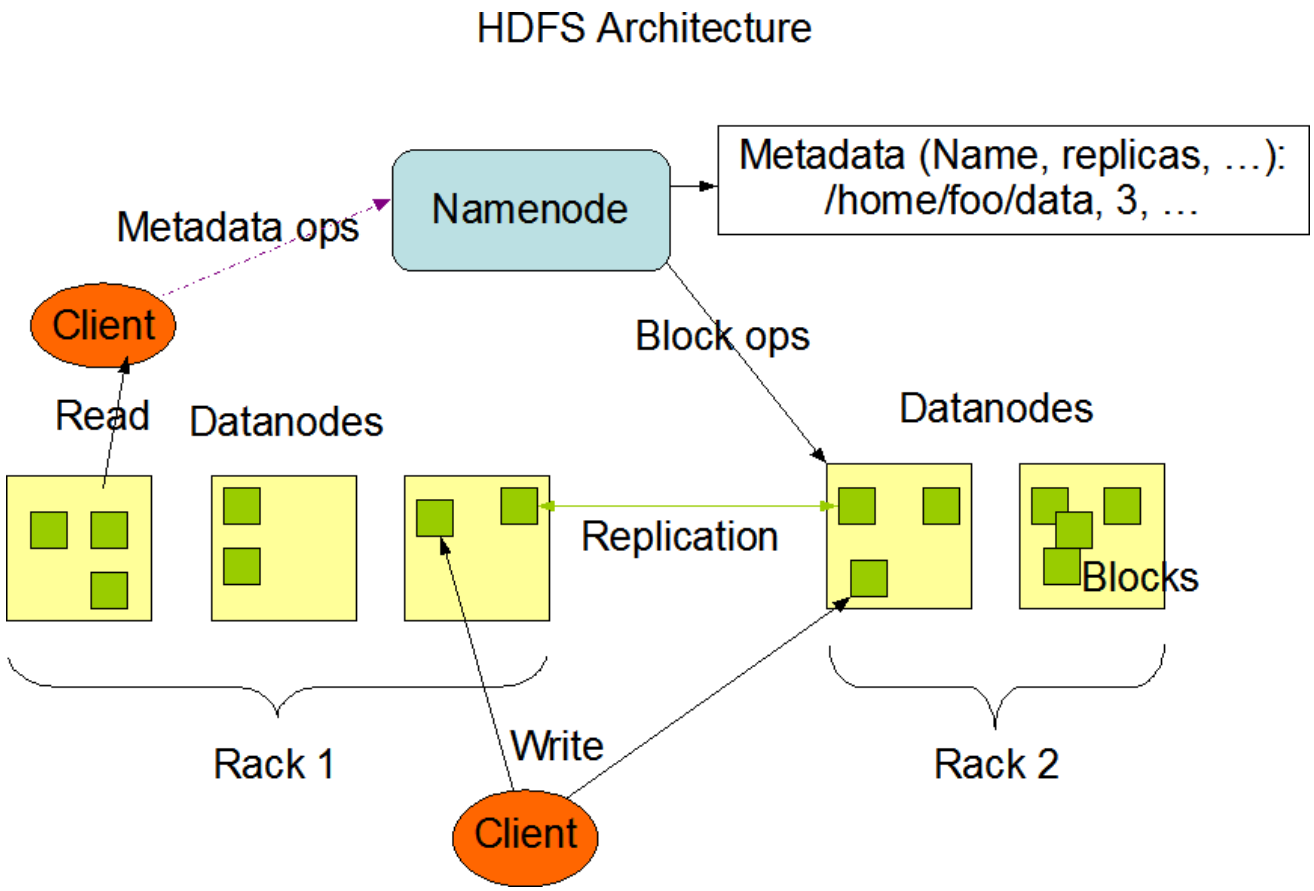
HDFS


DataNodes (slaveNode)
100 TB
file
NameNode
(masterNode)
D1
File divided into
blocks of 10TB each
The blocks are then repilcated
among data nodes

HDFS

- Assume that 100 TB file is inserted
- Masternode(namenode) will first divide the file into blocks of 10TB (default size is 128 MB)
  - blocks are stored across different datanodes(slavenode)
- Datanodes(slavenode)replicate the blocks among themselves and the information of what blocks they contain is sent to the master
- MasterNode has the record of everything, it knows the location and info of each and every single data nodes and the blocks they contain, i.e. nothing is done without the permission of masternode

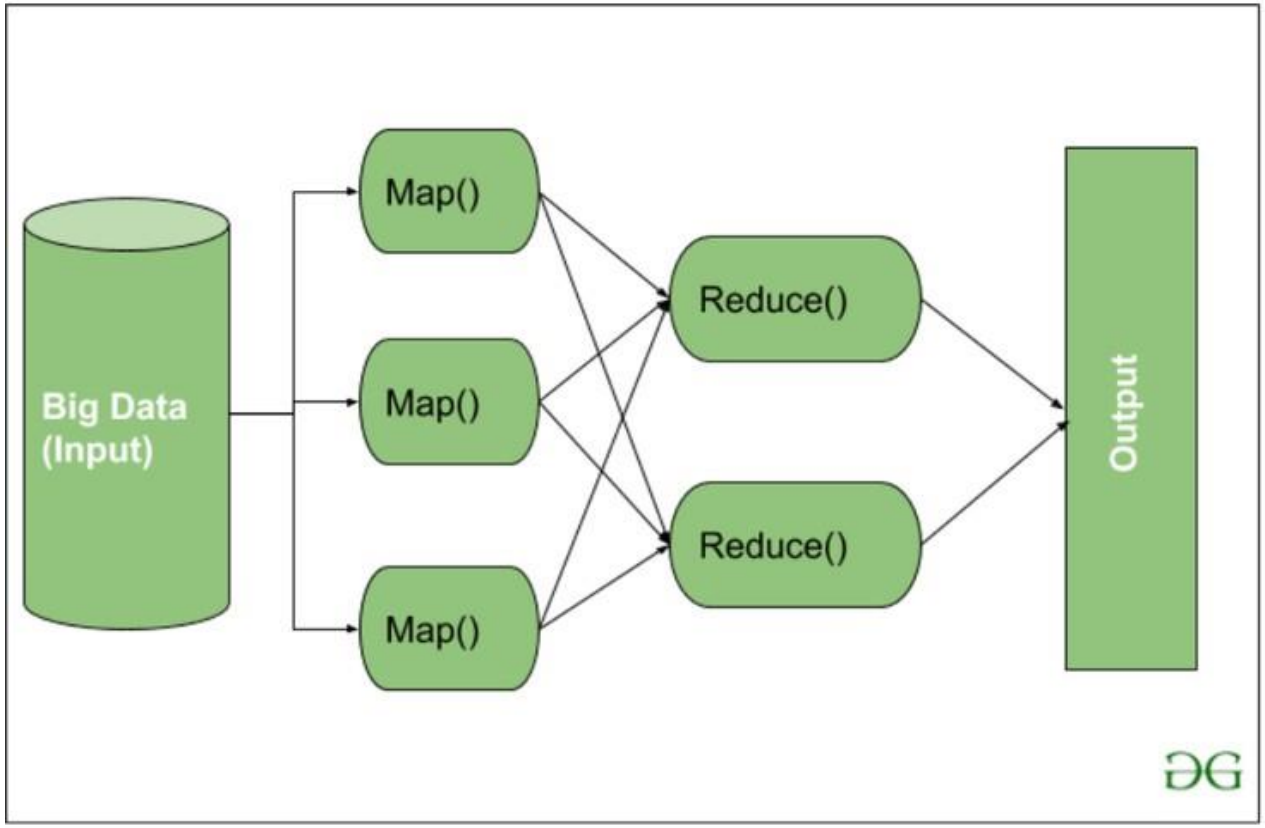
HDFS Architecture



HeartBeat: It is the signal that datanode continuously sends to namenode. If namenode doesn't receive heartbeat from a datanode then it will consider it dead.

MapReduce

- MapReduce: parallel data processing model for processing and analysis of massive scale data in distributed manner
- Two phases: Map and Reduce



MapReduce

Map Phase:

- process the input data (data are from HDFS block)
- converts data into several small chunks of data or tuples (key/value pairs)

Shuffling and Sorting Phase:

- output of various mapping parts are grouped together
- all the same values are deleted, and different values are grouped together based on same keys

Reduce Phase:

- takes the output from a shuffling and Sorting as an input and combines those data tuples into a smaller set of tuples
- produces a new set of output, which will be stored in the HDFS

	Input	Output
Map	<k1,v1>	list (<k2,v2>)
Reduce	<k2, list (v2)>	list (<k3,v3>)

MapReduce: Example