# Big Data

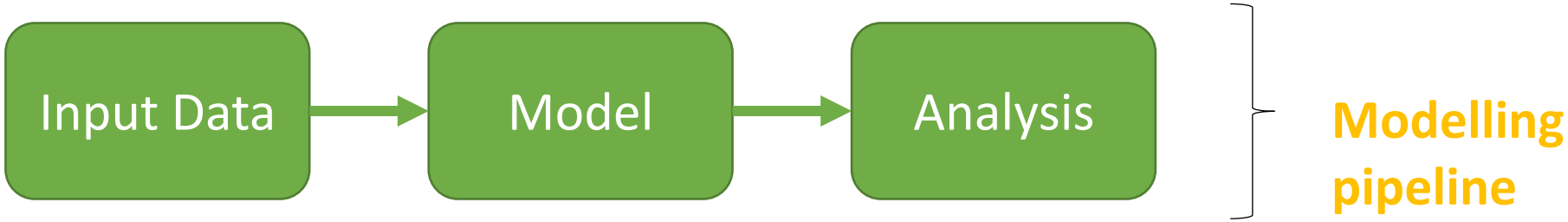
## Introduction

**Rachana B S, Usha Devi B G, Resma K S, Prafullata K A, Subramaniam K V**
Department of Computer Science and Engineering
**{rachanabs,ushadevibg, resma, prafullatak, subramaniamkv}@pes.edu**
+91 80 6666 3333 Extn 877

## What is Big Data?

There is no one standard single 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…




**Model** – is a human construct that better helps us understand real-world systems/phenomena.

With Big Data, this means….

# BIG DATA

## Big Data themes


- How to manage very large amounts of data (*data management)*
- and extract value and knowledge from them (*analytics)*
- Google: store index to WWW and search
- Amazon: store user purchases and make recommendations

***Large-Scale Data Management***

***Big Data Analytics***

***Data Science and Analytics***

# Big Data: Motivating Example

# BIG DATA

## Big Data themes


- How to manage very large amounts of data (*data management)*
- and extract value and knowledge from them (*analytics)*

- Google: store index to WWW and search
- Amazon: store user purchases and make recommendations

***Large-Scale Data Management***

***Big Data Analytics***

***Data Science and Analytics***

## High level approach: motivating example

- Machine Translation
- Translating a sentence from English → Hindi
- What would be the traditional approach?
- How will it differ from the Big Data approach?

| English | Hindi |
| --- | --- |
| Can you teach me? | क्या तुम मुझे सिखा सकते हो? |
| You make mistakes if you do things in a hurry. | जल्दबाज़ी में काम करोगे तो ग़लतियाँ तो होंगी ही। |

https://towardsdatascience.com/intuitive-explanation-of-neural-machine-translation-129789e3c59f

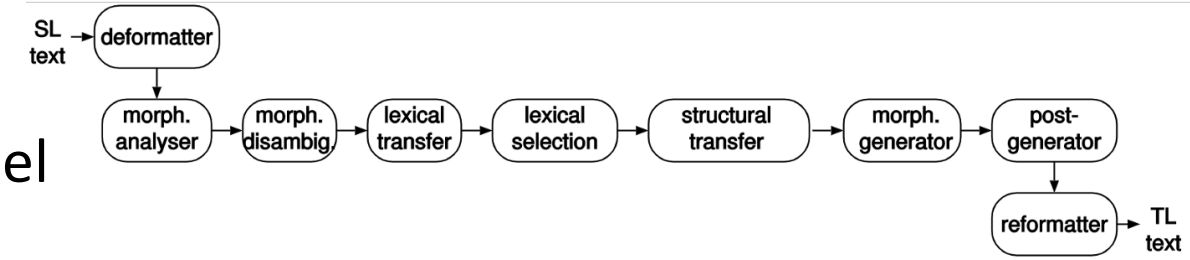
# BIG DATA

## Traditional Approach

- Understand the system – linguistic approach – rule based

- Requires a linguistic expert to build a model

- Model should include
  - Language structure → morphology, grammar
  - Meaning of the words
  - Mapping words from one language to another



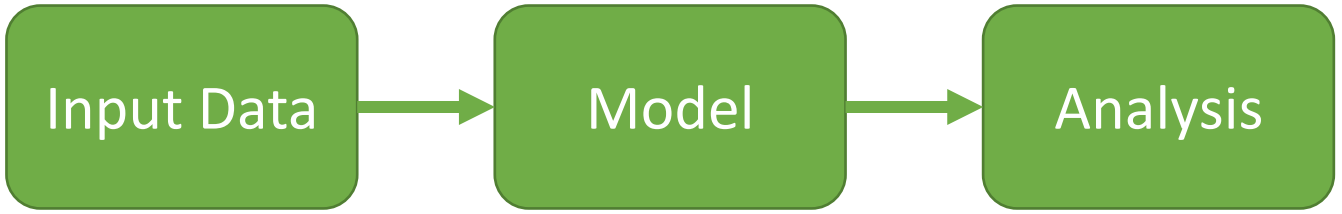

https://towardsdatascience.com/machine-translation-a-short-overview-91343ff39c9f

## Big Data Approach

- No attempt to understand language
- Gather data about different sentences and translations
  - Requires a parallel corpus
  - Millions of sentences and their translations
- Build a statistical model
- For example:
  - Every time the word cat appears in the English sentence
  - The hindi equivalent has *billi*
  - So infer that cat can be translated as *billi*


https://techmediahub.com/machine-translation-complete-useful-guide/

## Big Data and Analytics




Traditional Approach

The model is human generated

Big Data Approach

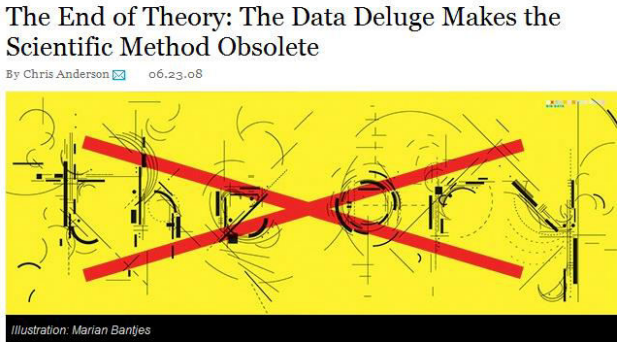
The model is machine generated

# BIG DATA

## What about domain knowledge?

- Correlation is enough?
- Gene sequencing of DNA fragments found in ocean by J. Craig Venter
  - 1000s of new species
  - No idea of what species looks like or any other info

- All models are wrong, and increasingly you can succeed without them
  - Peter Norvig, Google's research director
  - “The unreasonable effectiveness of data”


The End of Theory: The Data Deluge Makes the Scientific Method Obsolete

By Chris Anderson 06.23.08

Illustration: Marian Bantjes

http://www.wired.com/science/discoveries/magazine/16-07/pb_theory

## Conclusions from Peter Norvig's talk

- Algorithms are not important, data is
  - Domain knowledge (e.g., physics/grammar) is not important

- Demonstrates how images can be merged together using just data

- And translation of text giving examples of issues in segmentation


- Peter Norvig, Head Google Research, *The Unreasonable Effectiveness of Data*
  *https://www.youtube.com/watch?v=yvDCzhbjYWs*

## What about domain knowledge?

- Can we rely only on data alone?

- Does this mean that domain knowledge is obsolete?

# Big Data: Pitfalls in Analysis

## Issues in machine translation

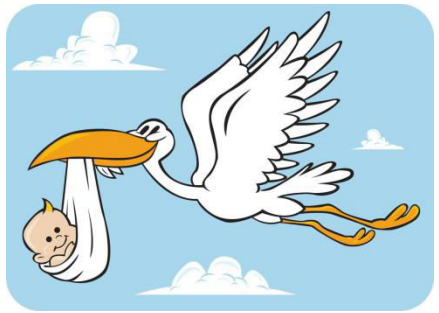
- What about *let the cat out of the bag*?
  - Naïve translation - *billi ko bag ke bahar chhod diya*
  - English meaning: reveal a secret


- To be able to solve this, we need information about the language → domain knowledge and some experimentation

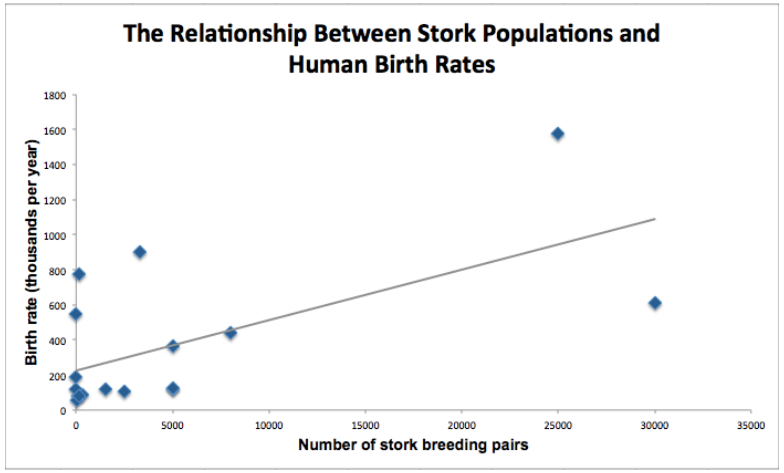
- Peter Norvig, Head Google Research, *The Unreasonable Effectiveness of Data* *https://www.youtube.com/watch?v=yvDCzhbjYWs*

# BIG DATA

## Pitfall : Spurious correlation



- C->A, C->B
  - *Does A->B?*
- Example:
  - Do storks deliver babies?
- Chart shows positive correlation between
  - Stork population and human birth rates in European countries
  - What it does not show is a hidden variable
    - Available nesting area?

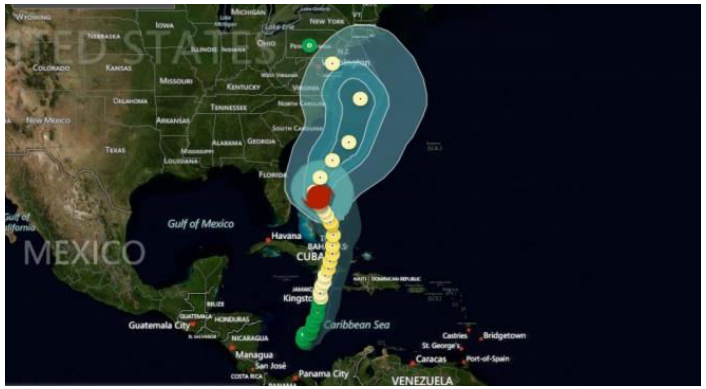



- http://en.wikipedia.org/wiki/Spurious_relationship
- http://www.cut-the-knot.org/do_you_know/misuse.shtml

# BIG DATA

## Pitfall : Gaps in the data

- Selection bias
- Convenience
- Example
  - Rutgers University study
  - Examine decision-making process in emergency
  - Study tweets during Hurricane Sandy
  - Most tweets from Manhattan!
  - If studying impact of Sandy: *Manhattan most impacted!*




- *More Data, More Problems: Is Big Data Always Right?* ARI ZOLDAN
  http://www.wired.com/insights/2013/05/more-data-more-problems-is-big-data-always-right/

# BIG DATA

## Pitfall : Gaps in the data

- Another example: medicine

- Missing data is always a challenge
  - but we also know that “negative results” are more likely to go missing.
  - This means we have a *biased sample*, overestimating the benefits of treatments.


- *The Information Architecture of Medicine is Broken* Ben Goldacre
  http://strataconf.com/strata2012/public/schedule/detail/22941
- https://www.youtube.com/watch?v=AK_EUKJyusg

# Big Data: How to address the issues?

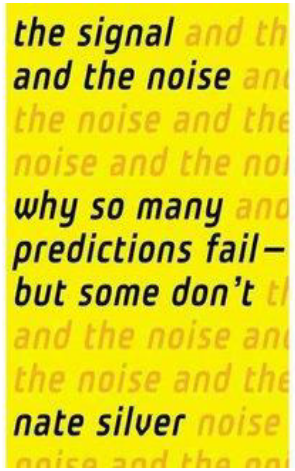
## Summary of the methods

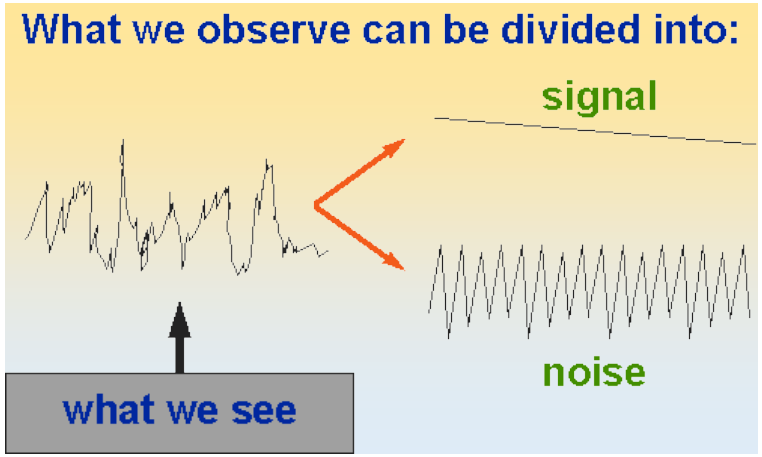

- Use domain knowledge to check model for validity
- Estimate errors

# BIG DATA

## Let's look to some experts

- Nate Silver book
  - The Signal and the noise
- On Time Magazine 2009 – 100 most influential people
- Correctly predict US 2008/2012 elections



# BIG DATA

## Example: Weather Forecasting

Why is weather forecasting very successful?

Chaotic (dynamic, non-linear system)

- Lorenz: 29.5168 instead of 29.517

Adjustment by humans

- Compute probabilities: how often predict rain, didn't rain?
- On ground reality

The effect of marketing/customer satisfaction in commercial weather forecasting.

- More sensitive about errors in predicting no rain than rain

## Big Data Error Estimation

- Purely empirical: cannot be analysed by theory
- Divide data into *training set* and *testing set*
- Develop algorithm using training set; estimate error from testing set
  - Can be used to compare analytics algorithms
- Examples
  - Nate Silver: weather prediction: human adjustment
  - Amazon recommendations
    - Derive model using historical data; make recommendations
    - Get statistics on how many people look at or buy recommendations

# Big Data: Summary and architecture

# BIG DATA

## Big Data themes


***Large-Scale Data Management***

***Big Data Analytics***

***Data Science and Analytics***

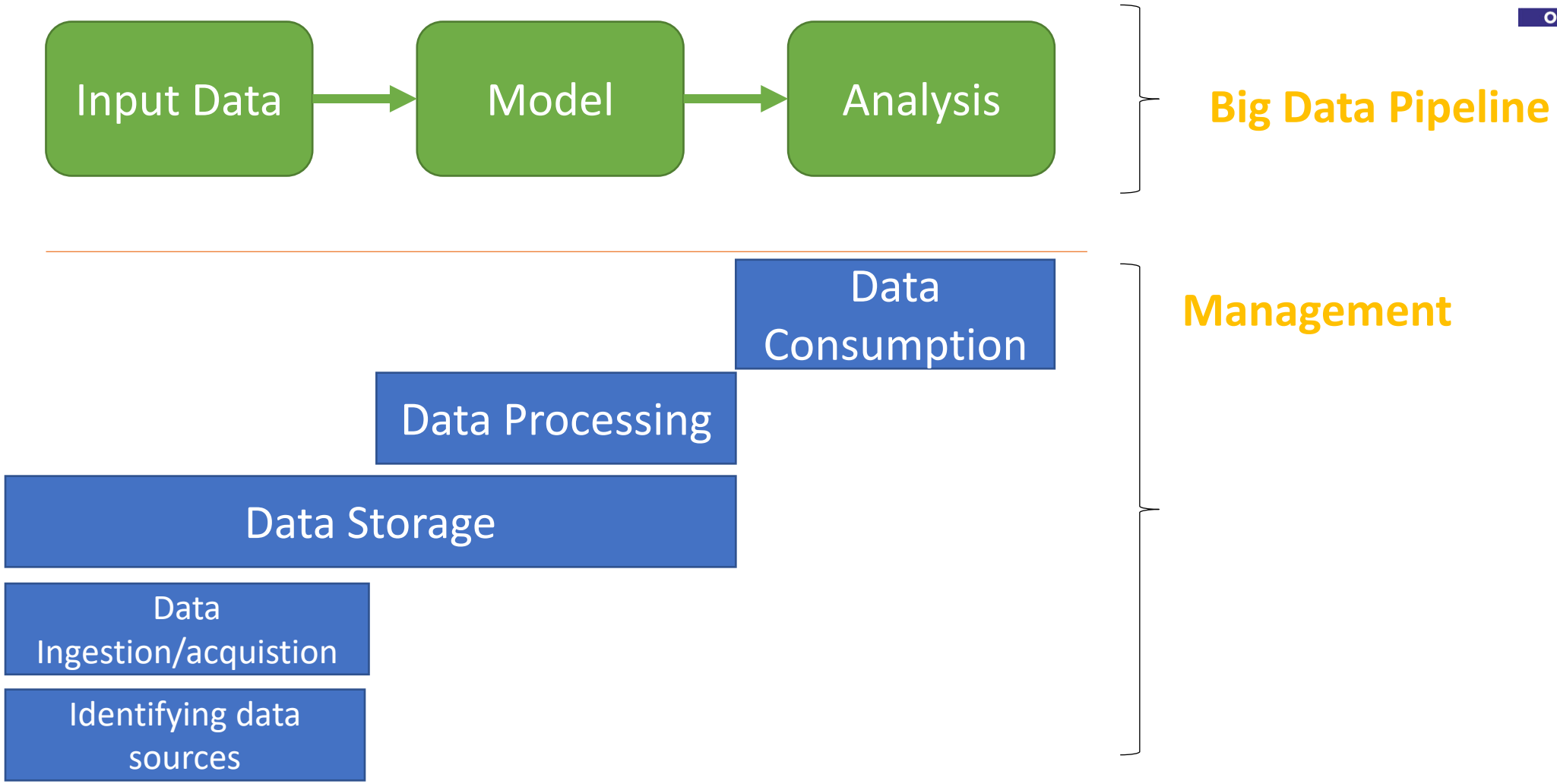
- How to manage very large amounts of data (*data management)*

- and extract value and knowledge from them (*analytics)*

- Google: store index to WWW and search

- Amazon: store user purchases and make recommendations

## Big Data and Analytics




# THANK YOU

**K V Subramaniam**

Department of Computer Science and Engineering

**subramaniamkv@pes.edu**

+91 80 6666 3333 Extn 877

Content Creators

Prof Prafullata K A

Prof Usha Devi

Prof Rachana

Prof Resma

# BIG DATA

## Characteristics and Architecture

**Prafullata Kiran Auradkar**

Department of Computer Science and Engineering

**prafullatak@pes.edu**

**Acknowledgements:**

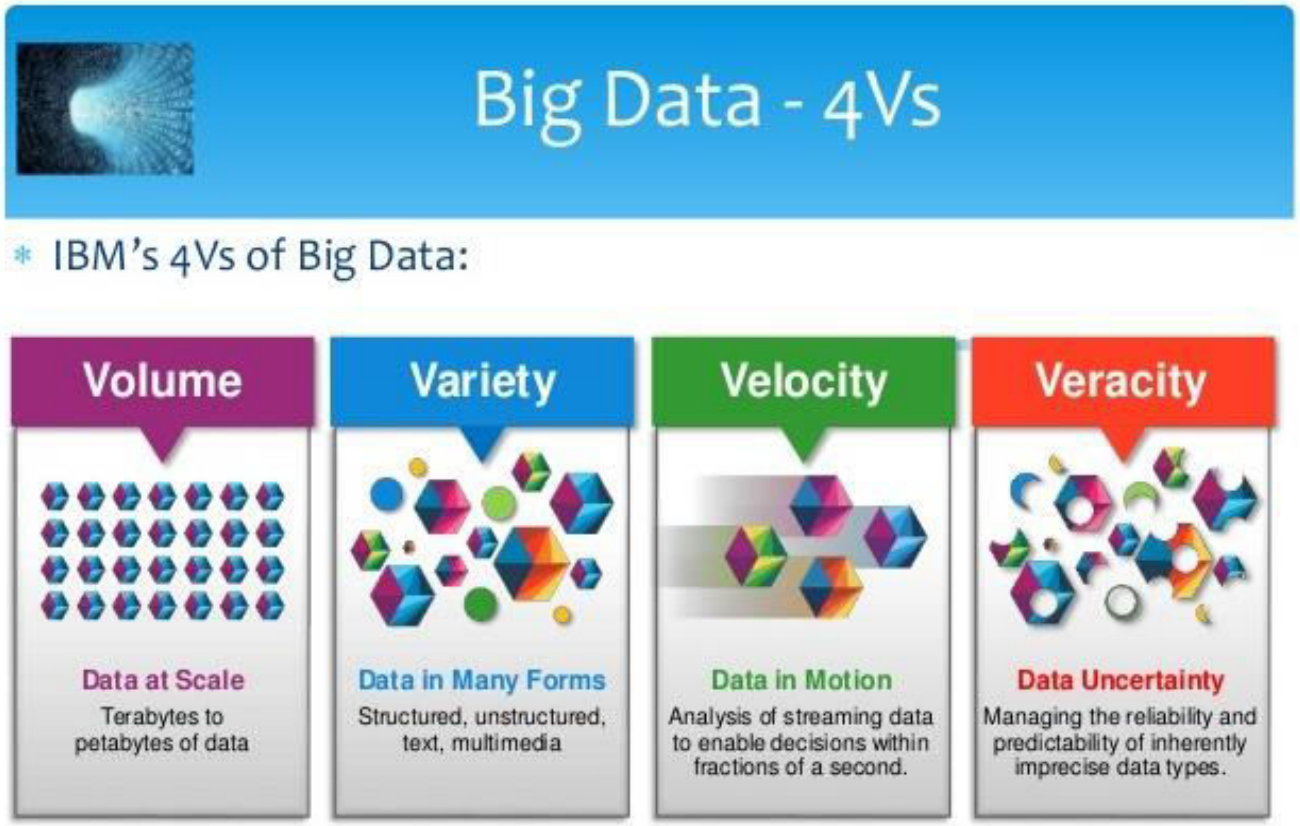
Significant information in the slide deck presented through the Unit 1 of the course have been created by **Dr. K V Subramaniam's** and would like to acknowledge and thank him for the same. There have been some information which I might have leveraged from **Dr. H L Phalachandra's** slide contents too. I may have supplemented the same with contents from books and other sources from Internet and would like to sincerely thank, acknowledge and reiterate that the credit/rights for the same remain with the original authors/publishers only. These are intended for classroom presentation only.

## Course Introduction Contd...

1. Big Data – Characteristics
2. Data Architecture Design
3. Data Format/Types
4. Big Data Platforms
5. Case Study - Google

# BIG DATA

## Characteristics – Quick Summary



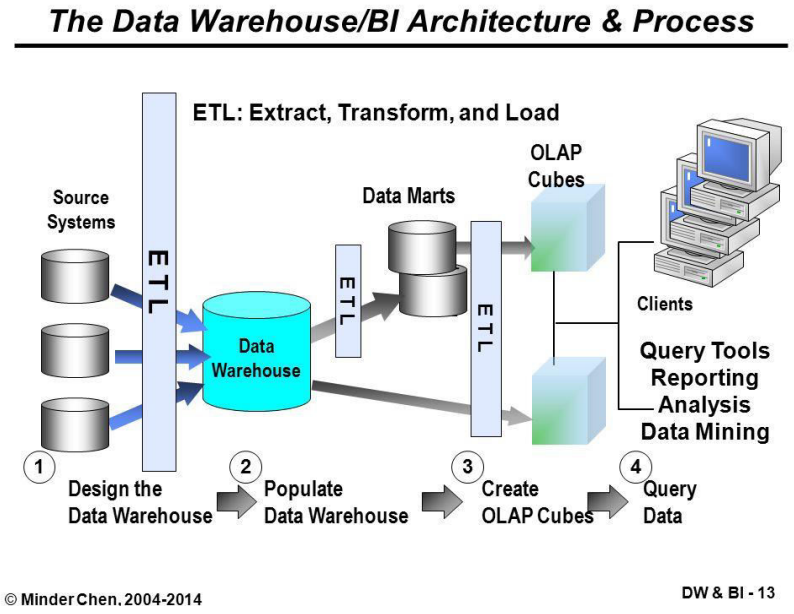

http://www.sliceofbi.com/2015/09/basics-of-big-data.html

# Big Data: Characteristics - Volume

# BIG DATA

## Old Style Data


http://www.techbridge.solutions/home/solutions/solutions-ibm-applications/oltp-host-concentrator-solution/




https://data-flair.training/blogs/business-intelligence-and-data-warehousing/

- Fixed schema and format
- Clean data
- Consistent
- Predictable data rates

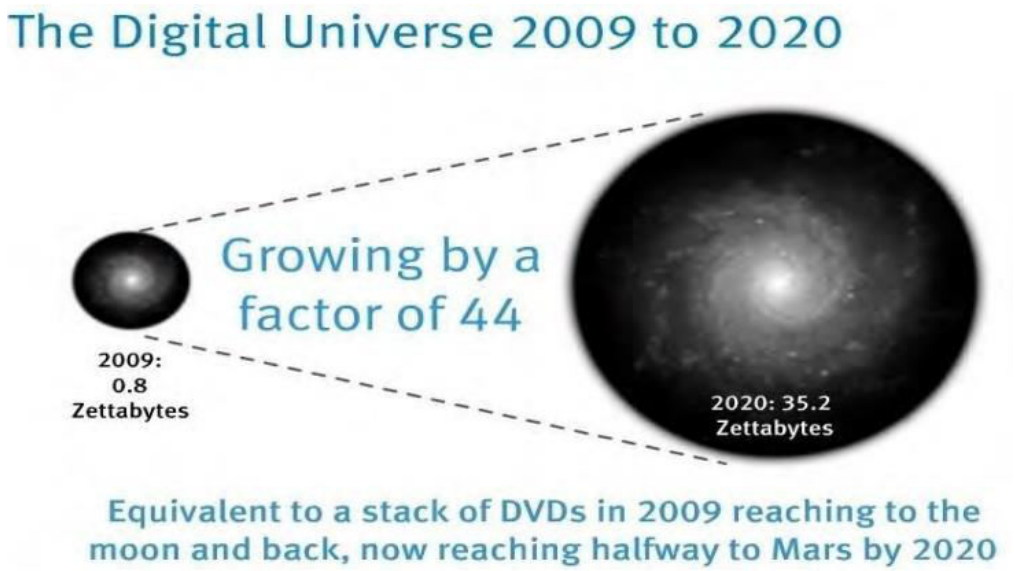
- Elaborate ETL procedure ( Not real time; once a day at best)
- Output mostly reports (Queries run in batches; take hours)

# BIG DATA

## Characteristics - Volume

- Old Style Data vs Big Data

  1-Scale (Volume)
  - 44x increase from 2009 2020
  - From 0.8 zetta bytes to 35zb
  - Data volume is increasing exponentially

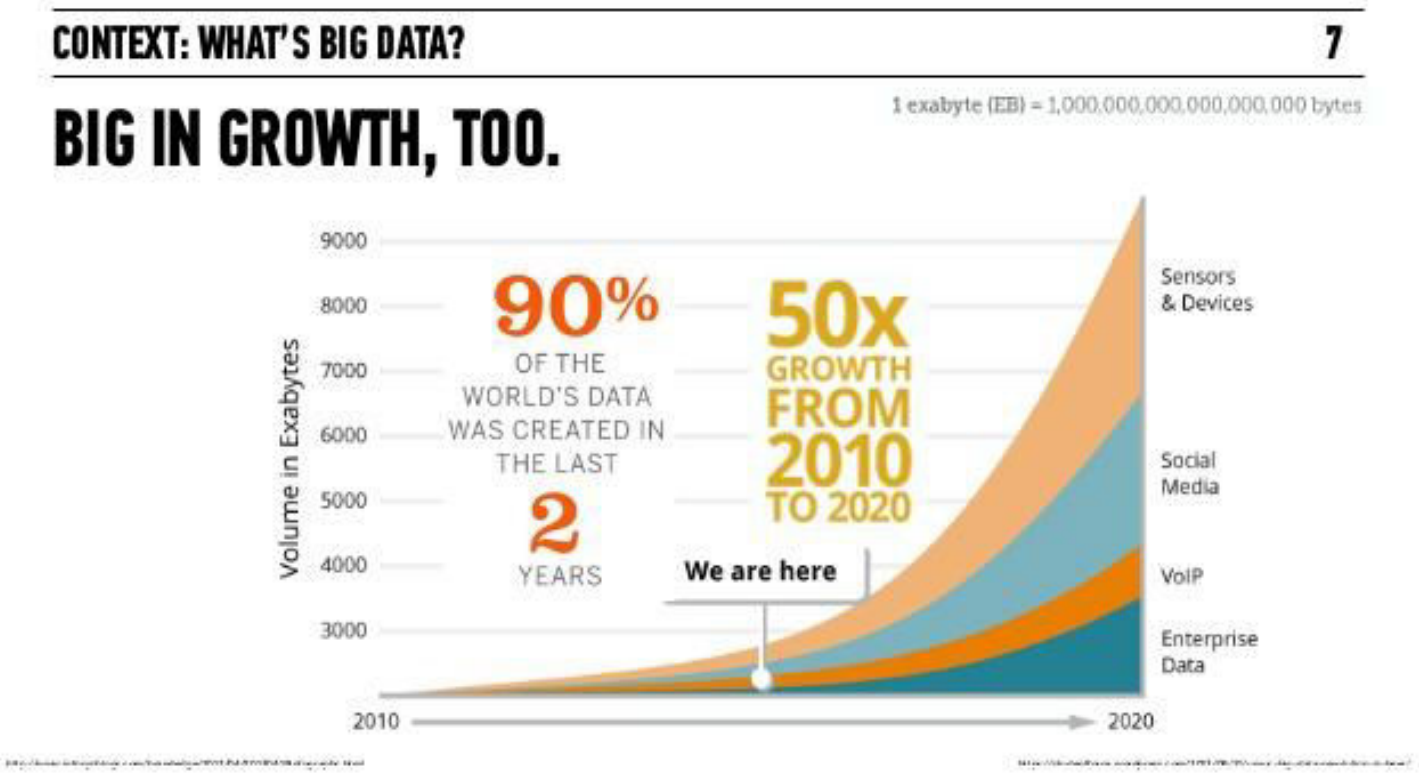



http://www.egsa.org/Portals/7/Documents/CommitteeDocs/Misc/F15_Data_Center_Market.pdf

# BIG DATA

## Characteristics - Volume

- Why is the volume so high? Who's Generating it?




| Old Style Data | Big Data |
|---|---|
| Enterprises only | Everybody |

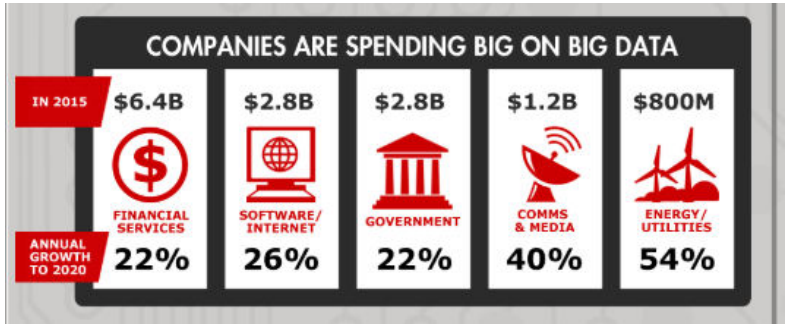
https://www.promptcloud.com/blog/want-to-ensure-business-growth-via-big-data-augment-enterprise-data-with-web-data/

# BIG DATA

## Characteristics - Volume

| Old Style Data | Big Data |
|---|---|
| Manual entry | Automated |

- How this VOLUME gets collected?
  - Financial Services
    - Stock trading, banking, financial services are all computerized (In India – no paper shares any more)
    - Every transaction is recorded
  - Energy / Utilities
    - Can put sensors in every home
    - Smart grid
  - Comms & media
    - Internet services




https://www.forbes.com/sites/louiscolumbus/2014/06/24/roundup-of-analytics-big-data-business-intelligence-forecasts-and-market-estimates-2014/#26c06d11388e

# Big Data: Characteristics - Variety

# BIG DATA
## Characteristics - Variety

- Why is analyzing Big Data *complex*?
- 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

| Old Style Data | Big Data |
|---|---|
| Fixed format / schema | Integrate Twitter, Maps, Facebook… |

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

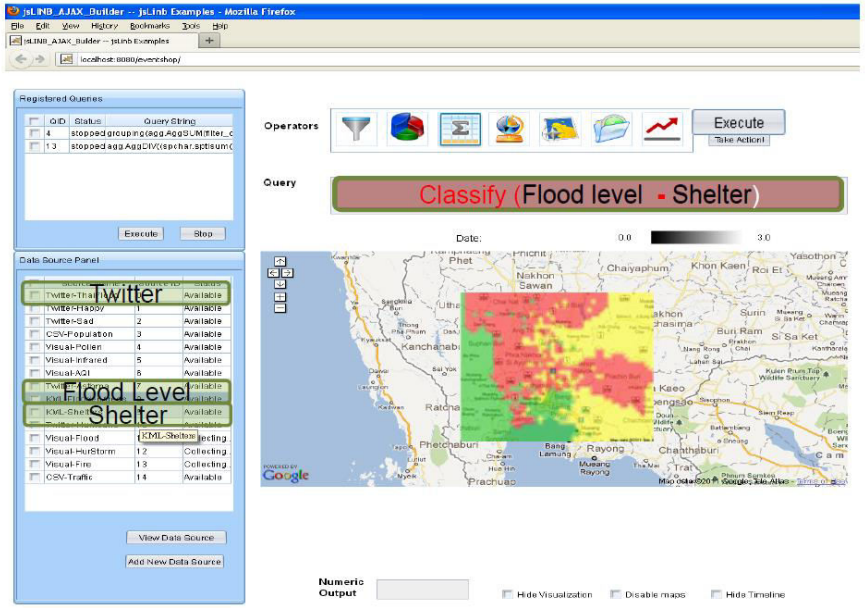

https://twitter.com/maritime_uk/status/1072159378931879937

# BIG DATA

## Characteristics - Variety

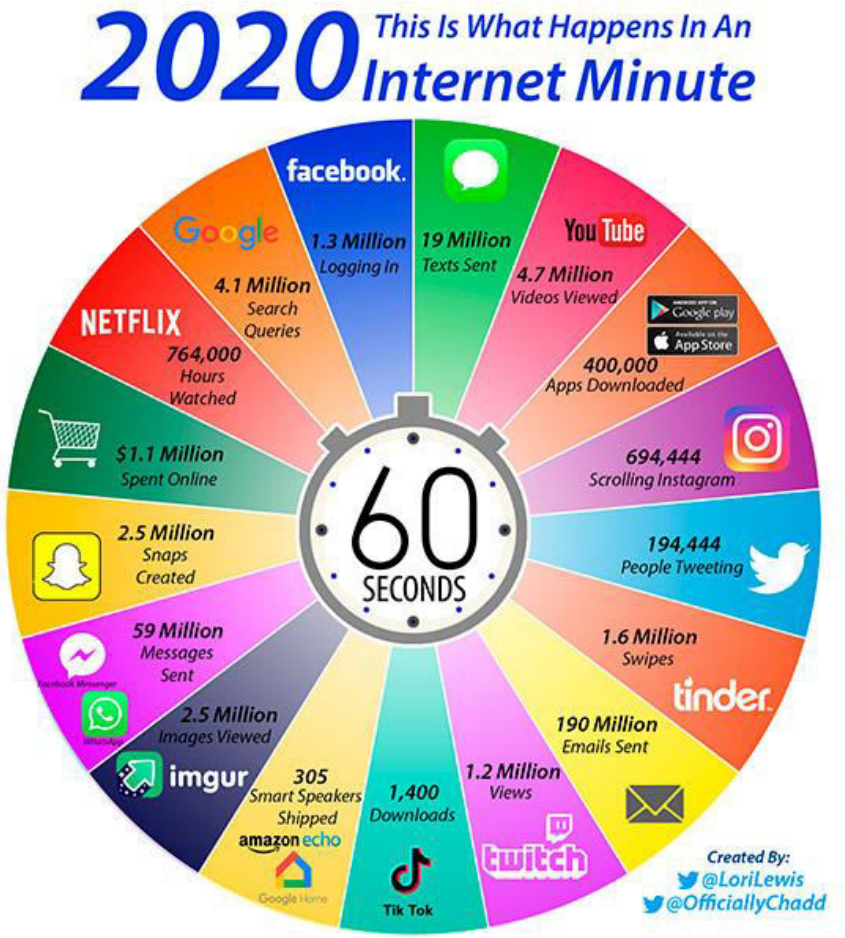
PES UNIVERSITY

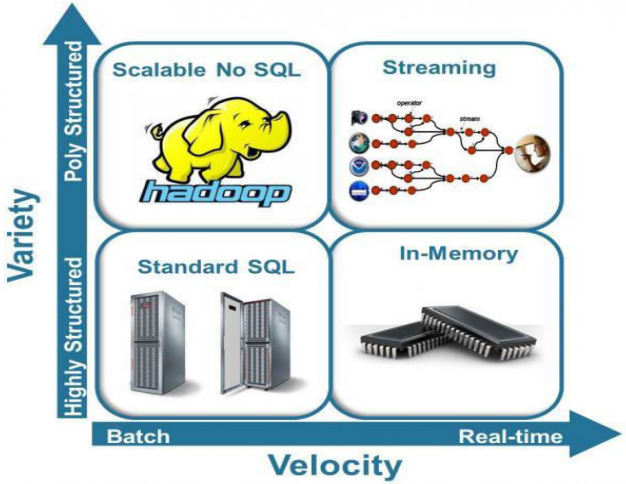
## EventShop: Thai Floods




# Big Data: Characteristics - Velocity

# BIG DATA
## Characteristics - Velocity




Source: Forrester Webinar: Big Data: Gold Rush or Illusion?, Sept 19, 2013

https://go.forrester.com/blogs/14-05-27-boost_your_digital_intelligence_with_big_data/

https://www.allaccess.com/merge/archive/31294/infographic-what-happens-in-an-internet-minute

# BIG DATA
## Characteristics - Velocity

### Nokia launches real-time mobile network analytics platform

*Staff writer* | July 28, 2016 | Telco Analytics Asia

https://www.nokia.com/about-us/news/releases/2016/06/30/nokia-real-time-mobile-network-analytics-provides-instant-correlation-of-network-wide-data-and-its-impact-on-customer-experience/

The analytics link the performance of applications and devices to network issues in real-time, effectively making every mobile device part of a network test bed.

This enables operators to pinpoint potential causes of service degradation much more rapidly than they can today since they no longer need to consult a myriad of tools and correlate the data from them manually. It also offers engineering teams a proactive way to understand over the top (OTT) application impacts on the network and optimize opportunities.

| Old Style Data | Big Data |
|---|---|
| Input data at night | Real-time input |
| Output daily report | Real-time response |

# Big Data: Characteristics - Veracity

# BIG DATA

## Characteristics - Veracity

Veracity

refers to the messiness or trustworthiness of the data.

| Old Style Data | Big Data |
|---|---|
| Clean data | Inconsistent data |

Accuracy cannot be controlled due to

- Hashtags
- Typos
- Abbreviations

Need to work with such data

Traditional: manually entered, fixed fields, less chance of error

# BIG DATA

## Characteristics - Veracity

- How people get multiple PAN cards

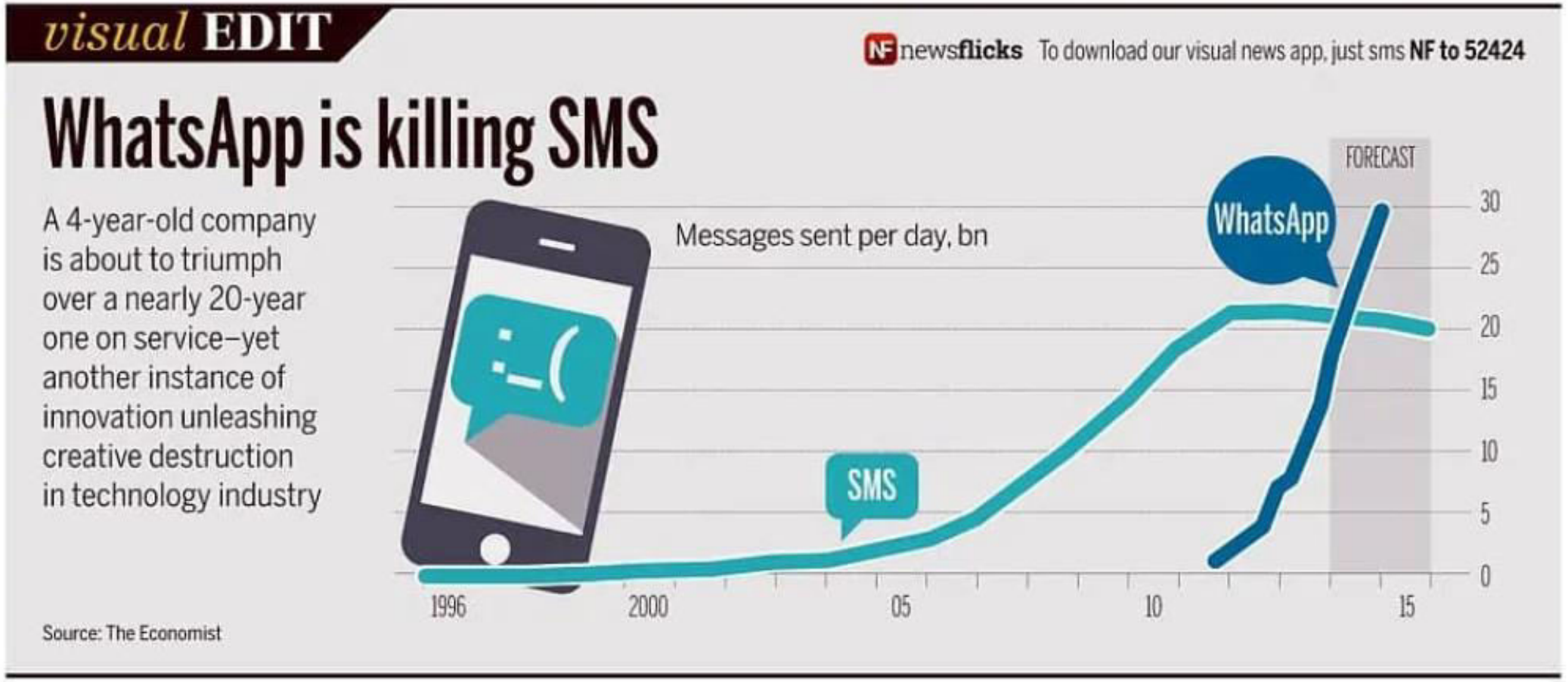
- Slight changes in name and address
  PES University, 100 ft Road…
  PES university, BSK III stage...
  Pes University, Dwaraka Nagar...

- Names and addresses don't match

- But postman / courier will deliver to correct place!!!

# BIG DATA

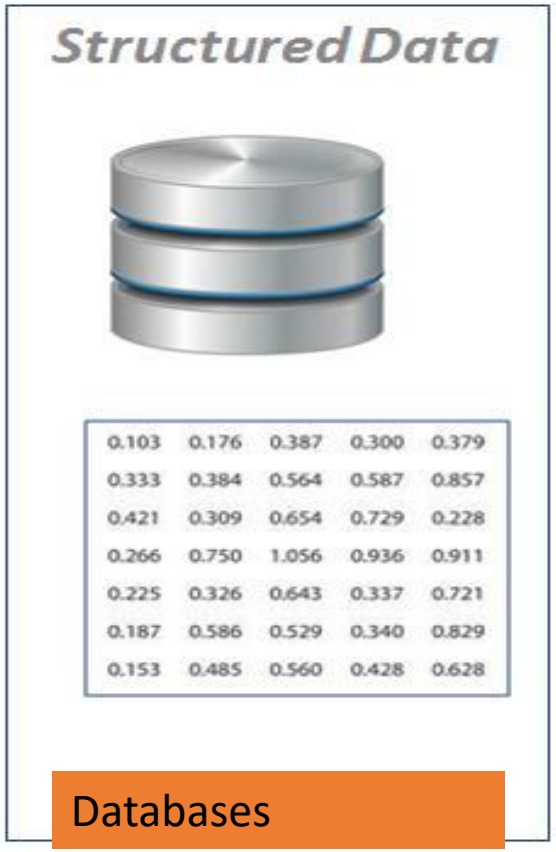
## Characteristics – 4 V s



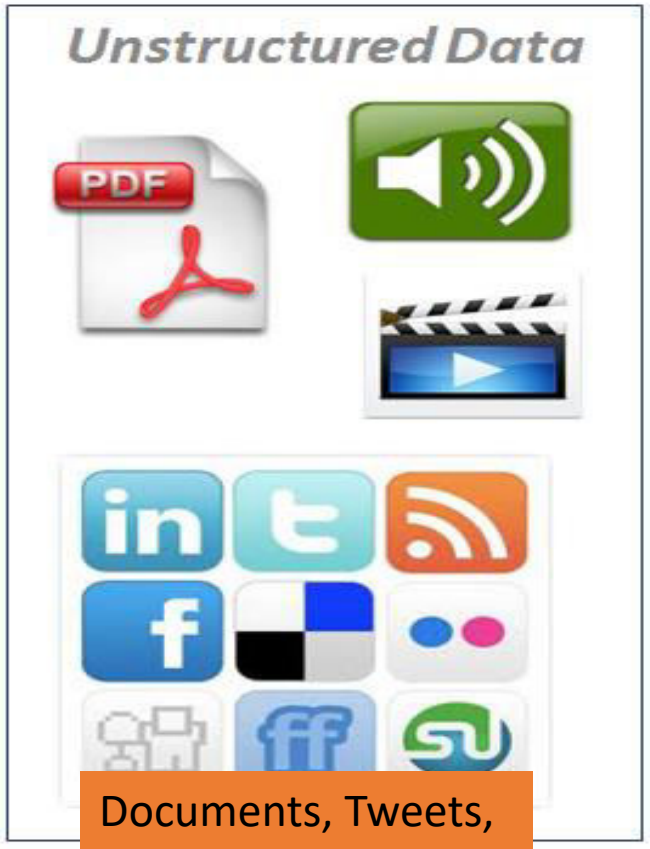

*WhatsApp was eventually successful in replacing SMS*

# Big Data: Data Architecture Design

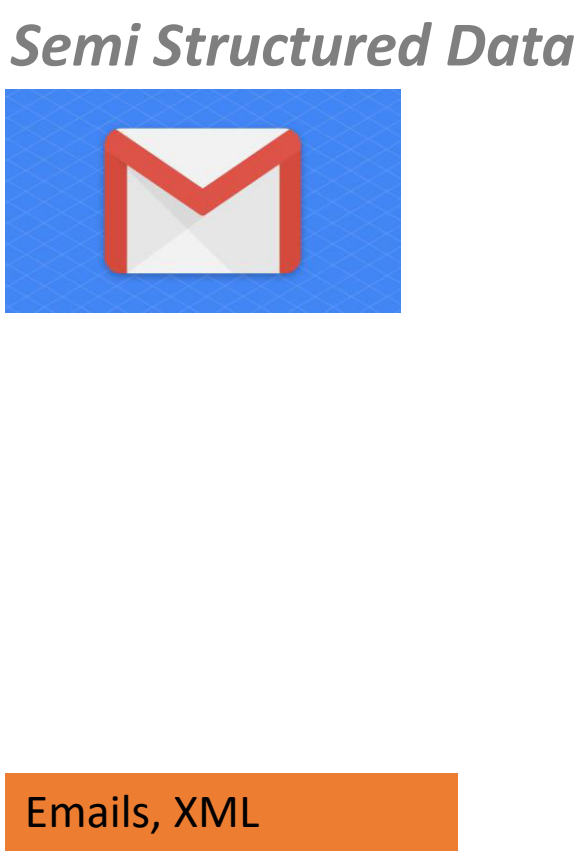
# BIG DATA
## Data Formats / Types




Databases




Documents, Tweets, Videos

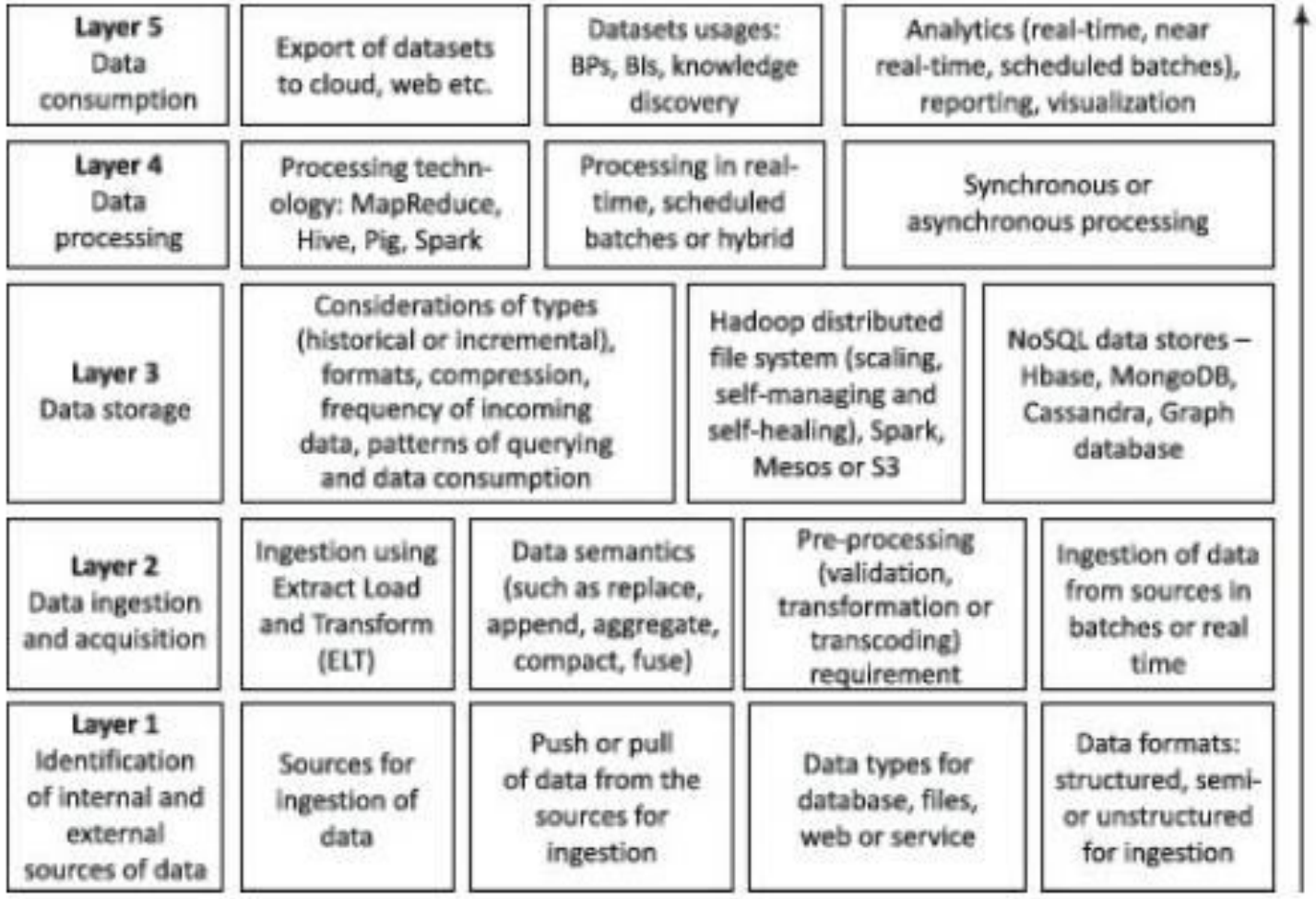



Emails, XML

http://biisfun.blogspot.com/2016/03/big-unstructured-data-vs-structured.html

# BIG DATA

## Data Architecture Design

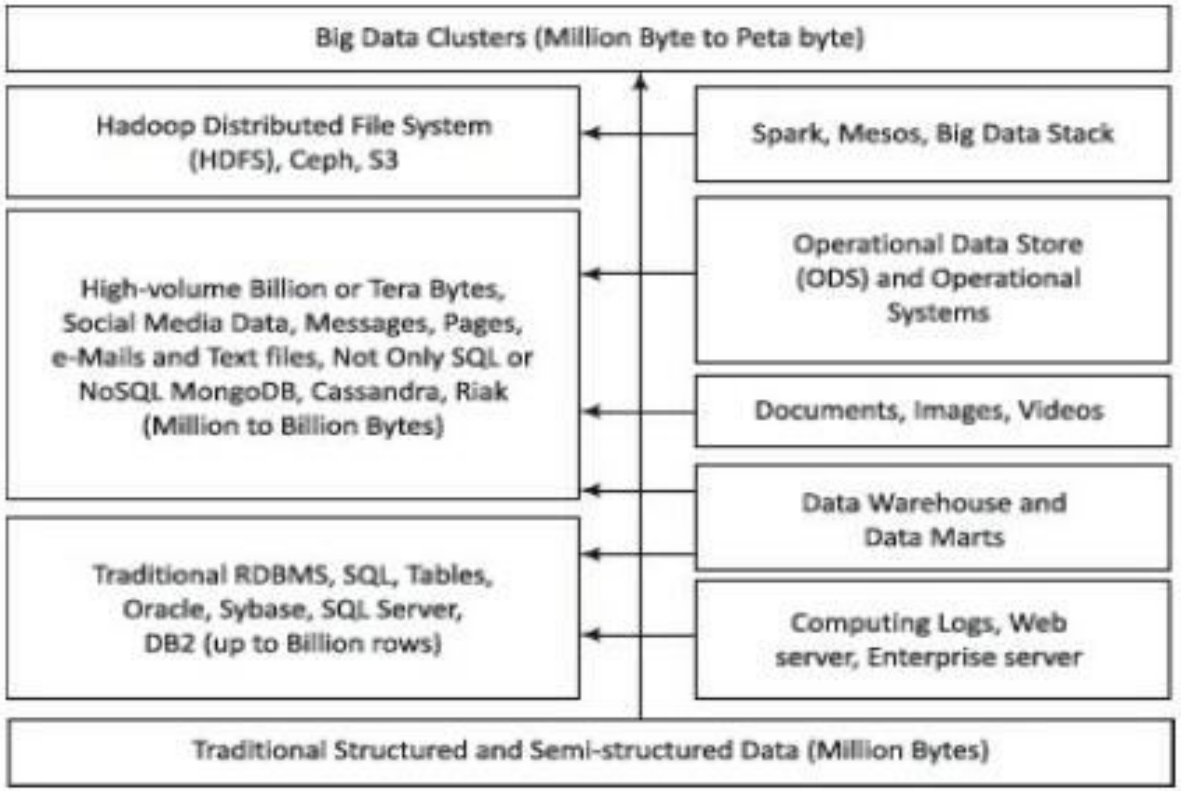



T1: Fig 1.2 – Logical layers in data processing architecture and their functions.

# BIG DATA

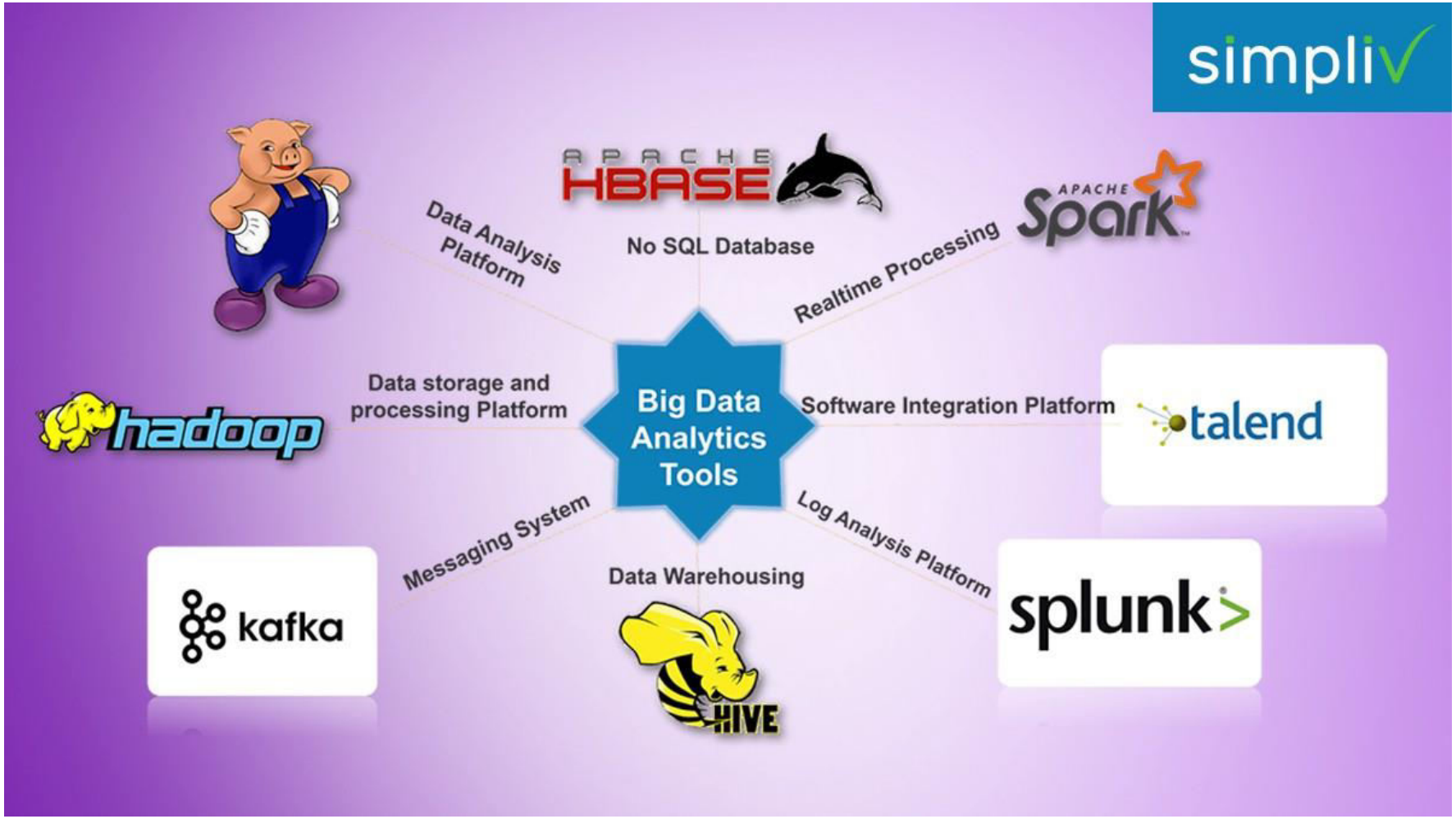
## Data Formats / Types : Big Data Storage

- Data storage for traditional and Big Data




T1: Fig 1.7 – Big data storage plan – RDBMS and NoSQL together

# BIG DATA
## Big Data Platforms

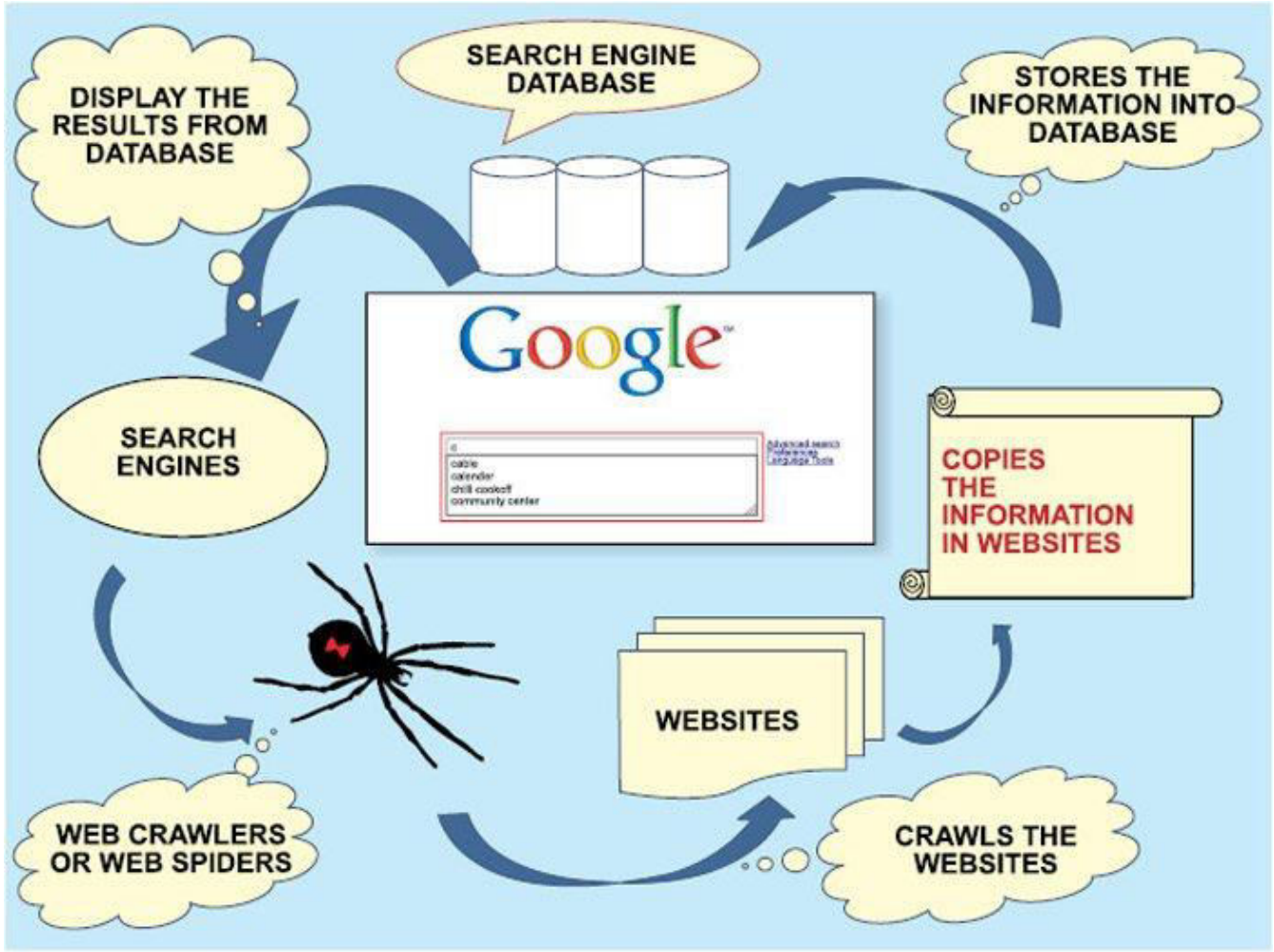



https://medium.com/@satyanageshan8/5-prominent-big-data-analytics-tools-to-learn-in-2019-ee1d4f5b98ce

# Big Data: Case Study – Google Search

# BIG DATA

## Case Study: How Google Search works?




https://www.electronicsforu.com/resources/how-google-search-engine-works

# BIG DATA

## Interesting facts about Google Search

- Google initial implementation
  - 24 M web pages, 322 M links
  - 5 days to compute Page Rank
    - Page rank is proportional to the popularity of the page
    - If more links point to a page, that page will be more popular
- Page Rank – Treats the web as a graph
  - User -
    - Starts at a random page
    - Takes a random link
  - Page Rank Assumptions –
    - The more popular a page - The better is its quality


https://en.wikipedia.org/wiki/PageRank


# THANK YOU

**Prafullata Kiran Auradkar**

Department of Computer Science and Engineering

**prafullatak@pes.edu**



BIG DATA

# Hadoop Distributed File System (HDFS)

**Prafullata Kiran Auradkar**

Department of Computer Science and Engineering

**prafullatak@pes.edu**

**Acknowledgements:**

Significant information in the slide deck presented through the Unit 1 of the course have been created by **Dr. K V Subramaniam's** and would like to acknowledge and thank him for the same. There have been some information which I might have leveraged from **Dr. H L Phalachandra's** slide contents too. I may have supplemented the same with contents from books and other sources from Internet and would like to sincerely thank, acknowledge and reiterate that the credit/rights for the same remain with the original authors/publishers only. These are intended for classroom presentation only.

# Why the need?

PES UNIVERSITY

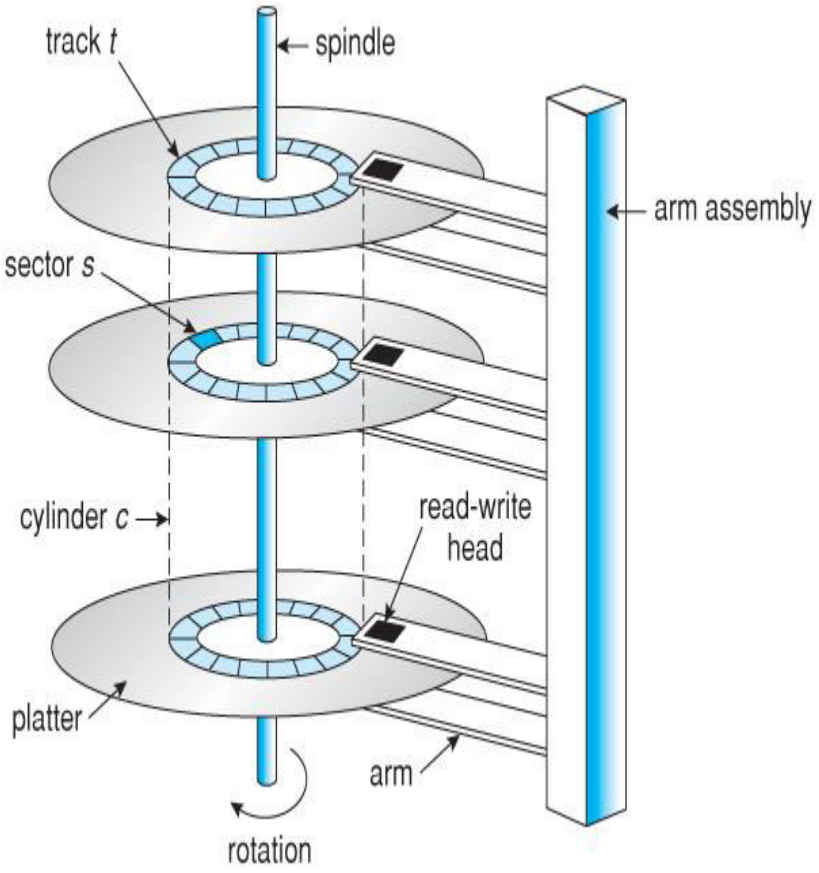
- As per RBI in May 2019,
  - #credit/debit card transactions~ 1.3 Billion (https://rbidocs.rbi.org.in/rdocs/ATM/PDFs/ATM052019E96EC259708C4ED9AD9E0C6B5E8B6DD5.PDF)
  - If each transaction requires about 10K of data

    **13 TB** of data

  - That's a lot of data and this is only for credit/debit card transactions
  - There are other transactions also
  - Suppose you want to look for fraudulent transactions
- How to store and process this data?

# Persistent Storage - Disks

- Block oriented devices
  - unit of data transfer is a block – e.g. 4KB on some Unix systems
- Can we store the persistent data directly on these blocks?
- Concerns:
  - Which block contains the data
  - Who will maintain the meta-data?
- Typically handled by Filesystems

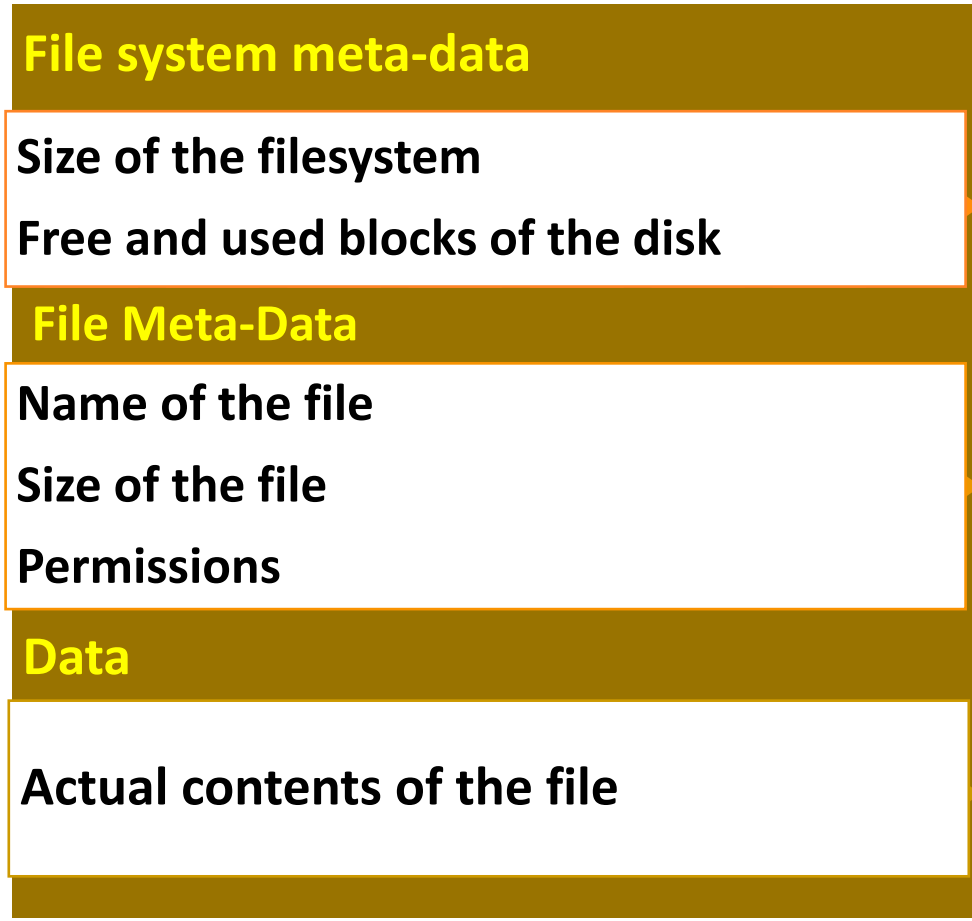


# File System Terminologies

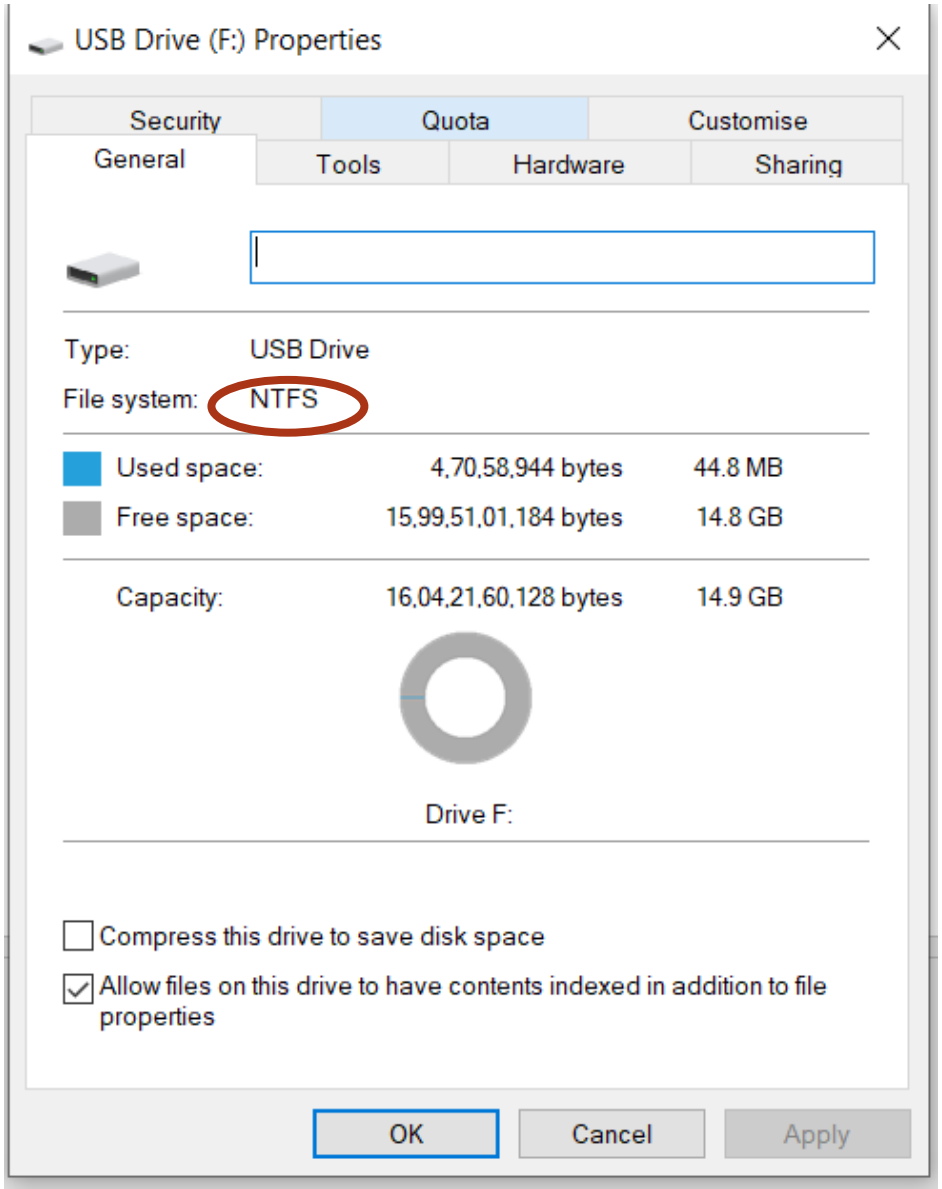
- **File system** is used to control how data is stored and retrieved. Without a file system, information placed in a storage medium would be one large body of data with no way to tell where one piece of information stops and the next begins.
- Data is separated and grouped into pieces and given a name called a **file**
- **Files** thus are named collection of related information on disks
- Desirable properties of files
  - Long term existence (stored on disk or other secondary storage and do not disappear when a user logs off )
  - Sharable between processes (have names and can have associated access permissions that permit controlled sharing)
  - Structure (files can be organized into hierarchical or more complex structure to reflect the relationships among files)
- The ***structure and logic rules*** used ***to manage the groups of information (files) and their names*** is what forms a file system. It also has a collection of functions that can be performed on files.
- File system also maintains a set of attributes associated with the file
- Typical operations include: Create, Delete, Open, Close, Read, Write
- Operating system can have multiple file systems
- File systems
  - Can be used on numerous different types of storage devices that uses different kinds of media
  - There are also many different kinds of file systems.
  - Each one has different structure and logic, properties of speed, flexibility, security, size and more.

# File System – design principles

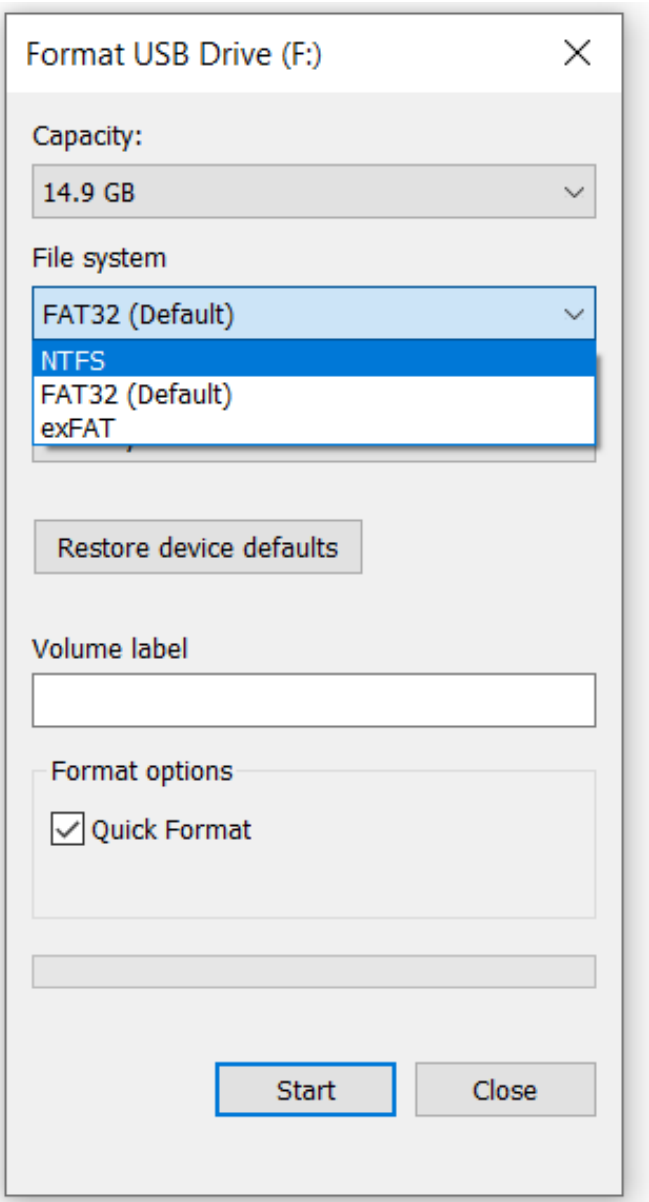
- Separates information into data and meta-data
- Mapping between these and block level abstraction provided by disks
- Block size

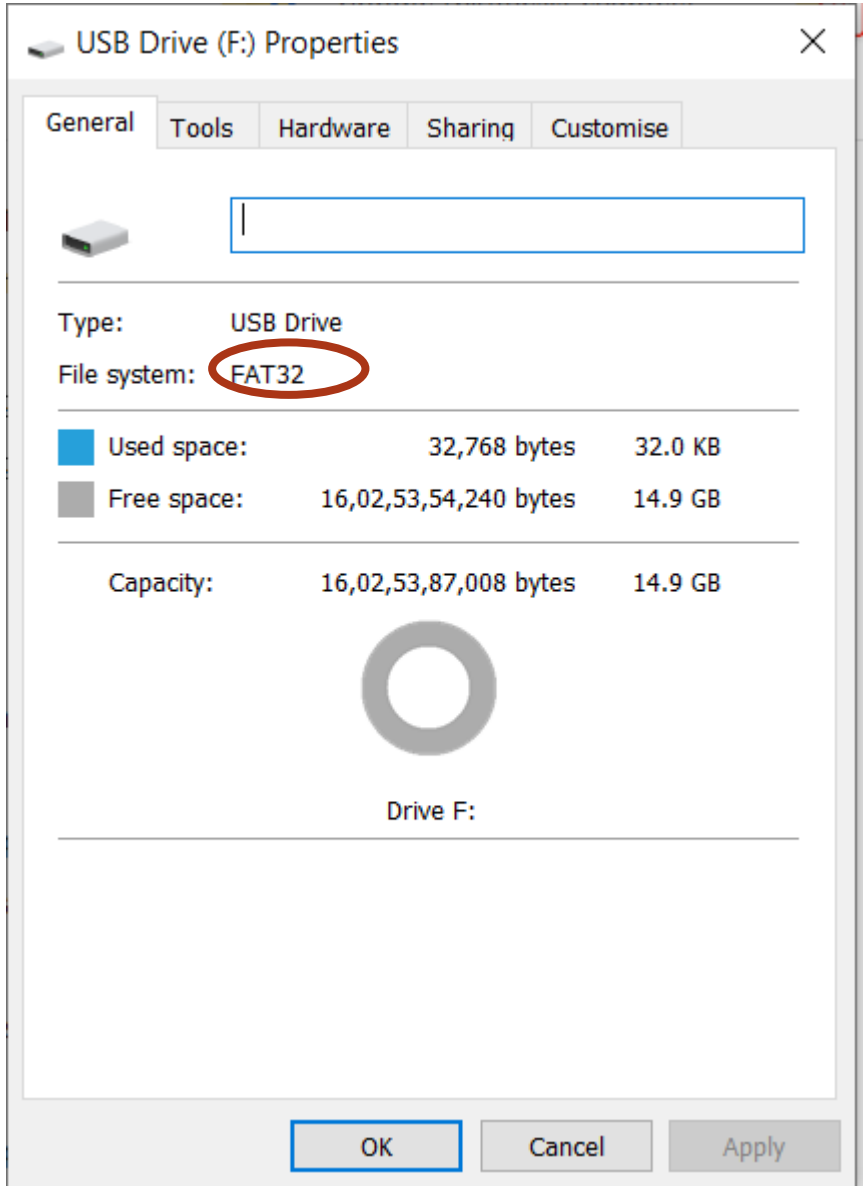


# File System – design principles

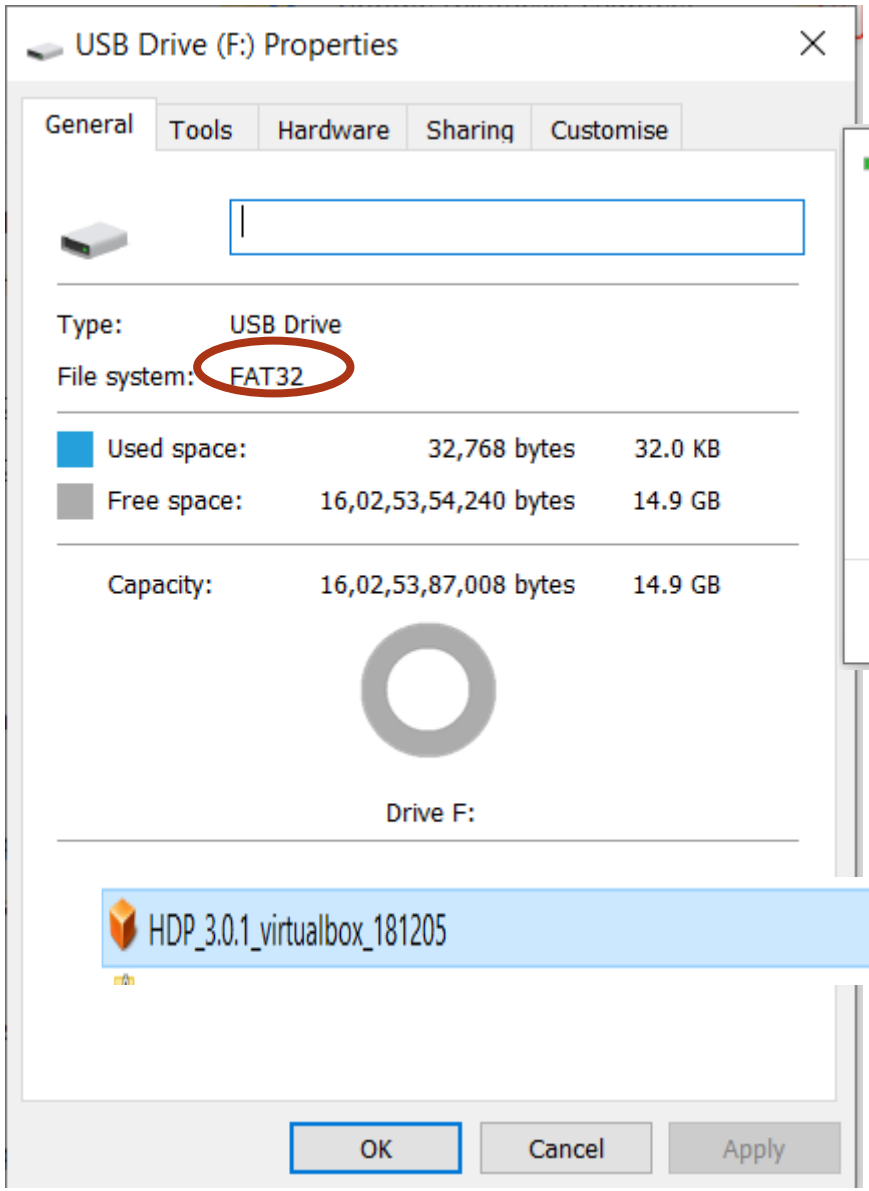


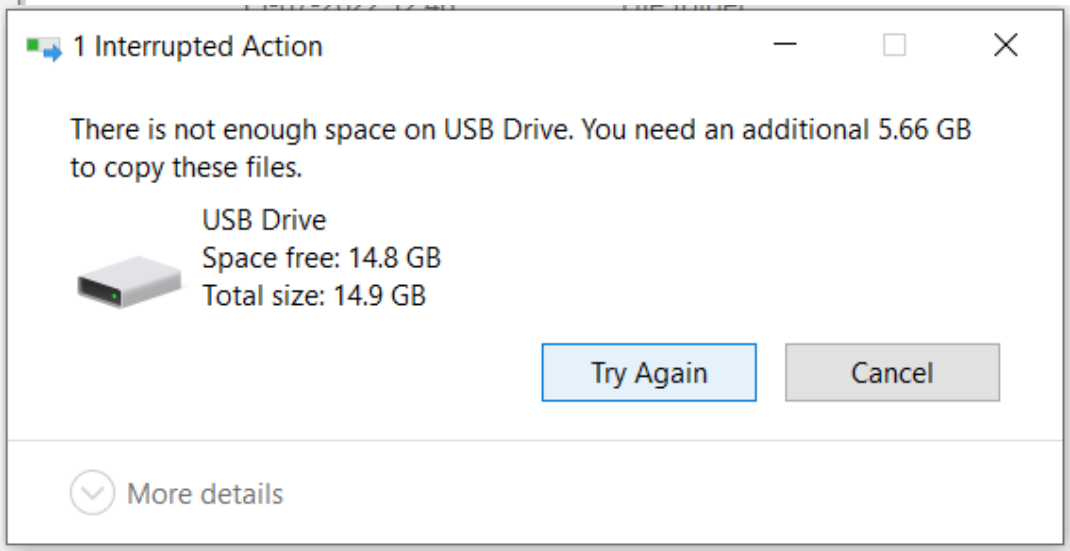
# File System – design principles

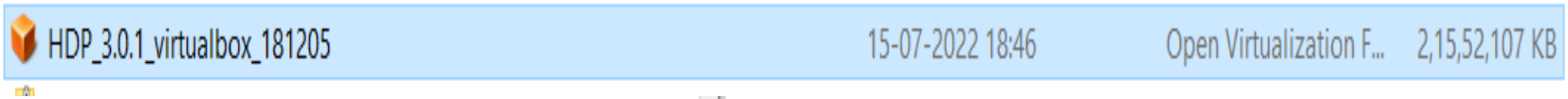


# File System – design principles

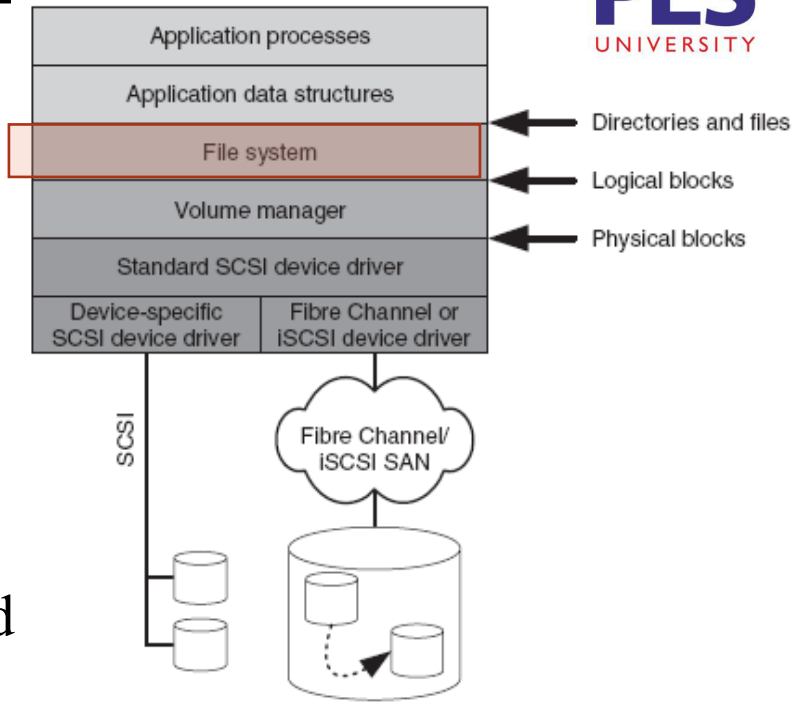


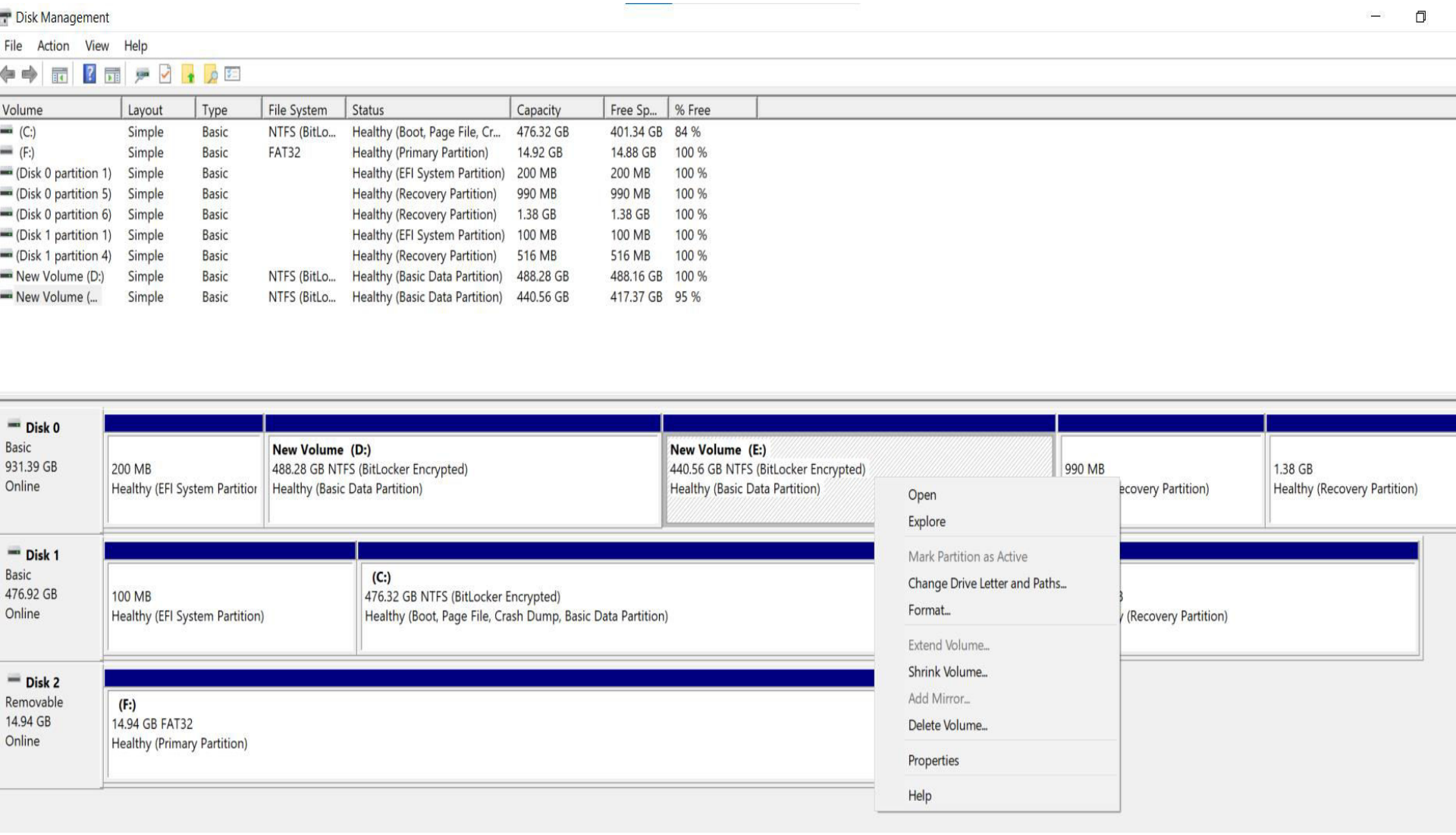
# File Systems : Local File Systems

- For storing data
  - An operating system will engage and schedule a storing level process
  - The filesystem manages where within the address space the data needs to be stored
  - Applications & users, use the storage capacity of the disks via directories and files.



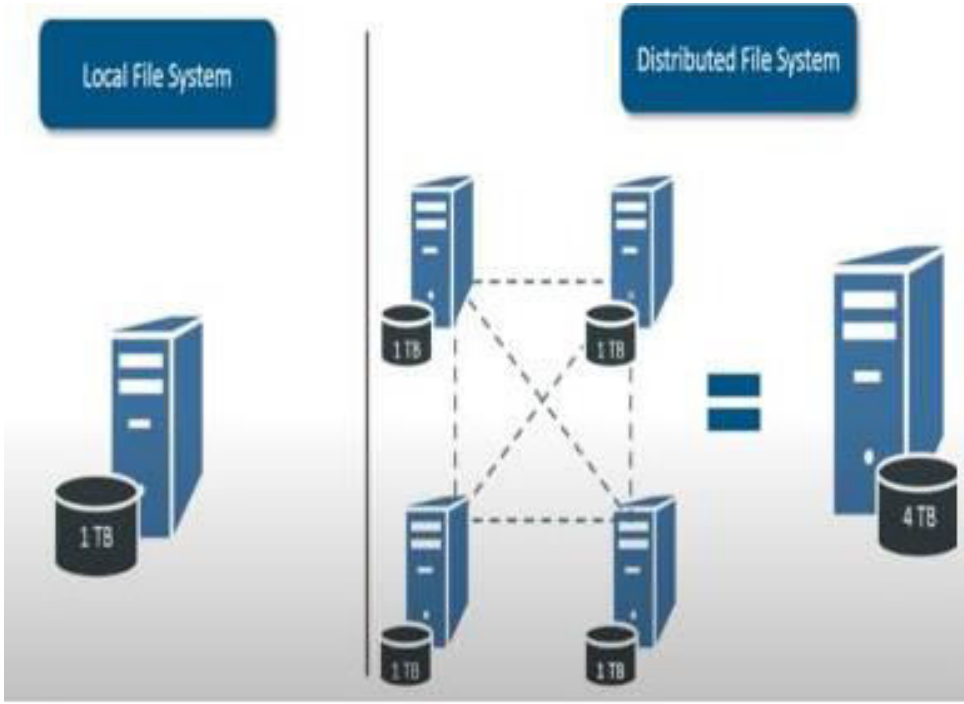

- File systems form an intermediate layer between applications & block oriented disks (with layers of volume management software which includes RAID controllers, virtualization systems) coming into play
- File systems and Volume managers are generic, and support all kinds of applications, and are not tuned for performance of any particular type of Applications

# File System – Local File Systems



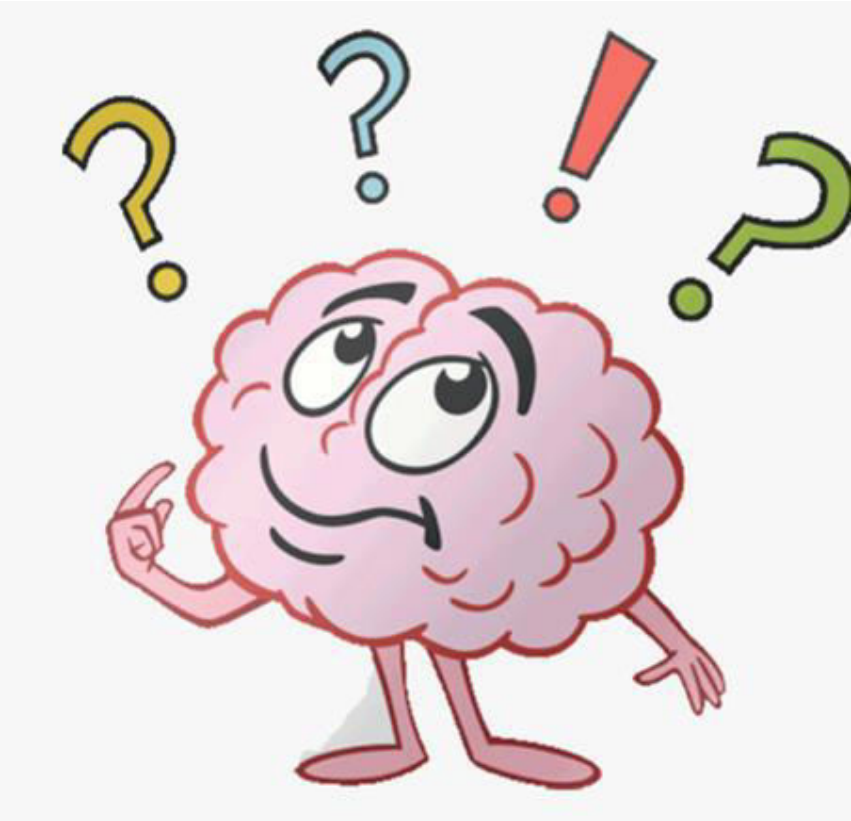
# Distributed File System

- Consider the case when data is so large that it cannot fit on a single disk
  - DFS manages files and folders across multiple computers
- DFS can organize and display files as if they are stored on one computer
  - It serves the same purpose as a traditional file system
- Designed to provide file storage and controlled access to files over local and wide area networks



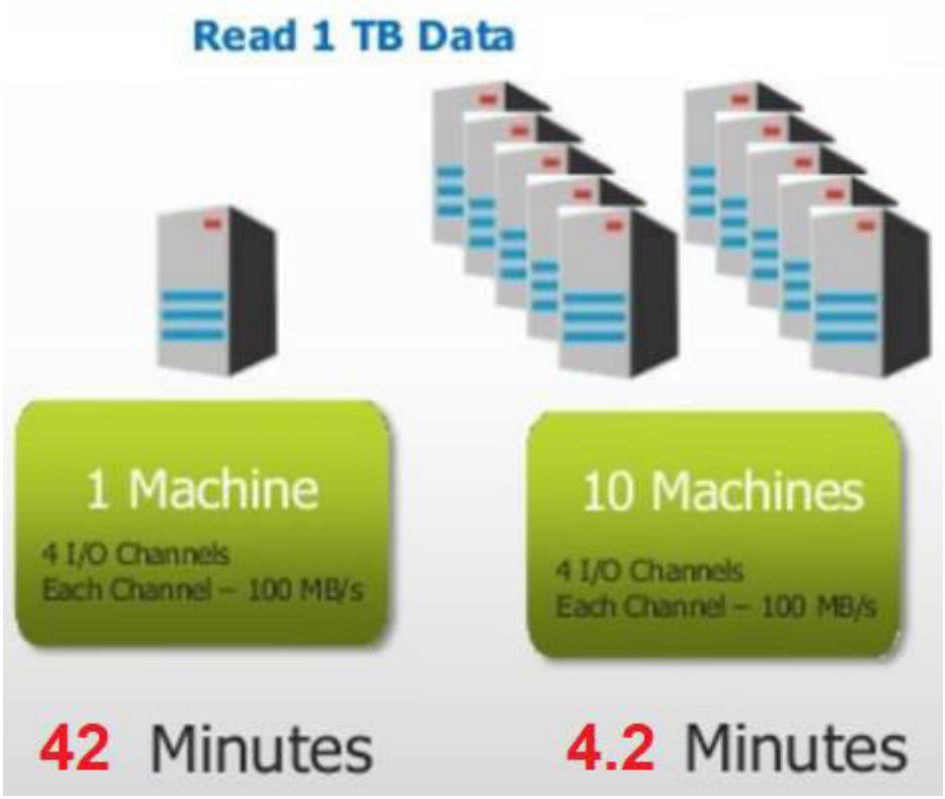
# Exercise

- Consider that you have 1 TB of data?
- Compare the time taken to read data in both the cases below
  - Single machine (4 I/O channels each channel 100mb/S)
  - 10 machines (each having 4 I/O channels each channel 100mb/s)



# Exercise

- Consider that you have 1 TB of data?
- Compare the time taken to read data in both the cases below
  - Single machine (4 I/O channels each channel 100mb/S)
  - 10 machines (each having 4 I/O channels each channel 100mb/s)



# What about existing FS?

PES UNIVERSITY

- NFS – Network File System
  - Small block sizes 8KB-16KB → large meta-data
  - Designed for large number of small files
- Handling scaling
  - Extending size of filesystem requires growing the volume
  - Can mainly handle scaling up.
- Scaling out – the need
  - A single machine cannot handle the load. Need a cluster
  - Originally not by design.
    - Was added later on need (Clustered Shared Volume in NTFS).
  - Need to support distributed operations

HDFS is based on the opensource version of GFS

# What about existing FS?

PES UNIVERSITY

**HDFS – Inspired by GFS**

GFS – Google File System (2003)

Distributed File system on a cluster of machines

Developed by Doug Cutting and Mike Cafarella

Origin - Apache Nutch

- Goal : web search engine on 1 Billion Pages

Open source

# HDFS – Hadoop Distributed File System

PES UNIVERSITY

"HDFS is a filesystem designed for storing very large files with streaming data access patterns, running on clusters of commodity hardware."

**Very large**

- Files can be MB/GB/TB in size
- Hadoop clusters that are PB are currently operational

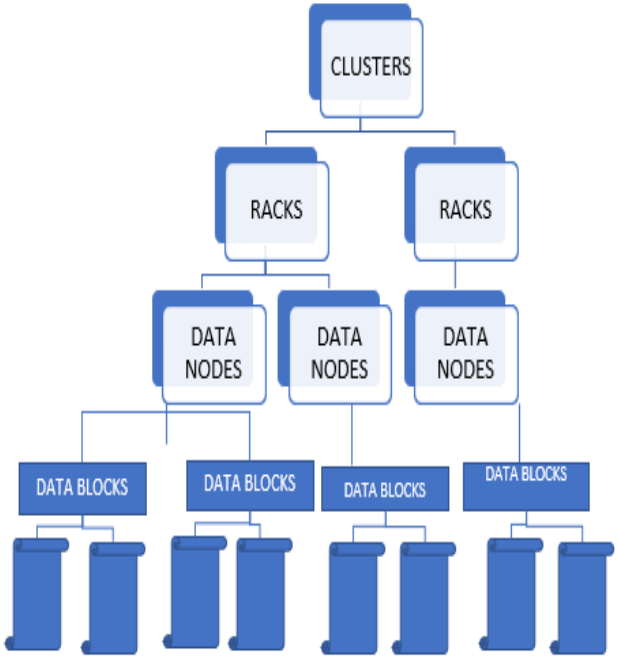
**Read Mostly data**

- most efficient data processing pattern is a write-once, read-many-times pattern.
- Each analysis will involve a large proportion of the dataset
- time to read the whole dataset is more important than the latency in reading the first record.

**Commodity hardware**

- Hadoop doesn't require expensive, highly reliable hardware
- Designed to run on clusters of commodity hardware

# Hadoop Distributed File System



# Exercise

- If you want to store a file on disk what constitutes
  - Data
  - Metadata

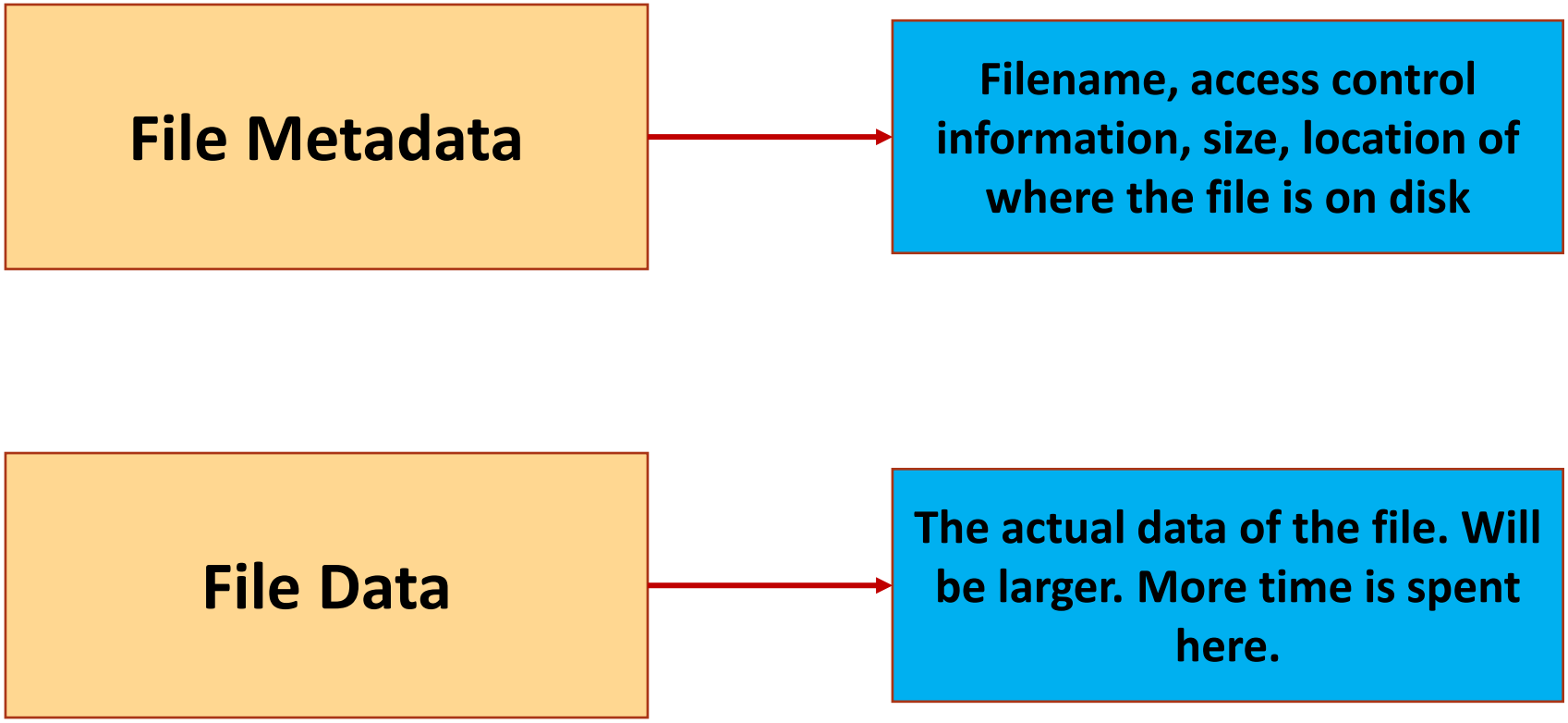
Data: much larger in size. Occupies multiple blocks

Metadata: smaller compared to data. Only information on filenames and blocks it occupies.

- What are their access patterns
  - How often do you think each one will be accessed during a normal file read
- How large are they (comparatively)? Why is this important?

Since data is much larger. Most time spent in fetching data

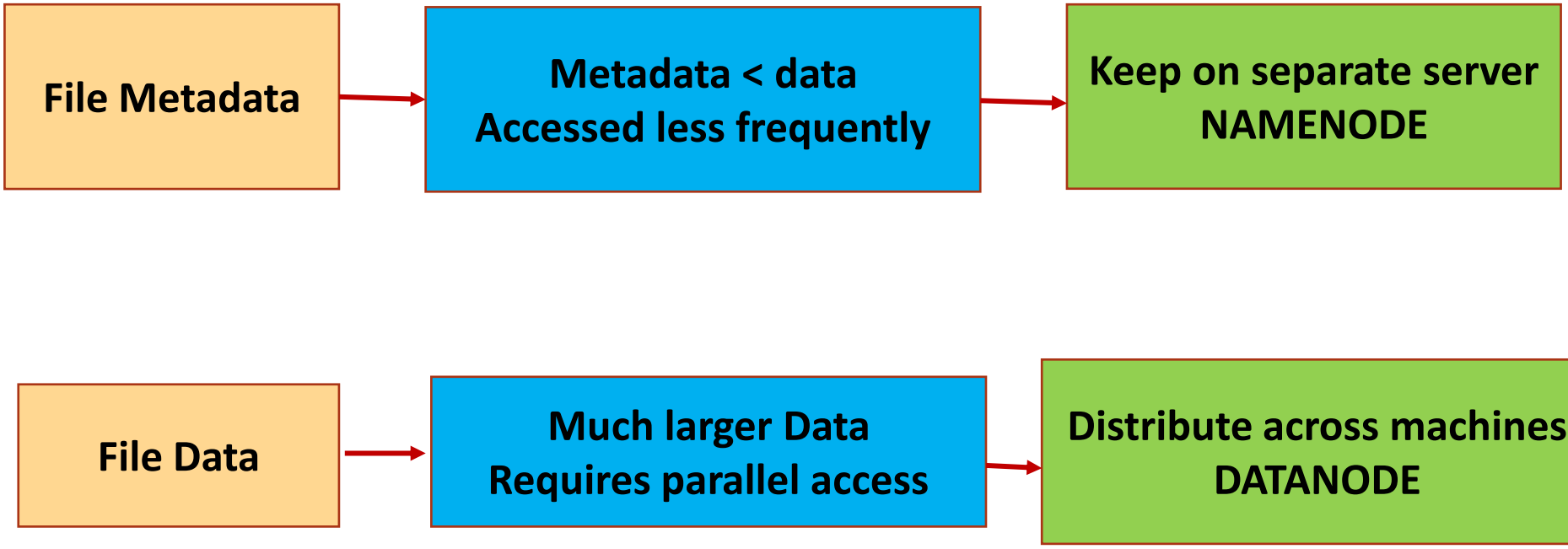
# HDFS Motivation



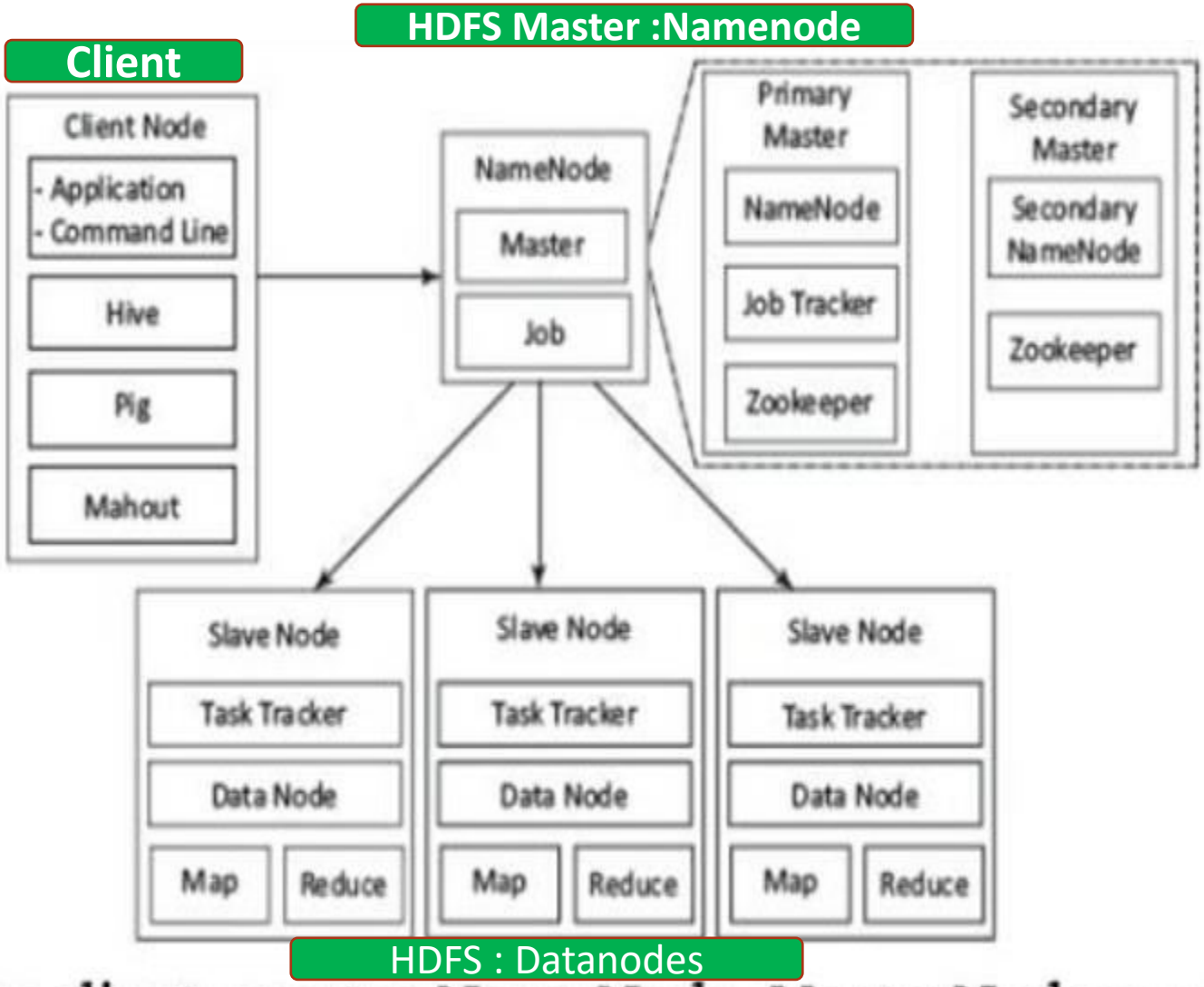
# HDFS Motivation

PES UNIVERSITY

## Solution

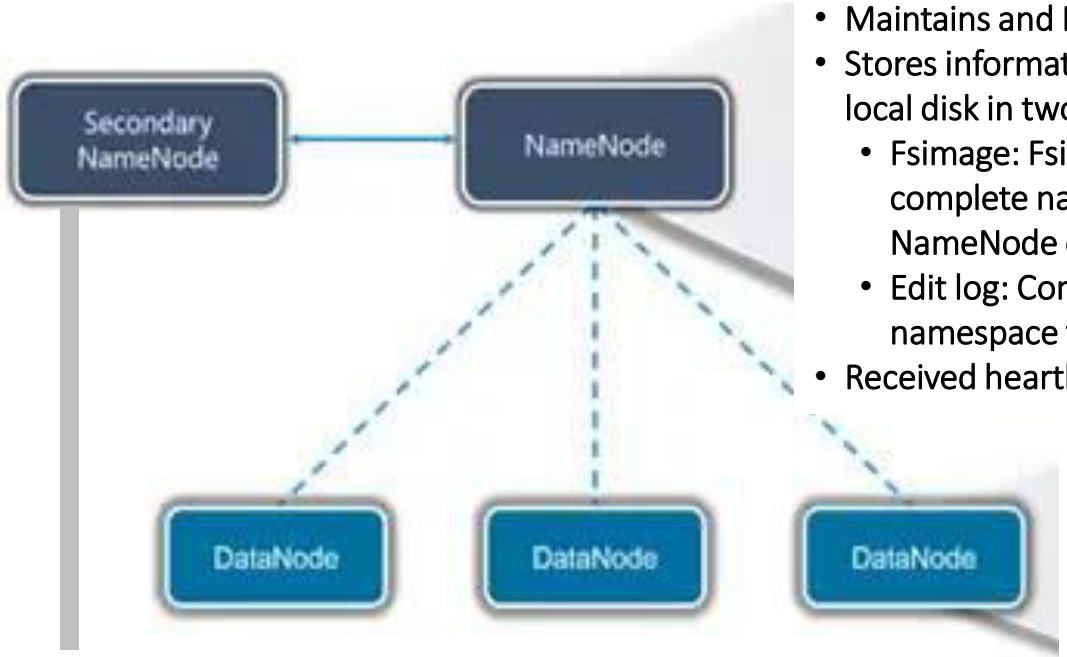


# HDFS – Master Slave Architecture




The client, master NameNode, MasterNodes and slave nodes

# HDFS - Architecture




## NameNode

- Master Daemon
- Maintains and Manages DataNodes
- Stores information about blocks locations, permissions, etc. on the local disk in two files:
  - Fsimage: Fsimage stands for File System image. It contains the complete namespace of the Hadoop file system since the NameNode creation.
  - Edit log: Contains all recent changes performed to the file system namespace to the most recent Fsimage
- Received heartbeat and block report from all the DataNodes

## Secondary NameNode

- Helper Node to Primary NameNode
- Housekeeping backup (not hot standby)
- Supports primary NameNode by downloading and merging the Fsimage file and edit logs file, updates the Fsimage and refreshes the edit logs
- It then sends this updated image to NameNode and enables NameNode to start faster

## DataNodes

DataNodes are the slave nodes and the workhorses of HDFS

These are inexpensive commodity hardware storing blocks of a file

These are responsible for serving the client read/write requests

Based on the instruction from the NameNode, DataNodes performs block creation, replication, and deletion.

DataNodes send a heartbeat to NameNode to report the health of HDFS.

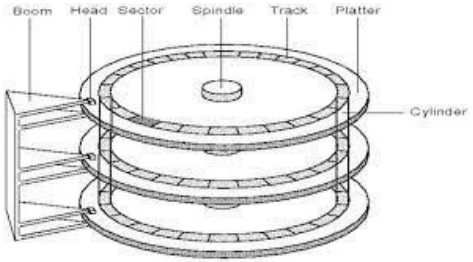
DataNodes also sends block reports to NameNode to report the list of blocks it contains.

# HDFS Blocks: What

- Disk Blocks:
  - Minimum data that can be read written.
  - Typically 512 bytes.
- HDFS blocks
  - Much larger unit
    - 128MB (in v2)
- Files in HDFS are broken into block-sized chunks, which are stored as independent units.
- A file in HDFS that is smaller than a single block does not occupy a full block's worth of underlying storage.
  - Use as many disk blocks as necessary.


File

HDFS Blocks




Mapped to disk blocks

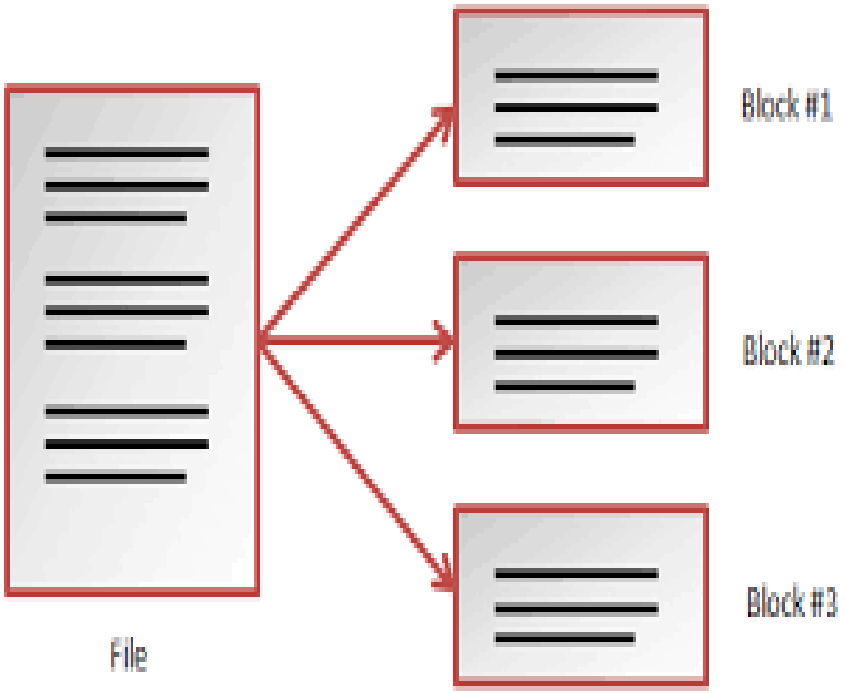
# HDFS Blocks: What


File

```
shashank@intel2:~$ hdfs dfs -ls test
22/08/03 16:24:30 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 10 items
-rw-r--r--   3 shashank supergroup          0 2021-01-05 19:06 test/_SUCCESS
-rw-r--r--   3 shashank supergroup        803 2021-01-05 19:06 test/_master.heap
-rw-r--r--   3 shashank supergroup  134217697 2021-01-05 19:05 test/part-00000_data_00001
-rw-r--r--   3 shashank supergroup  134217698 2021-01-05 19:05 test/part-00001_data_00001
-rw-r--r--   3 shashank supergroup  134217670 2021-01-05 19:05 test/part-00002_data_00001
-rw-r--r--   3 shashank supergroup  134217699 2021-01-05 19:05 test/part-00003_data_00001
-rw-r--r--   3 shashank supergroup  134217694 2021-01-05 19:06 test/part-00004_data_00001
-rw-r--r--   3 shashank supergroup  134217727 2021-01-05 19:06 test/part-00005_data_00001
-rw-r--r--   3 shashank supergroup  134217719 2021-01-05 19:06 test/part-00006_data_00001
-rw-r--r--   3 shashank supergroup  134217657 2021-01-05 19:06 test/part-00007_data_00001
```

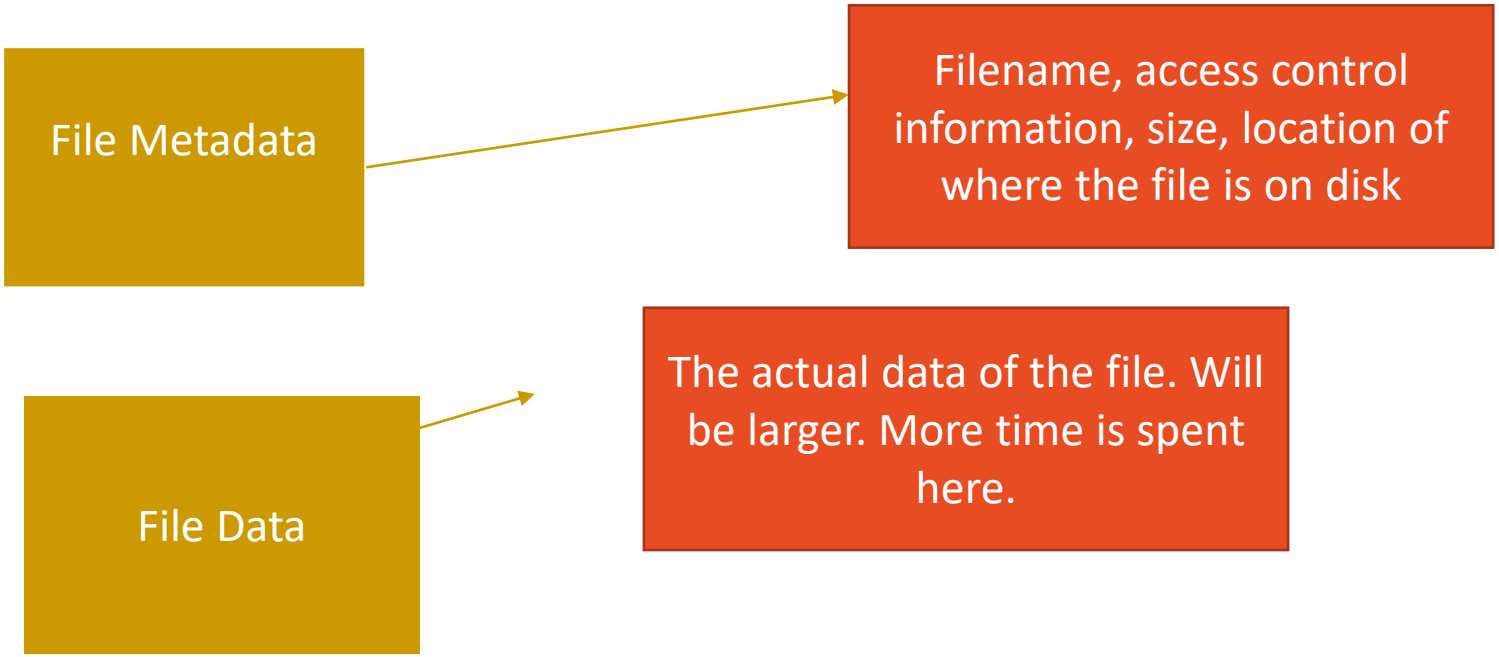
# HDFS Blocks: Why

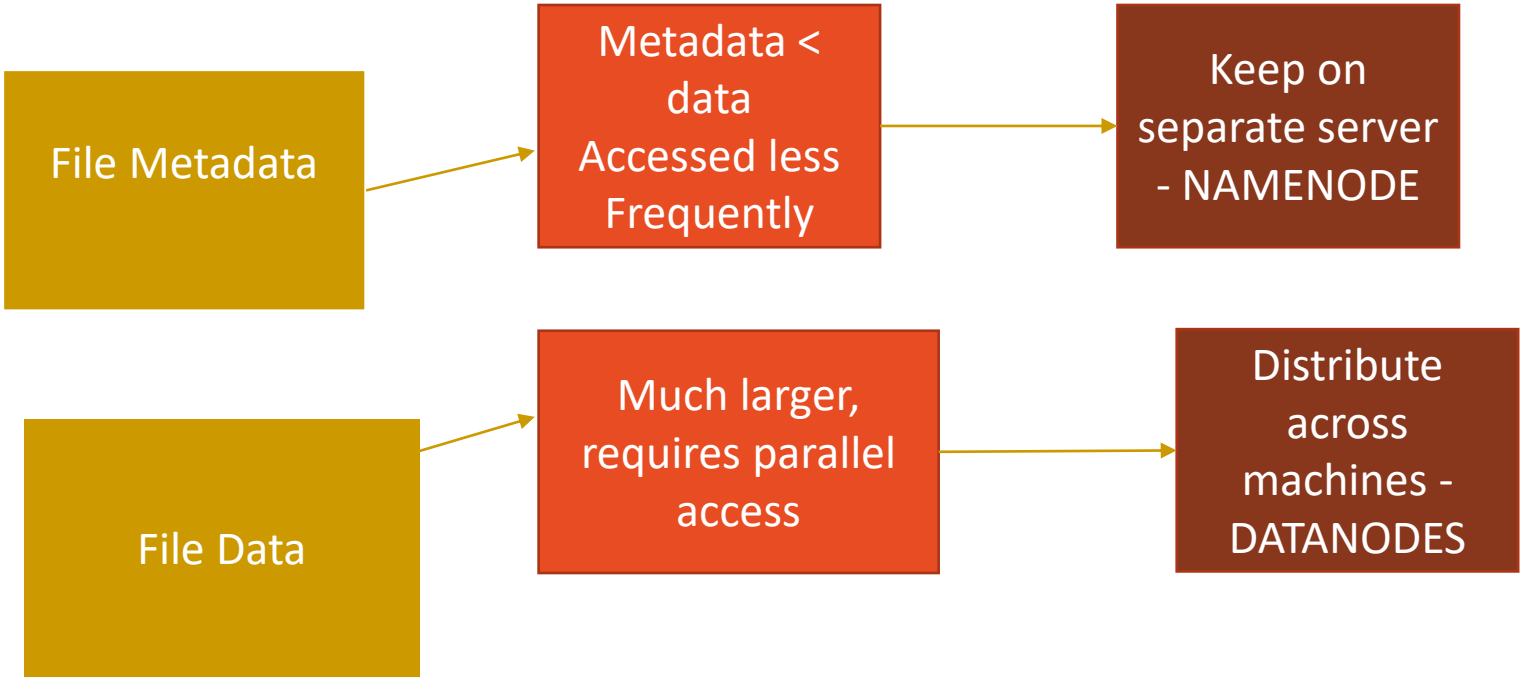
- Benefits of block abstraction.
  - A file can be larger than any single disk in the network.
    - Files can be distributed across disks
  - Simplifies the storage subsystem
  - Blocks fit well with replication for providing fault tolerance and availability.
- % **`hadoop fsck -files -blocks`**
  - will list the blocks that make up each file in the filesystem



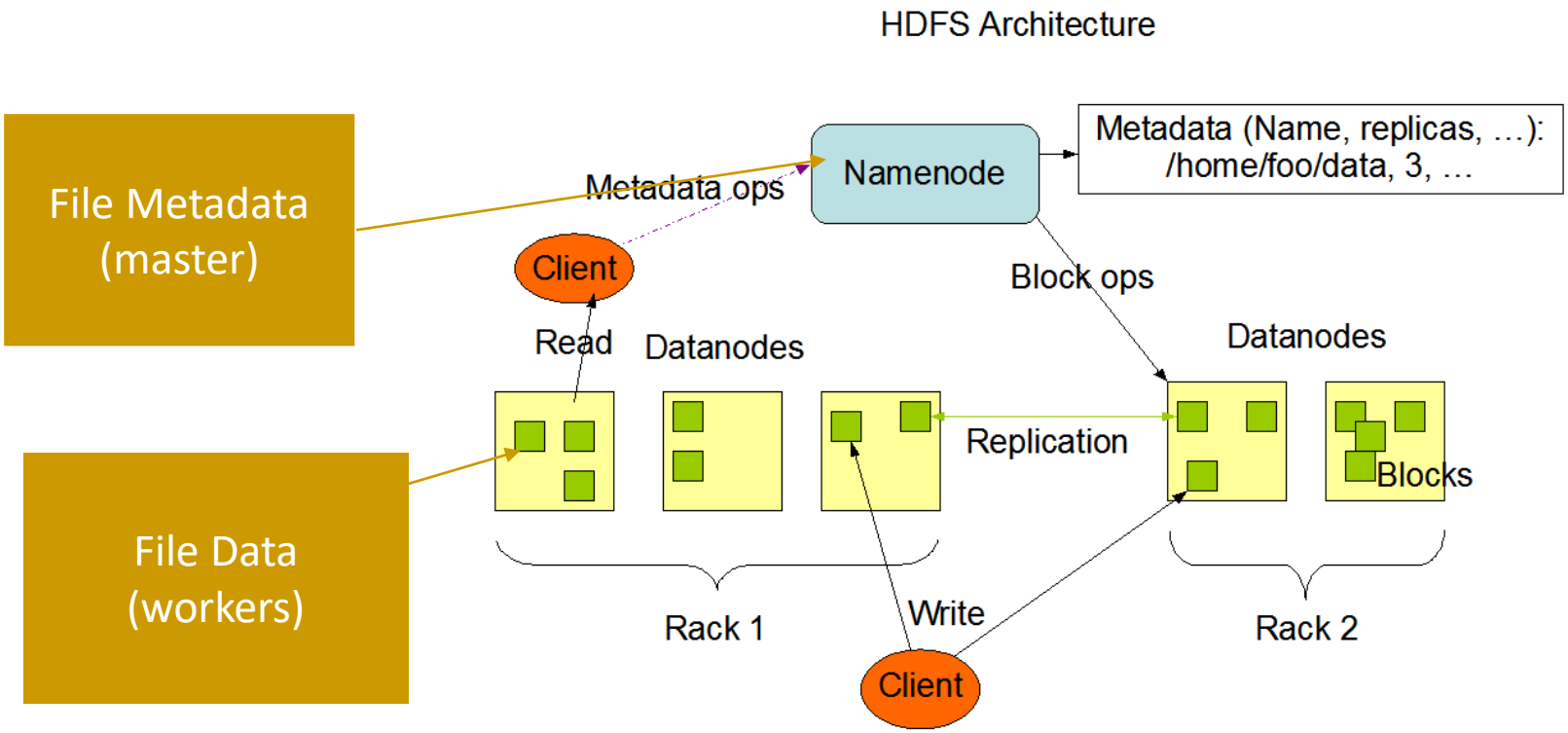
```
shashank@intel2:~$ hdfs fsck test
22/08/03 16:25:17 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Connecting to namenode via http://localhost:50070/fsck?ugi=shashank&path=%2Fuser%2Fshashank%2Ftest
FSCK started by shashank (auth:SIMPLE) from /127.0.0.1 for path /user/shashank/test at Wed Aug 03 16:25:18 IST 2022
..
/user/shashank/test/_master.heap:  Under replicated BP-1245458128-10.10.1.145-1600691084084:blk_1073788548_47736. Target Replicas is 3 but found 1 replica(s).
.
/user/shashank/test/part-00000_data_00001:  Under replicated BP-1245458128-10.10.1.145-1600691084084:blk_1073788528_47716. Target Replicas is 3 but found 1 replica(s).
.
/user/shashank/test/part-00001_data_00001:  Under replicated BP-1245458128-10.10.1.145-1600691084084:blk_1073788529_47717. Target Replicas is 3 but found 1 replica(s).
.
/user/shashank/test/part-00002_data_00001:  Under replicated BP-1245458128-10.10.1.145-1600691084084:blk_1073788530_47718. Target Replicas is 3 but found 1 replica(s).
.
/user/shashank/test/part-00003_data_00001:  Under replicated BP-1245458128-10.10.1.145-1600691084084:blk_1073788531_47719. Target Replicas is 3 but found 1 replica(s).
.
/user/shashank/test/part-00004_data_00001:  Under replicated BP-1245458128-10.10.1.145-1600691084084:blk_1073788532_47720. Target Replicas is 3 but found 1 replica(s).
.
/user/shashank/test/part-00005_data_00001:  Under replicated BP-1245458128-10.10.1.145-1600691084084:blk_1073788533_47721. Target Replicas is 3 but found 1 replica(s).
.
/user/shashank/test/part-00006_data_00001:  Under replicated BP-1245458128-10.10.1.145-1600691084084:blk_1073788534_47722. Target Replicas is 3 but found 1 replica(s).
.
/user/shashank/test/part-00007_data_00001:  Under replicated BP-1245458128-10.10.1.145-1600691084084:blk_1073788535_47723. Target Replicas is 3 but found 1 replica(s).
Status: HEALTHY
```

```
shashank@intel2:~$ hdfs dfs -ls lulc200
22/08/03 16:34:29 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 3 items
-rw-r--r--   3 shashank supergroup          0 2020-12-09 01:50 lulc200/_SUCCESS
-rw-r--r--   3 shashank supergroup        102 2020-12-09 01:59 lulc200/_master.heap
-rw-r--r--   3 shashank supergroup  217419422 2020-12-09 01:55 lulc200/finalResult.wkt
shashank@intel2:~$ hdfs fsck  lulc200
22/08/03 16:34:49 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Connecting to namenode via http://localhost:50070/fsck?ugi=shashank&path=%2Fuser%2Fshashank%2Flulc200
FSCK started by shashank (auth:SIMPLE) from /127.0.0.1 for path /user/shashank/lulc200 at Wed Aug 03 16:34:49 IST 2022
..
/user/shashank/lulc200/_master.heap:  Under replicated BP-1245458128-10.10.1.145-1600691084084:blk_1073787389_46577. Target Replicas is 3 but found 1 replica(s).
.
/user/shashank/lulc200/finalResult.wkt:  Under replicated BP-1245458128-10.10.1.145-1600691084084:blk_1073787376_46564. Target Replicas is 3 but found 1 replica(s).

/user/shashank/lulc200/finalResult.wkt:  Under replicated BP-1245458128-10.10.1.145-1600691084084:blk_1073787377_46565. Target Replicas is 3 but found 1 replica(s).
Status: HEALTHY
 Total size:    217419524 B
 Total dirs:    1
 Total files:   3
 Total symlinks:                0
 Total blocks (validated):      3 (avg. block size 72473174 B)
 Minimally replicated blocks:   3 (100.0 %)
 Over-replicated blocks:        0 (0.0 %)
 Under-replicated blocks:       3 (100.0 %)
 Mis-replicated blocks:         0 (0.0 %)
 Default replication factor:    3
 Average block replication:     1.0
 Corrupt blocks:                0
 Missing replicas:              6 (66.666664 %)
 Number of data-nodes:          1
 Number of racks:               1
FSCK ended at Wed Aug 03 16:34:49 IST 2022 in 3 milliseconds


The filesystem under path '/user/shashank/lulc200' is HEALTHY
shashank@intel2:~$
```

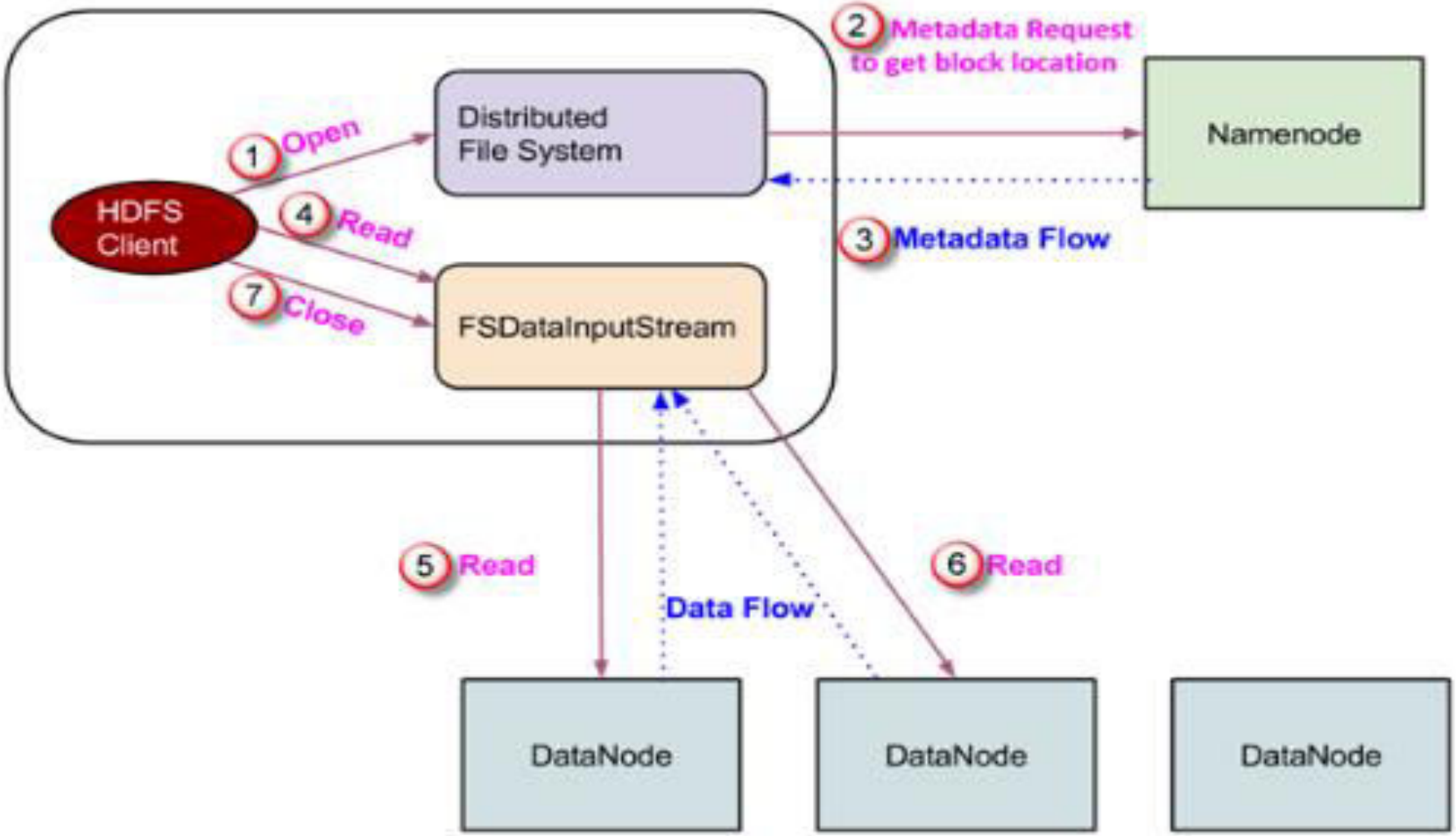

File Metadata
Filename, access control information, size, location of where the file is on disk
File Data
The actual data of the file. Will be larger. More time is spent here.


File Metadata
Metadata < data Accessed less Frequently
Keep on separate server - NAMENODE
File Data
Much larger, requires parallel access
Distribute across machines - DATANODES

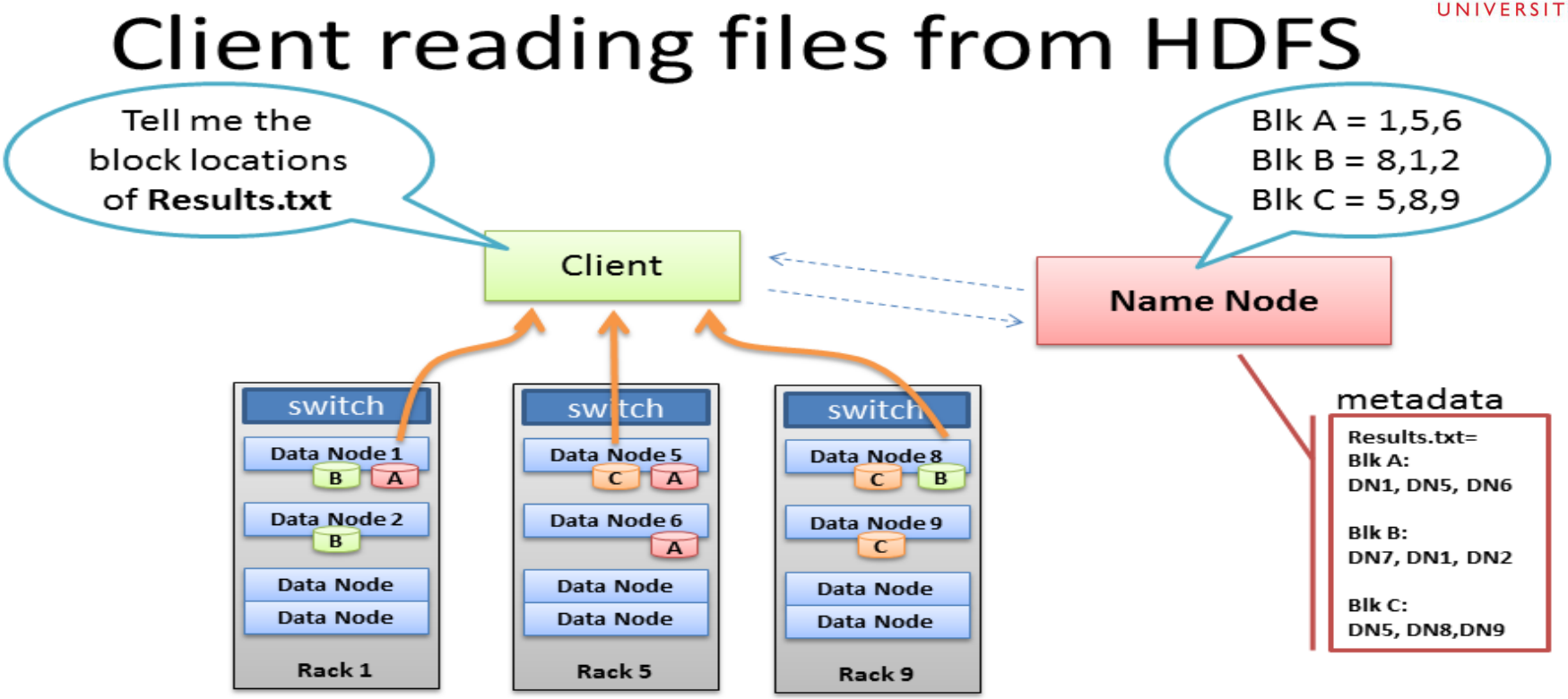
# HDFS Architecture



# Reading a file

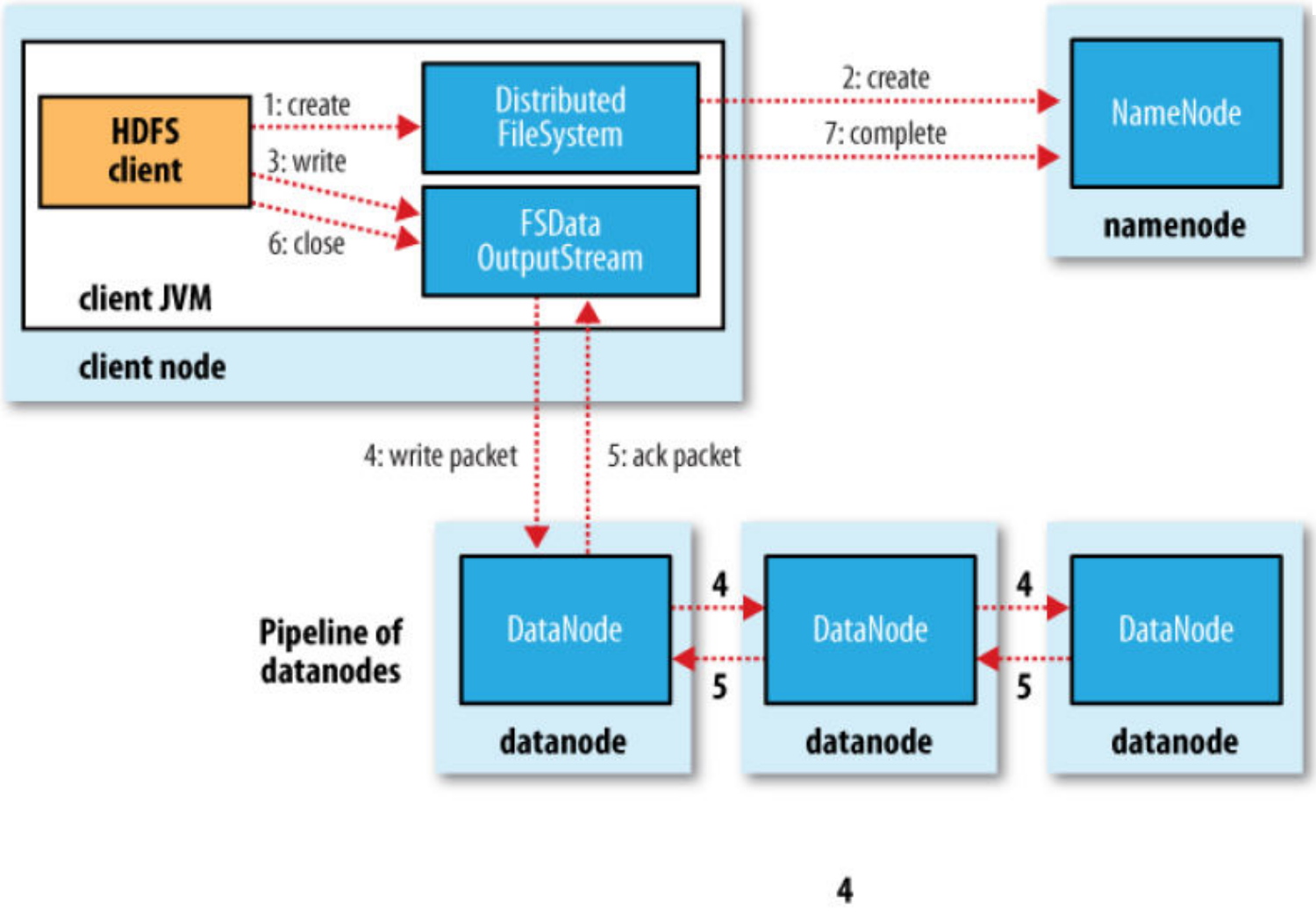


# Reading a file




- Client receives Data Node list for each block
- Client picks first Data Node for each block
- Client reads blocks sequentially

# Writing a File

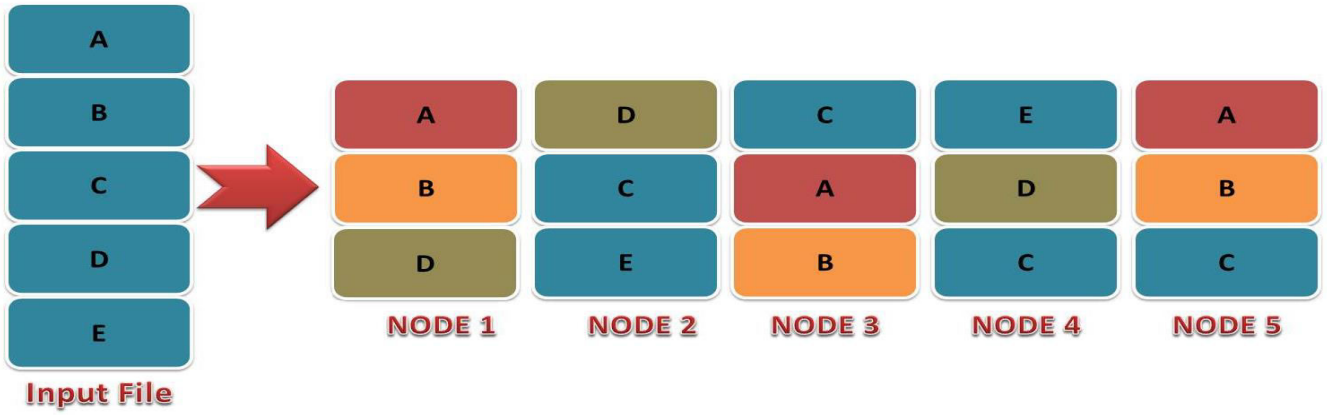


# HDFS Fault Tolerance - 1

- Fault tolerance in Hadoop HDFS refers to the working strength of a system in unfavorable conditions and how that system can handle such a situation
- HDFS is highly fault tolerant and handled faults.
  - There are approaches used for handling faults of data nodes or disks holding data blocks.
    - These could be based on replication, to a replication factor till Hadoop 3. So whenever if any machine/disk in the cluster goes down, then data is accessible from other machines/disk in which the same copy of data was created
    - Using Erasure coding in Hadoop 3
  - There are also approaches to handle faults of Name Node and the availability of the metadata pointing to the data blocks

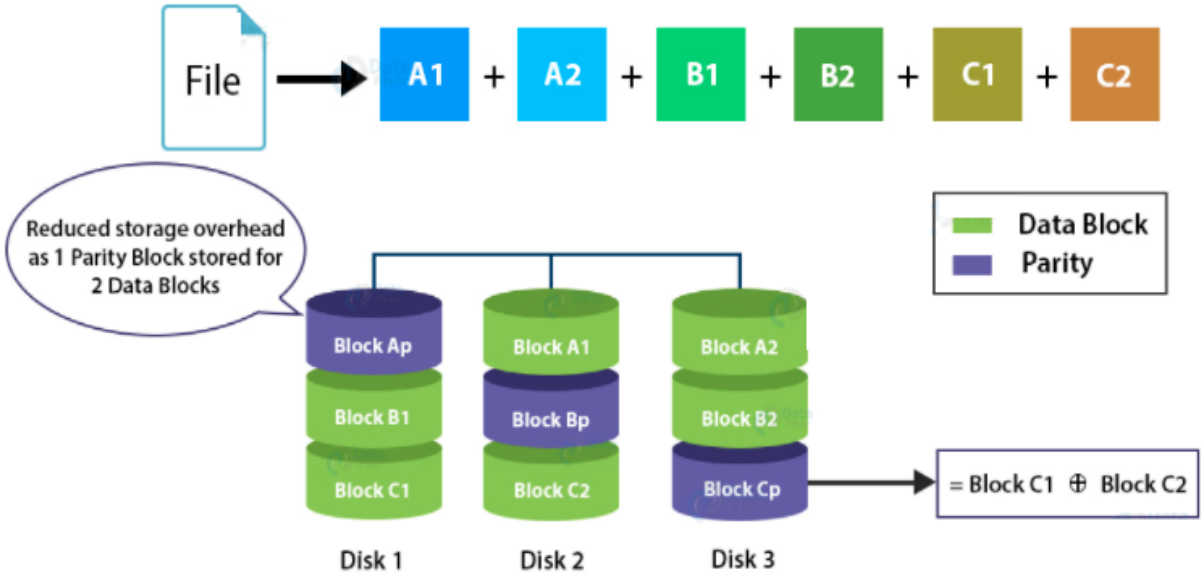
# HDFS Fault Tolerance - 2

- Fault tolerance of data using Replicas is achieved by creating a replica of users' data based on the replication factor on different machines in the HDFS cluster.
- So if any machine in the cluster goes down, then data is accessible from other machines in which the same copy of data was created



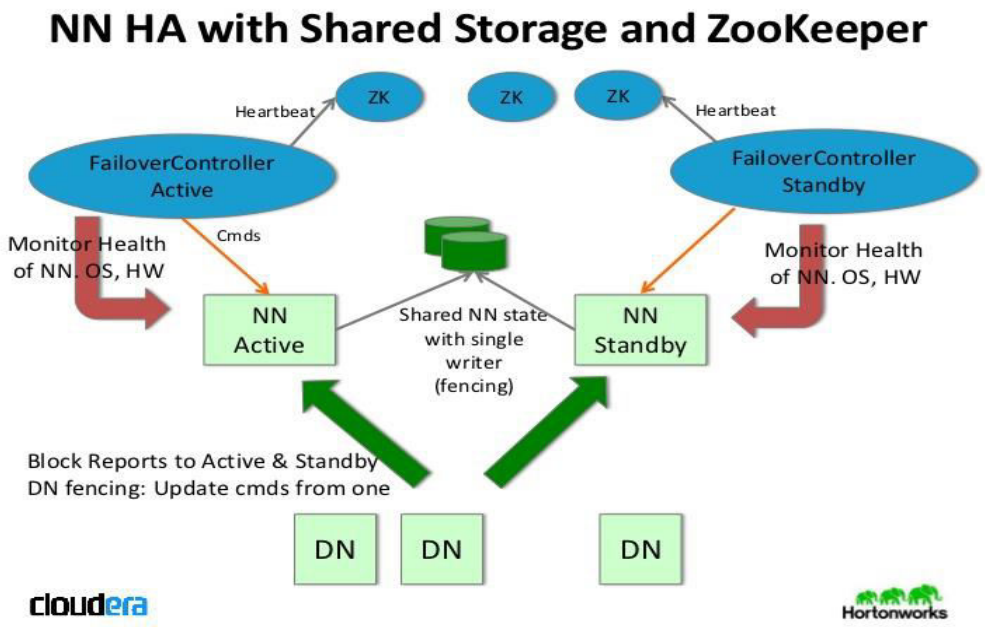
# HDFS Fault Tolerance - 3

- Fault tolerance of data could also be based on Erasure coding
- Erasure coding works similar to RAID by striping the file into small units of sequential blocks and storing them consecutively on various disks.
- For each strip of the original dataset, a certain number of parity cells are calculated using erasure coding algorithm called encoding and stored. If any of the machines fails, the block can be recovered from the parity cell.
- The error in any striping cell can be recovered from the calculation based on the remaining data and parity cells; the process known as decoding.



# HDFS Failover

- Failover Controller
  - Handles transition from active→standby
  - Failover controllers are pluggable,
  - ZooKeeper to ensure that only one namenode is active.
- Manual Trigger for maintenance
- Ungraceful failure
  - Is it a real failure?
  - Slow network – active treated as failed, but is up and running




HDFS uses Zookeeper for

- Failure Detection
  - Namenode maintains persistent session with ZK
  - If Machine crashes
    - Session would expire
    - Notify the other NameNode of the crash
- Active Namenode Election
  - On crash, other node takes exclusive lock
  - Indicating that it is the leader

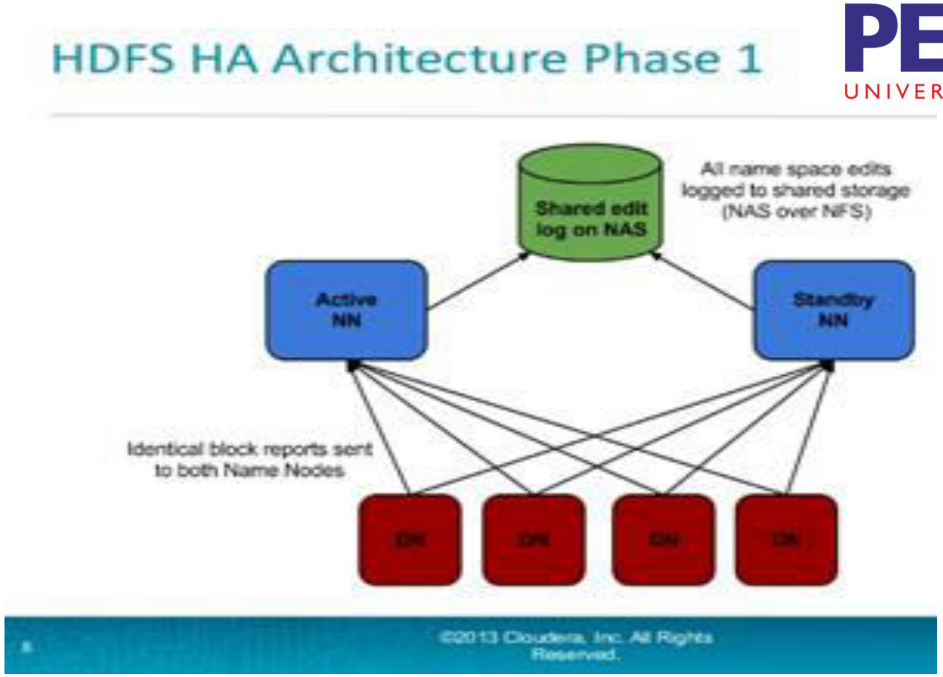
# HDFS High Availability (post 2.x)

PES UNIVERSITY

to failure
- Should recover quickly

## Configure
- Active – Standby configuration
- Edit logs
  - Store on shared storage
- Block Mappings
  - Stored in memory

## On Failure
- Standby takes over using shared edit logs and in-memory block mappings




| | | |
|---|---|---|
| Name node | Edit logs | Shared storage |
| Datanodes | block reports to both namenodes | Block mappings stored in memory |
| Clients | Transparency | Provide failover |

# Where HDFS doesn't score

PES UNIVERSITY

- Low latency data access
- Lots of small files
- Multiple writers, arbitrary file modifications
- Not POSIX compliant

# Where HDFS doesn't score

*Low Latency Data Access:*

Applications that require low-latency access to data, in the tens of milliseconds range.

The high throughput offered by HDFS may be at the cost of latency.

*Lots of Small Files:*

namenode holds filesystem metadata in memory,

limit to the number of files in a filesystem is governed by the amount of memory on the namenode.

lAs a rule of thumb, each file, directory, and block takes about 150 bytes.

- Million files → 300MB of memory.
- Billion files → ?

*Multiple writers, arbitrary file modifications:*

Files in HDFS may be written to by a single writer.

Writes are always made at the end of the file.

No support for multiple writers, or for modifications at arbitrary offsets in the file.

Not Posix compliant

Can't mount and use standard FS commands

Need a separate client


# THANK YOU

**Prafullata Kiran Auradkar**

Department of Computer Science and Engineering

**prafullatak@pes.edu**


# Big Data

## Map Reduce Programming model & Architecture

**Prafullata Kiran Auradkar**

Department of Computer Science and Engineering

**prafullatak@pes.edu**

**Acknowledgements:**

Significant information in the slide deck presented through the Unit 1 of the course have been created by **Dr. K V Subramaniam** and would like to acknowledge and thank him for the same. There have been some information which I might have leveraged from **Dr. H L Phalachandra's** slide contents too. I may have supplemented the same with contents from books and other sources from Internet and would like to sincerely thank, acknowledge and reiterate that the credit/rights for the same remain with the original authors/publishers only. These are intended for classroom presentation only.

PES UNIVERSITY

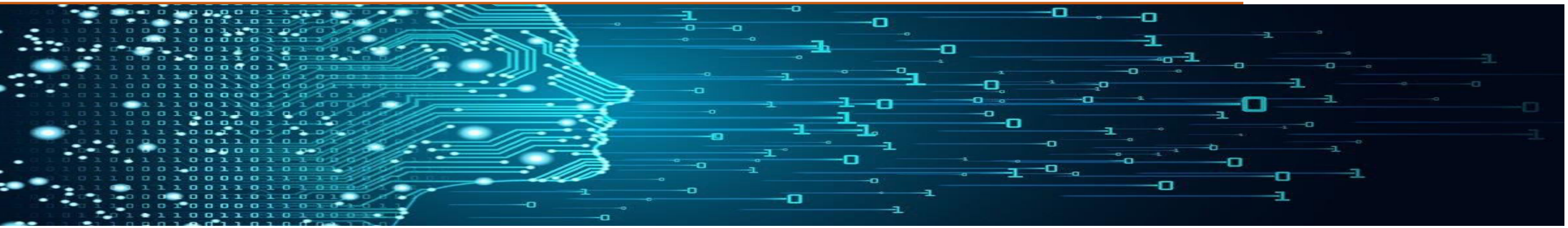
## What we have learnt so far..

- Large amounts of data to be processed.
- We have HDFS as a distributed store.
- We need to distribute the processing also.

PES UNIVERSITY

## What is Map Reduce ?

# Map Reduce Programming model and Architecture



## Why do we do Map-Reduce ?

- It’s the processing component of Apache Hadoop - a way to process extremely large data
- We are going to
  - Study Map-Reduce paradigm or the programming model for Map-Reduce
  - Study Hadoop architecture or how Map-Reduce works internally
    - Open Source implementation of Map-Reduce

# BIG DATA

## What is MapReduce ?

- Origin from Google, [OSDI'04] & built on principles of parallel and distributed computing
- Hadoop is the adoption of open-source implementation by Yahoo (now Apache project)
- MapReduce is the processing component of Hadoop with a programming model and processes massive amount of data with advantages of
  - *Parallelly* processing the data in a distributed computational environment and thus *increases processing performance*
  - *Data Locality – ability to process data where it, is by moving processing to data location rather than moving data which is more expensive*
- Its an execution framework for large-scale data processing
  - Which distributes the computing across distributed commodity servers and then pulls the results together
  - Programmer writes code as if writing for a single machine but the framework executes the process in distributed manner
  - Distributed implementation that hides all the messy details
    - Fault tolerance (thus provides HA)
    - I/O scheduling
    - parallelization

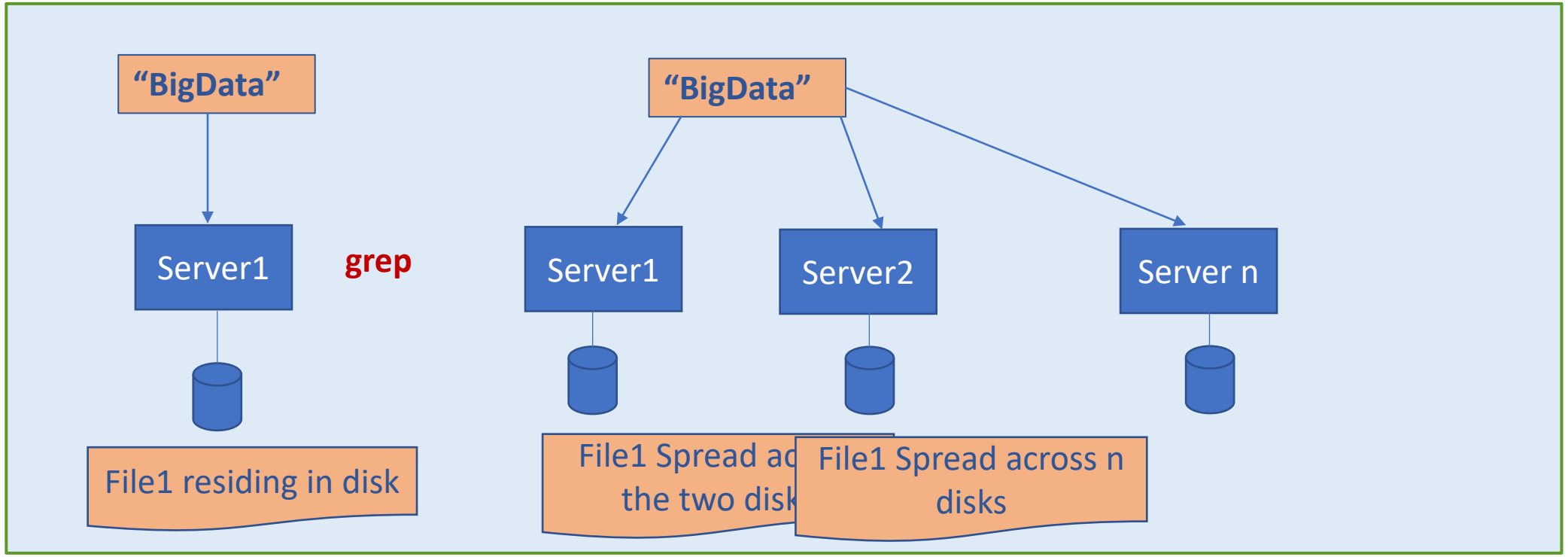
## Map Reduce - Motivation

- Lots of demands for very large scale data processing leading to the need for Scaling
- Scale out not up - Symmetric multi-processing machines (SMP) with large number of processor sockets (100s), large shared memory (GBs) are significantly more expensive as compared to Cluster of low end machines which are 4x more cost effective but lots of machines needed
- Build levels of abstraction which supports beneficial division of labor
  - A simple Programming model/Interface : A map + reduce
    (separating the abstraction of ***what*** from ***how*** of data intensive processing)
  - An application programmer develops using well defined interfaces and passes on the data **(what)** and hence is isolated from system-level details like locking, starvation, etc.
  - The framework (Map-Reduce) which is designed once and verified for correctness, takes care of the **How**. This includes the system-level details of writing distributed programs with details of threads, processes, machines Code that runs concurrently and needing to factor in Deadlocks, race conditions, etc.

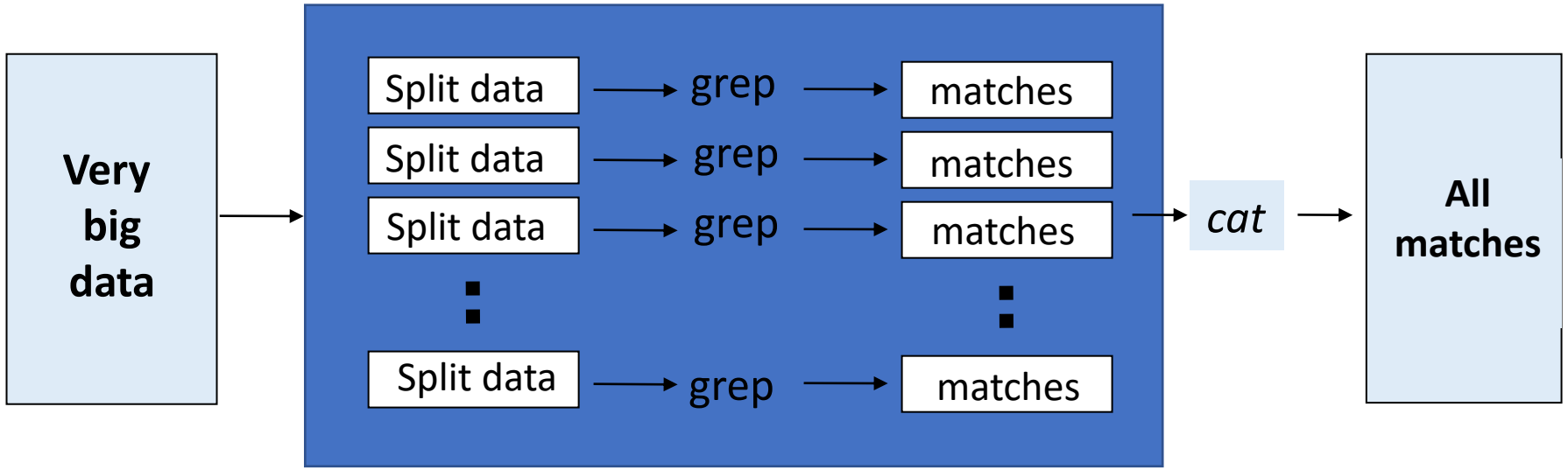
# Big Data: Illustrative Example leading to Map Reduce

## A MapReduce Example

- Consider a very large text file and you want to determine if a word exists in this file? E.g. “BigData” in File1
  - Say if the file is stored in HDFS across two machines.
  - How will you search to check if the word “BigData” is present in the file?

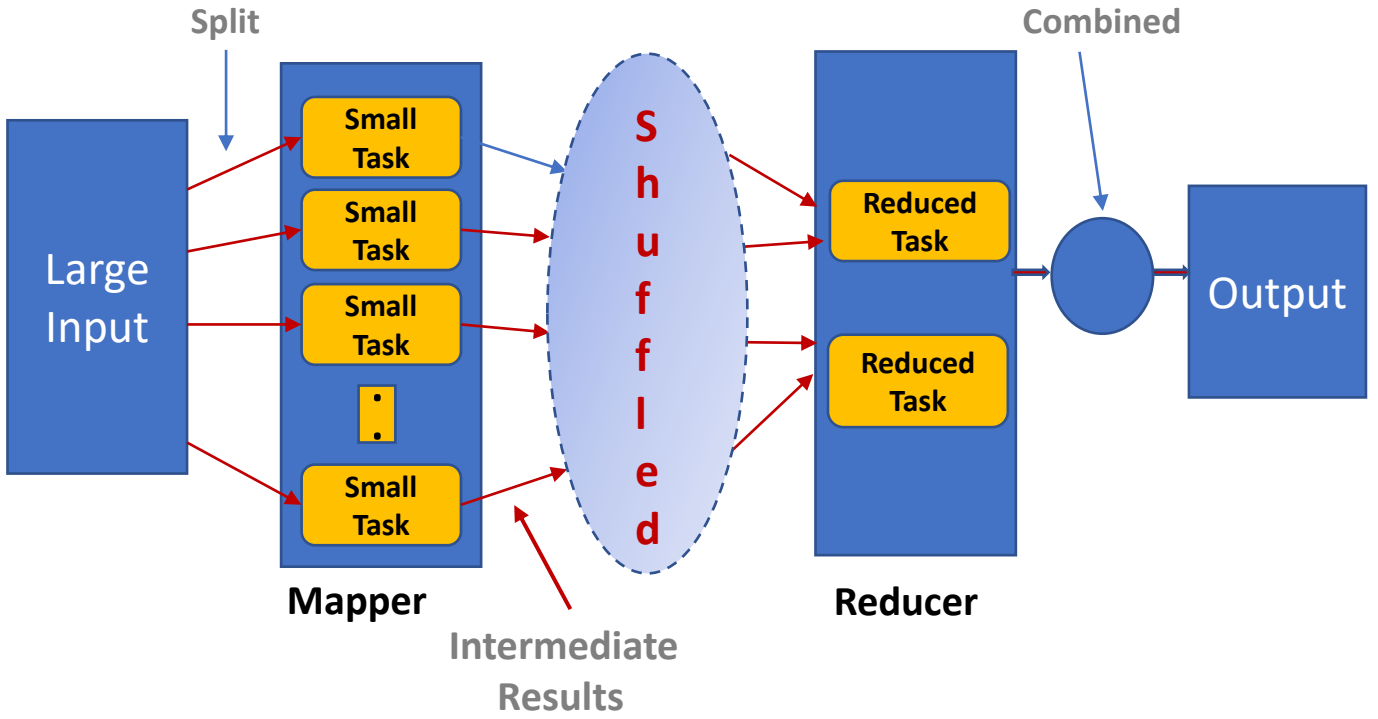


# Map Reduce: Distributed Grep-Solution



## Map Reduce - Insight

- Uses Divide and conquer – Partitions large problem into smaller subproblems
- Workers work on sub-problems in parallel (could be threads in a core or cores in multi-core processor, multiple processor in a machine, machines in a cluster) and produce intermediate results
- Intermediate results from workers are combined to form the final result
- Basic operations on the input
  - Map
  - Reduce

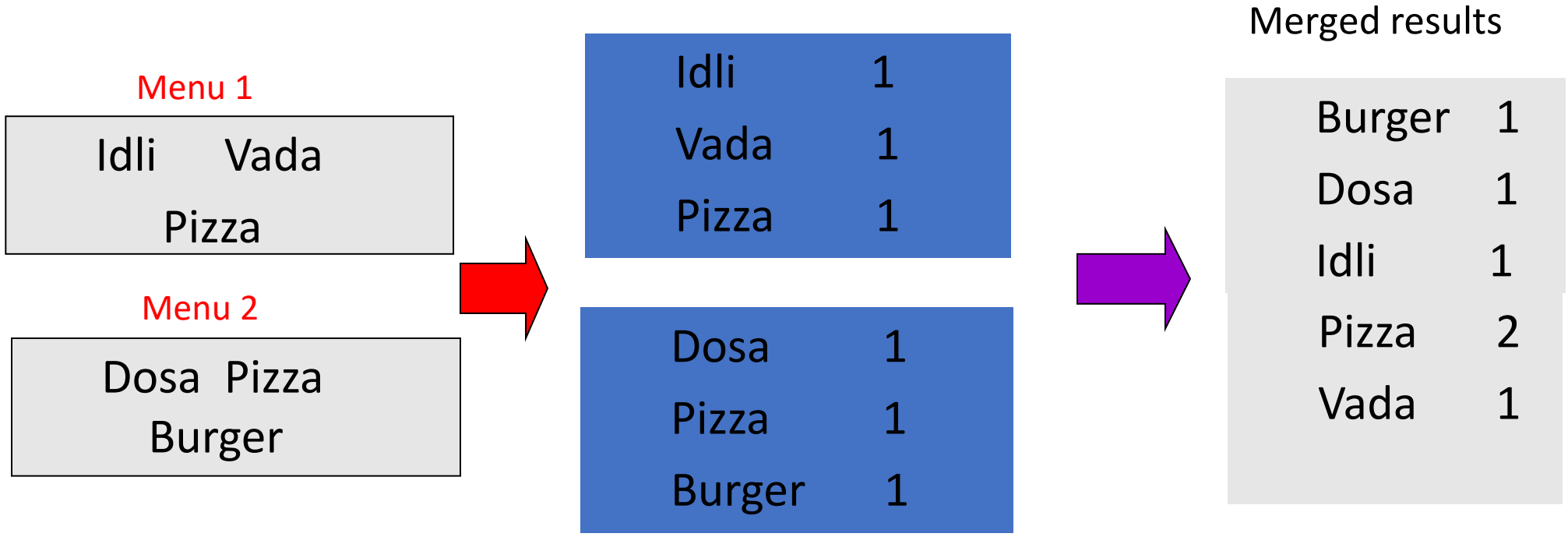



- To understand what Map/Reduce really are..

PES UNIVERSITY

# Lets consider a Distributed Word Count

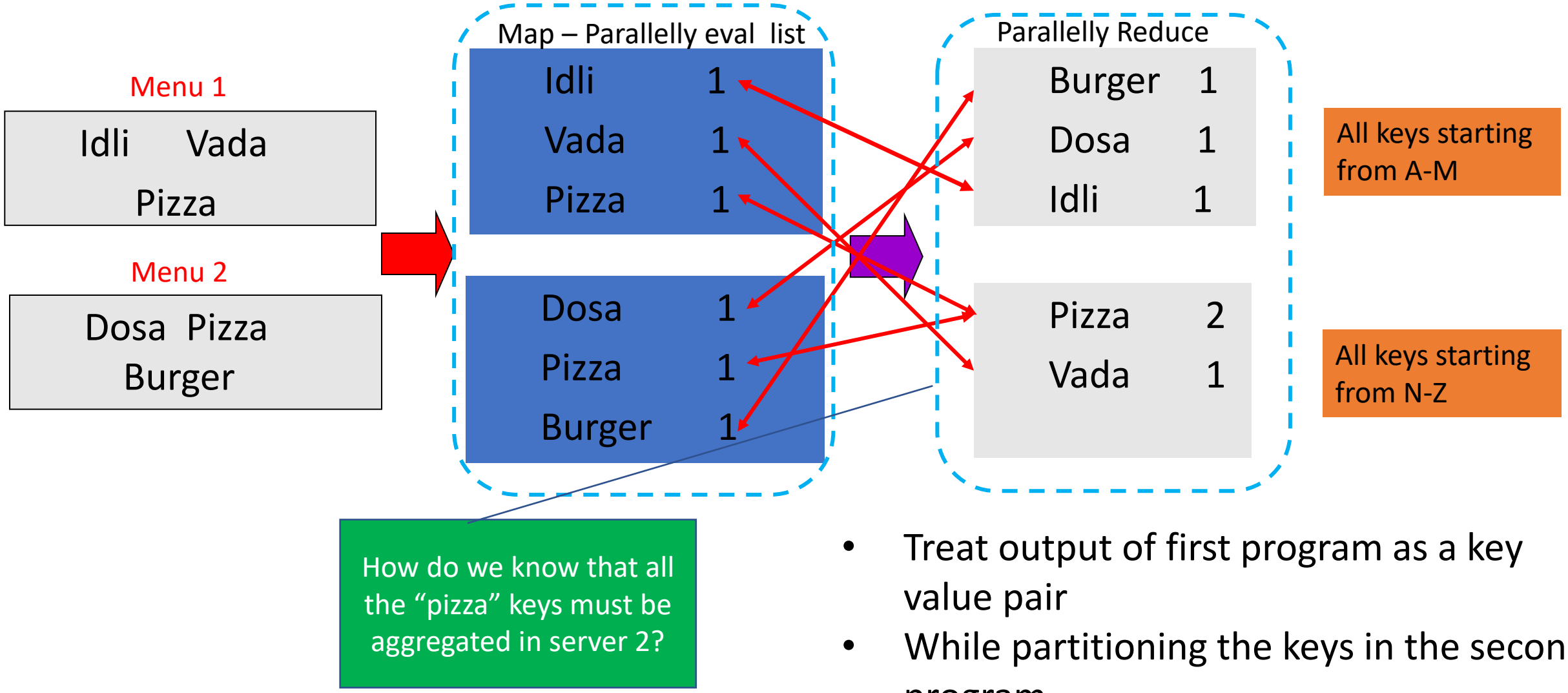
## Example: find the number of restaurants offering each item?




- Suppose we need parallelism of the merge program.
- Would the earlier approach work?
- What do we need to do?

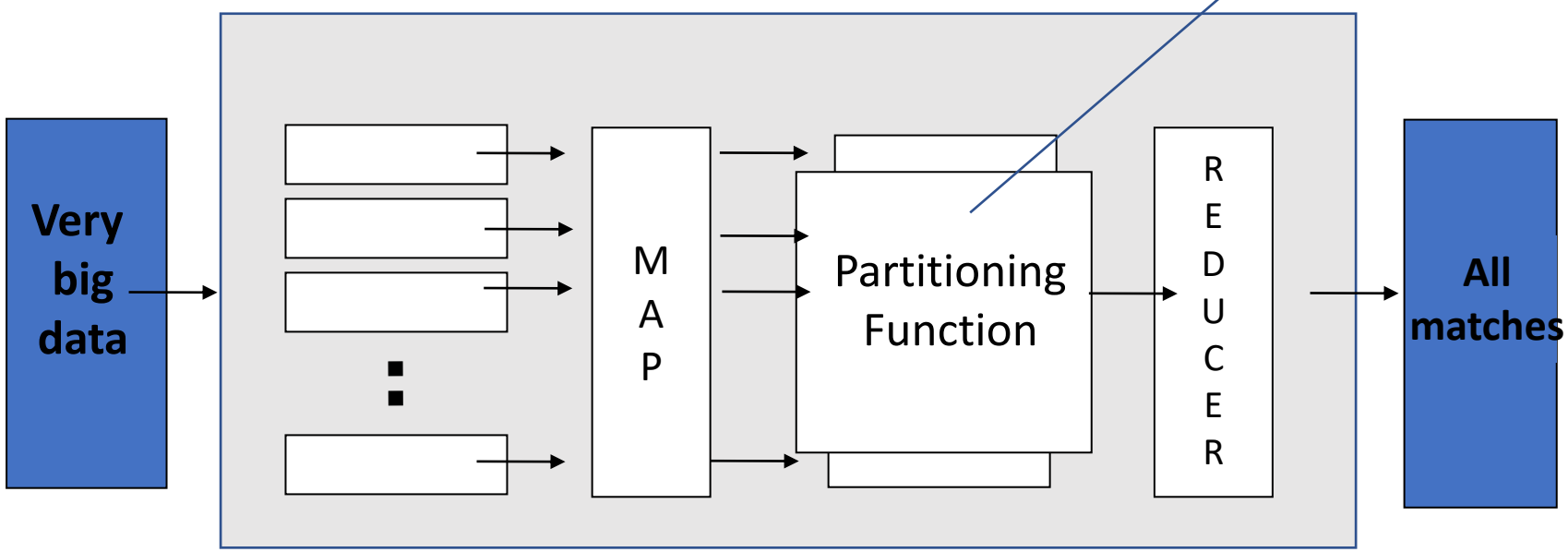
# BIG DATA

## Distributed Word Count




- Treat output of first program as a key value pair
- While partitioning the keys in the second program

# Map + Reduce




- Map:
  - Accepts *input* key/value pair
  - Emits *intermediate* key/value pair
- Reduce :
  - Accepts *intermediate* key/value* pair
  - Emits *output* key/value pair

# Map Reduce: A look at the code

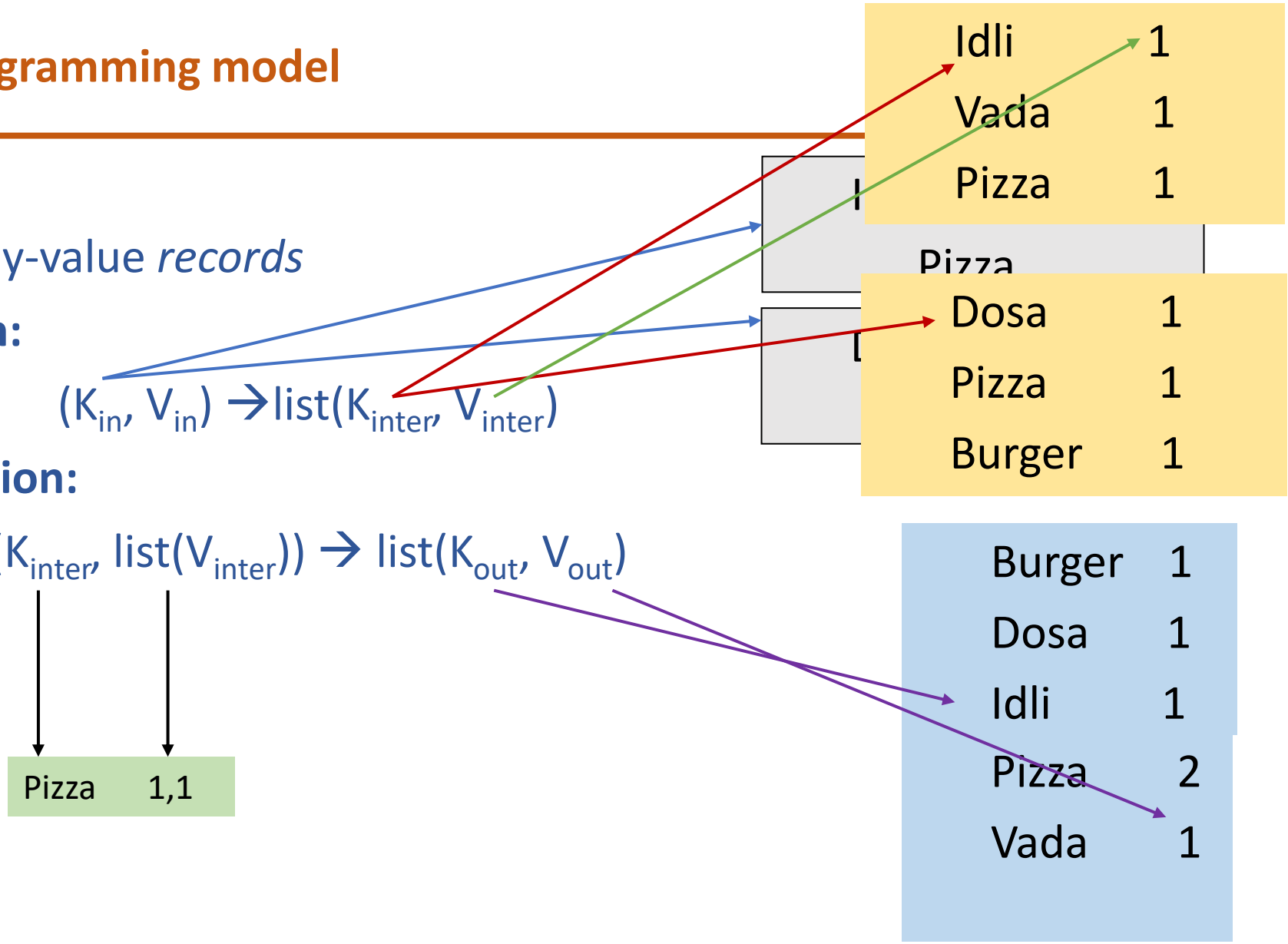
# Map Reduce Programming model

- **Data type:** key-value *records*
- **Map function:**

$(K_{in}, V_{in}) \rightarrow list(K_{inter}, V_{inter})$

- **Reduce function:**

$(K_{inter}, list(V_{inter})) \rightarrow list(K_{out}, V_{out})$



# Map Reduce- Mapper for Word Count

```
public static class TokenizerMapper
      extends Mapper<Object, Text, Text, IntWritable>{

   private final static IntWritable one = new IntWritable(1);
   private Text word = new Text();

   public void map(Object key, Text value, Context context
                   ) throws IOException, InterruptedException {
     StringTokenizer itr = new StringTokenizer(value.toString());
     while (itr.hasMoreTokens()) {
       word.set(itr.nextToken());
       context.write(word, one);
     }
   }
 }
```

(Key , value)

List (Key ,value)

$(K_{in}, V_{in}) \rightarrow list(K_{inter}, V_{inter})$

Hadoop data type

# Map Reduce -Reducer for Word Count

```
public static class IntSumReducer
      extends Reducer<Text,IntWritable,Text,IntWritable> {
   private IntWritable result = new IntWritable();

   public void reduce(Text key, Iterable<IntWritable> values,
                      Context context
                      ) throws IOException, InterruptedException {
     int sum = 0;
     for (IntWritable val : values) {
       sum += val.get();
     }
     result.set(sum);
     context.write(key, result);
   }
 }
```

Key ,List (value)

List (Key ,value)

$(K_{inter}, list(V_{inter})) \rightarrow list(K_{out}, V_{out})$

# Map Reduce - Main Program for Word Count

```
public static void main(String[] args) throws Exception {
 Configuration conf = new Configuration();
 String[] otherArgs = new GenericOptionsParser(conf, args).
                           getRemainingArgs();
 if (otherArgs.length < 2) {
   System.err.println("Usage: wordcount <in> [<in>...] <out>");
   System.exit(2);
 }
 Job job = new Job(conf, "word count");
 job.setJarByClass(WordCount.class);
 job.setMapperClass(TokenizerMapper.class);

 job.setReducerClass(IntSumReducer.class);
 job.setOutputKeyClass(Text.class);
 job.setOutputValueClass(IntWritable.class);
 for (int i = 0; i < otherArgs.length - 1; ++i) {
   FileInputFormat.addInputPath(job, new Path(otherArgs[i]));
 }
 FileOutputFormat.setOutputPath(job,
   new Path(otherArgs[otherArgs.length - 1]));
 System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

**Set Mapper and Reducer class**

- This does not state the machine where this needs to be run in
- This also does not state how the input needs to be distributed across mappers

# Map Reduce: Sample Exercise

PES UNIVERSITY

**Context : Search**

- We have a set of files each representing a web site
- Every file contains a line number and information as records.
- We are looking or searching for a pattern in the file
- Output should be all the line numbers which has the pattern

**Input:** file(lineNumber, line) records and pattern

**Output:** lines matching a given pattern

**What will be the mapper and reducer? What will be the keys?**

**Map:**

**Reduce:**

## Map Reduce , Search - Solution

**Input:** file (lineNumber, line) records and pattern

**Output:** lines matching a given pattern

**Map:**

```
if(line matches pattern):
        output(linenumber)
```

**Reduce:** identity function

–Alternative: no reducer (map-only job)

## Map Reduce , Functions in the Model

- **Map**
  - Process a key/value pair to generate intermediate key/value pairs
  - Sorts all key/value pairs before sending to reducer
- **Reduce**
  - Merge all intermediate values associated with the same key
  - Runs after all Map tasks are finished (why?)
- **Partition**
  - By default : **`hash(key) mod R`** （Well balanced)
  - There are cases where this can be more complex

# Map Reduce: Revision exercise

## Map Reduce, Sort

**Input:** (key, value) records

**Output:** same records, sorted by key

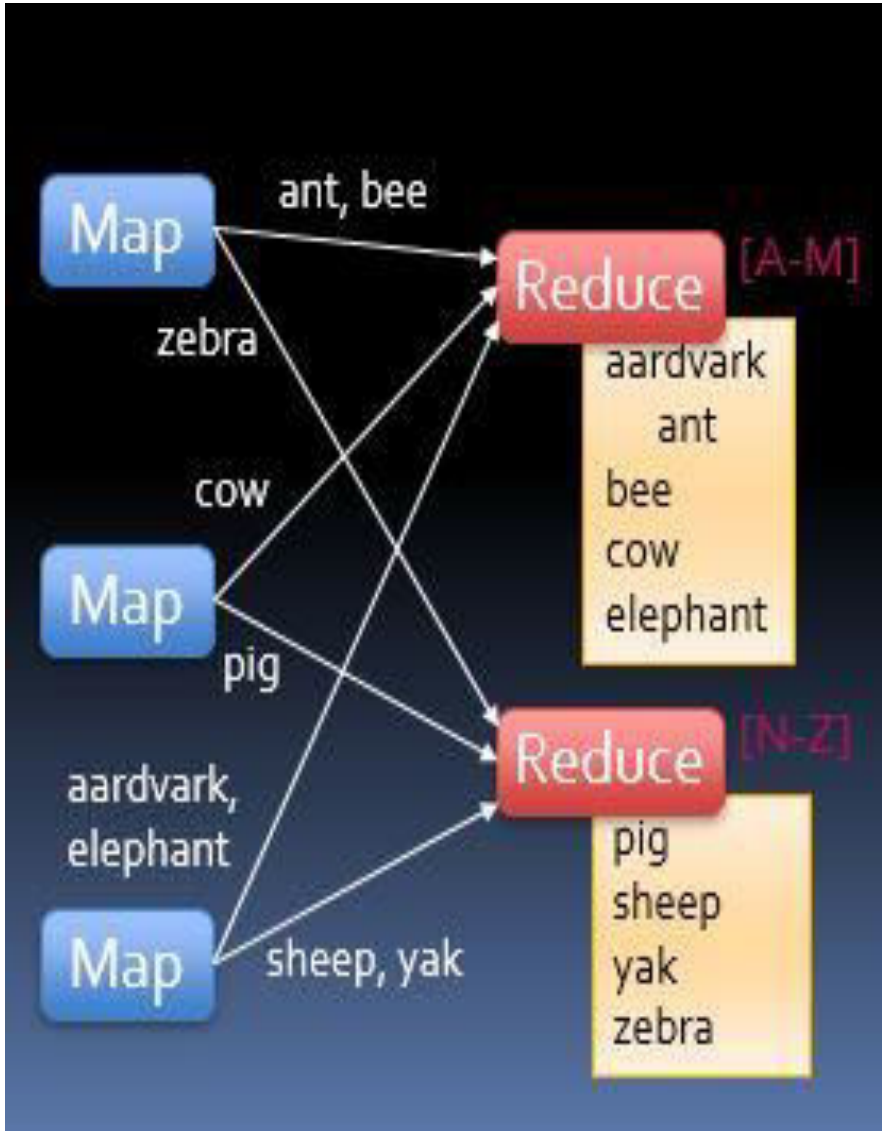
**What will be the mapper and reducer?**
**What will be the partition function?**

**Map:**

**Reduce:**

# Map Reduce, Sort - Solution

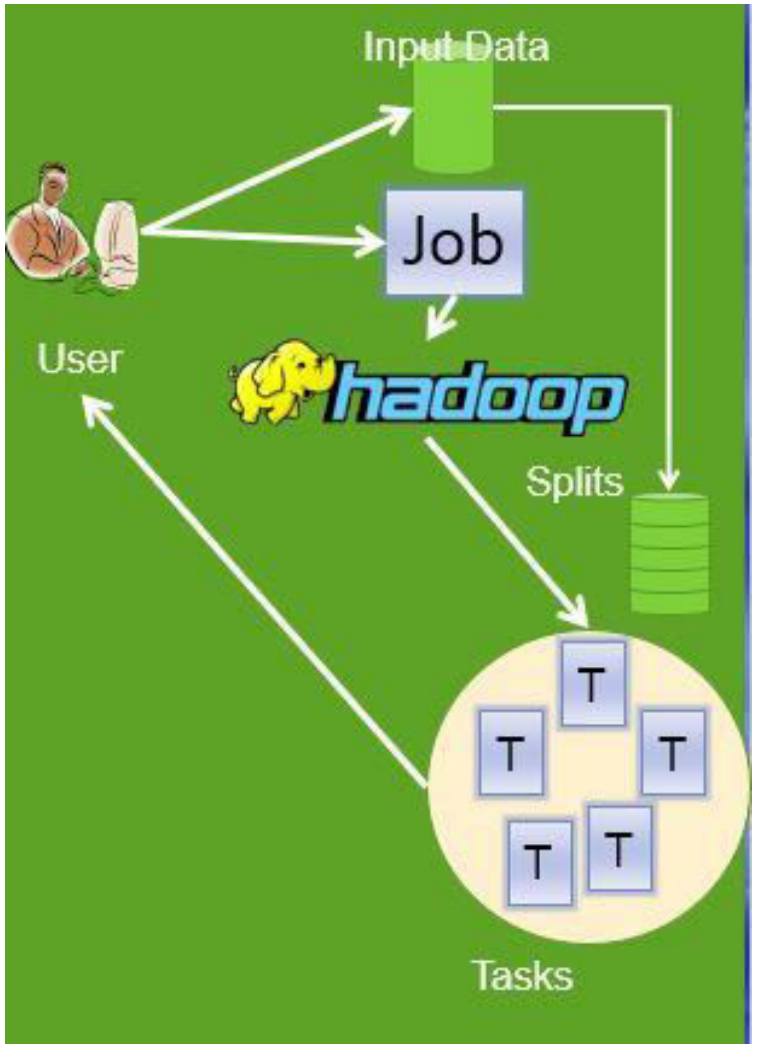
- **Input**: (key, value) records
- **Output**: same records, sorted by key
- **Map**: identity function
- **Reduce**: identity function
- **Trick**: Pick partitioning function *p* so that
  $k_1 < k_2 \Rightarrow p(k_1) < p(k_2)$
- Works because map sorts output keys.



# Map Reduce: Job Submission Flow

## Map Reduce , Hadoop Flow.

- User submits job
  - input data, MapReduce program, and configuration information
- How is parallelization achieved?
  - Divide input into smaller chunks – called **input splits**
- Hadoop divides jobs into tasks
  - Map tasks, reduce tasks
  - One map task per split
  - Tasks run in parallel



# MapReduce Flow for A SINGLE REDUCE Task

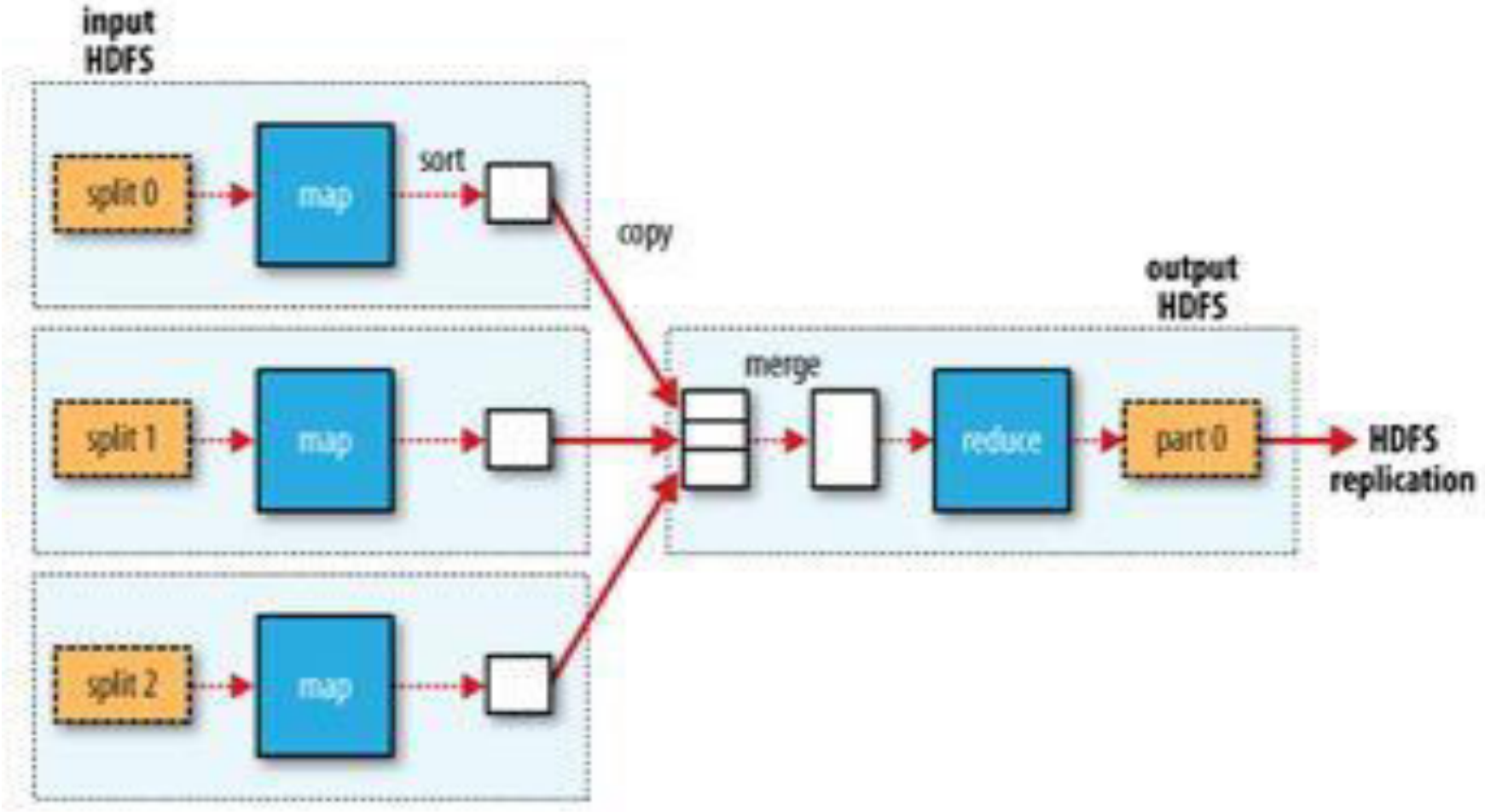



Figure: Map reduce data flow for single reducer task.

# Map Reduce:
# Exercise : Job Submission Flow with two Reducers

## MapReduce Flow

- How will it work when there are two reducers?
- Where will the outputs be?

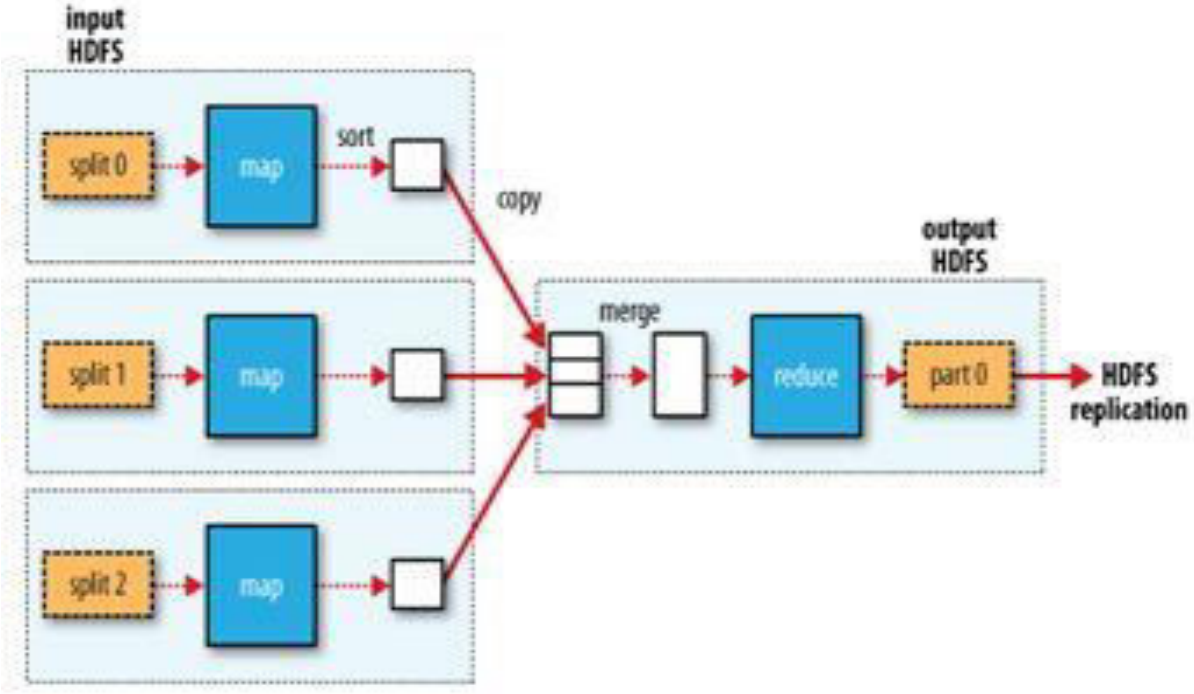
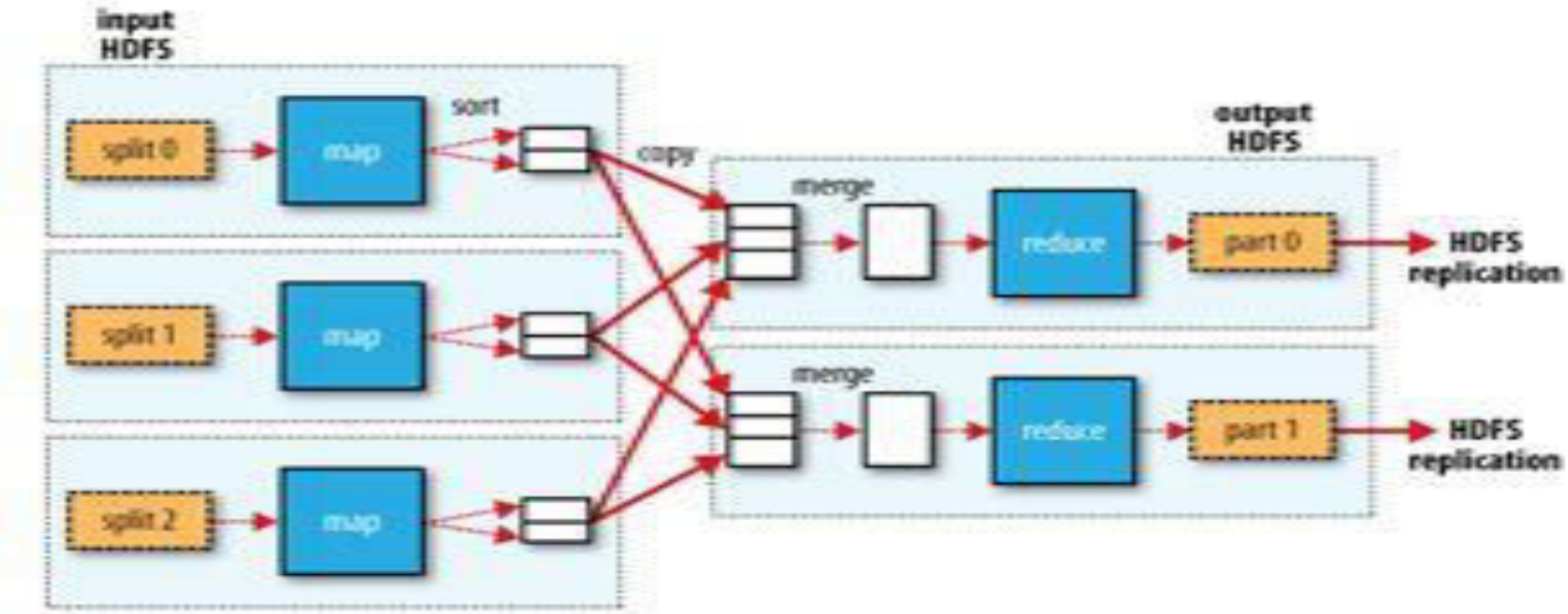



**Figure: Map reduce data flow for single reducer task.**

# MapReduce Flow for MULTIPLE REDUCE Tasks

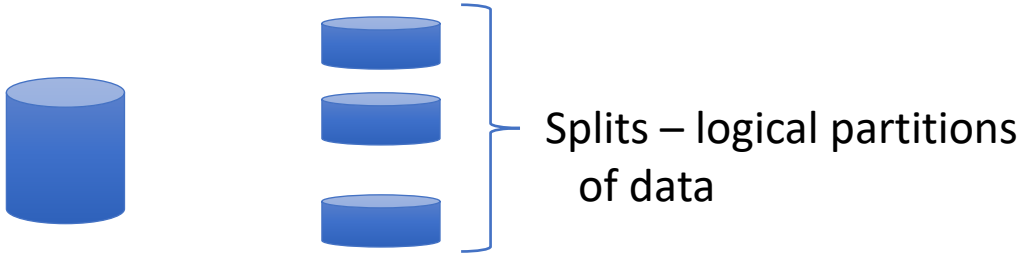



- Outputs are created on two different nodes and have to be merged.
- But available through HDFS on any node

# Map Reduce: Splits

## Map Reduce Split size considerations

- Split size proportional to parallelism
- Small split size
  - Advantages
    - Large #splits
    - Increased parallelism
    - Increased load balancing
  - Disadvantages
    - the overhead of managing the splits and of map task creation
    - begins to dominate the total job execution time.




Optimal split size == size of HDFS block (128MB) (default) on Hadoop v2

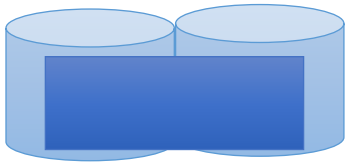
# Map Output

## Split == Block size

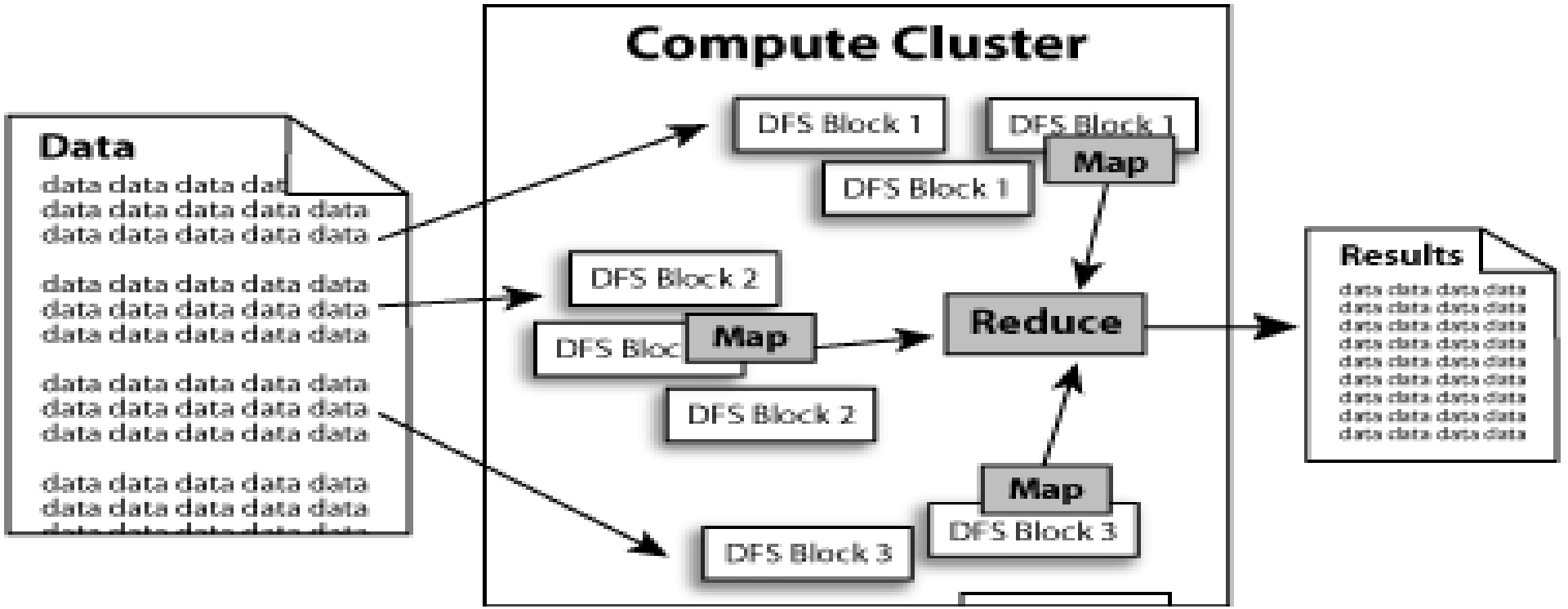
- All data required for Map
  - In the same node
  - No inter-node data transfer is required

## Split != block size

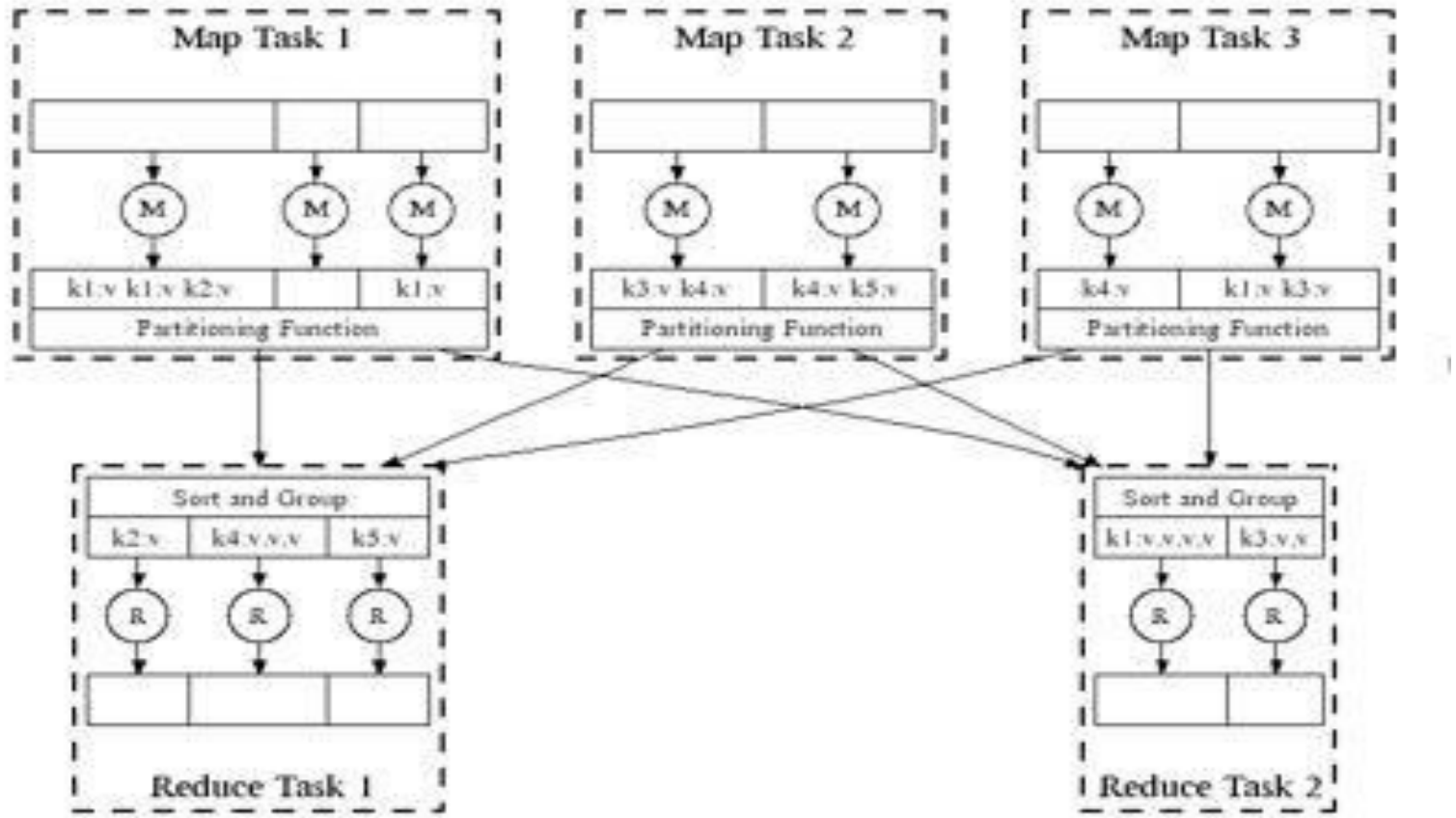
- Data transfer across multiple nodes
- Impacts performance



# Traditional: Move Data to Compute



# Big Data: Move Compute to Data




Parallel Execution

## Map Output

- **Where is Map output written to?**
  - Local disk and not HDFS
  - Why? Temporary output to be discarded after reduce.

- **Failure**
  - If the node running the map task fails
    - before the output has been consumed by reducer
  - Automatically rerun map task on another node

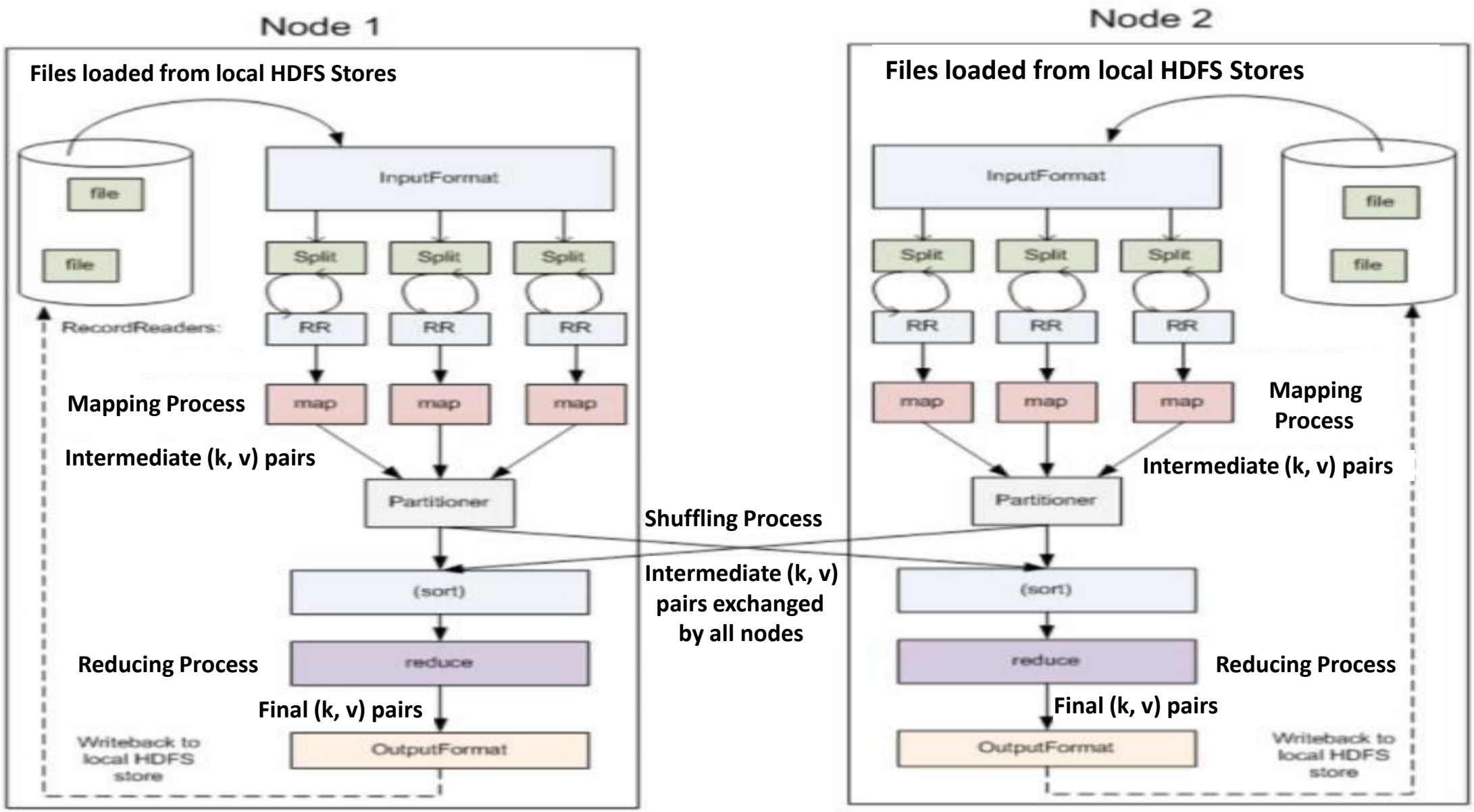
## Reduce Tasks

- Reduce tasks don't have the advantage of data locality
  - the input to a single reduce task is normally the output from all mappers.
- Sorted map outputs
  - have to be transferred across the network
  - Where to?
    - To the node where the reduce task is running
    - Merge data from different mappers
    - Then passed  to the user-defined reduce function.
- The output of the reduce is normally stored in HDFS for reliability.
- Where is the reduce output stored
  - 1 on the local node where the reduce happens
  - Other replicas on off-rack nodes.
  - Consumes network bandwidth

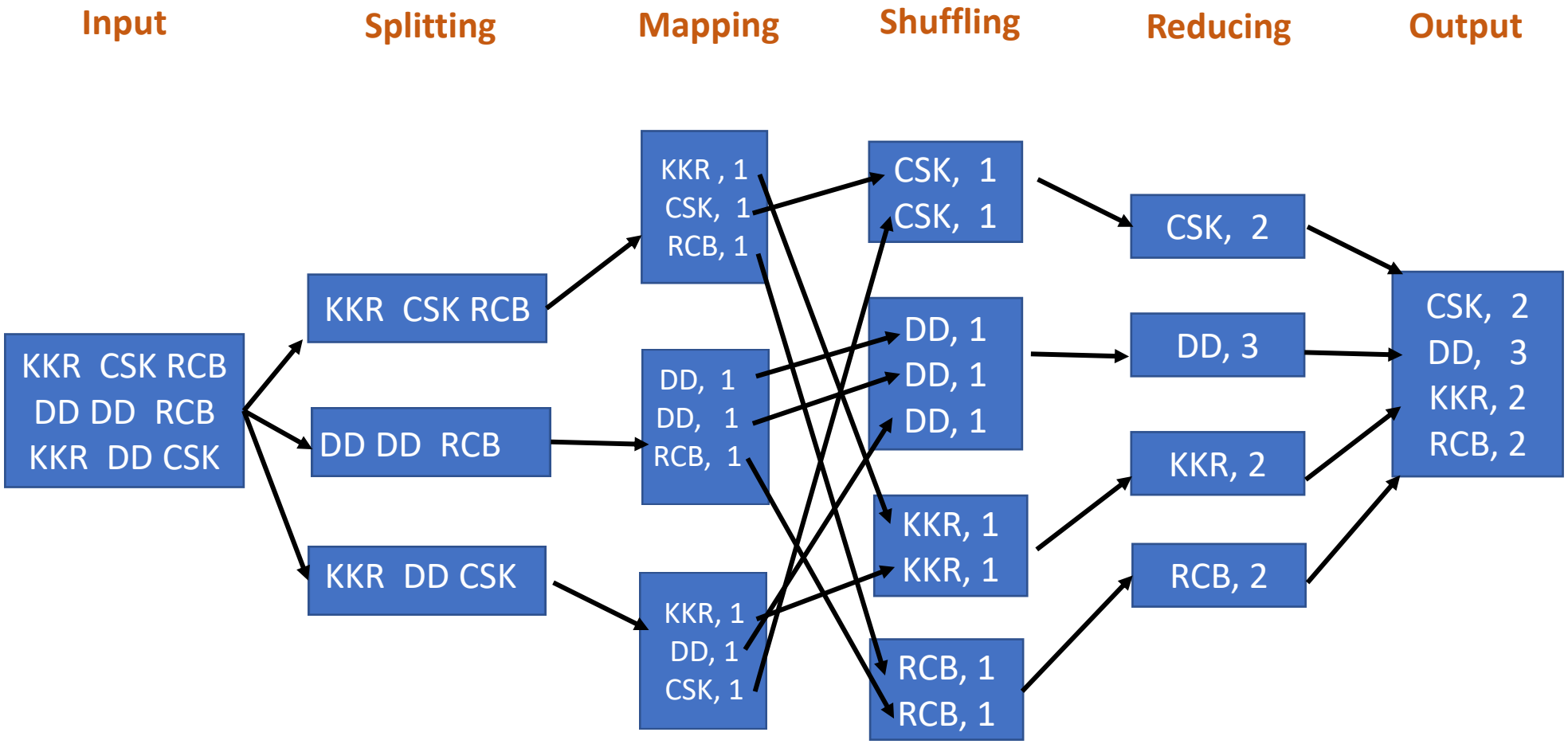
# Summary: MapReduce

1. Parallelize Map: easy! each map task is independent of the other!
   - All Map output records with same key assigned to same Reducer
2. Transfer data from Map to Reduce:
   - Shuffle data
   - All Map output records with same key assigned to same Reduce task
   - use partitioning function, e.g., hash(key)%number of reducers
3. Parallelize Reduce: easy! each reduce task is independent of the other!
4. Implement Storage for Map input, Map output, Reduce input, and Reduce output
   - Map input: from distributed file system (in our case HDFS)
   - Map output: to local disk (at Map node); uses local file system (in our case Linux FS)
   - Reduce input: from (multiple) remote disks; uses local file systems
   - Reduce output: to distributed file system

# Map Reduce: Working

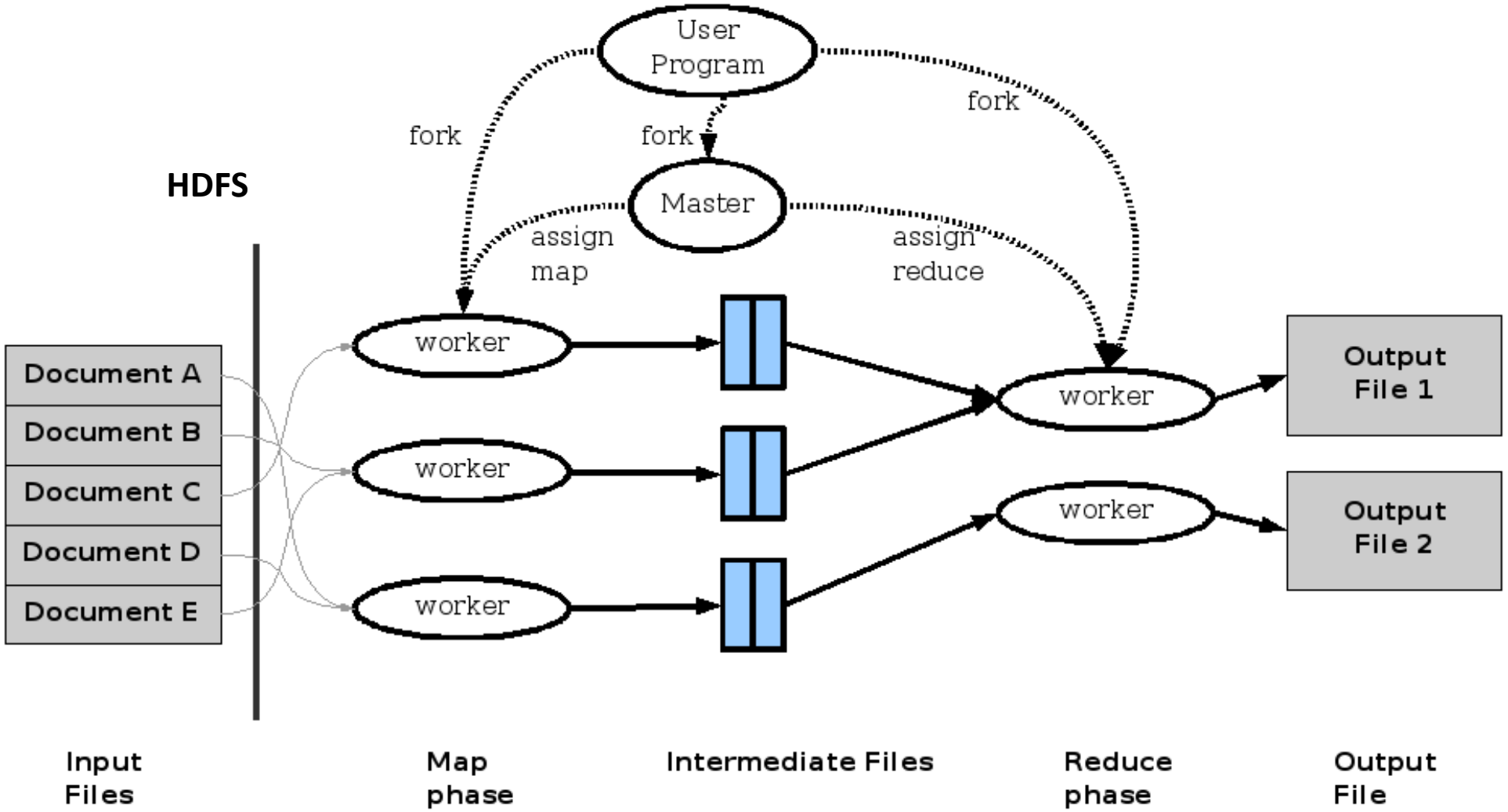

Node 1
Files loaded from local HDFS Stores
file
file
InputFormat
Split
Split
Split
RecordReaders:
RR
RR
RR
map
map
map
Mapping Process
Intermediate (k, v) pairs
Partitioner
Shuffling Process
Intermediate (k, v) pairs exchanged by all nodes
(sort)
reduce
Reducing Process
Final (k, v) pairs
OutputFormat
Writeback to local HDFS store
Node 2
Files loaded from local HDFS Stores
InputFormat
file
file
Split
Split
Split
RR
RR
RR
map
map
map
Mapping Process
Intermediate (k, v) pairs
Partitioner
(sort)
reduce
Reducing Process
Final (k, v) pairs
OutputFormat
Writeback to local HDFS store

# Shuffling - example

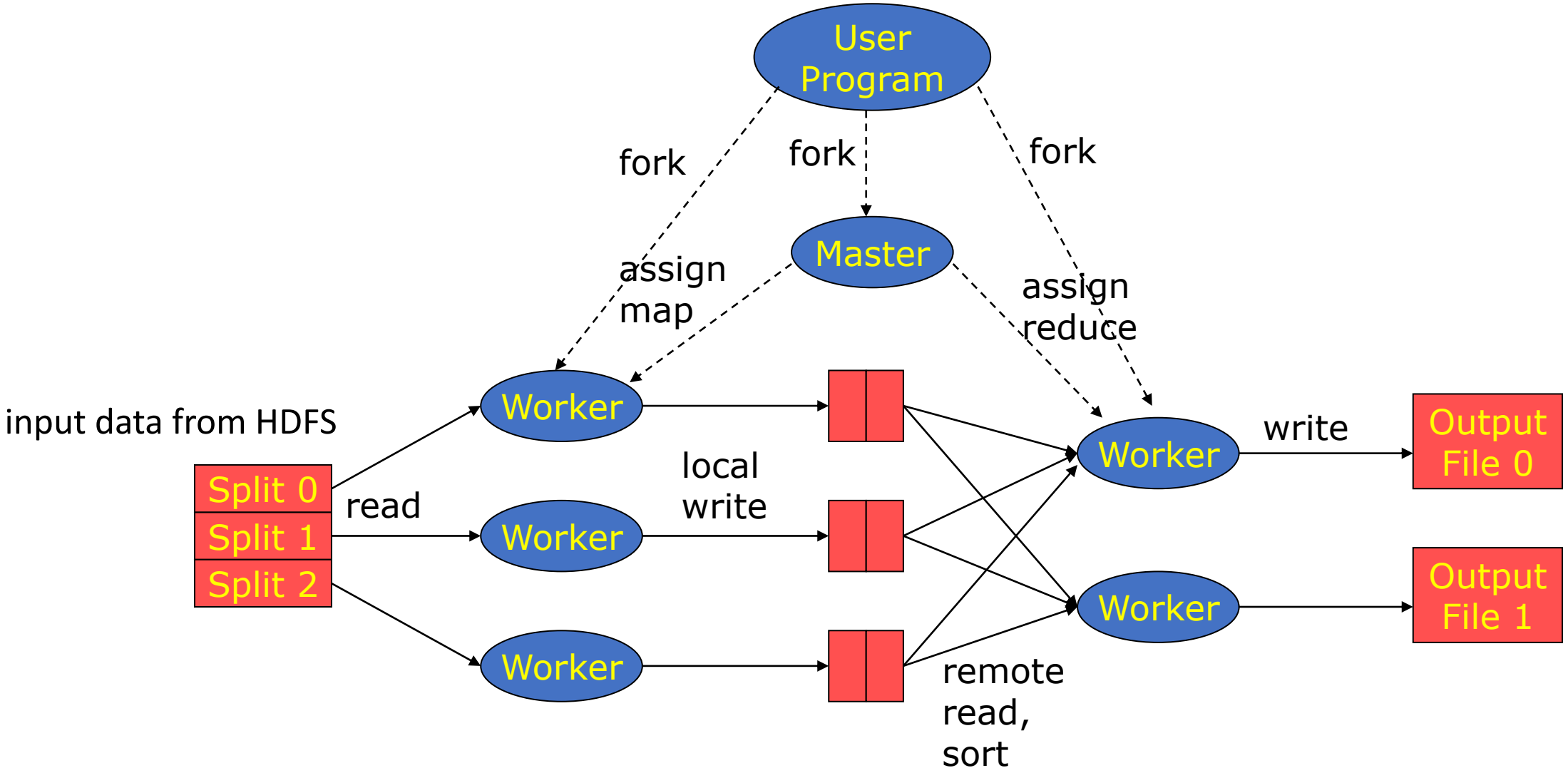



The overall Map-Reduce word count Process

# MapReduce: Execution overview



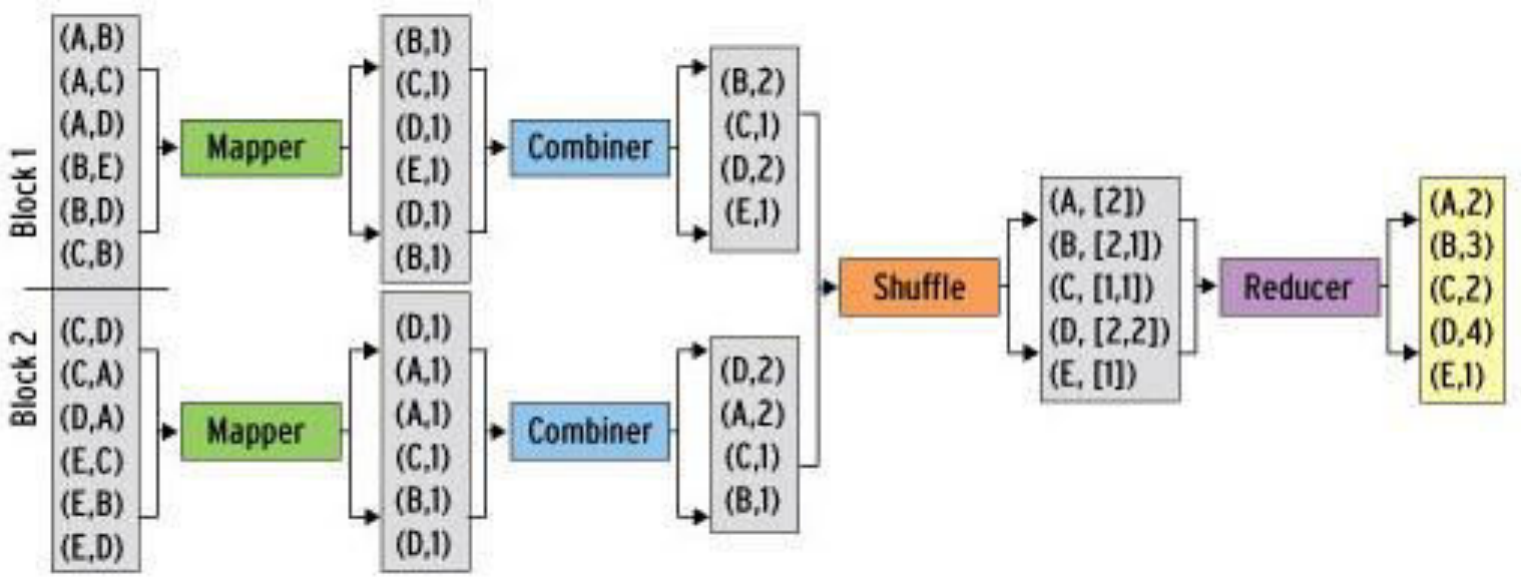
# Distributed Execution Overview




From Jeff Ullman's course slides

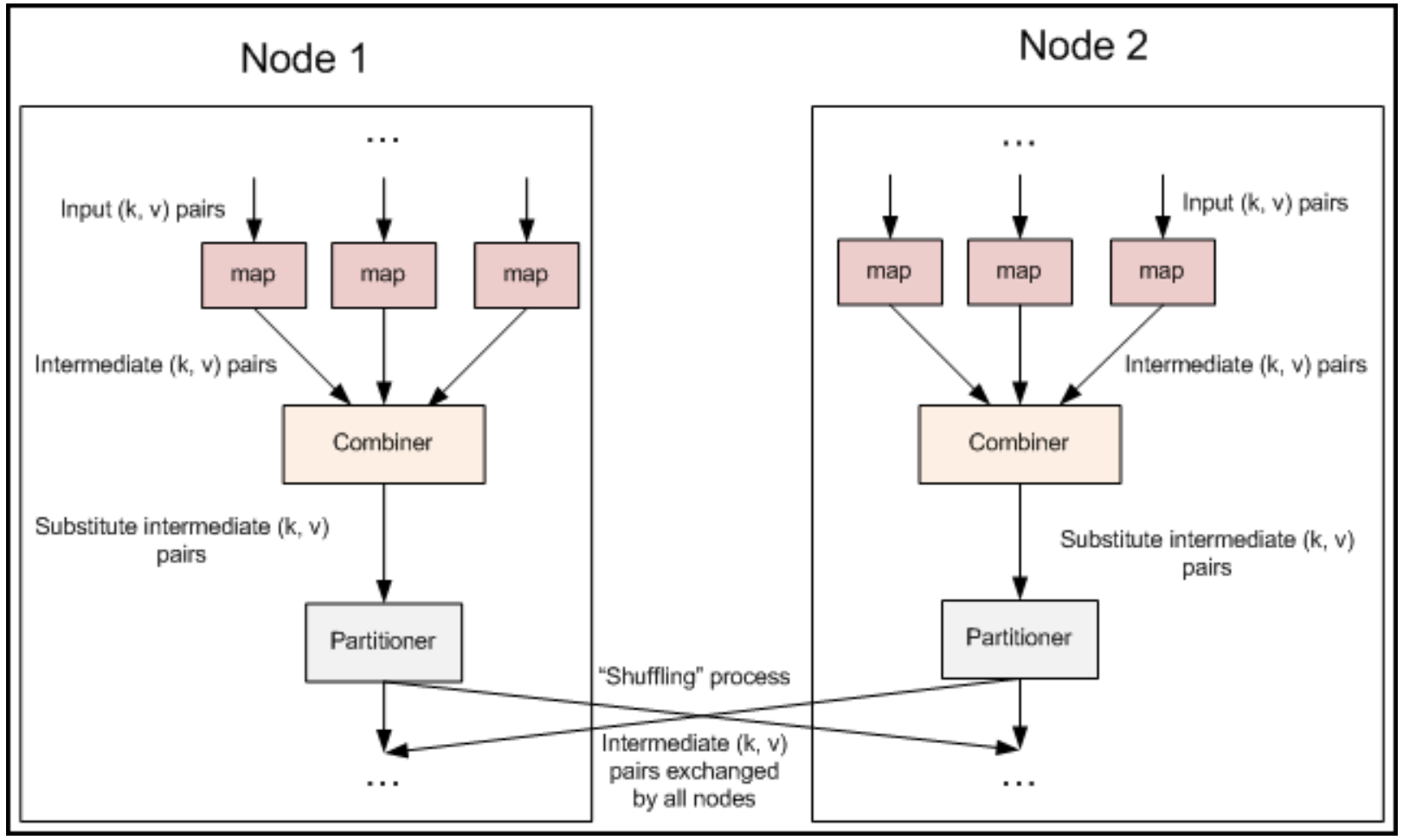
# Map Reduce: Combiners

# MapReduce – Combiners - motivation




- Combine multiple map outputs before doing a reduce
- Can write a combiner function in program
  - Combiner will be run before reduce
- Mini-reducer

# Combiner – when does it run?



# Map Reduce - Main Program for Word Count

```
public static void main(String[] args) throws Exception {
 Configuration conf = new Configuration();
 String[] otherArgs = new GenericOptionsParser(conf, args).
                           getRemainingArgs();
 if (otherArgs.length < 2) {
   System.err.println("Usage: wordcount <in> [<in>...] <out>");
   System.exit(2);
 }
 Job job = new Job(conf, "word count");
 job.setJarByClass(WordCount.class);
 job.setMapperClass(TokenizerMapper.class);
 job.setCombinerClass(IntSumReducer.class);
 job.setReducerClass(IntSumReducer.class);
 job.setOutputKeyClass(Text.class);
 job.setOutputValueClass(IntWritable.class);
 for (int i = 0; i < otherArgs.length - 1; ++i) {
   FileInputFormat.addInputPath(job, new Path(otherArgs[i]));
 }
 FileOutputFormat.setOutputPath(job,
   new Path(otherArgs[otherArgs.length - 1]));
 System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

**Combiner is set here**

# Review

## Review Question

- Suppose
  - We have a 2 GB file
  - Split size is 128 MB
  - We have 4 disks (one per node)
- How many splits are there?
- How many splits per disk?
- How many map tasks?
- How many map tasks per node?
- How many reduce tasks?

## Review Question - Solution

- Suppose
  - We have a 2 GB file
  - Split size is 128 MB
  - We have 4 disks (one per node)
- How many splits are there? - 16
- How many splits per disk? - 4
- How many map tasks? - 16
- How many map tasks per node? - 4
- How many reduce tasks? – User specified

## Sample Problem

Block 1

Far out in the uncharted backwaters of the unfashionable end of the Western Spiral arm of the Galaxy lies a small unregarded yellow sun. Orbiting this at a distance of roughly ninety-eight mi

Block 2

llion miles is an utterly insignificant little blue-green planet whose ape-descended life forms are so amazingly primitive that they still think digital watches are a pretty neat idea

- You run Word count using Hadoop on this data
- We know each block is an input split
- And each split is processed by a different mapper
- Do we get the right result?
- How will you solve this?

## Solution

Block 1

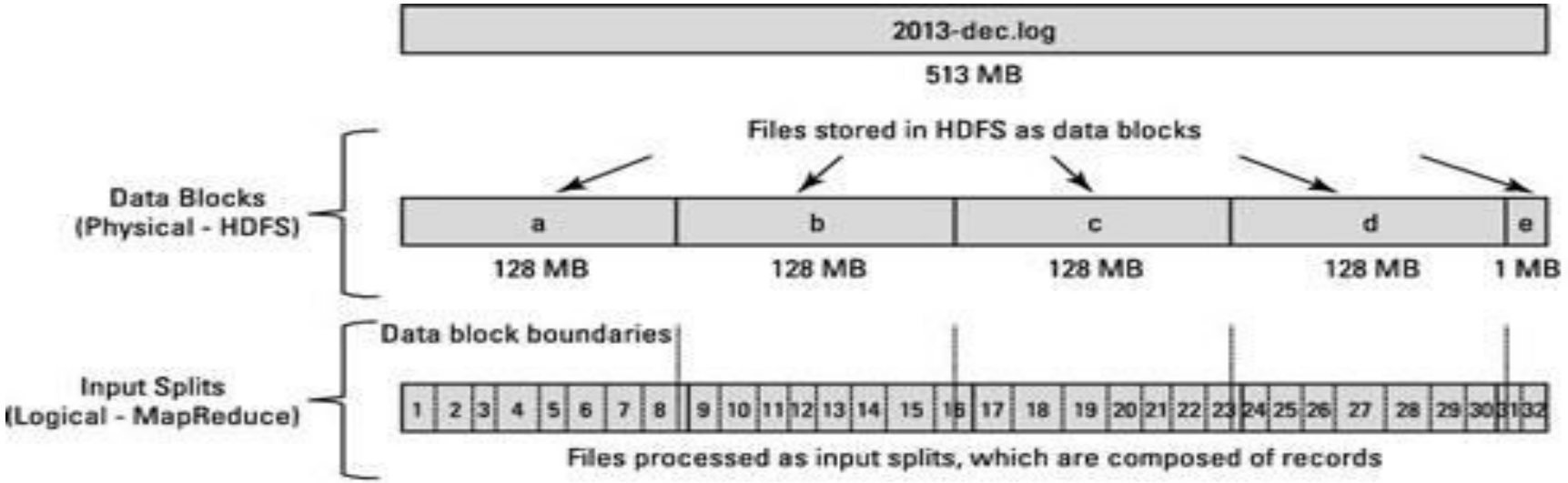
Far out in the uncharted backwaters of the unfashionable end of the Western Spiral arm of the Galaxy lies a small unregarded yellow sun. Orbiting this at a distance of roughly ninety-eight mi

Block 2

llion miles is an utterly insignificant little blue-green planet whose ape-descended life forms are so amazingly primitive that they still think digital watches are a pretty neat idea

- Each split further broken into records
- In this case the records would look like
  - Far
  - Out
  - in. …
- How to handle incomplete records?
  - Mapper 1 will process million
  - By looking for  EOR separator.

## Review Questions




map: (K1, V1) → list(K2, V2)
reduce: (K2, list(V2)) → list(K3, V3)

Record boundary → \n by default
Q: How can this be changed?

# Review Questions

## Review Questions

- Questions from T1, LOR 2.4

## Sample questions

- How many mappers and reducers will get started when trying to process a 230 MB file with Hadoop v2?
  - Ans: Block size = 128MB, so there will be two blocks. Assuming one block per split there will be 2 mappers
    - #reducers is configurable.
- Where is a combiner executed?
  - On the mapper.
- Write mappers and reducer pseudo code showing keys for counting #unique words in a file?
  - Similar to word count. Just that reducer does not have to write the count.

# Additional Notes, References and Videos

## Notes

- Chapter 2.4 from T1 – Rajkamal
- Tom whites book is an excellent reference for the programming component.

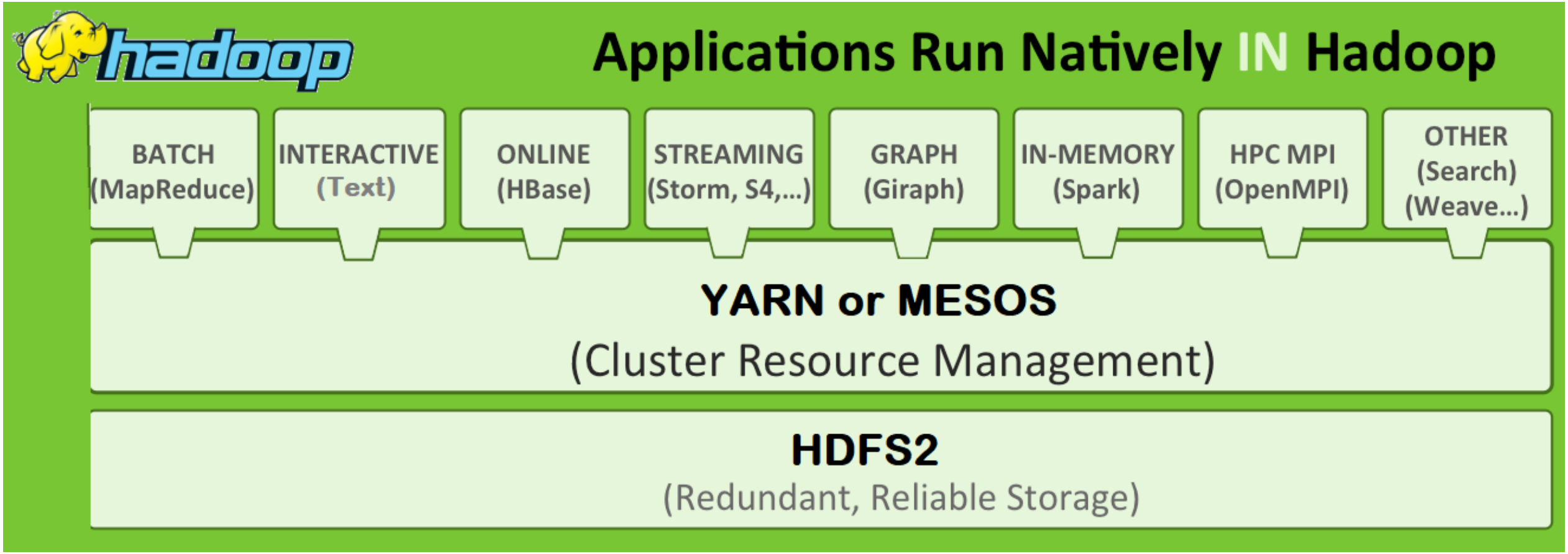

# THANK YOU

**Prafullata Kiran Auradkar**

Department of Computer Science and Engineering

**prafullatak@pes.edu**

## Notes


hadoop
Applications Run Natively IN Hadoop
BATCH (MapReduce)
INTERACTIVE (Text)
ONLINE (HBase)
STREAMING (Storm, S4,…)
GRAPH (Giraph)
IN-MEMORY (Spark)
HPC MPI (OpenMPI)
OTHER (Search) (Weave…)
YARN or MESOS
(Cluster Resource Management)
HDFS2
(Redundant, Reliable Storage)


# BIG DATA

## Hands On Session - 1

## MapReduce

**K V Subramaniam**
Computer Science and Engineering

# Overview

# BIG DATA

## MapReduce Handson Setup


- In order to execute the wordcount application, we need
- Ubuntu – virtualbox (example Ubuntu 20.04)
- Java version 8
- Hadoop v3.2.1
- Wordcount application

# BIG DATA

## Overview


- MapReduce is a programming model.
- Implementation for processing and generating large data sets with a parallel, distributed algorithm on a cluster.
- Mapper method performs filtering and sorting, and a Reduce method, performs an aggregate operation.
- We aim to solve a real world problem using MapReduce.

# BIG DATA

## Problem Statement


- Find the number of cars in every city which use gas as a mode of fuel using MapReduce.

# BIG DATA

## MapReduce Hands-on Exercise


### SPECIFICATIONS

1. Ubuntu 16.04+
2. Hadoop: 3.2
3. Python: 2.x/3.x
4. Java: 1.8
5. Dataset: You will be using the modified “Craigslist Used Cars Dataset” for this session. Please download the dataset from the below Google Drive link: https://drive.google.com/open?id=1GxEaY_aAlkMHfJN2Z1Cvt1O1yNtCp1gN

- **Columns of the Dataset :** The columns are indexed from [0-25] (Ex. Transmission is the 11th index)

- **Sample output :**

| City | Number of Cars that use Gas |
|---|---|
| Bangalore | 10 |
| Chennai | 12 |

- Actual output to be in a text file with each line of the answer having the pair <cityname> <number> .

## Pointers & Clues


- **Python Coders:**
  - start the code with the python shebang:
    *#!/usr/bin/python*
  - use sys module to read from stdin

- **Java Coders -** A sample word count program can be found here:
  - **https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html**


# Big Data

## Job Management and YARN

**Prafullata Kiran Auradkar**

Department of Computer Science and Engineering

**prafullatak@pes.edu**

**Acknowledgements:**

Significant information in the slide deck presented through the Unit 1 of the course have been created by **Dr. K V Subramaniam** and would like to acknowledge and thank him for the same. There have been some information which I might have leveraged from **Dr. H L Phalachandra's** slide contents too. I may have supplemented the same with contents from books and other sources from Internet and would like to sincerely thank, acknowledge and reiterate that the credit/rights for the same remain with the original authors/publishers only. These are intended for classroom presentation only.

## Overview

PES UNIVERSITY

### What we have learnt so far..

- Data processing distributed over a cluster – Map Reduce
- Job Submission Flow
- How does job management actually happen?
- How is failure management addressed?

… handled by **YARN**

PES UNIVERSITY

## Lecture Overview


- Need for YARN - history
- YARN Architecture
- Job submission lifecycle – YARN
- Scheduling
- Failure Handling
- Benefits of YARN

# Big Data: The need for YARN

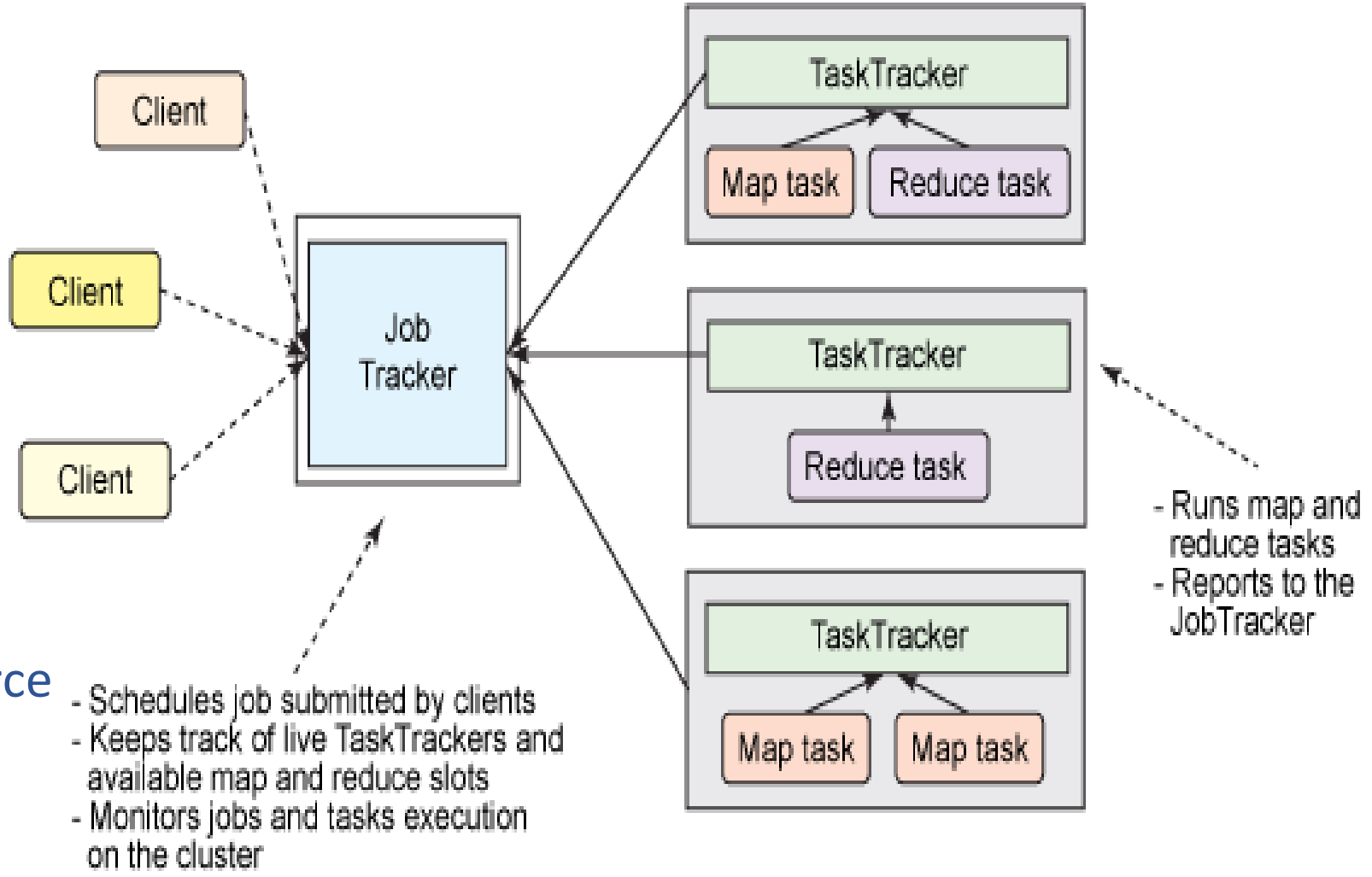
## Motivation

- Recall
  - Job – the entire map reduce application
  - Task – Individual mappers/reducers

- How do we
  - Allocate resources – determine which nodes will run the jobs
  - Monitor the tasks – start new tasks or restart failed/slow tasks
  - Monitor the overall state of the job?

## Hadoop 1.0 Job Management

- Job Tracker
  - Manage Cluster resources
  - Job scheduling
- Task Tracker
  - One per task
  - Manage the task
- Fault Tolerance, Cluster resource management and scheduling handled by JobTracker



# Hadoop 1.0 Issues

## Limits scalability

- Job tracker runs on a single machine and is responsible for cluster management, scheduling and monitoring

## Availability

- JobTracker is the single point of availability/failure

## Resource utilization problems

- Predefined #map/reduce slots. Utilization issues because map slots may be full but reduce slots are free.

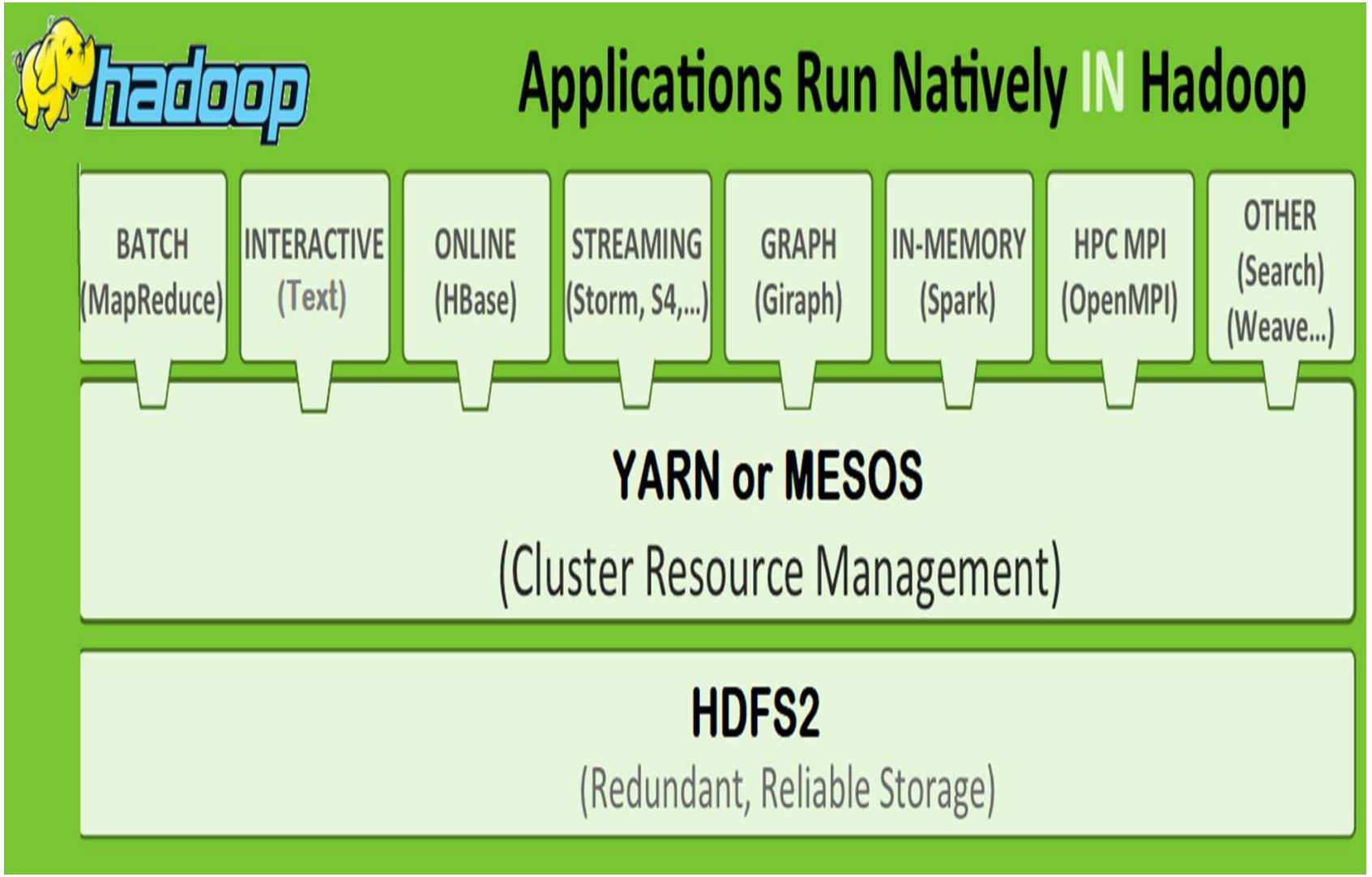
## Limitation in running MR applications

- Tightly integrated with Hadoop. Only MR apps can run. Can’t coexist with other applications.

# Big Data: YARN Architecture

## Map Reduce - Motivation

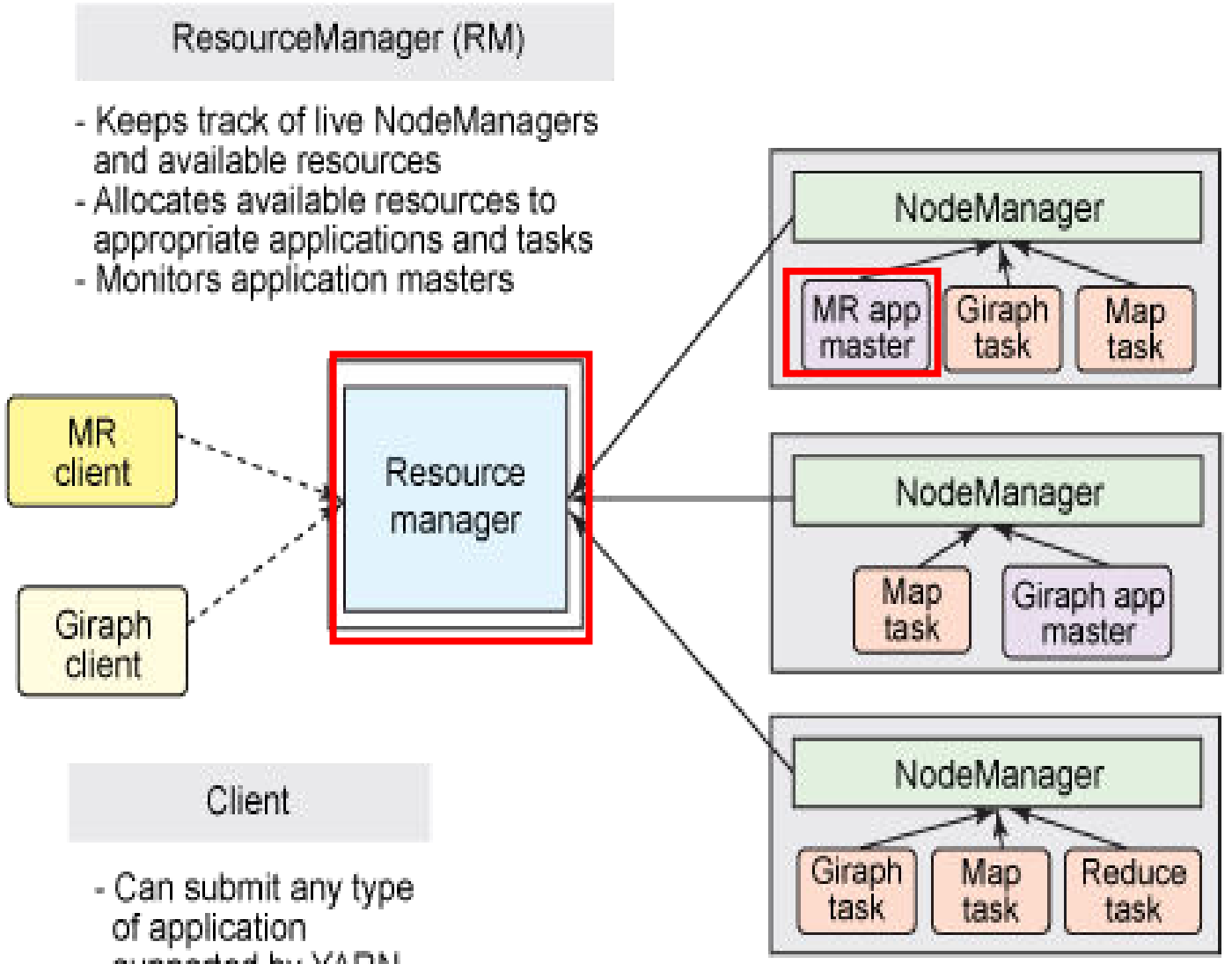
- Issues in managing clusters > 4000 nodes
- 2010 – MapReduce v2 with YARN
  - Yet Another Resource Negotiator
  - YARN Application Resource Negotiator!!



# BIG DATA

## YARN Architecture

- Split responsibility of Job Tracker
- Resource Manager – manage cluster wide resources
- Application Master – manage lifecycle of application

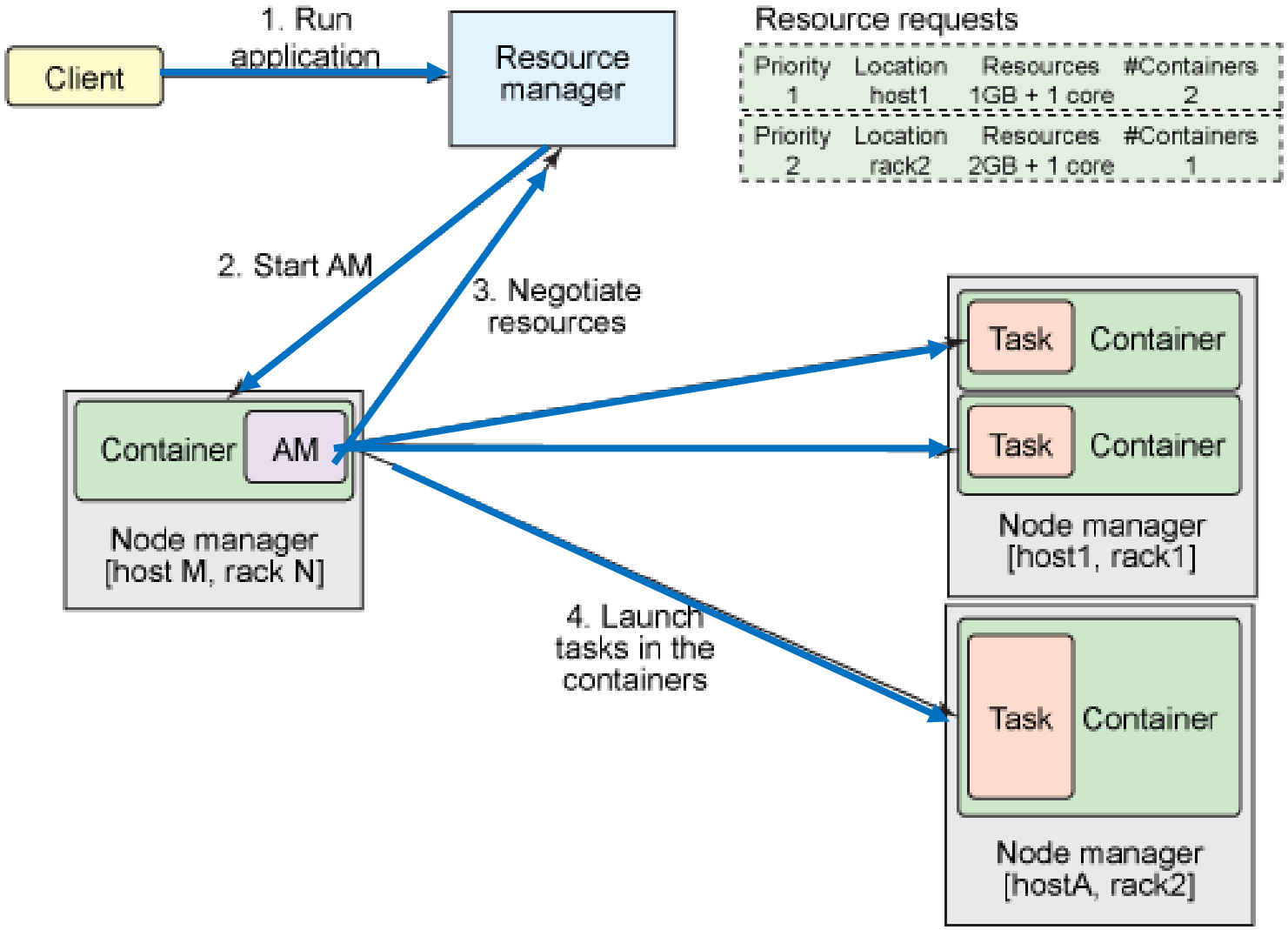


## YARN Components

| | |
|---|---|
| Resource Manager | • Arbitrates resources amongst all applications of the system |
| Node Manager | • Per machine slave<br>• Responsible for launching application containers<br>• Monitors resource usage |
| Application Master | • Negotiate appropriate resource containers from the scheduler<br>• Track and monitor the progress of the containers |
| Container | • Unit of allocation incorporating resources such as memory, CPU, disk |

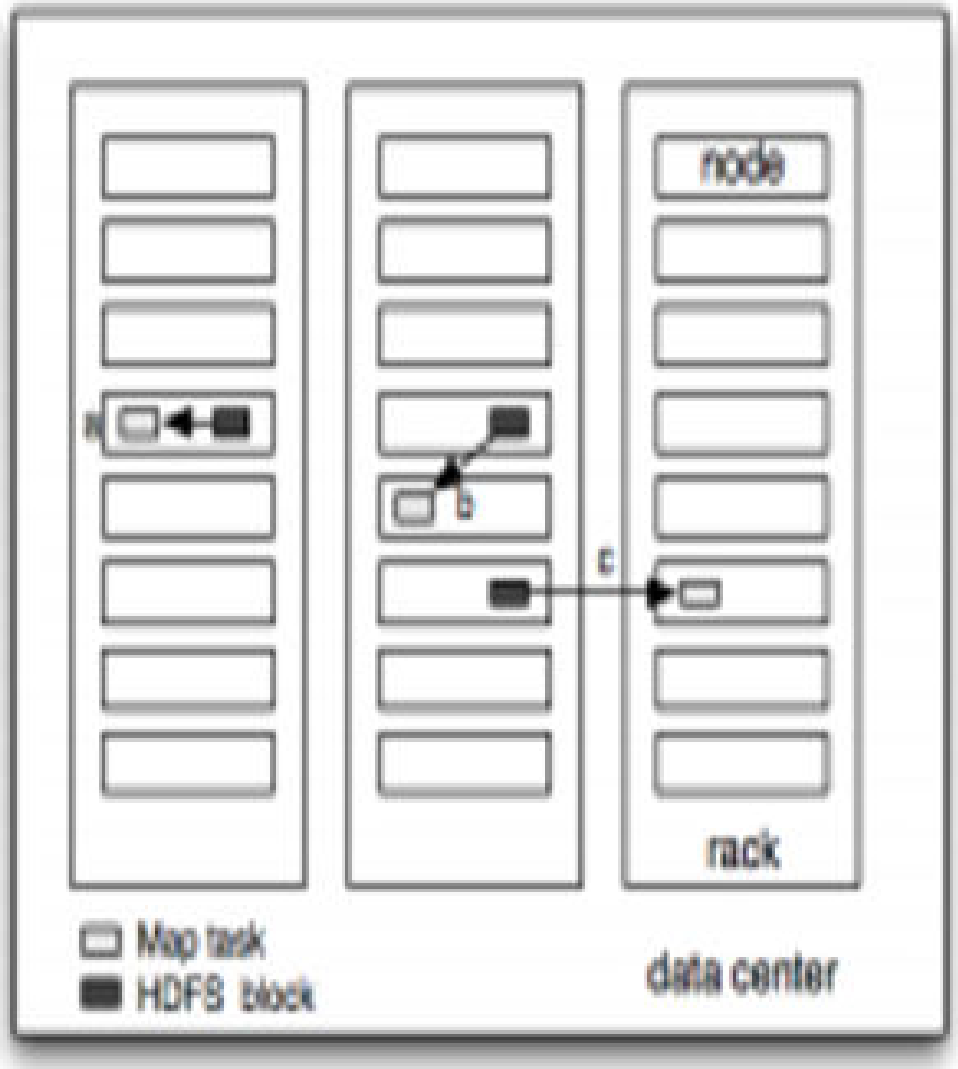
# Big Data: Job Submission - YARN

# YARN Working



## Data Locality in Map Reduce

- Attempts to run the map task on a node where the input data resides in HDFS.
  - *data locality optimization* - it doesn't use valuable cluster bandwidth.
- What happens when all nodes hosting the block replicas are busy?
  - look for a free map slot on a node in the same rack as one of the blocks.
- Very occasionally even this is not possible, so an off-rack node is used, which results in an inter-rack network transfer.



PES UNIVERSITY

# Scheduling in YARN

## Schedulers in Hadoop

- Early Hadoop versions, simplistic FIFO scheduler
  - In order of submission
  - each job would use the whole cluster
  - Resulting with jobs waiting for their turn.
- How to share resources fairly?
- Balance between
  - Production jobs
  - Ad-hoc jobs

## Fair Scheduler

- Aims to give every user a fair share of the cluster capacity over time.
- Jobs are placed in pools,
  - Default  each user gets their own pool.
- If a single job is running, it gets all of the cluster.
- As more jobs are submitted,
  - free task slots are given to the jobs in such a way as to give each user a fair share of the cluster.
- Short job – completes in reasonable time
- Long job – can continue making progress.

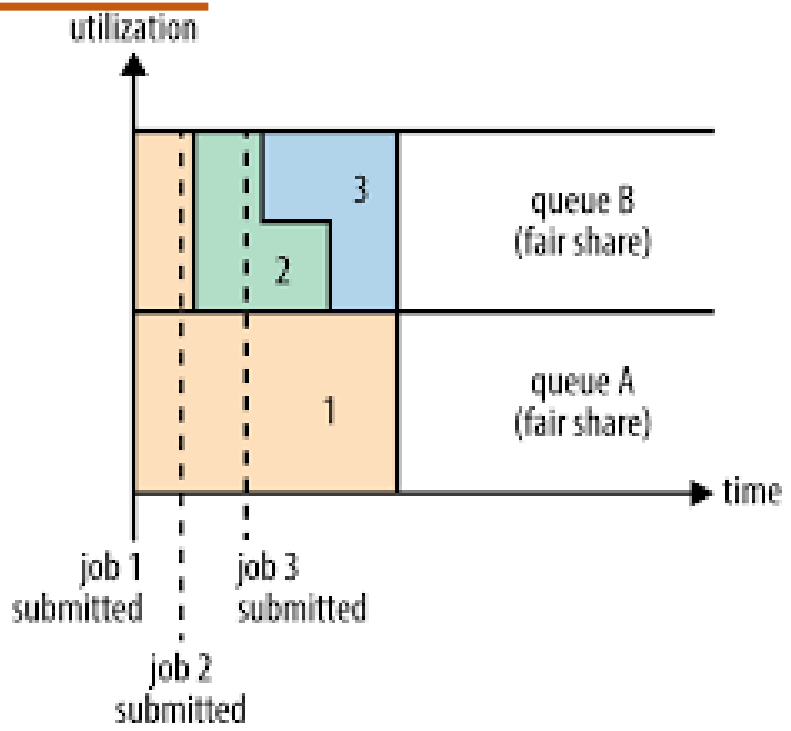



Image courtesy: Tom White, “Hadoop the definitive guide”

## Fair Scheduler

- Consider a user who submits more jobs
  - Scheduler ensures  that user does not hog the cluster
- Custom pools
  - Guaranteed minimum capacities with map/reduce slots
  - It is also possible to define custom pools with guaranteed minimum capacities defined in terms of the number of map
- The Fair Scheduler supports preemption
  - If pool not received its fair share over certain time
  - scheduler will kill tasks in pools running over capacity

## Capacity Scheduler

- Different approach
- number of queues (like the Fair Scheduler's pools),
  - Has an allocated capacity
  - Can be hierarchical
  - Within each queue  scheduled using FIFO (with priorities)
- Cannot use free or spare capacity even if it exists
- Like breaking up cluster into smaller clusters

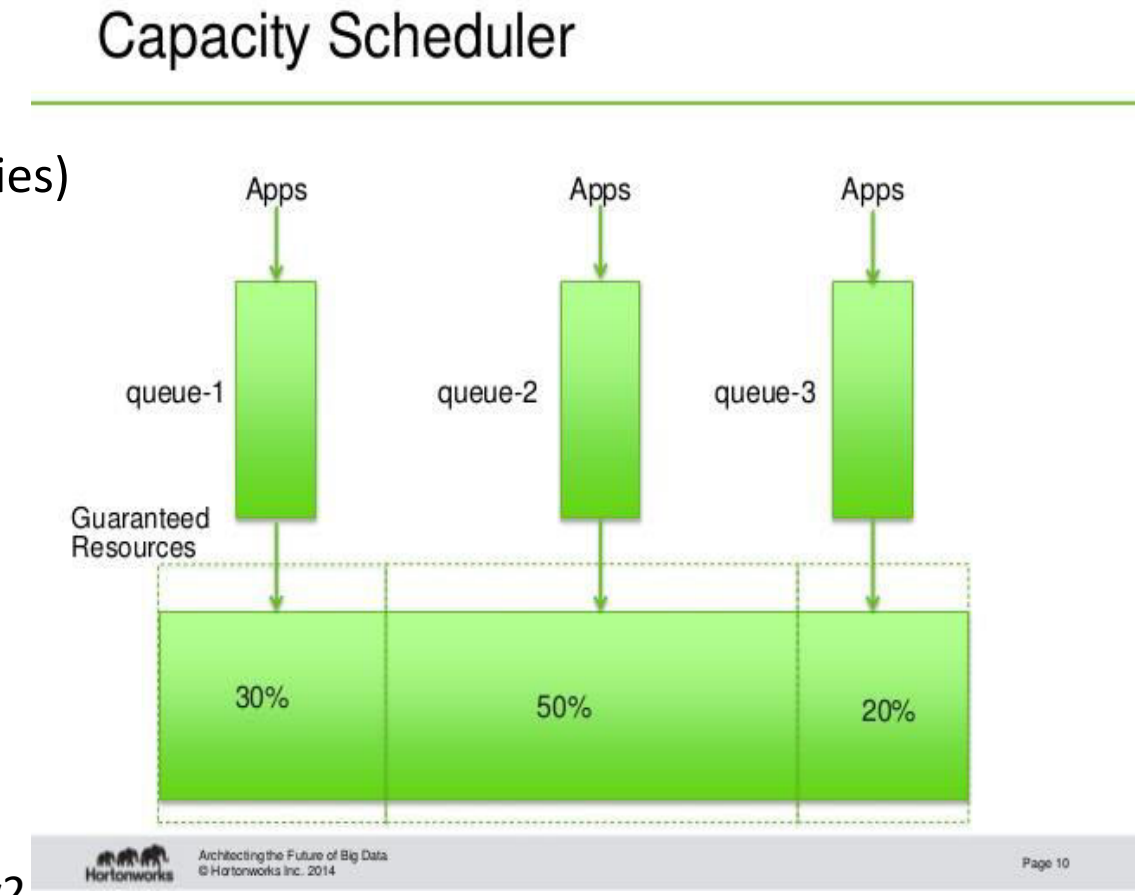



Image: https://www.slideshare.net/Hadoop_Summit/w-525hall1shenv2

# Handling Failures

## What can fail?

- Task
- Application Master
- Resource Manager
- Node Manager

## Task Failure

| | |
|---|---|
| Due to runtime exceptions | • JVM reports error back to parent application master |
| Hanging tasks | • Progress updates not happening for 10 mins<br>• Timeout value can be set. |
| Killed tasks | • Speculative duplicates can be killed |
| Recovery | • AM tries restarting task on a different node |

# Application Master Failure

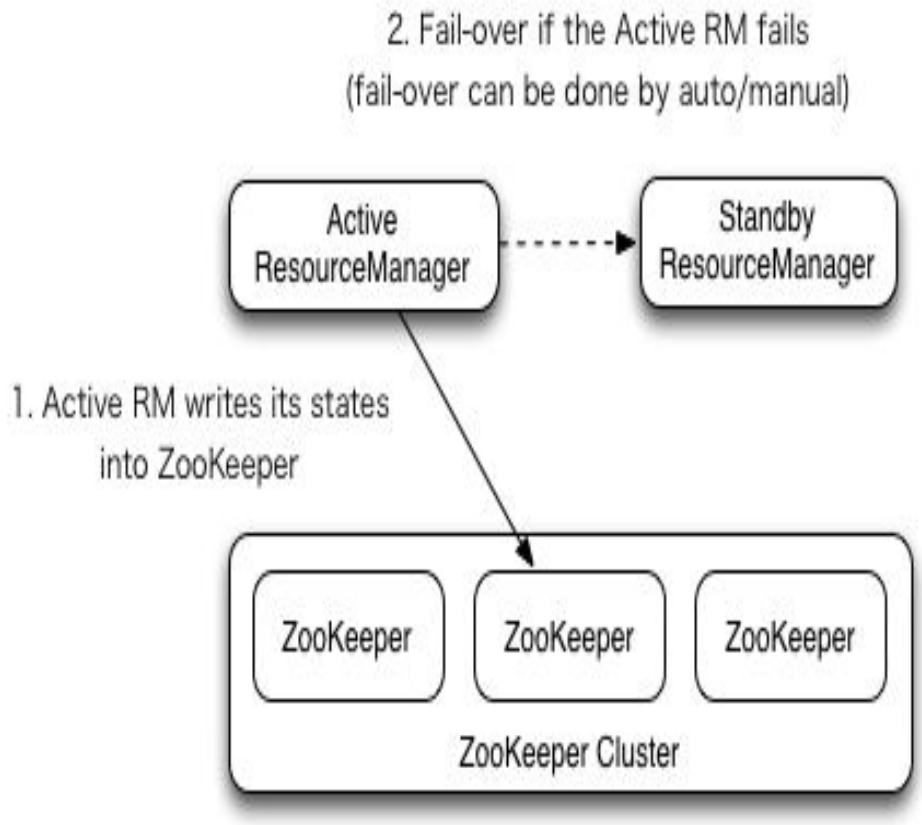
| | |
|---|---|
| When can failure occur? | • Due to hardware or network failures |
| How to detect for failures? | • AM sends periodic heartbeats to Resource Manager |
| Restart | • Max-attempts to restart application<br>  • Default = 2 |

## Node Manager Failure

| | |
|---|---|
| When can failure occur? | • Hardware, crashing, slow network |
| How to detect for failures? | • When a heartbeat is not received by RM for 10mins |
| Restart | • Tasks of incomplete jobs will be rerun – maybe on different node |

# Resource Manager Failure

| | |
|---|---|
| How is failure handled? | • Active Standby configuration |
| Impact | • More serious as all tasks fail |
| Restart | • Handled by failover controller |




https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/ResourceManagerHA.html

# Benefits of YARN

## YARN Benefits – Case Study @Yahoo

- YARN manages a very large cluster at Yahoo
  - Scalability – to over 40,000 servers with 100,000 CPUs, 455 PB of data
    - Runs over 850,000 jobs per day
  - Flexibility
    - Same cluster has Hadoop, Storm and Spark (100 node cluster) sharing resources using YARN

https://www.techrepublic.com/article/why-the-worlds-largest-hadoop-installation-may-soon-become-the-norm/

# Review Exercises

## Review Exercises

- All problems listed in T1 as part of LO2.5

## Solved Exercises

- A 1000 node YARN cluster has no jobs running. Two pools are configured with max of 50% of the resources. A new job requiring 600 nodes is submitted and on starting consumes all 600 nodes. Which YARN scheduler is active?
  - Either FIFO or Fair because they will use the entire cluster if there is no other job.
- Will the failure of task result in failure of the entire job?
  - No. Task will be restarted
- What are speculative duplicates?
  - Tasks that are started when AM determines that there is a slow running task.

# Additional Notes, Reference Material and Notes

## YARN Further Reading

- Chapter 2.5 of T1
- Chapter 4  in T2
- https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html
- There is a good description of YARN in the Tom White book.
- Also follow links from slides given before


# THANK YOU

**Prafullata Kiran Auradkar**

Department of Computer Science and Engineering

**prafullatak@pes.edu**