Big Data And Hadoop

By

Lokesh Singh

@SCTPL

Introduction to Big Data

What is Big Data?

What makes data, "Big" Data?

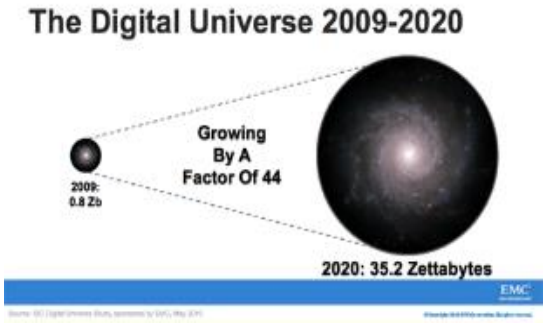
Big Data Definition

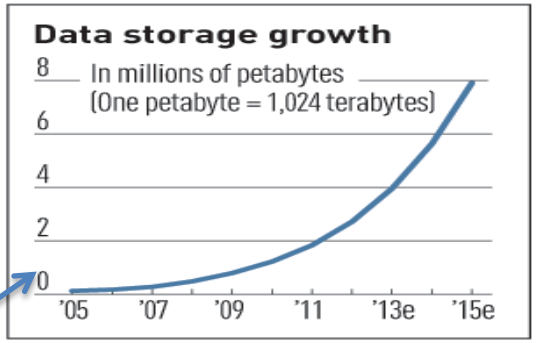
- No single standard definition…

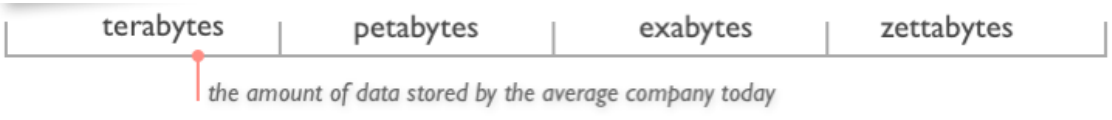
"***Big Data***" is data whose scale, diversity, and complexity require new architecture, techniques, algorithms, and analytics to manage it and extract value and hidden knowledge from it…

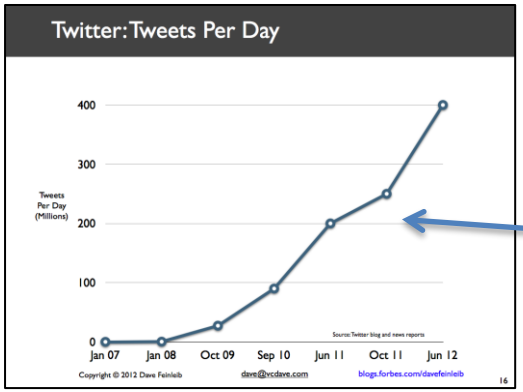
Characteristics of Big Data:
1-Scale (Volume)

- **Data Volume**
  - 44x increase from 2009
  - From 0.8 zettabytes to 35zb
- Data volume is increasing exponentially





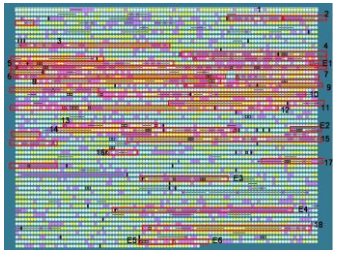


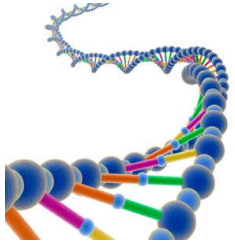


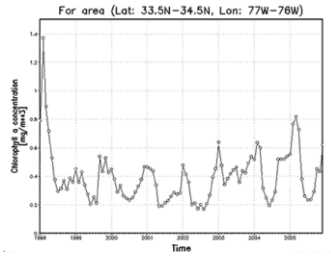
Exponential increase in collected/generated data

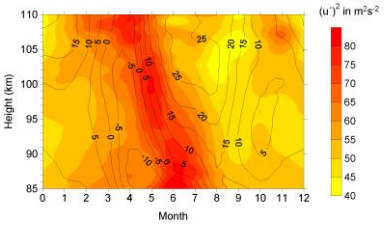
Characteristics of Big Data:
2-Complexity (Variety)

- Various formats, types, and structures
- Text, numerical, images, audio, video, sequences, time series, social media data, multi-dim arrays, etc…
- Static data vs. streaming data
- A single application can be generating/collecting many types of data









To extract knowledge➔ all these types of data need to linked together

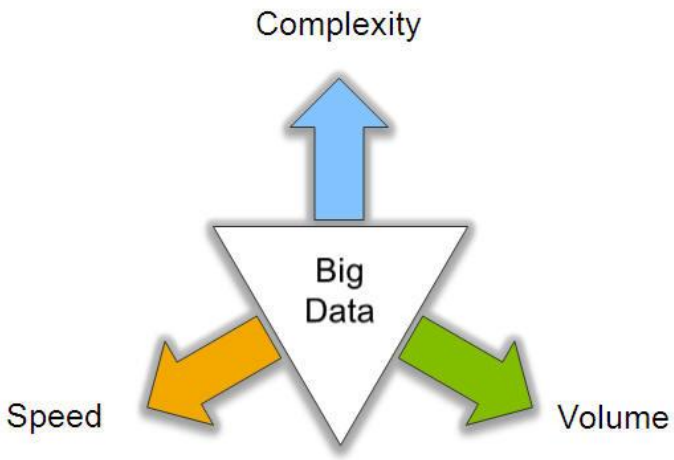
Characteristics of Big Data: 3-Speed (Velocity)

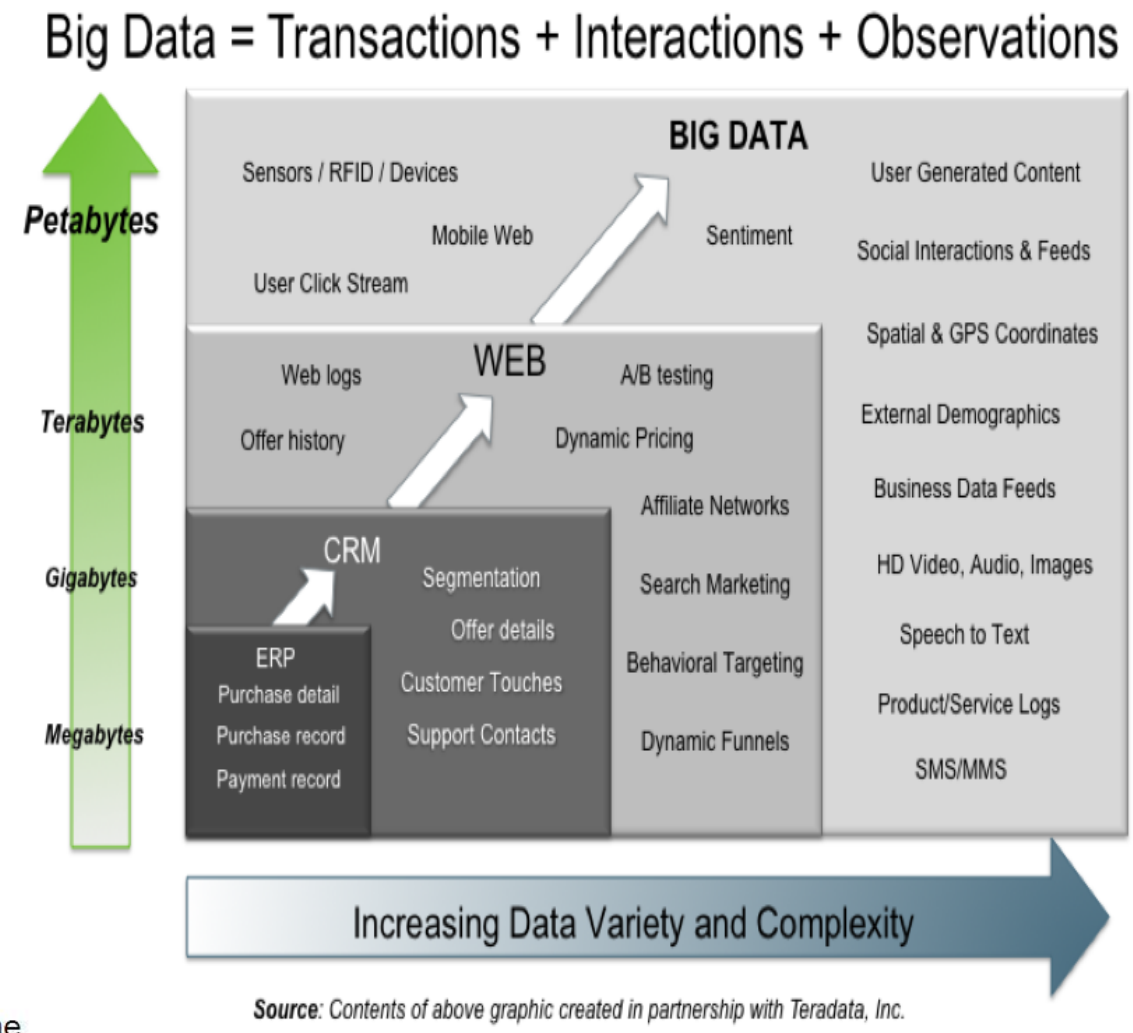
- Data is begin generated fast and need to be processed fast
- Online Data Analytics
- Late decisions ➔ missing opportunities

- **Examples**
  - **E-Promotions:** Based on your current location, your purchase history, what you like ➔ send promotions right now from store next to you.
  - **Healthcare monitoring:** sensors monitoring your activities and body ➔ any abnormal measurements require immediate reaction.

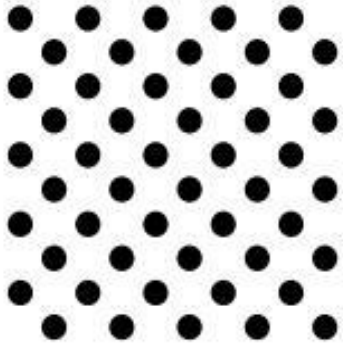
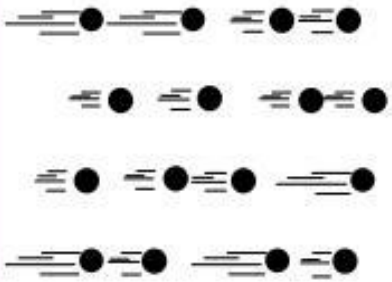
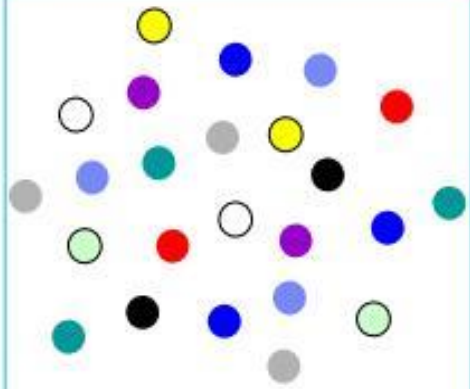
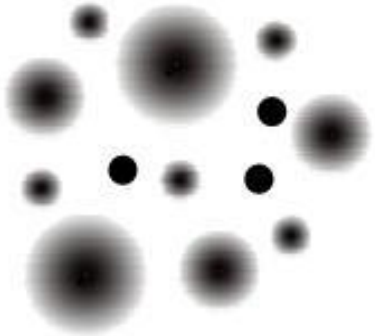
Big Data: 3V's


BIG
DATA?
VOLUME
Large amounts of data.
VELOCITY
Needs to be analyzed quickly.
VARIETY
Different types of structured and unstructured data.

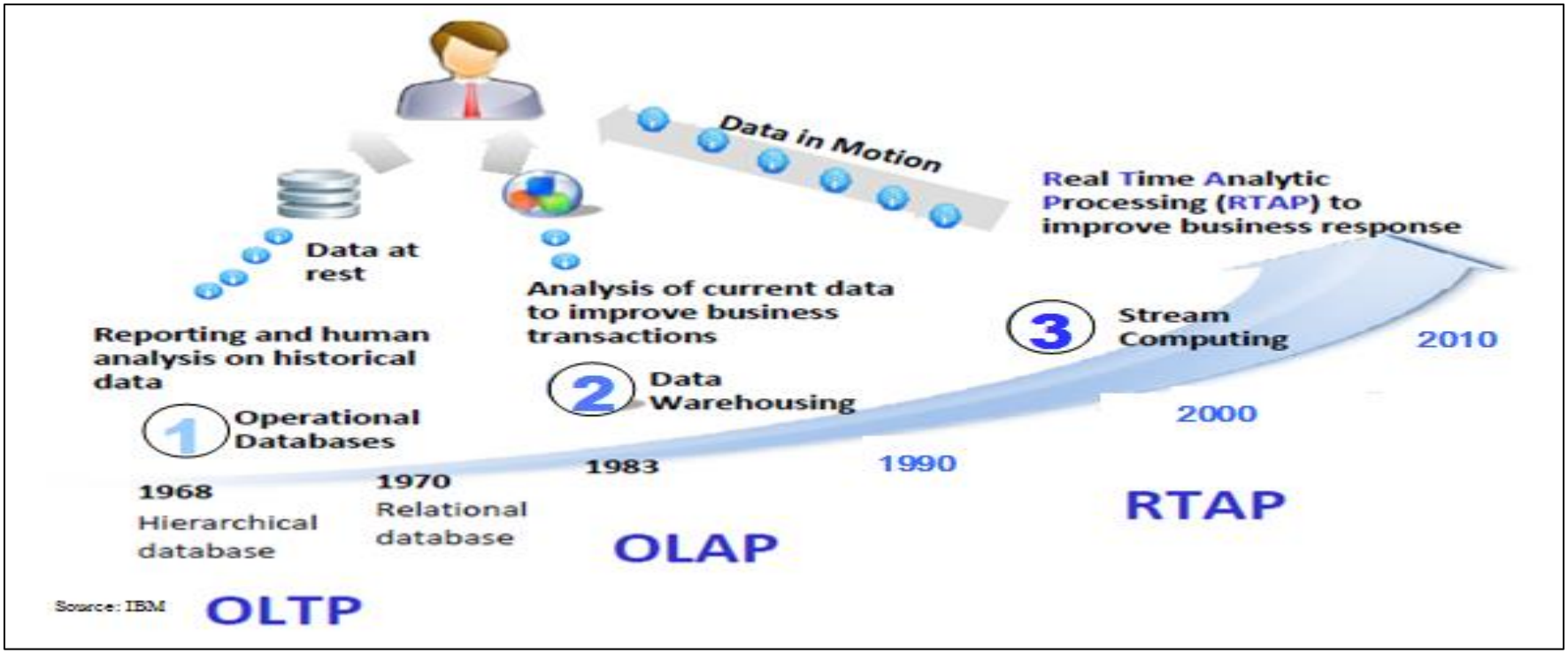

Complexity
Big Data
Speed
Volume


Big Data = Transactions + Interactions + Observations
Petabytes
Terabytes
Gigabytes
Megabytes
BIG DATA
Sensors / RFID / Devices
Mobile Web
User Click Stream
Sentiment
User Generated Content
Social Interactions & Feeds
Spatial & GPS Coordinates
External Demographics
Business Data Feeds
HD Video, Audio, Images
Speech to Text
Product/Service Logs
SMS/MMS
WEB
Web logs
Offer history
A/B testing
Dynamic Pricing
Affiliate Networks
Search Marketing
Behavioral Targeting
Dynamic Funnels
CRM
Segmentation
Offer details
Customer Touches
Support Contacts
ERP
Purchase detail
Purchase record
Payment record
Increasing Data Variety and Complexity
Source: Contents of above graphic created in partnership with Teradata, Inc.

Some Make it 4V's

Volume	Velocity	Variety	Veracity*
			
Data at Rest	**Data in Motion**	**Data in Many Forms**	**Data in Doubt**
Terabytes to exabytes of existing data to process	Streaming data, milliseconds to seconds to respond	Structured, unstructured, text, multimedia	Uncertainty due to data inconsistency & incompleteness, ambiguities, latency, deception, model approximations

Harnessing Big Data



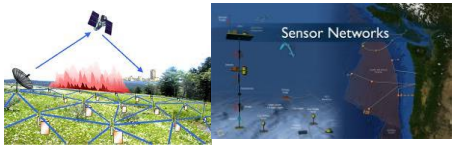
- **OLTP:** Online Transaction Processing (DBMSs)
- **OLAP:** Online Analytical Processing (Data Warehousing)
- **RTAP:** Real-Time Analytics Processing (Big Data Architecture & technology)

Who's Generating Big Data

Social media and networks
(all of us are generating data)

Scientific instruments
(collecting all sorts of data)

Mobile devices
(tracking all objects all the time)



Sensor technology and networks
(measuring all kinds of data)

- The progress and innovation is no longer hindered by the ability to collect data
- But, by the ability to manage, analyze, summarize, visualize, and discover knowledge from the collected data in a timely manner and in a scalable fashion

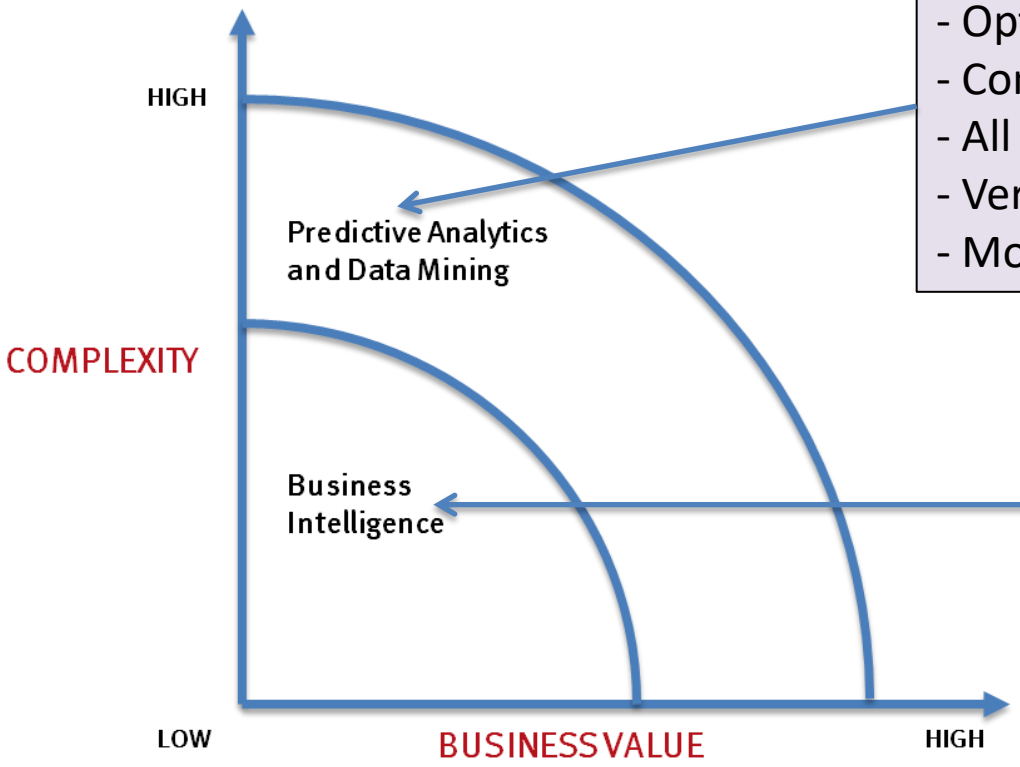
The Model Has Changed…

- **The Model of Generating/Consuming Data has Changed**

Old Model: Few companies are generating data, all others are consuming data

New Model: all of us are generating data, and all of us are consuming data

What's driving Big Data

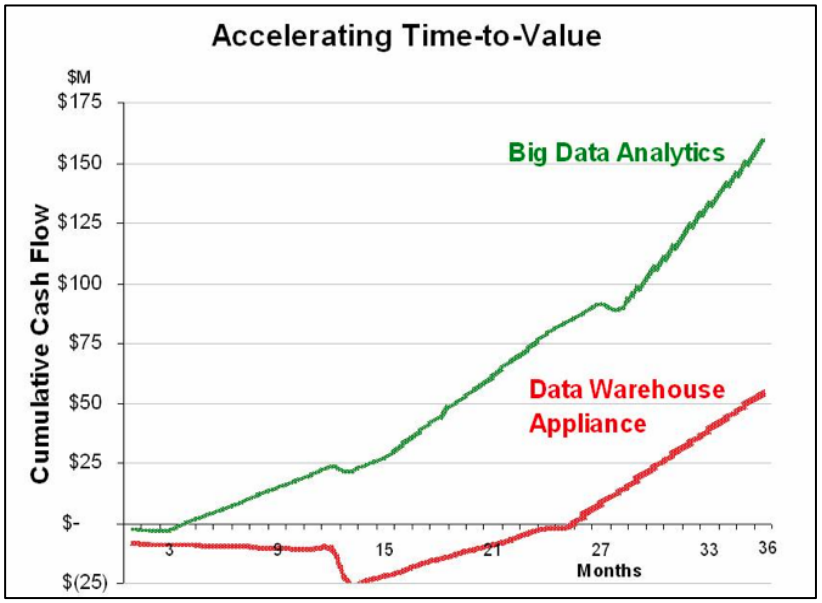


- Optimizations and predictive analytics
- Complex statistical analysis
- All types of data, and many sources
- Very large datasets
- More of a real-time

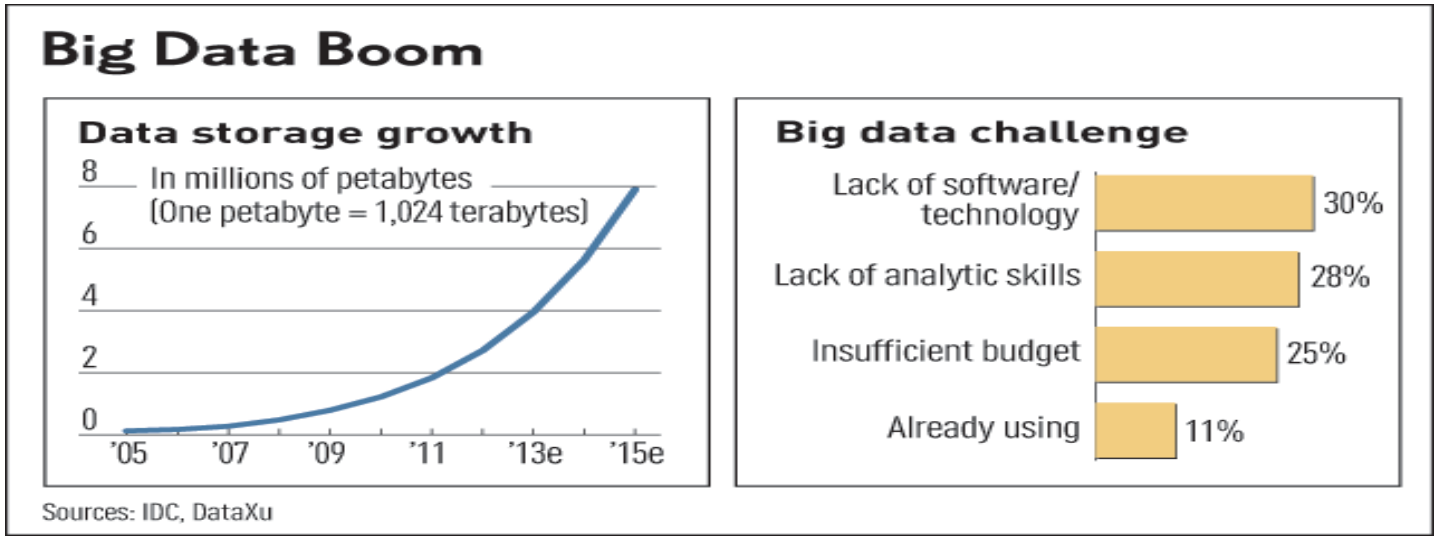
- Ad-hoc querying and reporting
- Data mining techniques
- Structured data, typical sources
- Small to mid-size datasets

Value of Big Data Analytics

- Big data is more real-time in nature than traditional DW applications
- Traditional DW architectures (e.g. Exadata, Teradata) are not well-suited for big data apps
- Shared nothing, massively parallel processing, scale out architectures are well-suited for big data apps



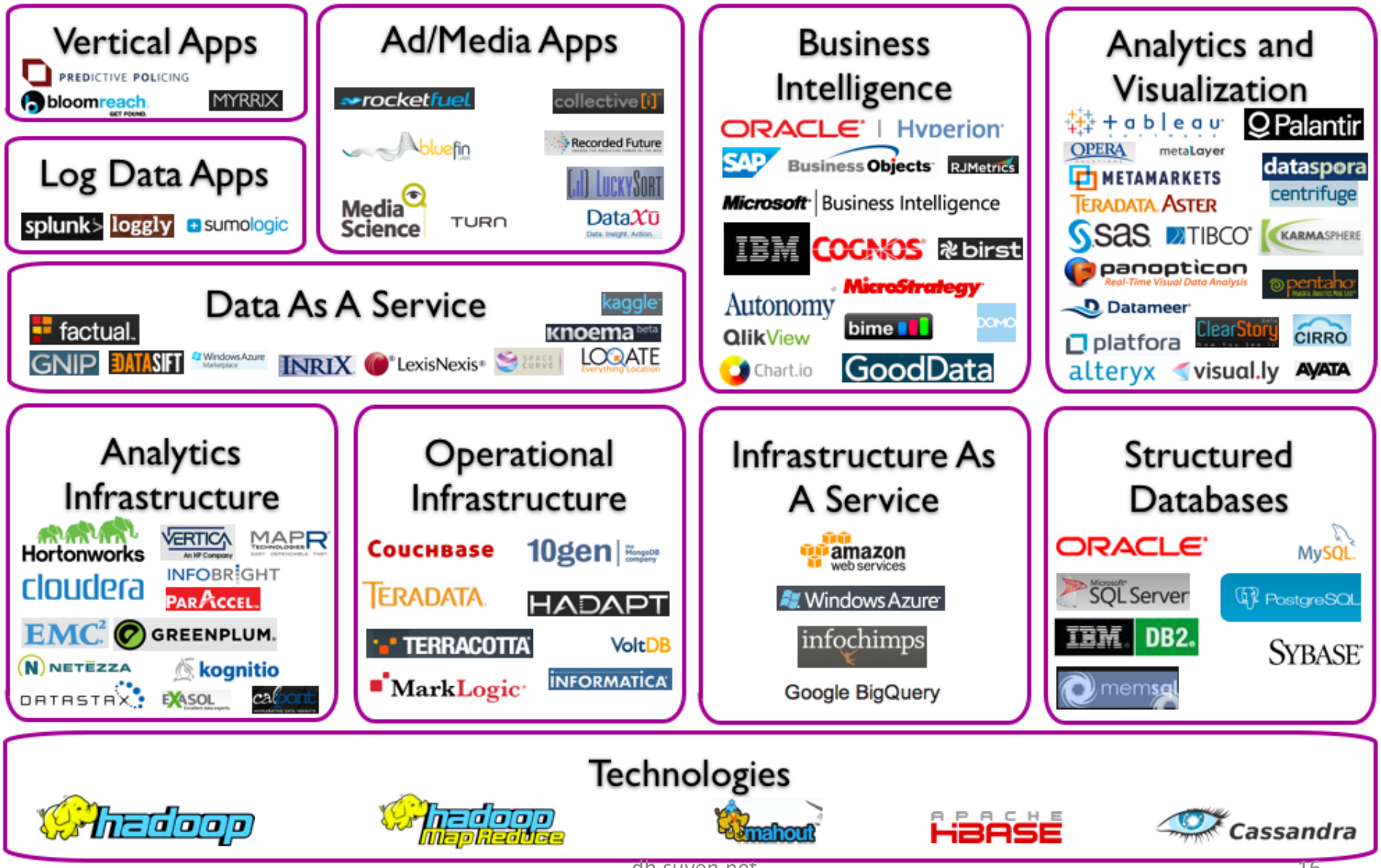
Challenges in Handling Big Data



- **The Bottleneck is in technology**
  - New architecture, algorithms, techniques are needed
- **Also in technical skills**
  - Experts in using the new technology and dealing with big data

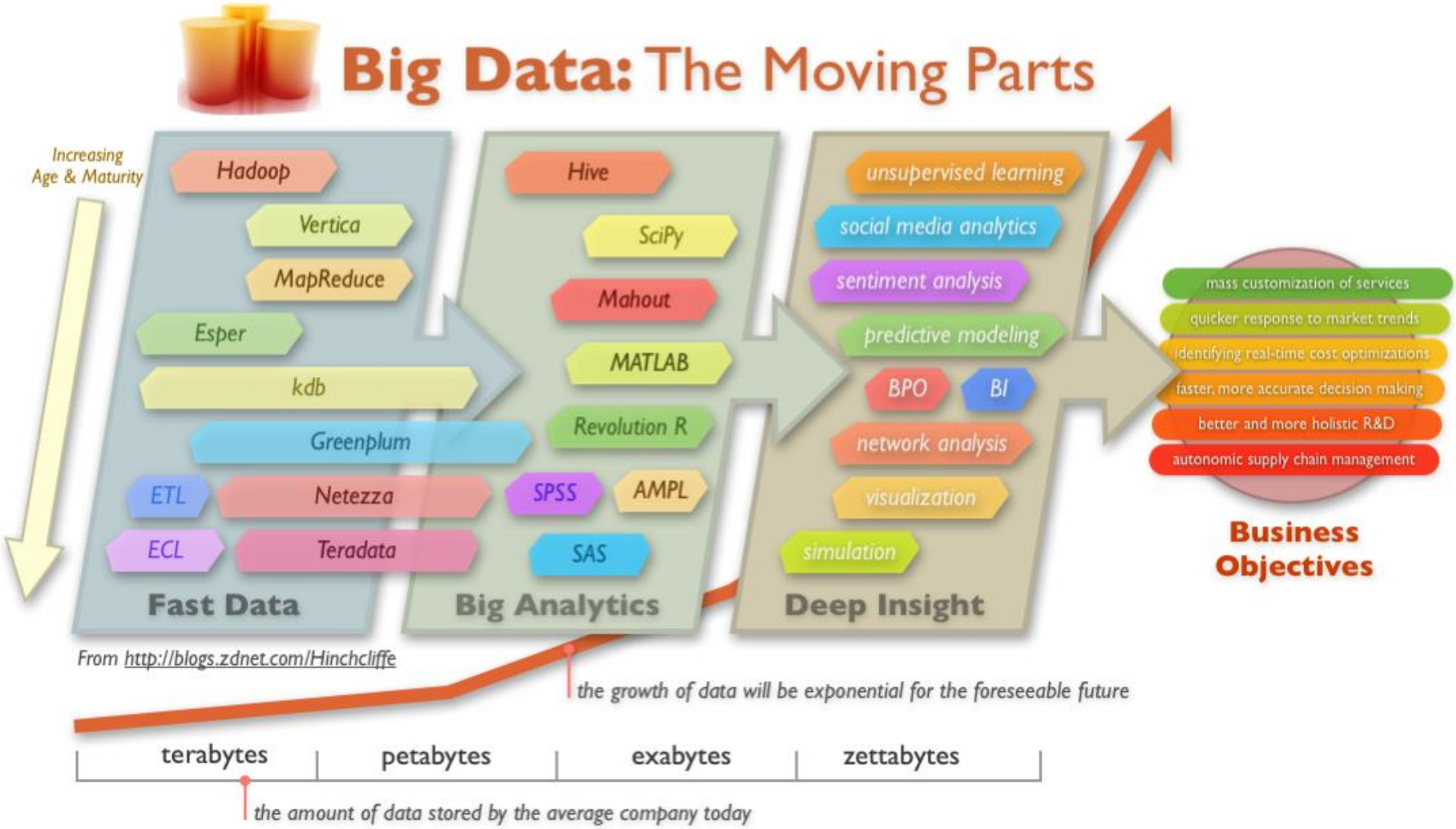
What Technology Do We Have For Big Data ??

Big Data Landscape



dave@vcdave.com
blogs.forbes.com/davefeinleib

Big Data Technology



Q/A ON BIG DATA

1)What is veracity in big data?

2)What is 3V's in big data?

3)What are tools used to analysed big data?

4)From where data is being generated largely ?

5) Does Big Data cause problem to bigger firm ?

6) How Big Data Can be handle ?

7) What can be the exact size of big data ?

Hadoop EcoSystem And Architecture

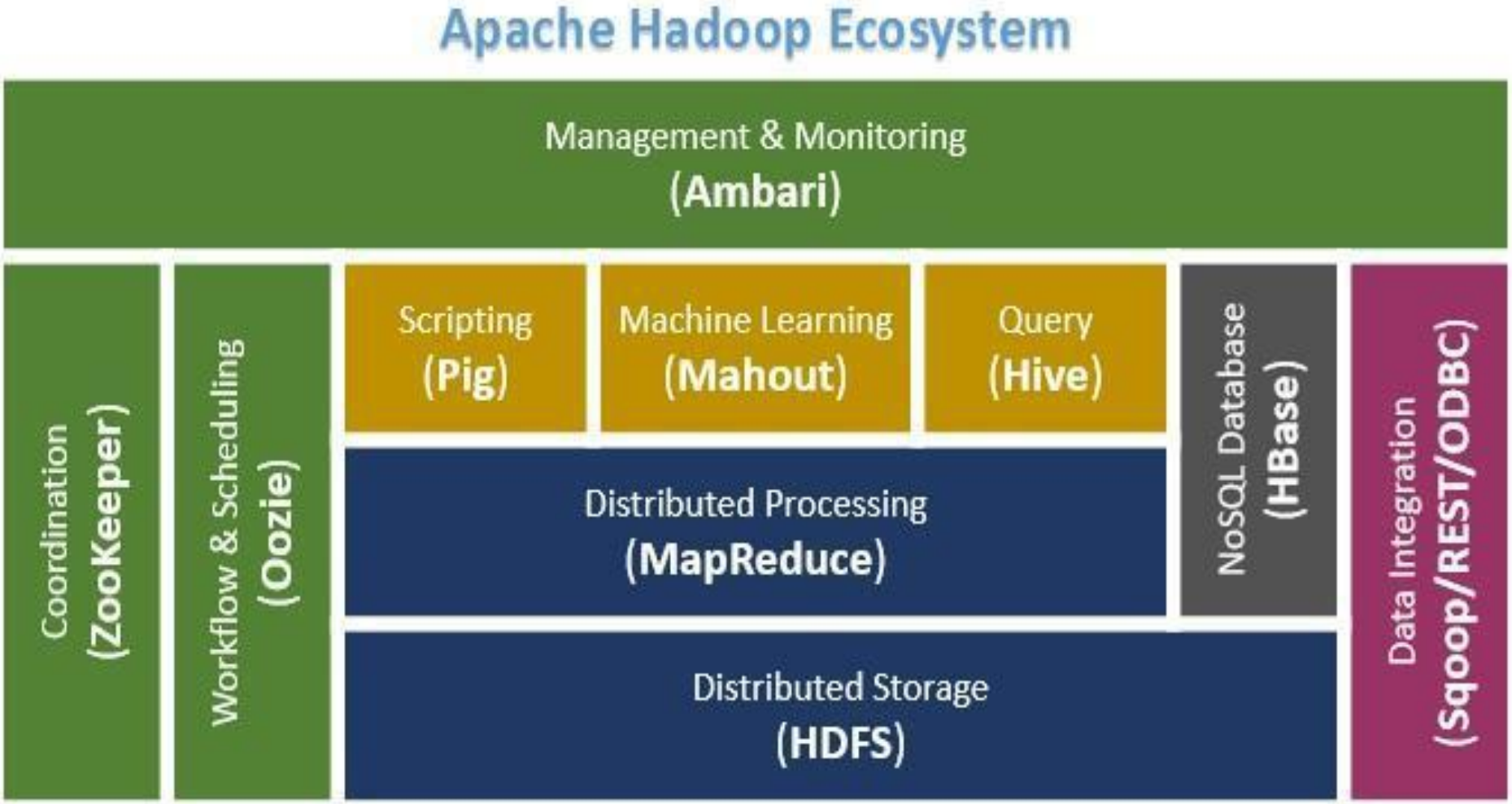
What is hadoop and what is its characteristics

"Apache Hadoop is an open-source software framework for storage and large-scale processing of data-sets on clusters of commodity hardware."

Characteristics

- Open source
- Distributed processing
- Distributed storage
- Scalable
- Reliable
- Fault-tolerant
- Economical
- Flexible

Hadoop Ecosystem And Explanation



Note: Hadoop knows only Jar files. Hadoop don't know other things.

Ecosystem continue..

Ambari: An integrated set of Hadoop administration tools for installing, monitoring, and maintaining a Hadoop cluster. Also included are tools to add or remove slave nodes.

Avro: A framework for the efficient **serialization** (a kind of transformation) of data into a compact binary format

Flume: A data flow service for the movement of large volumes of **log data** into Hadoop

HBase: A distributed columnar database that uses HDFS for its underlying storage. With HBase, you can store data in extremely **large tables** with variable column structures.

HCatalog:A service for providing a **relational view** of data stored in Hadoop, including a standard approach for tabular data.

HiveQL: A distributed data warehouse for data that is stored in HDFS; also provides a **query language that's based on SQL** (HiveQL).

Ecosystem continue..

Hue: A Hadoop administration interface with handy **GUI tools for browsing files, issuing Hive and Pig queries, and developing Oozie workflows.**

Mahout: A library of **machine learning statistical algorithms** that were implemented in MapReduce and can run natively on Hadoop.

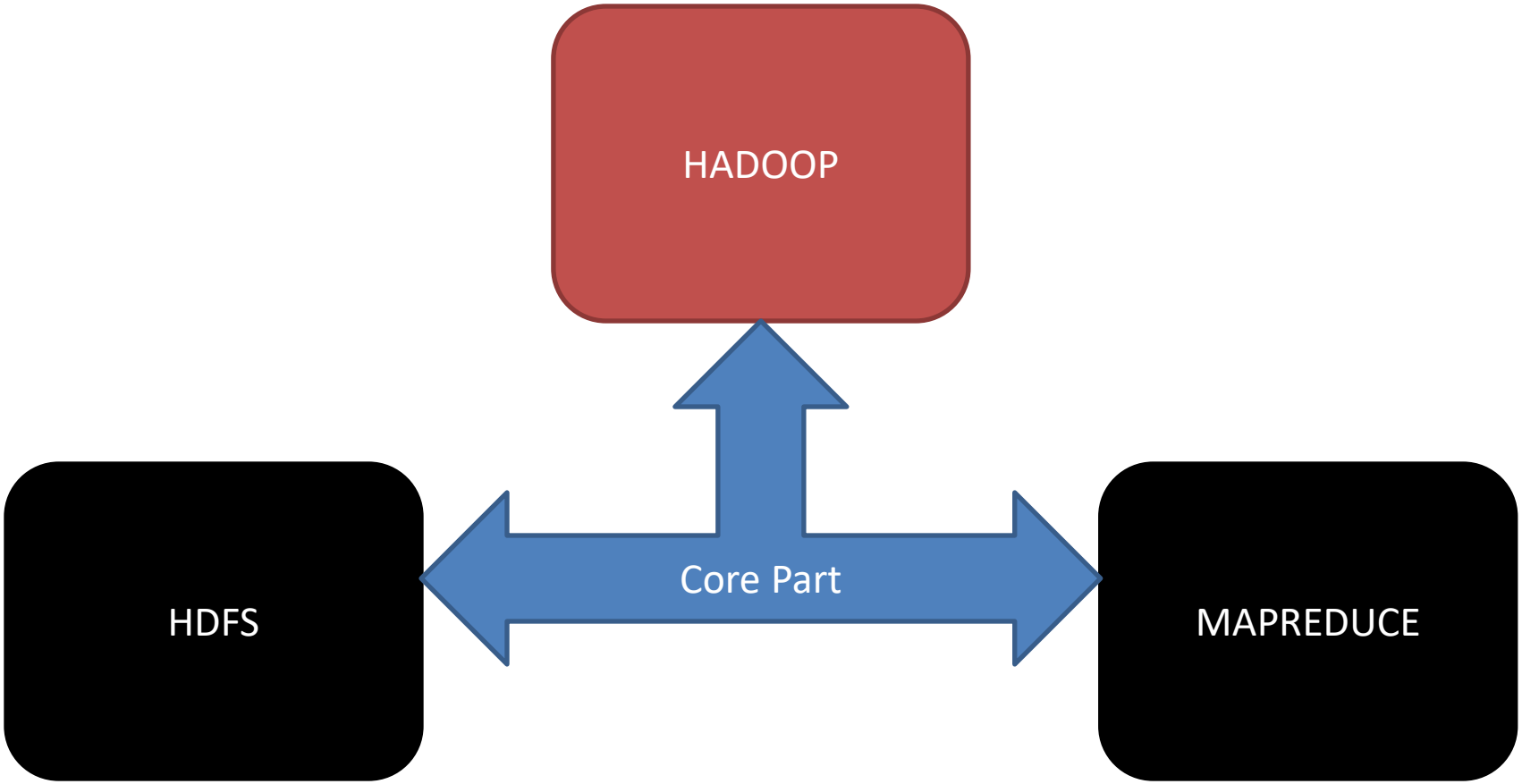
Oozie: A **workflow management tool** that can handle the scheduling and chaining together of Hadoop applications.

Pig : A platform for the **analysis of very large data sets that runs on HDFS** and with an infrastructure layer consisting of a compiler that produces sequences of MapReduce programs and a language layer consisting of the query language named Pig Latin.

Sqoop : A tool for efficiently **moving large amounts of data between relational databases and HDFS.**

Zookeeper: A simple interface to the centralized **coordination/monitor of services** (such as naming, configuration, and synchronization) used by distributed applications.

Hadoop core part



Hadoop core part is HDFS and MAPREDUCE.

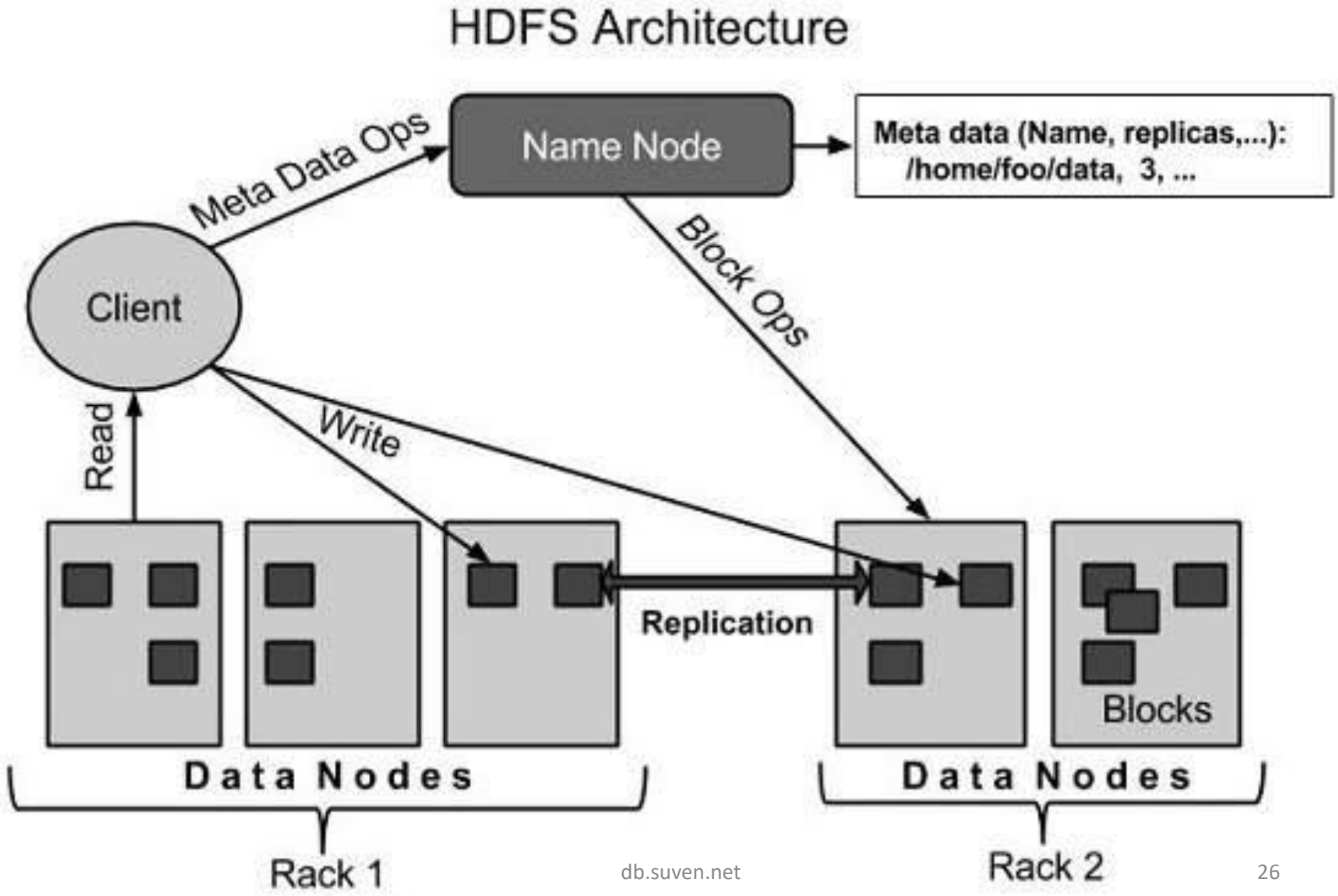
HDFS (Hadoop distributed file system)

HDFS is "***specially designed file system for storing huge datasets with cluster of commodity hardware***"

Defination

Cluster	Group of Systems connected to LAN
Commodity H/W	Cheap Hardware Things
Streaming access pattern data	write once – Read any number of time but Don't change

HDFS Architecture



1) HDFS is a distributed file system that is fault tolerant, scalable and extremely easy to expand.

2) HDFS is the primary distributed storage for Hadoop applications.

3) HDFS provides interfaces for applications to move themselves closer to data.

4) HDFS is designed to ‘just work’, however a working knowledge helps in diagnostics and improvements.

Components of HDFS

There are two (*and a half*) types of machines in a HDFS cluster

NameNode :– is the heart of an HDFS filesystem, it maintains and manages the file system metadata. E.g; what blocks make up a file, and on which datanodes those blocks are stored.

DataNode :- where HDFS stores the actual data, there are usually quite a few of these.

NOTE:- Namenode and Datanode are further explained in details ..

Blocks

A disk has a block size, which is the minimum amount of data that it can read or write. Filesystems for a single disk build on this by dealing with data in blocks, which are an integral multiple of the disk block size.

Filesystem blocks are typically a few kilobytes in size, whereas disk blocks are normally 512 bytes.

We have to install HDFS file system on the top of Linux. Linux block size is 4kb.If we want store 2kb of data in Linux, another 2kb of data will be wasted

File system is divided into blocks.

By default block size is **64mb (or even 128MB).**

If we store 32mb of data, reaming 32mb of data will we used for another purpose.

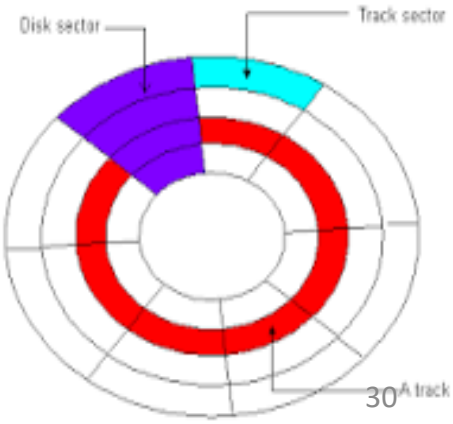
That's why it is called as 'Specially Designed File System'

Why Is a Block in HDFS So Large?

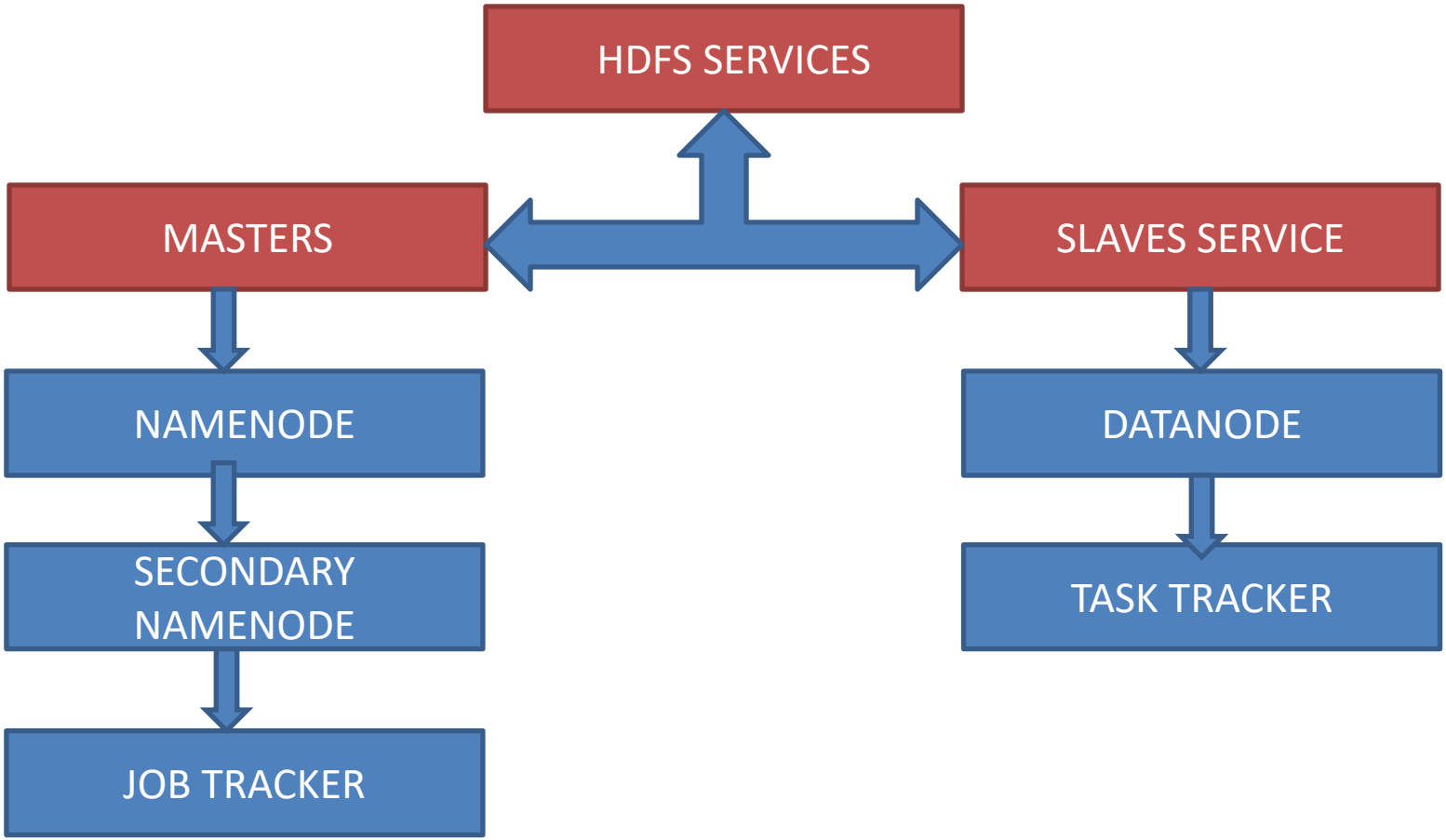
HDFS blocks are large compared to disk blocks, and the reason is to minimize the cost of seeks. By making a block large enough, the time to transfer the data from the disk can be significantly longer than the time to seek to the start of the block. Thus the time to transfer a large file made of multiple blocks operates at the disk transfer rate.

A quick calculation shows that if the seek time is around 10 ms and the transfer rate is 100 MB/s, to make the seek time 1% of the transfer time, we need to make the block size around 100 MB. The default is actually 64 MB, although many HDFS installations use 128 MB blocks. This figure will continue to be revised upward as transfer speeds grow with new generations of disk drives.

Seek Time is measured defines the amount of **time** it takes a **hard drive's** read/write head to find the physical location of a piece of data on the **disk**. Latency is the average **time** for the sector being accessed to rotate into position under a head, after a completed **seek**.



HDFS Services



Hdfs services are of two type master service and slave service .
Master services are namenode , secondary namenode and jobtracker whereas slave are datanodes and tasktracker

NAMENODE

The namenode is the commodity hardware that contains the GNU/Linux operating system and the namenode software. It is a software that can be run on commodity hardware. The system having the namenode acts as the master server and it does the following tasks:

•Manages the file system namespace.

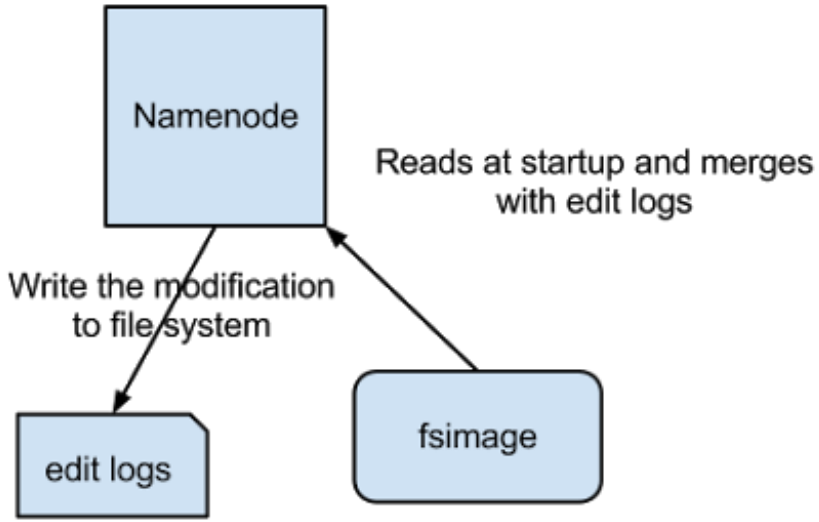
•Regulates client's access to files.

•It also executes file system operations such as renaming, closing, and opening files and directories.

Namenode holds the meta data for the HDFS like Namespace information, block information etc. When in use, all this information is stored in main memory. But these information also stored in disk for persistence storage.

NAMENODE CONTINUE....

The above image shows how Name Node stores information in disk.
Two different files are
1.fsimage - Its the snapshot of the filesystem when namenode started
2.Edit logs - Its the sequence of changes made to the filesystem after namenode started



Only in the restart of namenode , edit logs are applied to fsimage to get the latest snapshot of the file system. But namenode restart are rare in production clusters which means edit logs can grow very large for the clusters where namenode runs for a long period of time. The following issues we will encounter in this situation.

1.Editlog become very large , which will be challenging to manage it

2.Namenode restart takes long time because lot of changes has to be merged

3.In the case of crash, we will lost huge amount of metadata since fsimage is very old

NAMENODE CONTINUE….

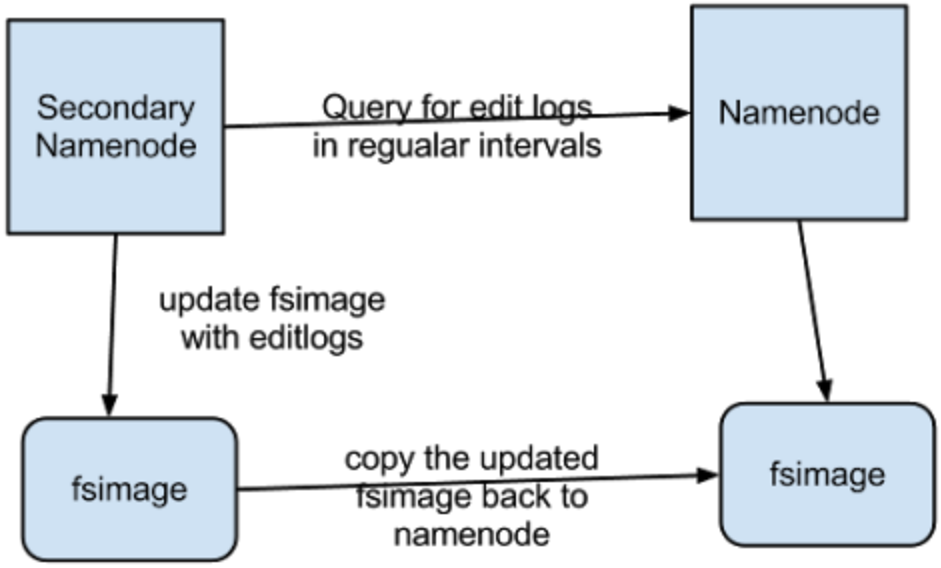
So to overcome this issues we need a mechanism which will help us reduce the edit log size which is manageable and have up to date fsimage ,so that load on namenode reduces .

It’s very similar to Windows Restore point, which will allow us to take snapshot of the OS so that if something goes wrong , we can fallback to the last restore point.

This explains NameNode functionality and challenges to keep the meta data up to date.

SECONDARY NAMENODE

Secondary Namenode helps to overcome the above issues by taking over responsibility of merging editlogs with fsimage from the namenode.



The above figure shows the working of Secondary Namenode:-
1.It gets the edit logs from the namenode in regular intervals and applies to fsimage

2.Once it has new fsimage, it copies back to namenode

3.Namenode will use this fsimage for the next restart,which will reduce the startup time

Secondary Namenode whole purpose is to have a checkpoint in HDFS. Its just a helper node for namenode.That's why it also known as checkpoint node inside the community.

So we now understood all Secondary Namenode does puts a checkpoint in

DATANODE

The datanode is a commodity hardware having the GNU/Linux operating system and datanode software. For every node (Commodity hardware/System) in a cluster, there will be a datanode. These nodes manage the data storage of their system.

•Datanodes perform read-write operations on the file systems, as per client request.

•They also perform operations such as block creation, deletion, and replication according to the instructions of the namenode.

DATANODE CONTINUE…

1. DataNode is responsible for storing the actual data in HDFS.

2. DataNode is also known as the Slave

3. NameNode and DataNode are in constant communication.

4. When a DataNode starts up it announce itself to the NameNode along with the list of blocks it is responsible for.

5. When a DataNode is down, it does not affect the availability of data or the cluster. NameNode will arrange for replication for the blocks managed by the DataNode that is not available.

6. DataNode is usually configured with a lot of hard disk space. Because the actual data is stored in the DataNode.

JOB TRACKER

•The primary function of the job tracker is resource management (managing the task trackers), tracking resource availability and task life cycle management (tracking its progress, fault tolerance etc.)

•The task tracker has a simple function of following the orders of the job tracker and updating the job tracker with its progress status periodically.

•The task tracker is pre-configured with a number of slots indicating the number of tasks it can accept. When the job tracker tries to schedule a task, ***it looks for an empty slot in the tasktracker running on the same server which hosts the datanode where the data for that task resides. If not found, it looks for the machine in the same rack***. There is no consideration of system load during this allocation.

•***HDFS is rack aware*** in the sense that the namenode and the job tracker obtain a list of rack ids corresponding to each of the slave nodes (data nodes) and creates a mapping between the IP address and the rack id. HDFS uses this knowledge to replicate data across different racks so that data is not lost in the event of a complete rack power outage or switch failure.

JOB TRACKER CONTINUE…

•***Job Performance*** - Hadoop does speculative execution where if a machine is slow in the cluster and the map/reduce tasks running on this machine are holding on to the entire map/reduce phase, then it runs redundant jobs on other machines to process the same task, and whichever task gets completed first reports back to the job tracker and results from the same are carried forward into the next phase.

•***Fault Tolerance -***

- The task tracker spawns different JVM processes to ensure that process failures do not bring down the task tracker.

- The task tracker keeps sending heartbeat messages to the job tracker to say that it is alive and to keep it updated with the number of empty slots available for running more tasks.

- From version 0.21 of Hadoop, the job tracker does some checkpointing of its work in the filesystem. Whenever, it starts up it checks what was it upto till the last CP and resumes any incomplete jobs. Earlier, if the job tracker went down, all the active job information used to get lost.

•The status and information about the job tracker and the task tracker are exposed vis jetty onto a web interface.

JOB TRACKER CONTINUE…

1.JobTracker process runs on a separate node and not usually on a DataNode.

2.JobTracker is an essential Daemon for MapReduce execution in MRv1. It is replaced by ResourceManager/ApplicationMaster in MRv2.

3.JobTracker receives the requests for MapReduce execution from the client.

4.JobTracker talks to the NameNode to determine the location of the data.

5.JobTracker finds the best TaskTracker nodes to execute tasks based on the data locality (proximity of the data) and the available slots to execute a task on a given node.

6.JobTracker monitors the individual TaskTrackers and the submits back the overall status of the job back to the client.

7.JobTracker process is critical to the Hadoop cluster in terms of MapReduce execution.

8.When the JobTracker is down, HDFS will still be functional but the MapReduce execution can not be started and the existing MapReduce jobs will be halted.

TASK TRACKER

1. TaskTracker runs on DataNode. Mostly on all DataNodes.

2. TaskTracker is replaced by Node Manager in MRv2.

3. Mapper and Reducer tasks are executed on DataNodes administered by TaskTrackers.

4. TaskTrackers will be assigned Mapper and Reducer tasks to execute by JobTracker.

5. TaskTracker will be in constant communication with the JobTracker signalling the progress of the task in execution.

6. TaskTracker failure is not considered fatal. When a TaskTracker becomes unresponsive, JobTracker will assign the task executed by the TaskTracker to another node.

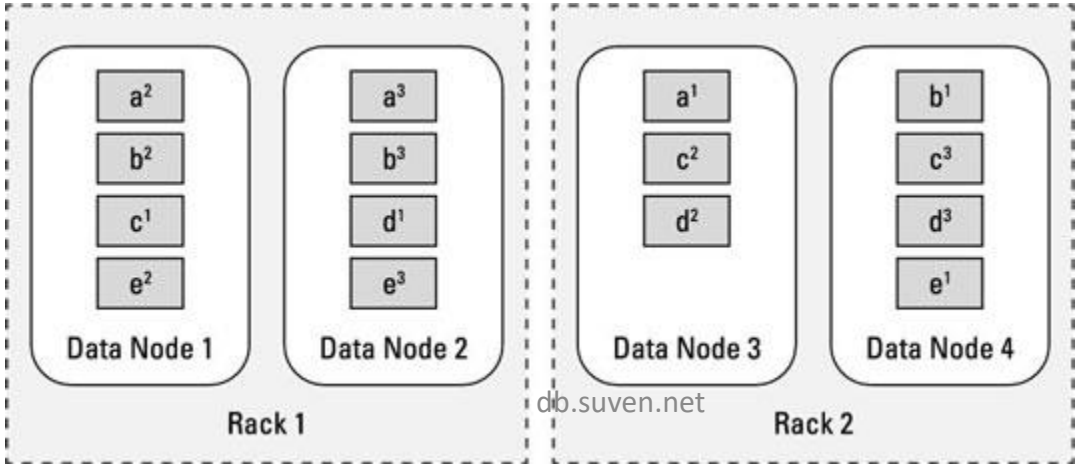
REPLICATION

Hadoop Distributed File System (HDFS) is designed to store data on inexpensive, and more unreliable, hardware. *Inexpensive* has an attractive ring to it, but it does raise concerns about the reliability of the system as a whole, especially for ensuring the high availability of the data.

Planning ahead for disaster, the brains behind HDFS made the decision to set up the system so that it would store three **(count 'em — three)** copies of every data block.

HDFS assumes that every disk drive and every slave node is inherently unreliable, so, clearly, care must be taken in choosing where the three copies of the data blocks are stored.

The figure shows how data blocks from the earlier file are *striped* across the Hadoop cluster — meaning they are evenly distributed between the slave nodes so that a copy of the block will still be available regardless of disk, node, or rack failures.



REPLICATION CONTINUE..

The file shown has five data blocks, labeled a, b, c, d, and e. If you take a closer look, you can see this particular cluster is made up of two racks with two nodes apiece, and that the three copies of each data block have been spread out across the various slave nodes.

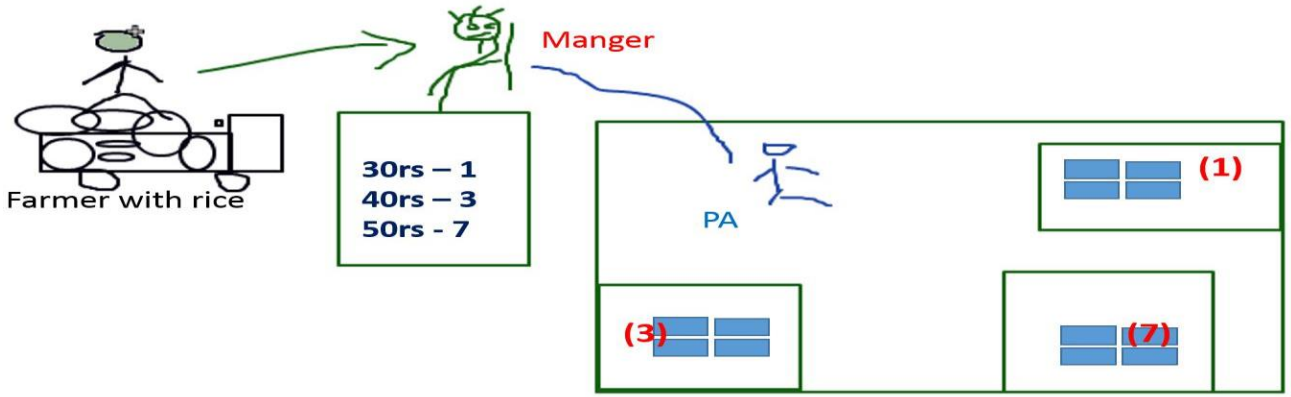
Every component in the Hadoop cluster is seen as a potential failure point, so when HDFS stores the replicas of the original blocks across the Hadoop cluster, it tries to ensure that the block replicas are stored in different failure points.

For example, take a look at Block A. At the time it needed to be stored, Slave Node 3 was chosen, and the first copy of Block A was stored there. For multiple rack systems, HDFS then determines that the remaining two copies of block A need to be stored in a different rack. So the second copy of block A is stored on Slave Node 1.

The final copy can be stored on the same rack as the second copy, but not on the same slave node, so it gets stored on Slave Node 2.

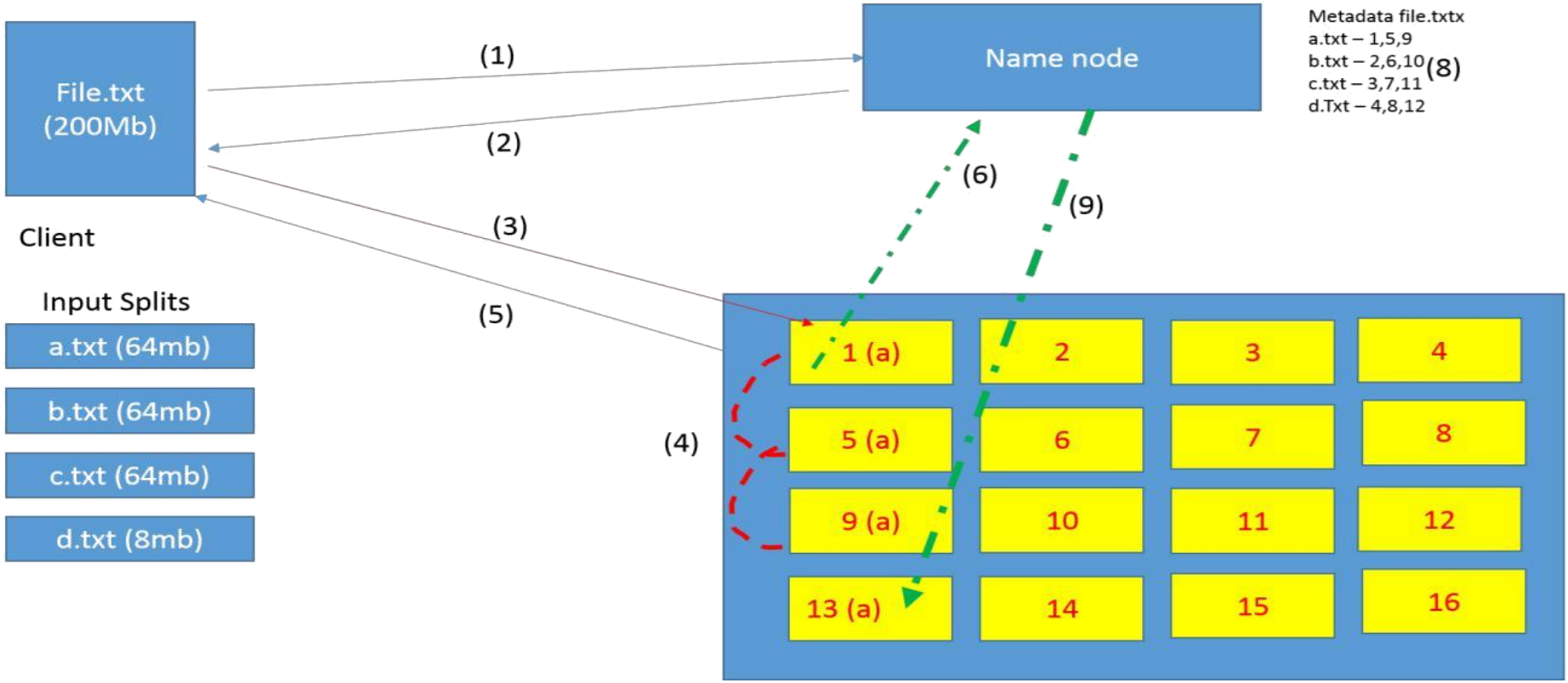
Concluding HDFS with farmer-godown story example to understand concept in depth

One day a farmer want to store his rice packets into a godown. He went to godown and contacted manager. Manager asked what type of rice packets you have. He said 50 packs of 30rs, 50 of 40rs, 100 of 50rs. He noted down on a bill paper then , he called his PA to store these 30rs packs in 1st room, 40rs packs in 3rd room, 50rs packs in 7th room.



Continue…

Considering the farmer example and co relating it with actual hadoop eco system as below



NOTE:- Write down the detail example in your notebook…

Q/A ON HDFS

1)What is HDFS?

2)What is default block size?

3)Why replication is important?

4)What is cheap commodity?

5) What is yarn?

6) What is another name of task tracker?

7)What are hdfs services?

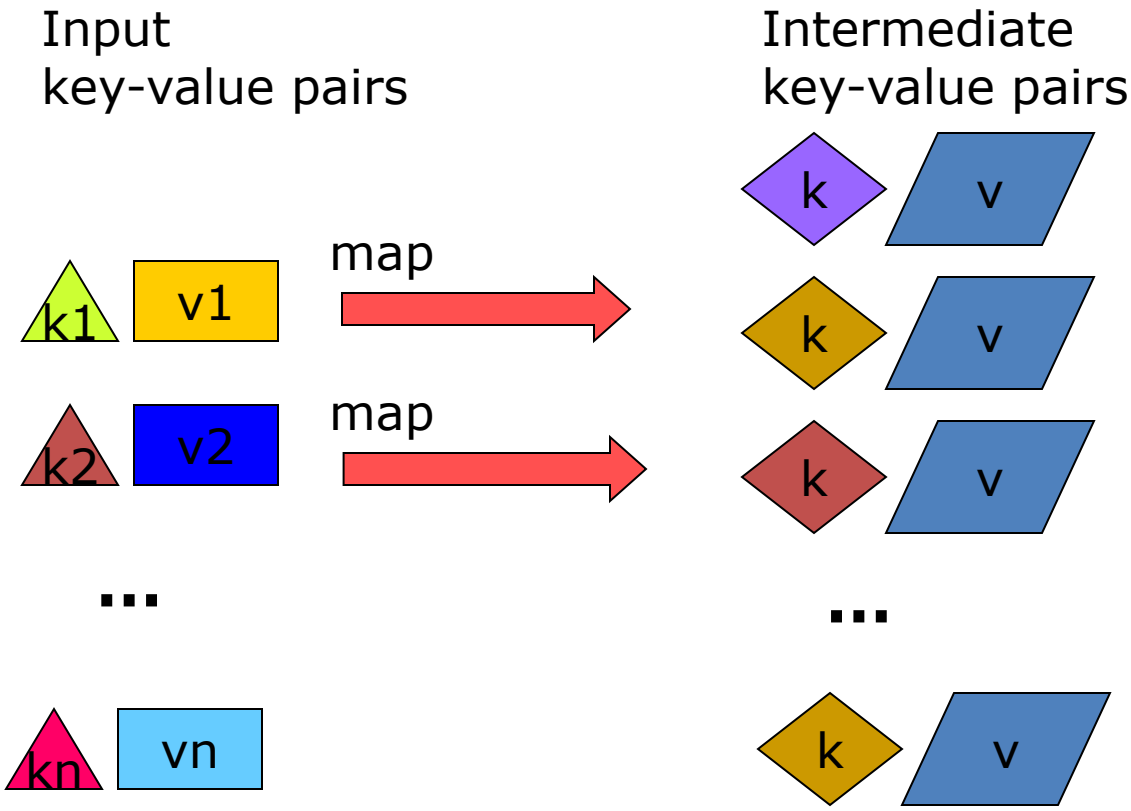
MapReduce: Insight

- Consider the problem of counting the number of occurrences of each word in a large collection of documents
- How would you do it in parallel ?
- Solution:
  - Divide documents among workers
  - Each worker parses document to find all words, outputs (word, count) pairs
  - Partition (word, count) pairs across workers based on word
  - For each word at a worker, locally add up counts

MapReduce Programming Model

- Inspired from map and reduce operations commonly used in functional programming languages like Lisp.
- Input: a set of key/value pairs
- User supplies two functions:
  - map(k,v) → list(k1,v1)
  - reduce(k1, list(v1)) → v2
- (k1,v1) is an intermediate key/value pair
- Output is the set of (k1,v2) pairs

MapReduce: The Map Step

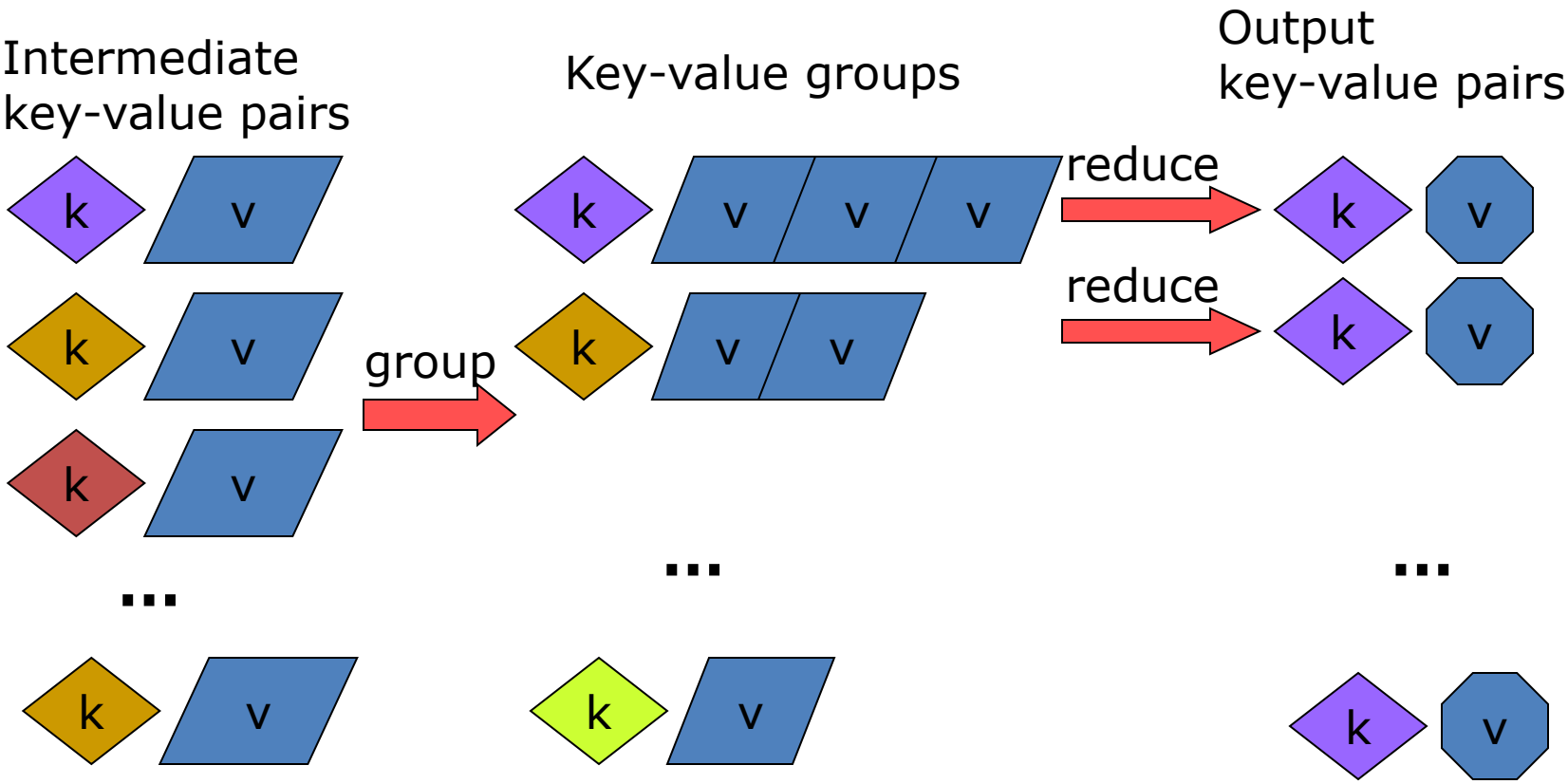


E.g. (doc—id, doc-content)

E.g. (word, wordcount-in-a-doc)

Adapted from Jeff Ullman's course slides

MapReduce: The Reduce Step

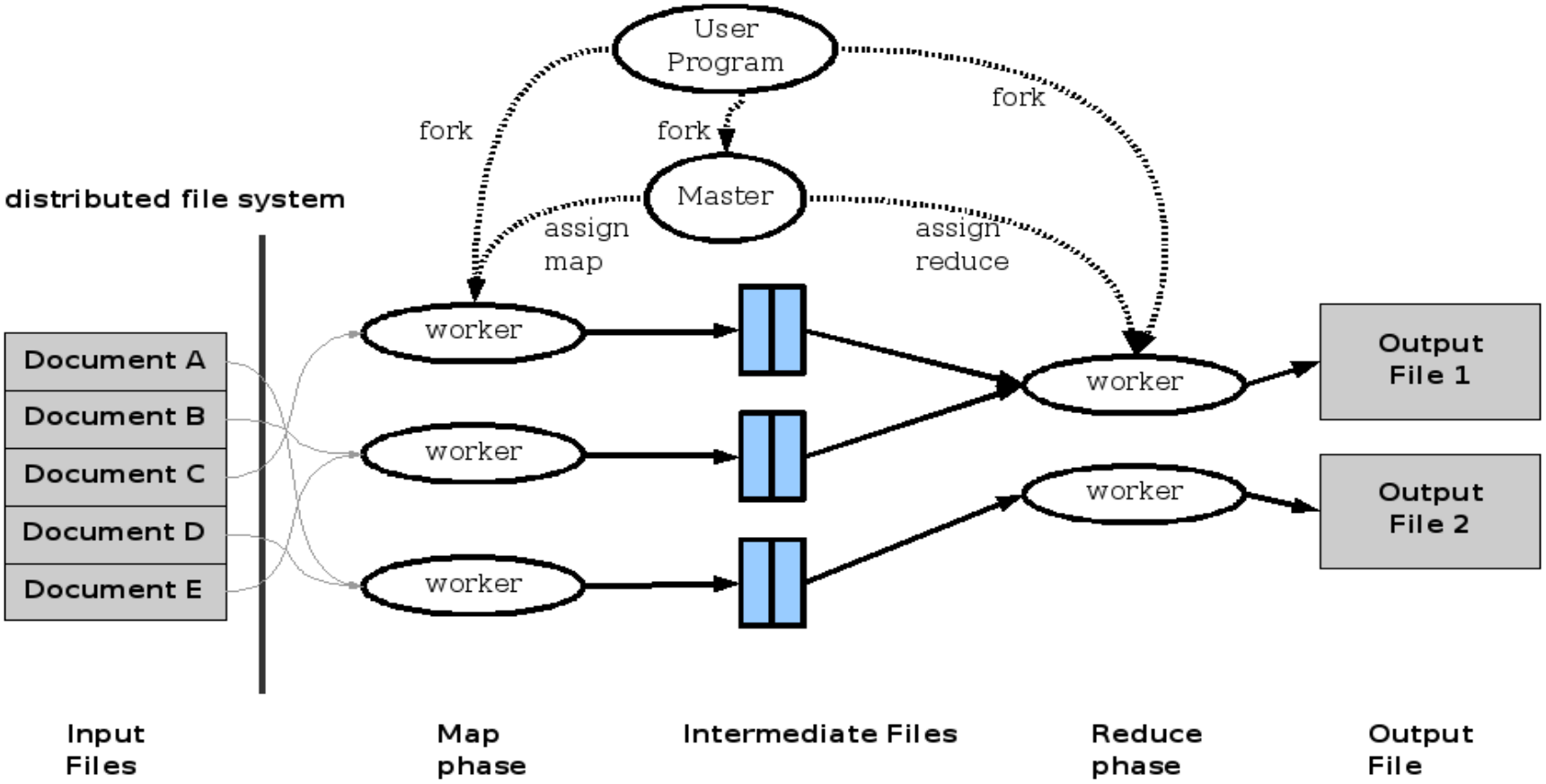


Adapted from Jeff Ullman's course slides

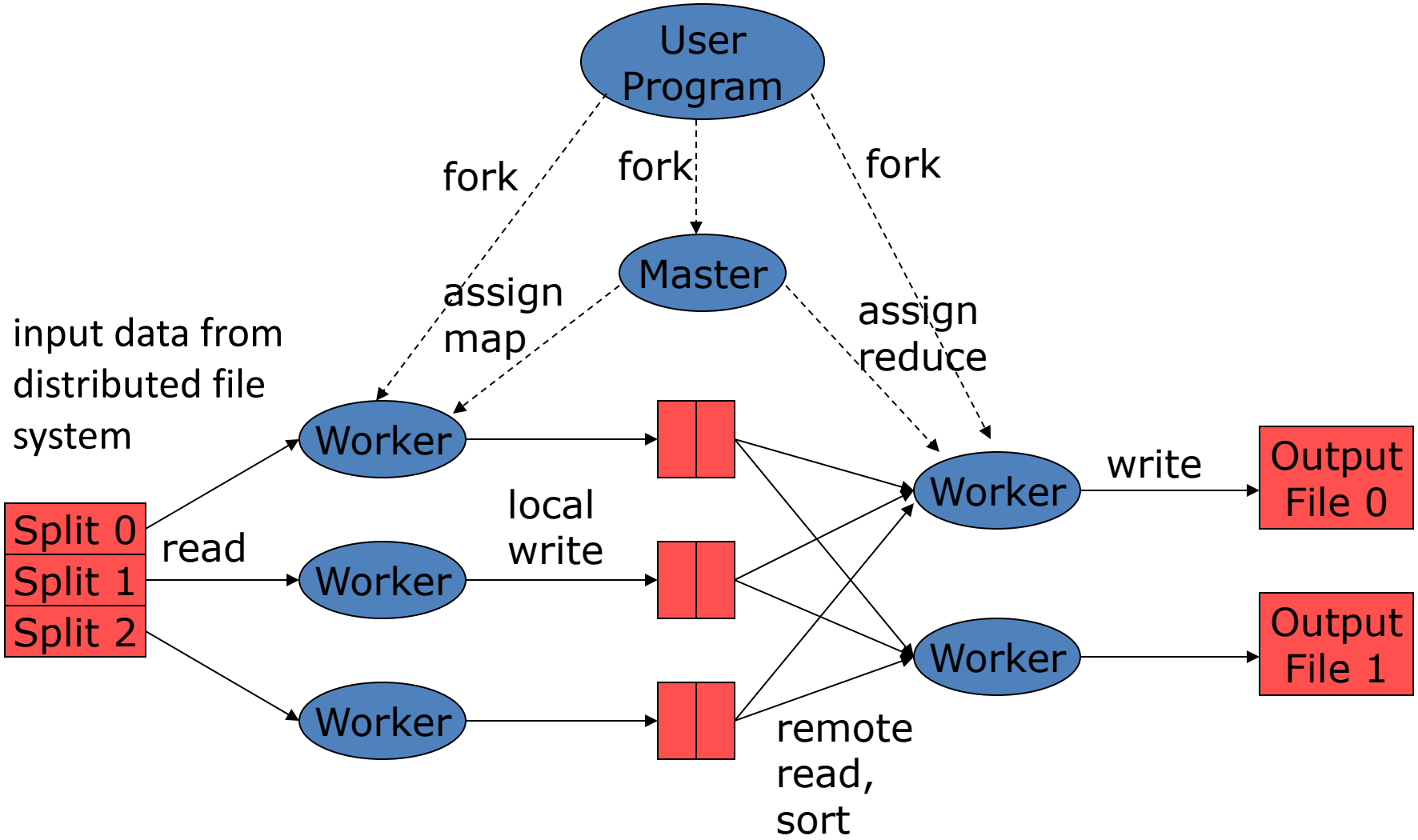
Pseudo-code

```
map(String input_key, String input_value):
// input_key: document name
// input_value: document contents
    for each word w in input_value:
        EmitIntermediate(w, "1");
// Group by step done by system on key of intermediate Emit above, and // reduce called on list of values in each group.
reduce(String output_key, Iterator intermediate_values):
// output_key: a word
// output_values: a list of counts
    int result = 0;
    for each v in intermediate_values:
        result += ParseInt(v);
    Emit(AsString(result));
```

MapReduce: Execution overview

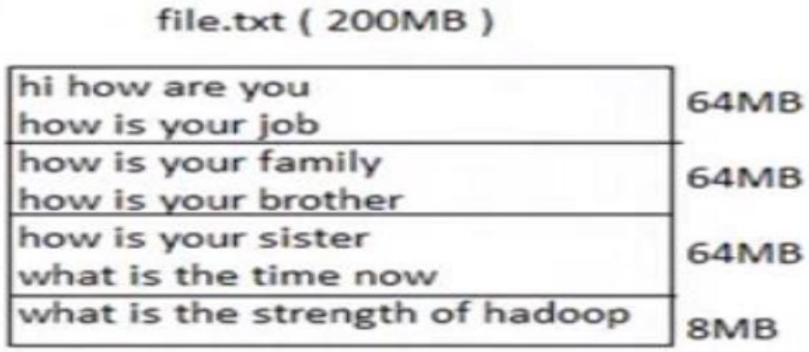


Distributed Execution Overview

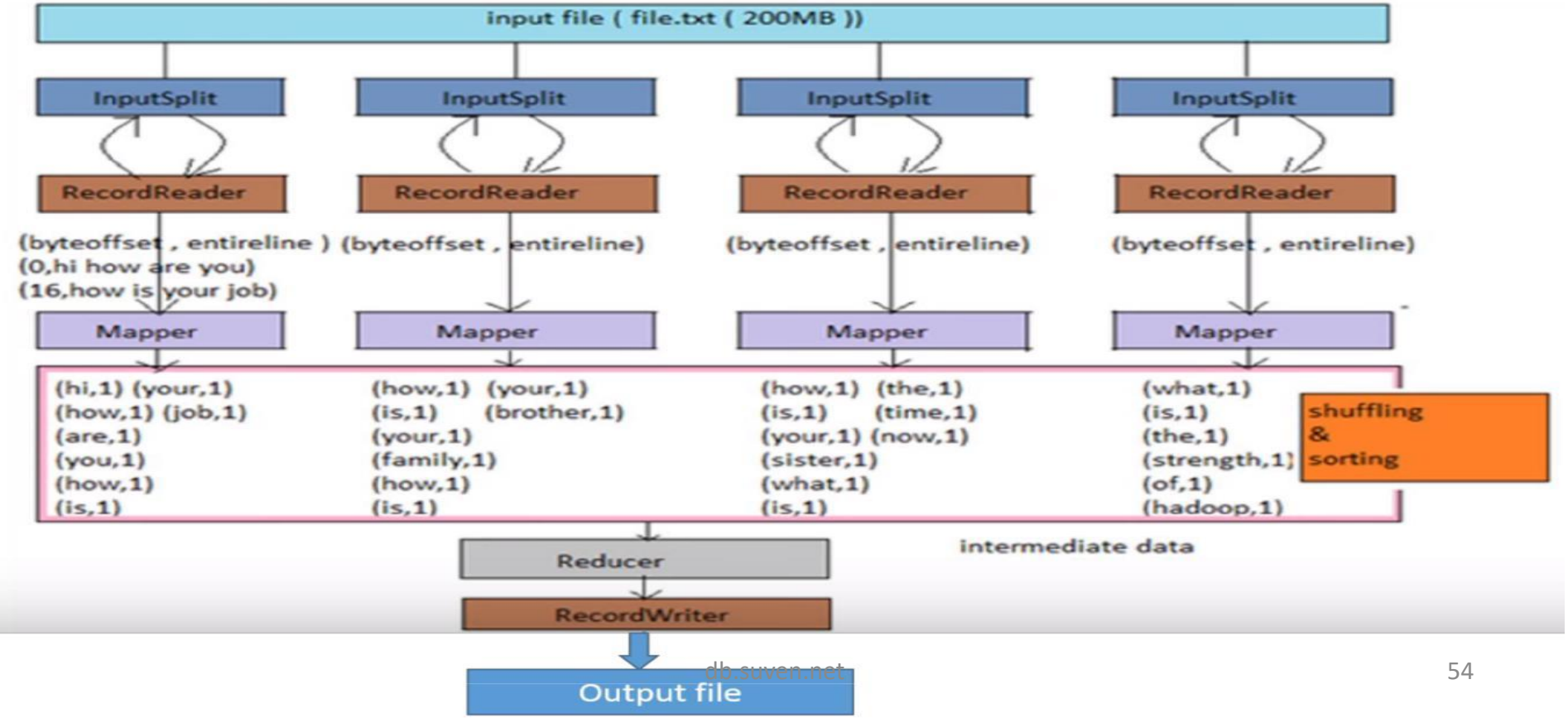


From Jeff Ullman's course slides

WORD COUNT FLOW CHART



MapReduce Flow Chart :



Q/A ON MAPREDUCE

1)What is recordreader?

2)What is recordwriter?

3)What is partitioner?

4)What is combiner?

5) What is inputsplit?

6) What is maximum output can be generated in mapreduce?

7)What are box classes in hadoop ?