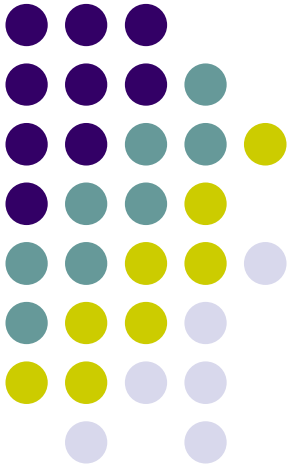
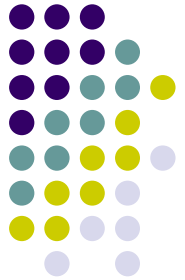
# Hadoop and Map-Reduce

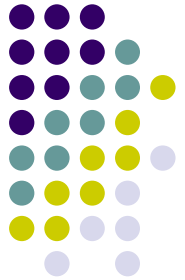
Swati Gore



# Contents

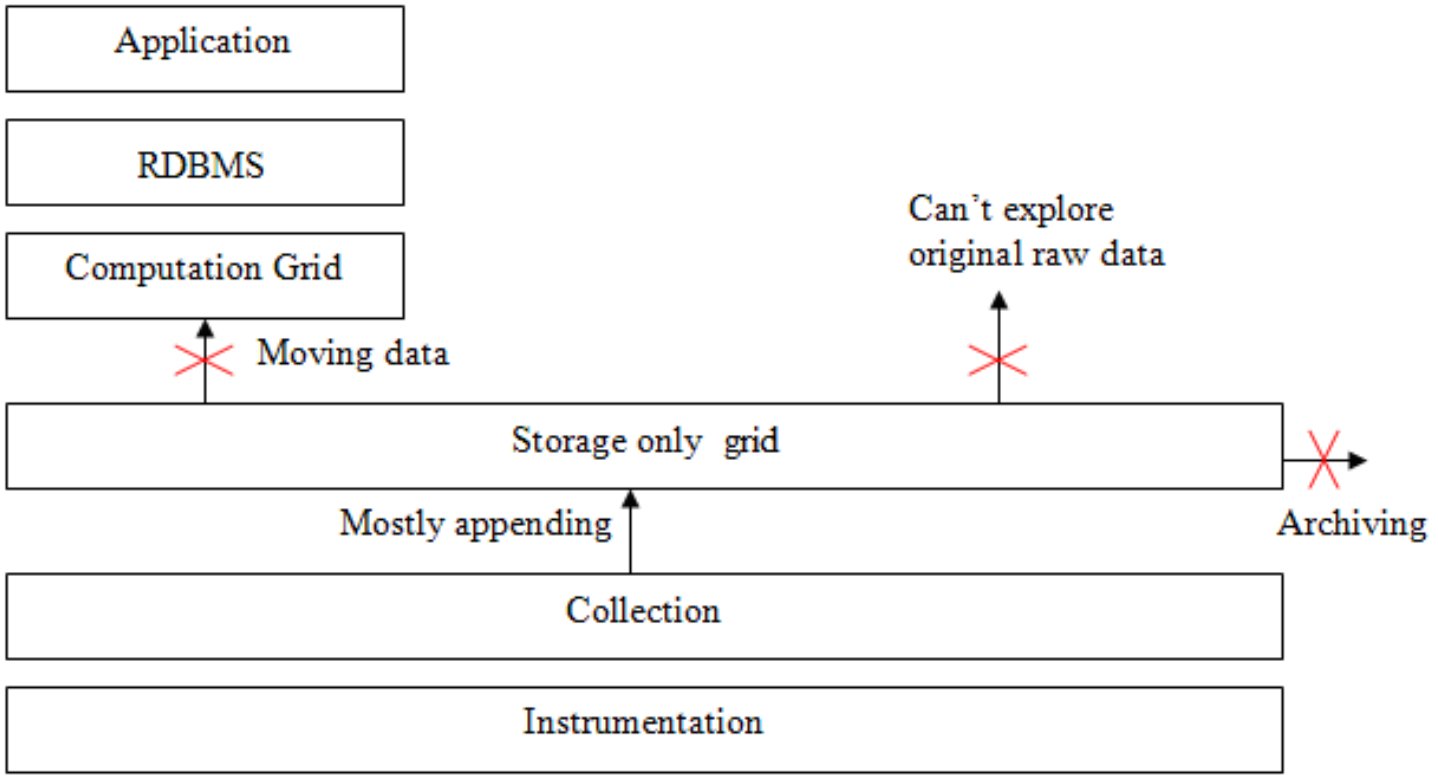


- Why Hadoop?
- Hadoop Overview
- Hadoop Architecture
- Working Description
- Fault Tolerance
- Limitations
- Why Map-Reduce not MPI
- Distributed sort



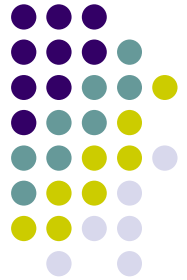
# Why Hadoop?

# Existing Data Analysis Architecture




Limitation of existing data analysis Architecture

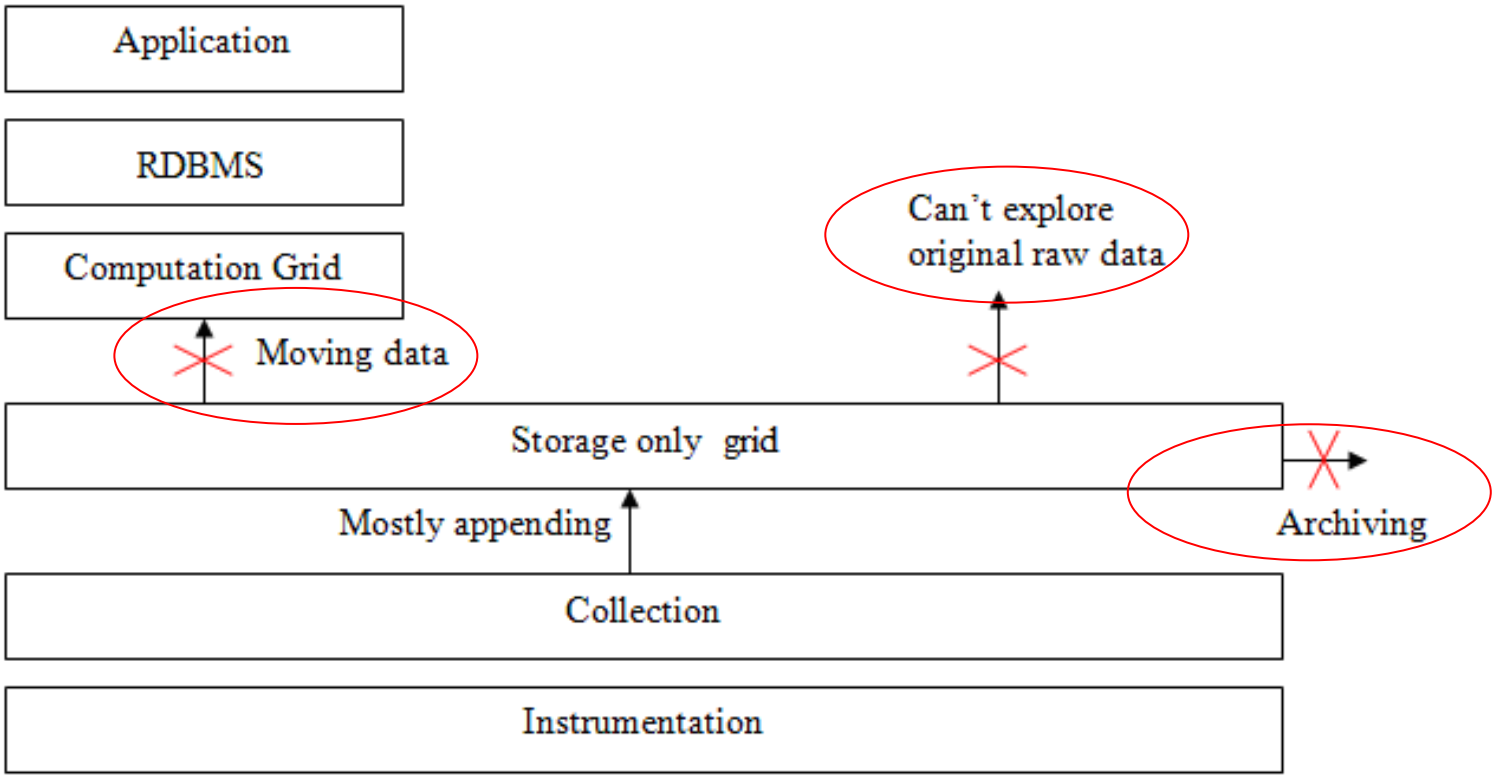
# Existing Data Analysis Architecture



- Instrumentation and Collection layer: obtains raw data from different sources like web server, cache registers, mobile devices, system logs etc and dumped on the storage grid.
- Storage grid: Store the raw data collected by Instrumentation and Collection layer.
- ETL Computation: Performs Extract Transform Load functions.
- Extract: Reading data from storage grid

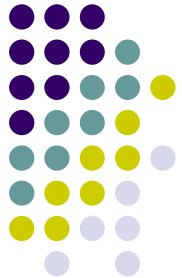
- Transform: Converting extracted data to the required form. In this case from unstructured to structured form.
- Load: Writing the transformed data to the target database.
- RDBMS: Like Oracle.
- Application Layer: Various applications that act on the data stored on RDBMS and obtain required information.

# Existing Data Analysis Architecture




Limitation of existing data analysis Architecture

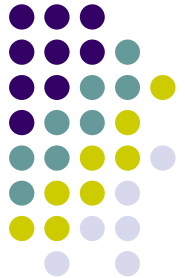
# Limitations!



Three limitations:

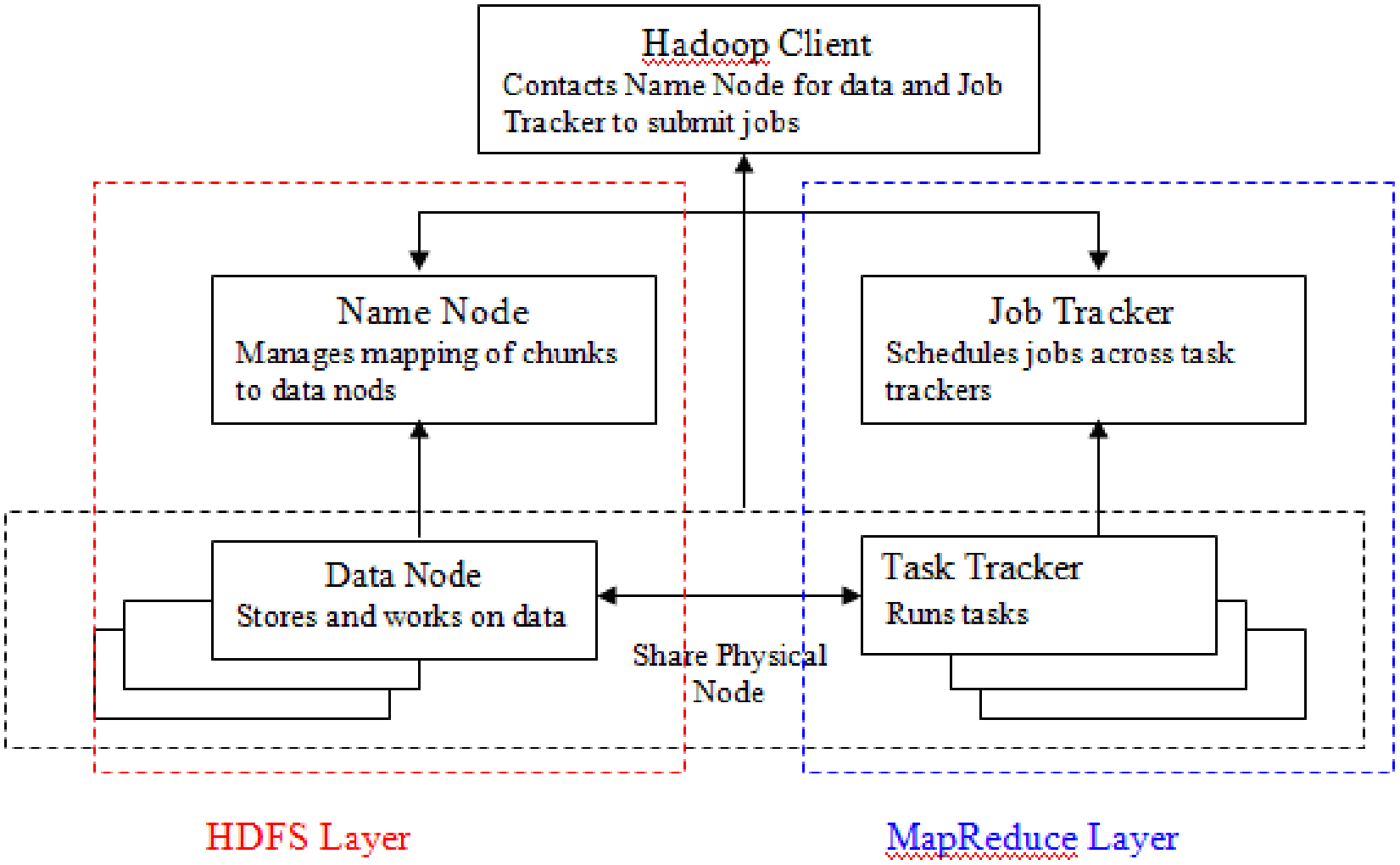
1. Moving stored data from storage grid to computation grid.
2. Lost of original raw data. Can not use it for any new information
3. Archiving leading to death of data

# Hadoop Overview

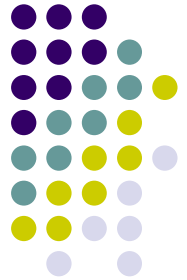


- Hadoop in a nut shell is an operating system.
- Has two major components
- HDFS: Hadoop Distributed File System
- Map-Reduce: Application that works on the data stored in HDFS and act as resources scheduler.

# Hadoop Overview

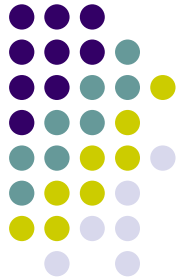


# Benefits of Hadoop over existing architectures



- Existing architecture employs *Schema-on-Write* (RDBMS) model
- Create Schema
- Transform data so as to fits it into this schema.
- Load data.
- Obtain required information.
- Efficient for compression, indexing, partitioning but difficult to adapt to changes.

# Example of *Schema-on-Write*

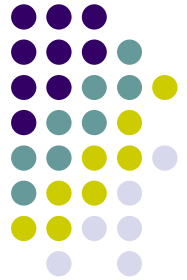


- Student database

| UIN | NAME | DEPARTMENT | GPA |
|---|---|---|---|
| 1 | TIM | CS | 4.0 |
| 2 | JOHN | ECE | 3.5 |
| ---- | | | |

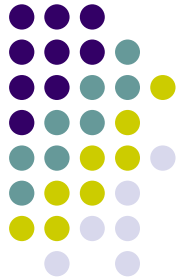
- Percentage of students getting placed?

# Steps needed to adapt to new requirements



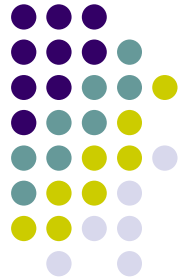
- Convince management to allow major change in existing database.
- Change the schema.
- Get required data.
- Then get desired information.

# Hadoop:
# Schema-on-read model

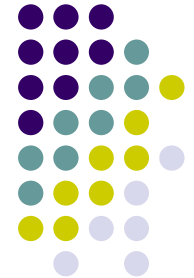


- Data is simply copied to the file without any transformation.
- Transformation done only when data is queried.
- Thus adapting to changes becomes very easy.

# Schema-on-read vs Schema-on-Write

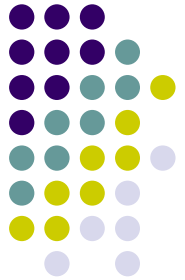


- Thus both systems have their  pros and cons.
- Even though SQL is good enough for solving complex data mining problems it still can't help us with say, image processing.
- Similarly Hadoop's Schema-on-read is not suited for ACID transactions like bank transactions



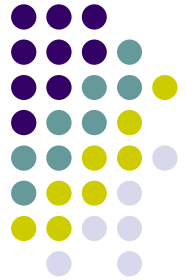
- Thus existing architecture is like you are given all ingredients chopped and ready to use for cooking. You need to make the dish.

- Whereas in Hadoop you have direct access to the pantry so you can choose any type of ingredients and make any dish you like.



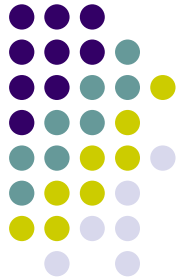
- Schema-on-Write take less time but it restricts ability while Schema-on-read does not impose any restriction, allowing us to make the best of raw objects.

# Languages used in Hadoop



- Hadoop gives us the flexibility to use any language to write our algorithms.
- We use Java Map-Reduce, Streaming Map-Reduce (works with any programming language like C++, Python), Crunch (Google), Pig latin (Yahoo), Hive (Facebook), Oozie (links all together).
- We don’t have to stick to SQL.

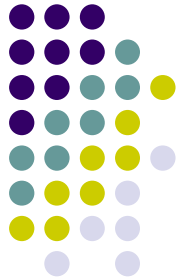
# Hadoop Architecture



- This architecture consists of three major components - the Client, the Master node and the slave nodes.

- Client: can be any organization which has a lot of unstructured data from different sources like web server, cache registers, mobile devices, system logs etc. and wants to analyze this data.

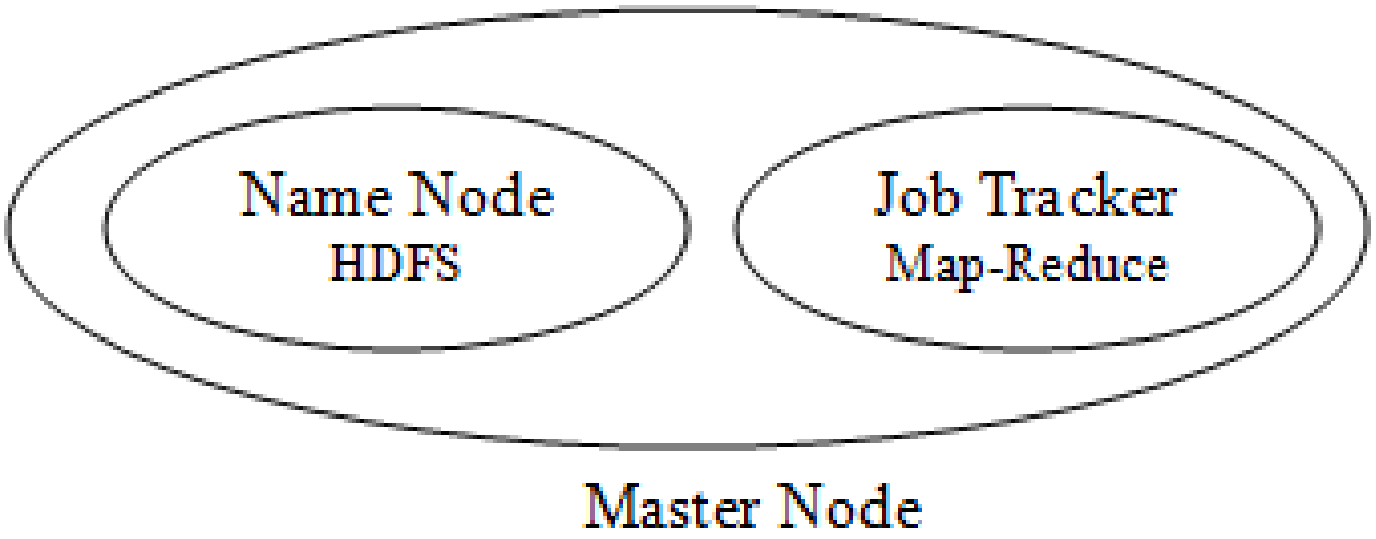
# Master node

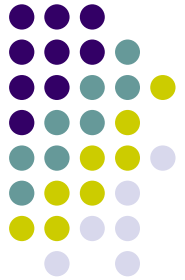


Responsible for two functionalities:

1. Distributing data obtained form the client to the HDFS
2. And assigning tasks to Map-Reduce layer.

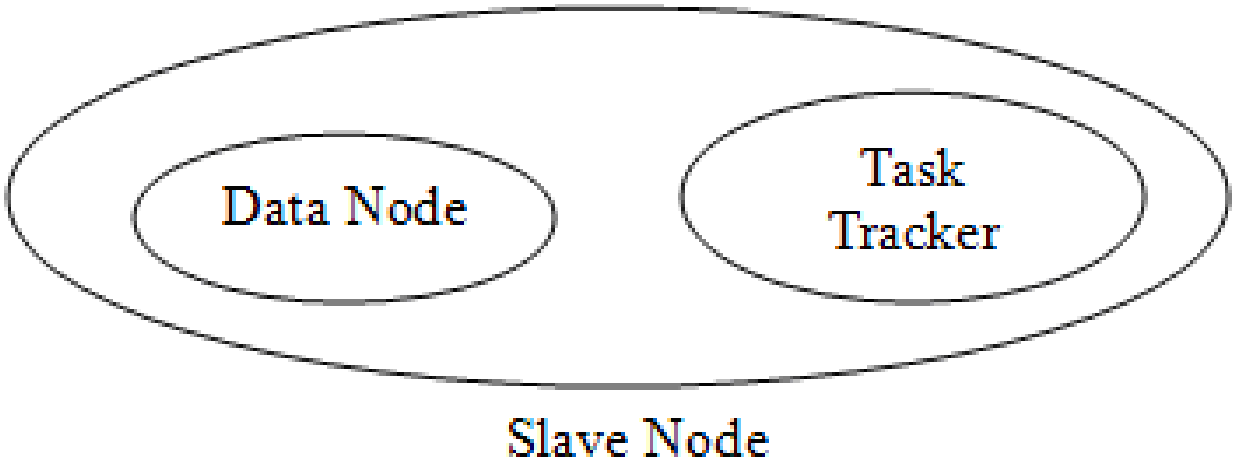
# Master Node



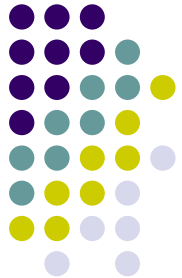


- These functions are handled by two different nodes:
- Name Node: helps in coordinating HDFS
- Job Tracker: helps in coordinating parallel execution of Map-Reduce.

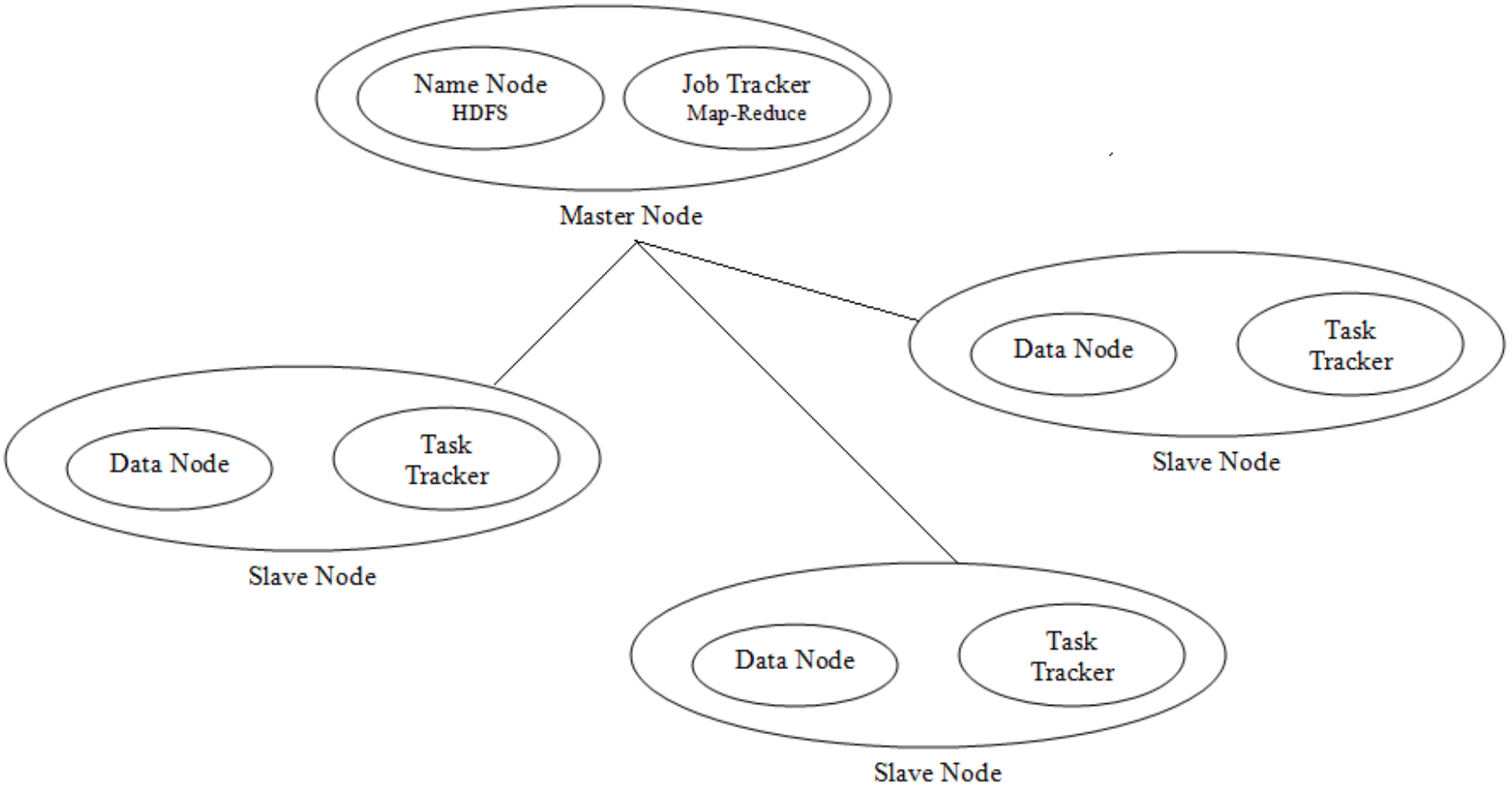
# Slave Node



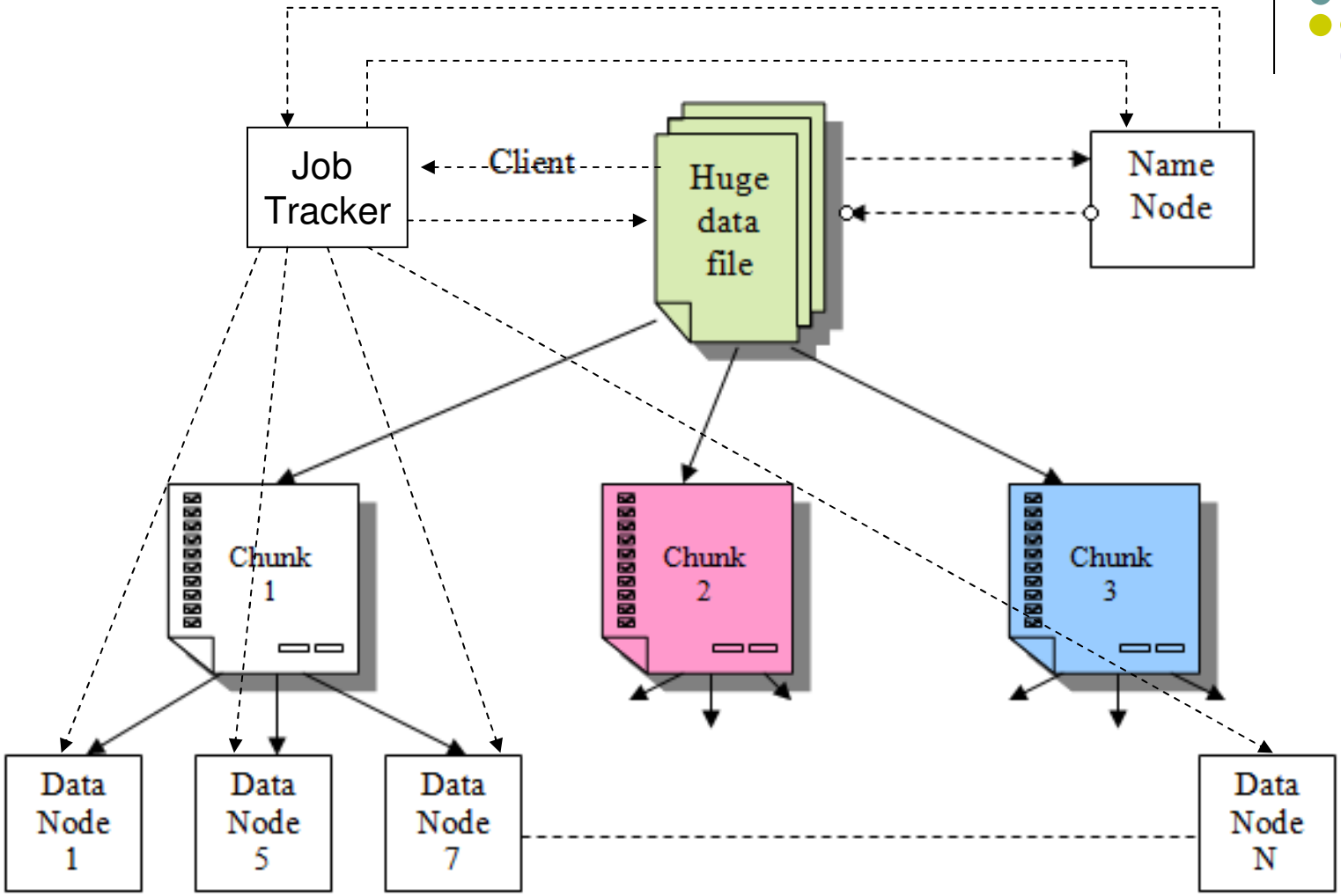
# Slave Node



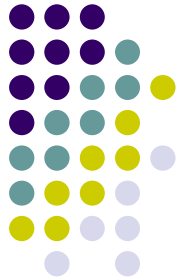
- Slave node consists of Data Node and Task Tracker.
- There are number slave nodes in a Hadoop cluster.
- These nodes communicate and receive instructions for the Master nodes and perform the assigned tasks.


Name Node
HDFS
Job Tracker
Map-Reduce
Master Node
Data Node
Task
Tracker
Slave Node
Data Node
Task
Tracker
Slave Node
Data Node
Task
Tracker
Slave Node

# Working

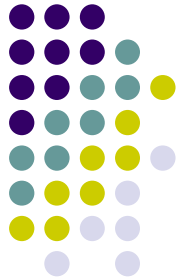

Job Tracker
Client
Huge data file
Name Node
Chunk 1
Chunk 2
Chunk 3
Data Node 1
Data Node 5
Data Node 7
Data Node N

# Name Node



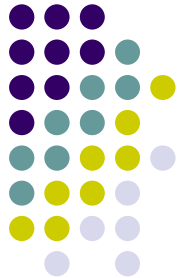
- Name Node is a critical component of the HDFS.
- Responsible for the distribution of the data throughout the Hadoop cluster.
- Obtains the data from client machine divides it in to chunks.
- Distributes same data chunk to three different data nodes leading to data redundancy.

# Name Node's Role



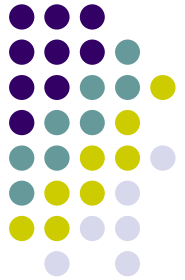
- Keeps track of What chucks belong to a file and which Data Node holds its copy.
- cluster's storage capacity.
- Making sure each chunk of file has the minimum number of copies in the cluster as required.
- Directs clients for write or read operation
- Schedule and execute Map Reduce jobs.

# Job Tracker



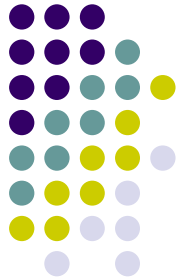
- Client submitted Map-Reduce logic.
- Responsible for scheduling the task to the slave nodes.
- Job Tracker consults the Name Node and assigns the task to the nodes which has the data on which this task would be performed.

# Data Node



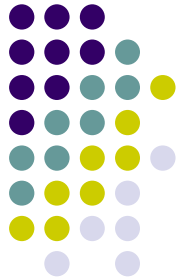
- One of the slave node part
- Responsible to store the chunk of that assigned to it by the Name Node.

# Task Tracker

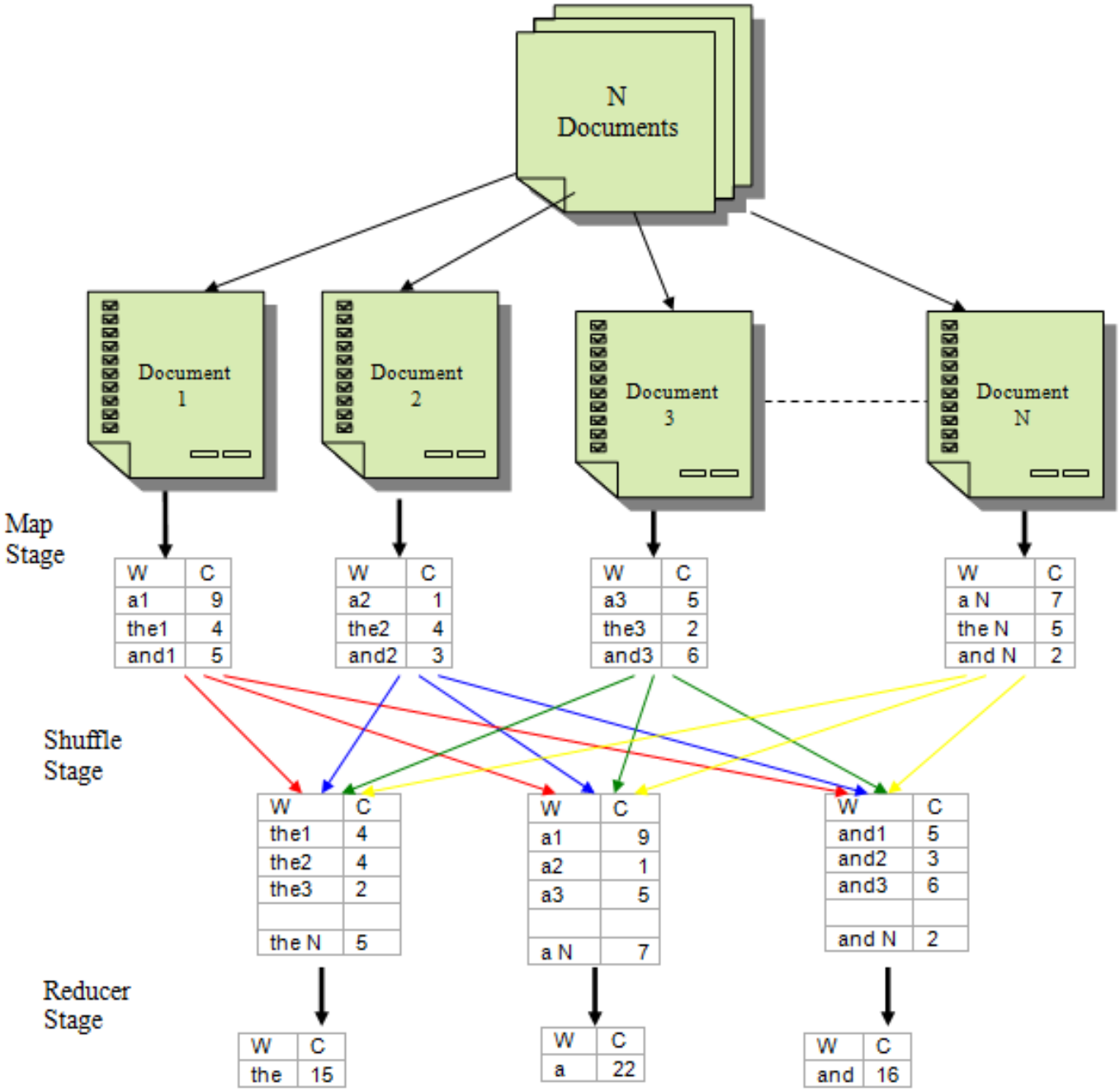


- Other part of slave node.
- Has the actual logic to perform the task.
- Performs the task (Map and reduce functions) on the data assigned to it by Master Node.

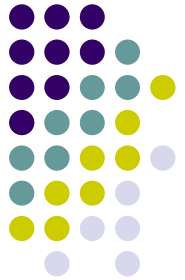
# Map-Reduce Function



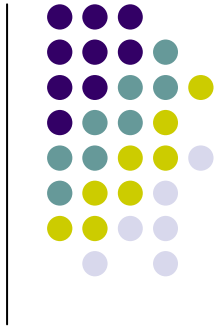
- Map: Takes a single <key, value> pair, and produces zero or more new <key, value> pairs that may be of different type.
- Reduce: Works on the output of Map function and produces desired result.


N
Documents
Document
1
Document
2
Document
3
Document
N
Map
Stage
W C
a1 9
the1 4
and1 5
W C
a2 1
the2 4
and2 3
W C
a3 5
the3 2
and3 6
W C
a N 7
the N 5
and N 2
Shuffle
Stage
W C
the1 4
the2 4
the3 2
the N 5
W C
a1 9
a2 1
a3 5
a N 7
W C
and1 5
and2 3
and3 6
and N 2
Reducer
Stage
W C
the 15
W C
a 22
W C
and 16

# Fault Tolerance

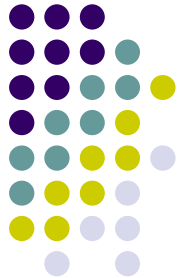


- A system is said to be fault tolerant if can continue to work properly and does not lose any data when some component crashes.
- HDFS achieves this by data redundancy.



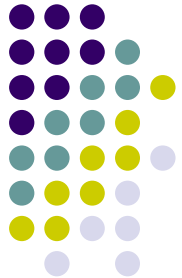
# Failure of Slave Node

# Data Node failure

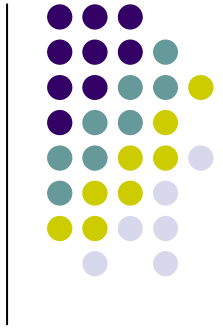


- Data Node send a periodic heartbeat to Name Node.
- Missing heartbeat helps detected Data Node failure.
- In such a case Name Node removes the crashed Data Node from the cluster and redistributes its chunks to other surviving Data Nodes.

# Task Tracker Failure

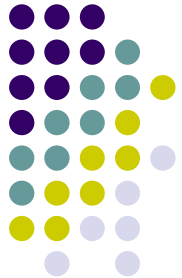


- Just like Data Node, Task Tracker's failure is detected by the Job Tracker after missing its heartbeat.
- Job Tracker then decides the next step:
- It may reschedule the job elsewhere or
- It may mark specific record to avoid or
- It may blacklist this Task Tracker as unreliable.



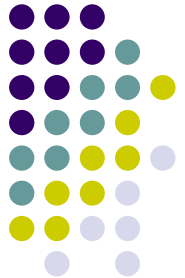
# Failure of Master Node

# Name Node failure



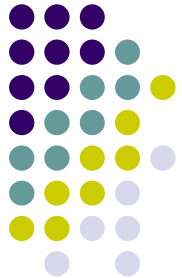
- Since Name Node contains all the important information required for the data distribution and computation,  If it crashes, HDFS could fail.
- To overcome this single point failure Hadoop introduced the Backup Node.

# Backup Node



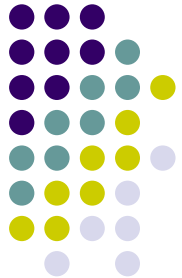
- Backup Node regularly contacts Name Node and maintains an up to date snapshots of Name Node's directory information.
- When Name Node fails Backup helps to revive the Name Node.

# Job Tracker Failure



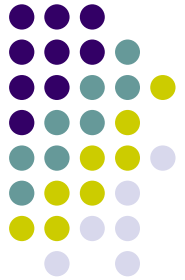
- Again a single point failure
- If it goes down, all running jobs fail.
- In the latest Hadoop architecture Job Tracker is split into application logic such that there can be a number of masters having different logic.
- Thus the cluster is capable of supporting different application logic.

# Name Node's rack-aware policy

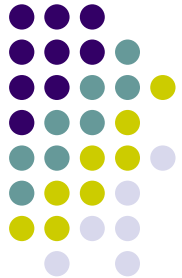


- A large Hadoop clusters is arranged in racks.
- Name Node assigns data blocks that involves minimum data flow across the racks.
- But it also keeps in mind that replicas need to be spread out enough to improve fault tolerance.

# Drawbacks/ limitations



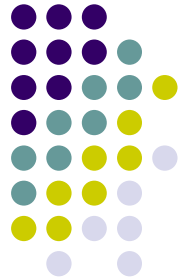
- Since Name Node is the single point for storage and management of metadata, this can be a bottleneck for supporting a huge number of files, especially a large number of small files.
- For a very robust system replica distribution should be on totally separate racks.



- But would lead to high communication overhead especially during write operation, as all replicas must be updated.

- Thus HDFS write two of the replicas on different nodes of the same rack and write the other one on a totally separate rack.

# Why Map-Reduce not MPI?



- Firstly Map-Reduce provides a transparent and efficient mechanism to handle fault where as errors in MPI are considered fatal and only way to handle them is to abort the job.
- Secondly in order for MPI to support Map-Reduce operations, it would need variable-sized reduction function because Reducer in Map-Reduce can return values that are of a different size than the input values, e.g., string concatenations.

- The basic difference is that Map-Reduce and MPI were developed by two different communities that are traditionally disjoint and were developed to cater different needs and capabilities.

- As the interests of both converge they can benefit by taking advantage of these respective technologies.

# Benefits of using Hadoop

- Firstly since both map and reduce functions can run in parallel, allow the runtime to be reduces to several optimizations.

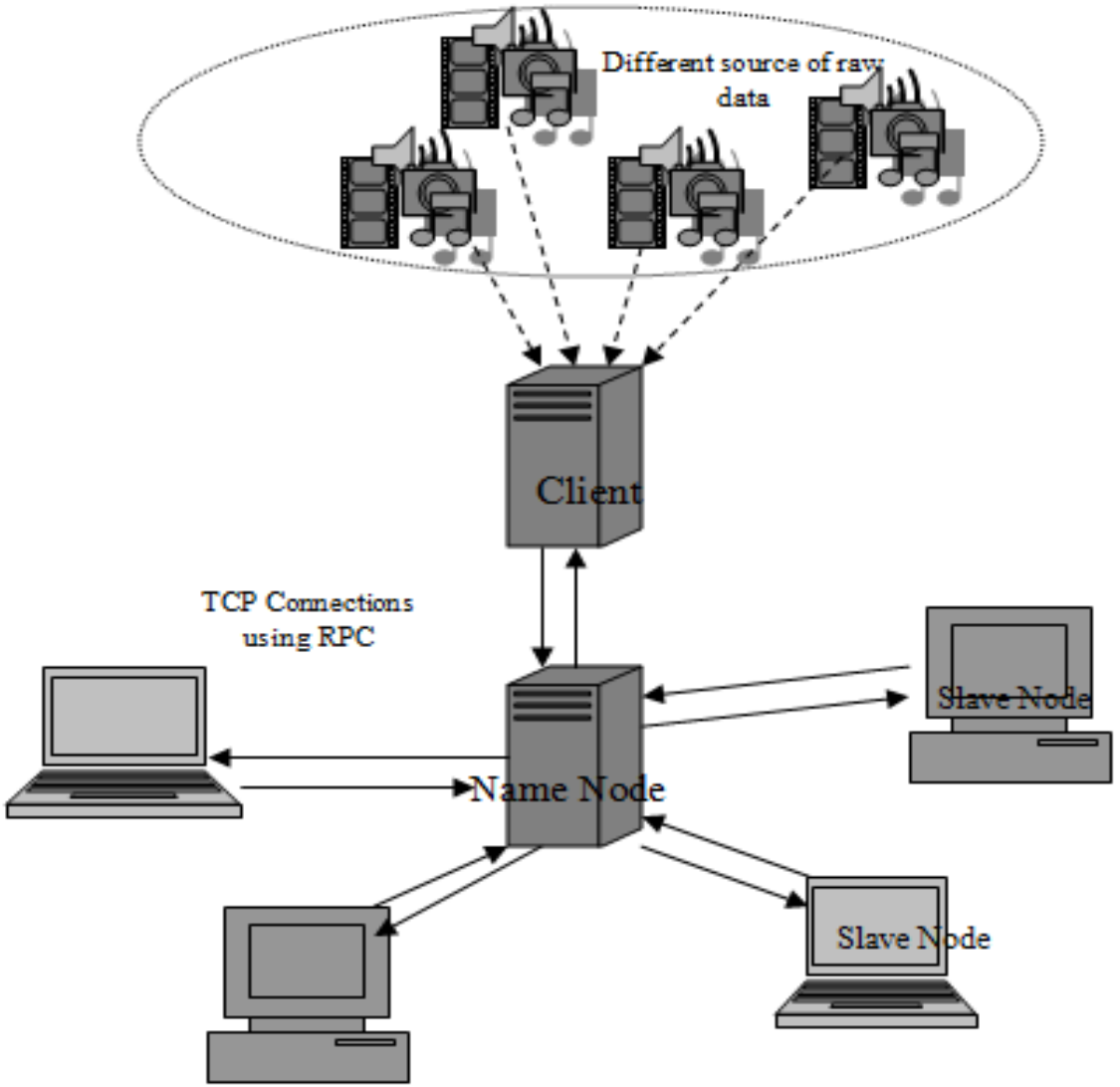
- Secondly Map-Reduce is fault resiliency which allows the application developer to focus on the important algorithmic aspects of his problem while ignoring issues like data distribution, synchronization, parallel execution, fault tolerance, and monitoring.

- Lastly be using Apache Hadoop, we avoid paying expensive software licenses; get flexibility to modify source code to meet our evolving needs; and avail themselves of leading-edge innovations coming from the worldwide Hadoop community.

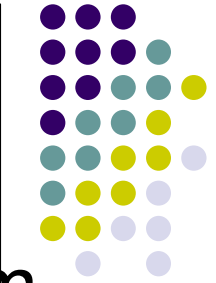
# The Communication Protocols

- All HDFS communication protocols are built on top of the TCP protocol.
- A client establishes a TCP connection with the Name Node machine.
- Once the connection is set they talk using the Client Protocol.
- Similarly the Data Nodes talk to the Name Node using the Data Node Protocol.
- Both the Client Protocol and the Data Node Protocol use Remote Procedure Call (RPC) abstraction.

# The Communication Protocols

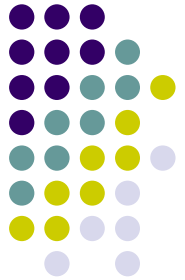


# Companies using Hadoop



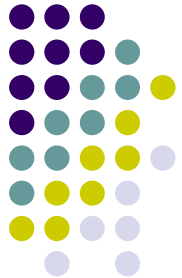
- Netflix's movie recommendation algorithm uses Hive (underneath using Hadoop, HDFS & Map-Reduce) for query processing and Business Intelligence.
- The Yahoo! Search Webmap is a Hadoop application that runs on a more than 10,000 core Linux cluster and produces data that is now used in every Yahoo! Web search query.
- Facebook uses largest Hadoop cluster in the world with 21 PB of storage.

# Applications of Map-Reduce



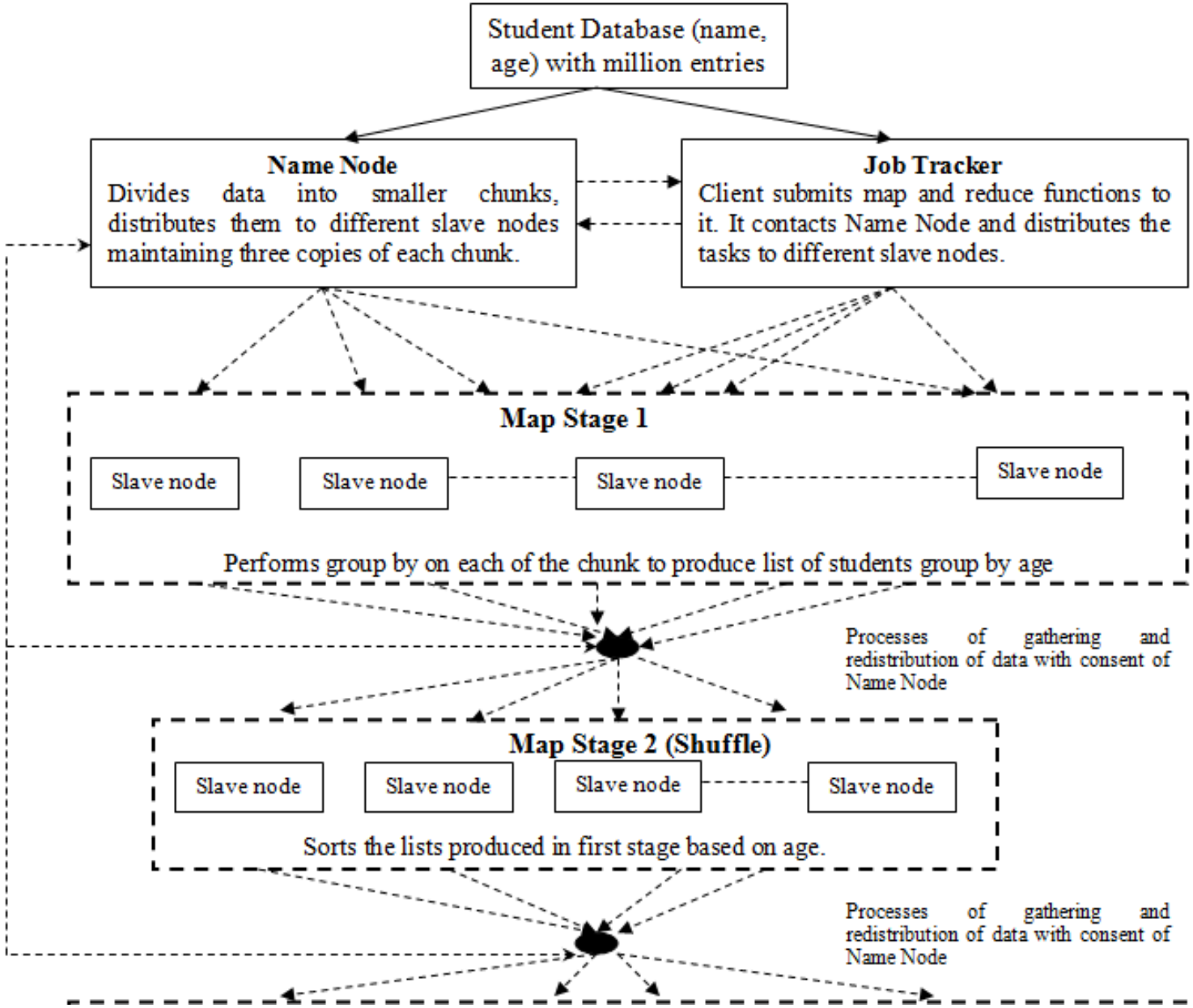
- distributed grep
- distributed sort
- web link-graph reversal
- term-vector per host
- web access log stats
- inverted index construction
- document clustering
- machine learning
- statistical machine translation

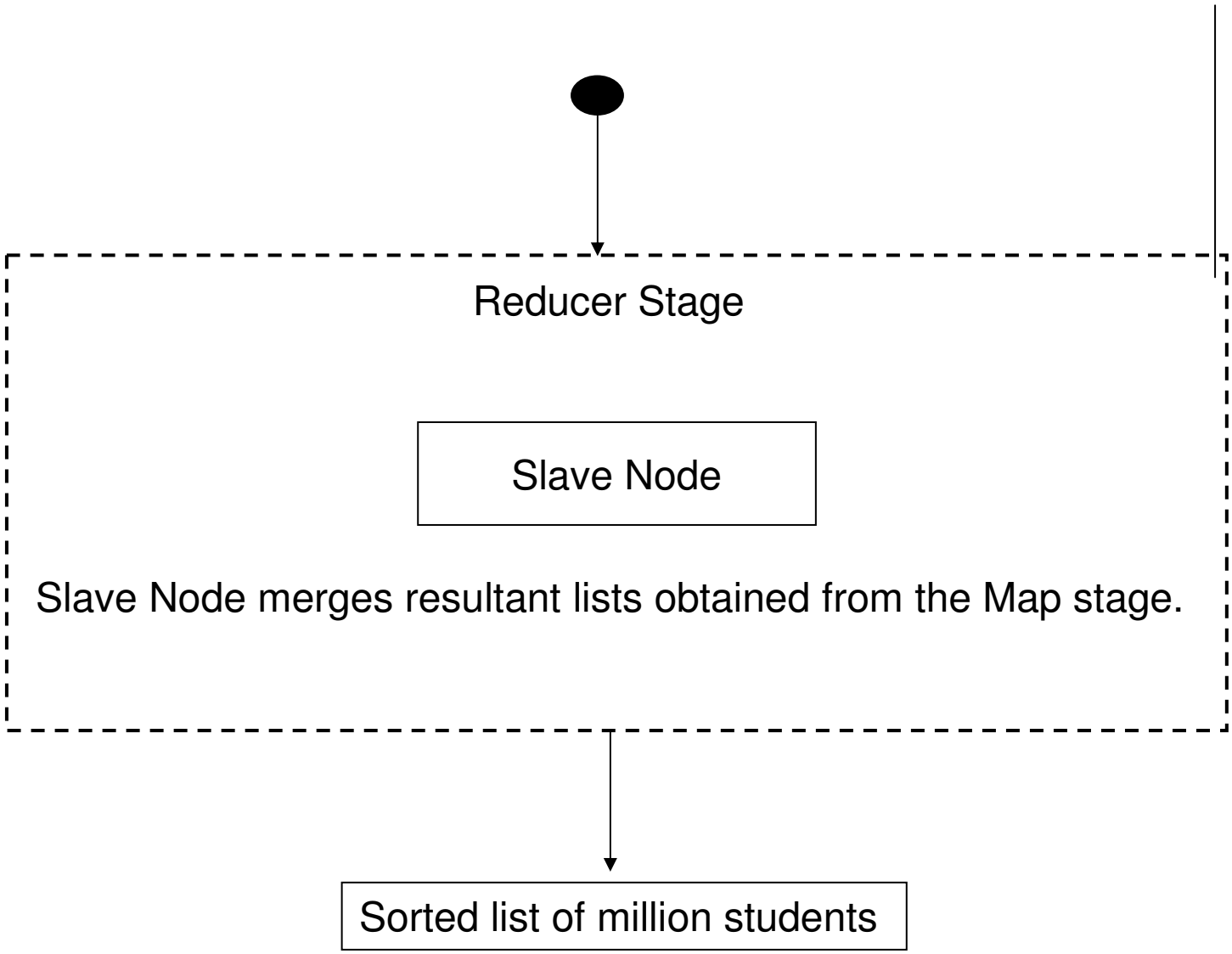
# Distributed Sort

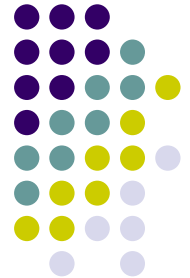


- Suppose we want to sort 1 million entries in a student database based on their age (Age-Name).
- We want to sort using Age.

# Flow Chart


Student Database (name, age) with million entries
Name Node
Divides data into smaller chunks, distributes them to different slave nodes maintaining three copies of each chunk.
Job Tracker
Client submits map and reduce functions to it. It contacts Name Node and distributes the tasks to different slave nodes.
Map Stage 1
Slave node
Slave node
Slave node
Slave node
Performs group by on each of the chunk to produce list of students group by age
Processes of gathering and redistribution of data with consent of Name Node
Map Stage 2 (Shuffle)
Slave node
Slave node
Slave node
Slave node
Sorts the lists produced in first stage based on age.
Processes of gathering and redistribution of data with consent of Name Node

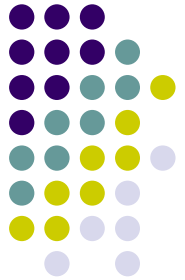

Reducer Stage
Slave Node
Slave Node merges resultant lists obtained from the Map stage.
Sorted list of million students



- Now we can ask a question that if there is only one reducer where the results are combined, how does it that advantage of the distributed framework?

- Won’t merging 1 million entries at one node cause processes to slow down?

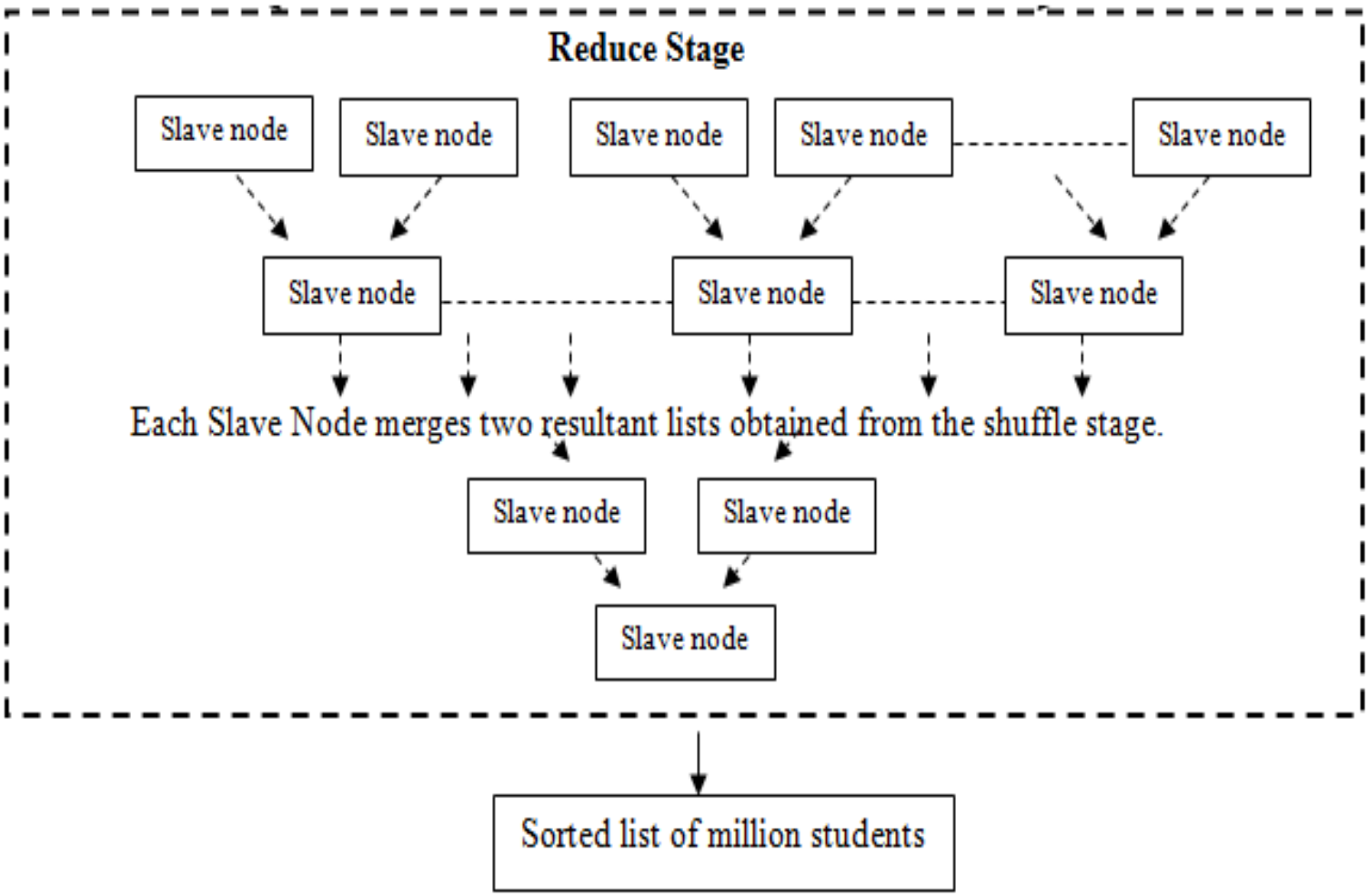
- The answer is yes!



- But it turns out that it is more efficient to merge partially sorted listed to produce a sorted list then to sort complete list.

- The reducers work is very simple it just looks at the smallest elements from the all the lists it obtained as input and pick the one that is the smallest.

- We can parallelize reducers work but dividing the resultant list in batches of two between a number of reducers.
- Each of them would merge two lists.
- Thus the final reducer would be responsible to merge only two lists.

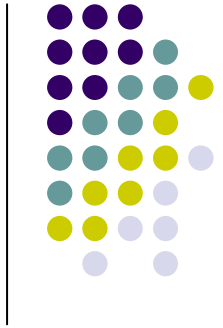

Reduce Stage
Slave node
Slave node
Slave node
Slave node
Slave node
Slave node
Slave node
Slave node
Each Slave Node merges two resultant lists obtained from the shuffle stage.
Slave node
Slave node
Slave node
Sorted list of million students

- In any other architecture this parallel execution would be restricted due to the topology which has to be defined before the execution starts.

- Where as in Hadoop cluster we are not bound by this restriction, the data distribution and execution is not the programmer's responsibility, its taken care by HDFS.

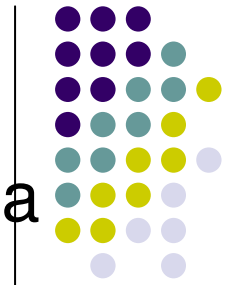
- Programmer just has to make sure the mapper and reducer functions are written in accordance with the Map-Reduce paradigm.

# CDH: Cloudera's Distribution Including Apache Hadoop

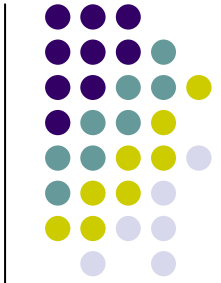


- Hadoop is only the kernel.
- Just like in Linux. We can get the kernel from kernel.org but there is Red Hat which is the distribution which has the kernel, libs, apache web server, xwindows which make it a complete product.
- Cloudera does the same for Hadoop.
- It adds HBase, Apache Flume, Hive, Oozie, installer which is not open source but is free,  we just need to submit the IP addresses of different system and it takes care of the rest of the installation.

# Conclusion



- Hadoop is an operating system that spans over a grid of computers providing an economically scalable solution for storing and processing large amount of structured and unstructured data over long period of time.
- Hadoop architecture provides a lot of flexibility which traditional systems didn't. It is very easy to add information, adapt to changing needs. It can answers queries which are very difficult to solve using RDBMS.
- Hadoop is inexpensive as compared to the other data store solution available in the market making it very attractive. It also uses commodity hardware, open-source software.



# Questions?

## Thank You