

Apache Hadoop Developer Training

Som Shekhar Sharma
Founder & Director, XG (XORGlocal)

Mr. Som Shekhar Sharma

- Total 7 years of IT experience
- 3 years on Big data technologies Hadoop, HPCC
- Experienced in NoSQL DBs (HBase, Cassandra, Mongo, Couch)
- Data Aggregator Engine: Apache Flume
- CEP : Esper
- Worked on Analytics, retail, e-commerce, meter data management systems etc
- Cloudera Certified Apache Hadoop developer (CDH3 and CDH4)
- Cloudera Certified Data Scientist
- Cloudera Certified Instructor

Training Experience

- 10 years of training/teaching experience
 - B-Tech (Electronics and Telecommunication, Network Theory, EMT, Mathematics, statistics)
 - MTech (Information Theory, Advance Semiconductor)
- 1.2 years of Training experience on (Both online and classroom)
 - Apache Hadoop Developer,Apache Hbase, Sqoop, Flume, oozie, hive, pig
 - Storm, Esper
 - Apache Kafka
- 2 years of Corporate Trainings (few mentioned below)
 - Oracle Financial Services
 - HCL Vertex Tech Park
 - Bank Of America, Gurgaon
 - Bank of America Chennai
 - Torry Harris, Bangalore
 - Ernst and Young, Bangalore



The Credential of
Cloudera Certified Developer for Apache Hadoop (CCDH)

is hereby granted to

Sharma Kolluru

to certify that he/she has completed to satisfaction

Exam CCD-333 (CDH3)

Test Date: **December 21, 2012**

Sarah Shekhar

Authorized
Signature

Cloudera, Inc.
220 Portage Avenue
Palo Alto, CA 94306
www.cloudera.com

Dec 21, 2012

Date Granted



The Credential of

Cloudera Certified Developer for Apache Hadoop (CCDH)

is hereby granted to

Kolluru Som Shekhar Sharma

to certify that he/she has completed to satisfaction

Exam CCD-470 (CDH4)

Test Date: March 26, 2013

A handwritten signature in black ink that reads "Sarah Spiegel".

Authorized
Signature

Cloudera, Inc.
220 Portage Avenue
Palo Alto, CA 94306
www.cloudera.com

Mar 26, 2013

Date Granted

Score Report

Exam Date: 05/25/2013
Testing ID: CLD7783
Registration: 256892954
Center ID: 55960



Dear Sharma Kolluru:

The results of your Cloudera test Data Science Essentials are reported below.

DS-200 requirements: scaled score of 500 or higher (Scale of 0-700)

Your Score: 518

DS-200 Result: PASS

Congratulations! You passed the DS-200: Data Science Essentials. You have successfully completed part one of your OCP certification.

You now qualify to participate in part two of your OCP certification: a Data Science Challenge and earn your OCP: Data Scientist certification.

You must participate in and successfully complete one Data Science Challenge within 24 months of the day listed on this score report.

If you have any questions please visit

<http://university.cloudera.com/certification.html> or email certification@cloudera.com.

You may not modify or change this document's contents in any way, nor may you appropriate any elements of this document for use in other electronic documents or printed materials. You may only print the document in its entirety. Any other use of the document must be approved by Cloudera, Inc.

Thank you very much for your interest in the OCP program.

Module 1

Motivation & Basics

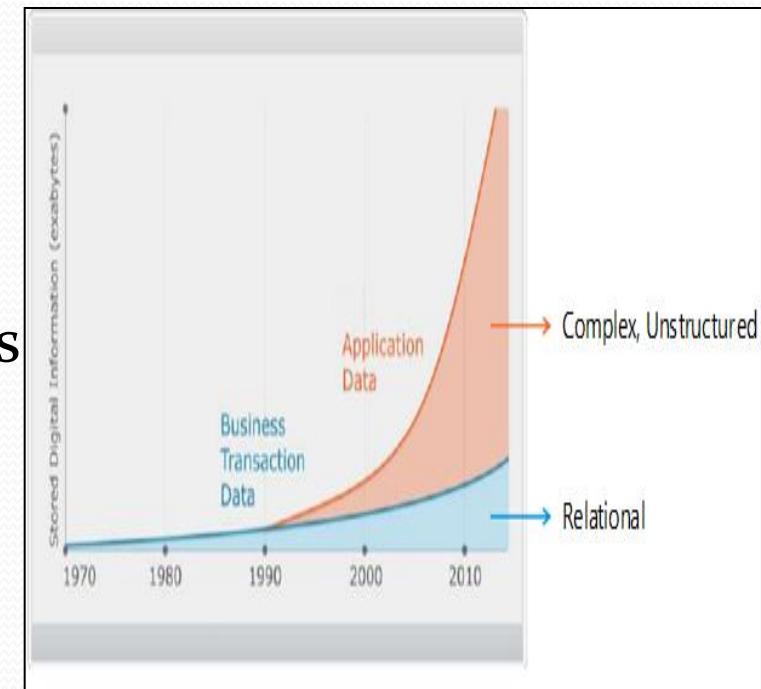
In this module you will learn

- What is big data?
- Challenges in big data
- Challenges in traditional Applications
- New Requirements
- Introducing Hadoop
- Brief History of Hadoop
- Features of Hadoop
- Over view of Hadoop Ecosystems
- Overview of MapReduce

Big Data Concepts

- Q.Which company comes to your mind first?
- Q.How many pages Google is storing behind the scenes?
- Q.How many photos FB is hosting?

- Volume
 - No more GBs of data
 - TB,PB,EB,ZB
- Velocity
 - High frequency data like in stocks
- Variety
 - Structure and Unstructured data



Real Definition of Big Data

- Getting Insights out of the data
- Finding out the why that transaction has happened in that way?

Large Amount Of data is required to analyze and hence the term Big Data

Analyze this statement “I am 59”

- It could be age
- It could be weight
- It could be height
- It could be FINE

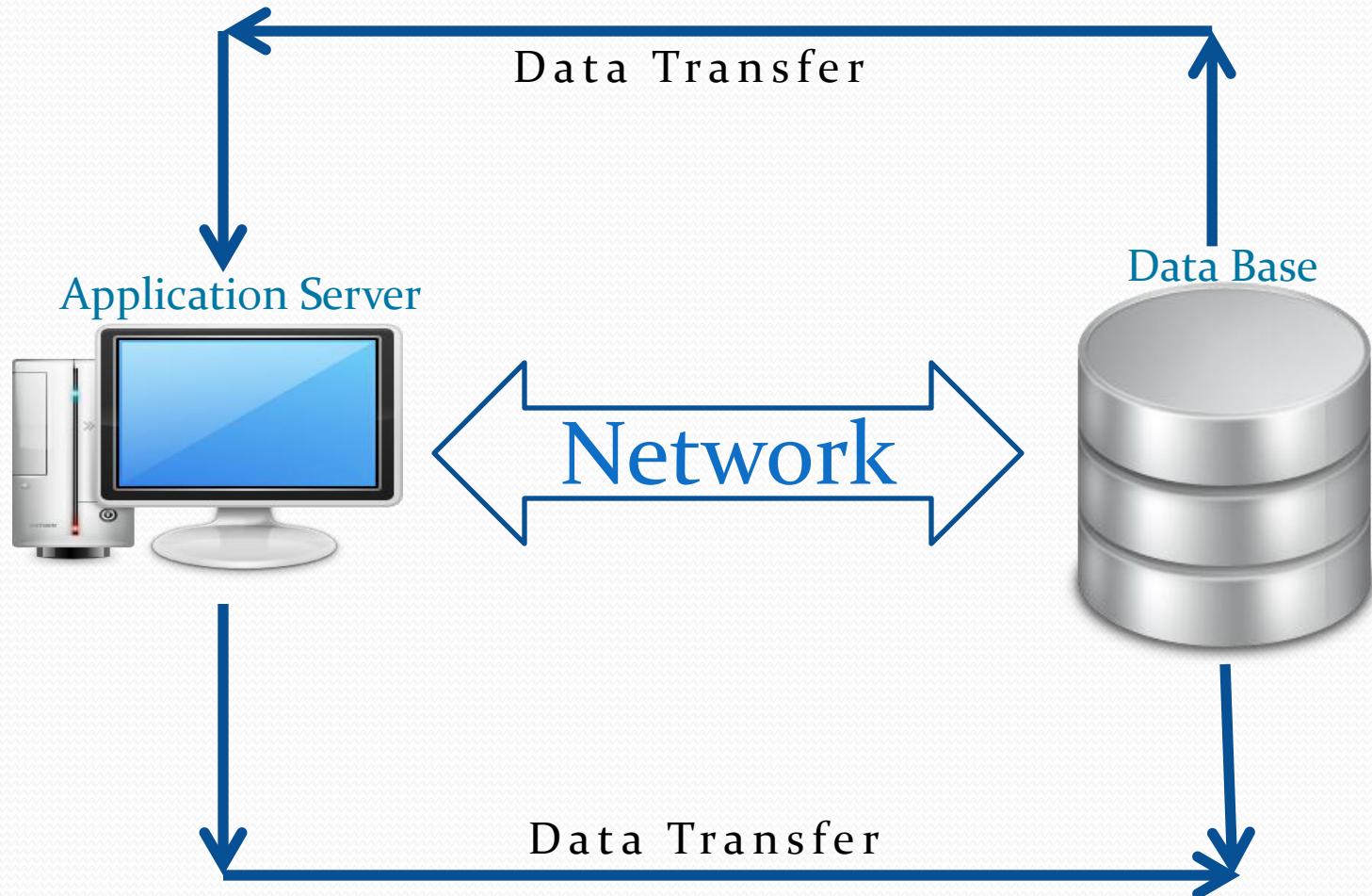
The sales of colgate tooth paste has decreased by 10% in Q4-2013 in BTM in Bangalore?

- It could be recession problem
- It could be bad rating in social media.

Challenges In Big Data

- Complex
 - No proper understanding of the underlying data
- Storage
 - How to accommodate large amount of data in single physical machine
- Performance
 - How to process large amount of data efficiently and effectively so as to increase the performance

Traditional Applications



Observation

- Network
 - No dedicated network line
 - Entire company's traffic has to go through same line
- Application
 - Size is constant
- Data
 - Large amount of data is transferred

Statistics – Part1

Assuming N/W bandwidth is 10MBPS

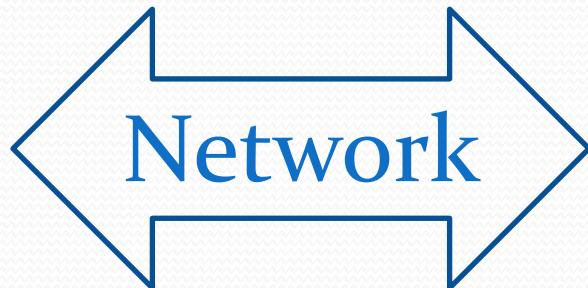
Application Size (MB)	Data Size	Total Round Trip Time (sec)
10	10MB	$1+1 = 2$
10	100MB	$10+10=20$
10	1000MB = 1GB	$100+100 = 200 (\sim 3.33\text{min})$
10	1000GB=1TB	$100000+100000=\sim 55\text{hour}$

- Calculation is done under ideal condition
- No processing time is taken into consideration

Analyze



10 MB



1 TB

Analyze



10 MB

Network



1 TB

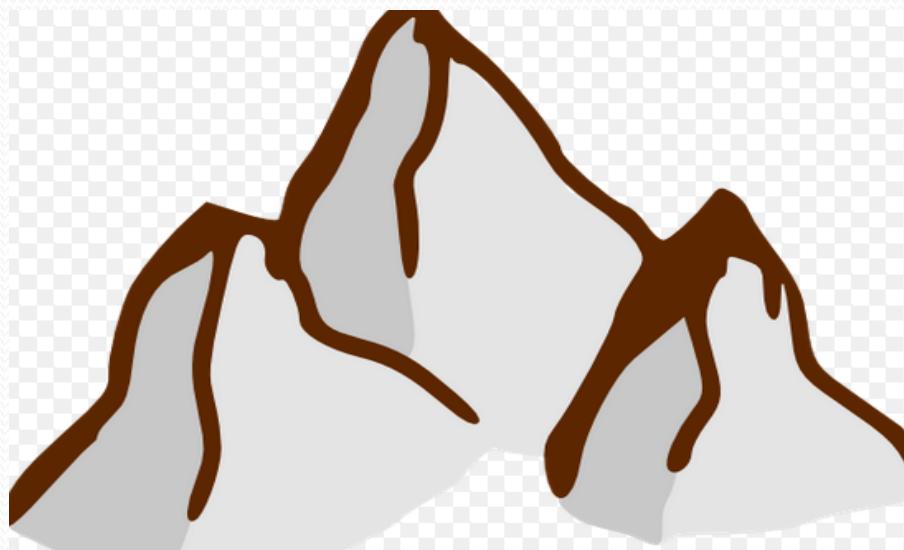


10 year

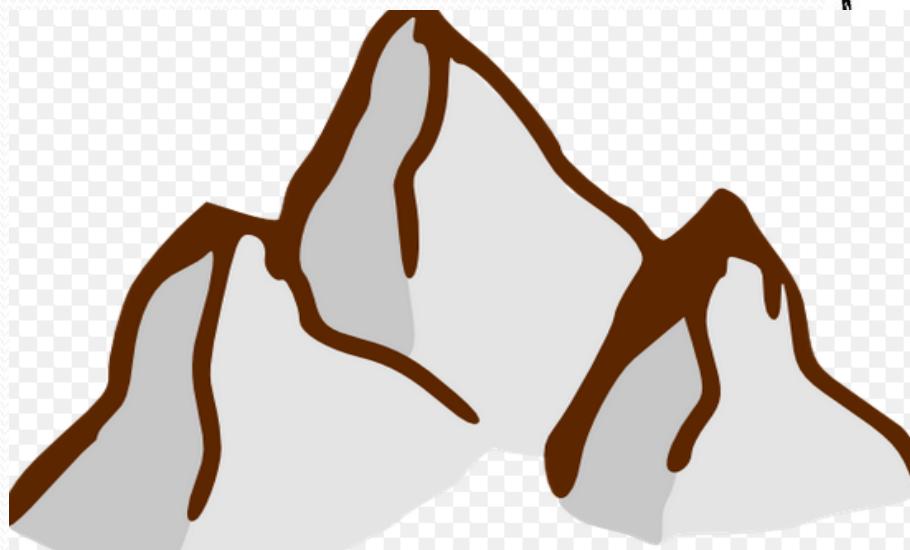


70 year

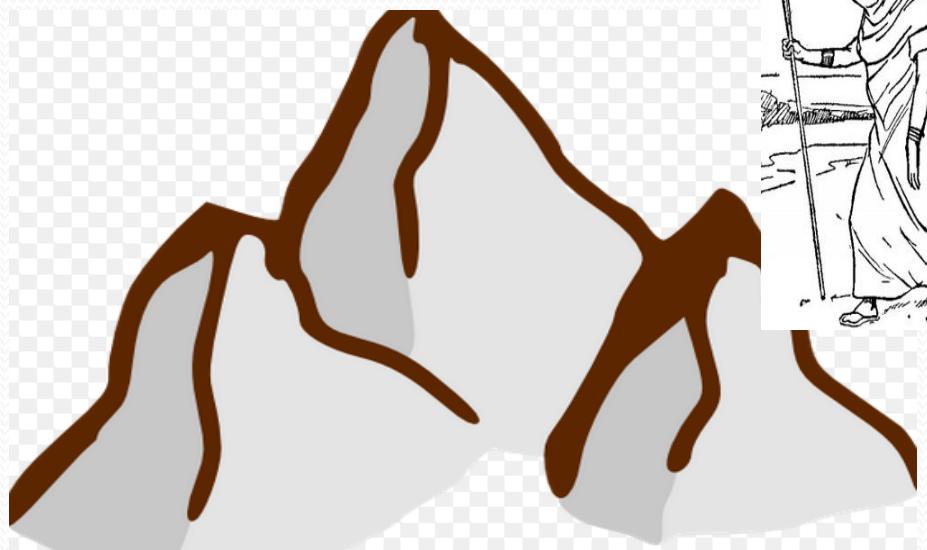
Analyze



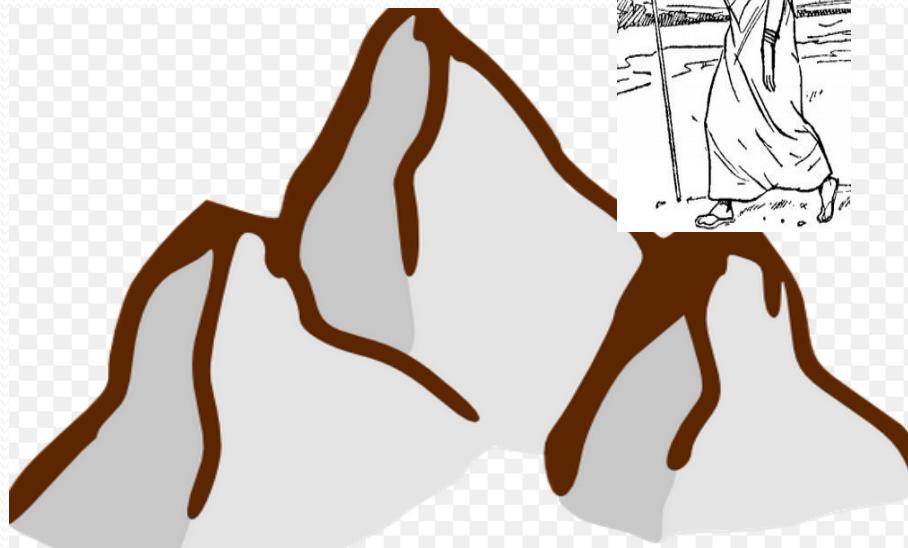
Analyze



Analyze



Analyze



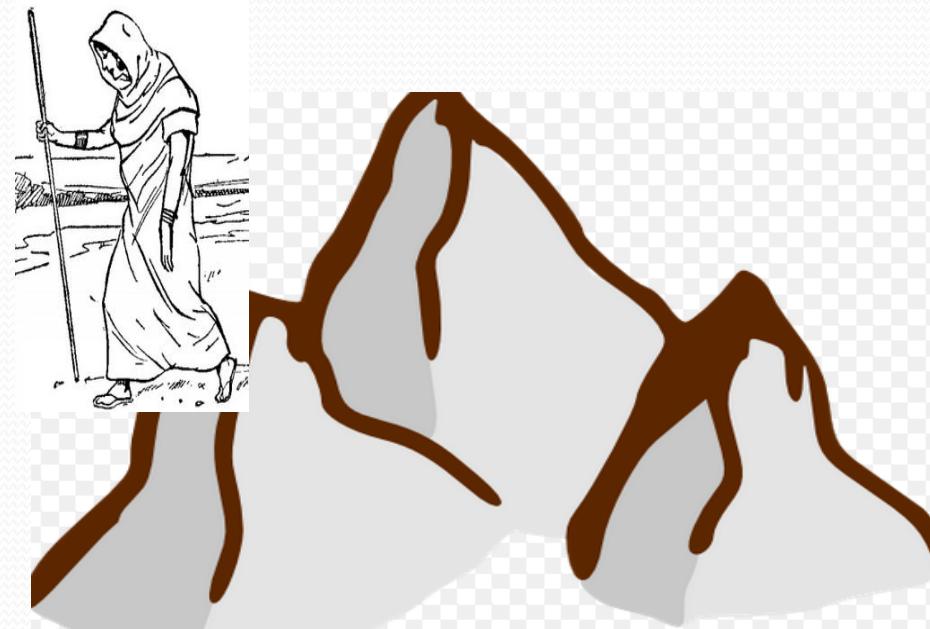
Analyze



Analyze



Analyze



Analyze

Takes too much of time.



Problem

- Data is moved back and forth over the low latency network where application is running
 - 90% of the time is consumed in data transfer

Solution



Solution

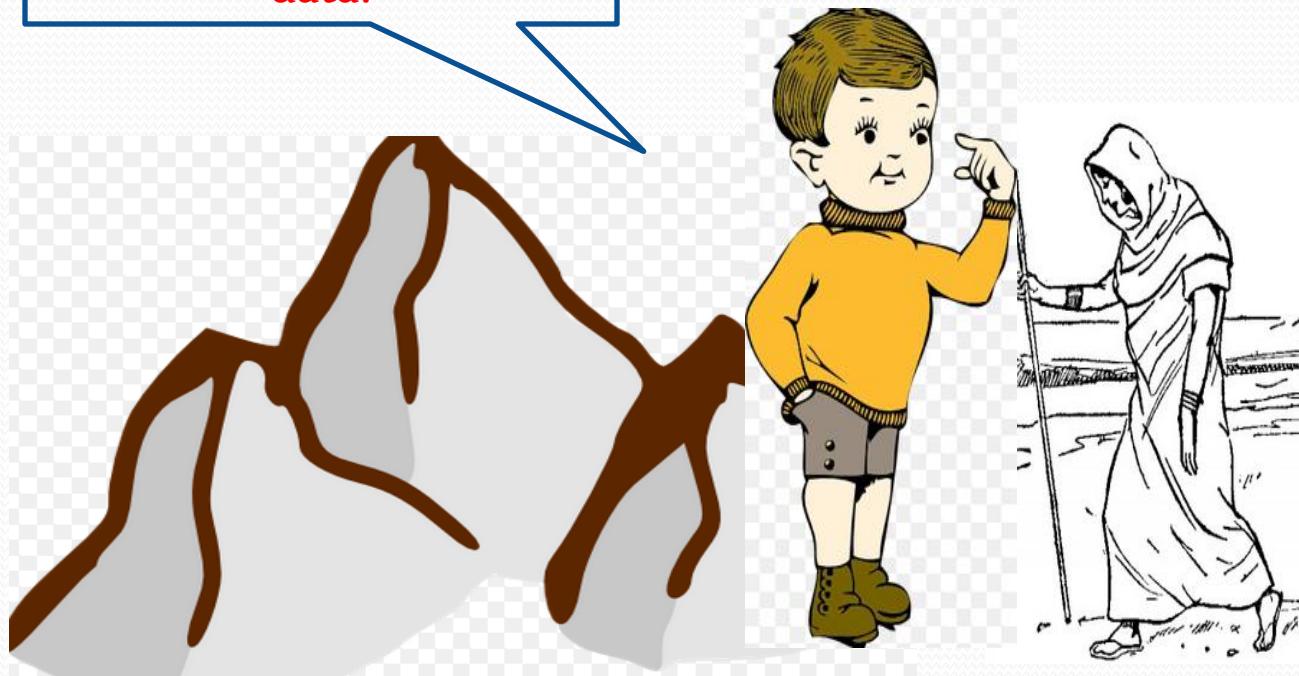


Solution



Solution

Moving the application
would be much faster than
data.



Don't move the data anymore, but run or execute the application on the machine where data is residing

Solution

- Achieving Data Localization
 - Moving the application to the place where the data is residing OR
 - Making data local to the application

Challenges in Traditional Application- Part2

- Efficiency and performance of any application is determined by how fast data can be read
- Traditionally primary motive was to increase the processing capacity of the machine
 - Faster processor
 - More RAM
 - Less data and complex computation is done on data

Statistics – Part 2

- How data is read ?
 - Line by Line reading
 - Depends on seek rate and disc latency

Average Data Transfer rate = 75MB/sec

Total Time to read 100GB = 22 min

Total time to read 1TB = 3 hours

How much time you take to sort 1TB of data??

Observation

- Large amount of data takes lot of time to read
- RAM of the machine also a bottleneck

Summary

- Storage is problem
 - Cannot store large amount of data
 - Upgrading the hard disk will also not solve the problem (Hardware limitation)
- Performance degradation
 - Upgrading RAM will not solve the problem (Hardware limitation)
- Reading
 - Larger data requires larger time to read

Solution Approach

Big
Data
(3TB)



What are the various options to store?

Compression

Upgrading hard disk

Solution Approach

Big
Data
(3TB)



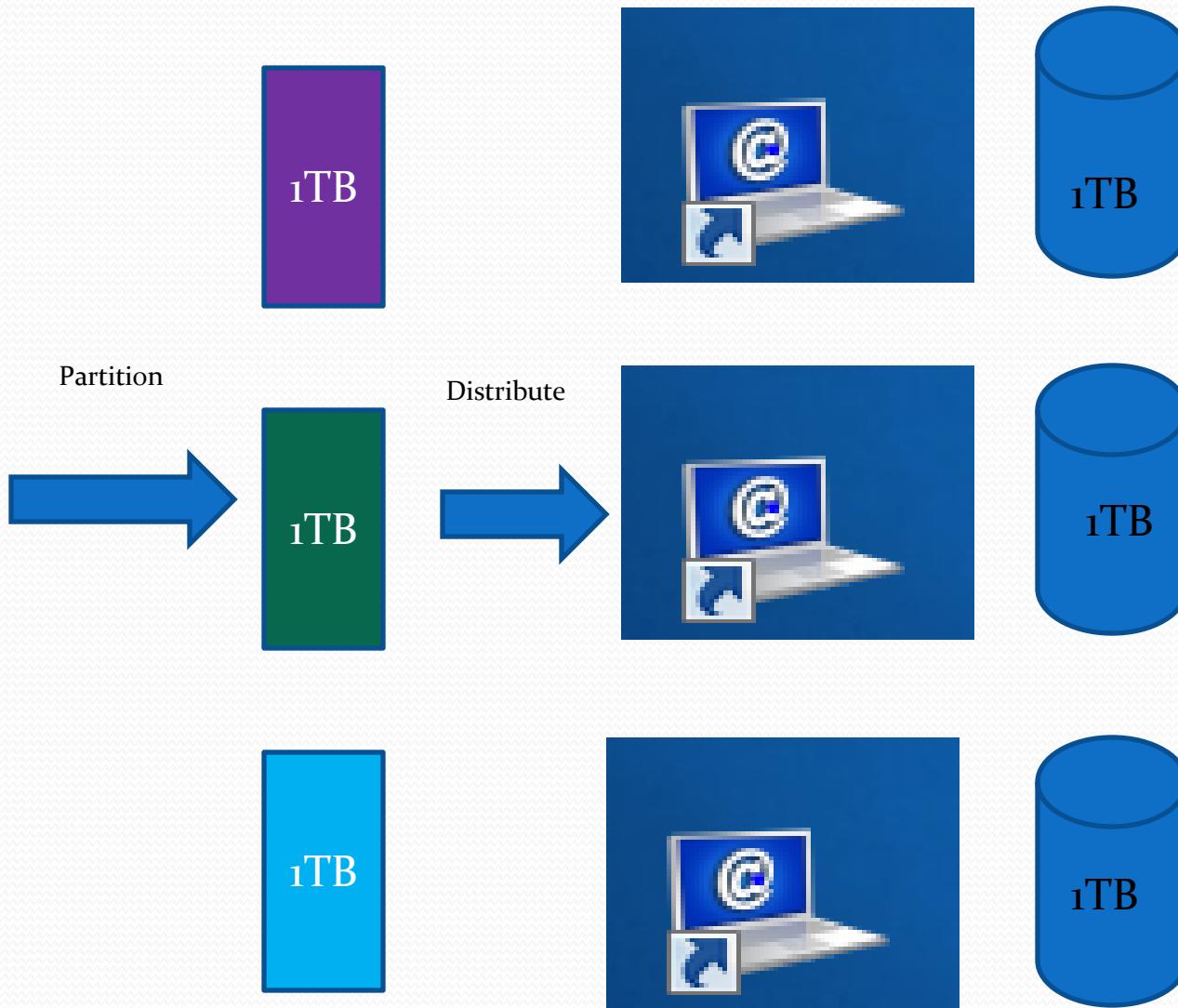
Solution Approach



Partition



Solution Approach



Solution Approach

Big
Data
(3TB)

Don't you think we have
achieved scaling in
storage?

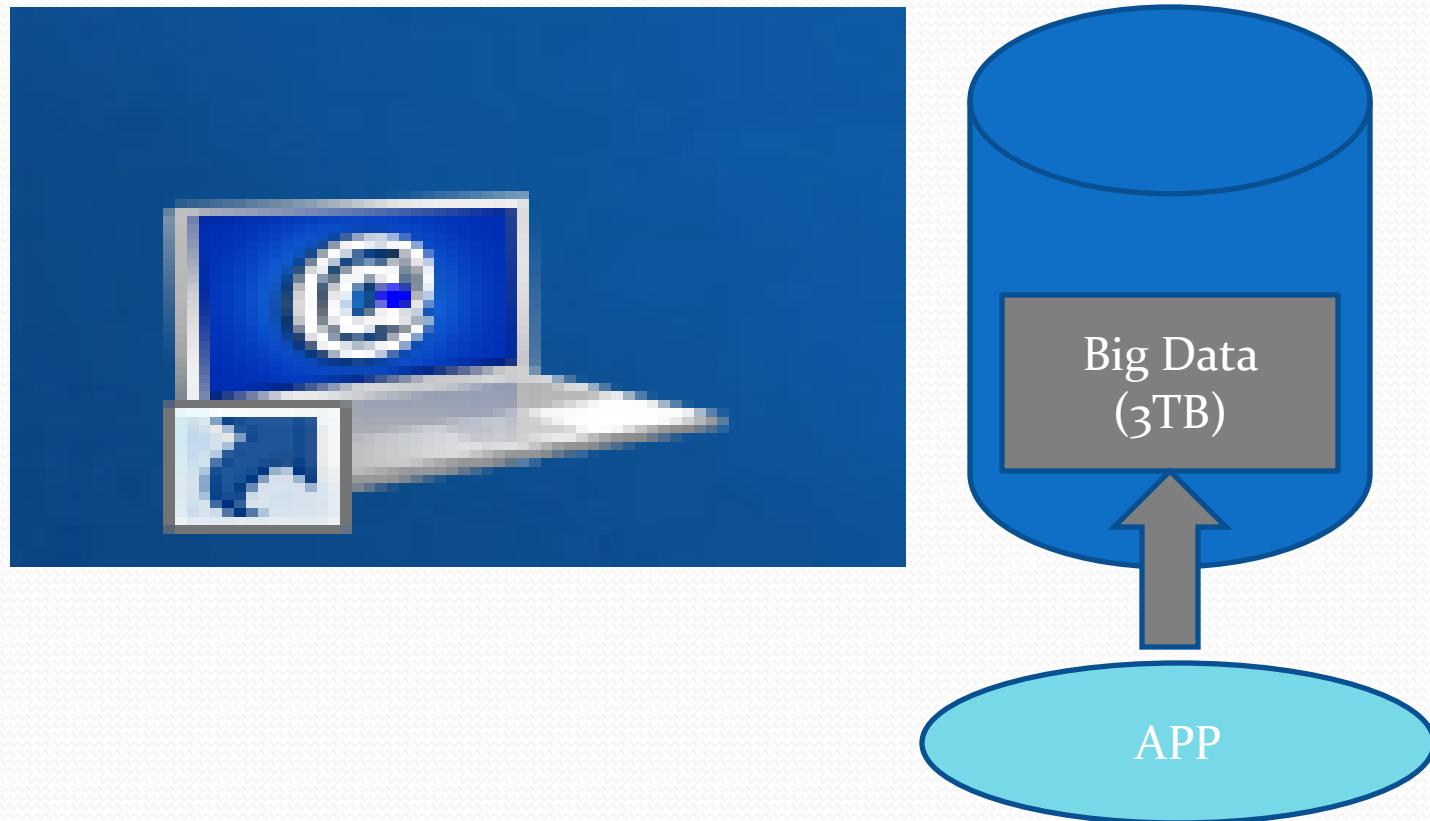


What about
performance?



Solution Approach

- Traditionally

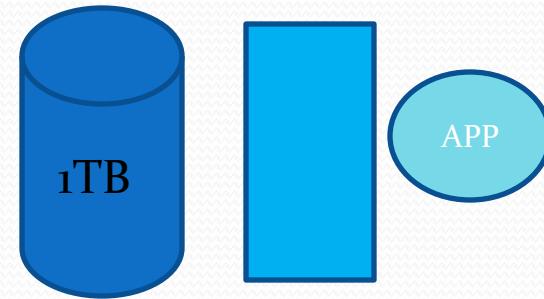
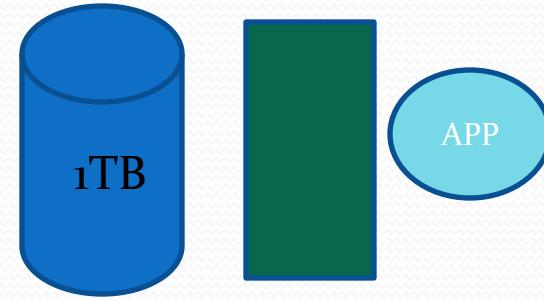
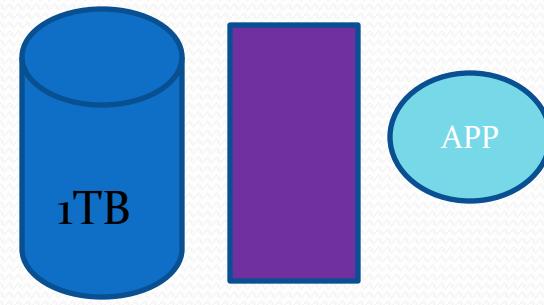


Solution Approach

Big Data (3TB)



Applications running parallelly and closer to the data



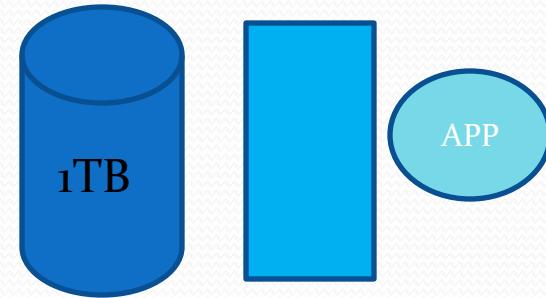
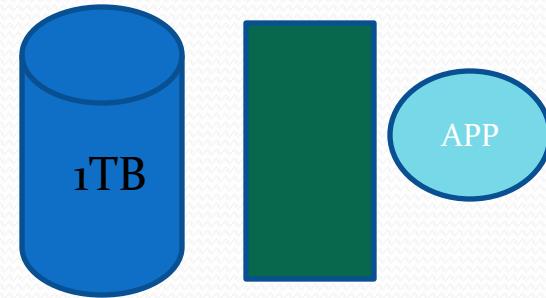
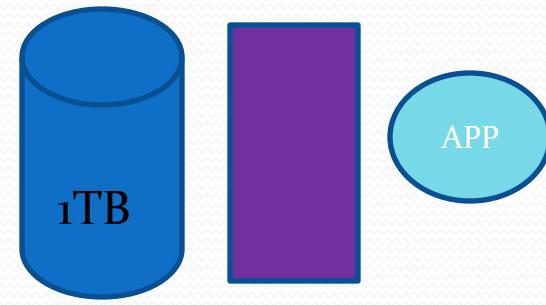
Solution Approach

Big
Data
(3TB)

Don't you think we have
achieved scaling in
performance?



Applications running parallelly and closer to the data



Solution Approach

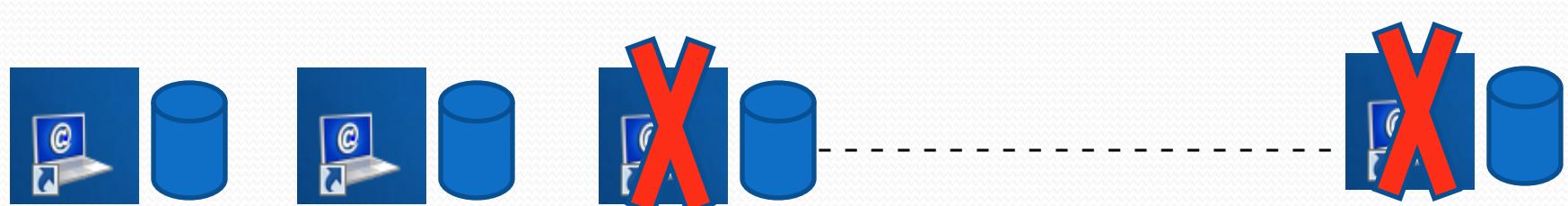
- Distributed Framework
 - Storing the data across several machine
 - Performing computation parallelly across several machines
- But we need some more enhancements in the existing solution to make it more robust and effective

New Approach - Requirements

- Supporting Partial failures
- Recoverability
- Data Availability
- Consistency
- Data Reliability
- Upgrading

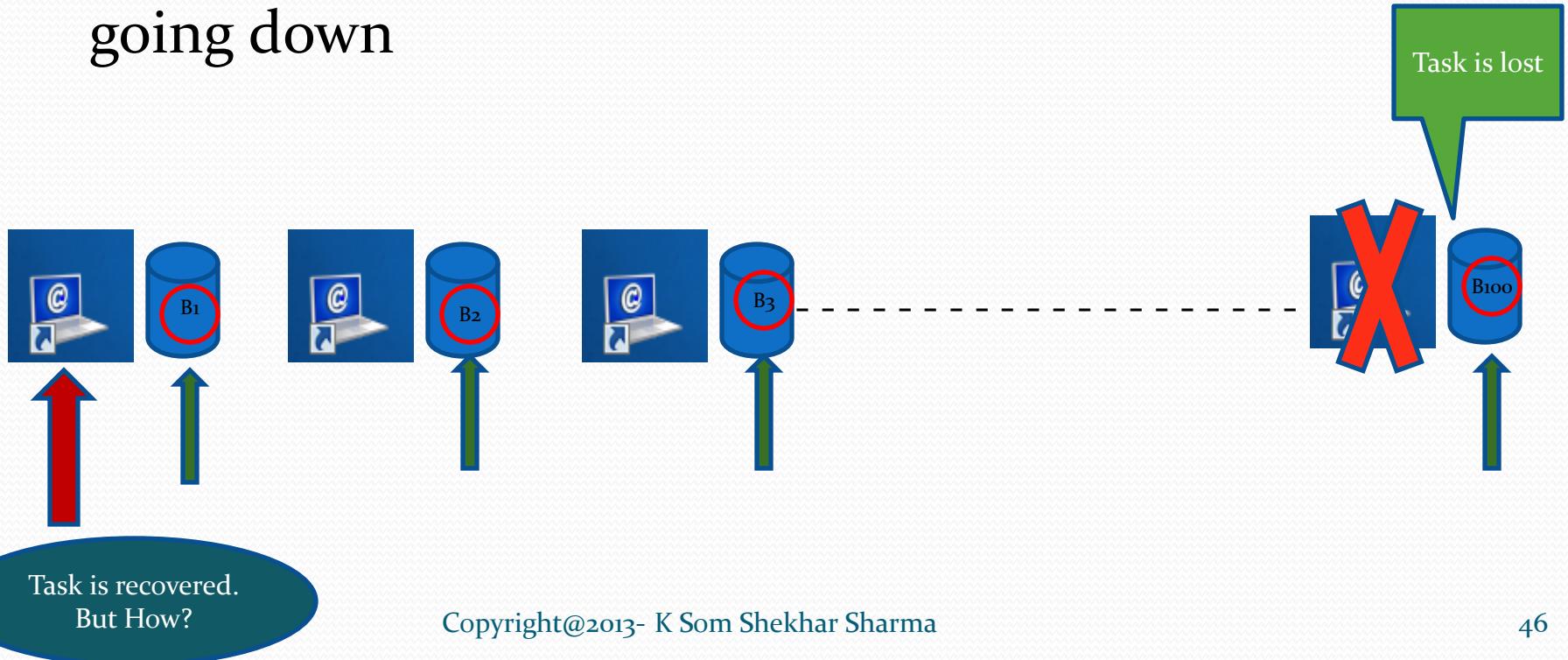
Supporting partial failures

- Should not shut down entire system if few machines are down
 - Should result into graceful degradation of performance
 - Expect delayed completion of Job rather than abrupt disruption in the service



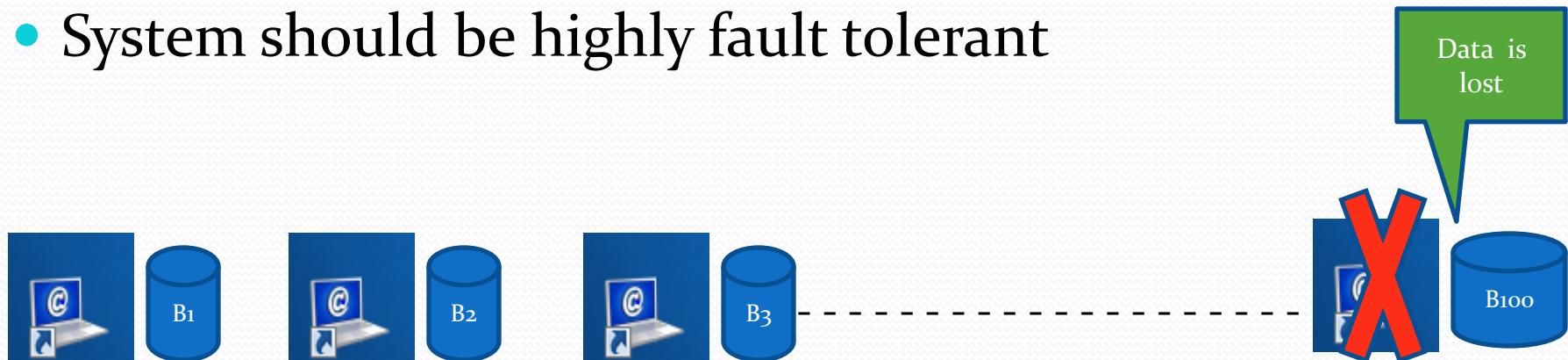
Recoverability

- If machines/components fails, task should be taken up by other working components
- Job execution must continue even if the machines are going down



Data Availability

- Failing of machines/components should not result into loss of data
- System should be highly fault tolerant



Keep Replica (copy or back up) of data

Consistency

- If machines/components are failing the outcome of the job should not be affected



5 machines are working to process a job

Consistency

- If machines/components are failing the outcome of the job should not be affected



4 machines are working to process the same Job, but result must be the same

Consistency

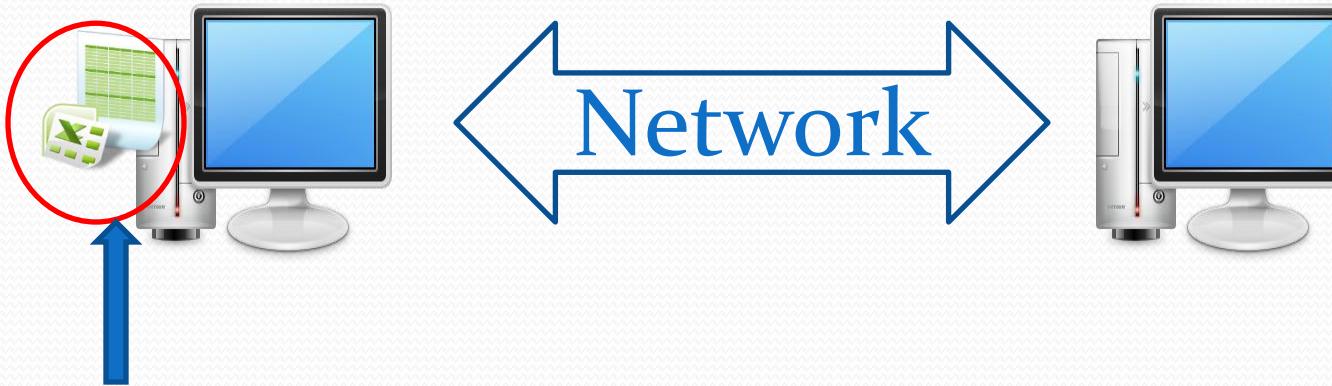
- If machines/components are failing the outcome of the job should not be affected



- 3 machines are working to process the same Job, but result must be the same
- The outcome of the job must always be the same, irrespective how many machines are working to accomplish the result

Data Reliability

- Data integrity and correctness should be at place



- You don't want to work with corrupted data
- Must have some kind of check sum sequence like MD5,SHA,CRC to verify the integrity of the data

Upgrading

- Adding more machines should not require full restart of the system
 - Should be able to add to existing system gracefully and participate in job execution

Upgrading cont'd



- We would like to add one more machine to the existing cluster.

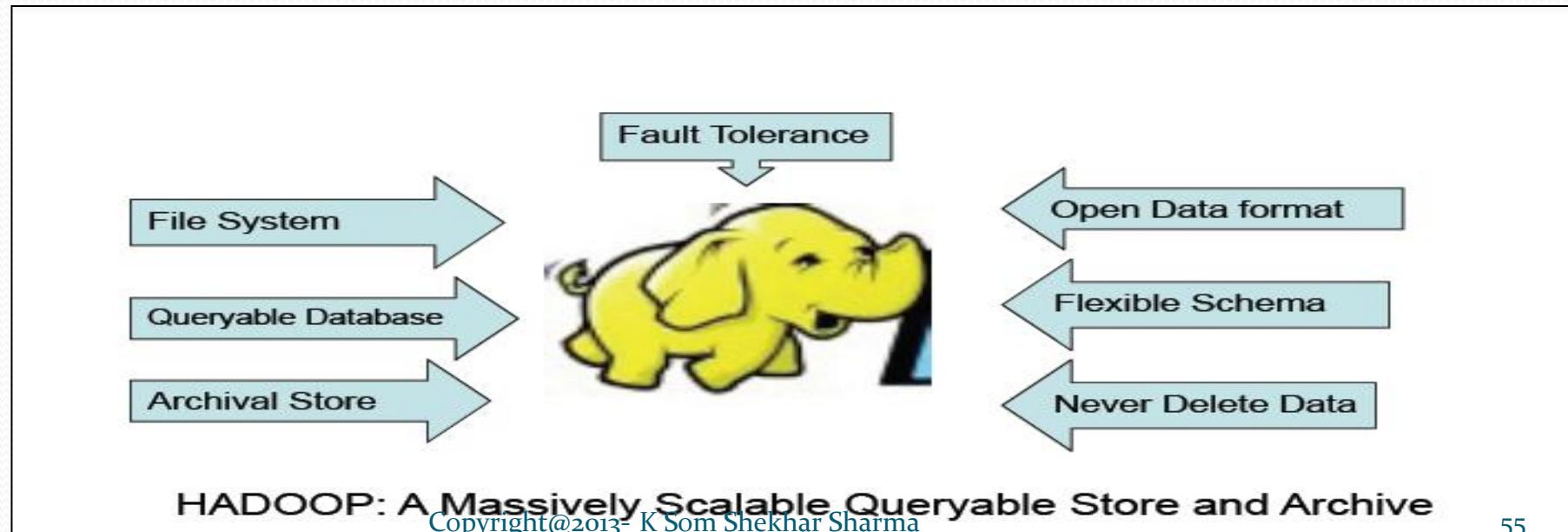
Upgrading cont'd



- The new machine should be able to add to the existing cluster without any disruption in the service.
- Newly added machine should get the data and also should participate in job execution

Introducing Hadoop

- Distributed framework that provides scaling in :
 - Storage
 - Performance
 - IO Bandwidth



Brief History of Hadoop

- **Dec 2004** - Google GFS paper published
- **July 2005** - Nutch uses MapReduce
- **Feb 2006** - Starts as a Lucene subproject
- **Apr 2007** - Yahoo! on 1000-node cluster
- **Jan 2008** - An Apache Top Level Project
- **Jul 2008** - A 4000 node test cluster
- **May 2009** - Hadoop sorts Petabyte in 17 hours

What makes Hadoop special?

- No high end or expensive systems are required
 - Built on commodity hardwares
 - Can run on your machine !
- Can run on Linux, Mac OS/X, Windows, Solaris
 - No discrimination as its written in java
- Fault tolerant system
 - Execution of the job continues even if nodes are failing
 - It accepts failure as part of system.

What makes Hadoop special?

- Highly reliable and efficient storage system
- In built intelligence to speed up the application
 - Speculative execution
- Fit for lot of applications:
 - Web log processing
 - Page Indexing, page ranking
 - Complex event processing

Features of Hadoop

- Partition, replicate and distributes the data
 - Data availability, consistency
- Performs Computation closer to the data
 - Data Localization
- Performs computation across several hosts
 - MapReduce framework

Partitions, Replicate and Distributes

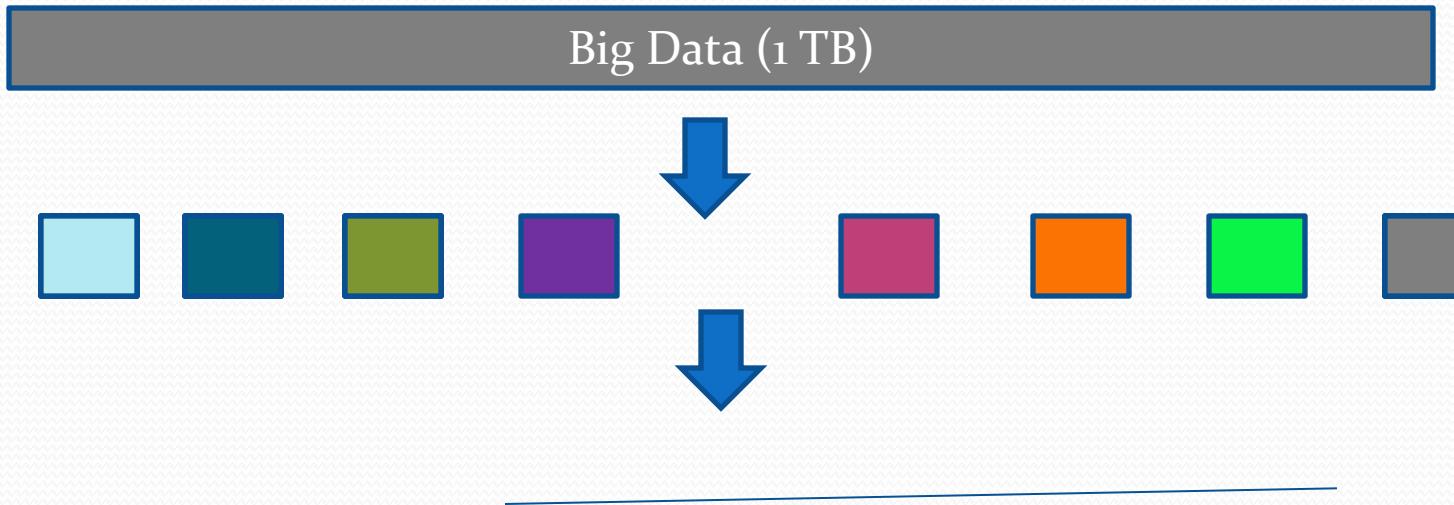
Big Data (1 TB)

Partitions

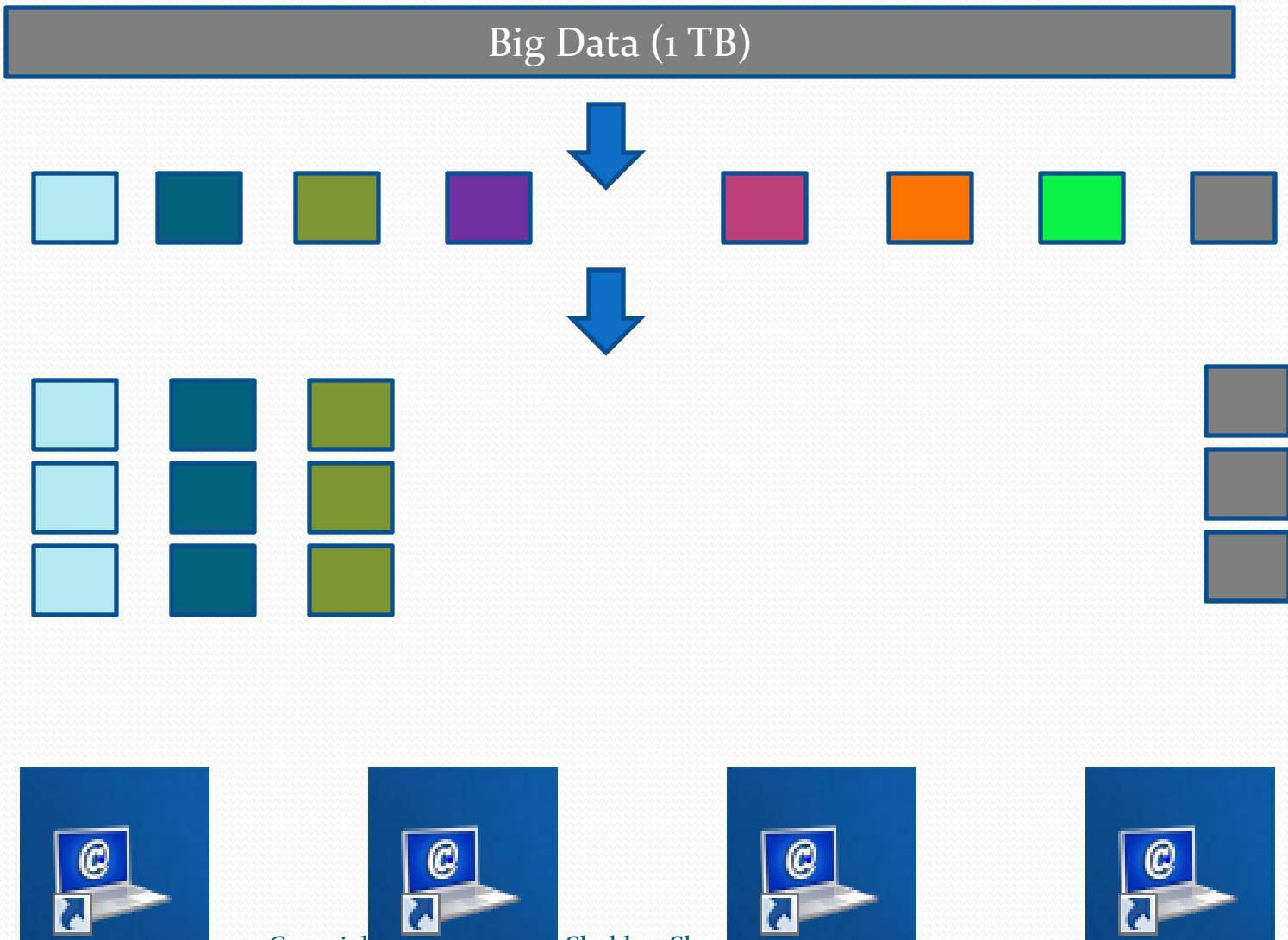
Big Data (1 TB)



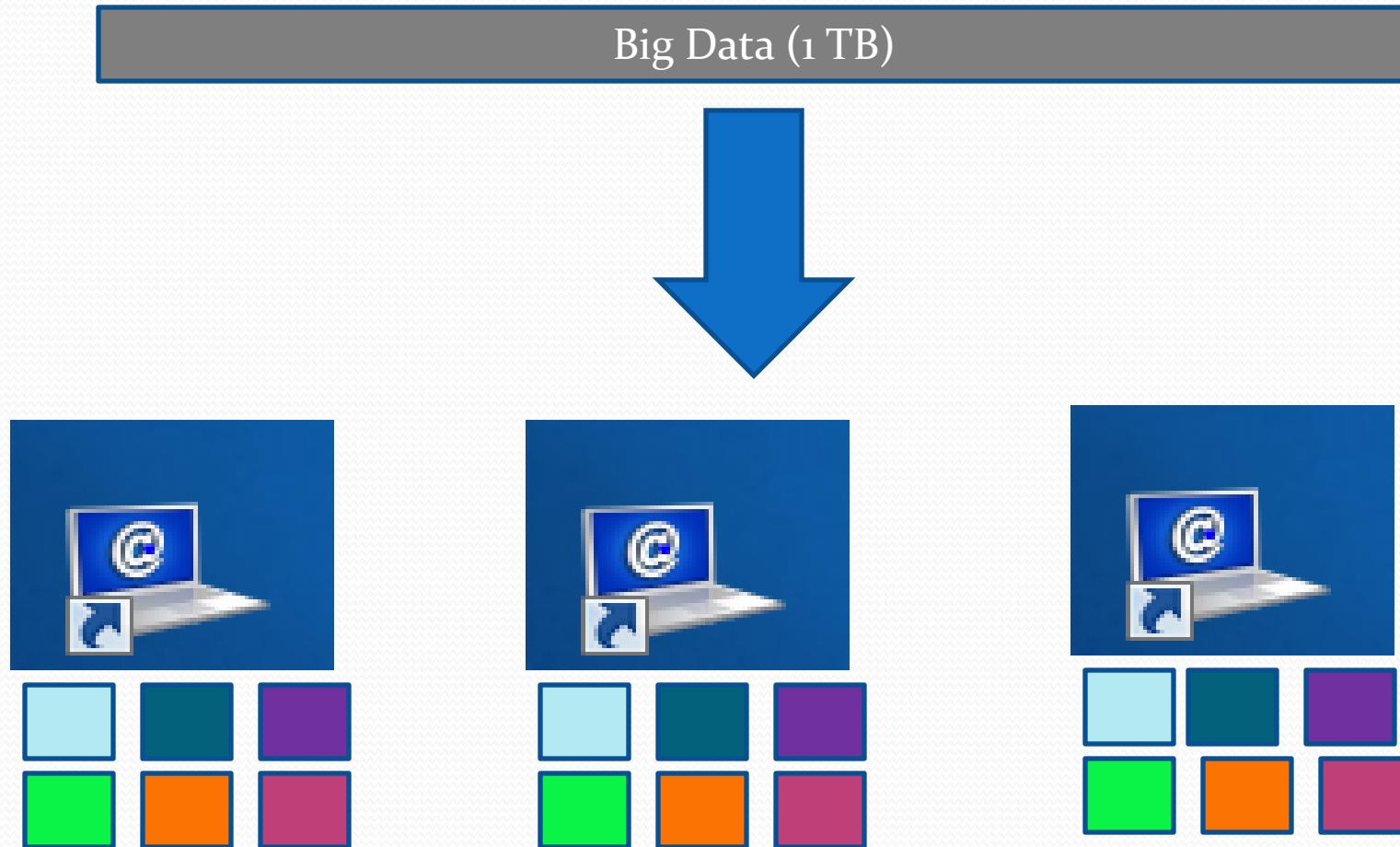
Replication



Distribution

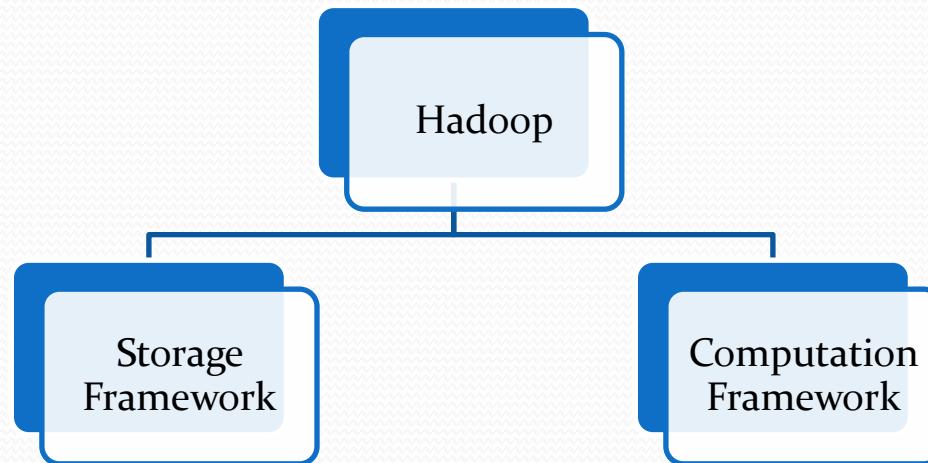


Partitions, Replicate and Distributes



Hadoop Components

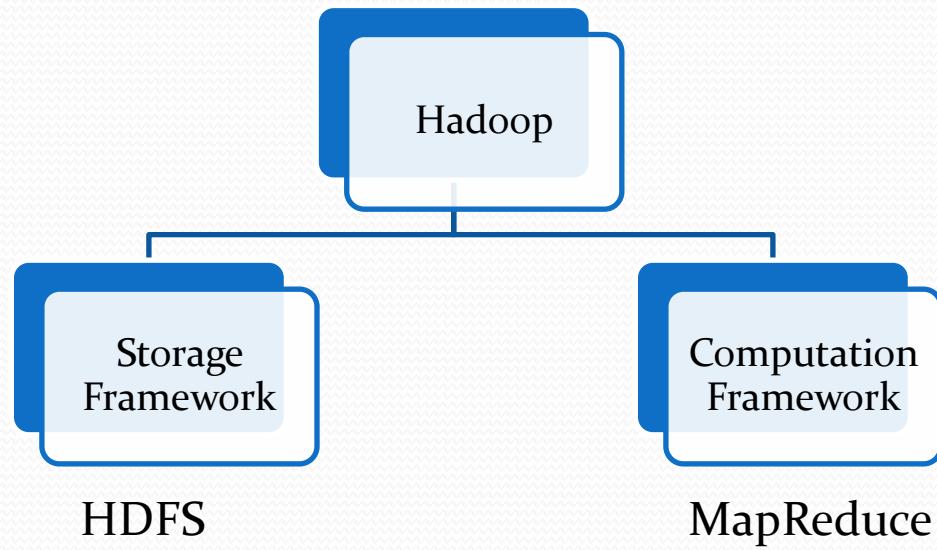
- Hadoop is bundled with two independent components



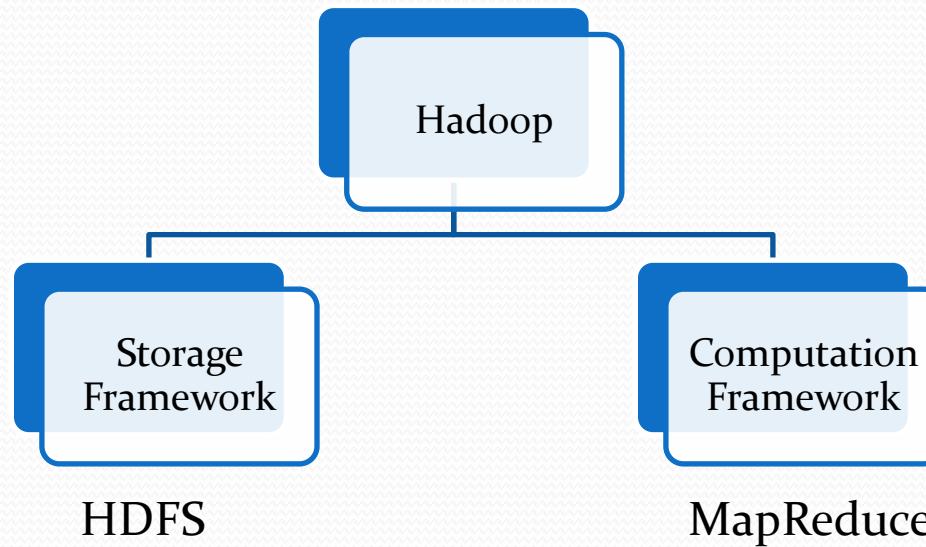
Hadoop can be used

- For storage only
- For computation only
- Both

Hadoop Components



Hadoop Components



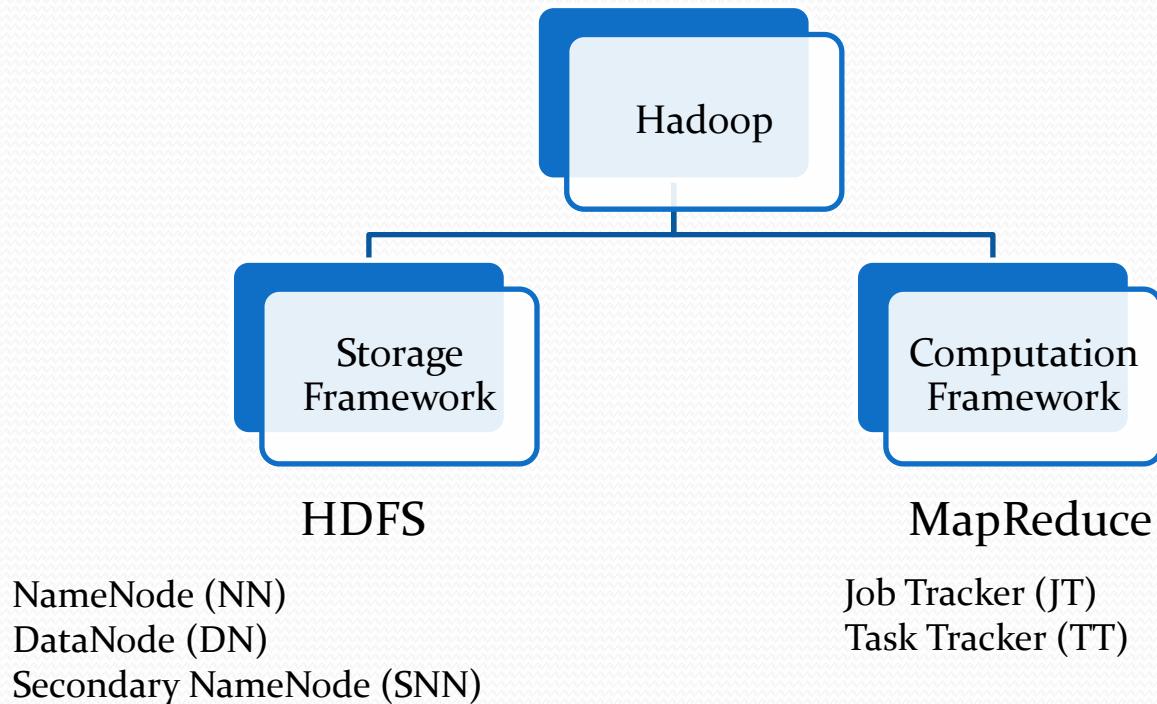
Scaling in terms of

- Storage
- IO bandwidth

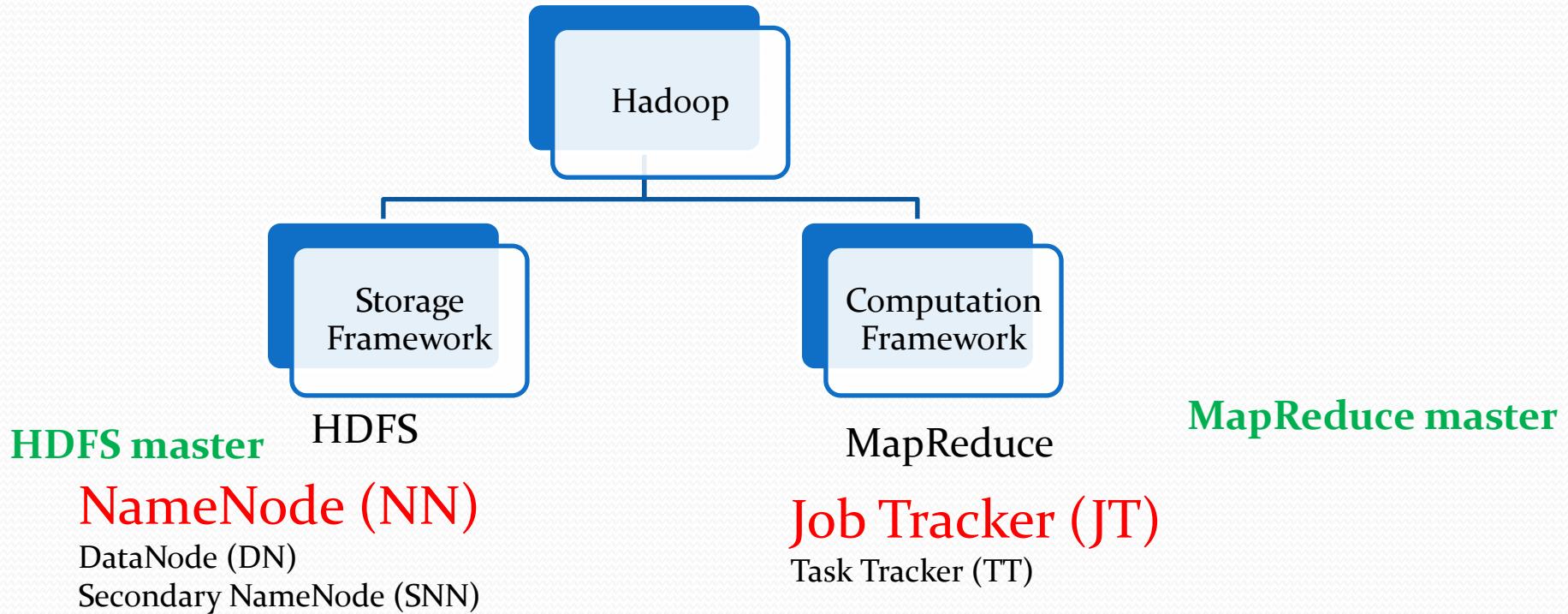
Scaling in terms of

- performance

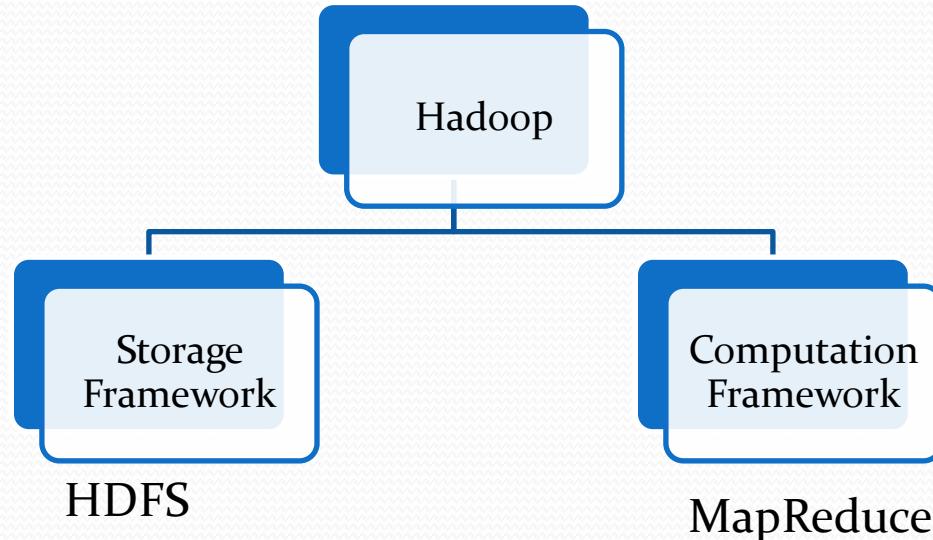
Hadoop process



Hadoop process



Hadoop process



NameNode (NN)

DataNode (DN)

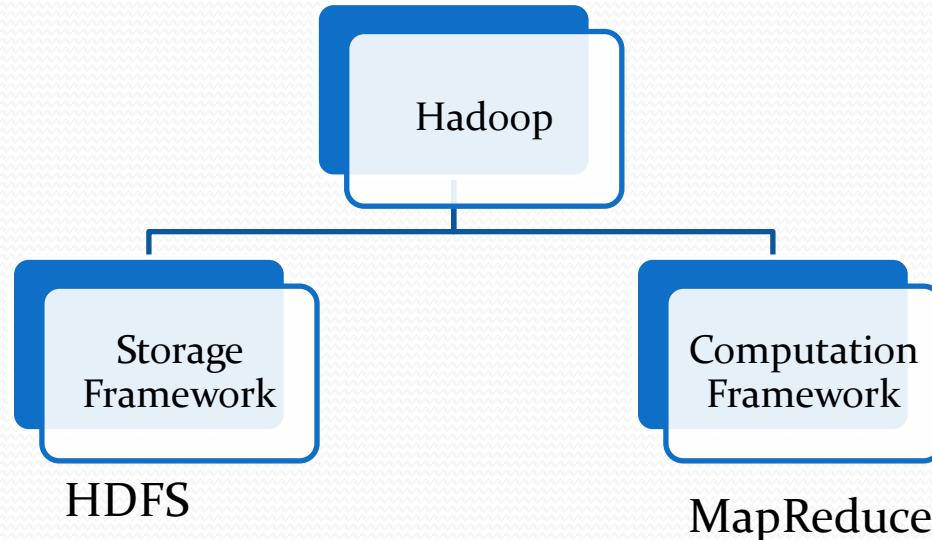
Secondary NameNode (SNN)

Job Tracker (JT)

Task Tracker (TT)

NN is HDFS master
JT is MapReduce master

Hadoop process



NameNode (NN)

DataNode (DN)

Secondary NameNode (SNN)

Job Tracker (JT)

Task Tracker (TT)

NN and JT are SPOC and SPOF

Hadoop Process cont'd

- Two masters :
 - NameNode
 - If down cannot access HDFS
 - Job tracker
 - If down cannot run MapReduce Job, but still you can access HDFS

Does Hadoop solves every one problem?

- I am DB guy, I am proficient in writing SQL and trying very hard to optimize my queries, but still not able to do so. Moreover I am not Java geek. Will this solve my problem?
 - *Hive*
- Hey, Hadoop is written in Java, and I am purely from C++ back ground, how I can use Hadoop for my big data problems?
 - *Hadoop Pipes*
- I am a statistician and I know only R, how can I write MR jobs in R?
 - *RHIPE (R and Hadoop Integrated Environment)*
- Well how about Python, Scala, Ruby, etc programmers? Does Hadoop support all these?
 - *Hadoop Streaming*

RDBMS and Hadoop

- Hadoop is not a data base
- RDBMS Should not be compared with Hadoop
- RDBMS should be compared with NoSQL or Column Oriented Data base

RDBMS Challenges – Not Suitable for evolving data sets

Scenario : You have 1 million rows with 3 columns

	Col1	Col2	Col3
R1	V ₁₁	V ₂₁	V ₂₁
R2	V ₂₁	V ₂₂	V ₂
-	-	-	-
-	-	-	-
R1million	Vx	Vy	Vz

RDBMS Challenges – Not Suitable for evolving data sets

	Col1	Col2	Col3	Col4	Col5
R1	V ₁₁	V ₂₁	V ₂₁	Null	Null
R2	V ₂₁	V ₂₂	V ₂	Null	Null
-	-	-	-	-	-
-	-	-	-	-	-
R1million	Vx	Vy	Vz	Null	Null
R1.1	V _{1,1}	V _{1.1}	V _{1,1}	V _{1,1}	V _{1,1}
-	-	-	-	-	-
R2million	V _{2,1}				

RDBMS Challenges – Not Suitable for evolving data sets

	Col1	Col2	Col3	Col4	Col5
R1	V ₁₁	V ₂₁	V ₂₁	Null	Null
R2	V ₂₁	V ₂₂	V ₂	Null	Null
-	-	-	-	-	-
-	-	-	-	-	-
R1million	Vx	Vy	Vz	Null	Null
R1.1	V _{1,1}				
-	-	-	-	-	-
R2million	V _{2,1}				

How many nulls?

RDBMS Challenges – Not Suitable for evolving data sets

	Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9
R1	V11	V21	V21	Null	Null	Null	Null	Null	Null
R2	V21	V22	V2	Null	Null	Null	Null	Null	Null
-	-	-	-		-	-	-	-	-
-	-	-	-	Null	Null	-	-	-	-
R1million	Vx	Vy	Vz			-	-	-	-
R1.1	V1,1	V1.1	V1.1	V1.1	V1.1	-	-	-	-
-	-	-	-	-	-	-	-	-	-
R2million	V2,1	V2,1	V2,1	V2,1	V2,1	Null	Null	Null	Null

Now How many nulls?

Is there any way we can store only those columns which has value so as to avoid storing null?

RDBMS Challenge-Row Oriented In Nature

	Col1	Col2	Col3
R1	V ₁₁	V ₂₁	V ₂₁
R2	V ₂₁	V ₂₂	V ₂
-	-	-	-
-	-	-	-
R1million	Vx	Vy	Vz

Select * from T where Col2 = Vx;

It does a full table scanning

Is there any way we can avoid full table scanning?

RDBMS Challenges- Driven by Relational Math

- Driven by relational math

Is there any schema less database?

Downside of RDBMS

- Rigid Schema
 - Once schema is defined it cannot be changed
 - Any new additions in the column requires new schema to be created
 - Leads to lot of nulls being stored in the data base
 - Cannot add columns at run time
- Row Oriented in nature
 - Rows and columns are tightly bound together
 - Firing a simple query “*select * from myTable where colName='foo'*” requires to do the full table scanning

NoSQL DataBases

- Invert RDBMS upside down
- Column family concept (No Rows)
 - One column family can consist of various columns
 - New columns can be added at run time
 - Nulls can be avoided
 - Schema can be changed
- Meta Information helps to locate the data
 - No table scanning

Challenges in Hadoop-Security

- Poor security mechanism
- Uses “*whoami*” command
- Cluster should be behind firewall
- Already integrated with “Kerberos” but very trickier

Challenges in Hadoop- Deployment

- Manual installation would be very time consuming
 - What if you want to run 1000+ Hadoop nodes
- Can use puppet scripting
 - Complex
- Can use Cloudera Manager
 - Free edition allows you to install Hadoop till 50 nodes

Challenges in Hadoop- Maintenance

- Why and how the machines are failing?
- Always resort to the log file if something goes wrong
 - Should not happen that log file size is greater than the data

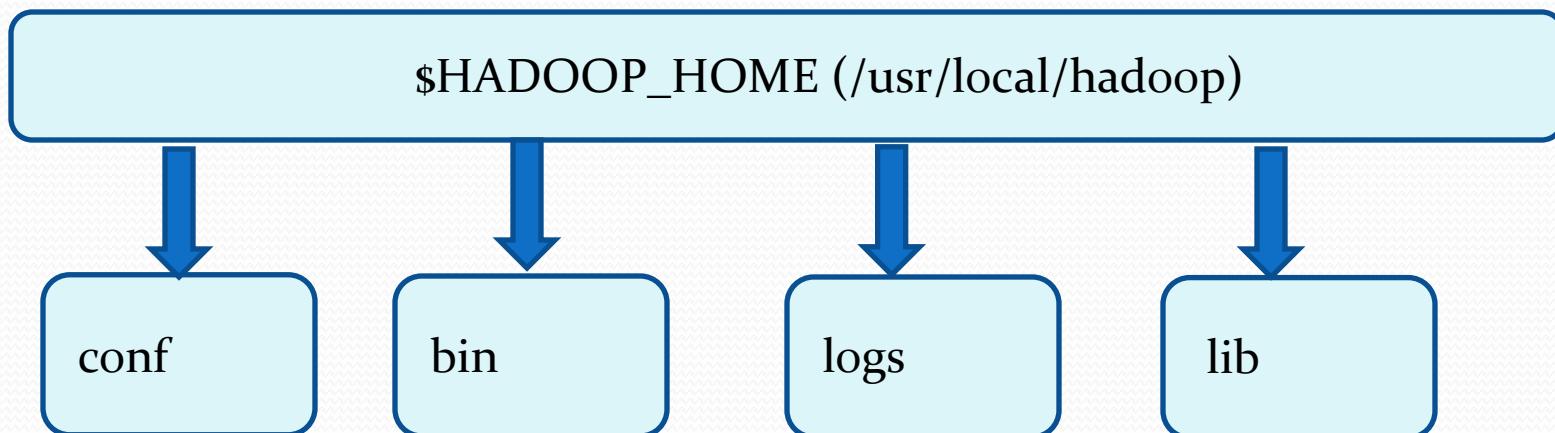
Challenges in Hadoop- Debugging

- Tasks run on separate JVM on Hadoop cluster
 - Difficult to debug through Eclipse

MODULE 2

Directory Structure of Hadoop

Hadoop Distribution- hadoop-1.0.3



- mapred-site.xml
 - core-site.xml
 - hdfs-site.xml
 - masters
 - slaves
 - hadoop-env.sh
- start-all.sh
 - stop-all.sh
 - start-dfs.sh, etc
- All the log files for all the corresponding process will be created here
- All the 3rd party jar files are present
 - You will be requiring while working with HDFS API

conf Directory

- All the hadoop related properties needs to go into one of these files
 - mapred-site.xml
 - All the map reduce related properties should be specified
 - Property names should have the prefix of “mapred/mapreduce”
 - mapred.job.tracker, mapred.reduce.tasks etc
 - hdfs-site.xml
 - All the HDFS related properties should be specified
 - Property names should have the prefix of dfs
 - dfs.block.size, dfs.replication, dfs.hosts etc
 - core-site.xml
 - All the core related properties should be specified
 - Property names should not have the prefix of mapred or dfs
 - hadoop.tmp.dir, fs.default.name, io.sort.mb

bin directory

- Place for all the executable files
- You will be running following executable files very often
 - start-all.sh (For starting the Hadoop cluster)
 - stop-all.sh (For stopping the Hadoop cluster)
 - start-dfs.sh
 - stop-dfs.sh
 - start-mapred.sh
 - stop-mapred.sh

logs Directory

- Place for all the logs
- Log files will be created for every process running on Hadoop cluster
 - NameNode logs
 - DataNode logs
 - Secondary NameNode logs
 - Job Tracker logs
 - Task Tracker logs

MODULE 3

Single Node (Pseudo Mode) Hadoop Set up

Introduction

Pseudo Mode / Single Node



Multi Node



Installation Steps

- Pre-requisites
- Java Installation
- Creating dedicated user for hadoop
- Password-less ssh key creation
- Configuring Hadoop
- Starting Hadoop cluster
- MapReduce and NameNode UI
- Stopping Hadoop Cluster

Note

- The installation steps are provided for CentOS 5.5 or greater
- Installation steps are for 32 bit OS
- All the commands are marked in blue and are in *italics*
- Assuming a user by name “*training*” is present and this user is performing the installation steps

Note

- Hadoop follows master-slave model
- There can be only 1 master and several slaves before hadoop-2.x version
- Hadoop-2.x versions
 - HDFS-HA mitigating single point of failure of NN
 - YARN mitigating single point of failure of JT
- In pseudo mode, Single Node is acting both as master and slave
- Master machine is also referred to as NameNode machine
 - The machine which runs name node process
- Slave machines are also referred to as DataNode machine
 - Machines which data node process

Pre-requisites

- Edit the file /etc/sysconfig/selinux
 - Change the property of SELINUX from enforcing to disabled
 - You need to be root user to perform this operation
- Install ssh
 - *yum install open-sshserver open-sshclient*
 - *chkconfig sshd on*
 - *service sshd start*

Installing Java

- Download Sun JDK (≥ 1.6) 32 bit for linux
- Download the .tar.gz file
- Follow the steps to install Java
 - *tar -zxf jdk.x.x.x.tar.gz*
 - Above command will create a directory from where you ran the command
 - Copy the Path of directory (Full Path)
 - Create an environment variable JAVA_HOME in .bashrc file (This file is present inside your user's home directory)

Installing Java cont'd

- Open */home/training/.bashrc* file

```
export JAVA_HOME=PATH_TO_YOUR_JAVA_HOME
```

```
export PATH=$JAVA_HOME/bin:$PATH
```

- Close the file and run the command *source .bashrc*

Verifying Java Installation

- Run `java -version`
 - This command should show the Sun JDK version

```
Applications Places System dev@localhost:~  
File Edit View Terminal Tabs Help  
[dev@localhost ~]$ java -version  
java version "1.6.0_31"  
Java(TM) SE Runtime Environment (build 1.6.0_31-b04)  
Java HotSpot(TM) Client VM (build 20.6-b01, mixed mode, sharing)  
[dev@localhost ~]$
```

1.6.0_31

major version minor version

Copyright@2013. K. Som Shekhar Sharma

Disabling IPV6

- Hadoop works only on IPv4 enabled machine not on IPv6.
- Run the following command to check
 - `cat /proc/sys/net/ipv6/conf/all/disable_ipv6`
 - The value of 0 indicates that ipv6 is disabled
- For disabling IPv6, edit the file **/etc/sysctl.conf**

```
net.ipv6.conf.all.disable_ipv6 = 1  
net.ipv6.conf.default.disable_ipv6 = 1  
net.ipv6.conf.lo.disable_ipv6 = 1
```

Why SSH?

- As such ssh is not required in hadoop
- SSH in hadoop is required to just start and stop the process

Why SSH?

India



>>I need to start the process
in these machine

>>What are the options?

USA



Why SSH?

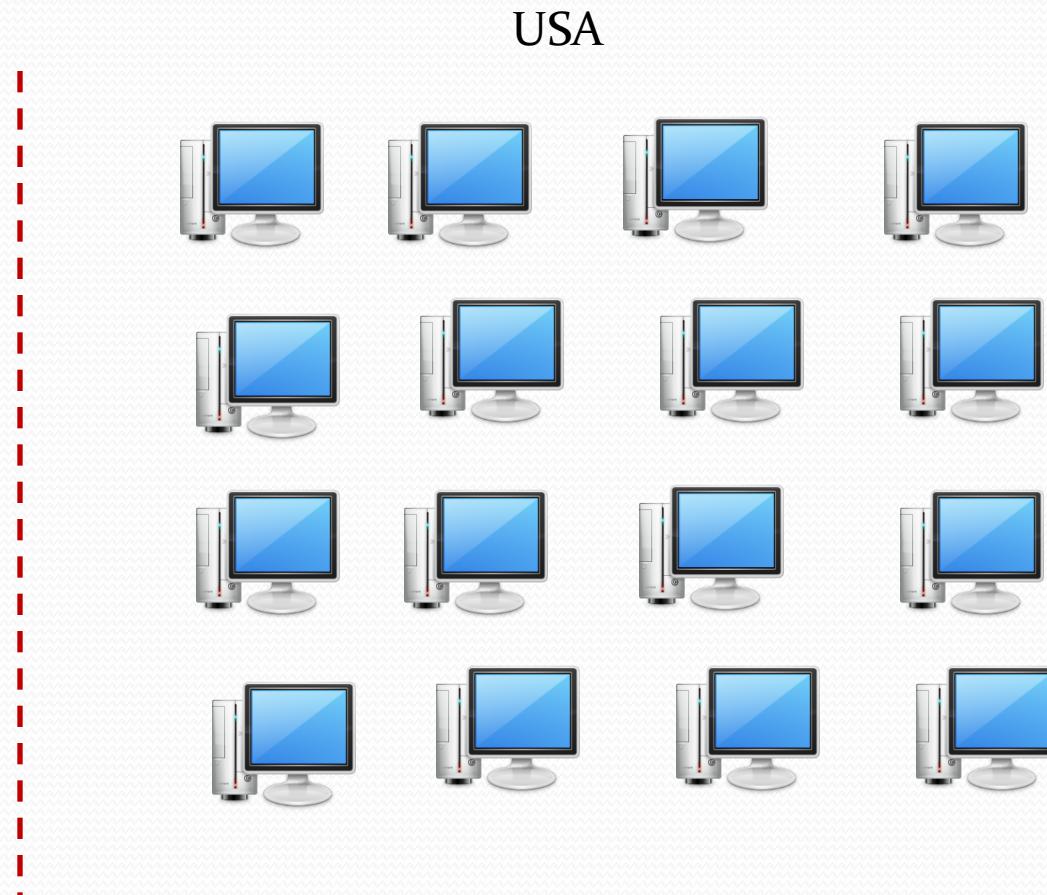


Option 1:

```
ssh machine1:  
  username:  
  password:  
  start the process
```

```
ssh machine2:  
  username:  
  password:  
  start the process
```

What is the problem?



Why SSH?



Option 1:

```
ssh machine1:  
  username:  
  password:  
  start the process
```

```
ssh machine2:  
  username:  
  password:  
  start the process
```

What is the problem?
Cumbersome activity of entering
the user name and password for all
the machines



Why SSH?



Option 2: Dedicated user in all the machines.

ssh machine1:
password:
start the process

ssh machine2:
password:
start the process

What is the problem?
Cumbersome activity of entering
the password for all the machines

USA



Why SSH?



Option 3:
Should be able to do the
communication in password less
manner



USA



Configuring passwordless SSH key

- Run the following command to create a password less key
 - `ssh-keygen -t rsa -P “”`
 - The above command will create two files under `/home/training/.ssh` folder
 - `id_rsa` (private key)
 - `id_rsa.pub` (public key)
- Copy the contents of public key to a file `authorized_keys`
 - `cat /home/training/.ssh/id_rsa.pub >> /home/training/.ssh/authorized_keys`
- Change the permission of the file to 755 or 650
 - `chmod 755 /home/training/.ssh/authorized_keys`
- The permission set should not be too open (777) or too restrictive (660,666). Else communication cannot be done in passwordless manner

Verification of passwordless ssh

- Run *ssh localhost*
 - Above command should not ask you password and you are in localhost user instead of training user
- Run *exit* to return to training user

Configuring Hadoop

- Download Hadoop tar ball and extract it
 - *tar -zxf hadoop-1.0.3.tar.gz*
 - Above command will create a directory which is Hadoop's home directory
 - Copy the path of the directory and edit .bashrc file

```
export HADOOP_HOME=/home/training/hadoop-1.0.3
```

```
export PATH=$PATH:$HADOOP_HOME/bin
```

Edit \$HADOOP_HOME/conf/mapred-site.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>
  <name>mapred.job.tracker</name>
  <value>localhost:54311</value>
</property>

</configuration>
```

This property specifies the address of Job tracker

Job tracker is running on localhost and listening on port 54311

Edit \$HADOOP_HOME/conf/hdfs-site.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
```

```
<!-- Put site-specific property overrides in this file. -->
```

```
<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>

<property>
<name>dfs.block.size</name>
<value>67108864</value>
</property>
</configuration>
```

Number of replica per block

In pseudo mode set as 1.

What if we set more than 1? Is it going to store those number of replica?

Specifies the chunk size into which a given file will be broken

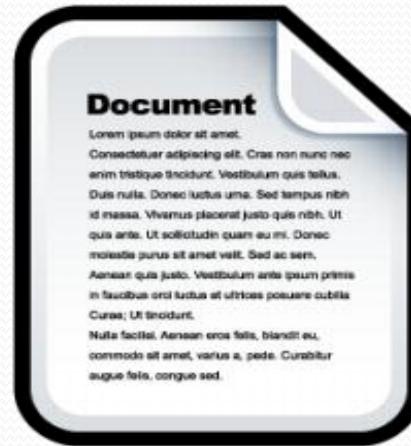
Value is 64MB In bytes ($64 * 1024 * 1024$)

Block Size

File Size = 128MB

Block Size = 64MB

Replication Factor = 1

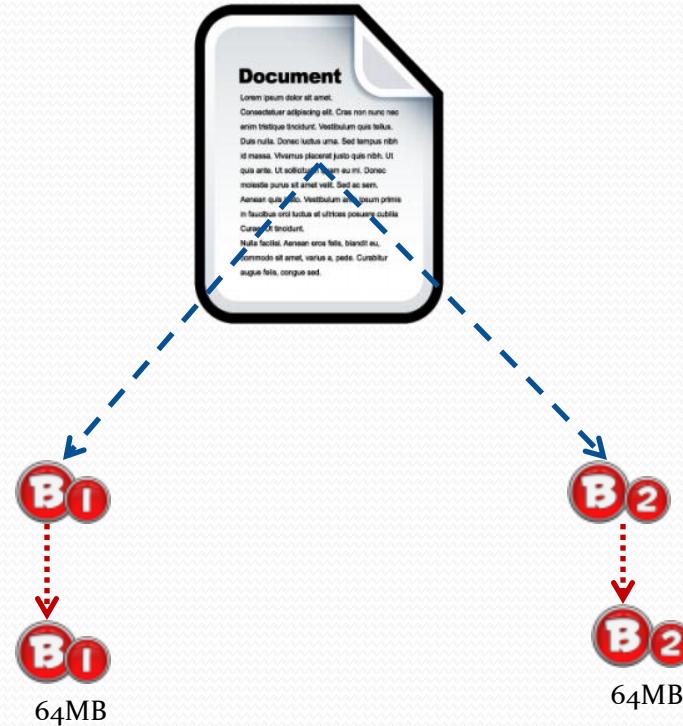


Block Size

File Size = 128MB

Block Size = 64MB

Replication Factor = 1



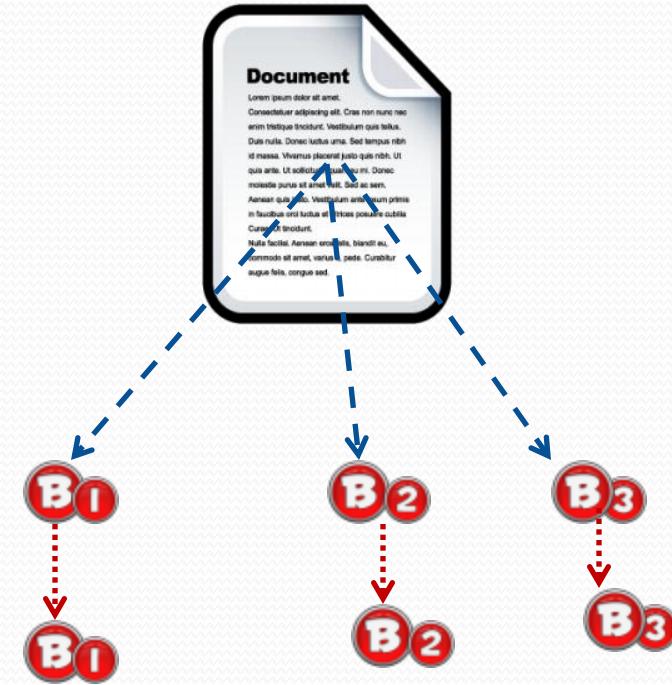
Replica are physically stored with slave machines or data node machines

Block Size

File Size = 130MB

Block Size = 64MB

Replication Factor = 1

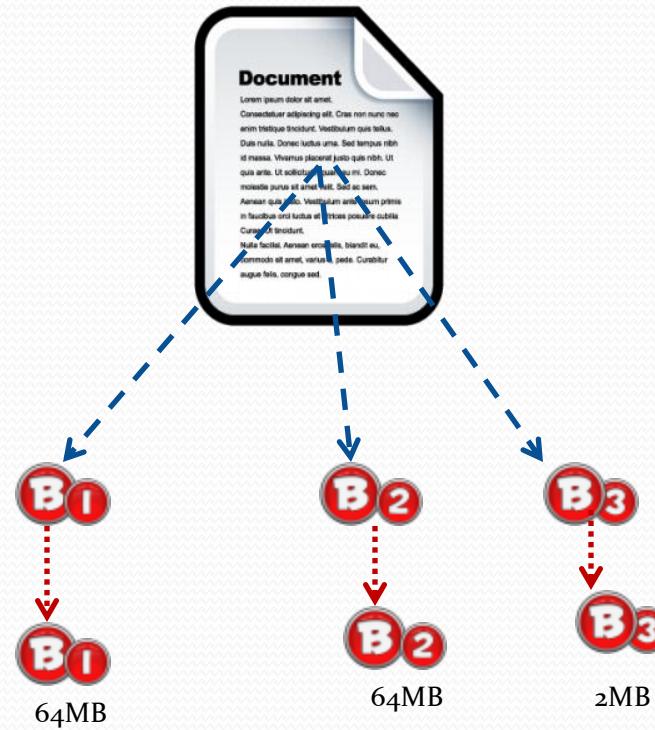


Block Size

File Size = 130MB

Block Size = 64MB

Replication Factor = 1



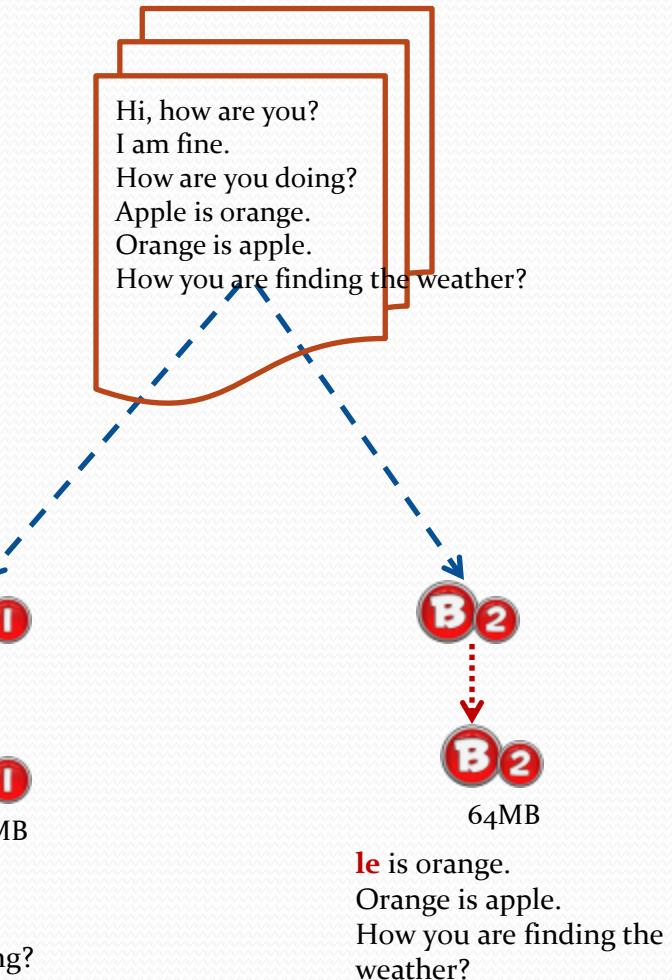
Don't consider 62MB is wasted. When it is physically stored it occupies 2MB space only

Block Size

File Size = 128MB

Block Size = 64MB

Replication Factor = 1



- Record boundary or line boundary is not respected
- When you are analyzing this data, APPLE should be consider as single word

Edit \$HADOOP_HOME/conf/core-site.xml

```
<?xml version="1.0"?>
<?xmlstylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/home/training/hadoop-temp</value>
    <description>A base for other temporary directories.</description>
  </property>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:54310</value>
  </property>
</configuration>
```

This property specifies the address of NameNode

NameNode is running on localhost and listening on port 54310

hadoop.tmp.dir

- The value of this property is a directory on a local file system
- This directory consist of entire HDFS information
 - If this directory gets deleted or corrupted entire information will be lost
- The default value of this property is /tmp folder on local file system.
 - NOTE /tmp folder is volatile folder, so once system is restarted entire information will be lost
- The value of this property should point to a directory which is not affected by system restart.
- You don't have to create this directory manually.

Edit \$HADOOP_HOME/conf/hadoop-env.sh

```
export JAVA_HOME=PATH_TO_YOUR_JDK_DIRECTORY
```

Note

- No need to change your *masters* and *slaves* file as you are installing Hadoop in pseudo mode / single node
- Default value is localhost in those files.
 - localhost(127.0.0.1) is acting as master as well as slave

Creating HDFS (Hadoop Distributed File System)

- Run the command
 - *hadoop namenode -format*
- This command will create *hadoop-temp* directory
(check hadoop.tmp.dir property in core-site.xml)

Start Hadoop's Processes

- Run the command
 - *start-all.sh*
- The above command will run 5 process
 - NameNode
 - DataNode
 - SecondaryNameNode
 - JobTracker
 - TaskTracker
- Run *jps* to view all the Hadoop's process

Viewing NameNode UI

- In the browser type localhost:50070

The screenshot shows the NameNode UI running in a browser window. The title bar says "NameNode - Windows Photo Viewer". The main content area displays the following information:

NameNode 'localhost.localdomain:54310'

Started: Mon Oct 15 12:23:51 PDT 2012
Version: 1.0.3, r1335192
Compiled: Tue May 8 20:31:25 UTC 2012 by hortonfo
Upgrades: There are no upgrades in progress.

[Browse the filesystem](#) → [Browse HDFS](#)
[Namenode Logs](#)

Cluster Summary

167 files and directories, 81 blocks = 248 total. Heap Size is 31.38 MB / 966.69 MB (3%)

Configured Capacity	:	17.15 GB
DFS Used	:	201.56 MB
Non DFS Used	:	9.88 GB
DFS Remaining	:	7.07 GB
DFS Used%	:	1.15 %

→ *HDFS storage capacity*

Viewing MapReduce UI

- In the browser type localhost:50030

localhost Hadoop Map/Reduce Administration

Quick Li

State: RUNNING

Started: Mon Oct 15 12:24:43 PDT 2012

Version: 1.0.3, r1335192

Compiled: Tue May 8 20:31:25 UTC 2012 by hortonfo

Identifier: 201210151224

Cluster Summary (Heap Size is 15.31 MB/966.69 MB)

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes	Graylisted Nodes
0	0	0	1	0	0	0	0	2	2	4.00	0	0

Hands On

- Open the terminal and start the hadoop process
 - *start-all.sh*
- Run *jps* command to verify whether all the 5 hadoop process are running or not
- Open your browser and check the namenode and mapreduce UI
- In the NameNode UI, browse your HDFS

Hands On

- Stop all the process.
- start-dfs.sh
 - How many process will start?
 - Refresh job tracker UI and Namenode UI
- start-mapred.sh
 - How many process will start?
 - Refresh Job tracker UI and NameNode UI
- stop-all.sh

Hands On

- Running the process individually. Useful for debugging when some process are not running
- Process will run on console
- Open a new terminal to start each of the process
 - hadoop namenode
 - hadoop datanode
 - hadoop secondarynamenode
 - hadoop jobtracker
 - hadoop tasktracker
- Process which are started manually needs to stopped manually. Use **ctrl + c**
- If you want to run the process in back ground press **ctrl + z**
- If they are running in back ground, get the process id by running the **jps** and use “**kill -9 process_id**” command

MODULE 4

Multi Node Hadoop Cluster Set up

Multinode setup



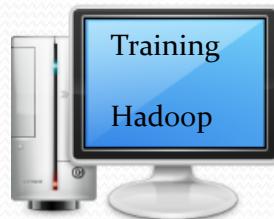
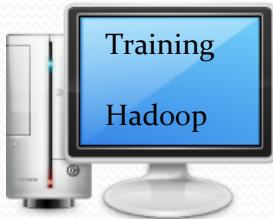
Multinode setup

- Create a dedicated user in all the machines



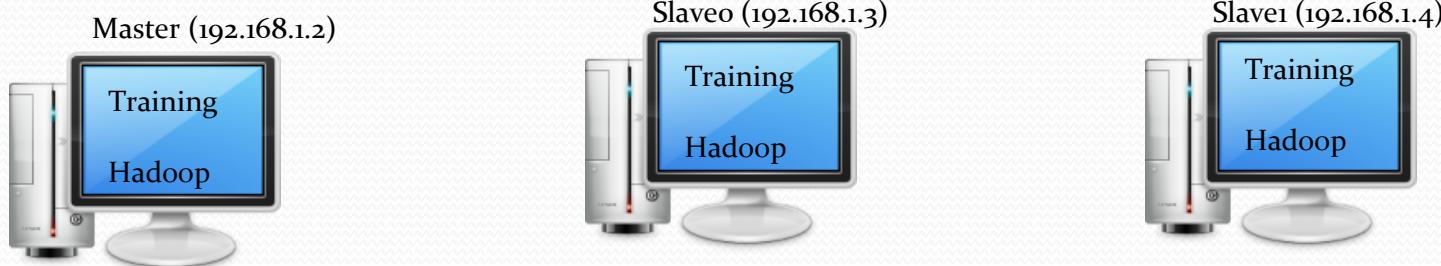
Multinode setup

- Create a dedicated user in all the machines
- Install Hadoop on all the machine



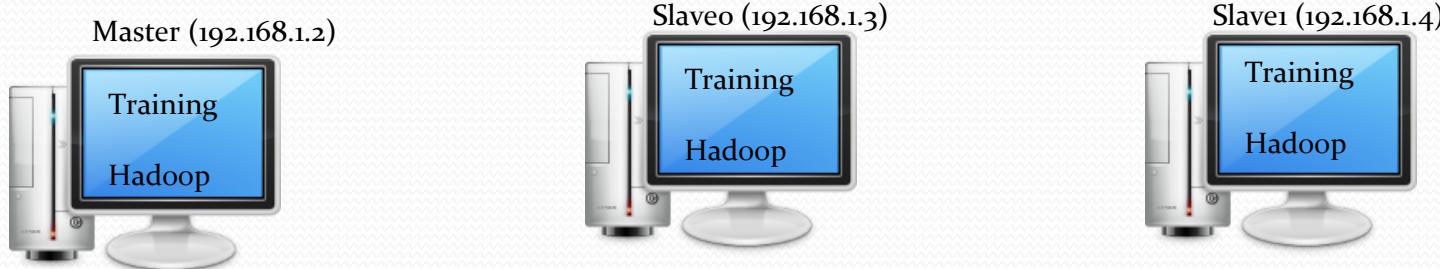
Multinode setup

- Create a dedicated user in all the machines
- Install Hadoop on all the machine
- You wont work with local host. Need to work with real IP.
- Always work with hostnames



Multinode setup

- Create a dedicated user in all the machines
- Install Hadoop on all the machine
- You wont work with local host. Need to work with real IP.
- Always work with hostnames
- Use same JDK across all the machine
- Make sure password less communication is working



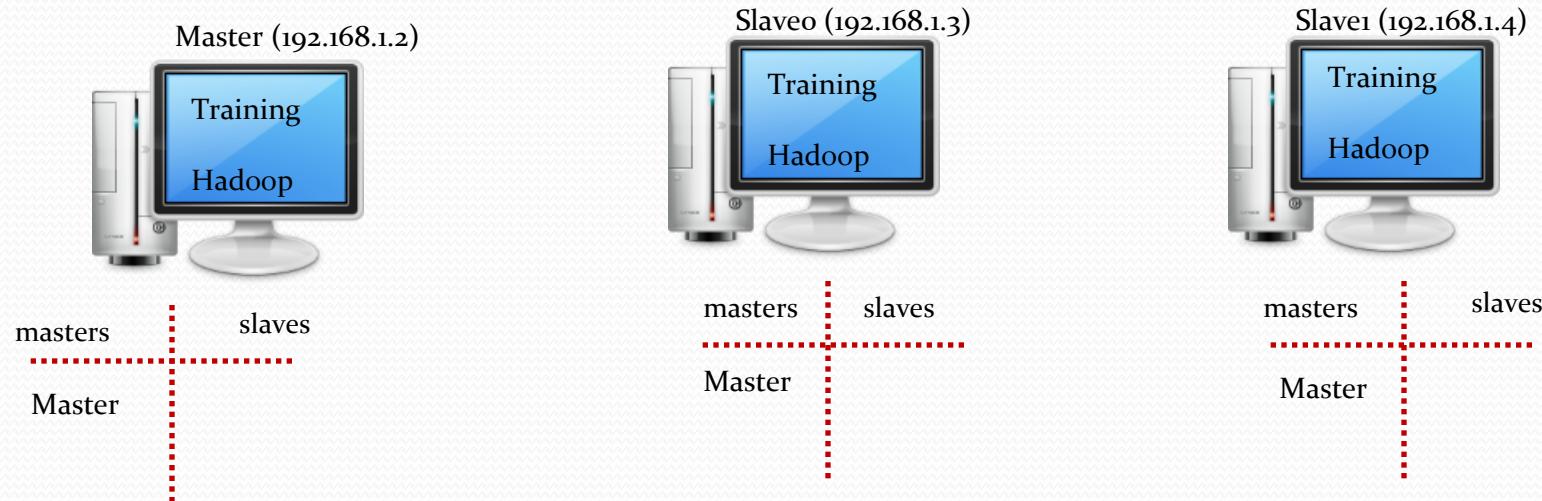
Multinode setup

- Decide master machine



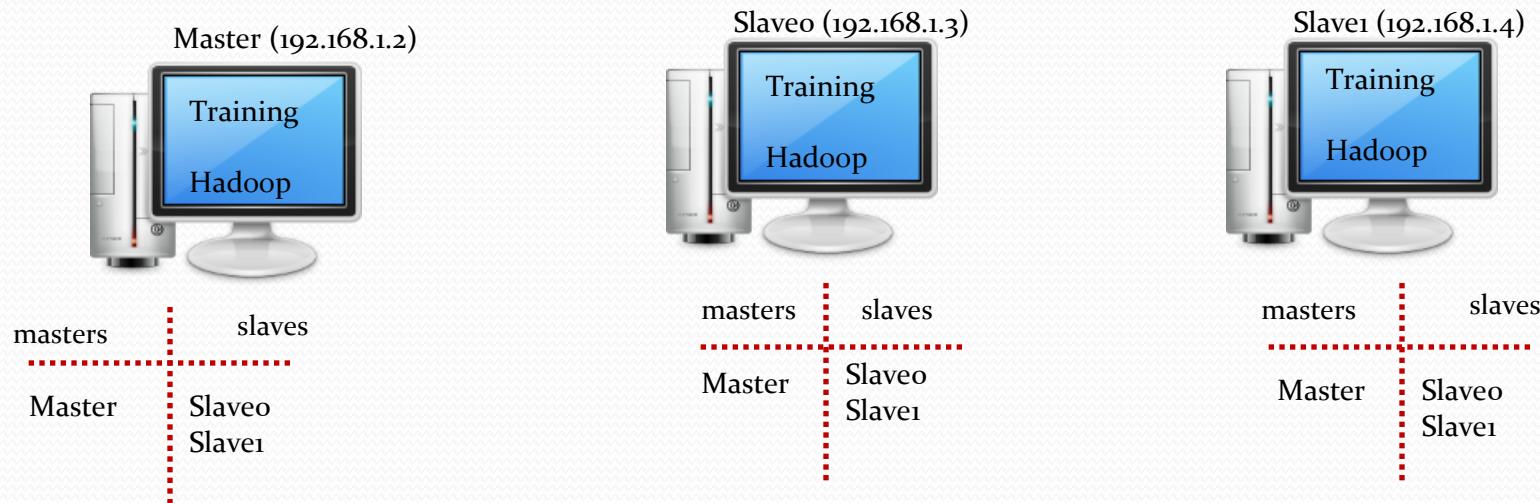
Multinode setup

- Decide master machine



- Its not mandatory to provide the slaves entry
- Providing the slaves entry will help in starting the process, else you need to start the process manually

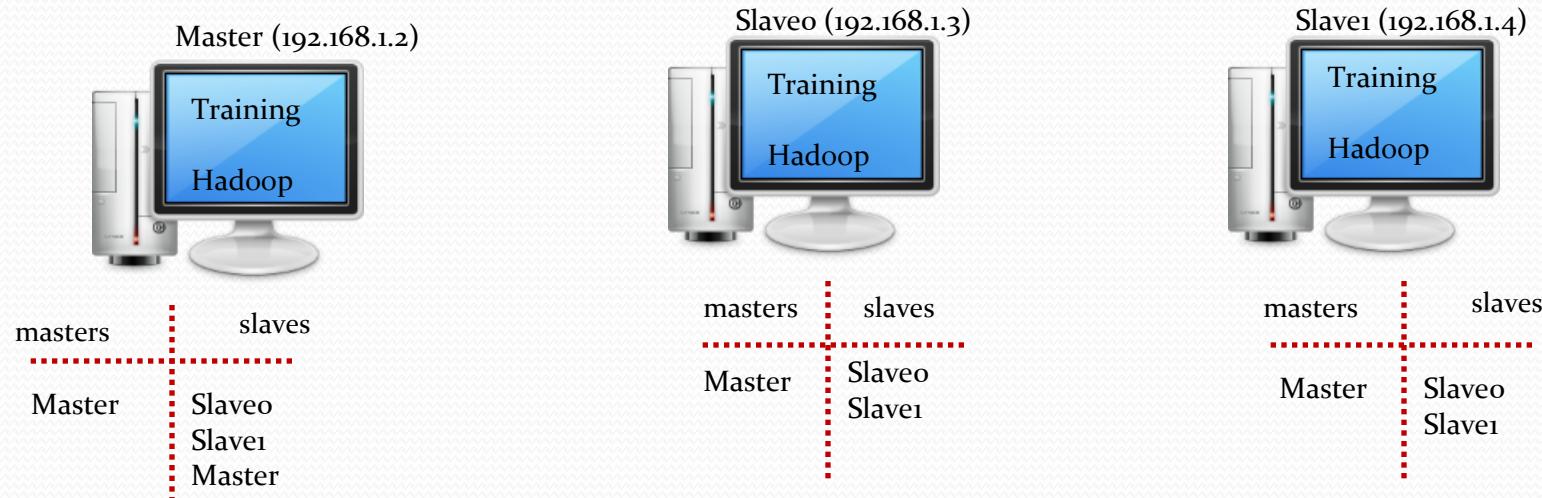
Multinode setup



- There can be only 1 master process, so every one must know where NN and JT are running

Multinode setup

- In case if you want to make your master machine to act as slave



Edit \$HADOOP_HOME/conf/mapred-site.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>
    <name>mapred.job.tracker</name>
    <value>IP_OF_MASTER:54311</value>
</property>

</configuration>
```

Edit \$HADOOP_HOME/conf/hdfs-site.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
<name>dfs.replication</name>
<value>3</value>
</property>

<property>
<name>dfs.block.size</name>
<value>67108864</value>
</property>
</configuration>
```

Edit \$HADOOP_HOME/conf/core-site.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. --&gt;

&lt;configuration&gt;
  &lt;property&gt;
    &lt;name&gt;hadoop.tmp.dir&lt;/name&gt;
    &lt;value&gt;/home/training/hadoop-temp&lt;/value&gt;
    &lt;description&gt;A base for other temporary directories.&lt;/description&gt;
  &lt;/property&gt;
  &lt;property&gt;
    &lt;name&gt;fs.default.name&lt;/name&gt;
    &lt;value&gt;hdfs://IP_OF_MASTER:54310&lt;/value&gt;
  &lt;/property&gt;
&lt;/configuration&gt;</pre>
```

NOTE

- These are the set of configuration which needs to be the same across all machines
- Do the configuration in one machine and use linux “rsync” command to sync the files

Start the Hadoop cluster

- Run the following command from the master machine
 - `start-all.sh`

Notes On Hadoop Process – Single Node

Pseudo Mode / Single Node



Notes On Hadoop Process – Single Node

Pseudo Mode / Single Node



Notes On Hadoop Process – Single Node

Pseudo Mode / Single Node



Notes On Hadoop Process – Single Node

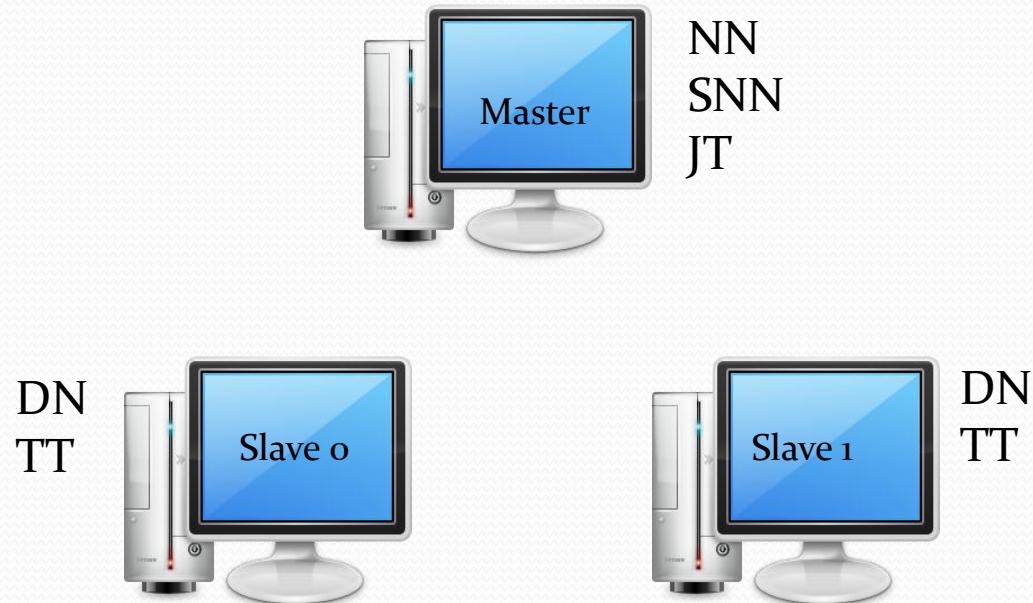
Pseudo Mode / Single Node



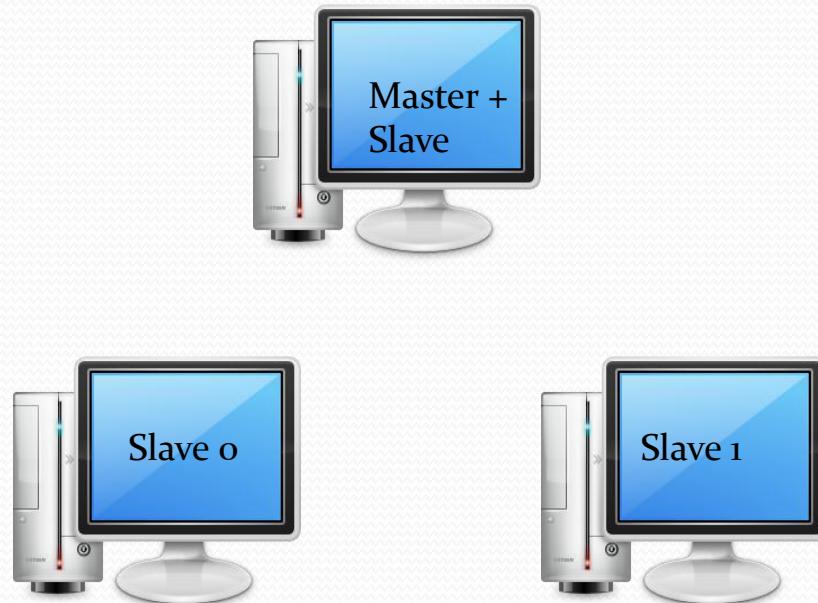
Notes On Hadoop Process – MultiNode



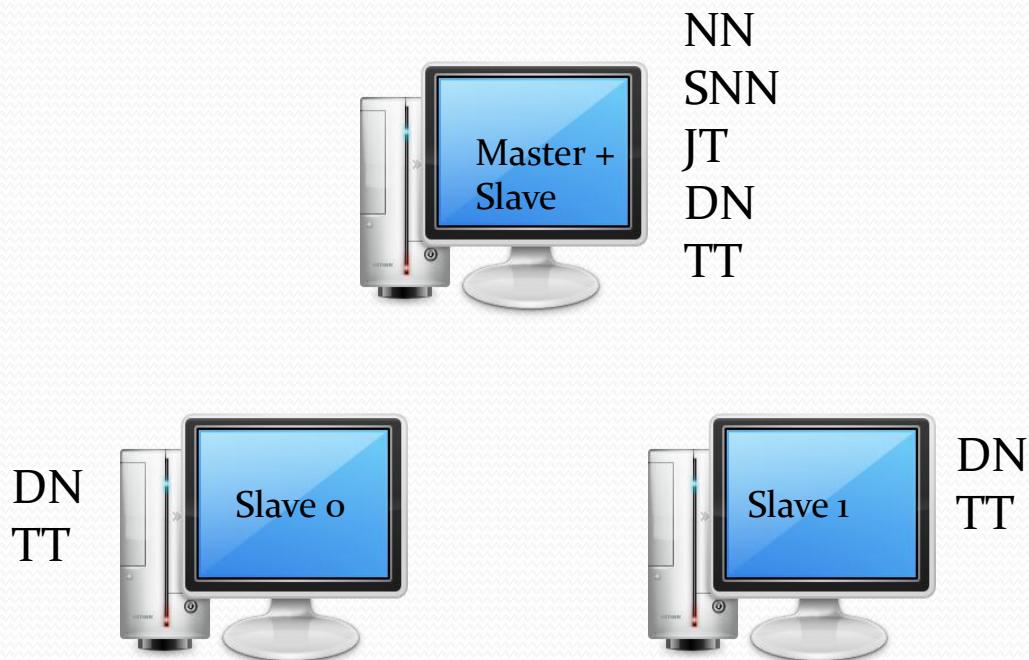
Notes On Hadoop Process – MultiNode



Notes On Hadoop Process – MultiNode



Notes On Hadoop Process – MultiNode



MultiNode



>> I have big data with me and
I would like to put my data onto
HDFS which can be processed
later by my another team



Master



Slaveo



Slave1

MultiNode



>>What should I know in order to put my file on to HDFS?

>> Whom I need to contact?



Master



Slave0



Slave1

Multinode



NN
SNN
JT



Master



Slaveo



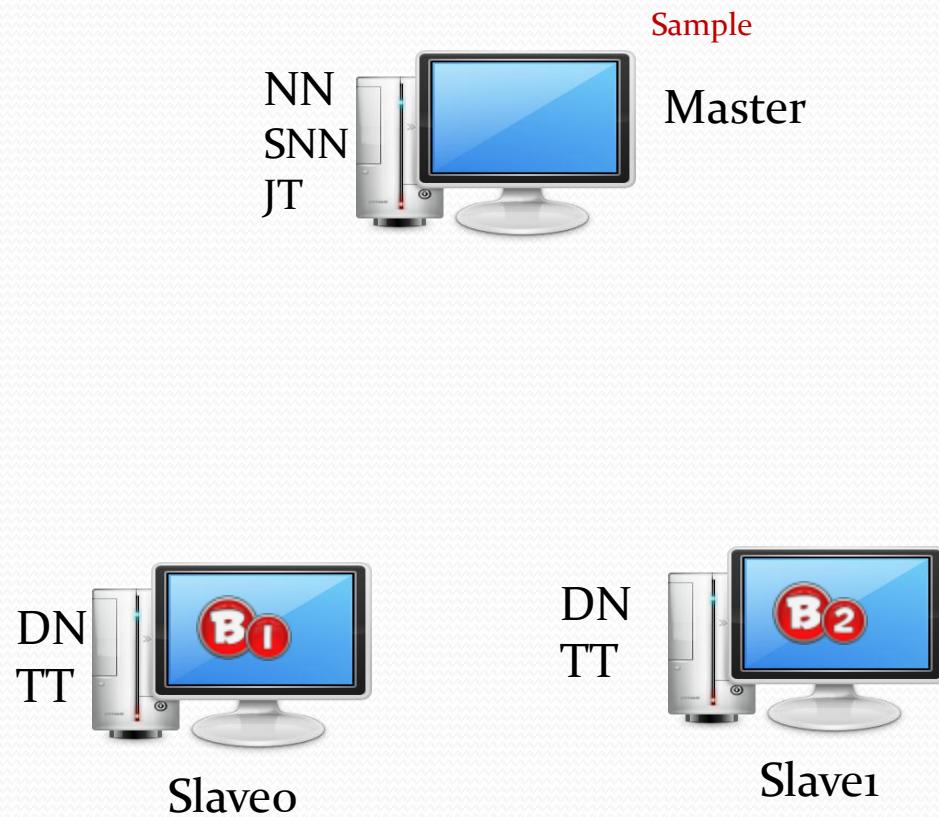
Slave1

Multinode

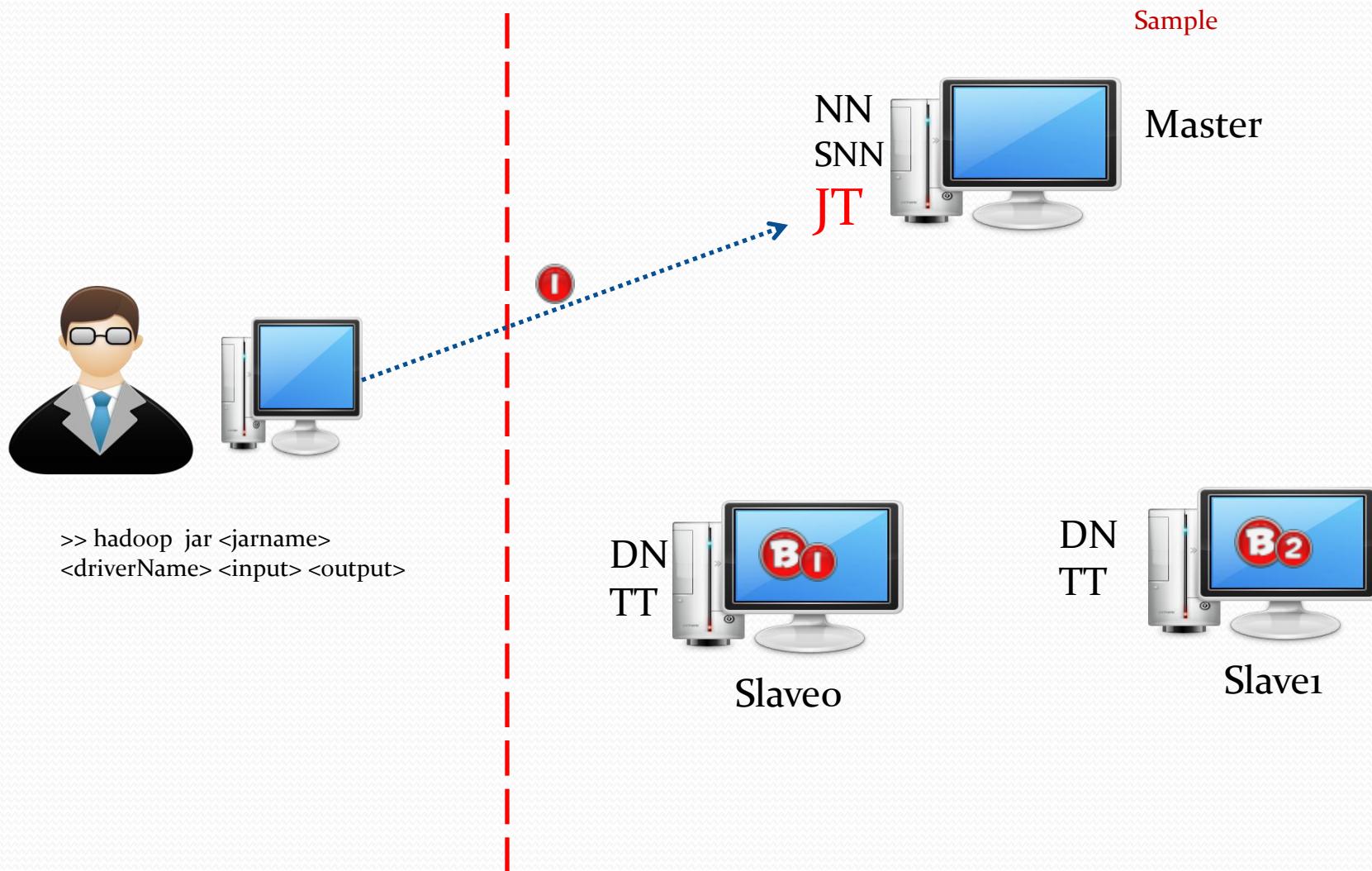


>> I have written MapReduce Job to analyze the sample file on HDFS

>> What I should know inorder to submit my MapReduce Job?



Multinode



Hadoop Set up in Production

NameNode



JobTracker



SecondaryNameNode



DataNode
TaskTracker



DataNode
TaskTracker

MODULE 5

Important Concepts

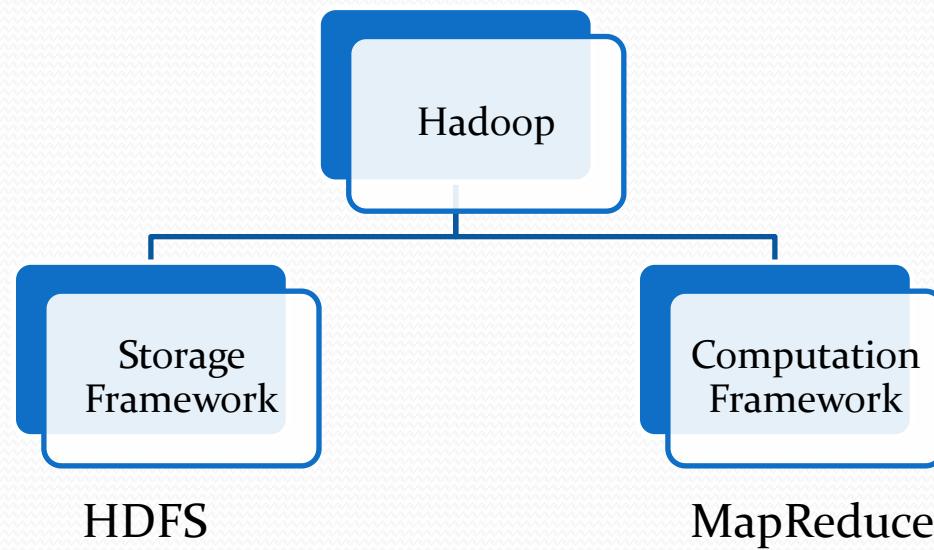
In this module you will learn

- hadoop.tmp.dir
- Incompatible name space ids
- safemode

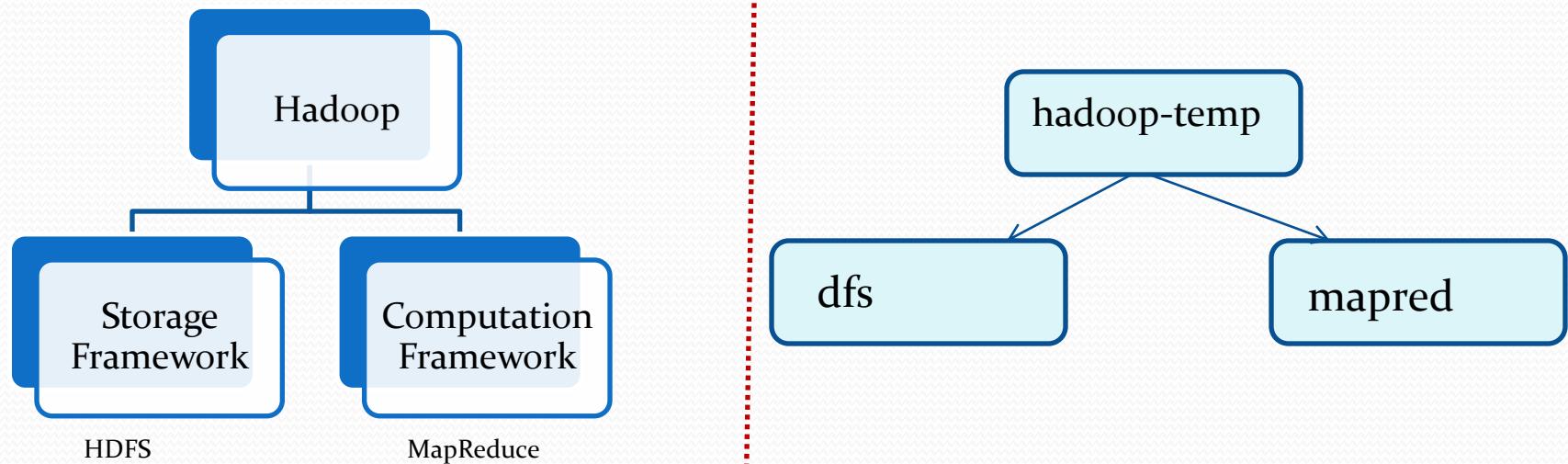
`hadoop.tmp.dir`

- Value of this property is a directory
 - `/home/training/hadoop-temp`
- This directory consist of hadoop file system information
 - Consist of meta data image
 - Consist of blocks etc

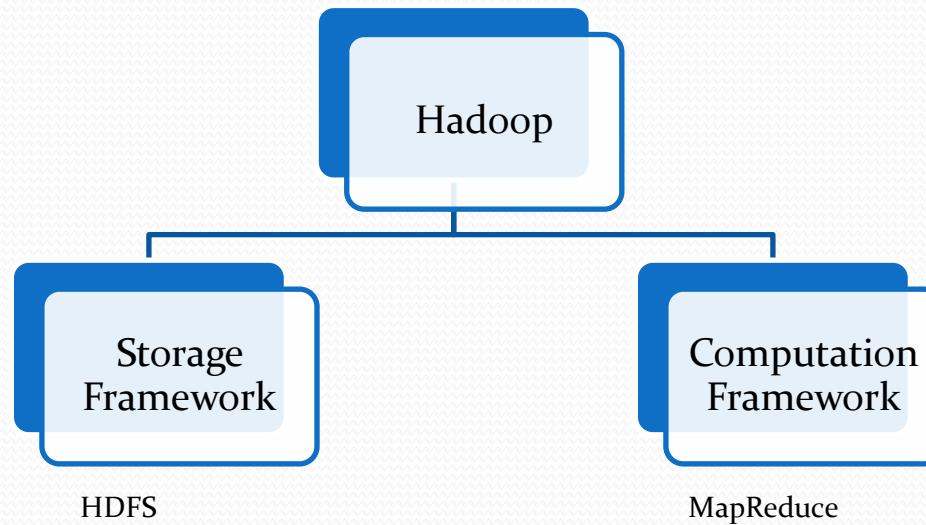
hadoop-temp directory structure



hadoop-temp directory structure

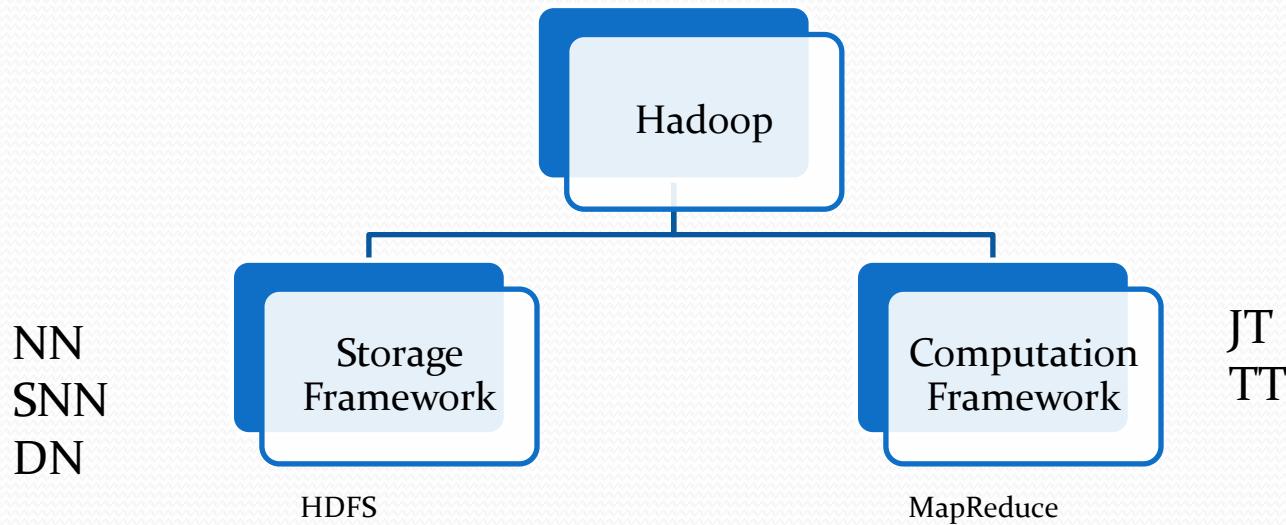


hadoop-temp directory structure

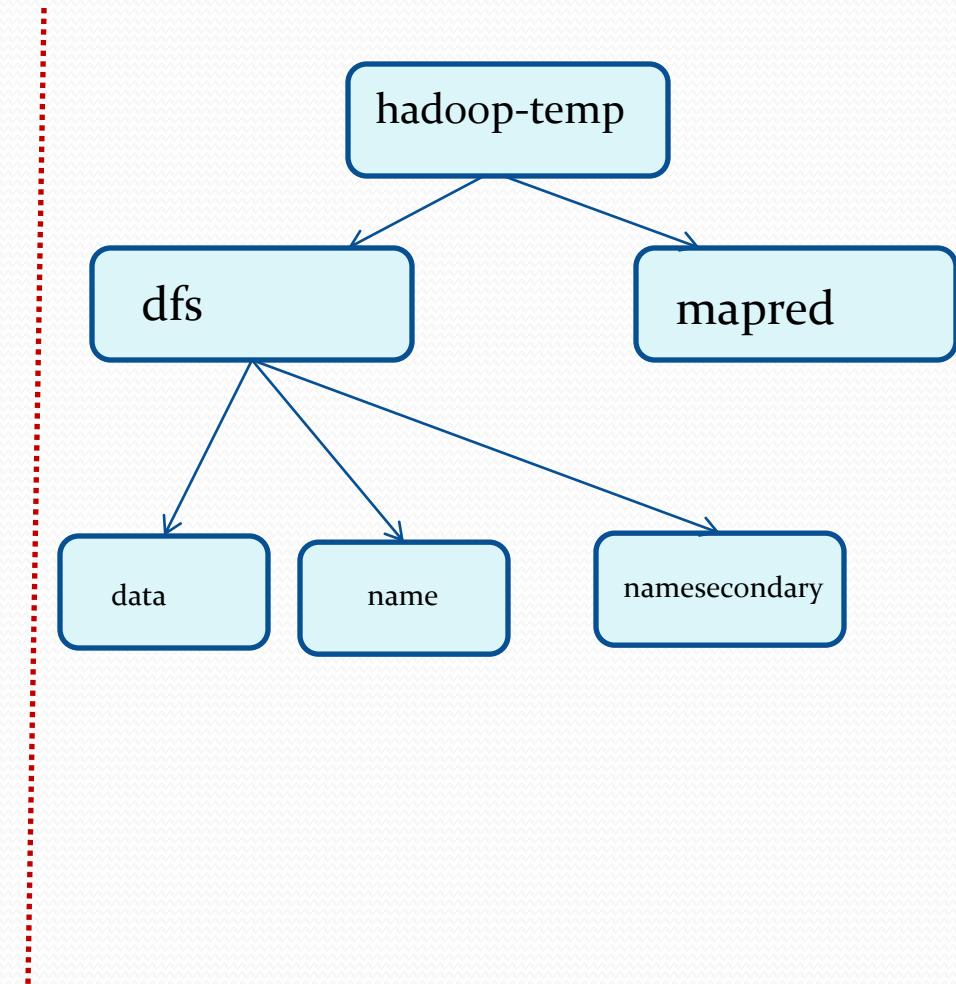
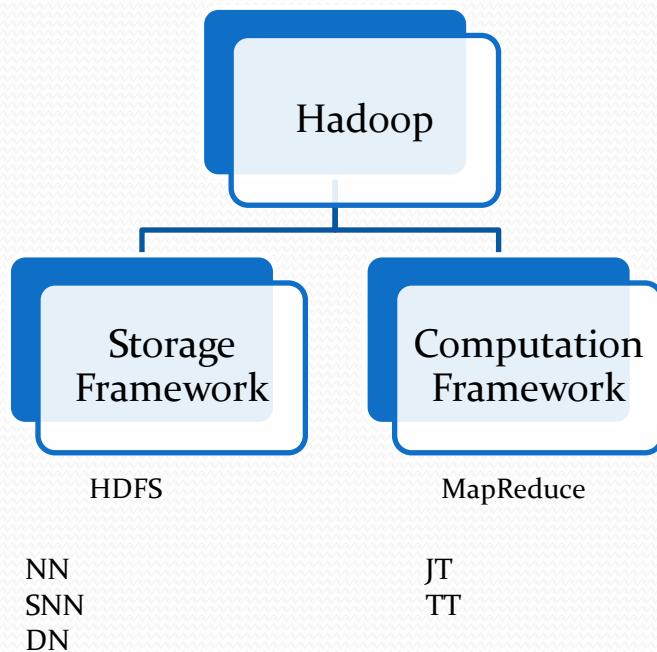


HDFS is managed by how many process?

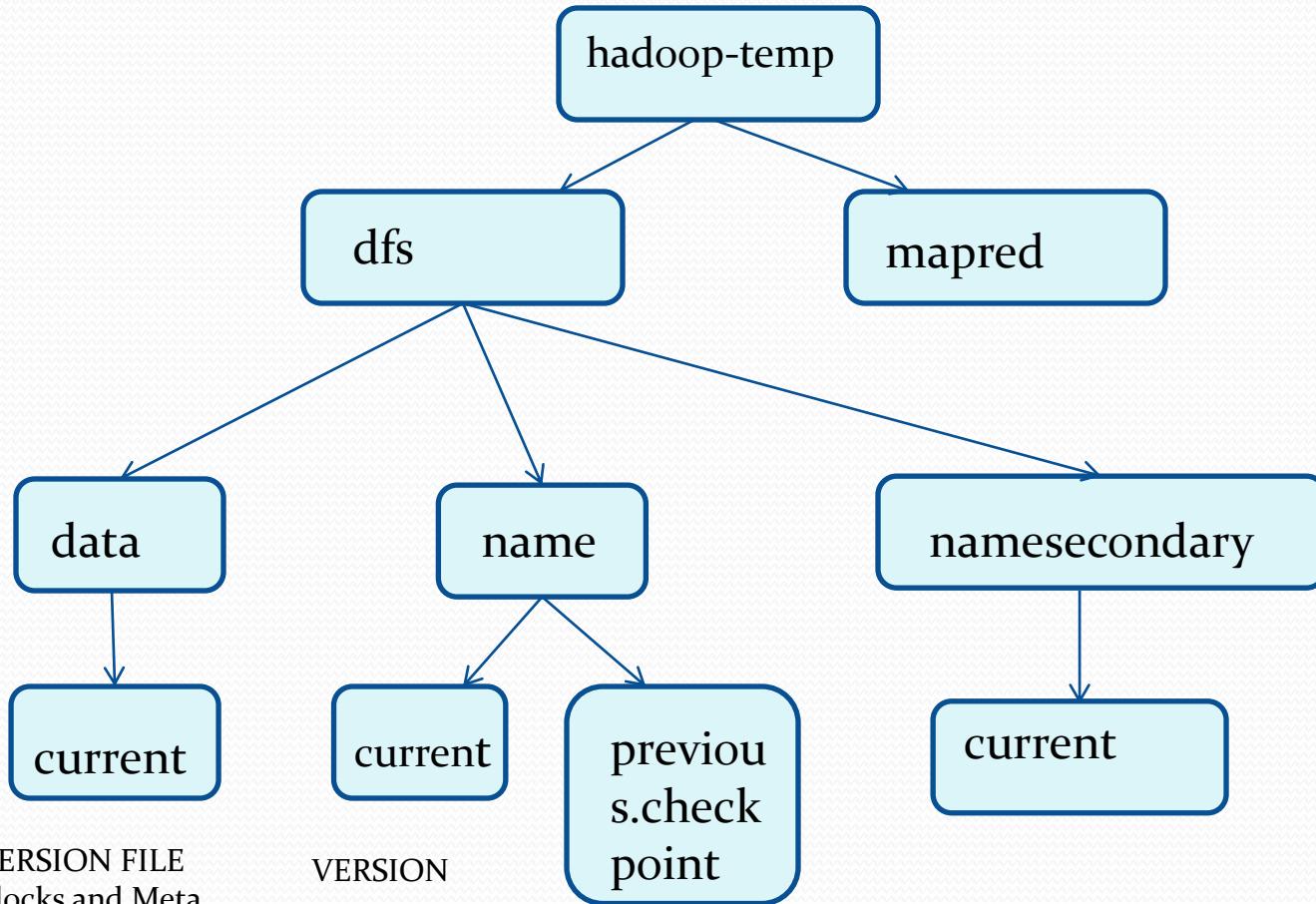
hadoop-temp directory structure



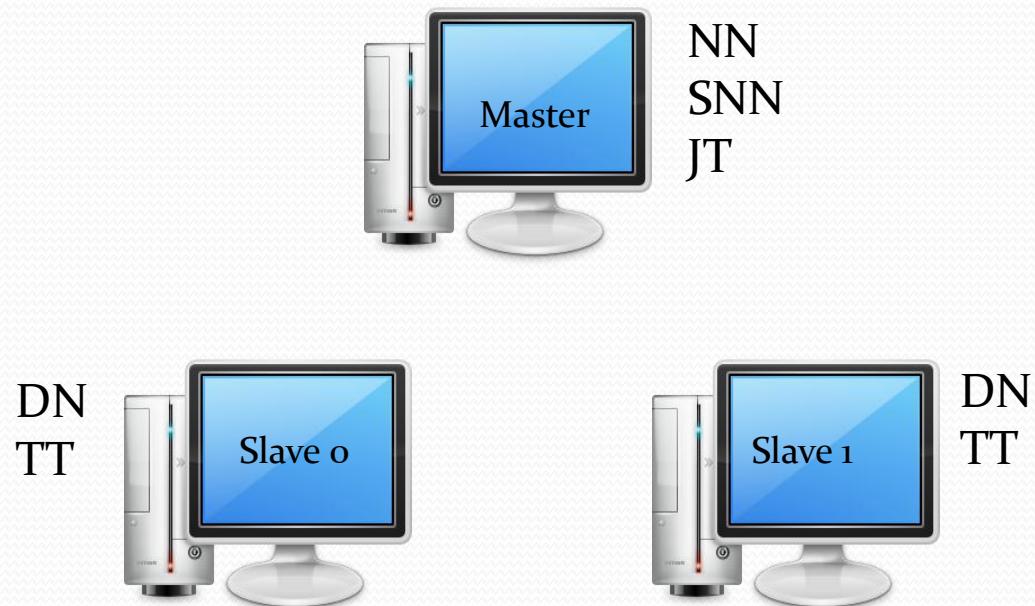
hadoop-temp directory structure



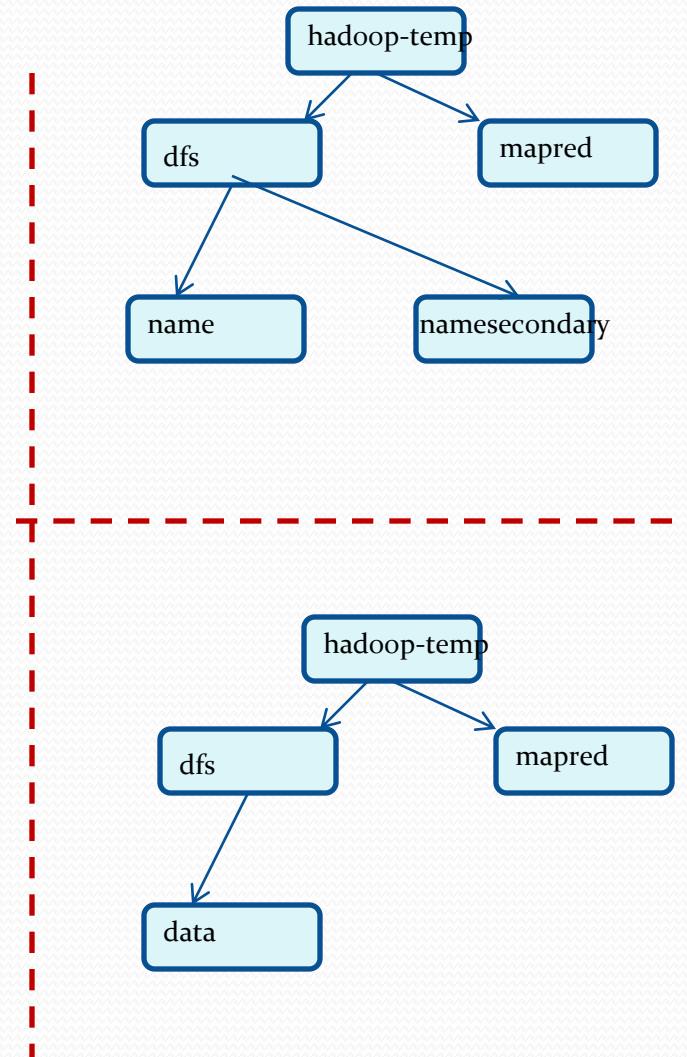
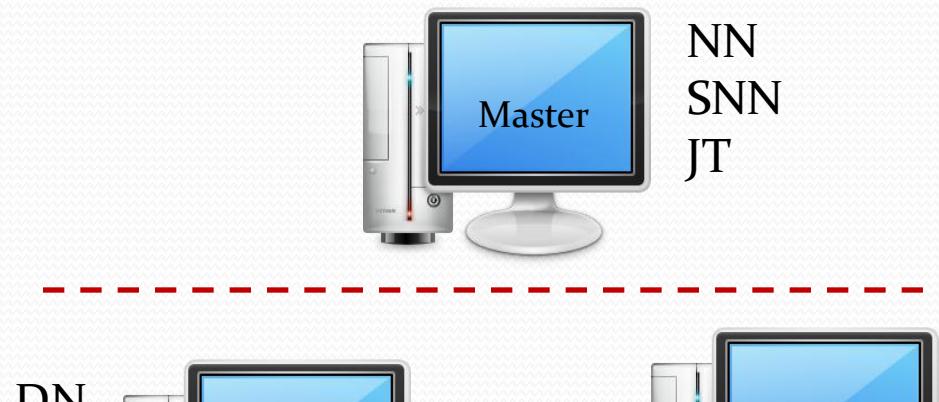
Directory Structure of HDFS on Local file system and Important files



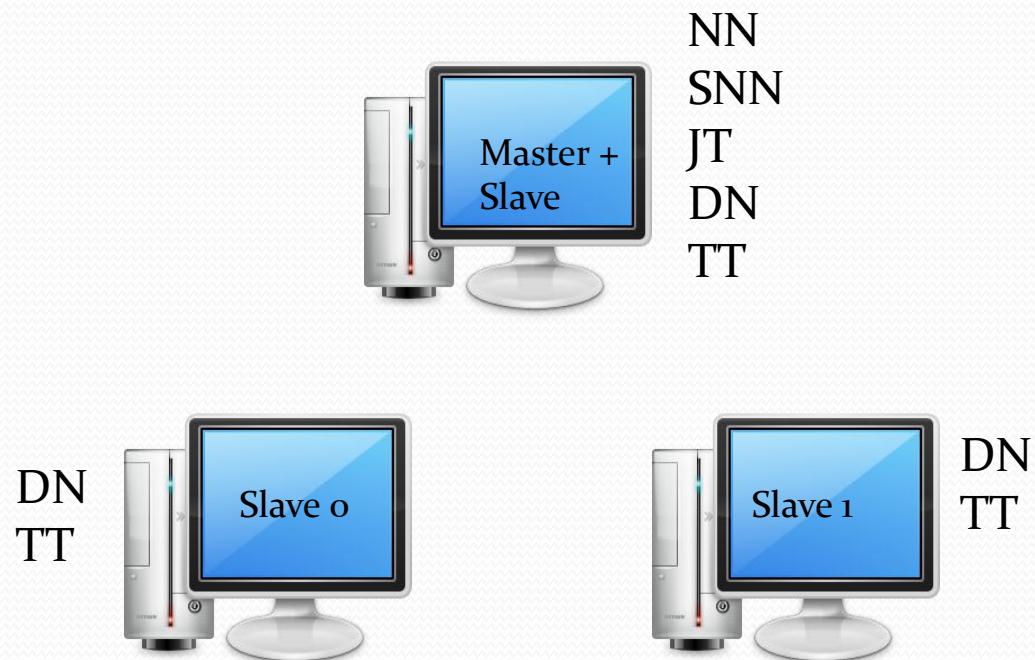
Directory structure on MultiNode



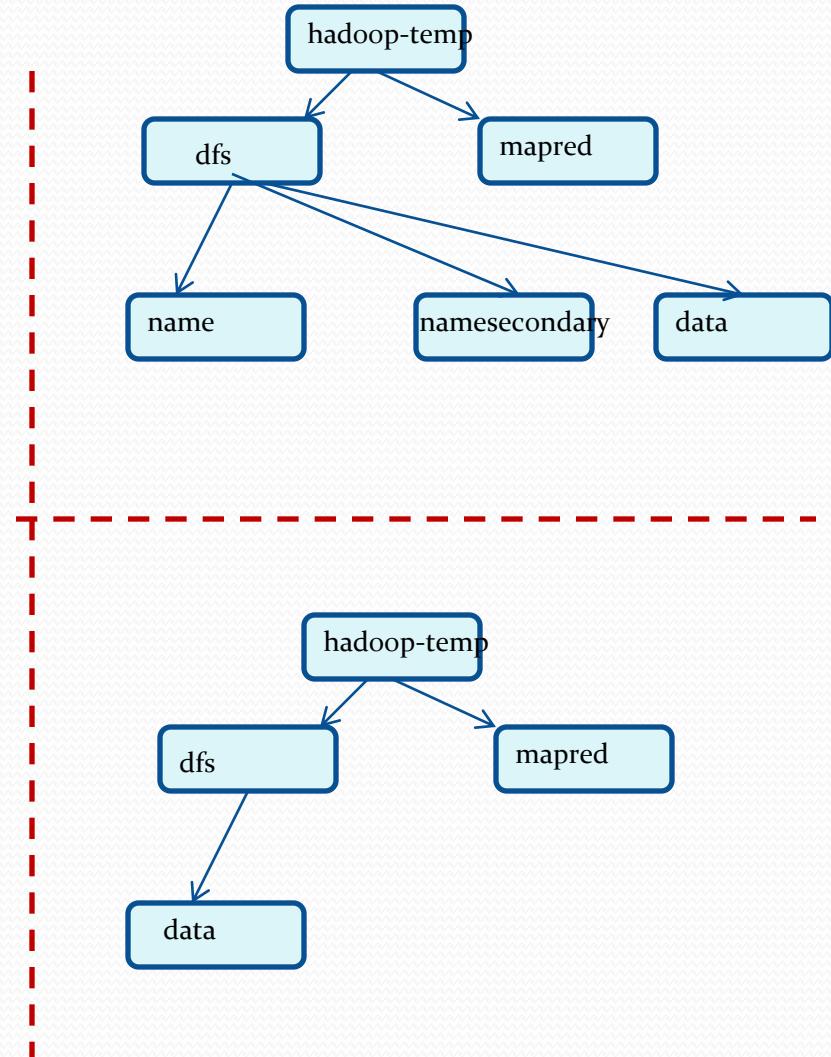
Directory structure on MultiNode



Directory structure on MultiNode



Directory structure on MultiNode



NameSpace ID's

- Data Node NameSpace ID

```
namespaceID=1580234492
storageID=DS-1396273819-127.0.0.1-50010-1349416136847
cTime=0
storageType=DATA_NODE
layoutVersion=-32
```

- NameNode NameSpace ID

```
#Sun Oct 21 00:09:27 PDT 2012
namespaceID=1580234492
cTime=0
storageType=NAME_NODE
layoutVersion=-32
[dev@localhost current]$ █
```

- NameSpace ID has to be same
- You will get “Incompatible NameSpace ID error” if there is mismatch
 - DataNode will not come up

NameSpace ID's cont'd

- Every time NameNode is formatted, new namespace id is allocated to each of the machines (*hadoop namenode -format*)
- DataNode namespace id has to be matched with NameNode's namespace id.
- Formatting the namenode will result into creation of new HDFS and previous data will be lost.

SafeMode

- Starting Hadoop is not a single click
- When Hadoop starts up it has to do lot of activities
 - Restoring the previous HDFS state
 - Waits to get the block reports from all the Data Node's etc
- During this period Hadoop will be in safe mode
 - It shows only meta data to the user
 - It is just Read only view of HDFS
 - Cannot do any file operation
 - Cannot run MapReduce job

SafeMode cont'd

- For doing any operation on Hadoop SafeMode should be off
- Run the following command to get the status of safemode
 - *hadoop dfsadmin -safemode get*
- For maintenance, sometimes administrator turns ON the safe mode
 - *hadoop dfsadmin -safemode enter*

SafeMode cont'd

- Run the following command to turn off the safemode
 - *hadoop dfsadmin -safemode leave*

Assignment

Q1. If you have a machine with a hard disk capacity of 100GB and if you want to store 1TB of data, how many such machines are required to build a hadoop cluster and store 1 TB of data?

Q2. If you have 10 machines each of size 1TB and you have utilized the entire capacity of the machine for HDFS? Then what is the maximum file size you can put on HDFS

- 1TB
- 10TB
- 3TB
- 4TB

MODULE 6

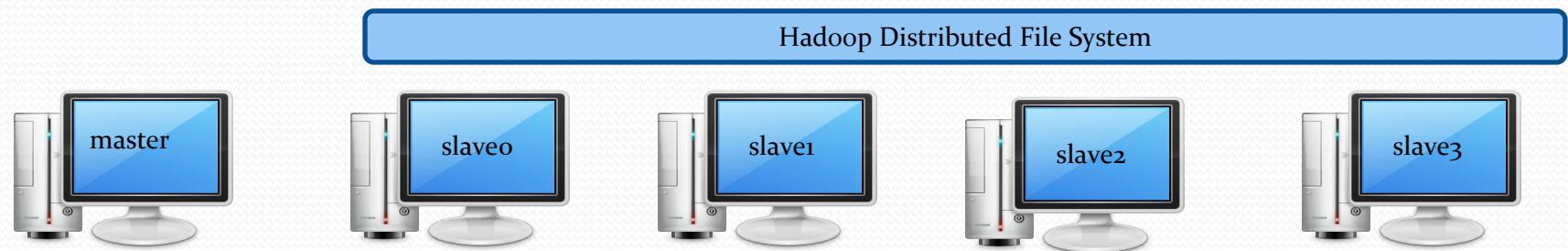
HDFS Shell Commands

In this module you will learn

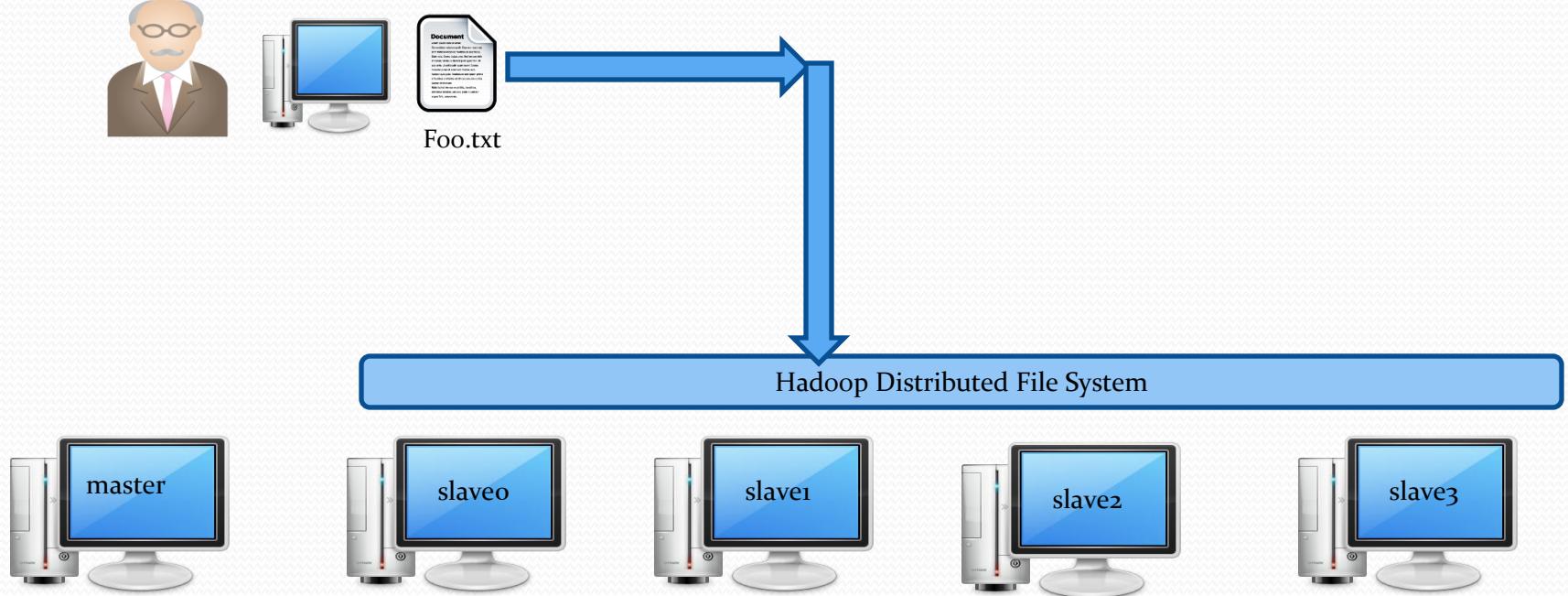
- How to use HDFS Shell commands
- Command to list the directories and files
- Command to put files on HDFS from local file system
- Command to put files from HDFS to local file system
- Displaying the contents of file

What is HDFS?

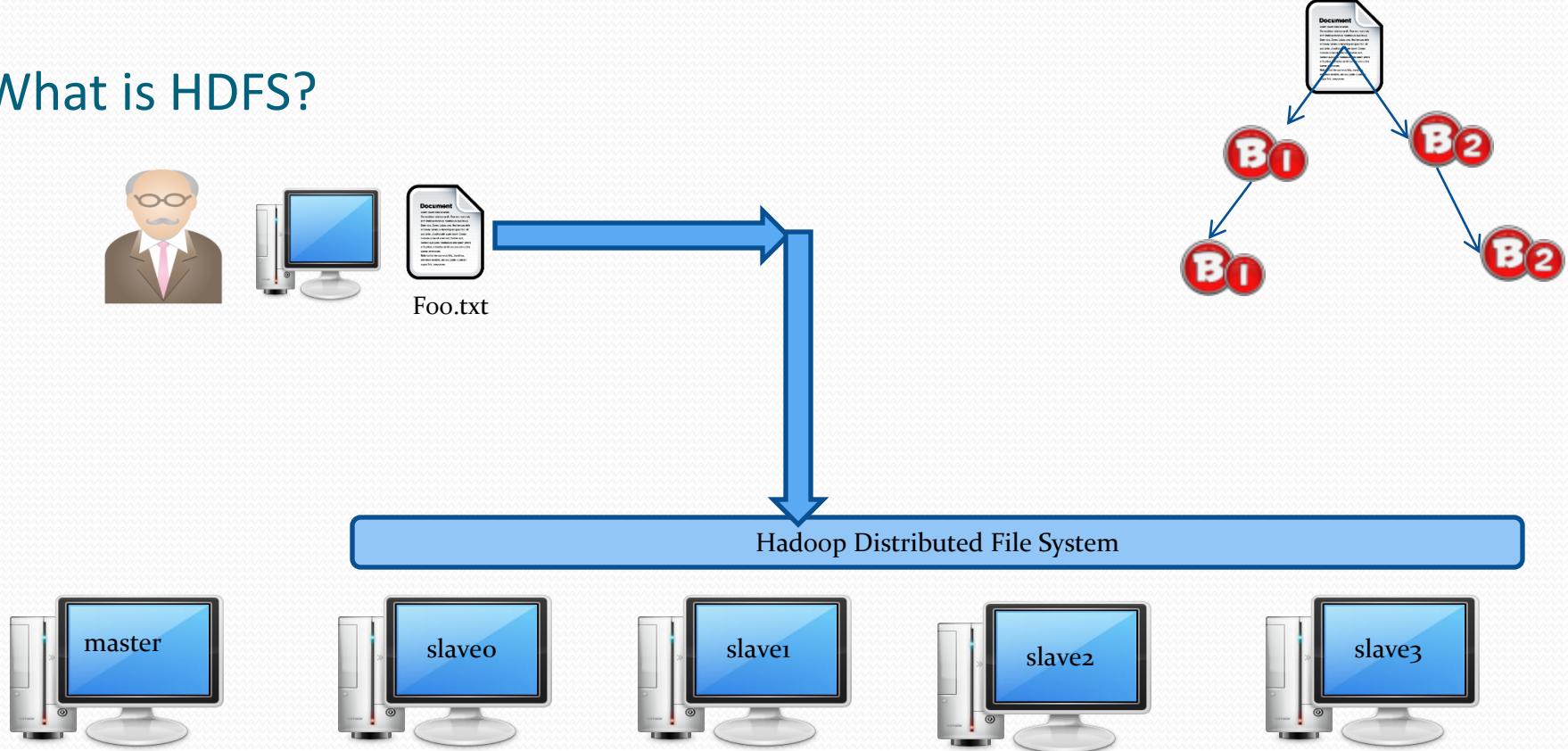
- It's a layered or Virtual file system and does not modify underlying file system
- The HDFS is accessible only when the process are running (NN and DN)



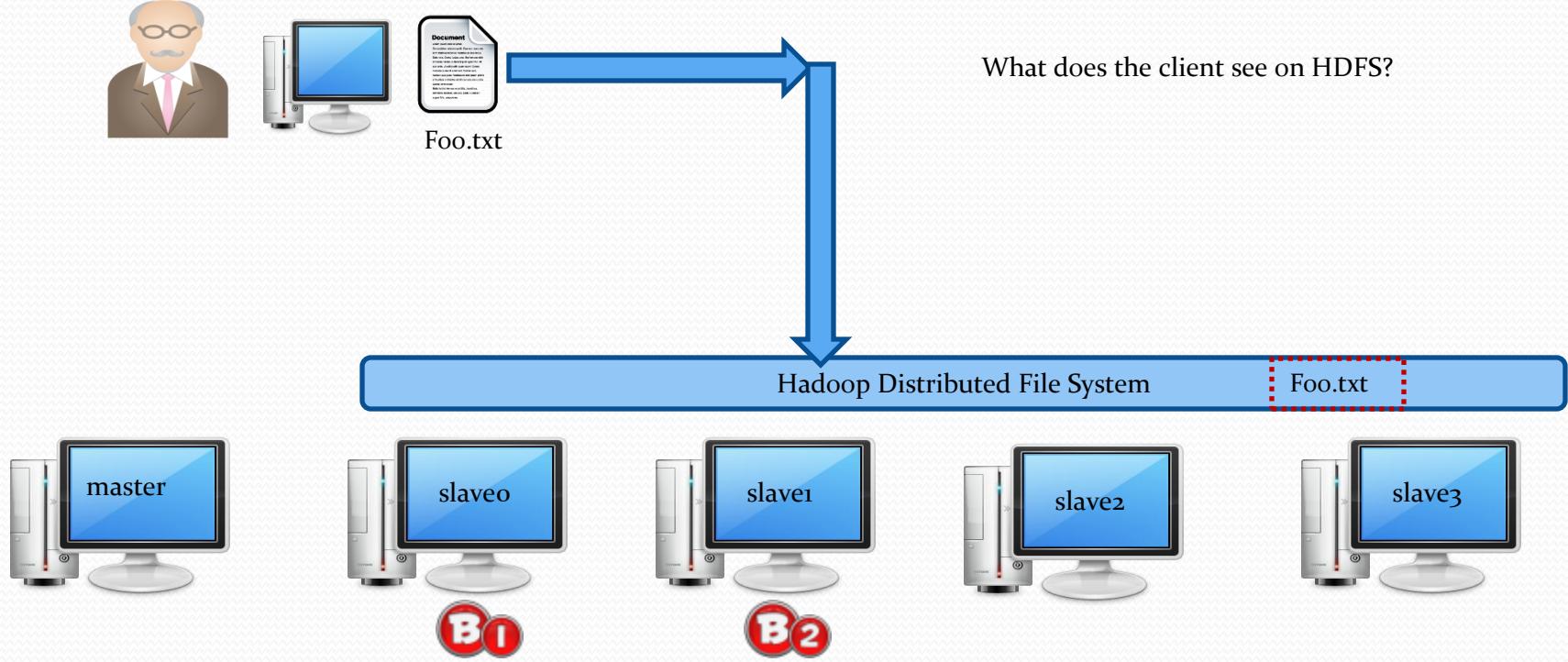
What is HDFS?



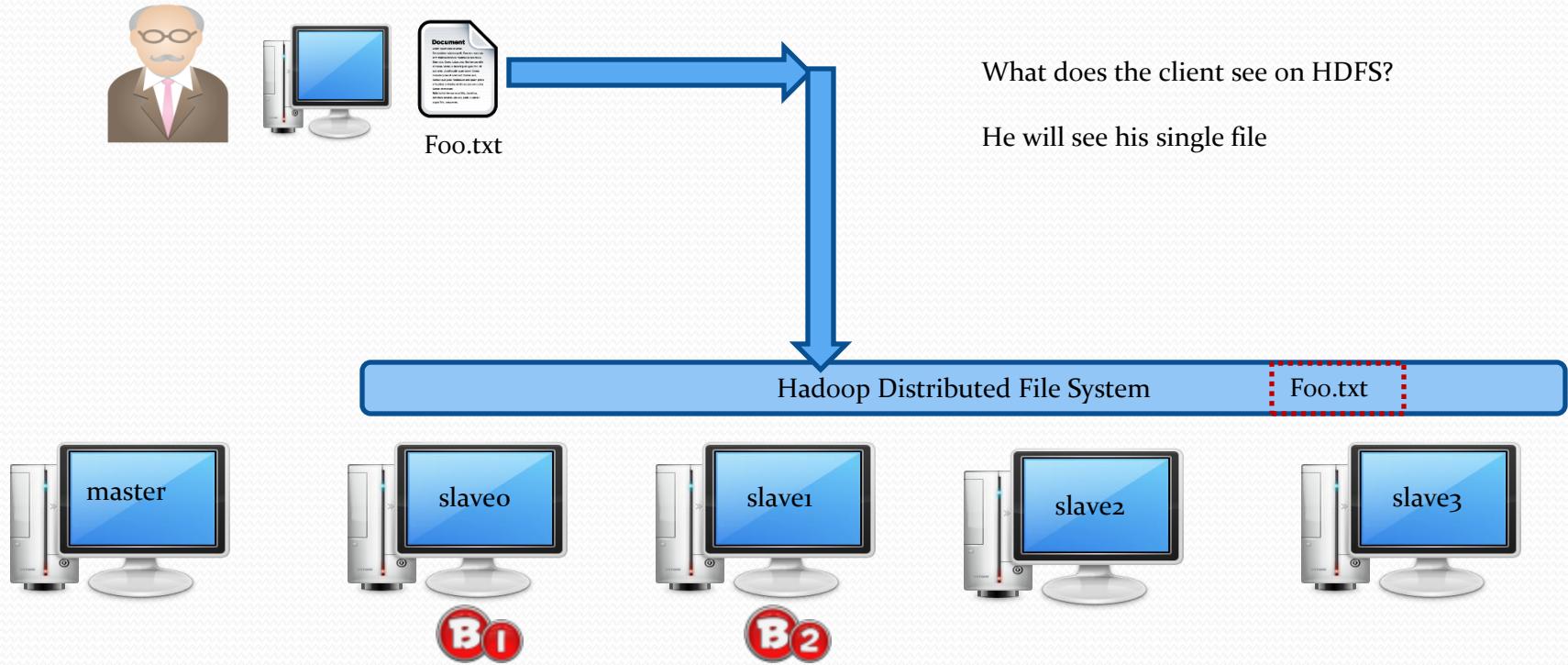
What is HDFS?



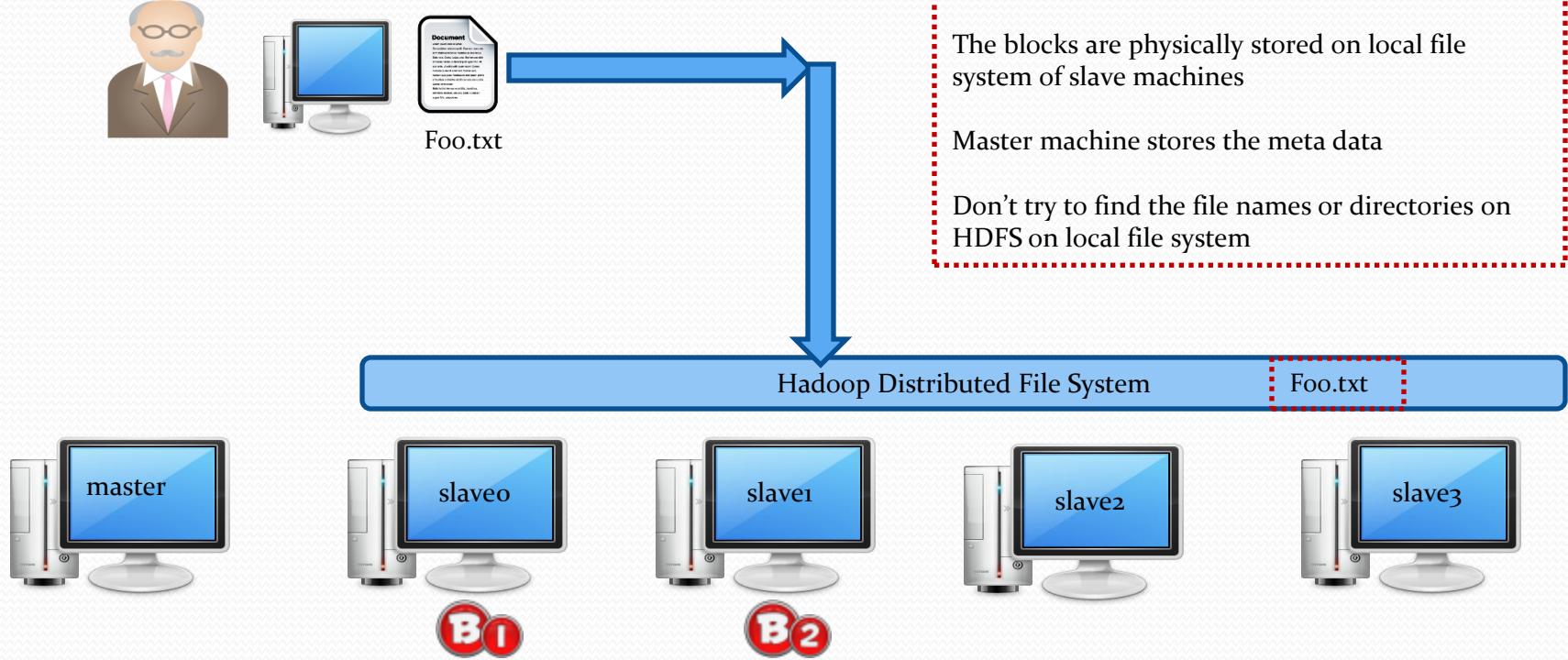
What is HDFS?



What is HDFS?

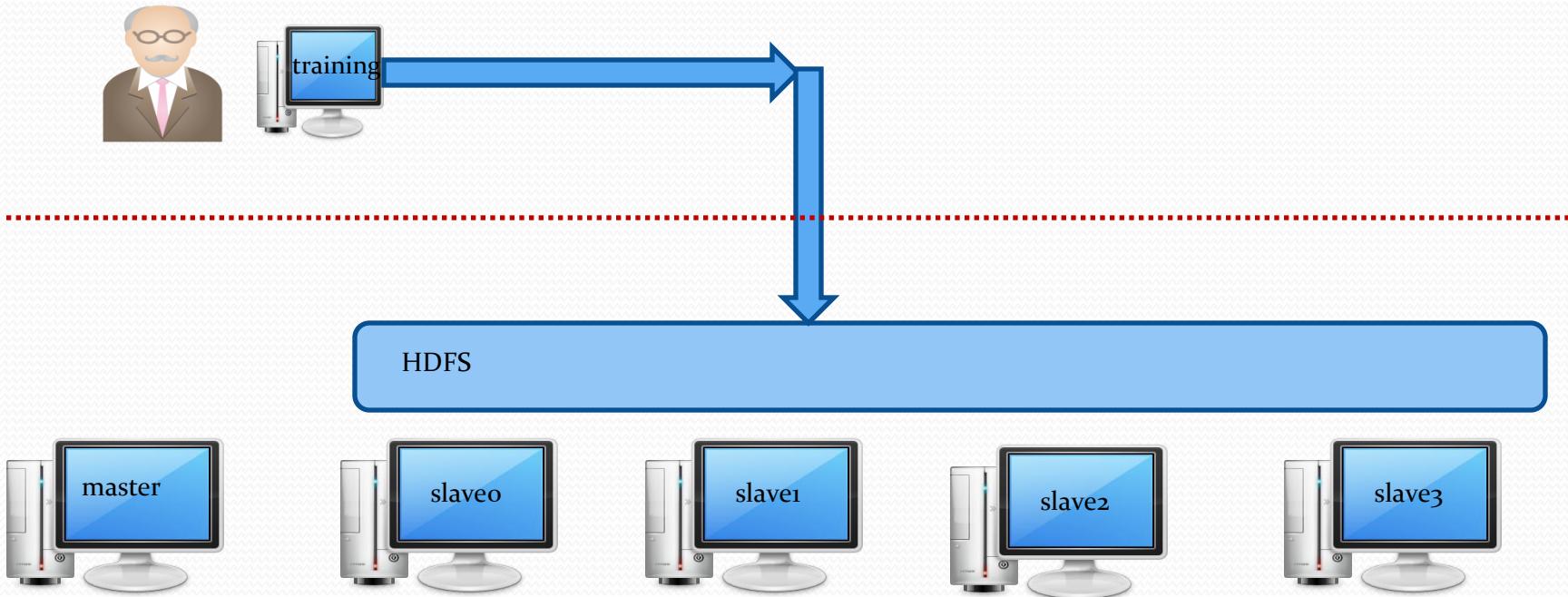


What is HDFS?



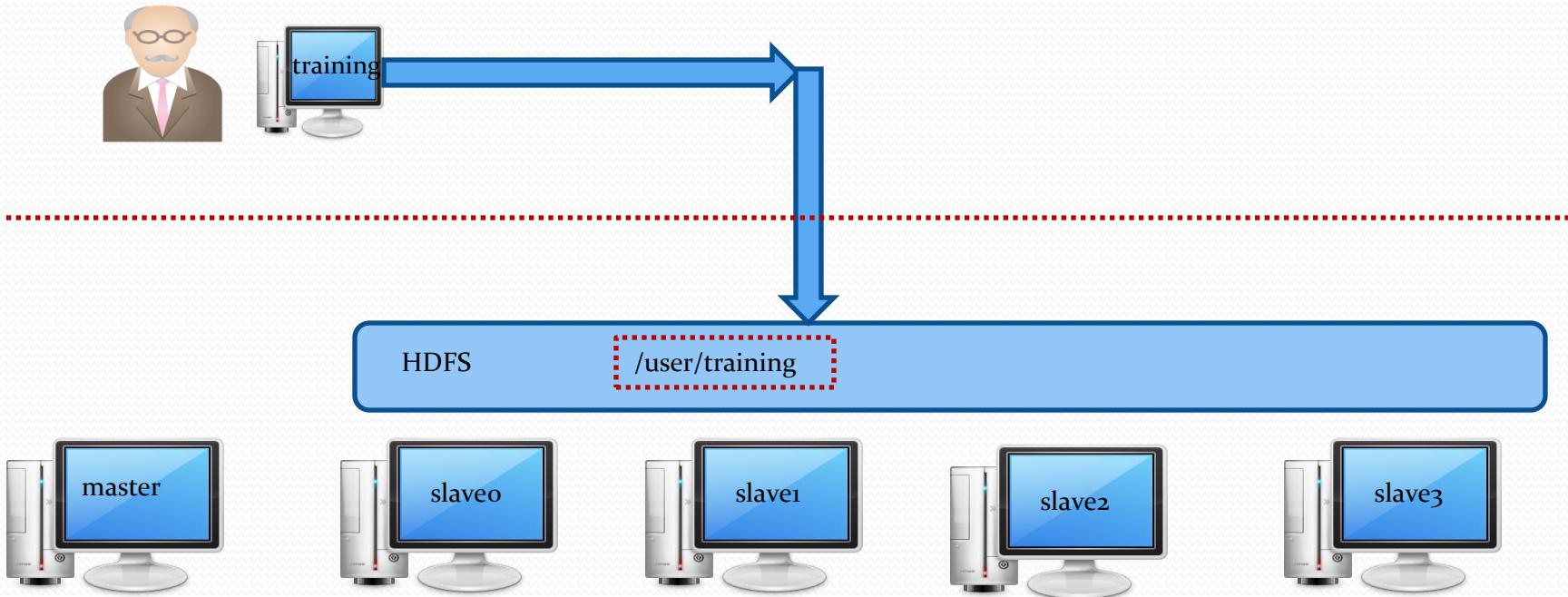
HDFS User Home Directory

- `fs.default.name/user/<username>`



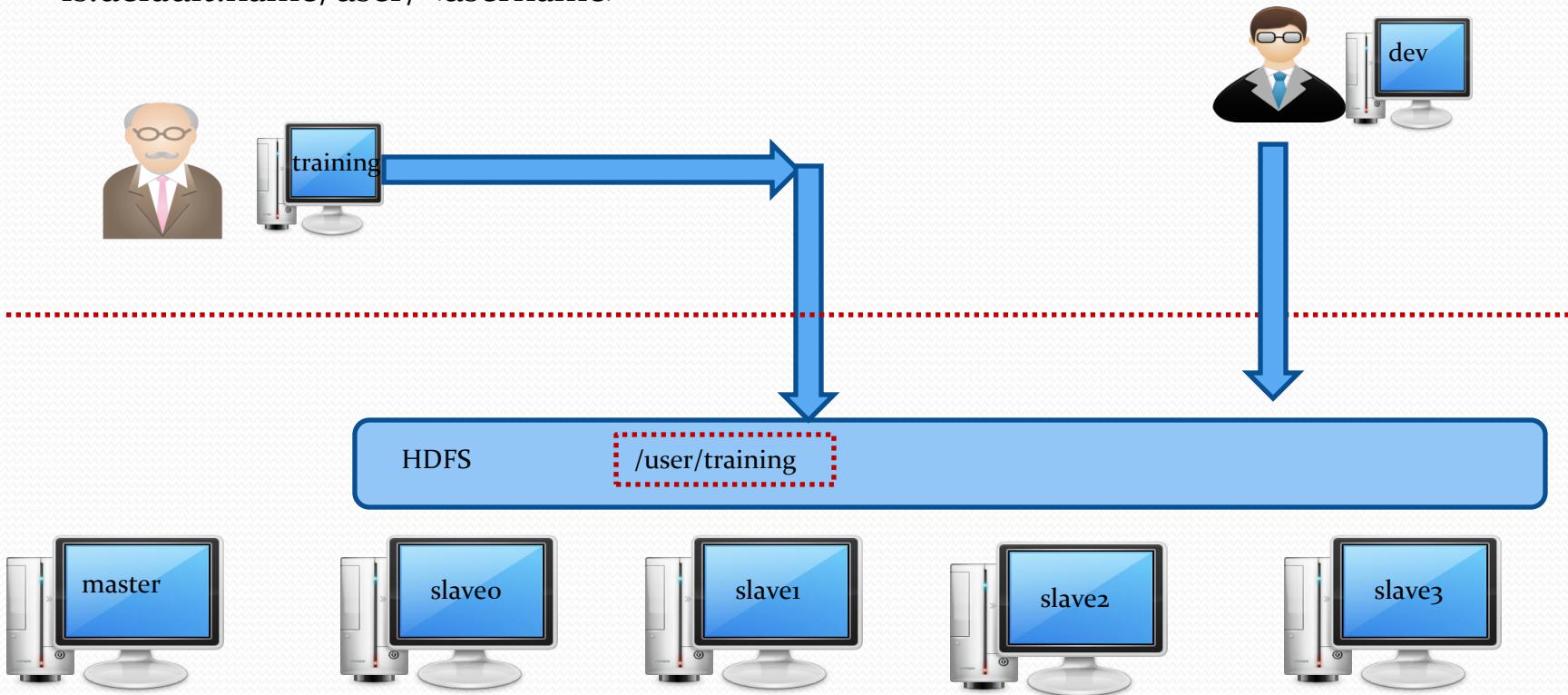
HDFS User Home Directory

- `fs.default.name/user/<username>`



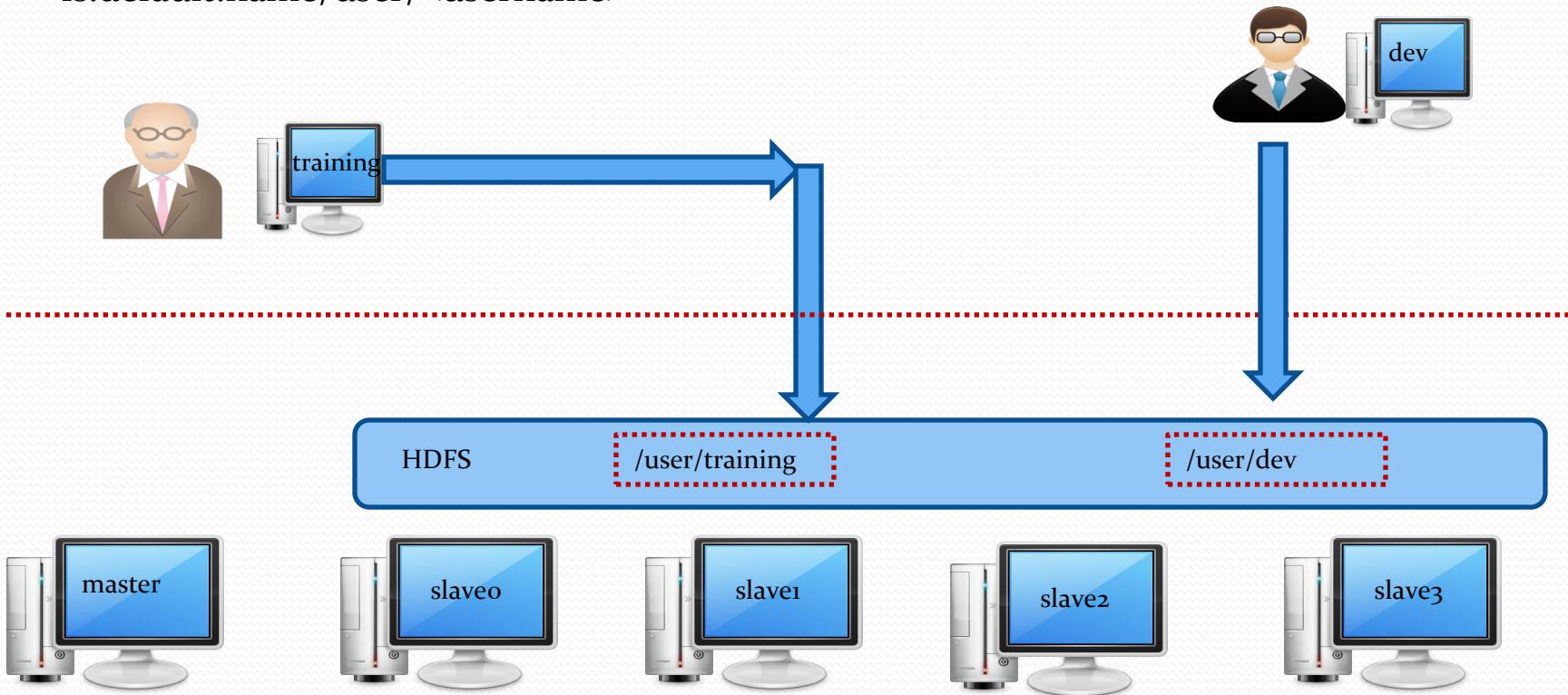
HDFS User Home Directory

- `fs.default.name/user/<username>`



HDFS User Home Directory

- `fs.default.name/user/<username>`



Accessing HDFS through command line

- Remember it is not regular file system
- For accessing HDFS through command line use “*hadoop fs -options*”
 - “*options*” are the various commands
 - `hadoop fs -mkdir myDir`
 - Does not support all the Linux commands
 - Cannot open a file in HDFS using VI editor or any other editor for editing. That means you cannot edit the file residing on HDFS
 - HDFS doesn't support random write. It supports only appends (End of File)
 - While performing operations if explicit path is not specified then operations are performed relative to user's home directory on HDFS
 - `hadoop fs -mkdir myDir`
 - The `myDir` will be created under user's home directory on HDFS.

Common Operations

- Creating a directory on HDFS
- Putting a local file on HDFS
- Listing the files and directories
- Copying a file from HDFS to local file system
- Viewing the contents of file
- Listing the blocks comprising of file and their location

Creating directory on HDFS

Assuming “training” user is performing the commands

```
hadoop fs -mkdir /user/training/foo
```

```
hadoop fs -mkdir bar
```

```
hadoop fs -mkdir foo/bar
```

```
hadoop fs -mkdir /foo
```

Listing directories and files

Assuming “training” user is performing the commands

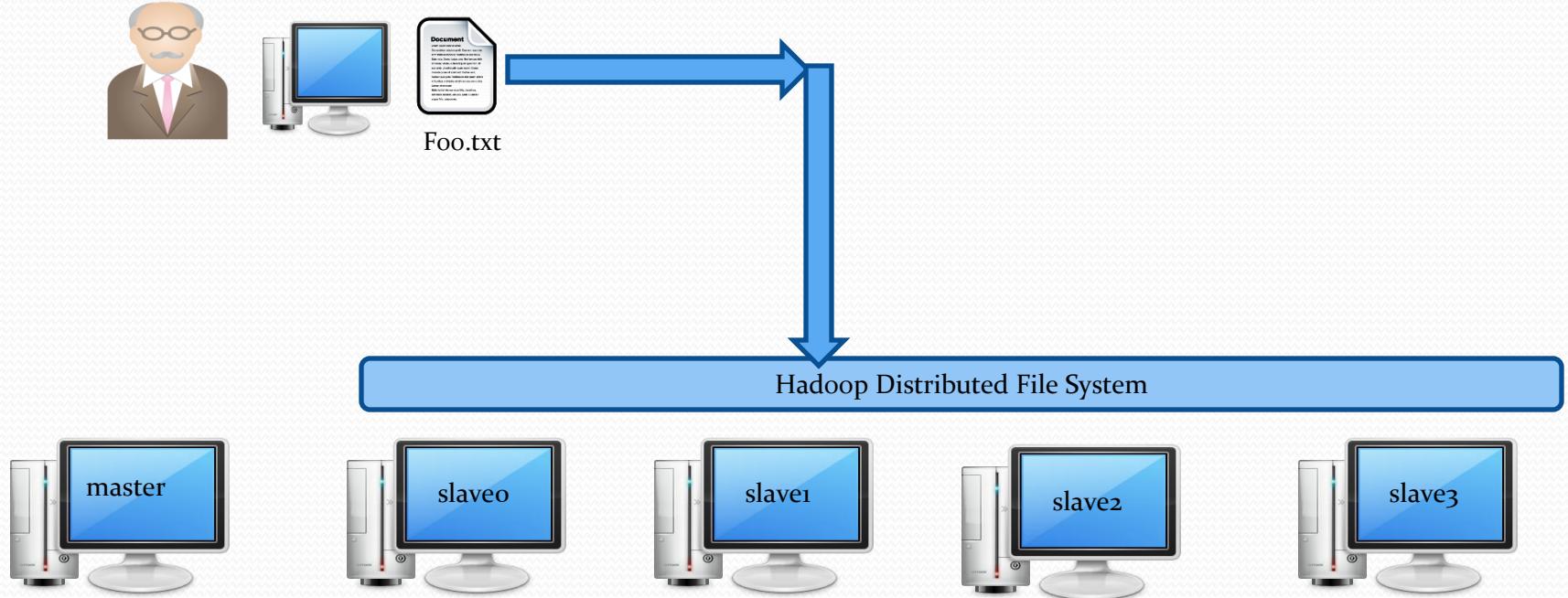
```
hadoop fs -ls /user/training
```

```
hadoop fs -ls
```

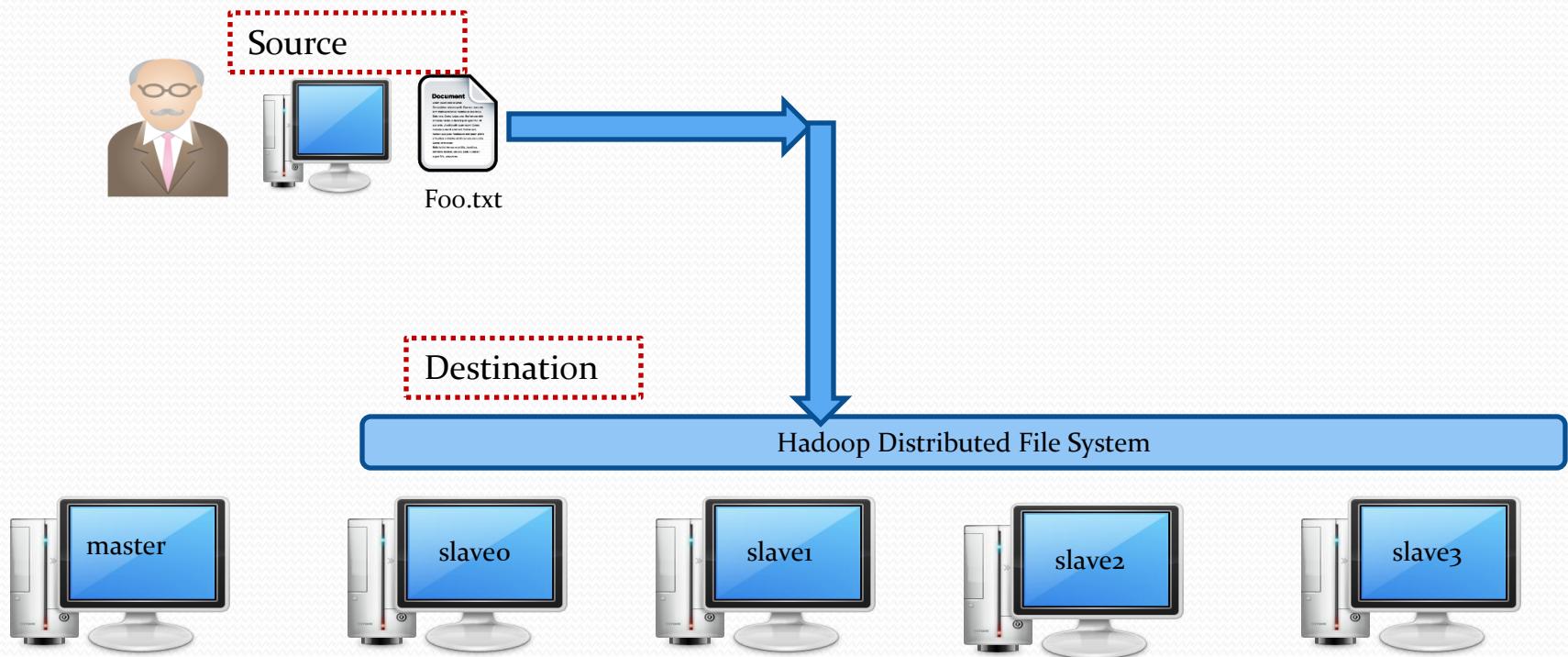
```
hadoop fs -ls foo
```

```
hadoop fs -ls foo/bar
```

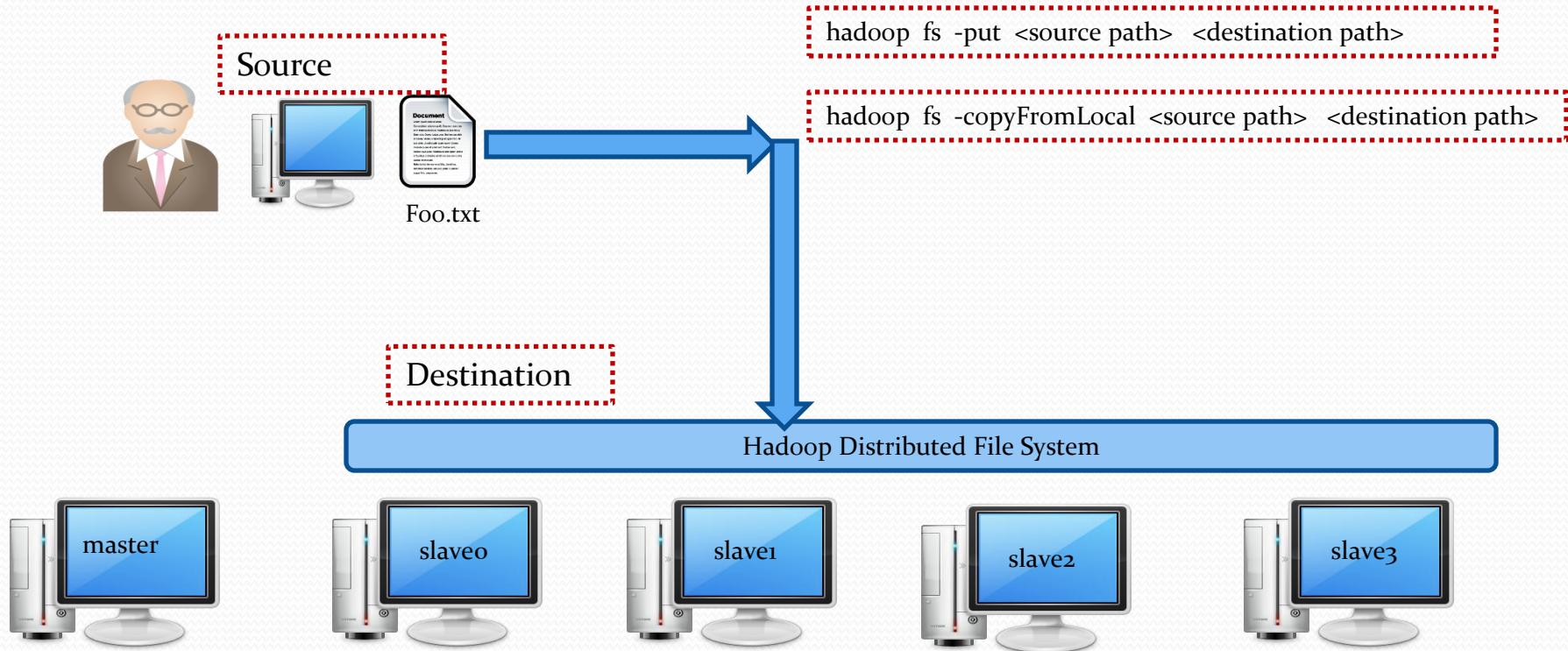
Putting a file on HDFS



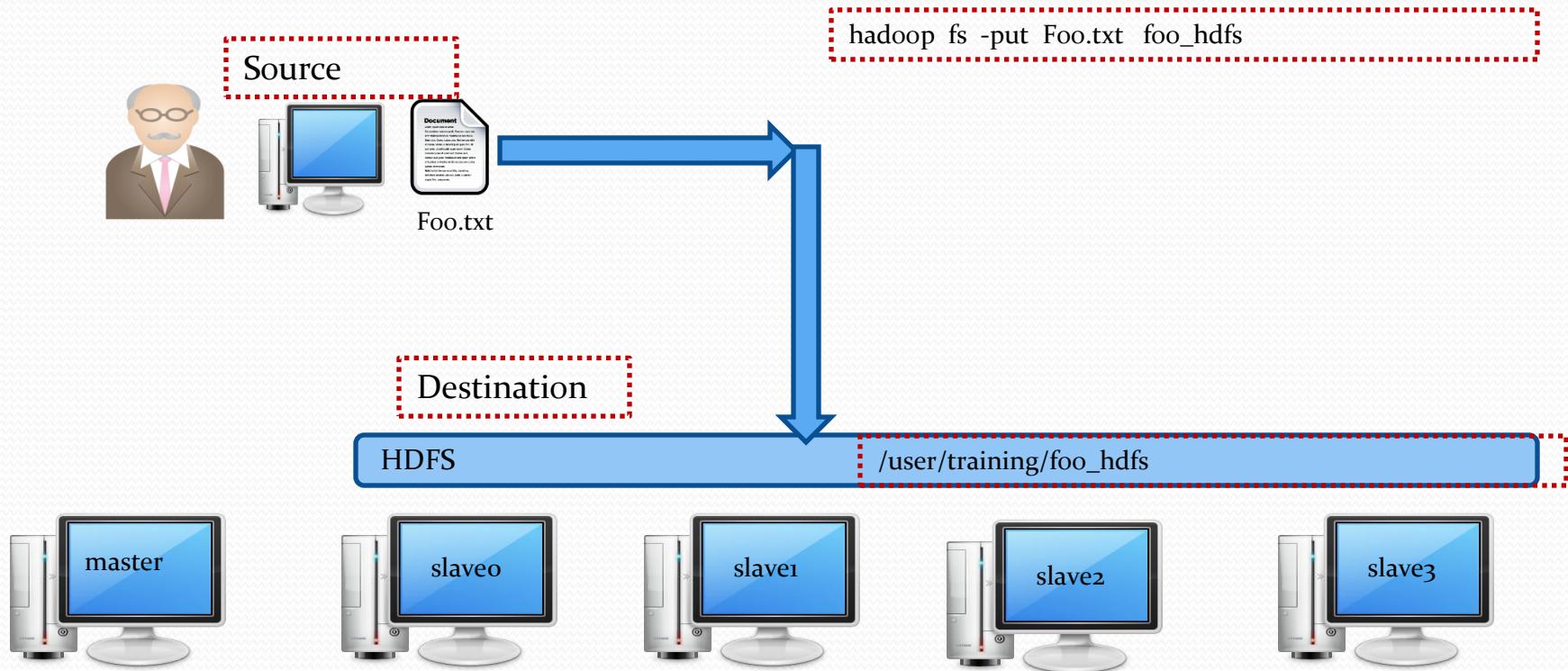
Putting a file on HDFS



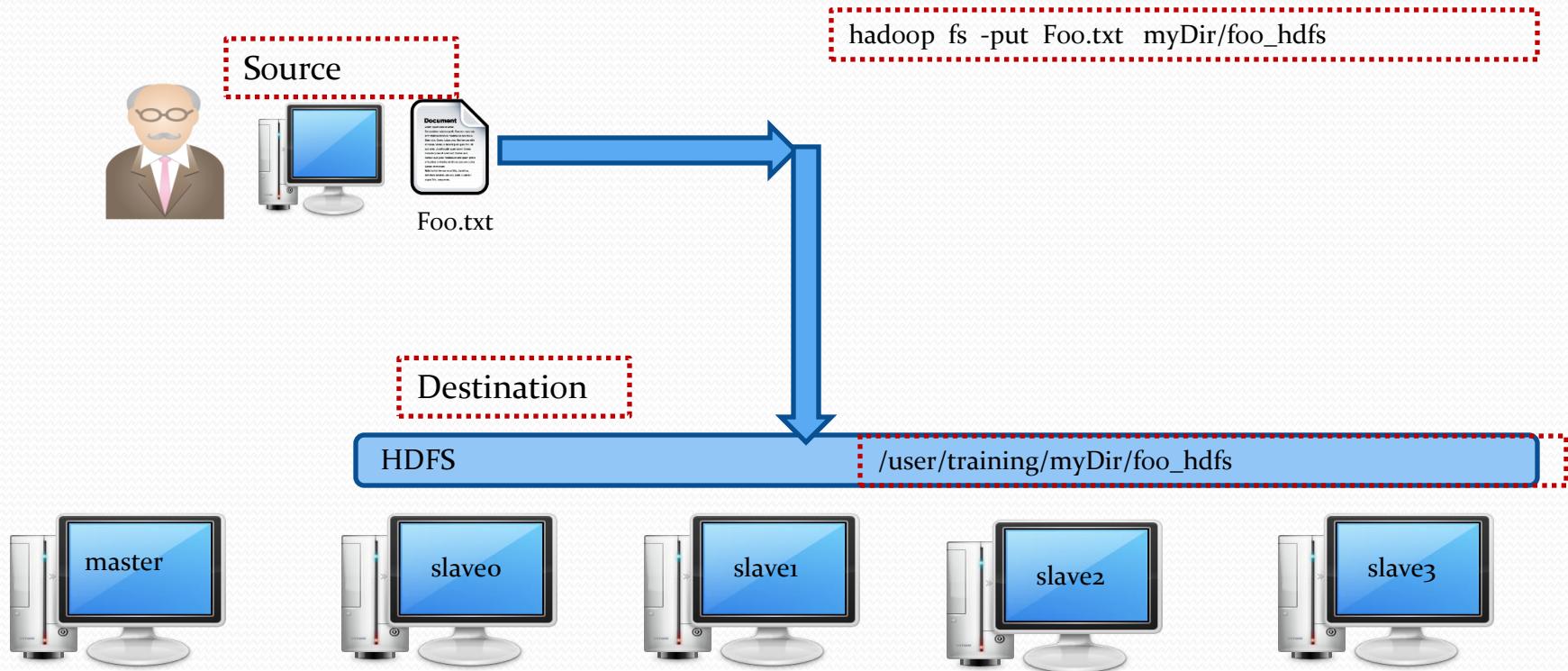
Putting a file on HDFS



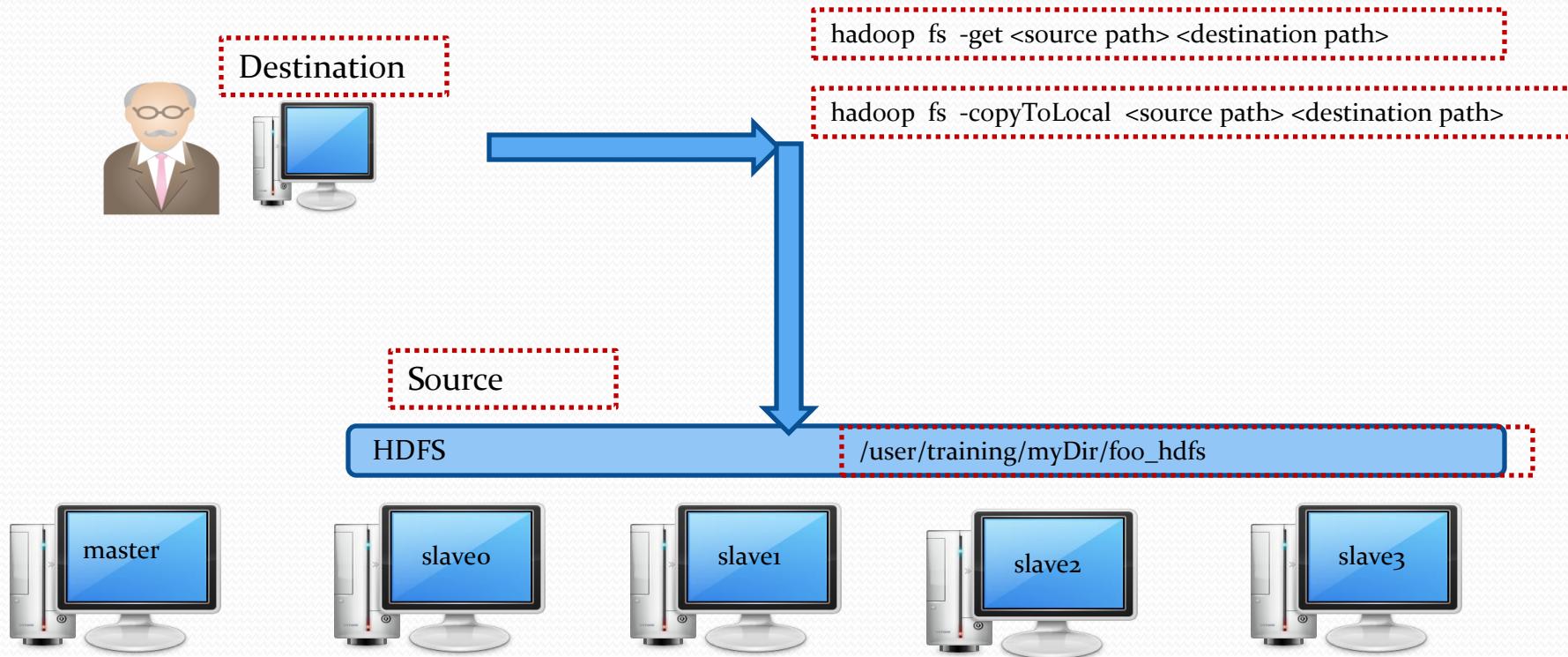
Putting a file on HDFS



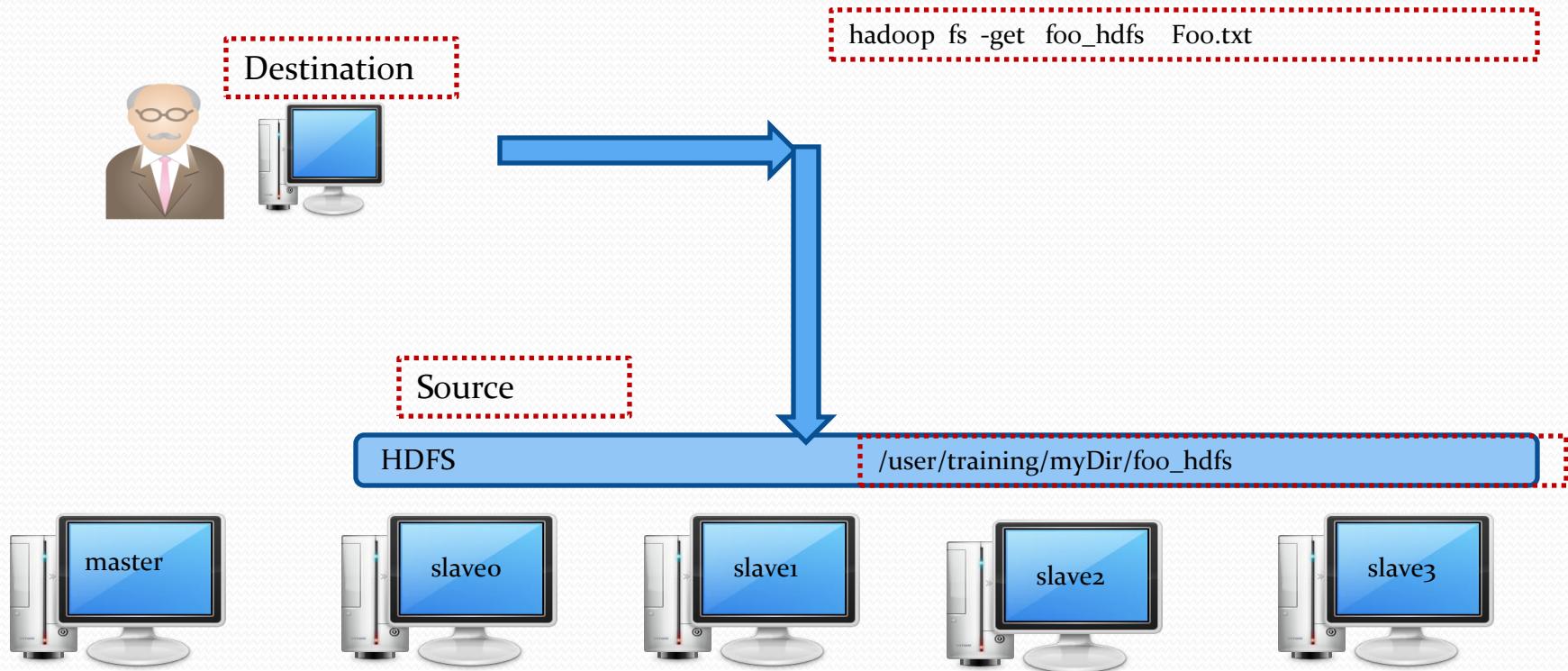
Putting a file on HDFS



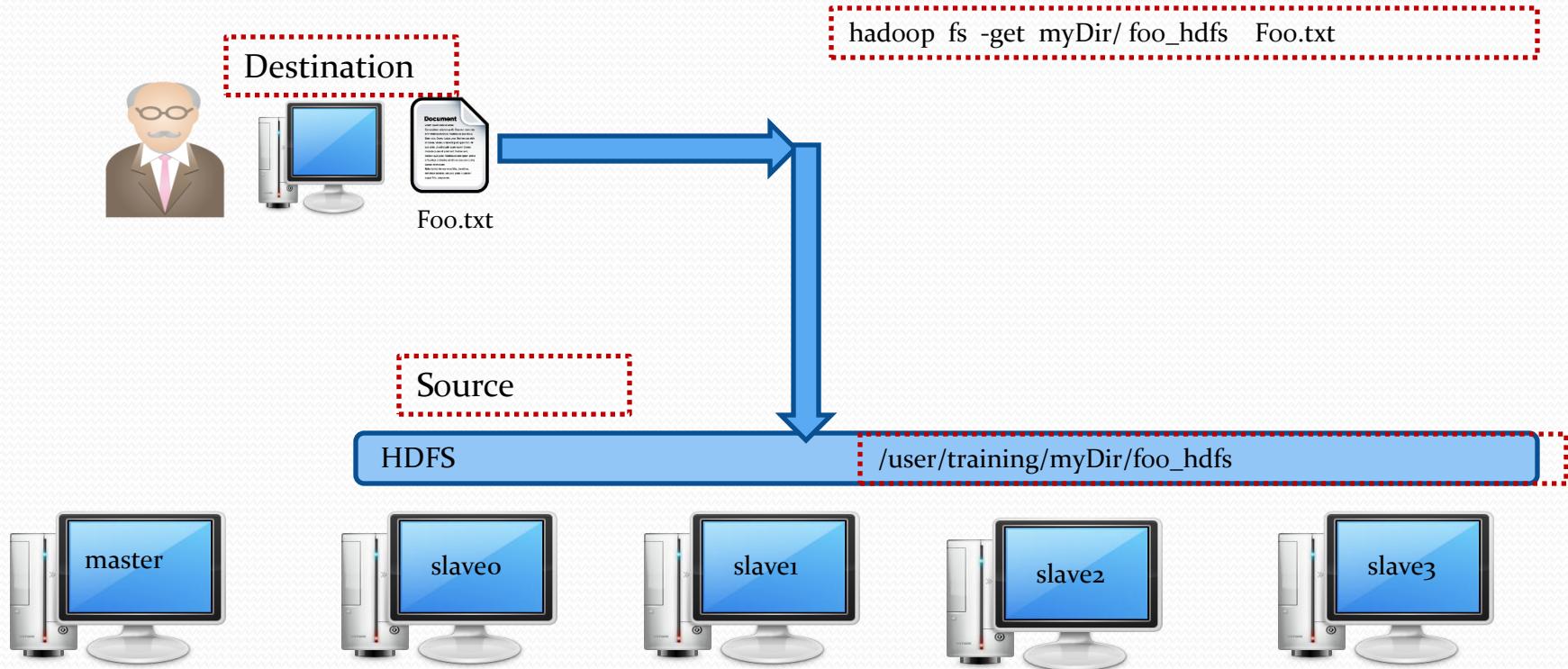
Getting a file from HDFS



Getting a file from HDFS



Getting a file from HDFS



Viewing the contents of file on console

```
hadoop fs -cat foo
```

Getting the blocks and its locations

```
hadoop fsck /user/training/sample -files -blocks -locations
```

- This will give the list of blocks which the file “*sample*” is made of and its location
- This command needs to be run from the machine where namenode process is running

Hands On

- Refer the Hands On document

Module 7

HDFS Components

In this module you will learn

- Various process running on Hadoop
- Working of NameNode
- Working of DataNode
- Working of Secondary NameNode
- Working of JobTracker
- Working of TaskTracker
- Writing a file on HDFS
- Reading a file from HDFS

Process Running on Hadoop

Master



- NameNode
- Secondary NameNode
- Job Tracker



Slave-0

- DataNode
- TaskTracker



Slave-1

- DataNode
- TaskTracker

How HDFS is designed?

- For Storing very large files -
- Scaling in terms of storage
- Mitigating Hardware failure
 - Failure is common rather than exception
- Designed for batch mode processing
- Write once read many times
- Build around commodity hardware
- Compatible across several OS (linux, windows, Solaris, mac)
- Highly fault tolerant system
 - Achieved through replica mechanism

How HDFS is designed?

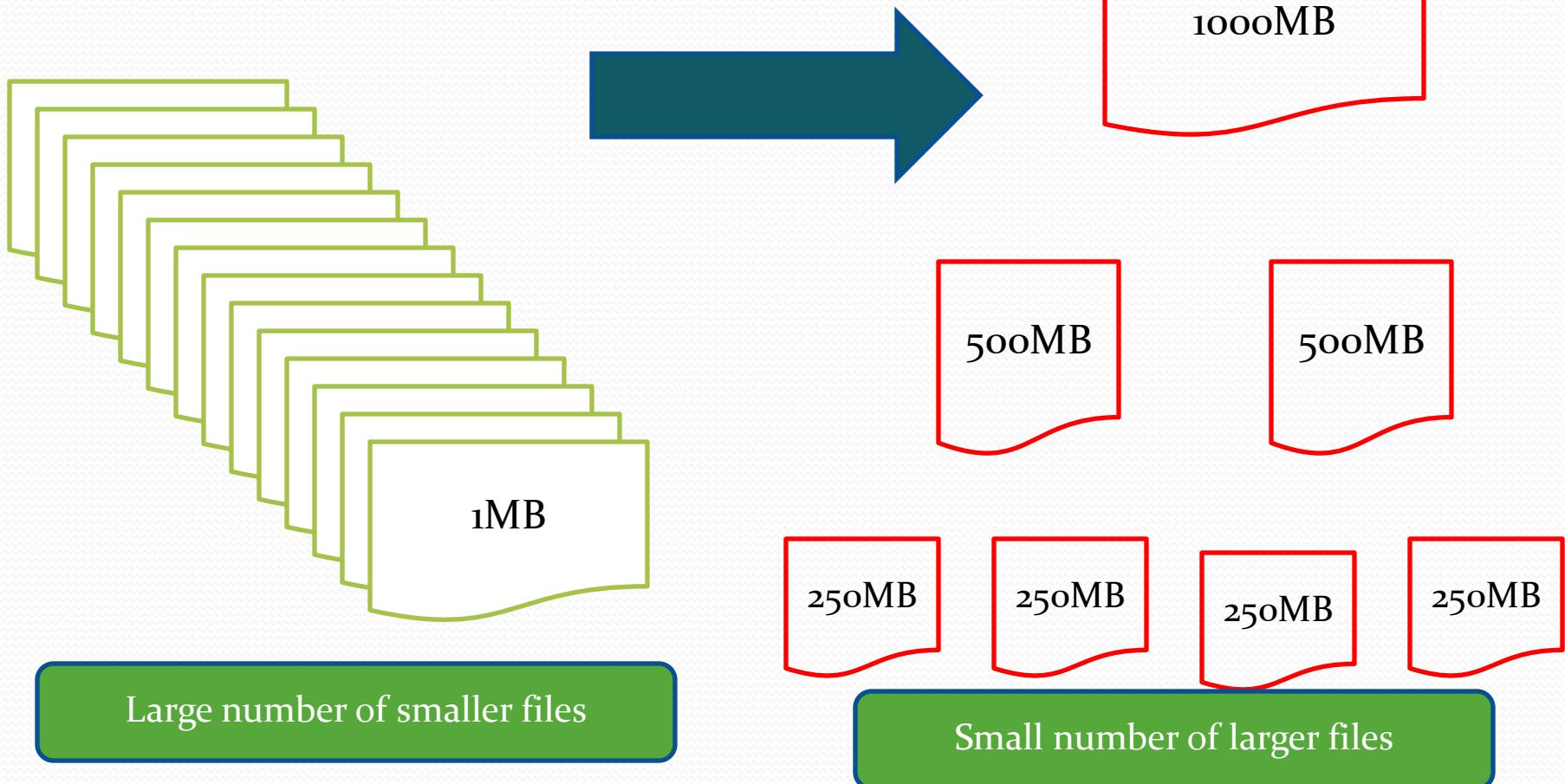
- For Storing very large files

Is it not suitable for storing small files?

What is small file and large file w.r.t Hadoop?

HDFS is efficient for storing small number of larger files rather than
large number of smaller files

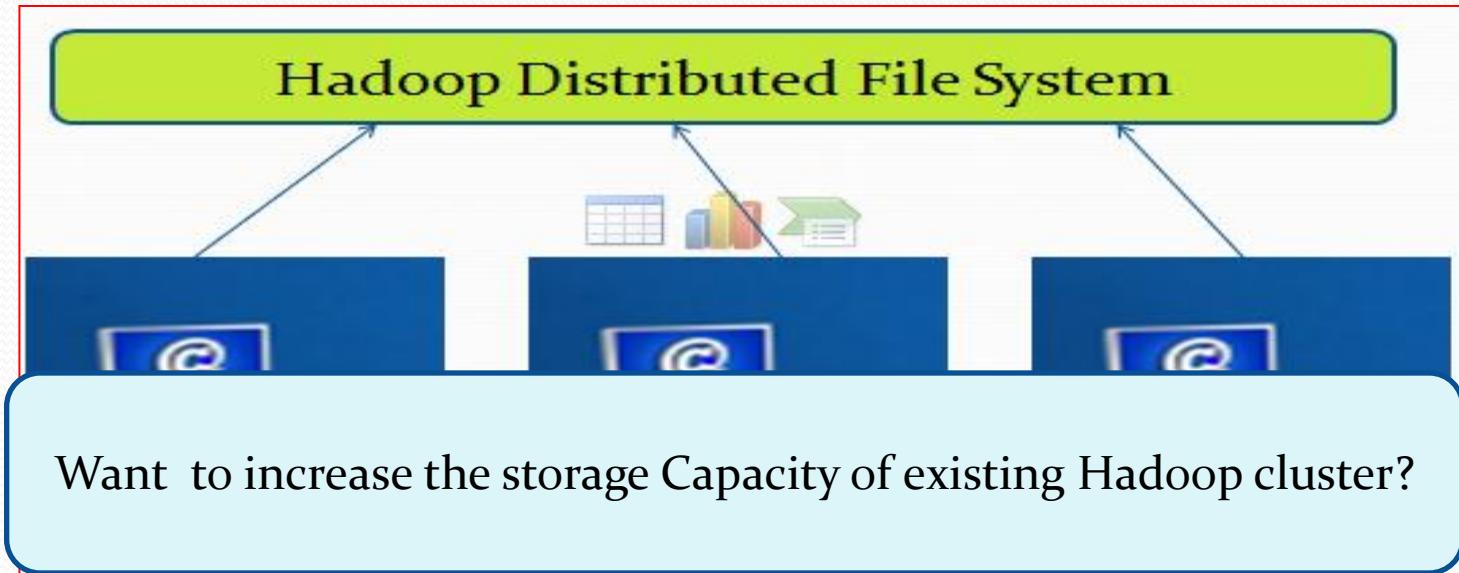
Small & large files cont'd



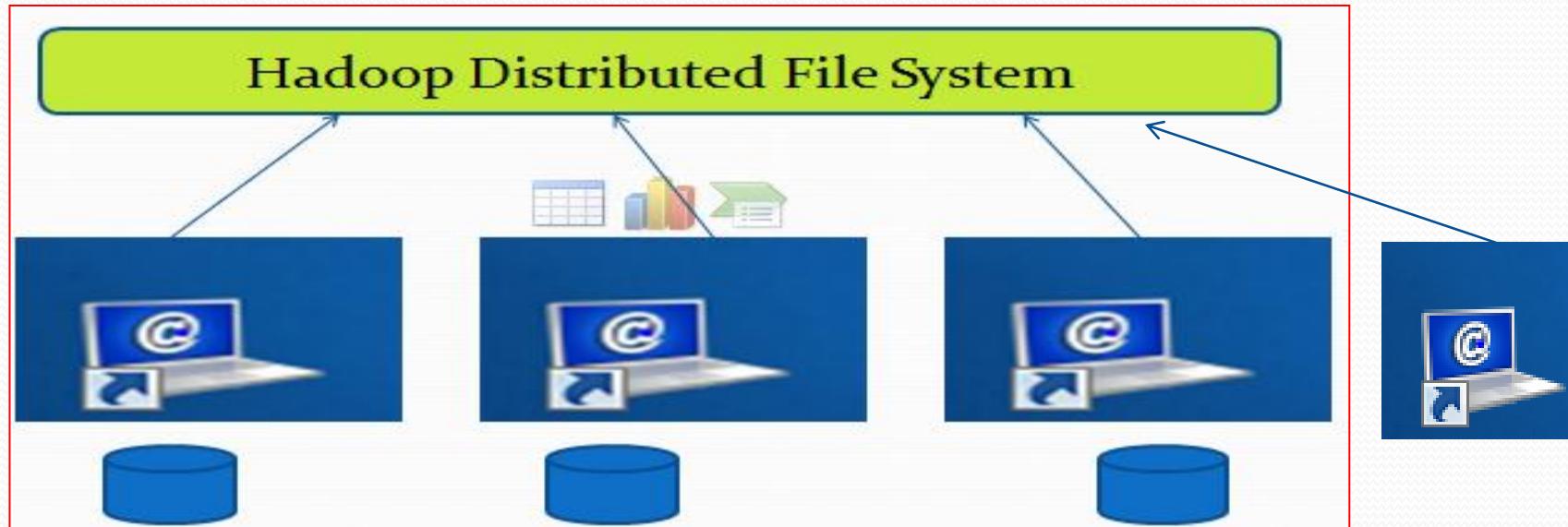
Storing Large Files

- Large files means size of file is greater than block size (64MB)
- Hadoop is efficient for storing large files but not efficient for storing small files
 - Small files means the file size is less than the block size
- Its better to store a bigger files rather than smaller files
 - Still if you are working with smaller files, merge smaller files to make a big file
- Smaller files have direct impact on MetaData and number of task
 - Increases the NameNode meta data
 - Lot of smaller files means lot of tasks.

Scaling in terms of storage



Scaling in terms of storage

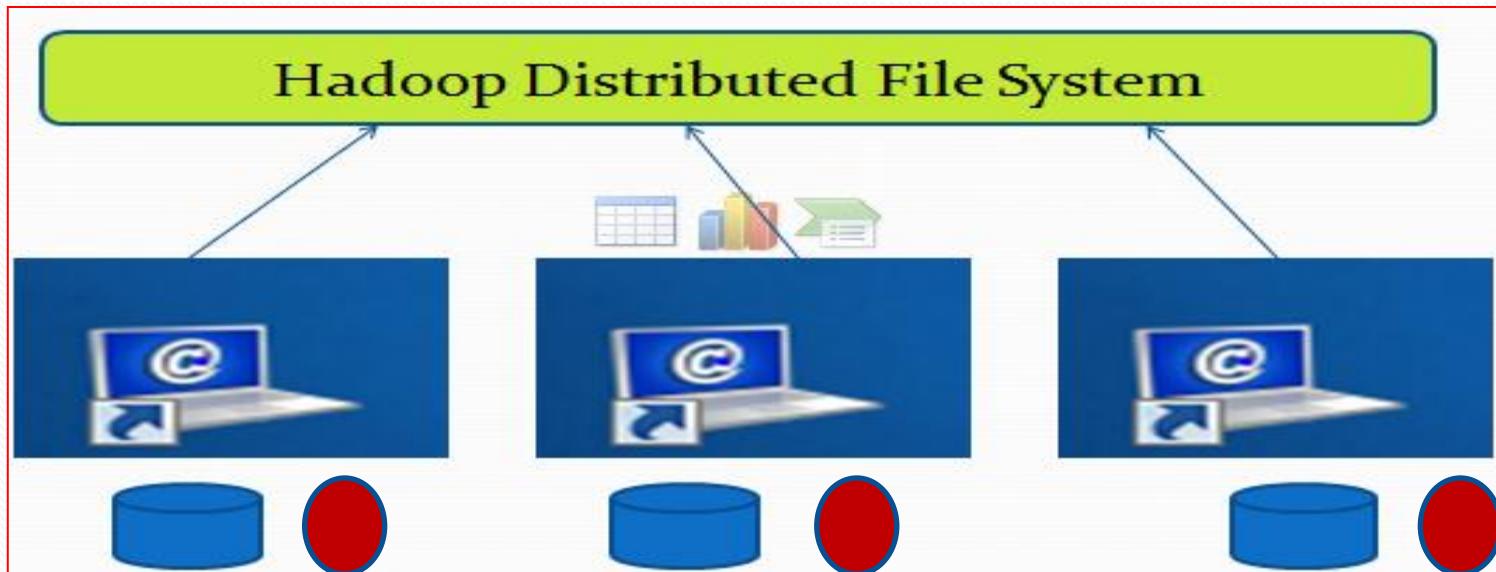


Add one more node to the existing cluster

Mitigating Hard Ware failure

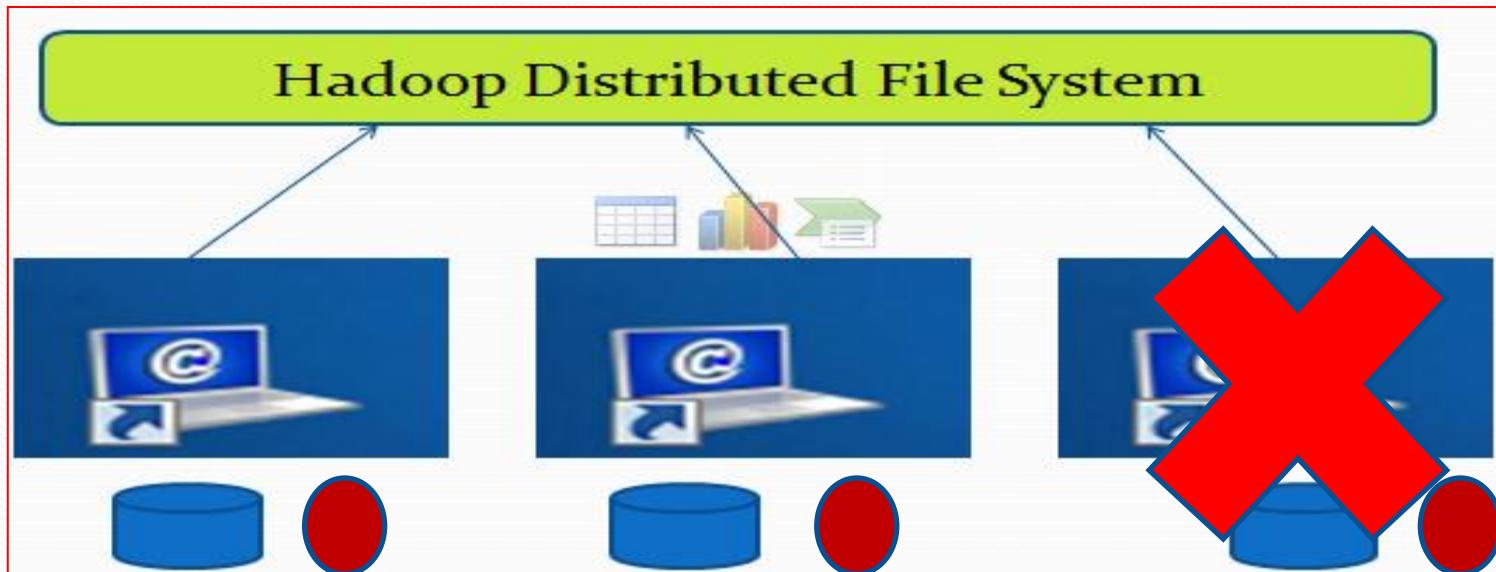
- HDFS is machine agnostic
- Make sure that data is not lost if the machines are going down
 - By replicating the blocks

Mitigating Hard Ware failure cont'd



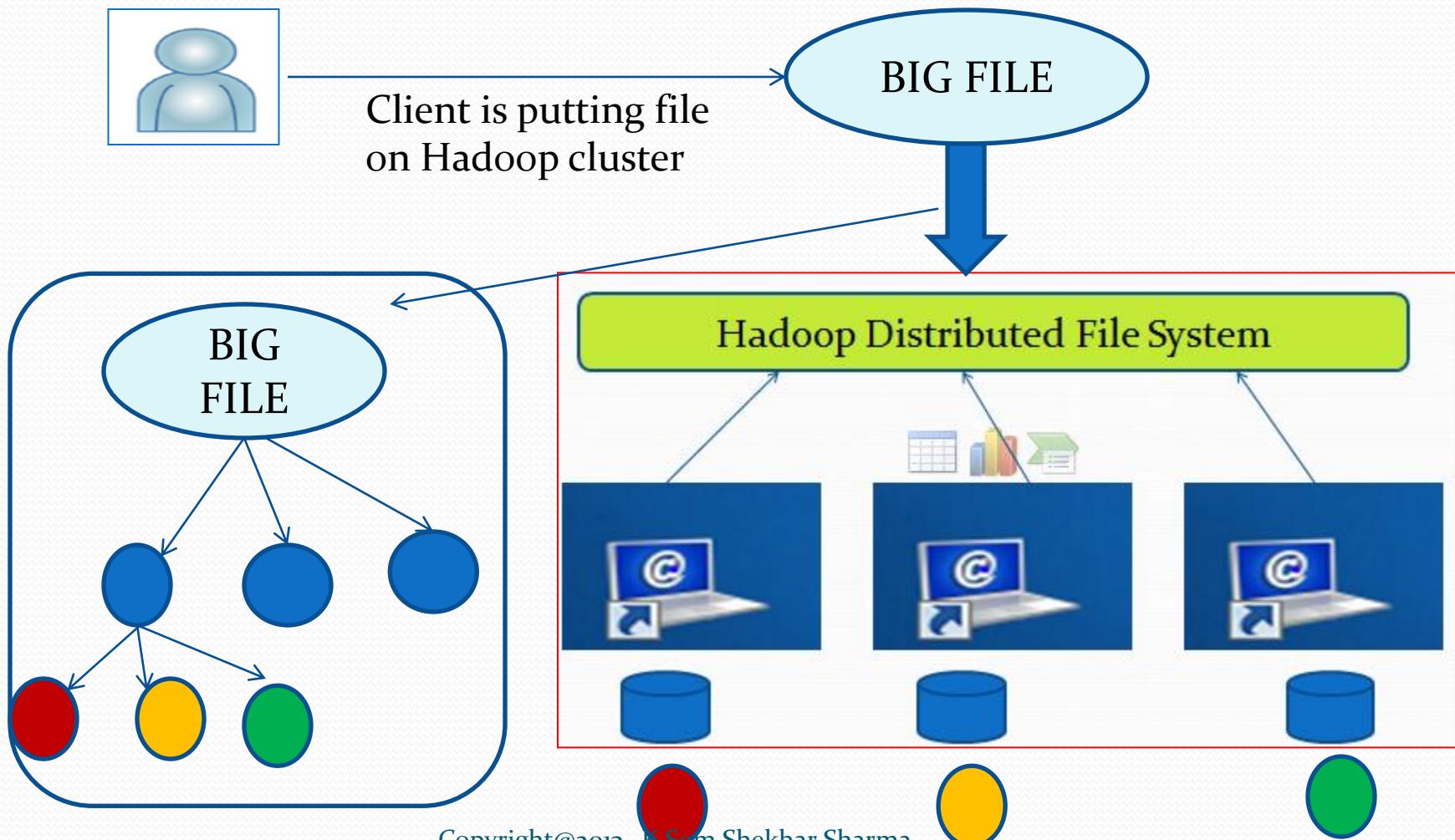
- Replica of the block is present in three machines (Default replication factor is 3)
- Machine can fail due to numerous reasons
 - Faulty Machine
 - N/W failure, etc

Mitigating Hard Ware failure cont'd

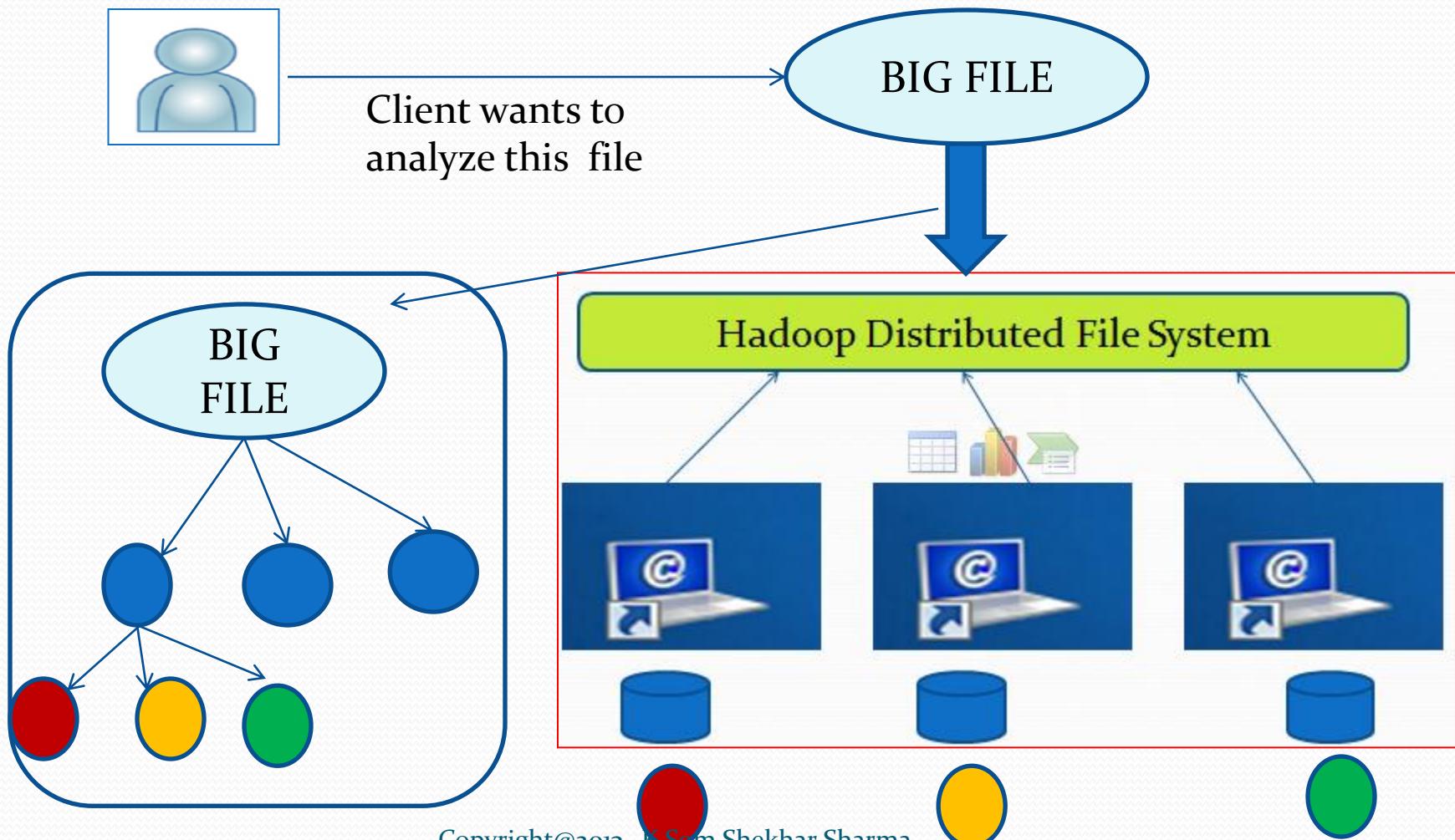


- Replica is still intact in some other machine
- NameNode try to recover the lost blocks from the failed machine and bring back the replication factor to normal (More on this later)

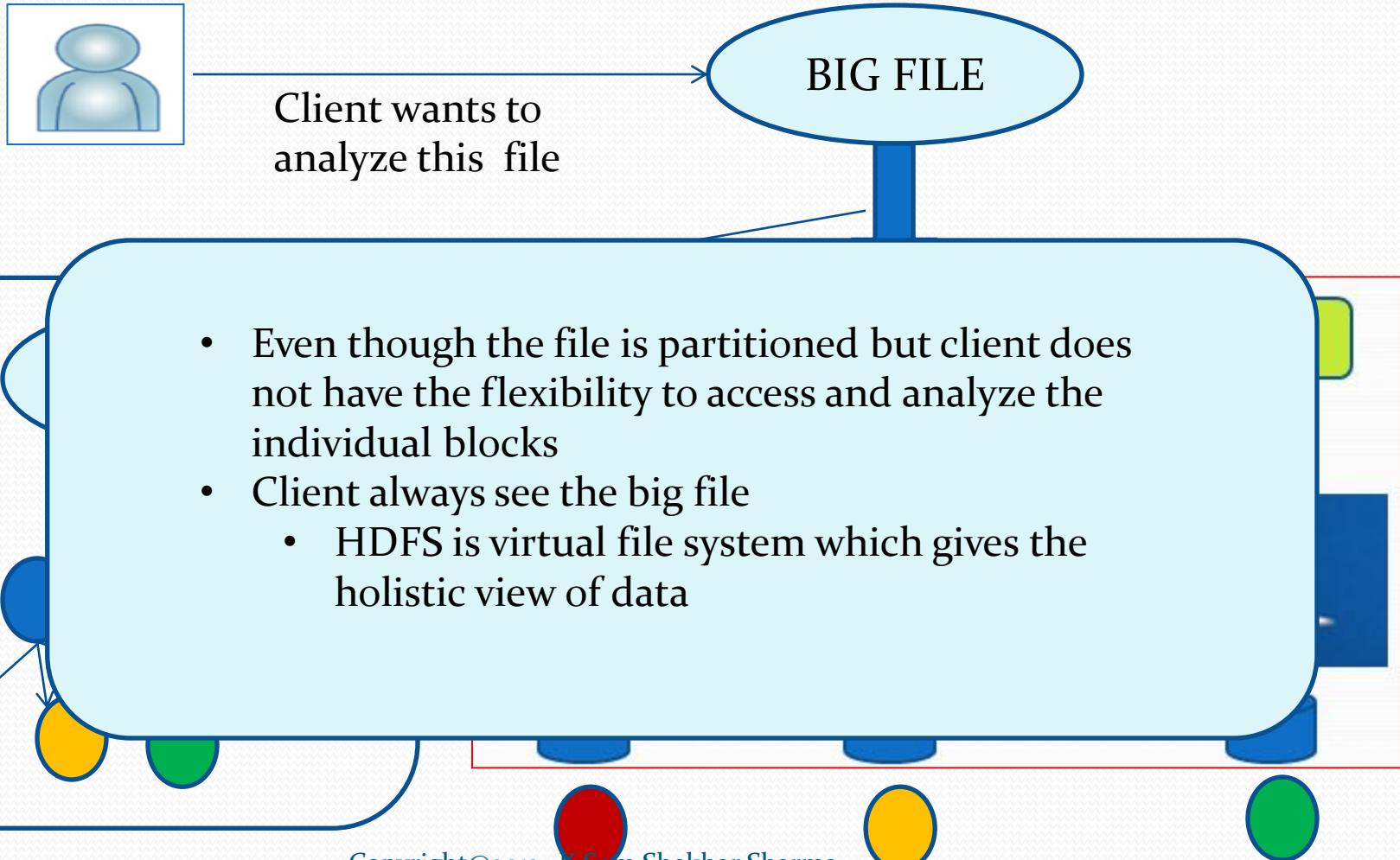
Batch Mode Processing



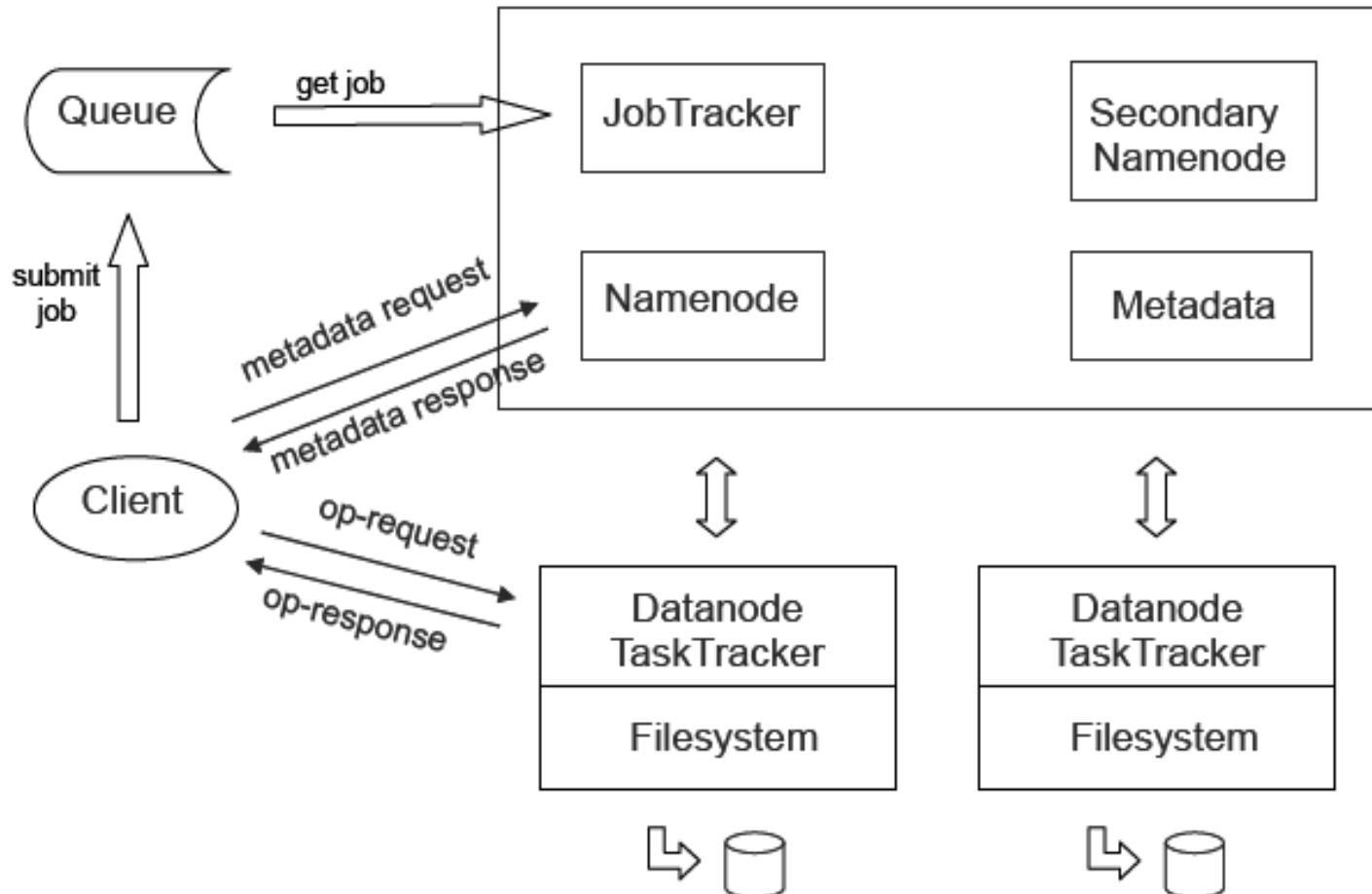
Batch Mode Processing cont'd



Batch Mode Processing cont'd



High level Overview

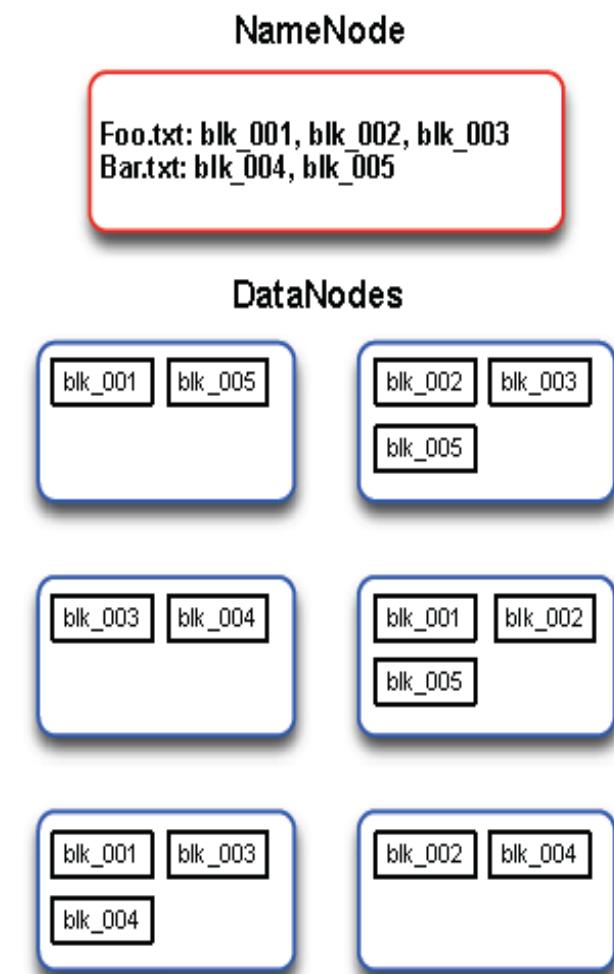


NameNode

- Single point of contact to the outside world
 - Client should know where the name node is running
 - Specified by the property *fs.default.name*
- Stores the meta data
 - List of files and directories
 - Blocks and their locations
- For fast access the meta data is kept in RAM
 - Meta data is also stored persistently on local file system
 - */home/training/hadoop-temp/dfs/name/previous.checkpoint/fsimage*

NameNode cont'd

- Meta Data consist of mapping Of files to the block
- Data is stored with Data nodes



NameNode cont'd

- If NameNode is down, HDFS is inaccessible
 - Single point of failure
- Any operation on HDFS is recorded by the NameNode
 - */home/training/hadoop-temp/dfs/name/previous.checkpoint/edits* file
- Name Node periodically receives the heart beat signal from the data nodes
 - If NameNode does not receives heart beat signal, it assumes the data node is down
 - NameNode asks other alive data nodes to replicate the lost blocks

DataNode

- Stores the actual data
 - Along with data also keeps a meta file for verifying the integrity of the data
 - */home/training/hadoop-temp/dfs/data/current*
- Sends heart beat signal to NameNode periodically
 - Sends block report
 - Storage capacity
 - Number of data transfers happening
 - Will be considered down if not able to send the heart beat
- NameNode never contacts DataNodes directly
 - Replies to heart beat signal

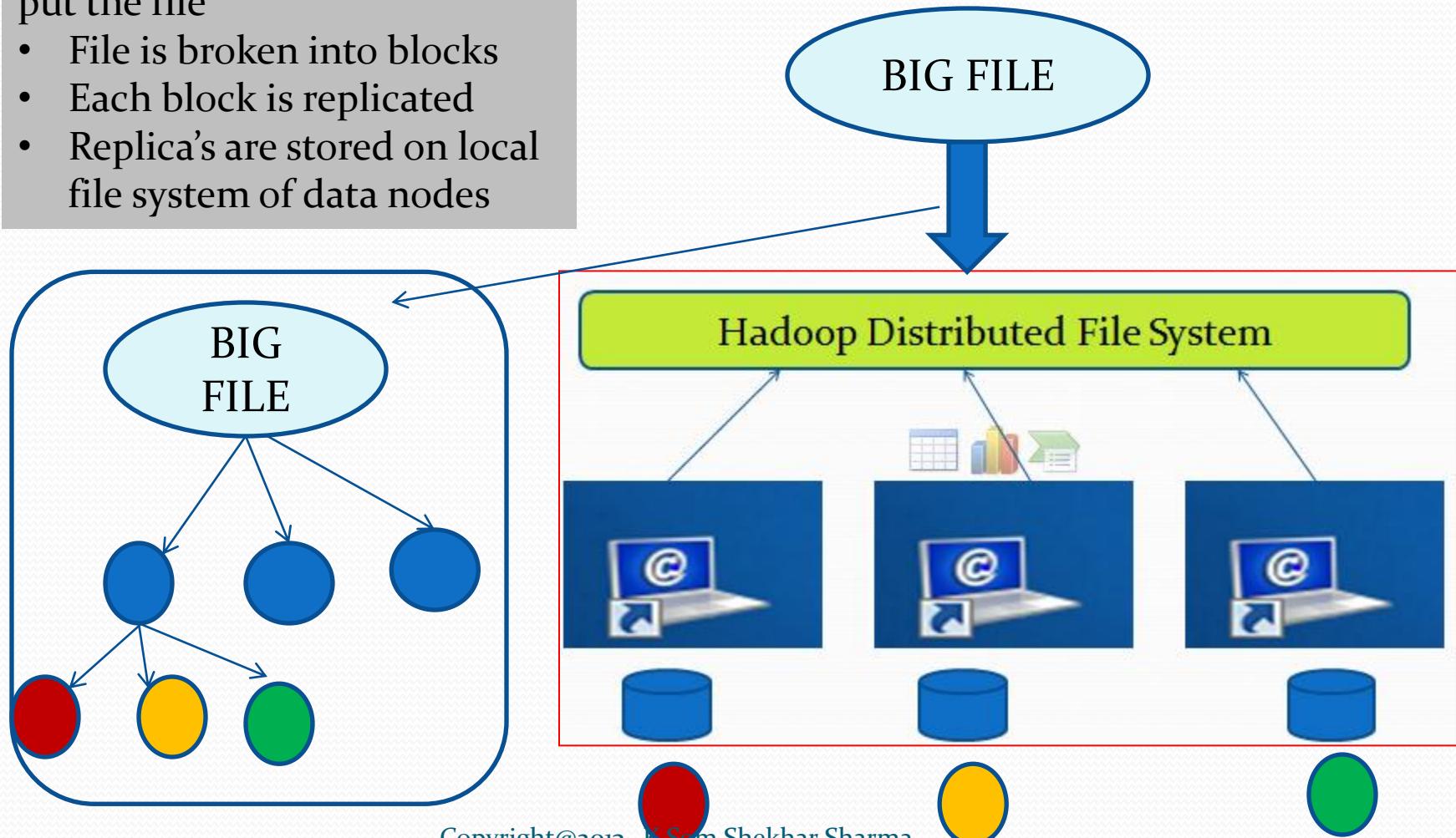
DataNode cont'd

- Data Node receives following instructions from the name node as part of heart beat signal
 - Replicate the blocks (In case of under replicated blocks)
 - Remove the local block replica (In case of over replicated blocks)
- NameNode makes sure that replication factor is always kept to normal (default 3)

Recap- How a file is stored on HDFS

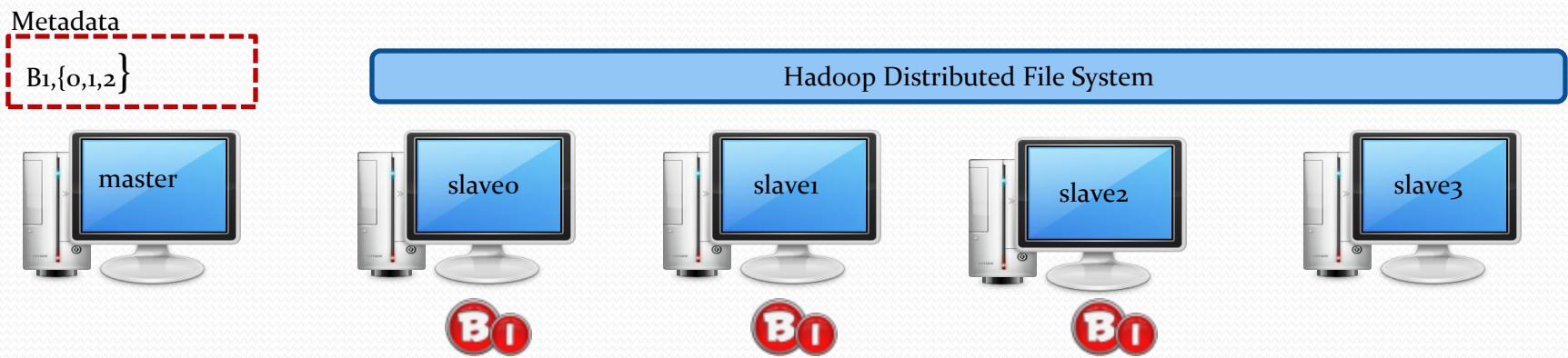
Behind the scenes when you put the file

- File is broken into blocks
- Each block is replicated
- Replica's are stored on local file system of data nodes



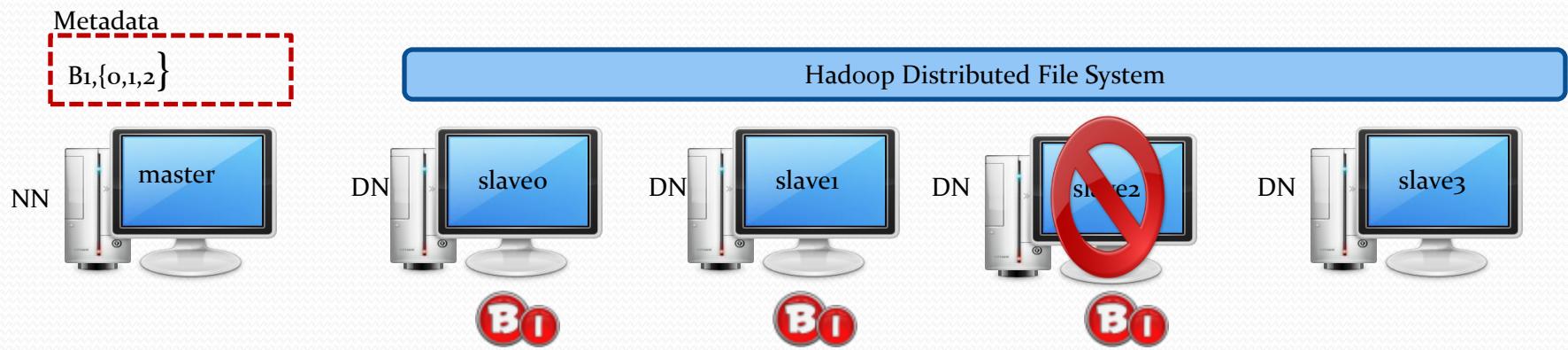
Normal Replication factor

Number of replica = 3



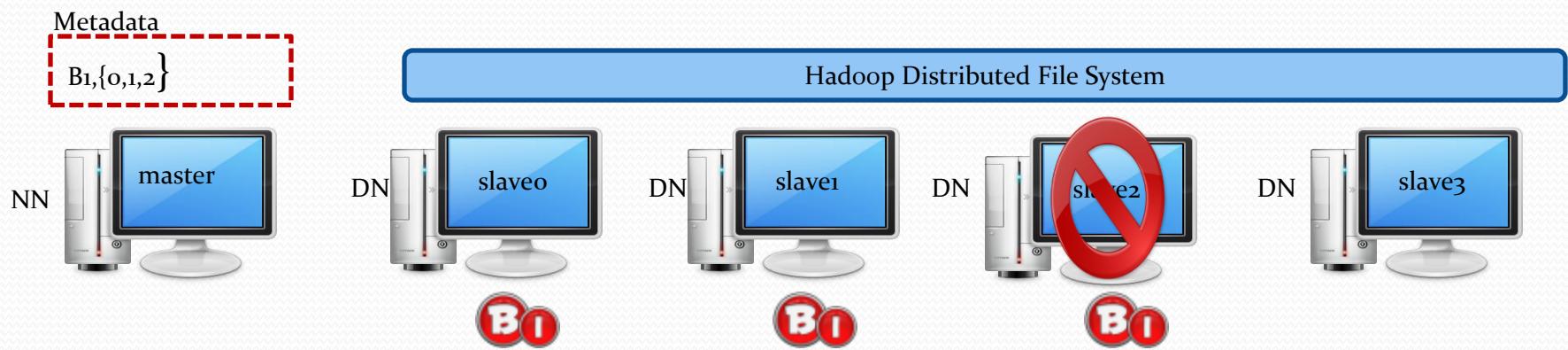
Normal Replication factor

Number of replica = 3



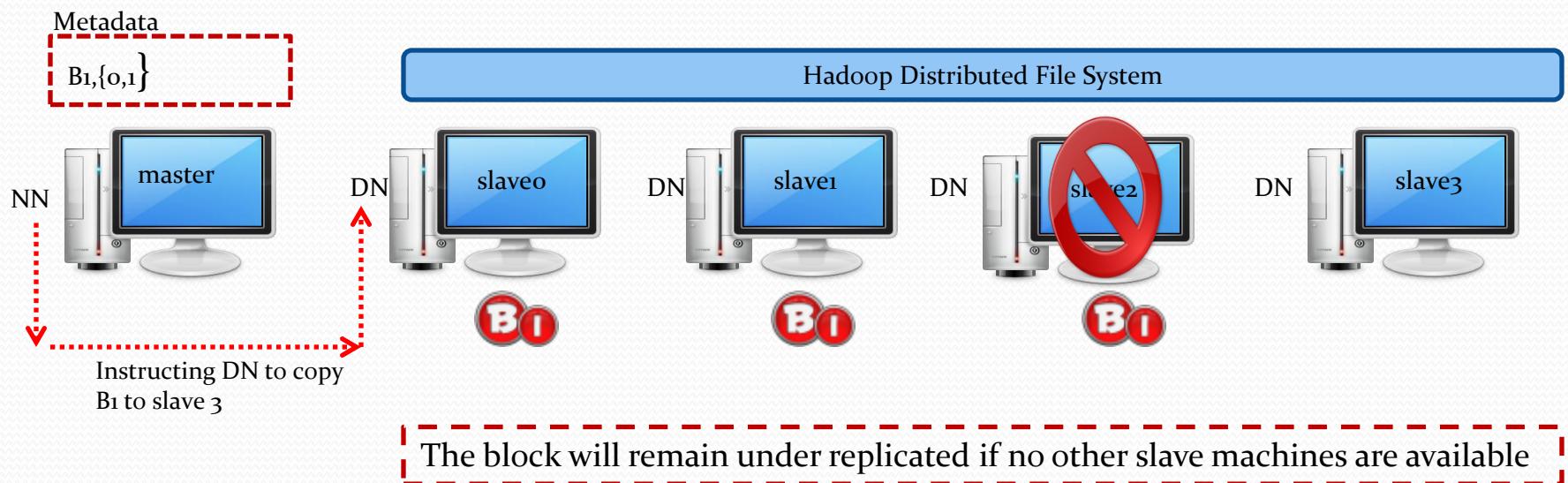
Under Replication factor

Normal is 3
Current number of replica=2



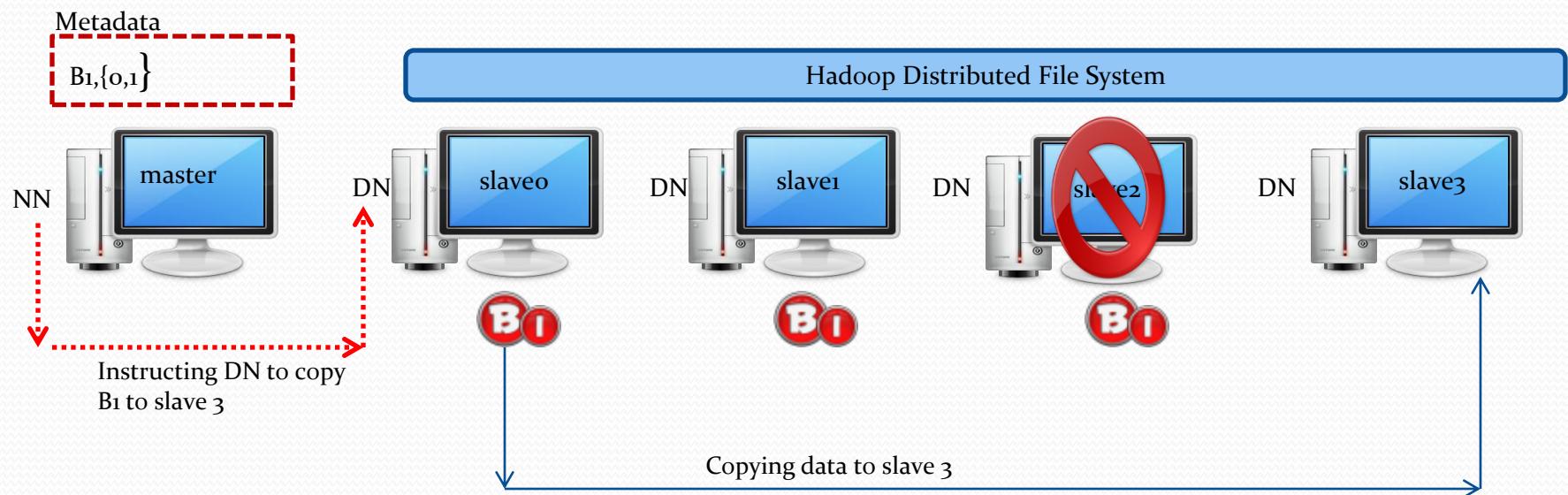
Under Replication factor

Normal is 3
Current number of replica=2



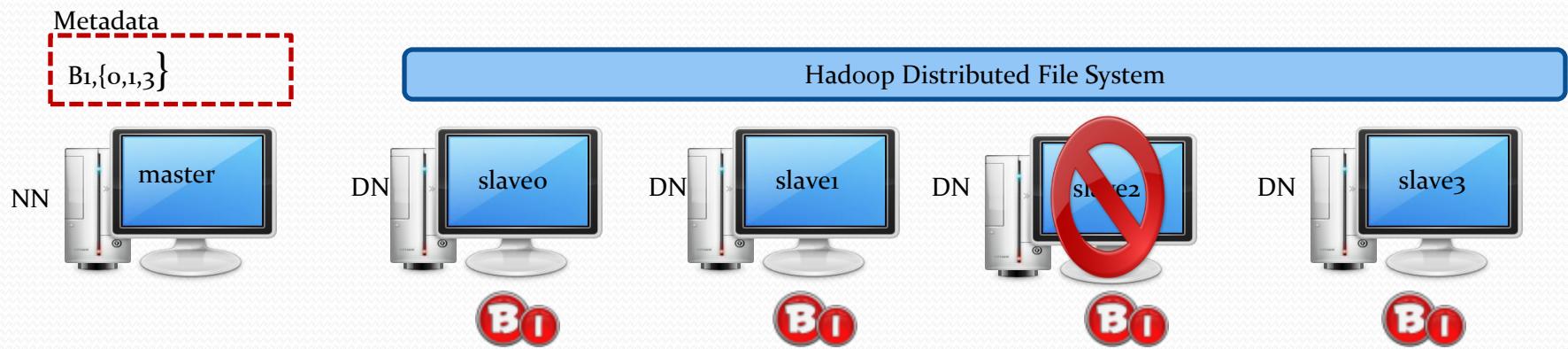
Under Replication factor

Normal is 3
Current number of replica=2



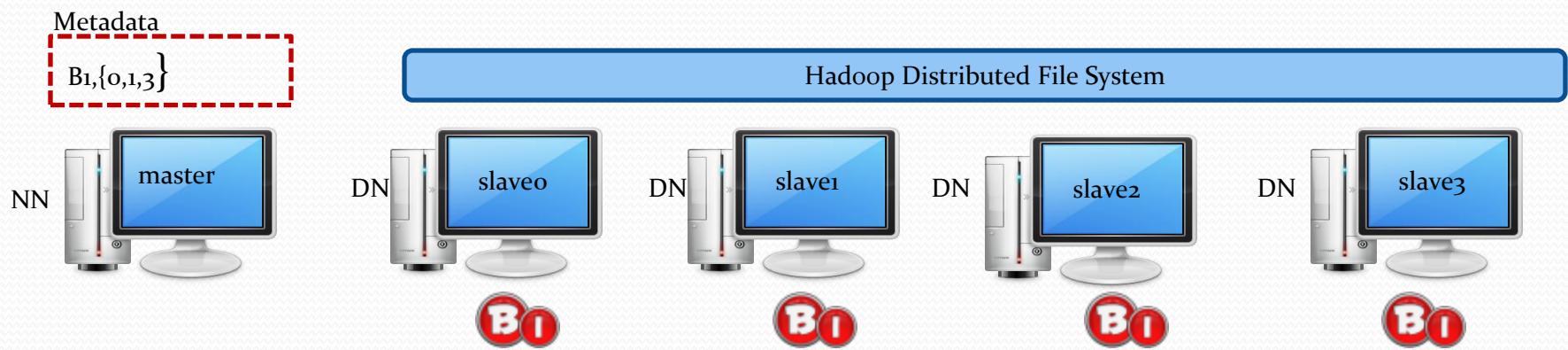
Under Replication factor

Replication factor back to normal



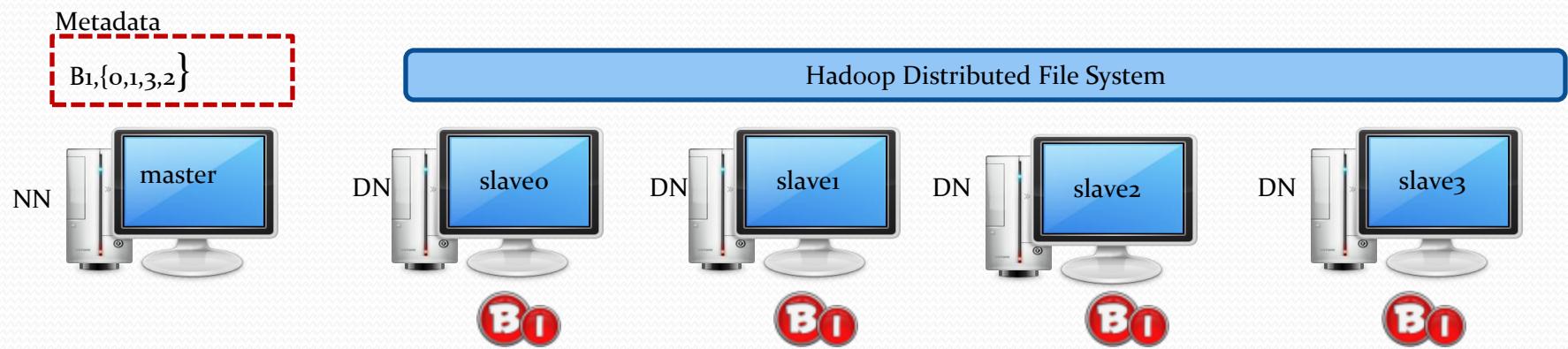
Over Replication factor

Normal is 3
Current number of replica=4

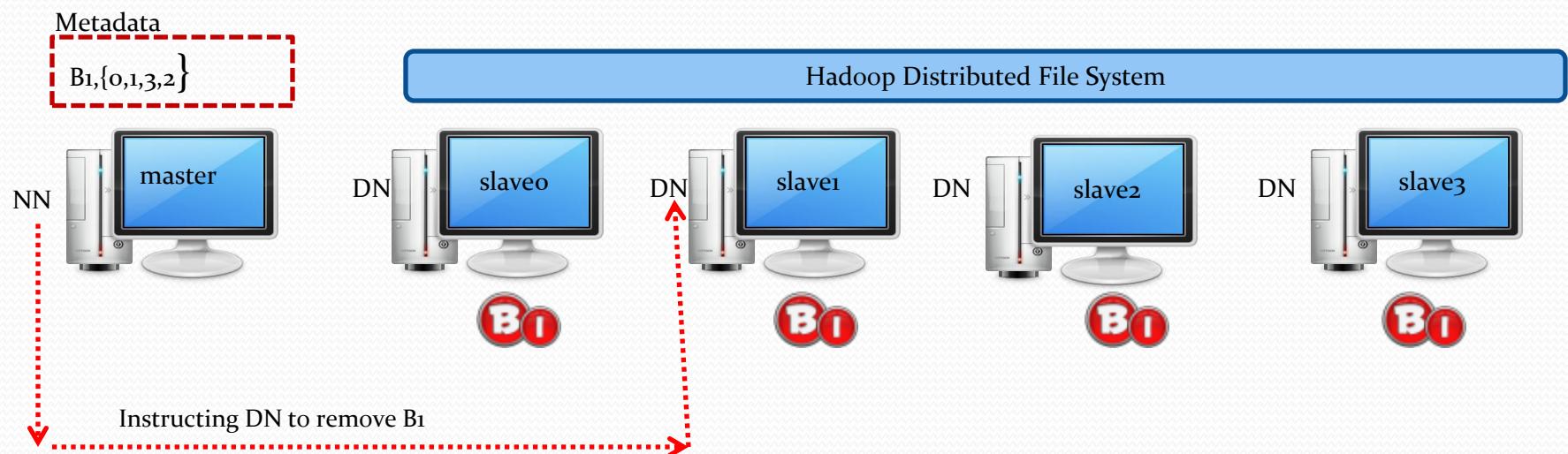


Over Replication factor

Normal is 3
Current number of replica=4

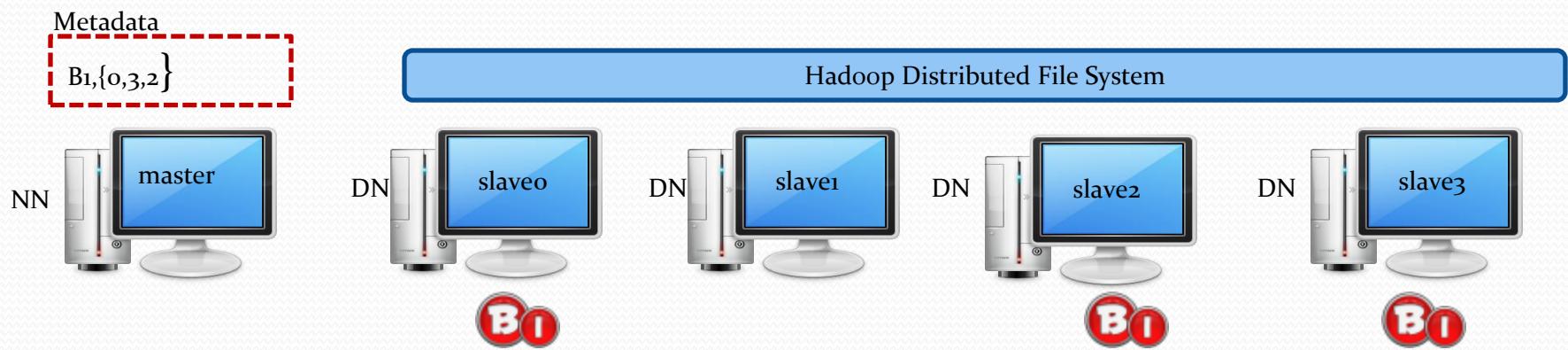


Over Replication factor



Over Replication factor

Replication factor back to normal



Secondary NameNode cont'd

- Not a back up or stand by NameNode
- Only purpose is to take the snapshot of NameNode and merging the log file contents into metadata file on local file system
- It's a CPU intensive operation
 - In big cluster it is run on different machine

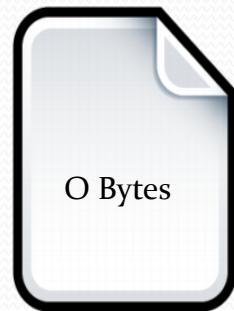
Secondary NameNode cont'd

- Two important files (present under this directory `/home/training/hadoop-temp/dfs/name/previous.checkpoint`)
 - Edits file
 - Fsimage file
- When starting the Hadoop cluster (`start-all.sh`)
 - Restores the previous state of HDFS by reading *fsimage* file
 - Then starts applying modifications to the meta data from the *edits* file
 - Once the modification is done, it empties the *edits* file
 - This process is done only during start up

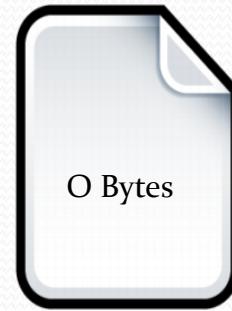
Secondary NameNode cont'd

- Initial Set Up

Single node Hadoop Set up



fsimage



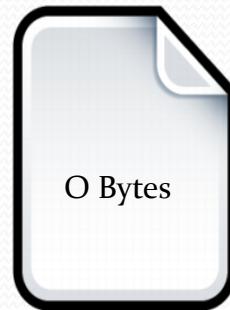
Edits

These files are present under ~/hadoop-temp/dfs/name/previous.checkpoint

Secondary NameNode cont'd

- start-all.sh

Single node Hadoop Set up



fsimage



Edits

Secondary NameNode cont'd

- Step1: Restore previous HDFS state by reading fsimage file. File is empty, so nothing will happen



Single node Hadoop Set up



fsimage



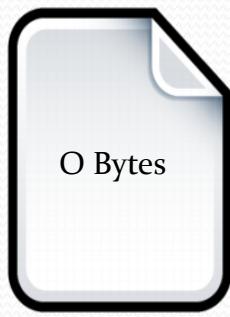
Edits

Secondary NameNode cont'd

- Step2: Start applying modification from the edits file.



Single node Hadoop Set up



fsimage



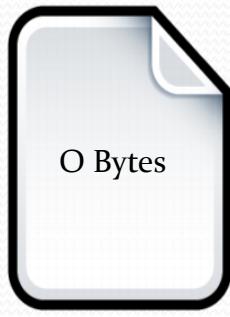
Edits

Secondary NameNode cont'd

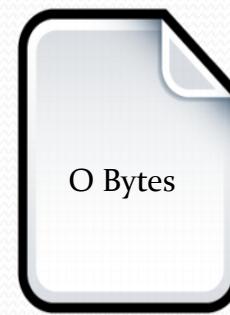
- Step3: Once the modification is done, edits file is emptied and merged with fsimage file



Single node Hadoop Set up



fsimage



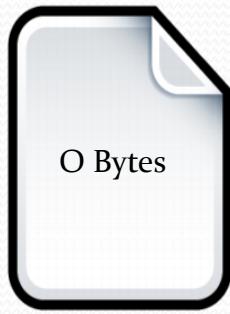
Edits

Secondary NameNode cont'd

- Hadoop will be in safemode during this operation



Single node Hadoop Set up



fsimage



Edits

Secondary NameNode cont'd

- Hadoop is out of safemode and you started doing operation.



Single node Hadoop Set up



fsimage



Edits

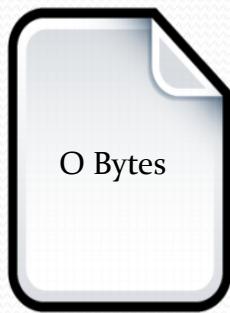
Operation which changes the HDFS state will be recorded and edits file will increase

Secondary NameNode cont'd

- Hadoop is out of safemode and you started doing operation.



Single node Hadoop Set up



fsimage



Edits

Operation which changes the HDFS state will be recorded and edits file will increase

Secondary NameNode cont'd

- After some time, edits file size will start increasing



Single node Hadoop Set up



fsimage



Edits

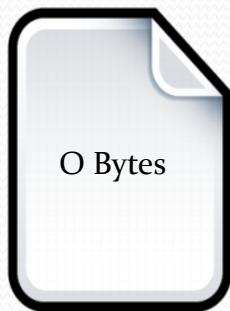
Operation which changes the HDFS state will be recorded and edits file will increase

Secondary NameNode cont'd

- stop-all.sh



Single node Hadoop Set up



fsimage



Edits

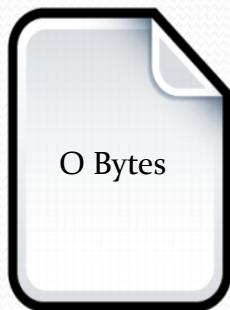
Operation which changes the HDFS state will be recorded and edits file will increase

Secondary NameNode cont'd

- start-all.sh



Single node Hadoop Set up



fsimage



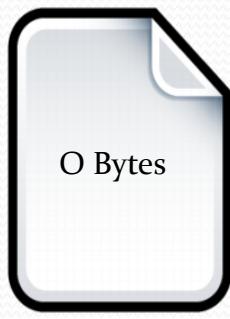
Edits

Secondary NameNode cont'd

- step1: Restore the previous HDFS state by reading fsimage. Nothing is there and it would be faster



Single node Hadoop Set up



fsimage



Edits

Secondary NameNode cont'd

- step2. Starts applying modification from edits file



Single node Hadoop Set up



fsimage



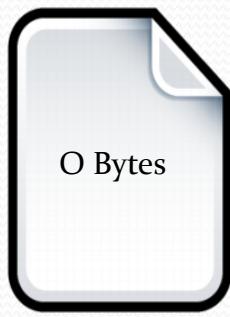
Edits

Secondary NameNode cont'd

- step3. Once the modification is done, empty the edits file and merge with the fsimage file. CPU intensive operation. It will remain in safemode for little longer amount of time



Single node Hadoop Set up



fsimage



Edits

Secondary NameNode cont'd

- During this period Hadoop will be in safe mode



Single node Hadoop Set up



fsimage



Edits

Secondary NameNode cont'd

- Edits file can grow big over a period of time. Next restart will take too much of time



Single node Hadoop Set up



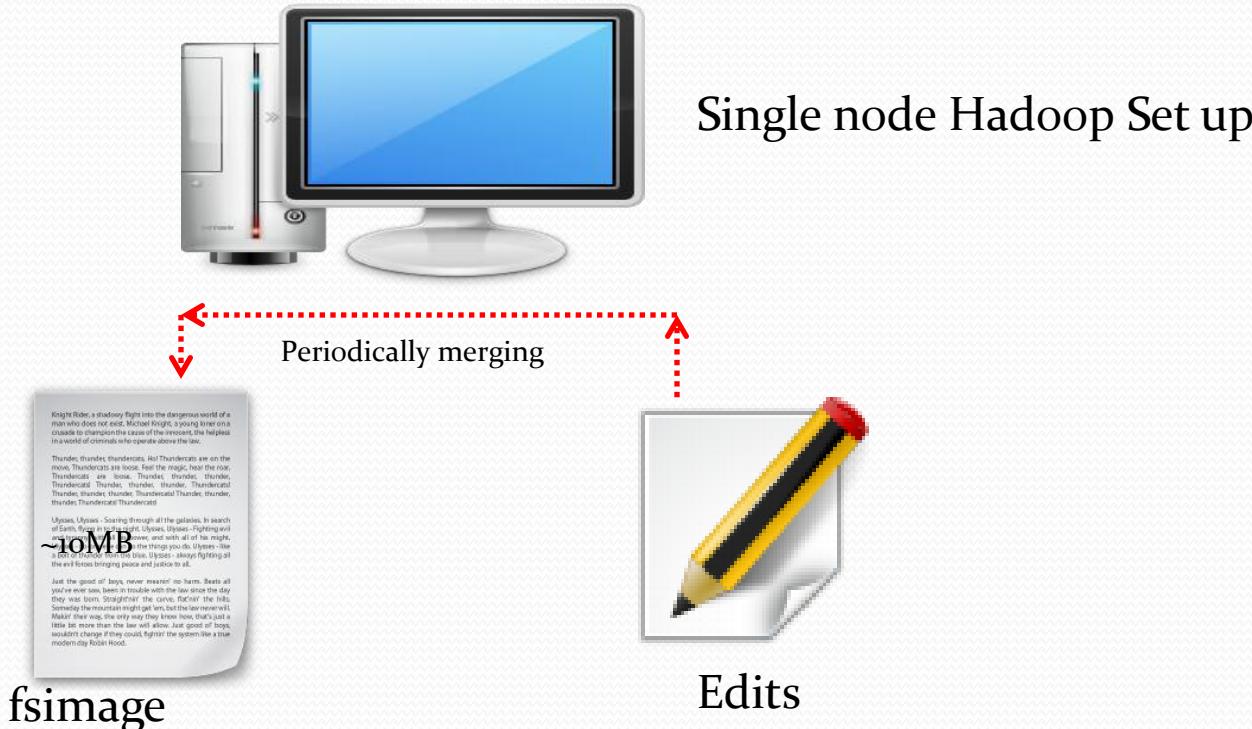
fsimage



Edits

Secondary NameNode cont'd

- SNN periodically merges the edits file with the fsimage file to keep the size in sizeable limit



Job Tracker

- MapReduce master
- Client submits the job to JobTracker
- JobTracker talks to the NameNode to get the list of blocks
- Job Tracker locates the task tracker on the machine where data is located
 - Data Localization
- Job Tracker then first schedules the mapper tasks
- Once all the mapper tasks are over it runs the reducer tasks

Job Tracker



Master



Slave0



Slave1

Job Tracker



Master



Slave0



Slave1

Job Tracker



>> I have big data with me and I would like to put my data onto HDFS which can be processed later by my another team



Master



Slave0



Slave1

Job Tracker



>>What should I know in order to put my file on to HDFS?

>> Whom I need to contact?

>>Which command do I need to use?



Master



Slave0



Slave1

Job Tracker



```
>>hadoop fs -put sample  
sample
```



Master



Slave0



Slave1

Job Tracker



```
>>hadoop fs -put sample  
sample
```



NN
SNN
JT



Master

DN
TT



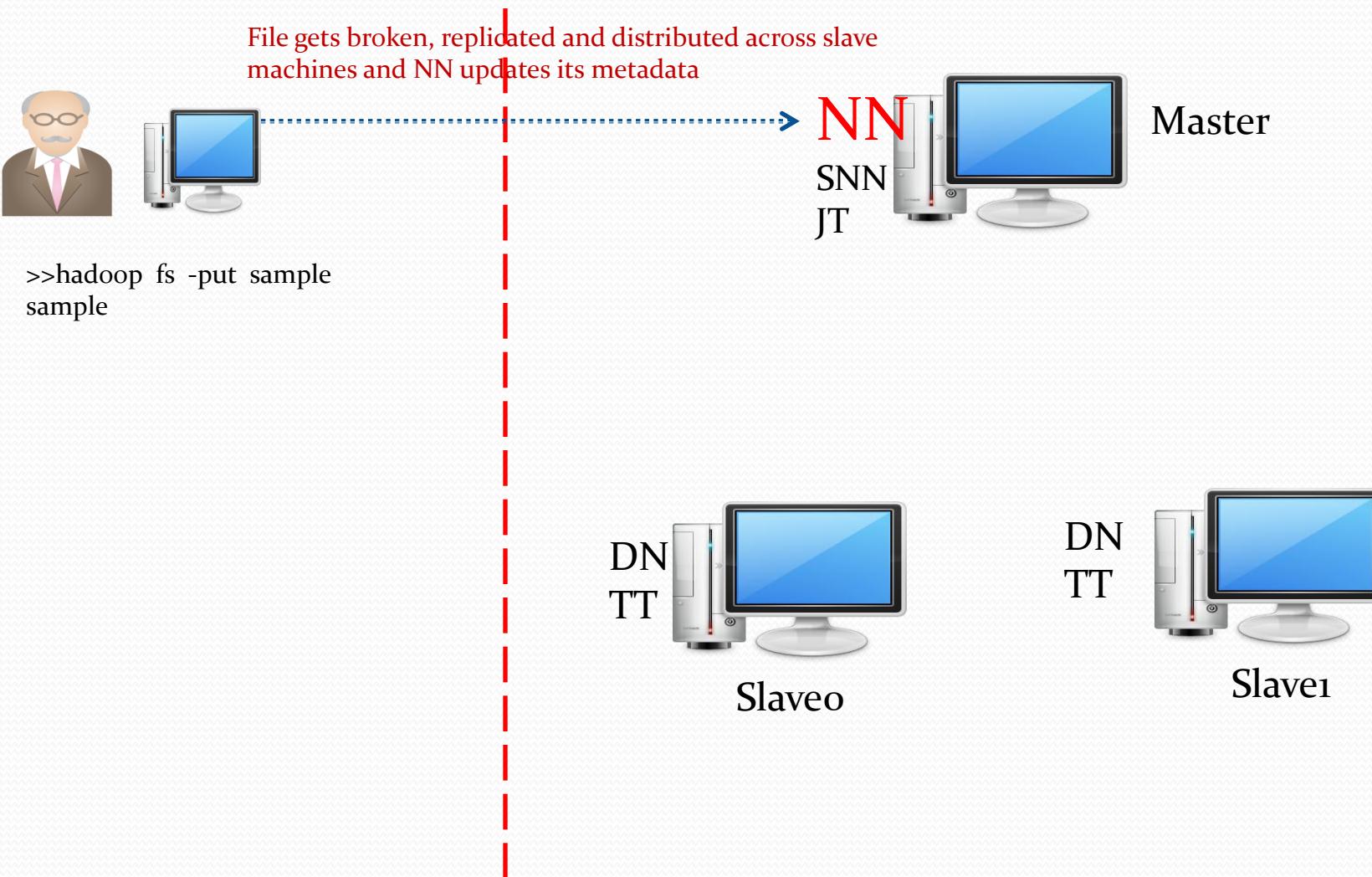
Slave0

DN
TT



Slave1

Job Tracker



Job Tracker



```
>>hadoop fs -put sample  
sample
```



NN
SNN
JT



Sample , {B₁,B₂}
B₁ -> Slaveo
B₂->Slave1
Master

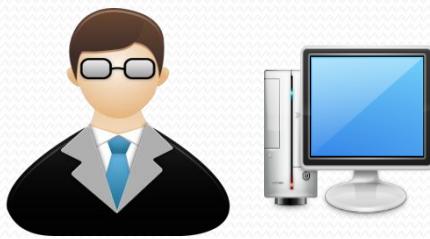


Slaveo



Slave1

Job Tracker



>> I have written MapReduce Job to analyze the sample file on HDFS

>> What I should know inorder to submit my MapReduce Job?



Sample , {B₁,B₂}
B₁ -> Slave0
B₂->Slave1
Master

Job Tracker



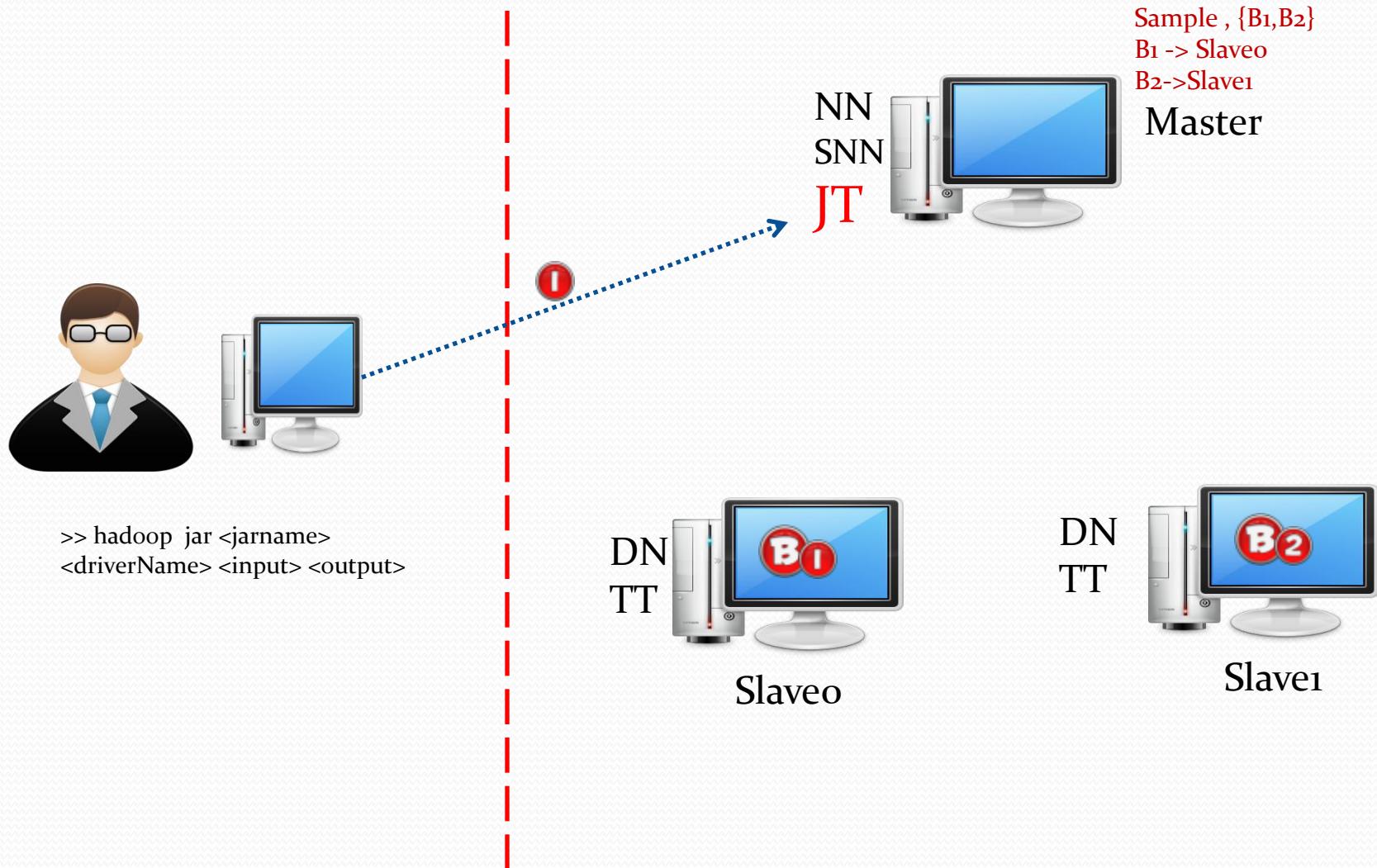
```
>> hadoop jar <jarname>  
<driverName> <input> <output>
```



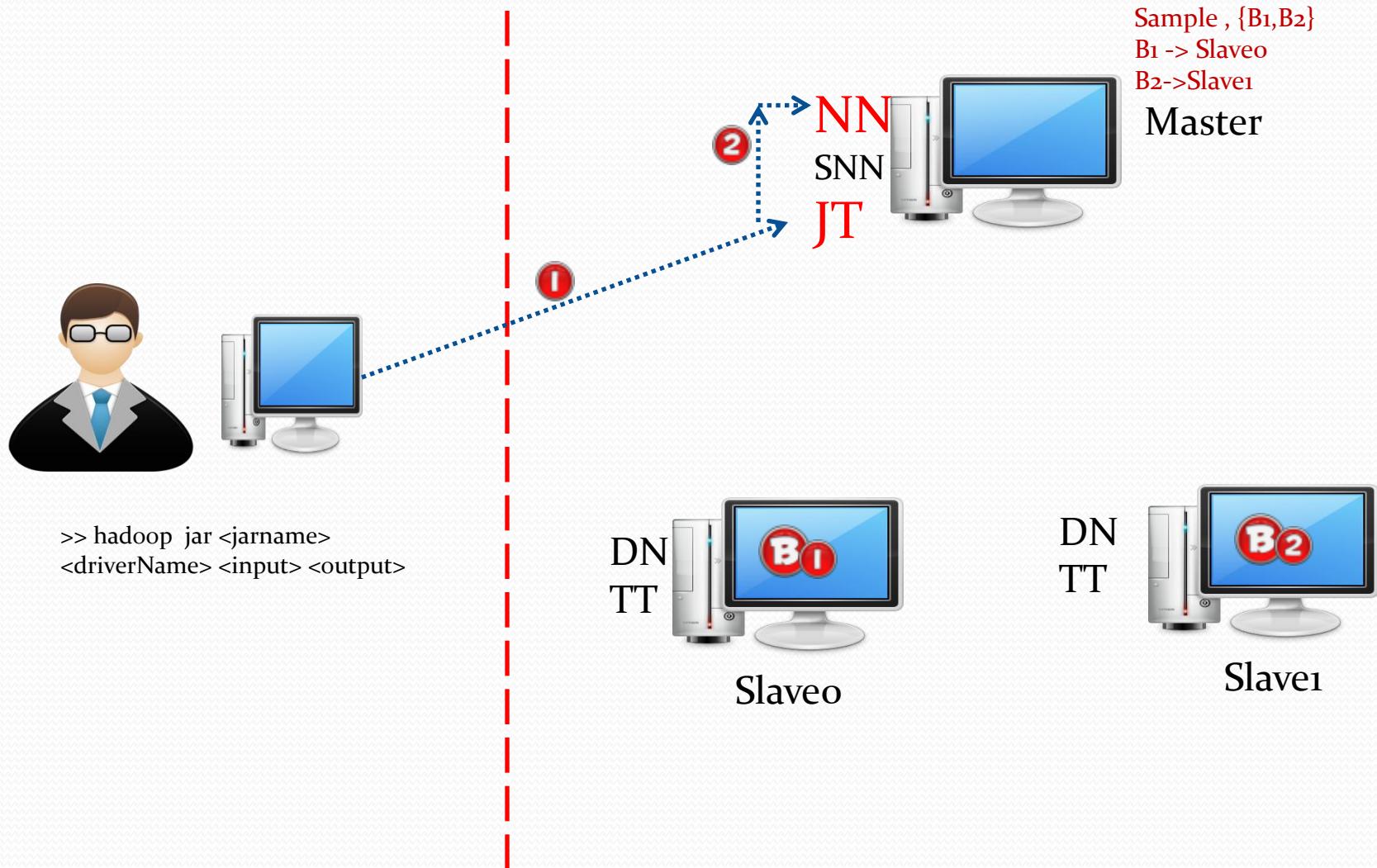
Sample , {B₁,B₂}
B₁ -> Slaveo
B₂->Slave1
Master



Job Tracker



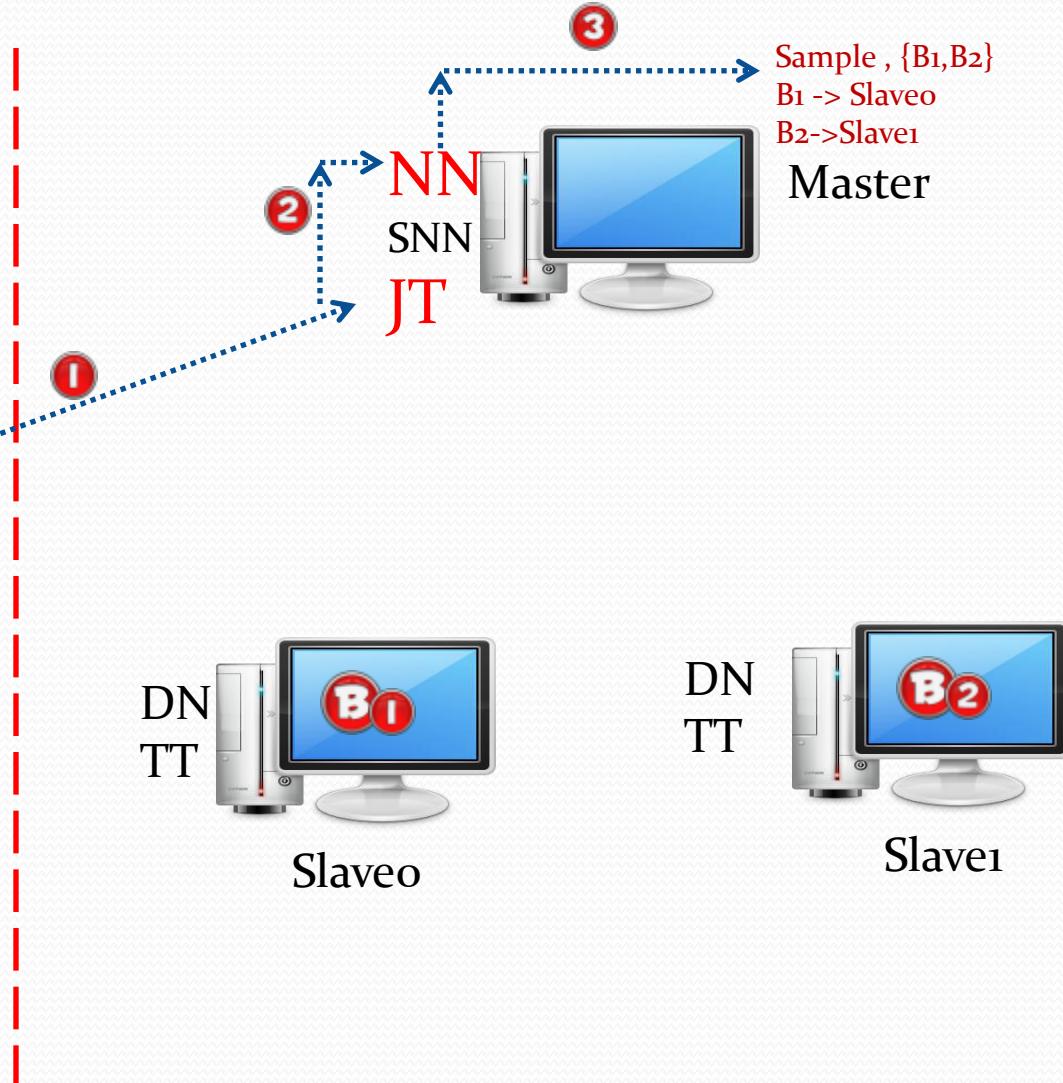
Job Tracker



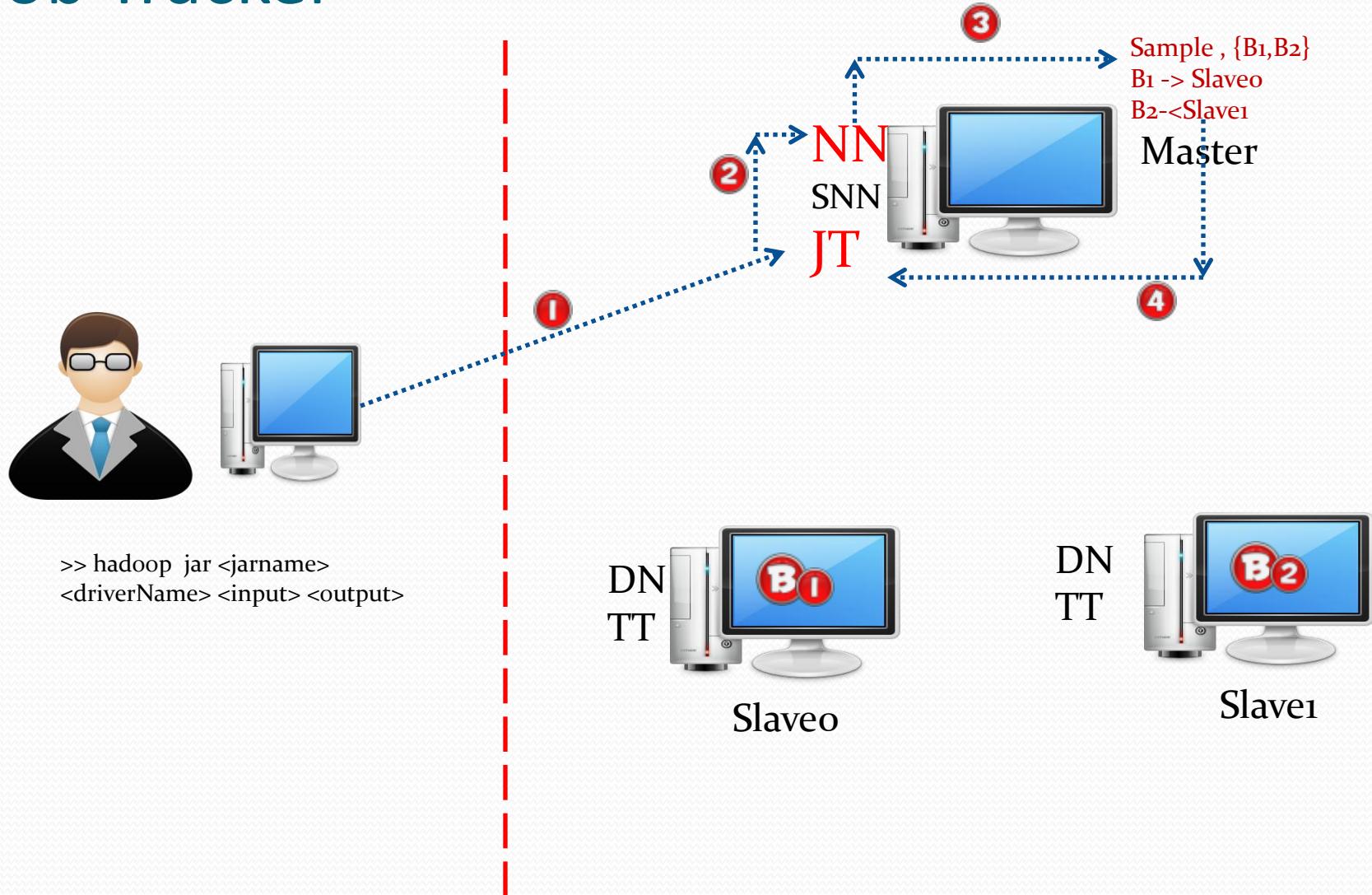
Job Tracker



```
>> hadoop jar <jarname>  
<driverName> <input> <output>
```



Job Tracker

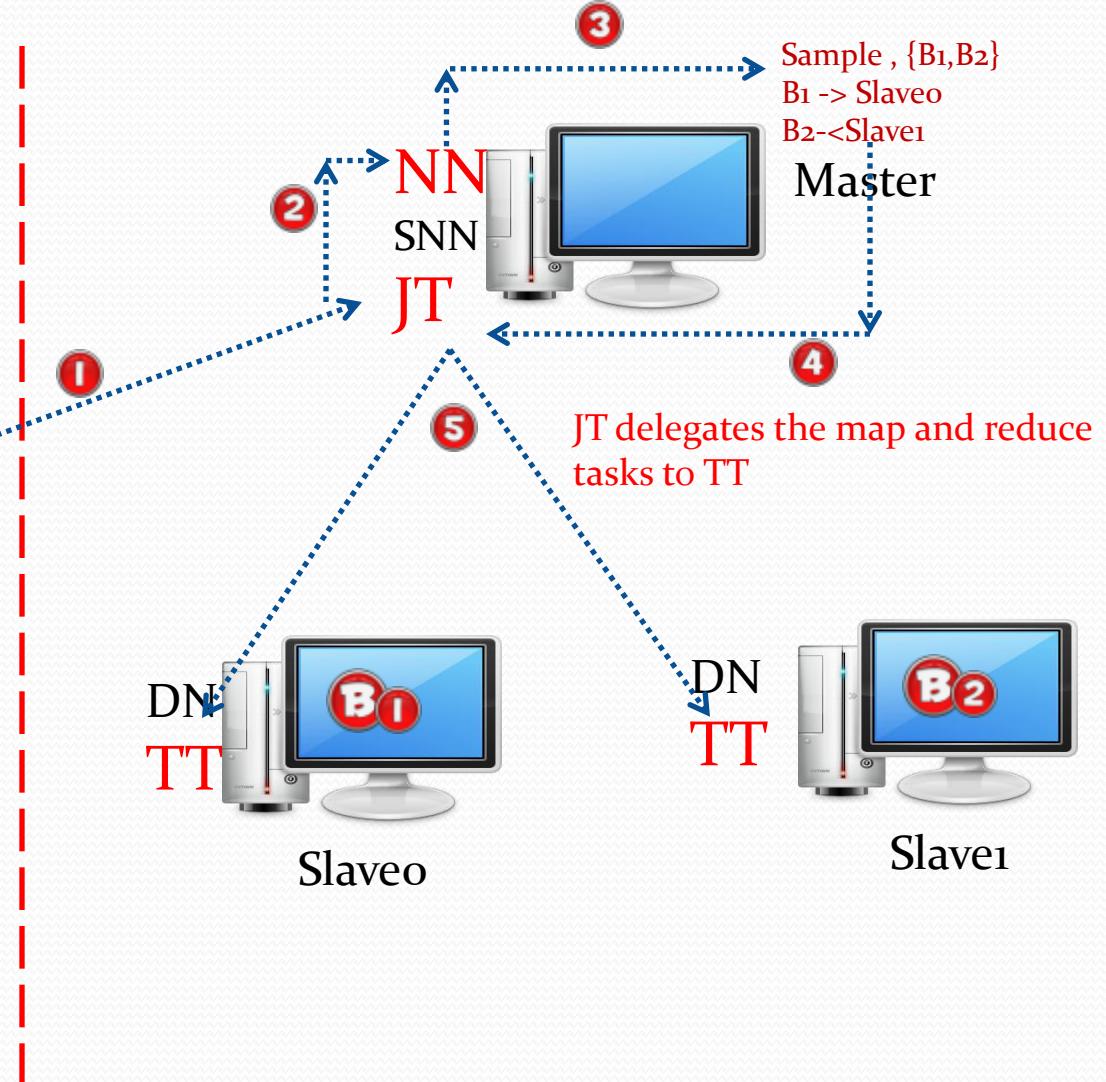


```
>> hadoop jar <jarname>  
<driverName> <input> <output>
```

Job Tracker



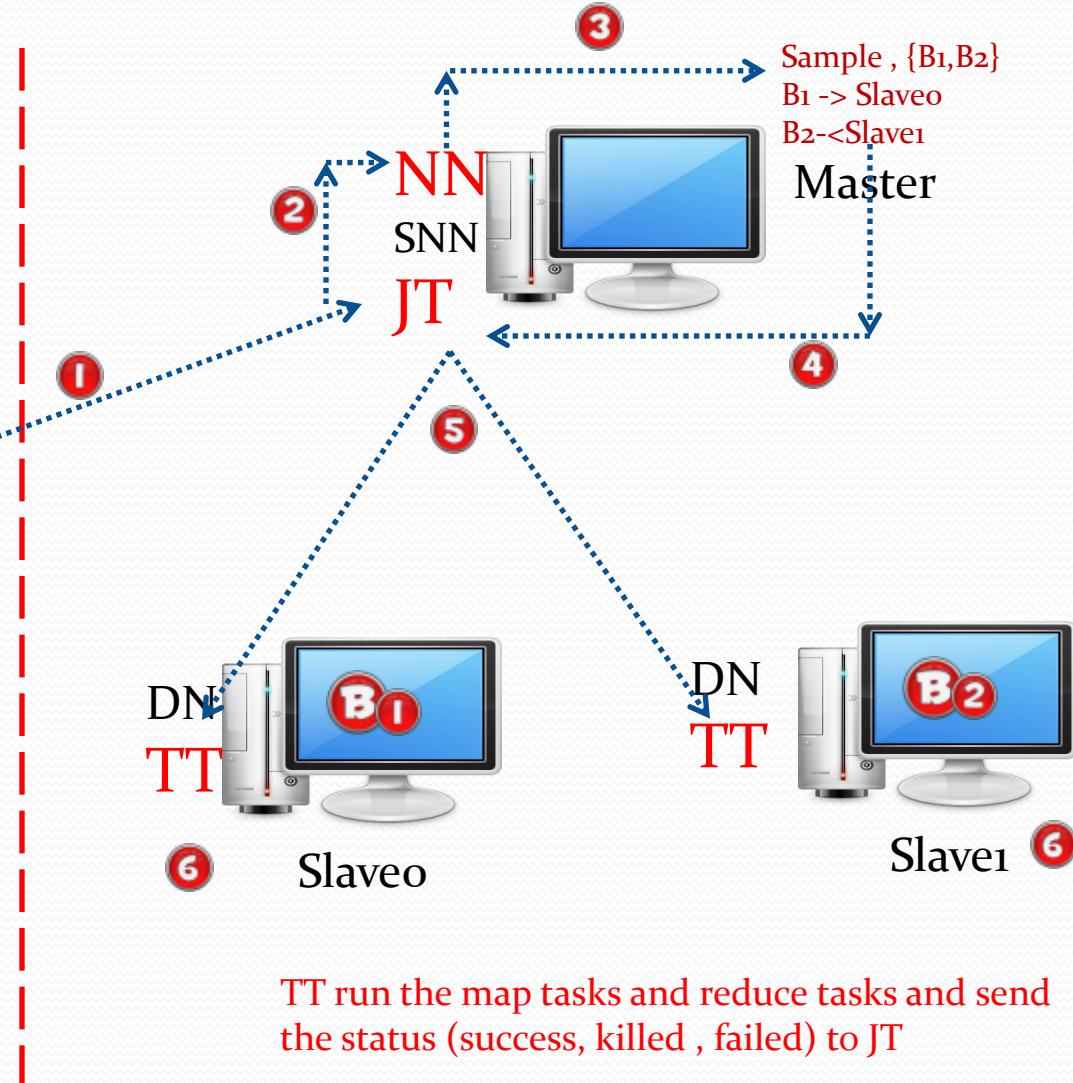
```
>> hadoop jar <jarname>  
<driverName> <input> <output>
```



Job Tracker



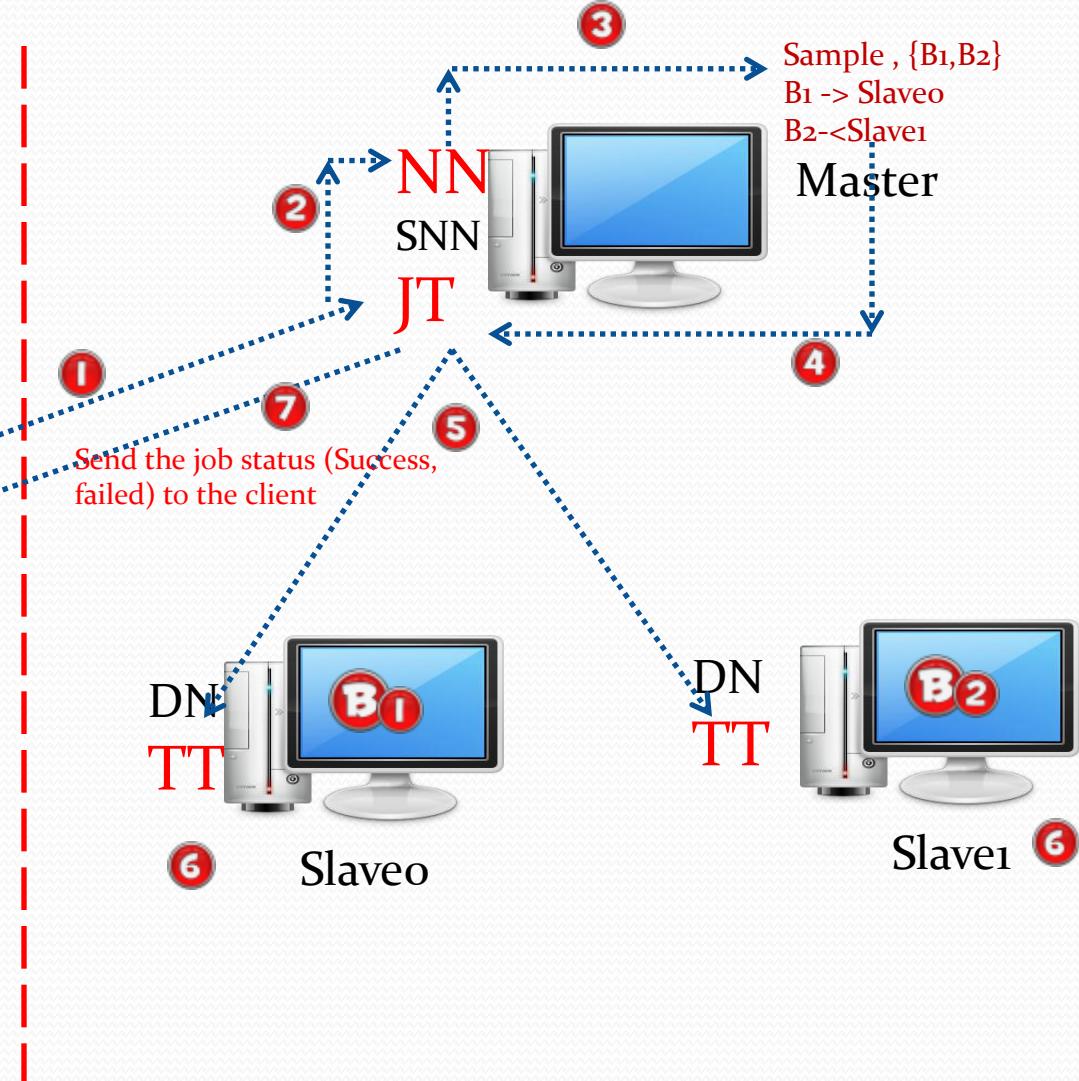
```
>> hadoop jar <jarname>  
<driverName> <input> <output>
```



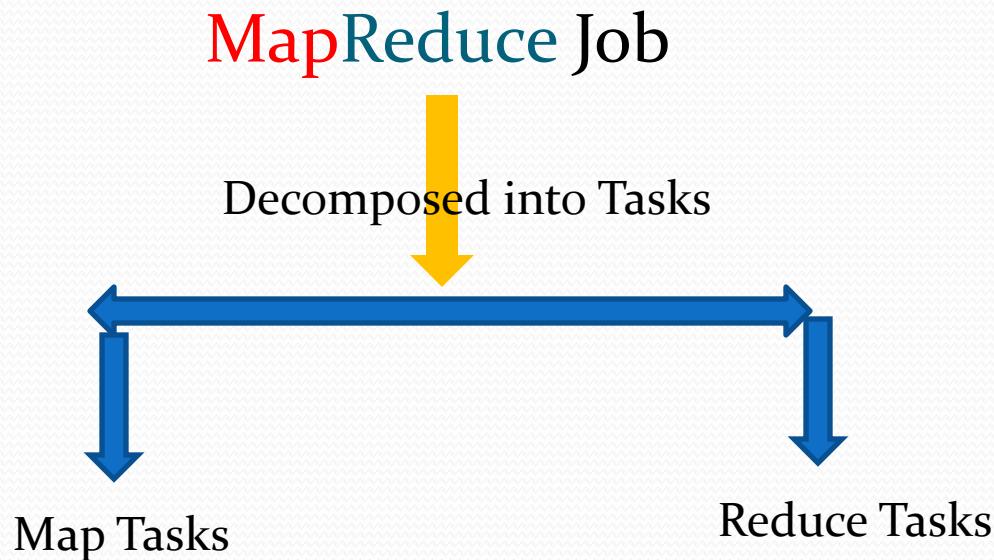
Job Tracker



```
>> hadoop jar <jarname>  
<driverName> <input> <output>
```

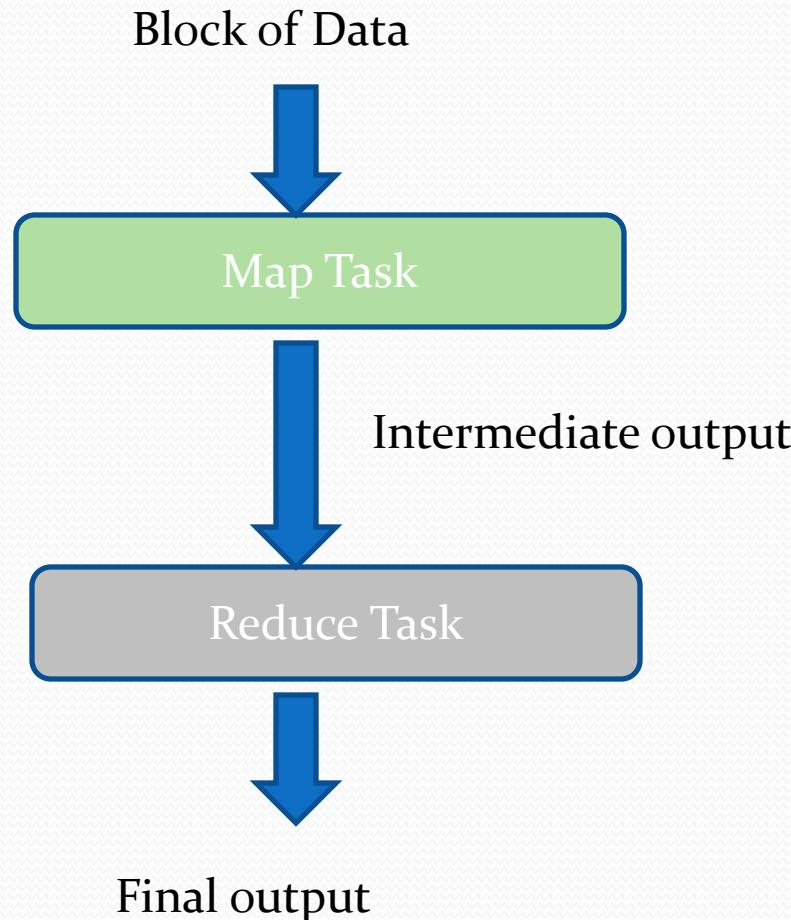


What is MapReduce Job?



- MapReduce Job always starts with map task and ends with reduce task
- Reduce task will never ever start until and unless all the map tasks are finished

What is MapReduce Job?



What is MapReduce Job?

DN
TT



DN
TT



DN
TT



What is MapReduce Job?

DN
TT



DN
TT

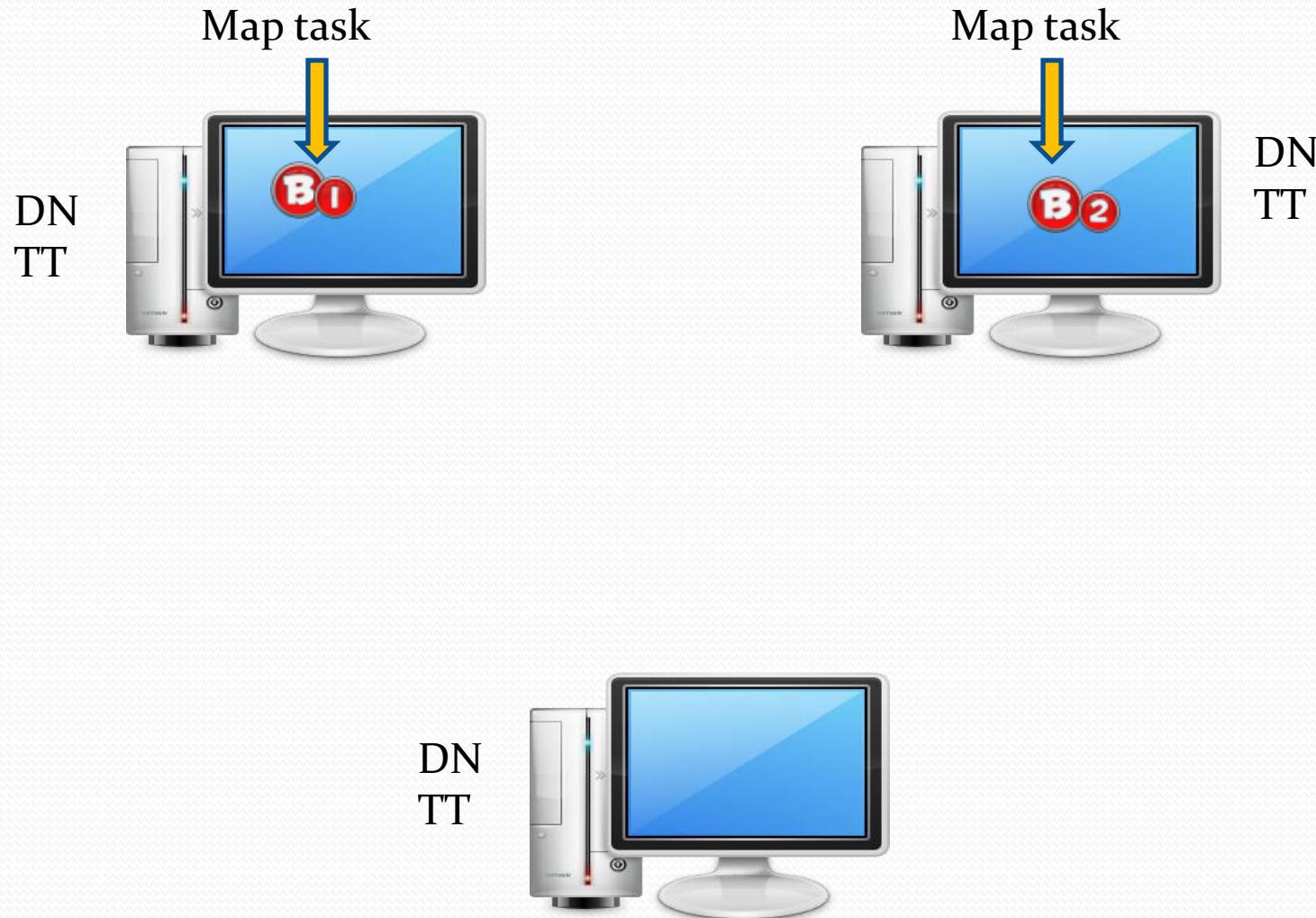


- JT decided to run map task on the above two slave machines and reducer on the below slave machine

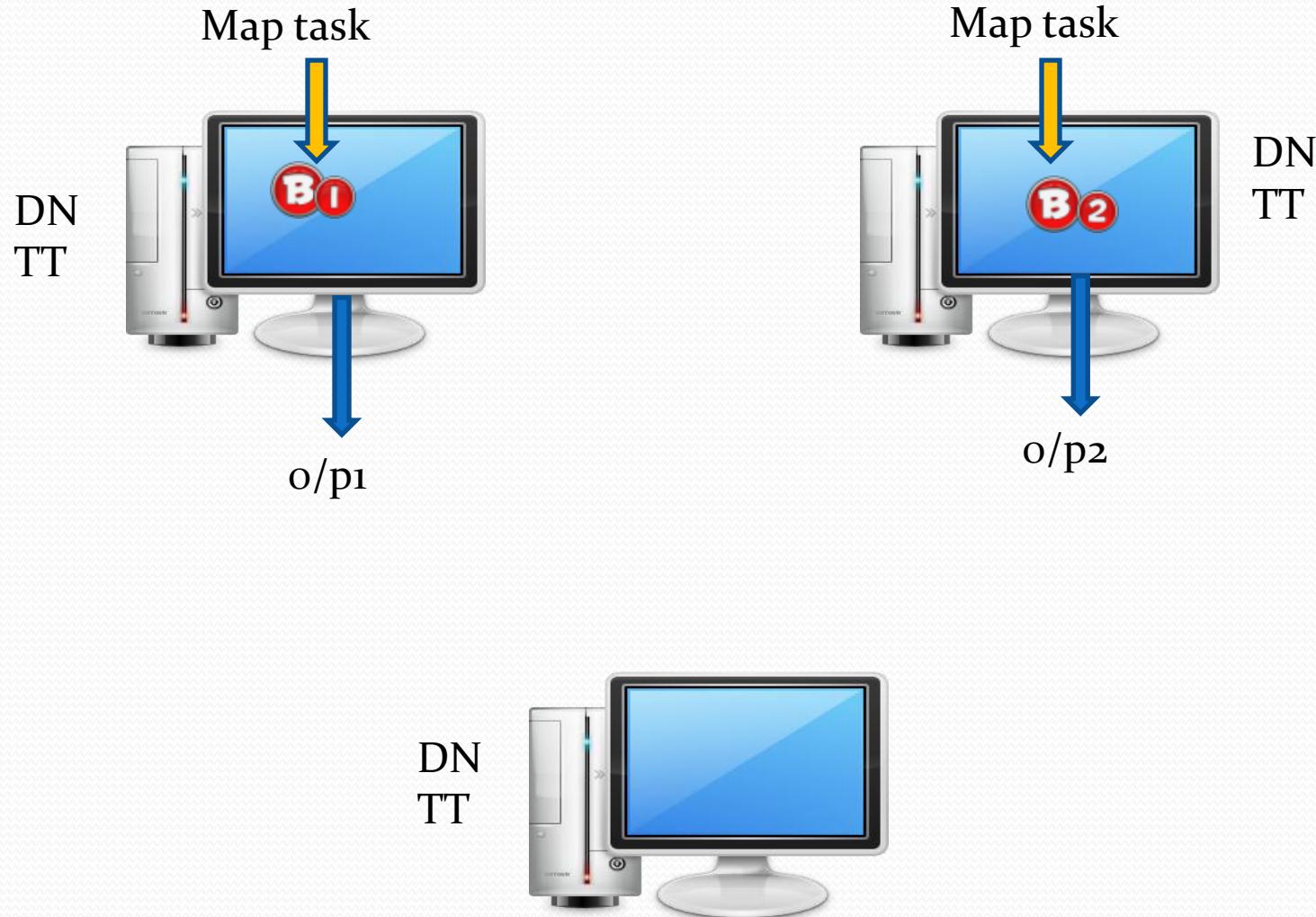
DN
TT



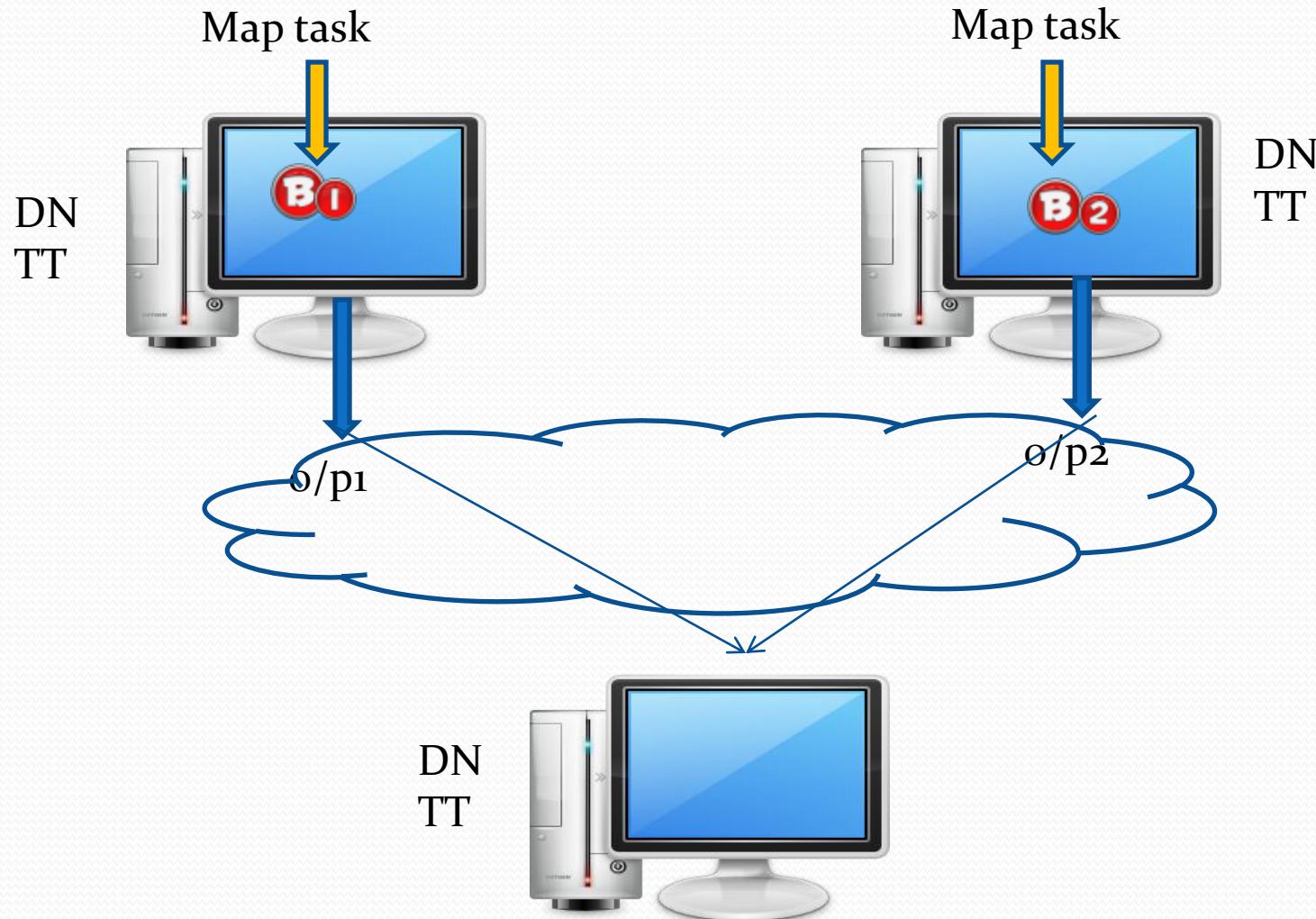
What is MapReduce Job?



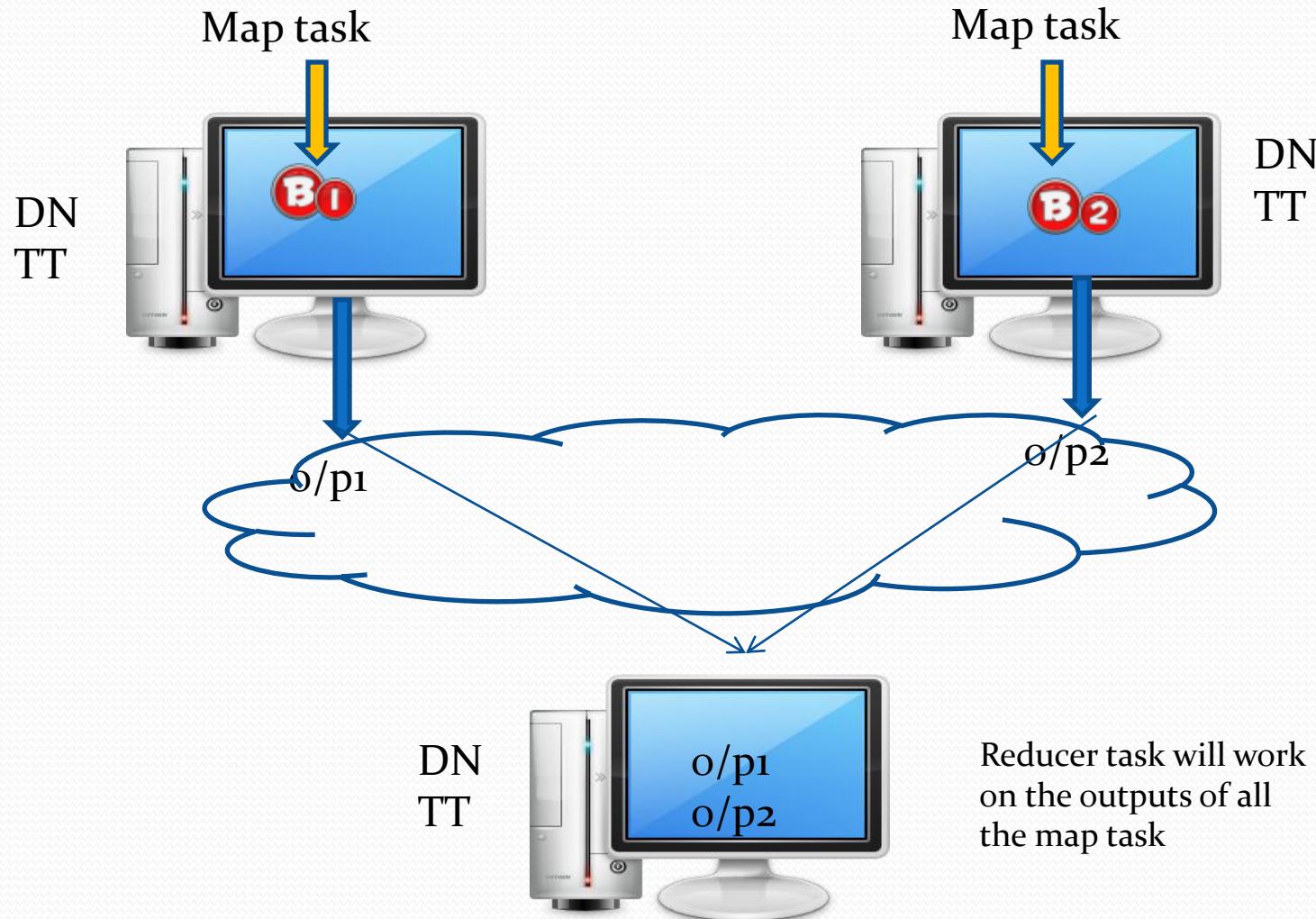
What is MapReduce Job?



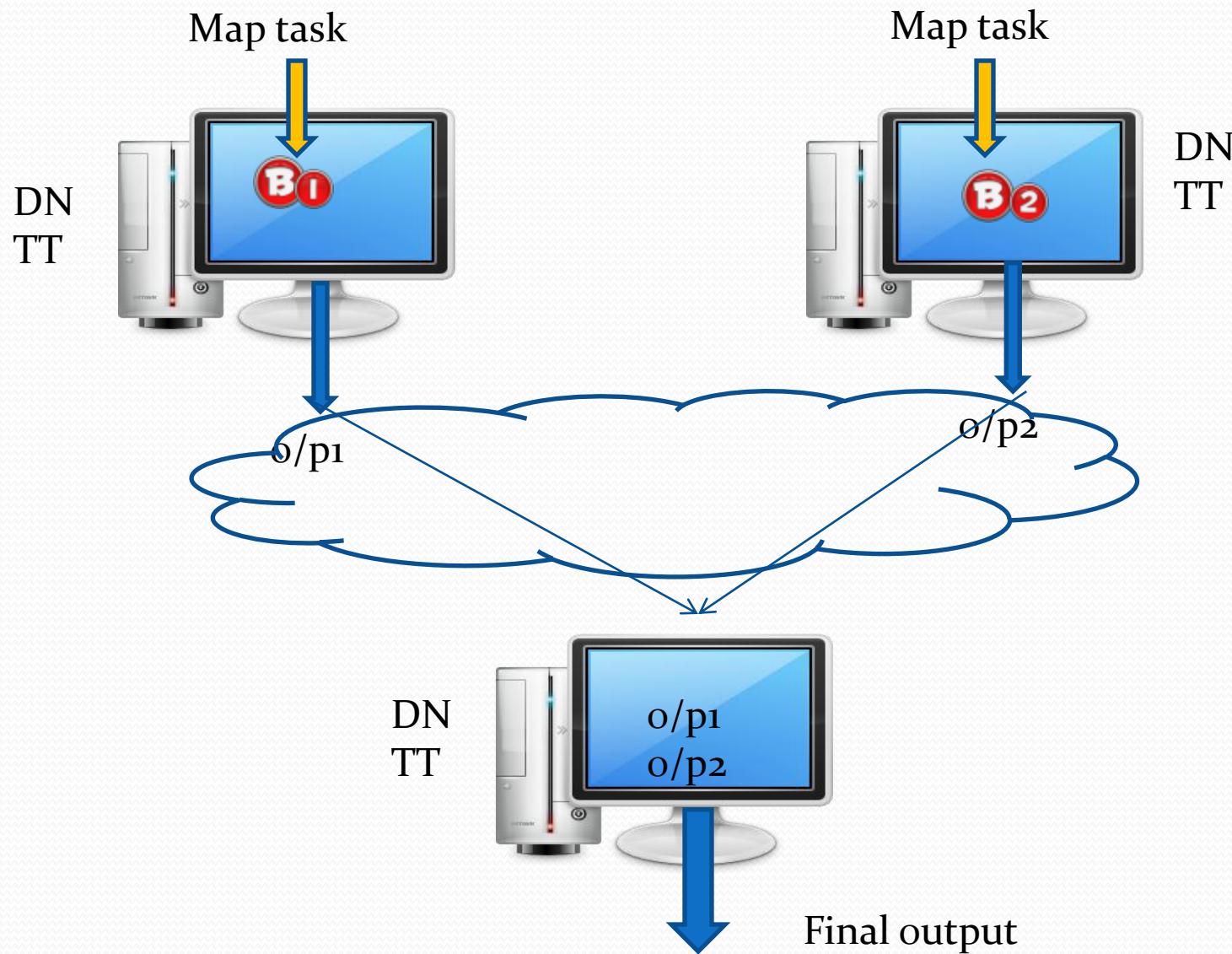
What is MapReduce Job?



What is MapReduce Job?



What is MapReduce Job?



What is MapReduce Job?

DN
TT



DN
TT



- Don't you think map task always needs to be run on the machine where data is present.
 - But it's not always possible
 - JT always try to schedule the map task on the machine where data is present
- Do you think data localization is applicable for reducer task?

DN
TT



Major Responsibility of JT

- Resource Scheduling
 - Allocating the resources to each and every job that are getting submitted
- Task Monitoring
 - Each and every job can have millions of map and reduce tasks
 - The tasks needs to be monitored and incase if they are failing or running slow, JT has to take care of this
- Does it make sense why JT needs to run on separate physical machine?

YARN-Yet Another Resource Negotiator

- No JT
- Resource Manager (RM)
- Per Job Application Master (AM)
- Node Manager

Task Tracker

- Responsible for running tasks (map or reduce tasks) delegated by job tracker
- For every task separate JVM process is spawned
- Periodically sends heart beat signal to inform job tracker
 - Regarding the available number of slots
 - Status of running tasks
 - Progress report of the tasks
 - By this report, JT will come to know whether a task is running slow or not

Task Tracker



DN
TT

- Every Task tracker is preconfigured with certain map slots and reduce slots
- Default number of map slots per TT is 2
- Default number of reduce slots per TT is 2

Map/Reduce Administration

hortonfo

ze is 15.25 MB/966.69 MB)

Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes	Graylisted Nodes
0	0	0	0	2	2	4.00	0	0

Task Tracker



DN
TT

- What does it mean?
- This TT will not run more than two map tasks parallelly at any give point of time.
 - If more than 2 map tasks needs to be scheduled, rest needs to wait till the slot gets free

Task Tracker

DN
TT



DN
TT



DN
TT



This map slot is busy in running some other job's map task



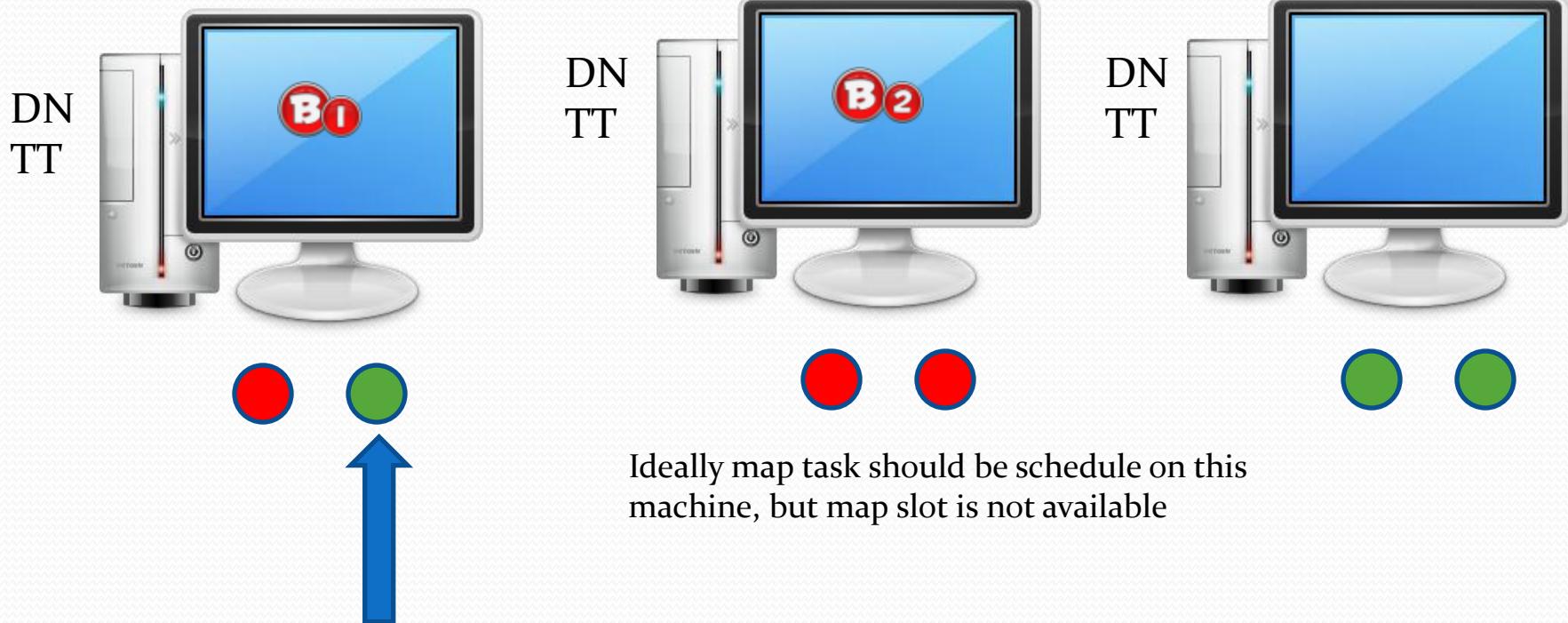
This map slot is available and JT can schedule the map task

Task Tracker



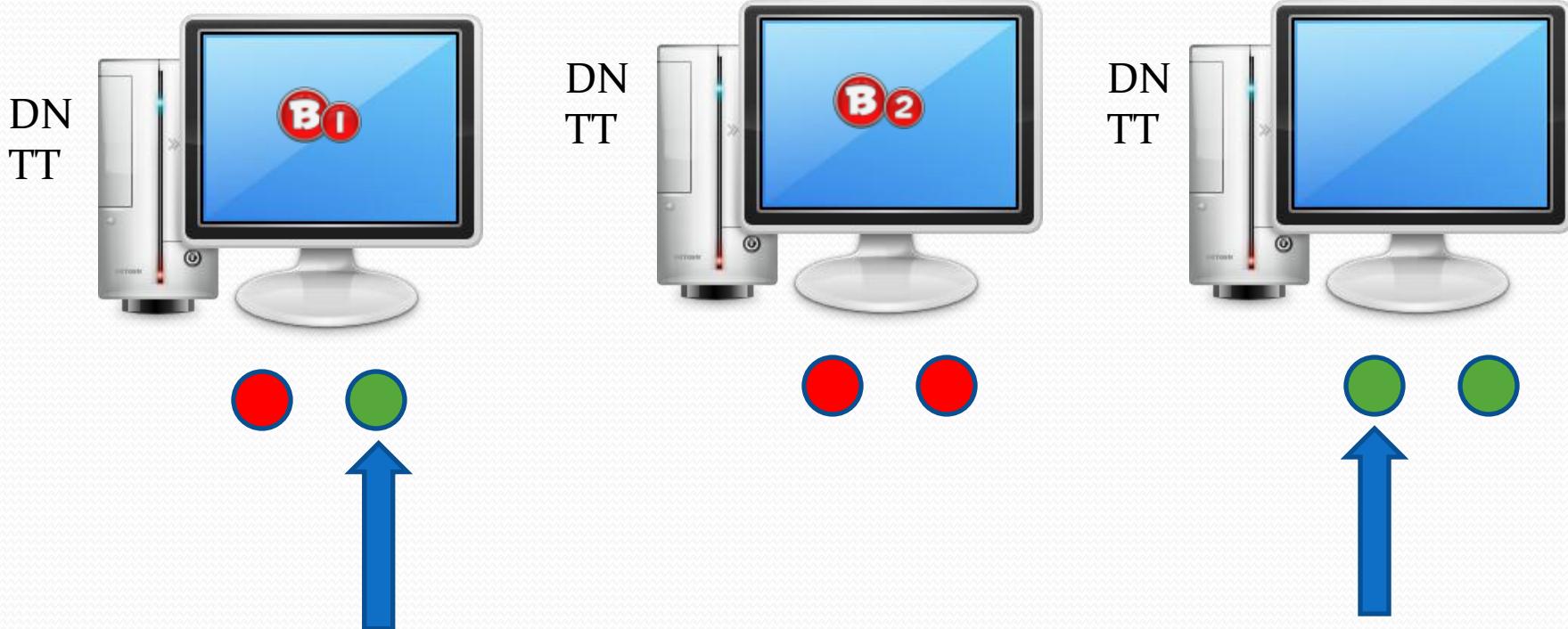
Ideally map task needs to be run on the machine where data is present to achieve data localization

Task Tracker



Map slot is available and you can schedule the map task that can process B₁

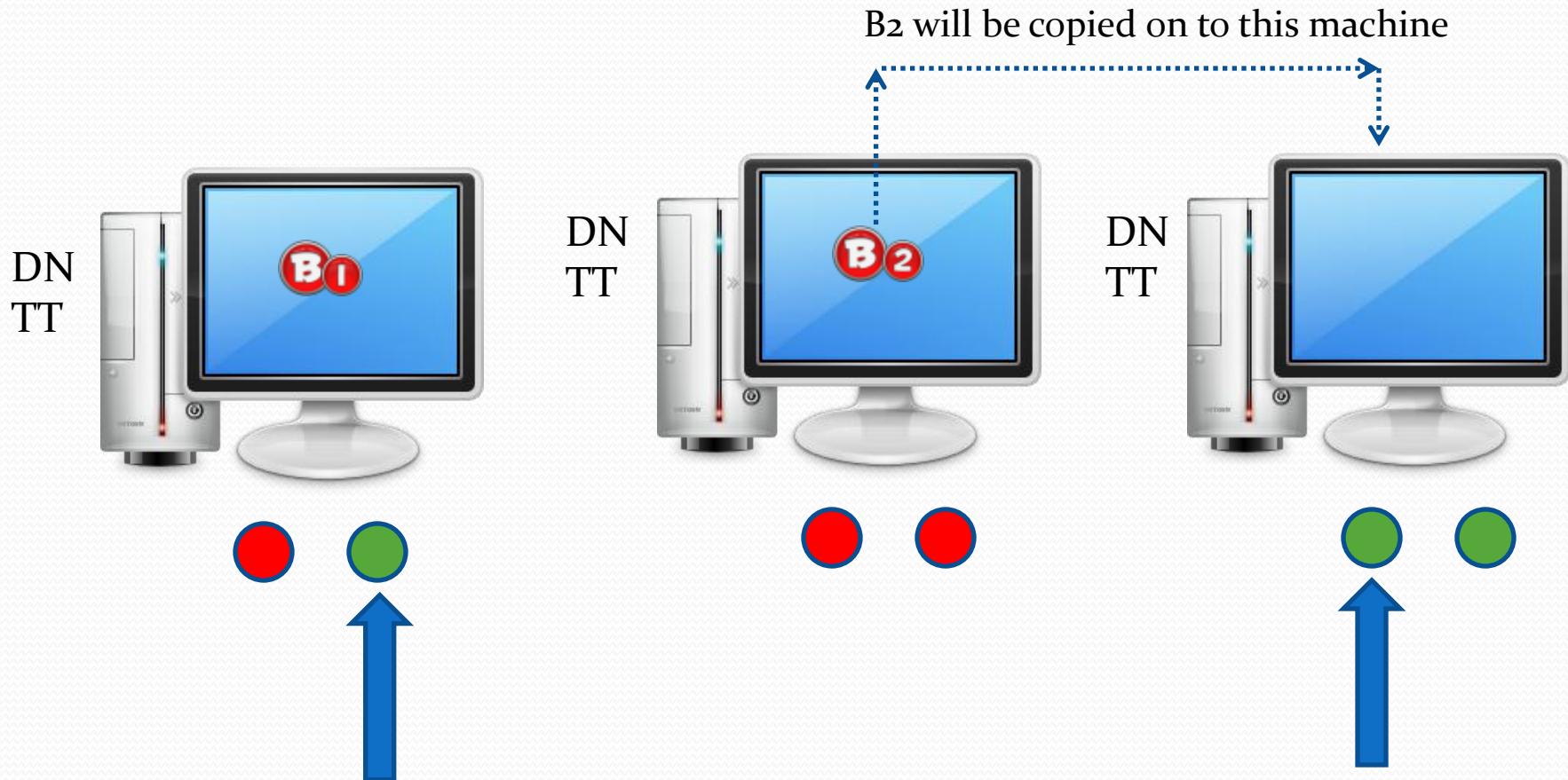
Task Tracker



Map slot is available and you can schedule the map task that can process B₁

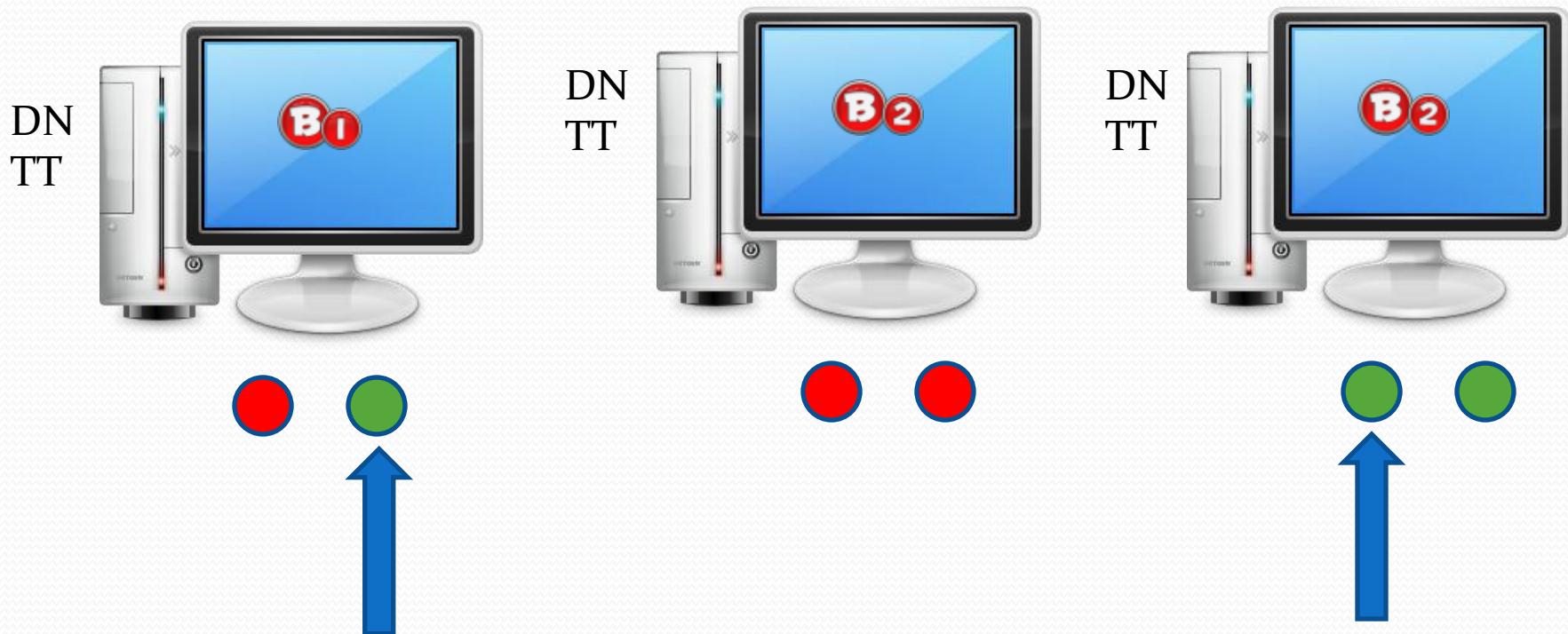
Map task is running on this machine to process B₂. But B₂ is not here

Task Tracker



Map slot is available and you can schedule
the map task that can process B1

Task Tracker



Map slot is available and you can schedule
the map task that can process B1

Task Tracker

DN
TT



DN
TT



DN
TT

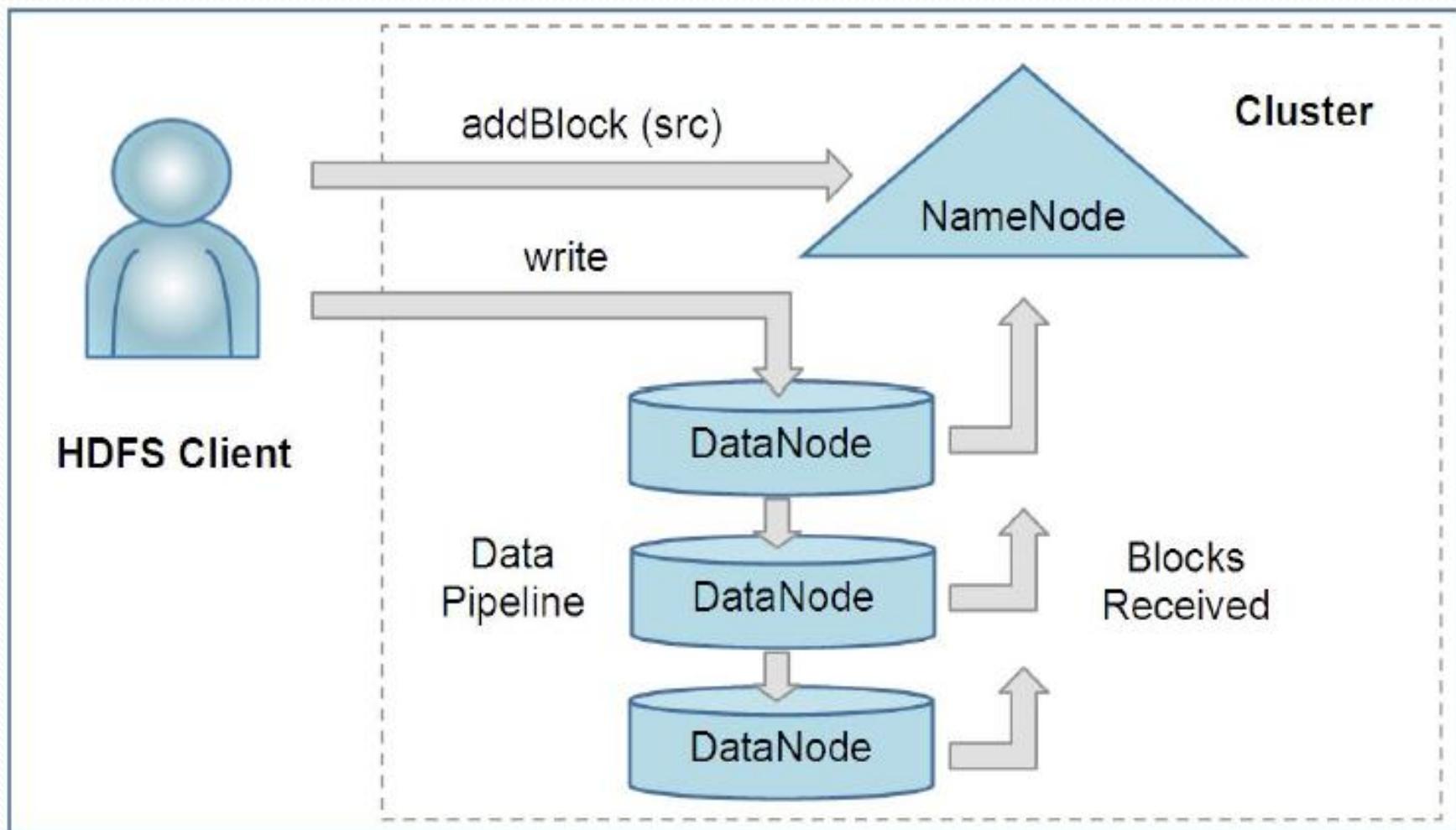


Once the job is finished, the block of data which was copied will be deleted

Note

- Job tracker always try to schedule the map tasks on the machine where data is present.
- Its always not possible
 - It depends on the availability of the map slot with the TT on that machine

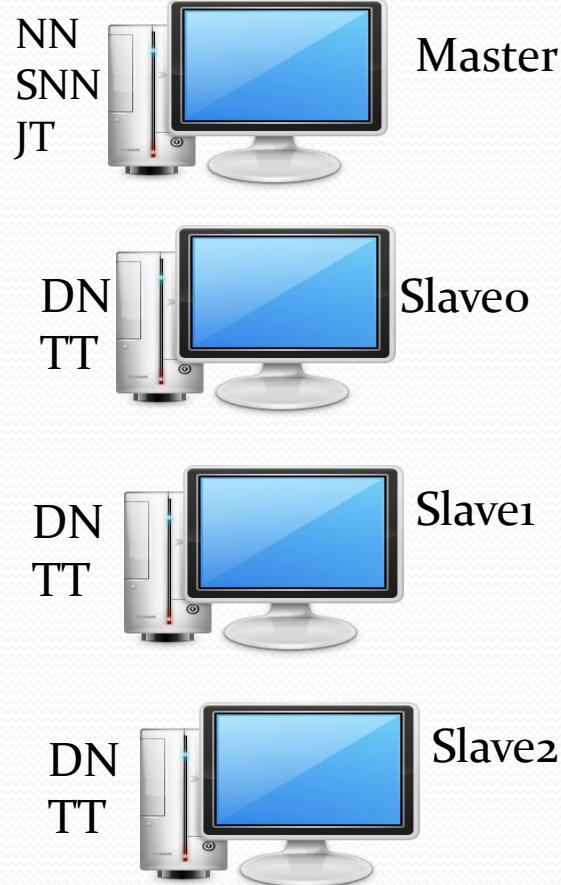
Writing a file to HDFS



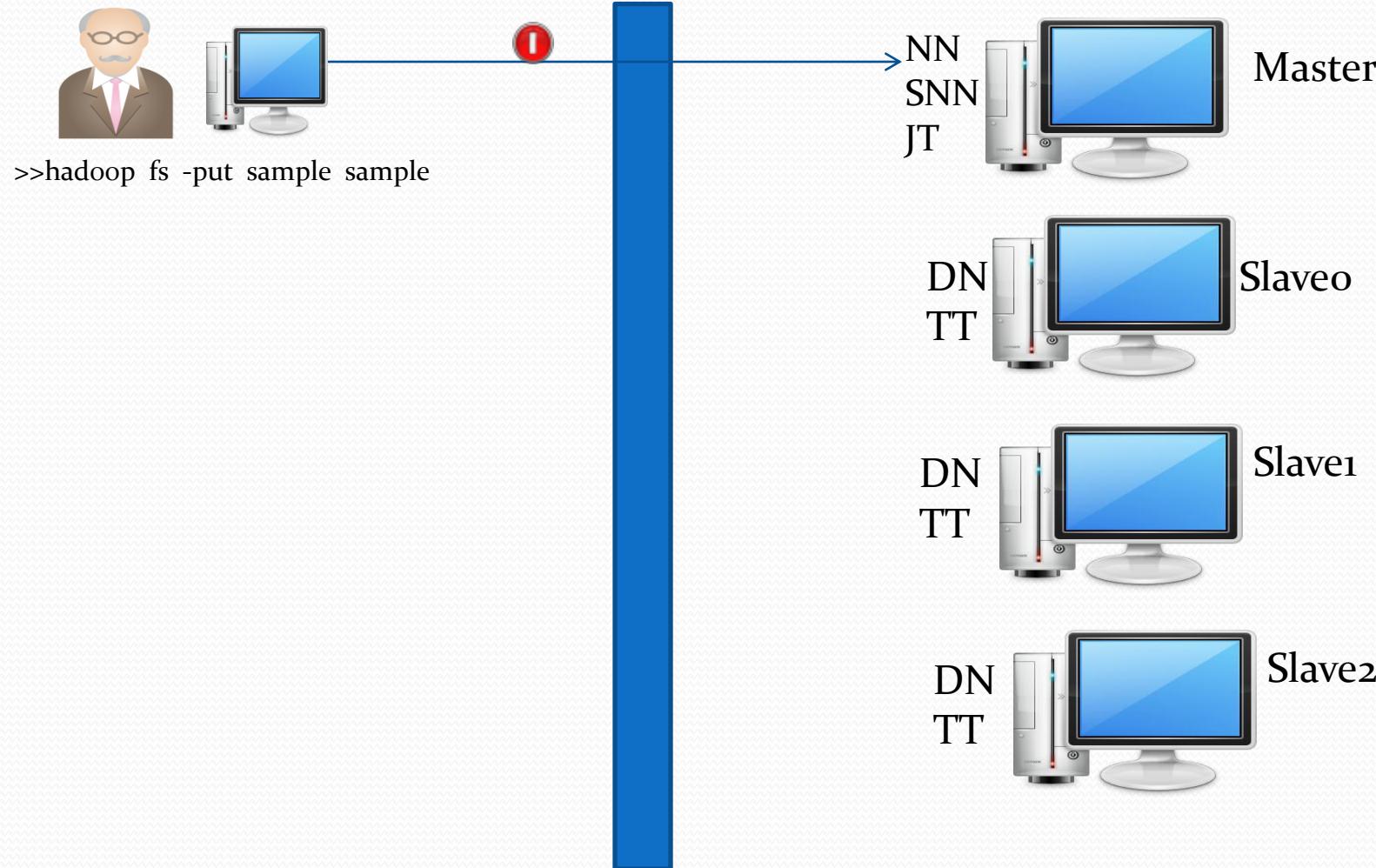
Writing a file to HDFS



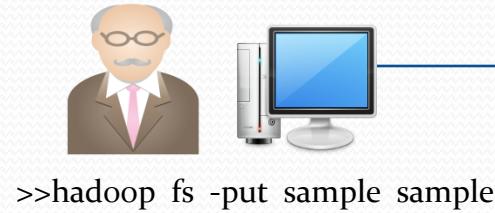
```
>>hadoop fs -put sample  
sample
```



Writing a file to HDFS



Writing a file to HDFS



①

②

Creates a block id and ask 3 DN to host the replica

NN
SNN
JT



Master

DN
TT



Slave0

DN
TT



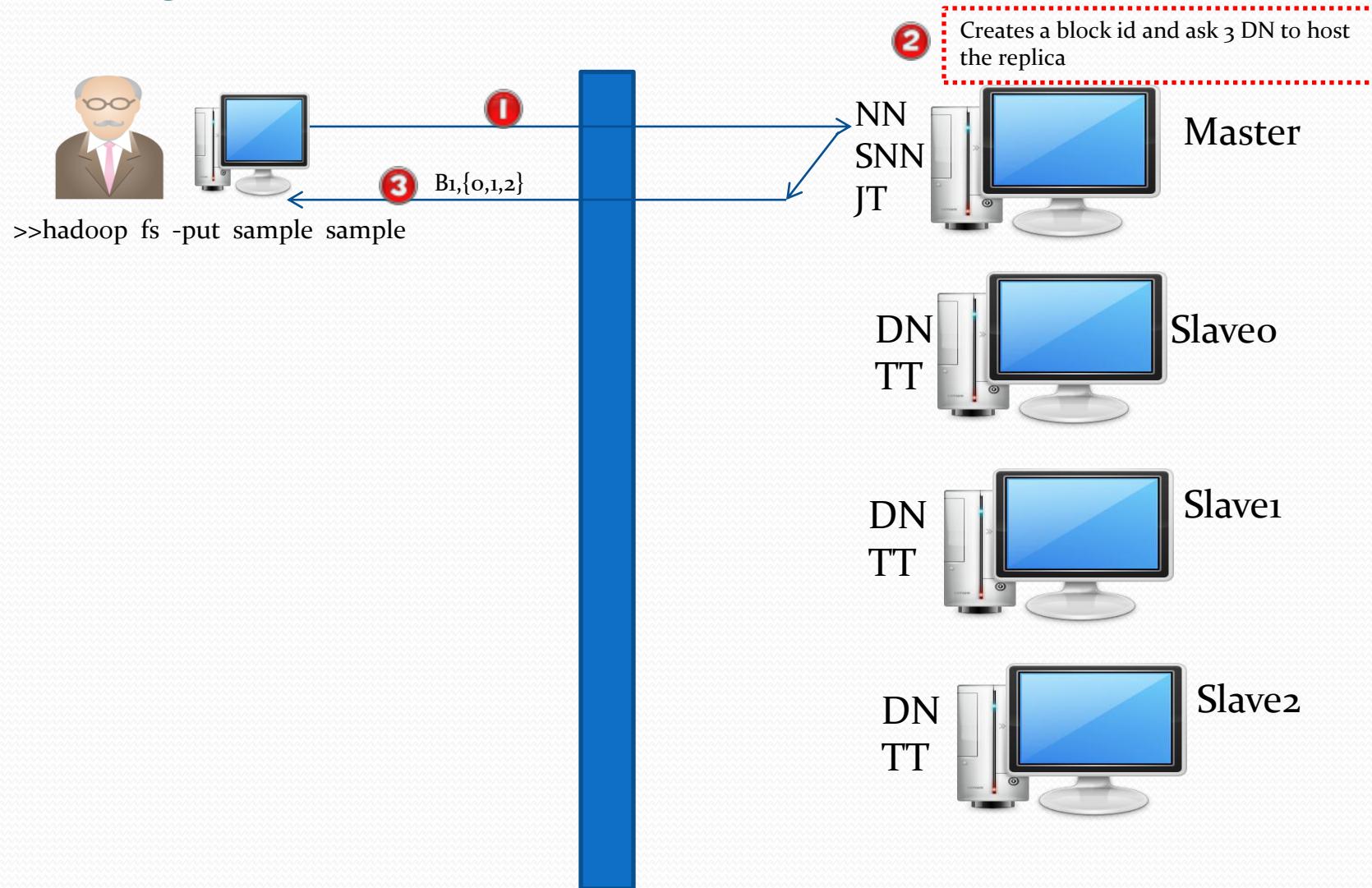
Slave1

DN
TT

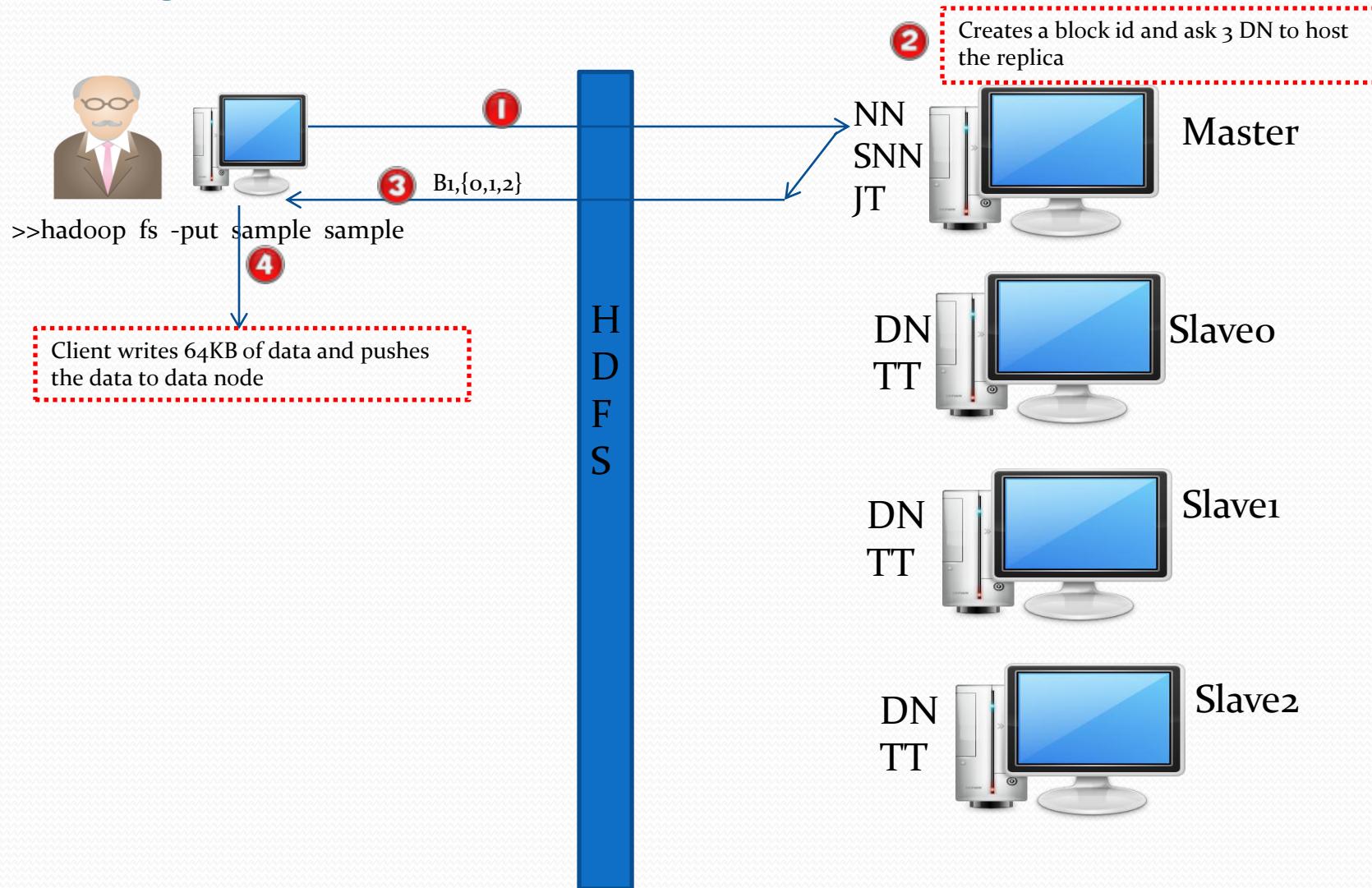


Slave2

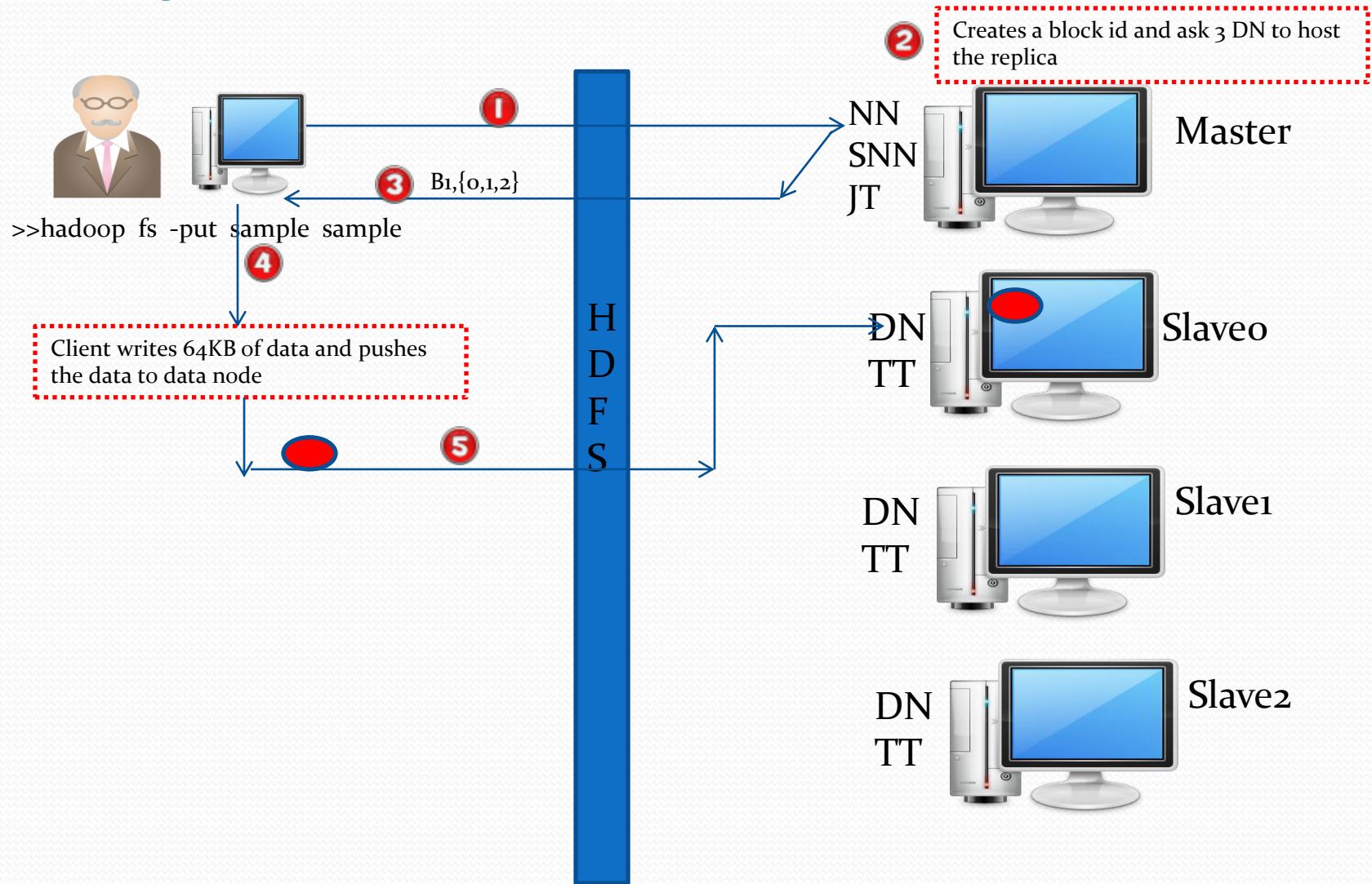
Writing a file to HDFS



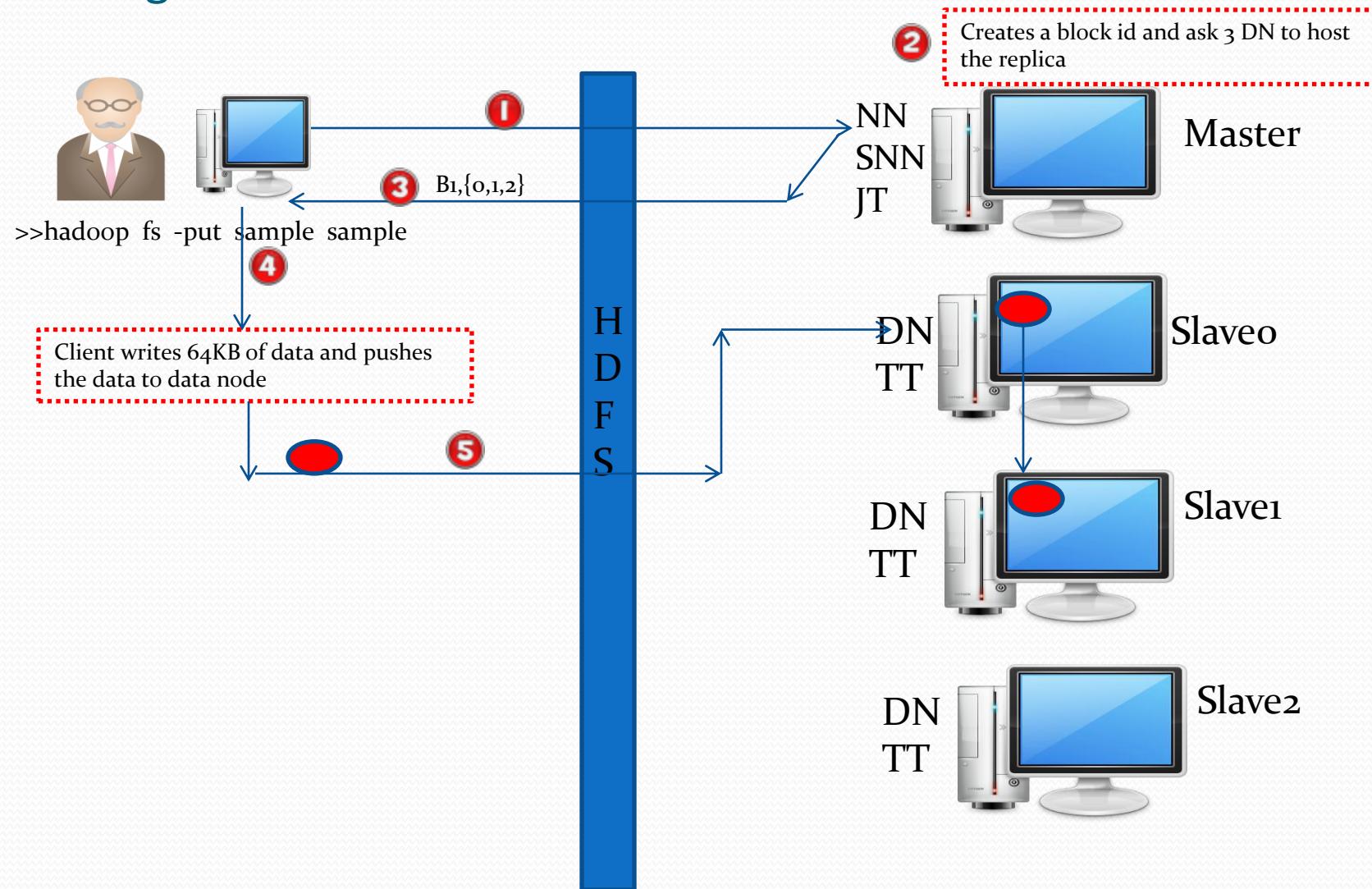
Writing a file to HDFS



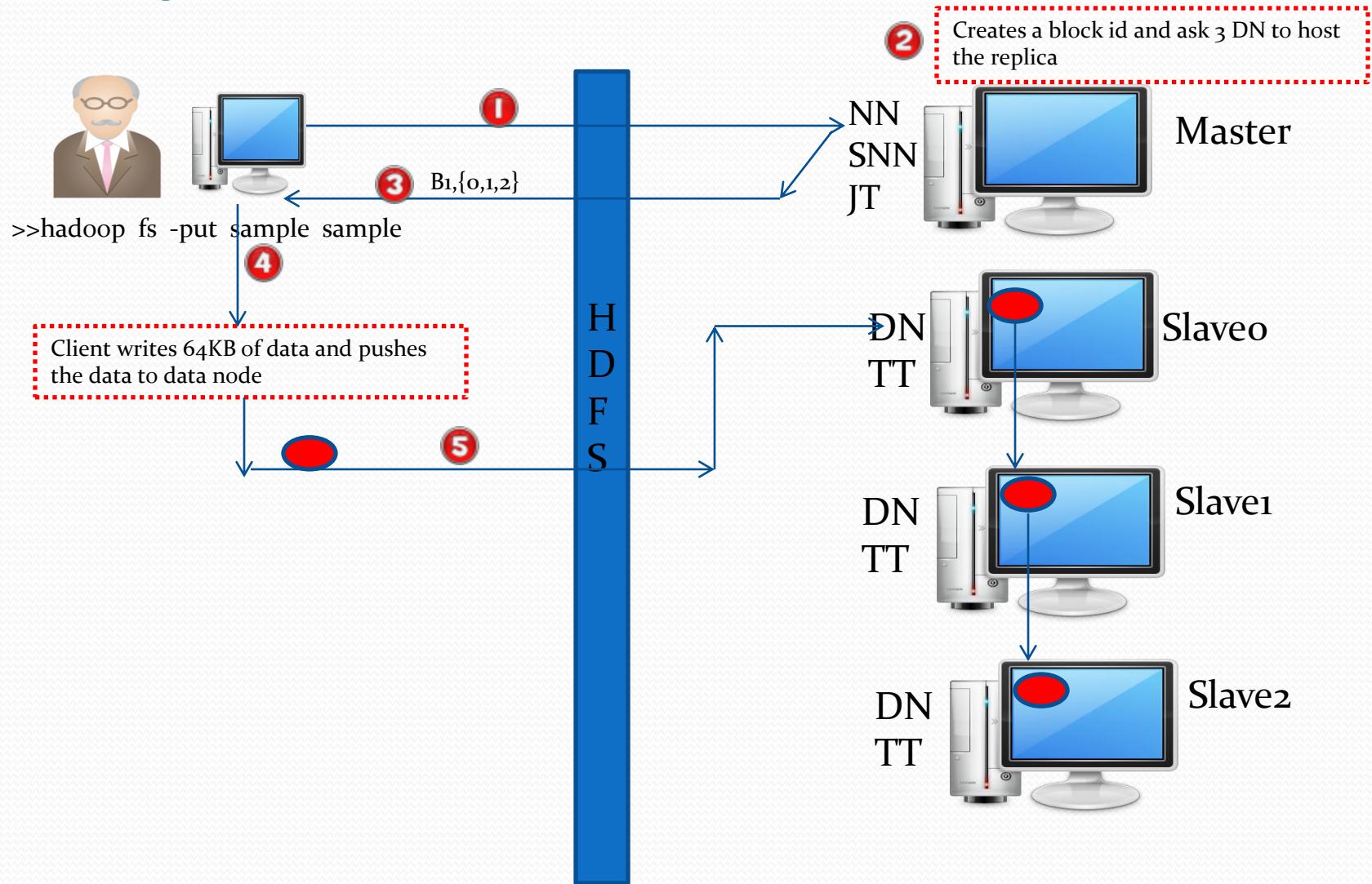
Writing a file to HDFS



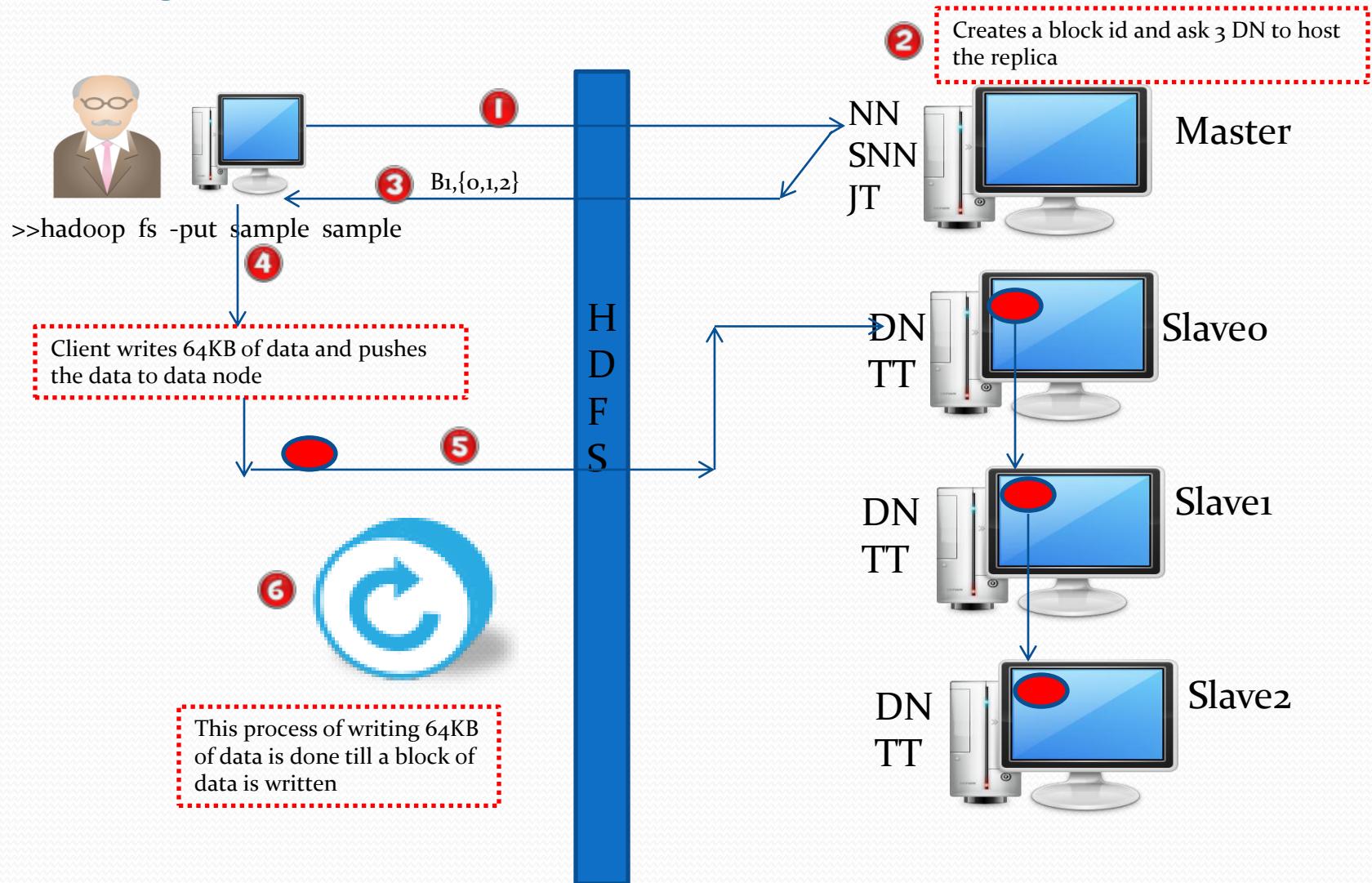
Writing a file to HDFS



Writing a file to HDFS



Writing a file to HDFS

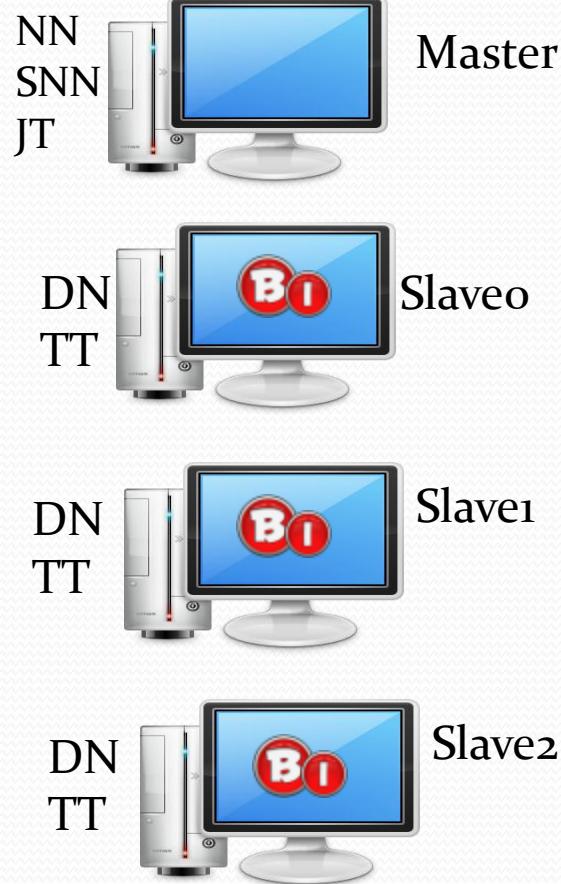


Writing a file to HDFS



>>hadoop fs -put sample sample

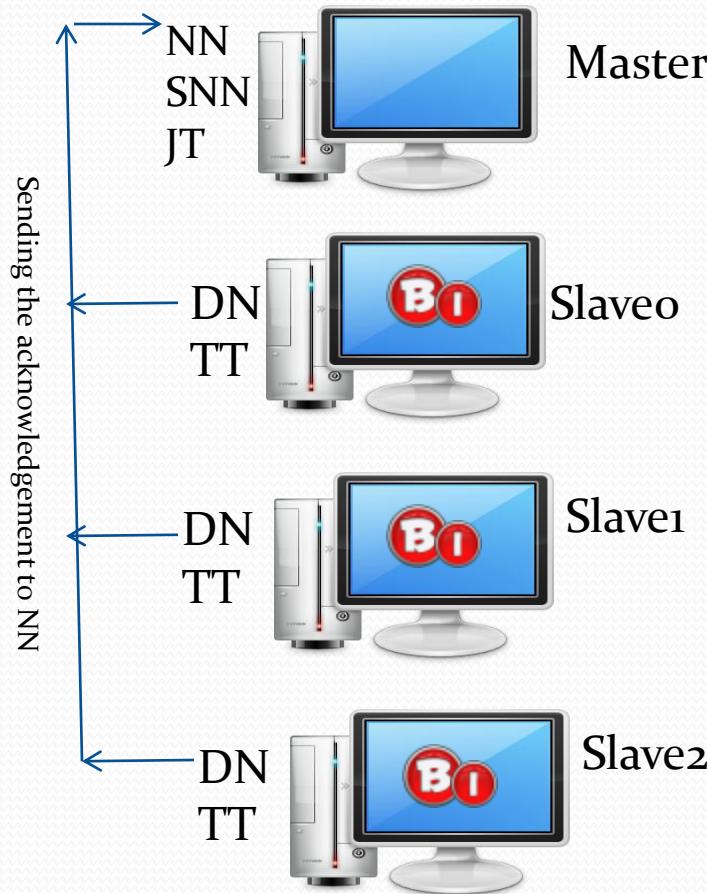
H
D
F
S



Writing a file to HDFS



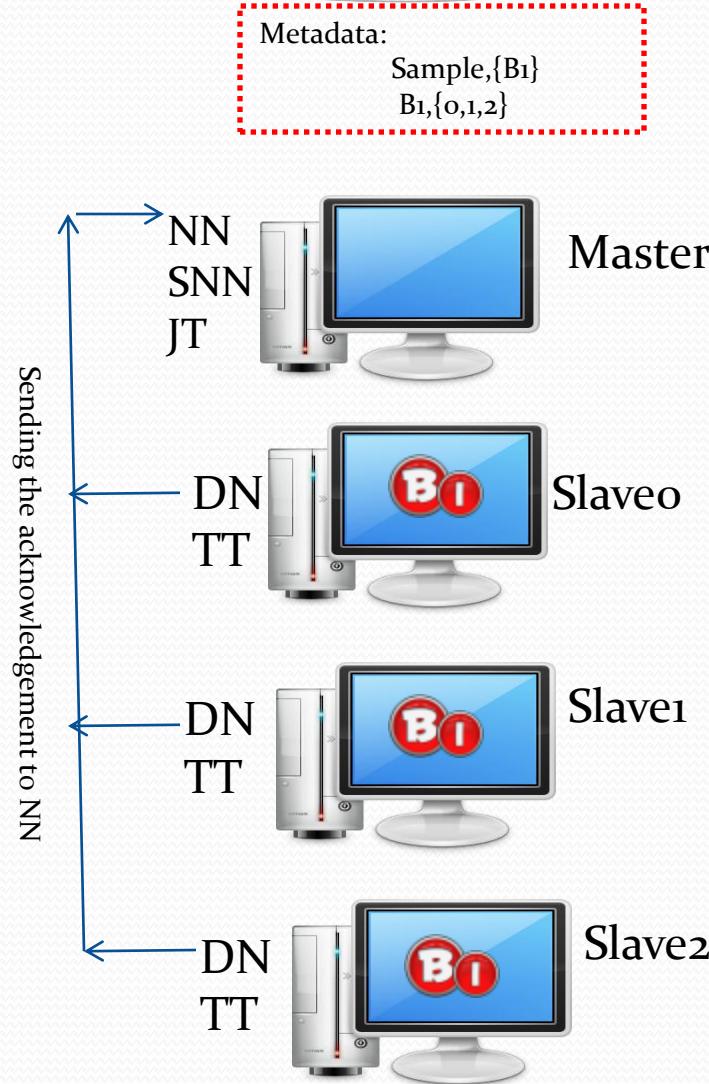
>>hadoop fs -put sample sample



Writing a file to HDFS



>>hadoop fs -put sample sample



Writing a file to HDFS

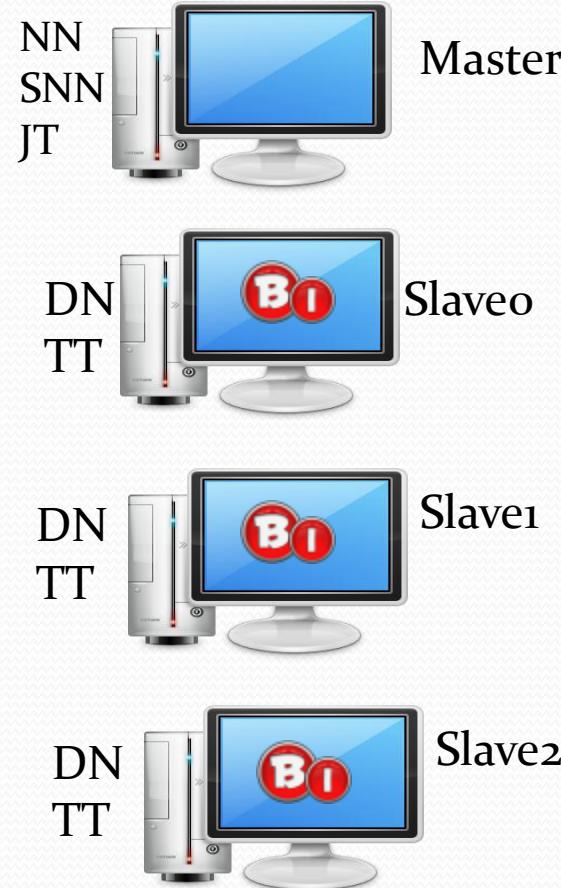


```
>>hadoop fs -put sample sample
```



If another block needs to be written, then the same steps are repeated

Metadata:
Sample,{B1}
B1,{0,1,2}



Writing a file to HDFS

- Blocks are written independently.
- Answer the following questions:
 - What if the client wants to write another block of data after the first block has completed successfully and fails in between? What would happen to the first block which was successfully written?
 - If a client has written 3 blocks of data and he currently writing 4th block, another client used cat command to read the file? Will the other client be able to read the data?

Reading a file from HDFS

- Connects to NN
- Ask NN to give the list of data nodes that is hosting the replica's of the block of file
- Client then directly read from the data nodes without contacting again to NN
- Along with the data, check sum is also shipped for verifying the data integrity.
 - If the replica is corrupted client intimates NN, and client try to get the data from other DN
 - NN once intimated about the corrupted block will delete and then replicate to bring the replication factor to normal

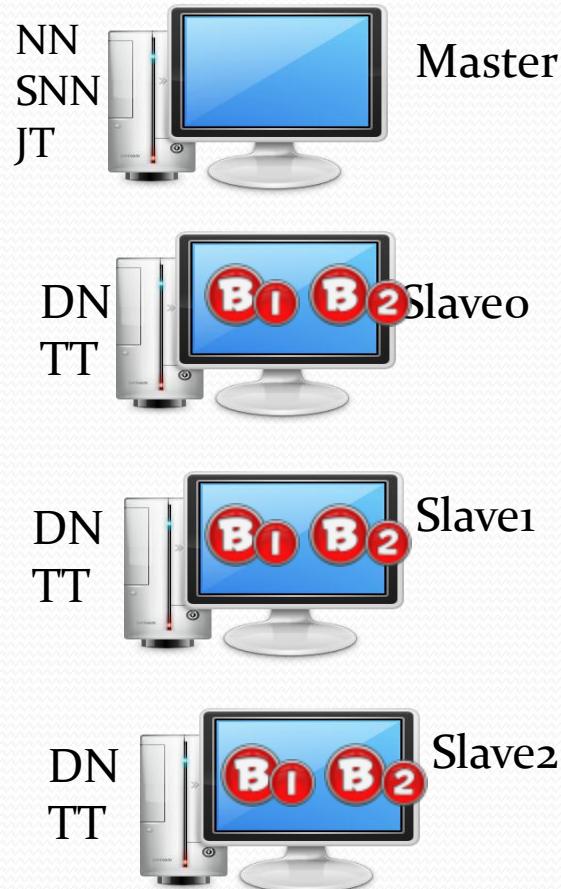
Reading a file to HDFS



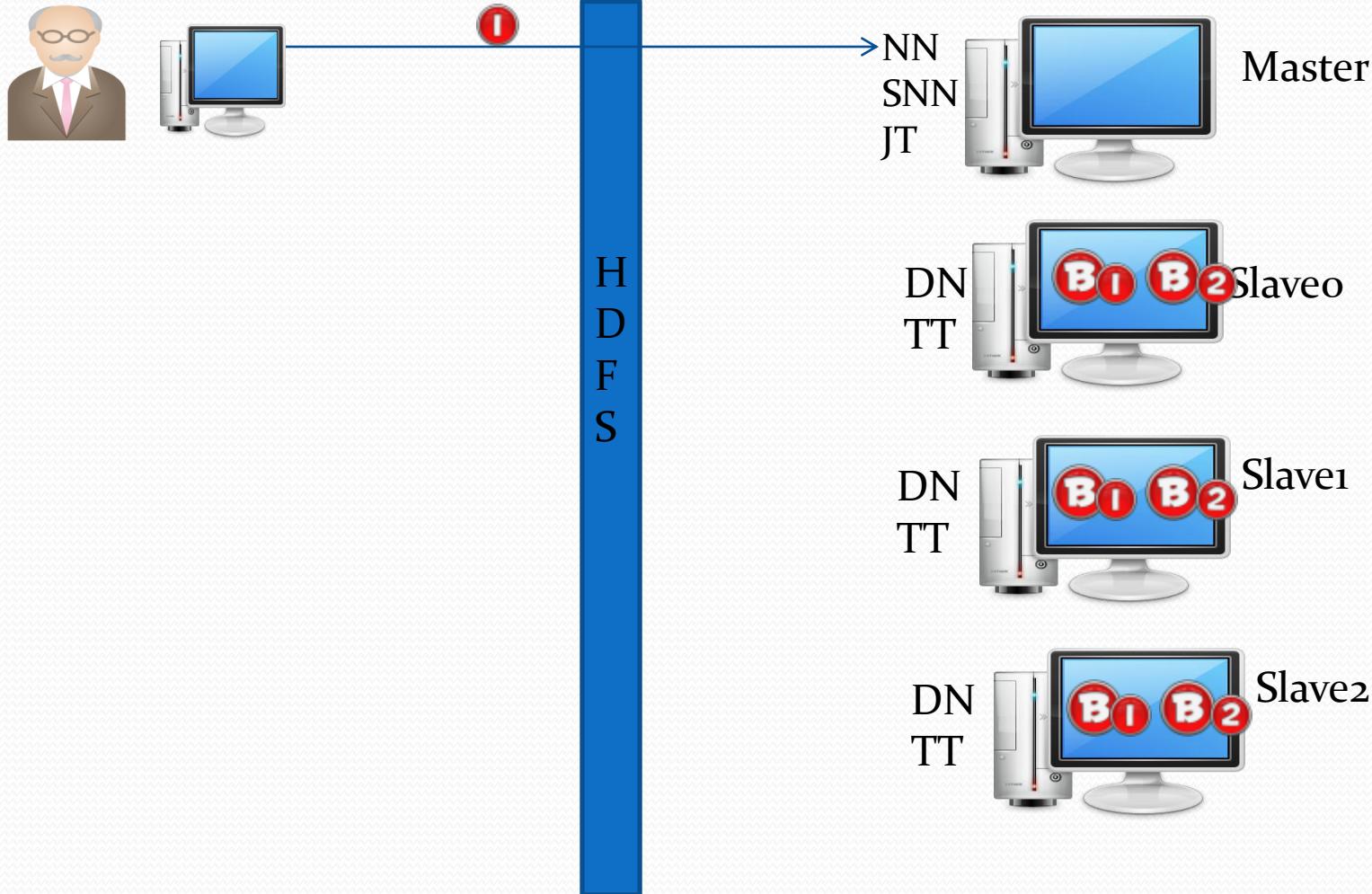
```
>>hadoop fs -get sample sample_local
```



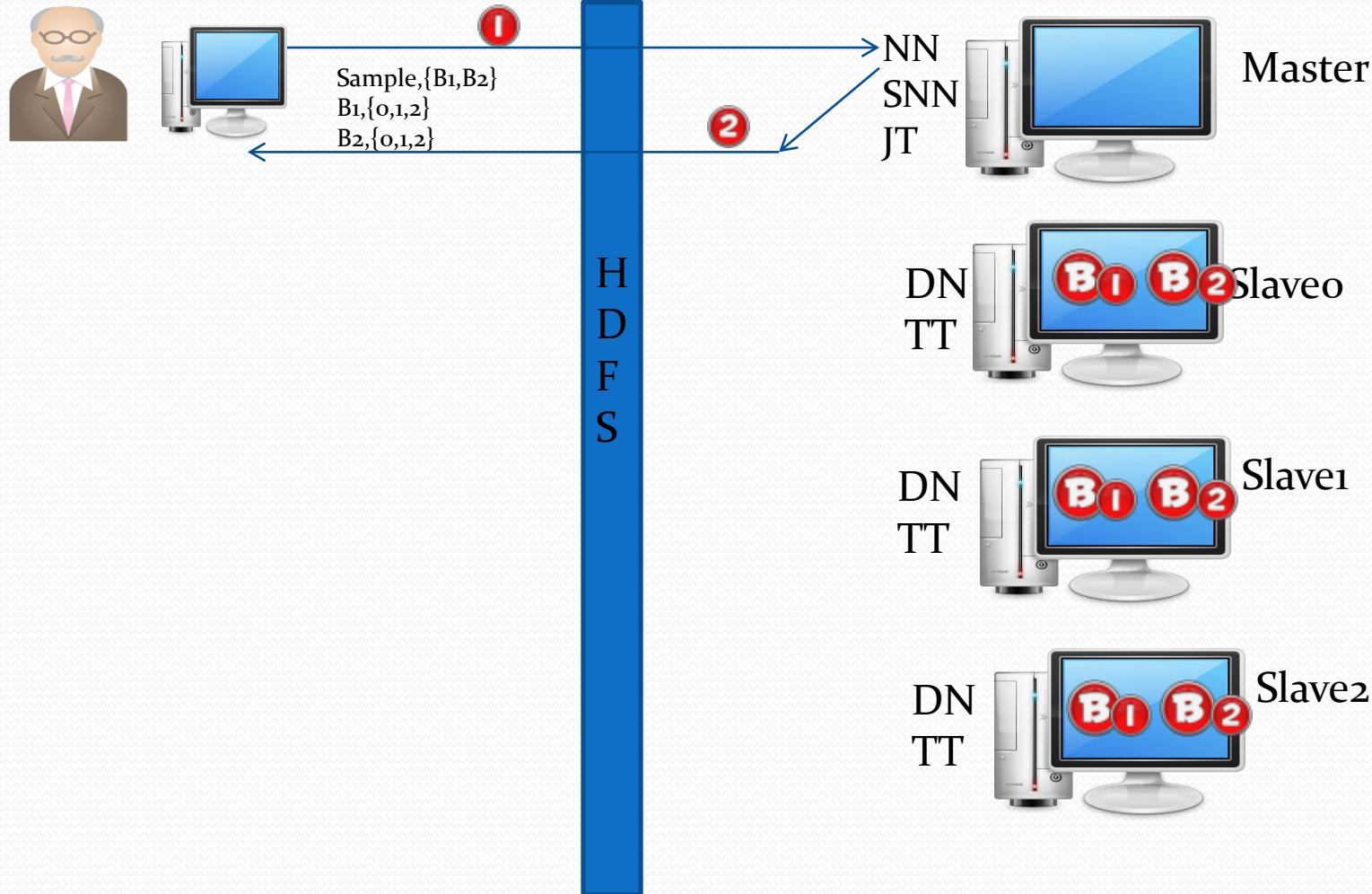
Metadata:
Sample,{B1,B2}
B1,{0,1,2}
B2,{0,1,2}



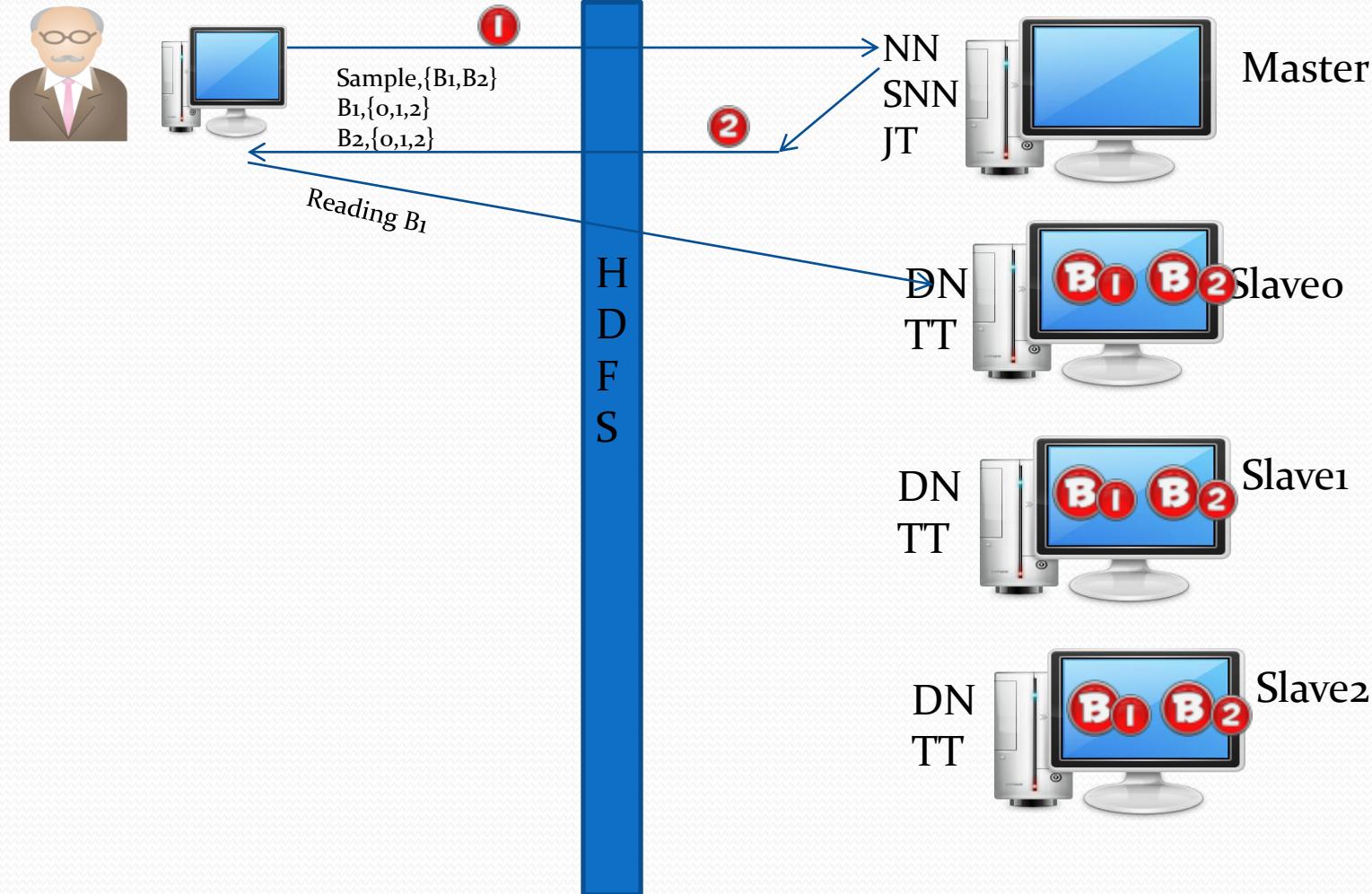
Reading a file to HDFS



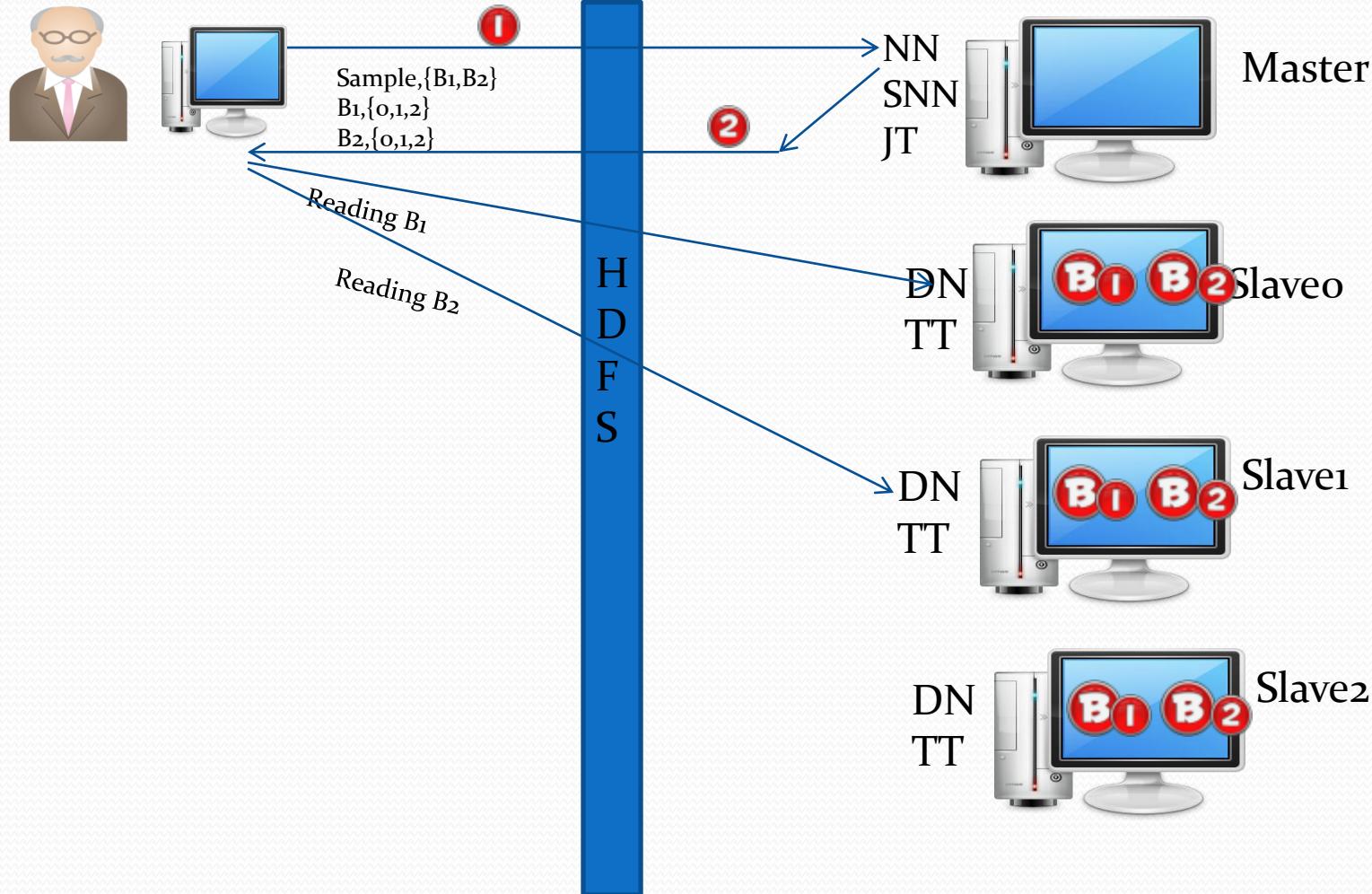
Reading a file to HDFS



Reading a file to HDFS



Reading a file to HDFS



Reading a file to HDFS

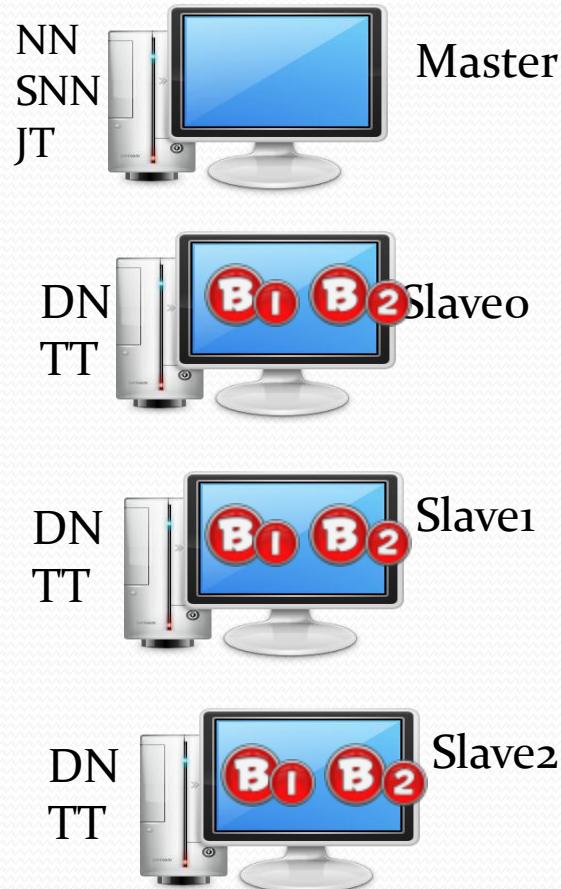


```
>>hadoop fs -get sample sample_local
```

H
D
F
S

It merges B1 and B2 and make a single file sample_local

Metadata:
Sample,{B1,B2}
B1,{0,1,2}
B2,{0,1,2}



How does HDFS handles corruption of blocks

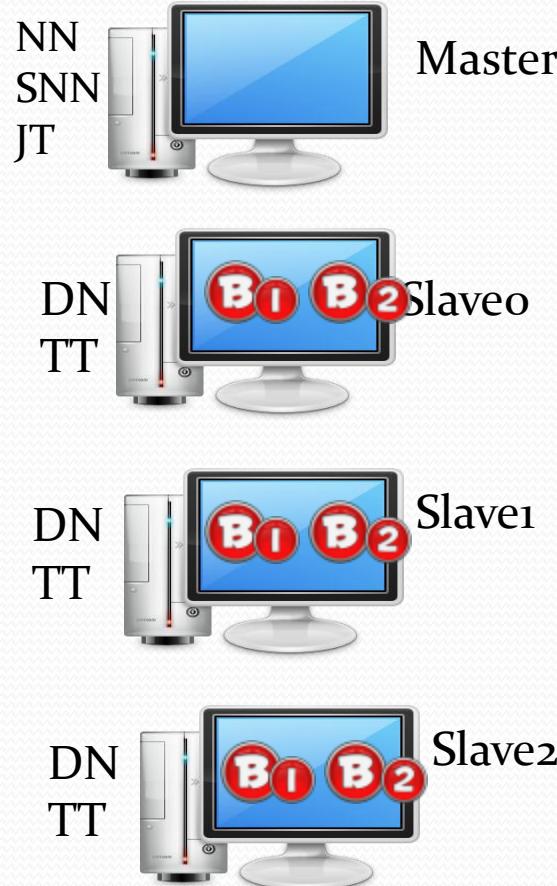


```
>>hadoop fs -get sample sample_local
```

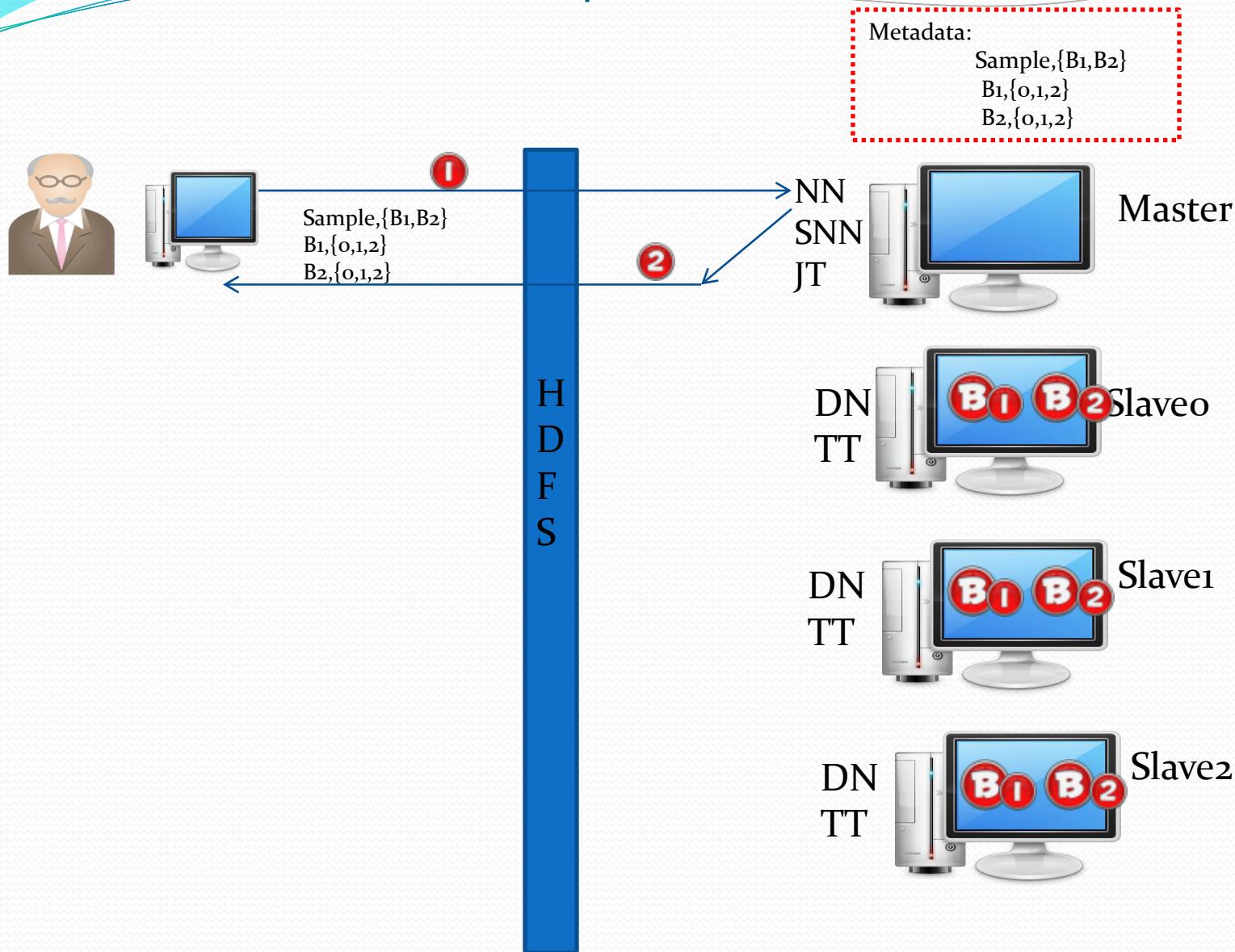
H
D
F
S

It merges B1 and B2 and make a single file sample_local

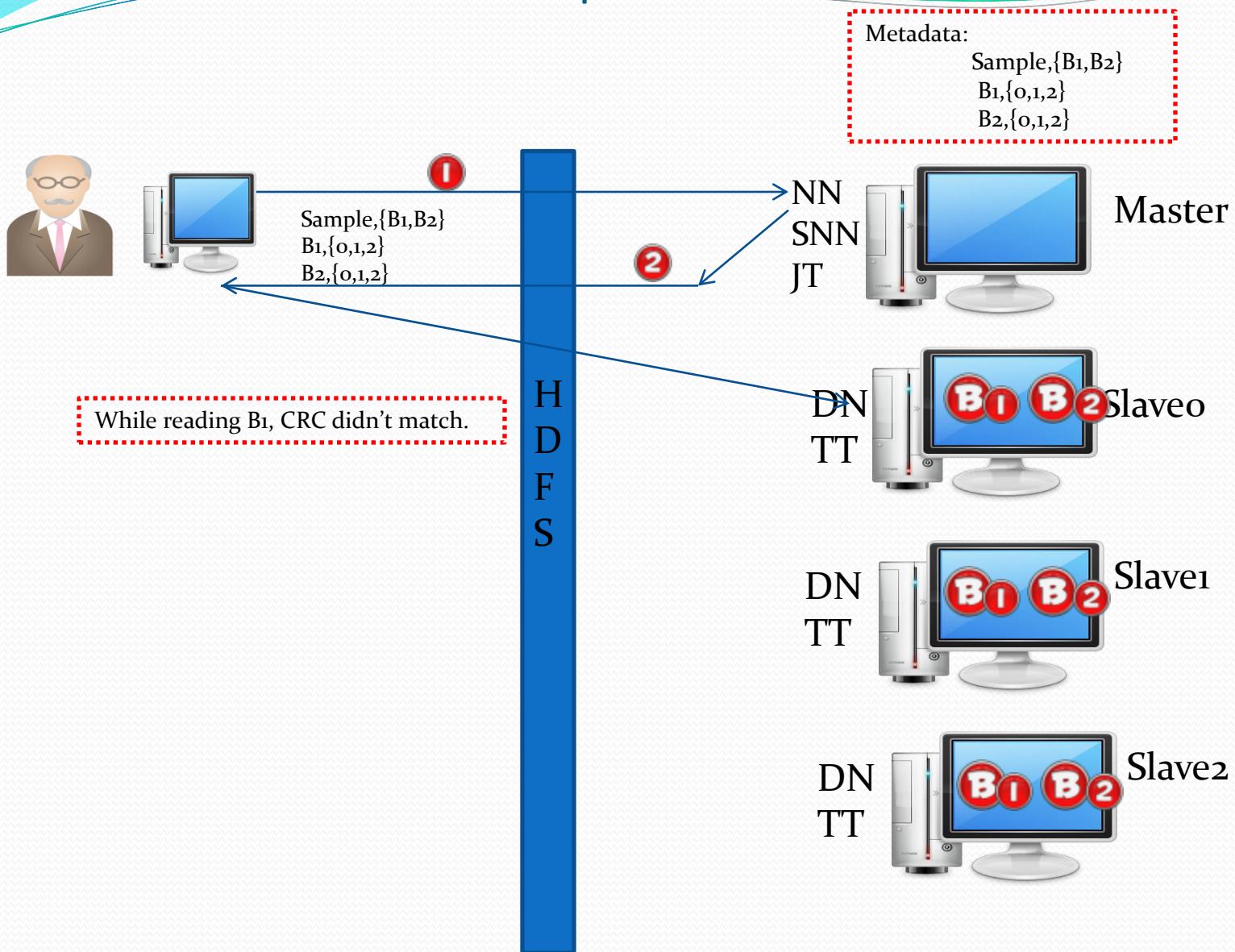
Metadata:
Sample,{B1,B2}
B1,{0,1,2}
B2,{0,1,2}



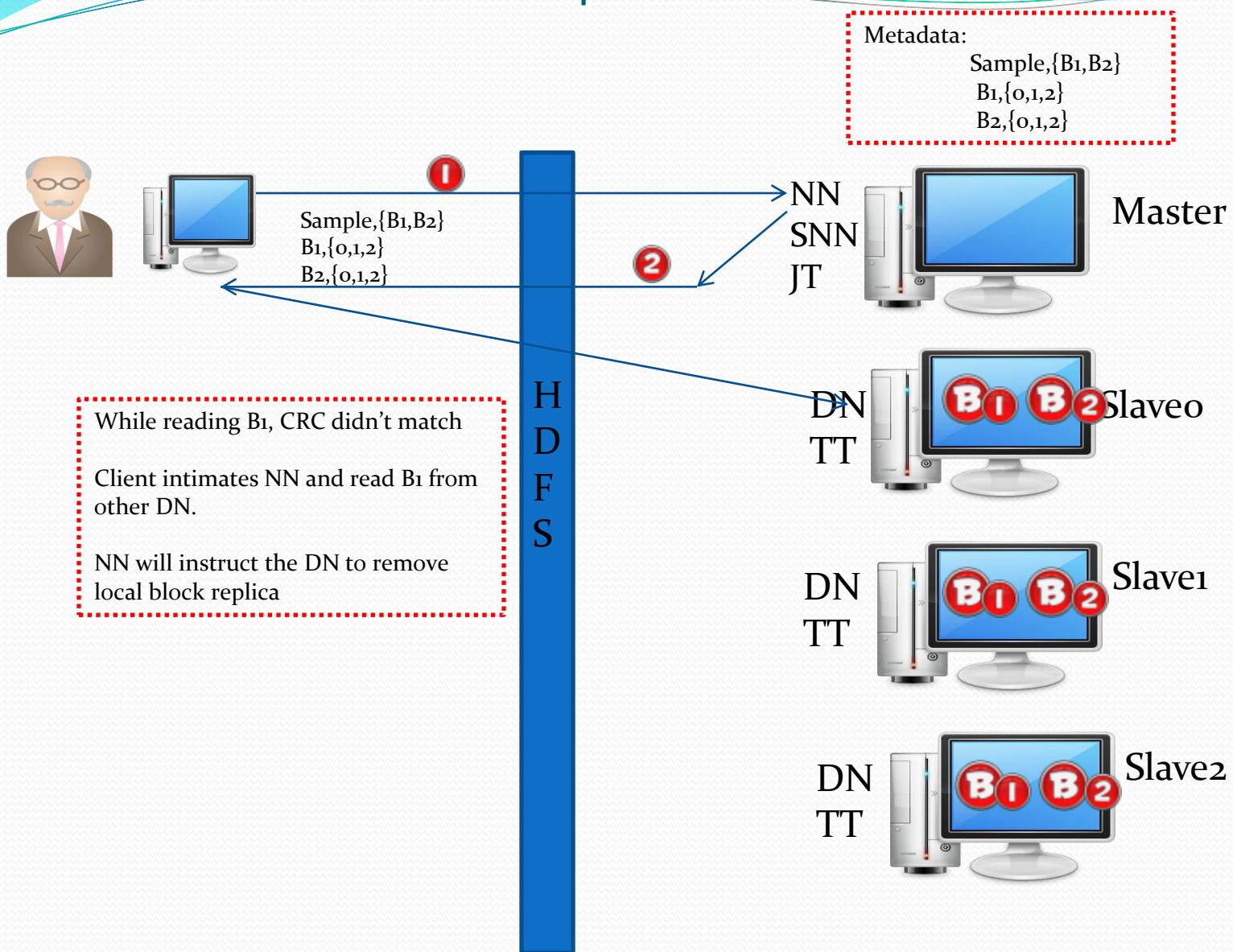
How does HDFS handles corruption of blocks



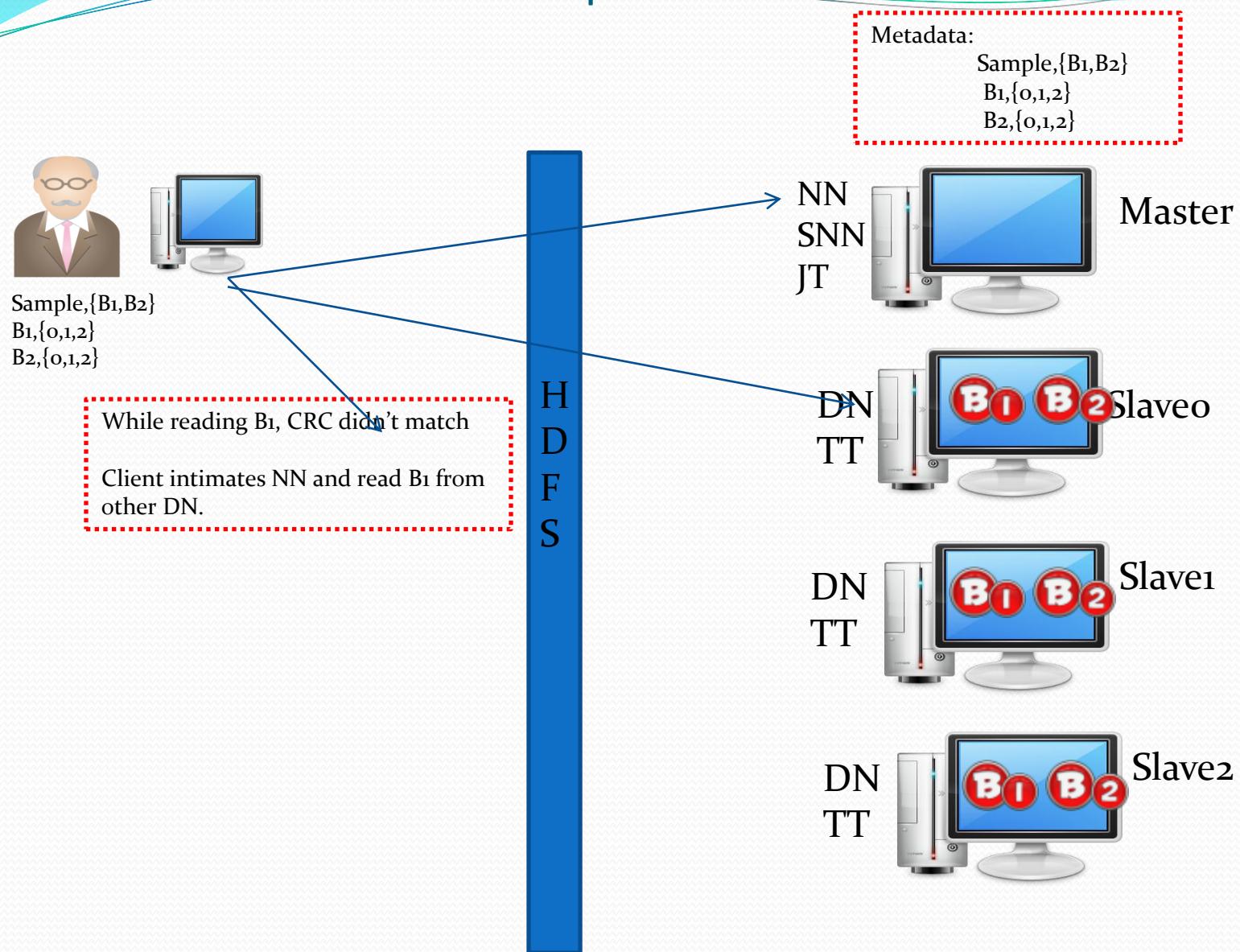
How does HDFS handles corruption of blocks



How does HDFS handles corruption of blocks



How does HDFS handles corruption of blocks



Module 8

HDFS API

Accessing the file system

- Require 3 things:
 - Configuration object
 - Path object
 - FileSystem instance

Hadoop's Configuration

- Encapsulates client and server configuration
- Use *Configuration* class to access the file system.
- *Configuration* object requires how you want to access the Hadoop cluster
 - Local File System
 - Pseudo Mode
 - Cluster Mode

Hadoop's Path

- File on HDFS is represented using Hadoop's Path object
- Path is similar to HDFS URI such as
`hdfs://localhost:54310/user/dev/sample.txt`

FileSystem API

- General file system Api

- *public static FileSystem get(Configuration conf) throws IOException*
- *public static FileSystem get(URI uri, Configuration conf) throws IOException*

Accessing the file system contd..

Step1 : Create a new configuration object

```
Configuration conf = new Configuration();
```

Step2 : Setting the name node path

```
conf.set("fs.default.name","hdfs://localhost:54310");
```

Step 3: Get the filesystem instance

```
FileSystem fs = FileSystem.get(conf);
```

Accessing the file system contd..

Step1 : Create a new configuration object

```
Configuration conf = new Configuration();
```

- Create a configuration object
 - Using this configuration object, you will connect to the Hadoop cluster
 - You need to tell this configuration object where is your NameNode and Job Tracker running

Accessing the file system contd..

Step1 : Create a new configuration object

```
Configuration conf = new Configuration();
```

Step2 : Setting the name node path

```
conf.set("fs.default.name","hdfs://localhost:54310");
```

- Setting the property “fs.default.name”, which tells the address of the name node

Accessing the file system contd..

Step1 : Create a new configuration object

```
Configuration conf = new Configuration();
```

Step2 : Setting the name node path

```
conf.set("fs.default.name","hdfs://localhost:54310");
```

Step 3: Get the filesystem instance

```
FileSystem fs = FileSystem.get(conf);
```

- Finally getting the file system instance
- Once you get the file system you can do any kind of file operations on HDFS

Hands On

- Refer Hands-on document

Module 9

Map Reduce Framework - Introduction

In this module you will learn

- What is MapReduce job?
- What is input split?
- What is mapper?
- What is reducer?

What is MapReduce job?

- It's a framework for processing the data residing on HDFS
 - Distributes the task (map/reduce) across several machines
- Consist of typically 5 phases:
 - Map
 - Partitioning
 - Sorting
 - Shuffling
 - Reduce
- A single map task works typically on one block of data (dfs.block.size)
 - Number of map tasks = number of blocks the data set has
 - By default each map task process a block of data (dfs.block.size)
 - Typically means the amount of data each map task process can be changed and hence the number of map tasks that will be running can be changed. (More on this later)
- After all map tasks are completed the output from the map is passed on to the machines where reduce task will run

What is MapReduce job?

File Size = 128MB

Block Size = 64MB

Replication Factor = 3

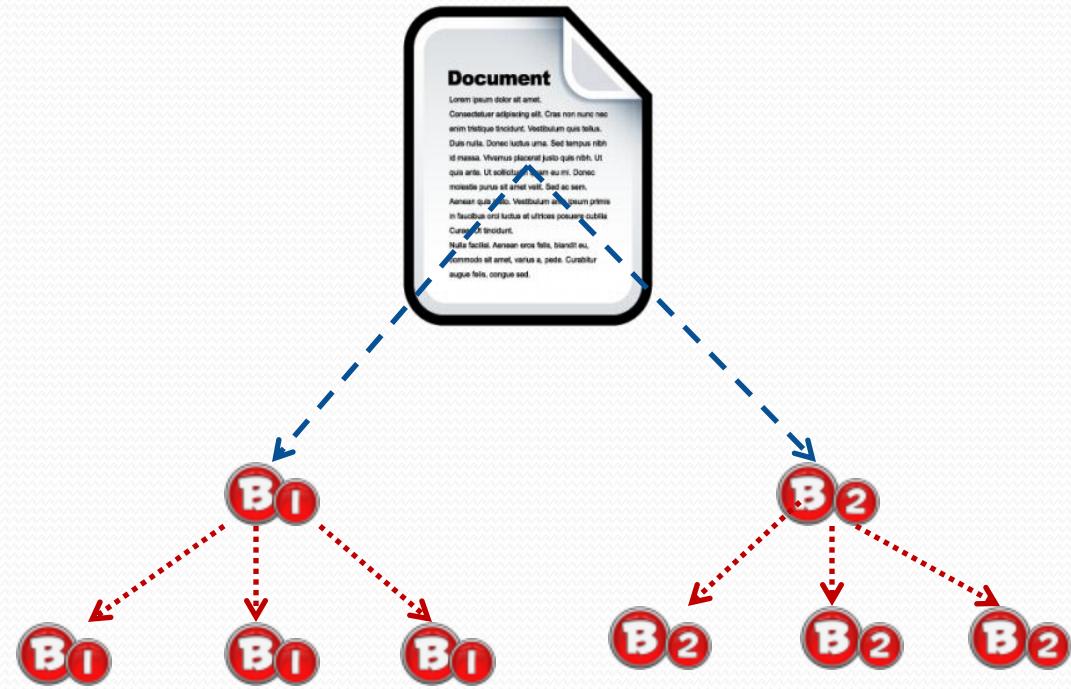


What is MapReduce job?

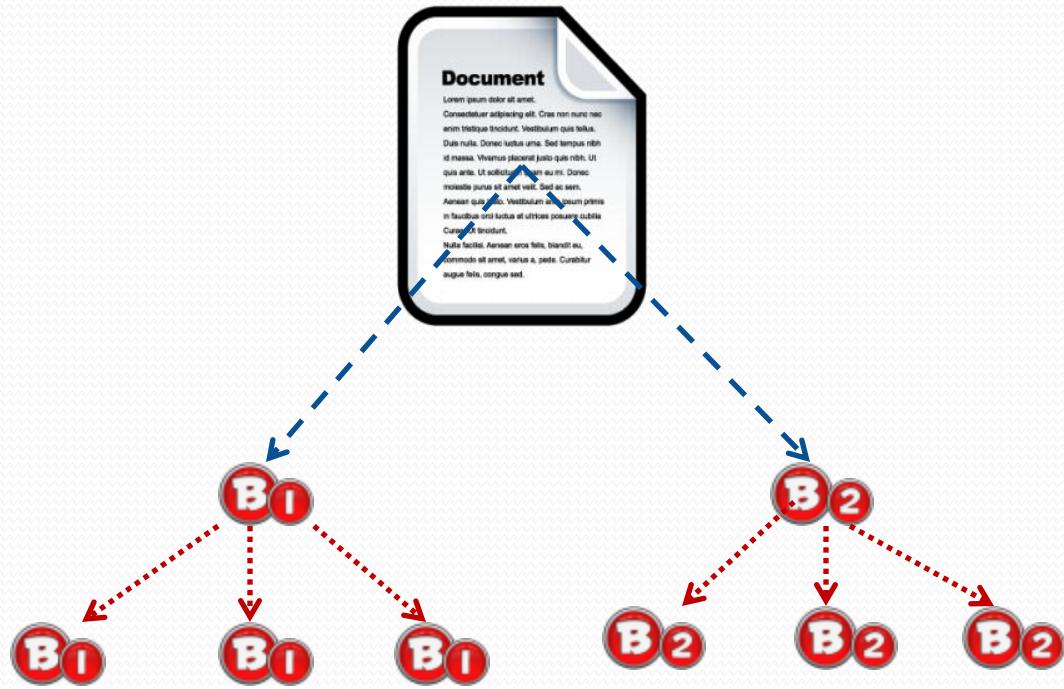
File Size = 128MB

Block Size = 64MB

Replication Factor = 3



What is MapReduce job?



Replica are distributed across several slave machines

What is MapReduce job?



What is MapReduce job?



How many number of blocks?



What is MapReduce job?



Number of blocks = 2

Number of map tasks = 2

Each map task runs on its own JVM and hence they are independent



What is MapReduce job?



Each map task is processing different blocks of data



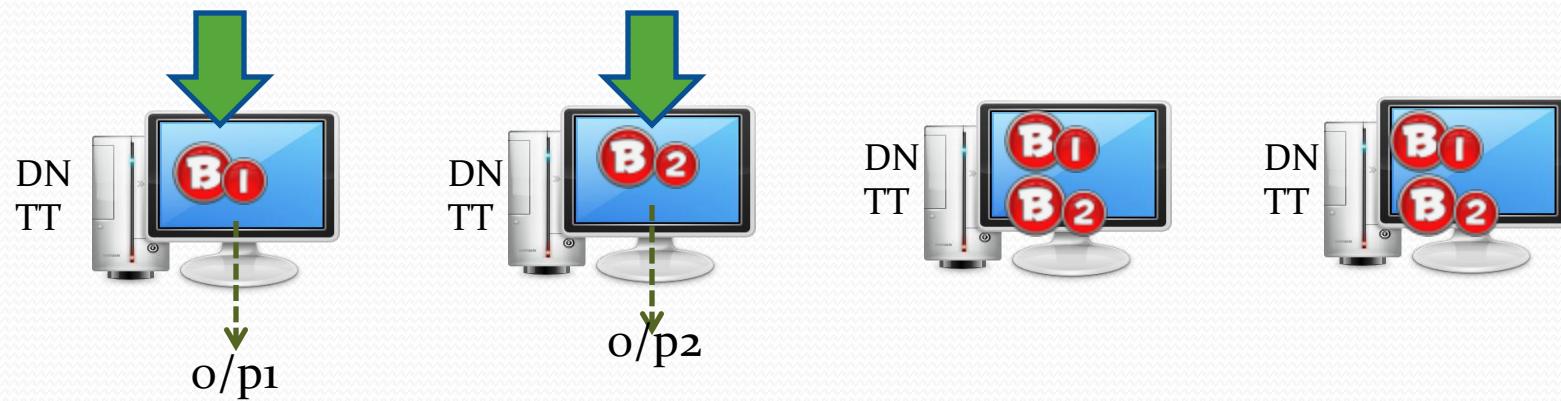
What is MapReduce job?



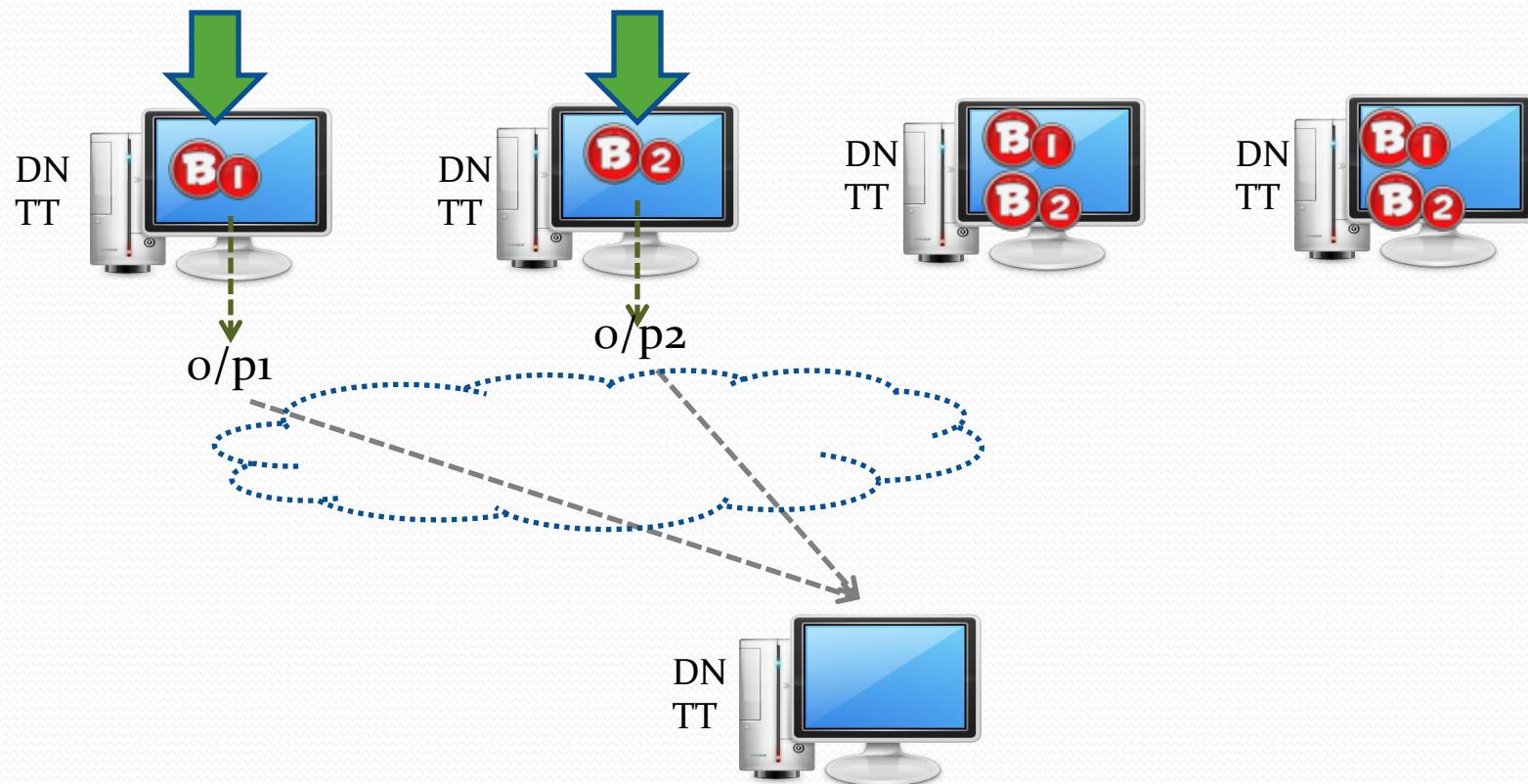
Once the map task is completed, the output will be transferred to the machine where reducer task is running



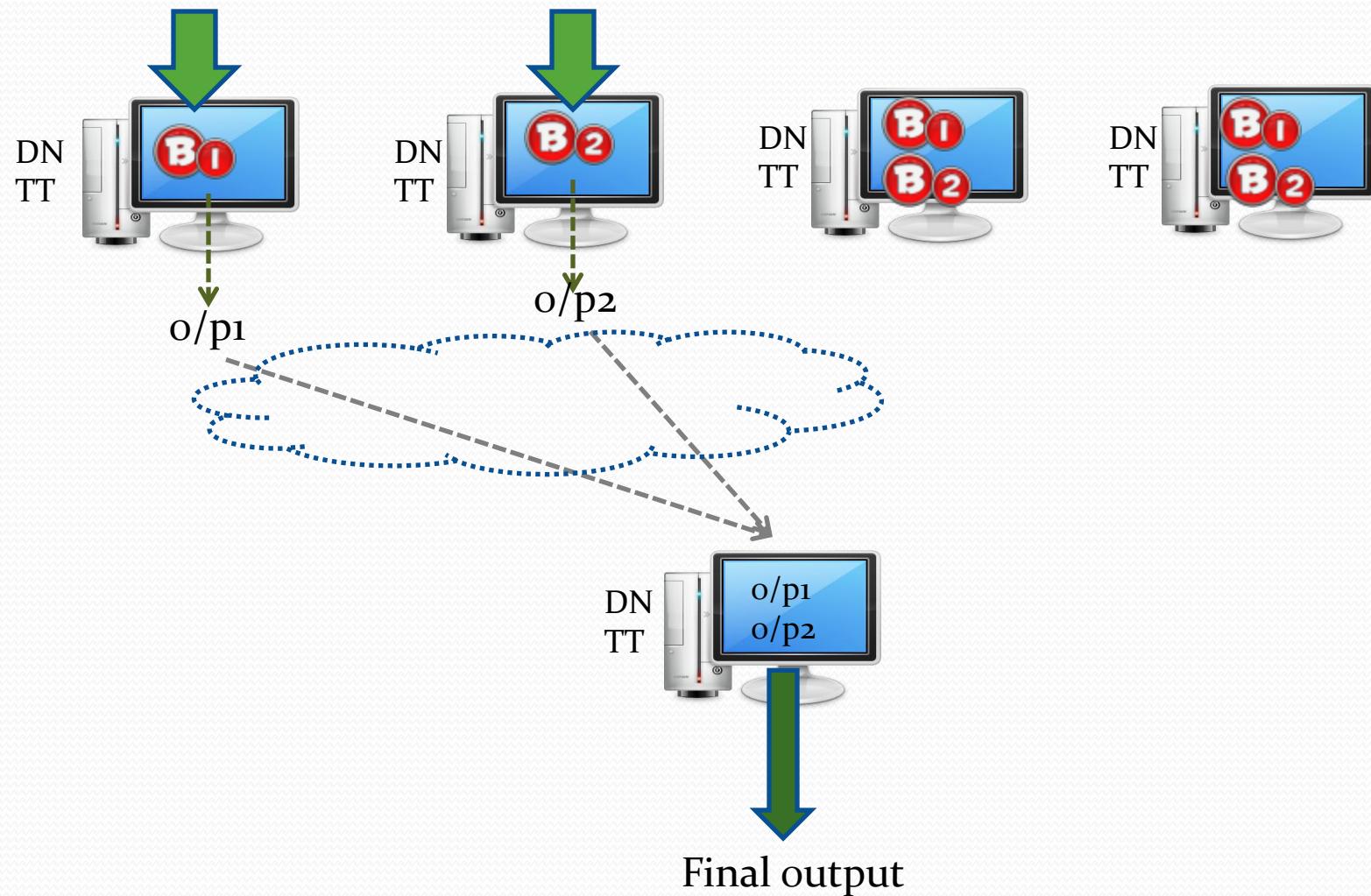
What is MapReduce job?



What is MapReduce job?



What is MapReduce job?



MapReduce Terminology

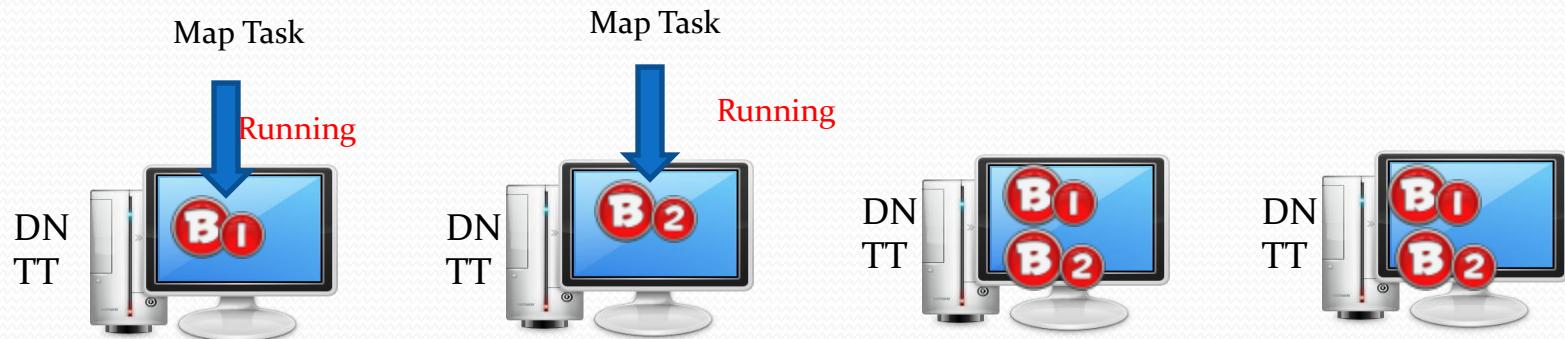
- What is job?
 - Complete execution of mapper and reducers over the entire data set
 - Map Reduce job is also known as batch mode.
- What is task?
 - Single unit of execution (map or reduce task)
 - Map task executes typically over a block of data (dfs.block.size)
 - Reduce task works on mapper output
- What is “*task attempt*”?
 - Instance of an attempt to execute a task (map or reduce task) successfully.
 - If a task fails, again it is re-attempted.
 - By default every task has 4 attempts
 - If a task fails 4 times successively, then irrespective of how many tasks have completed successfully entire job fails and no output is generated

Task Attempt



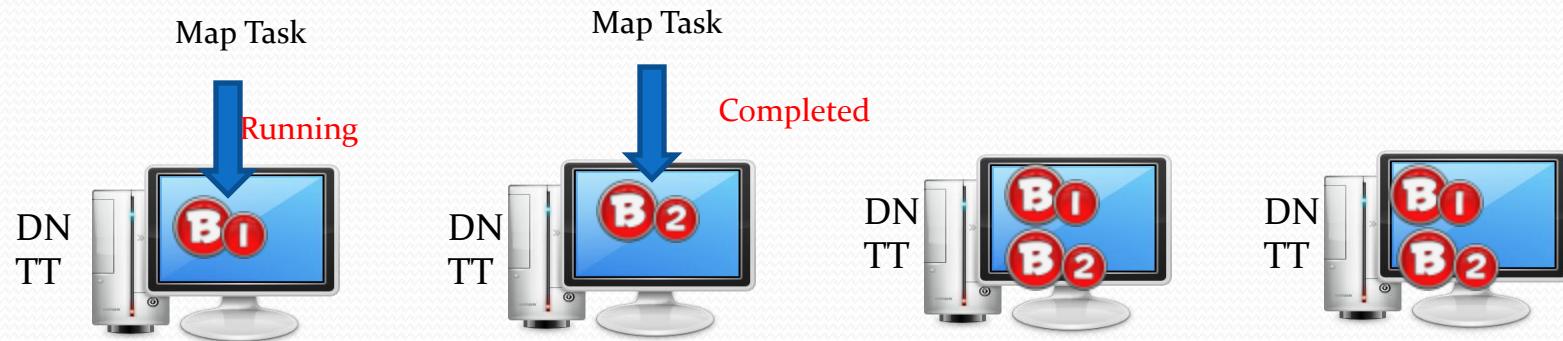
Number of blocks = 2
Number of map tasks = 2

Task Attempt

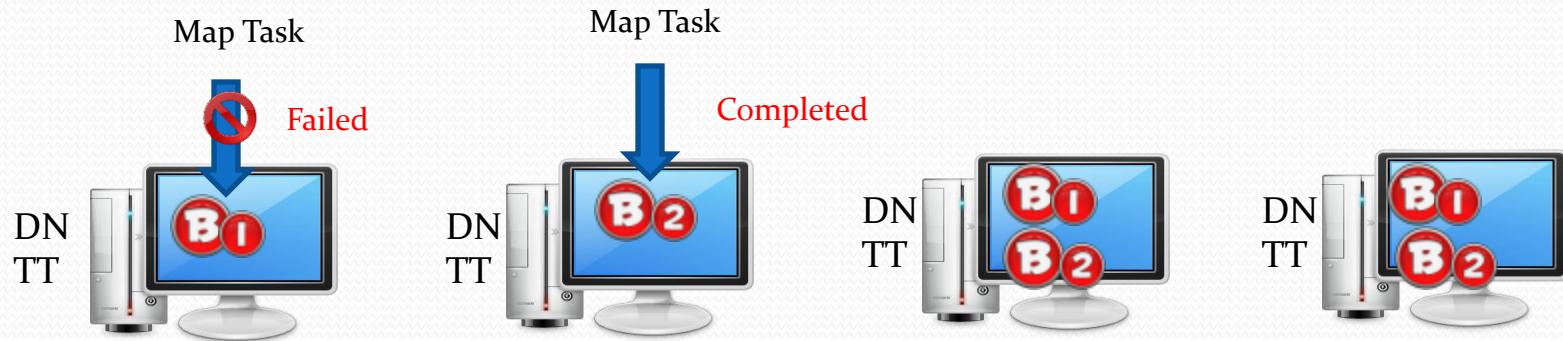


Tasks are running independently and processing block of data

Task Attempt

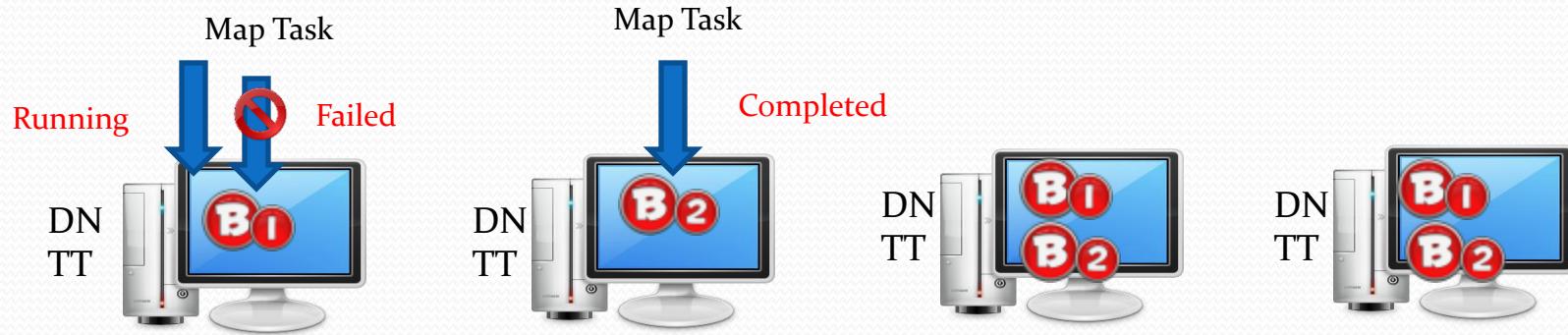


Task Attempt



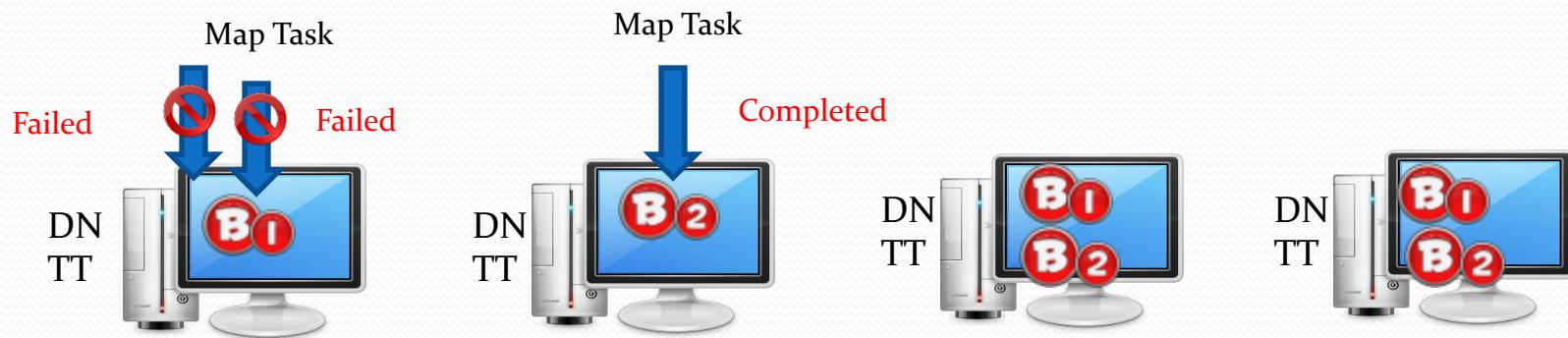
- Task failed (N/w failure, machine failure)
- First attempt failed
- Second re-attempt will be done

Task Attempt



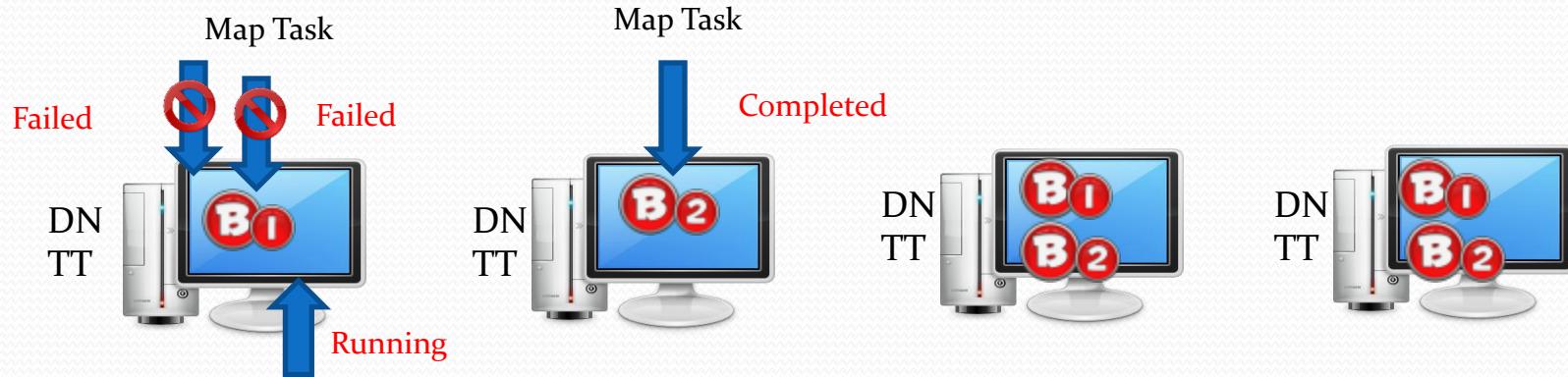
- Task failed (N/w failure, machine failure)
- First attempt failed
- Second re-attempt will be done

Task Attempt



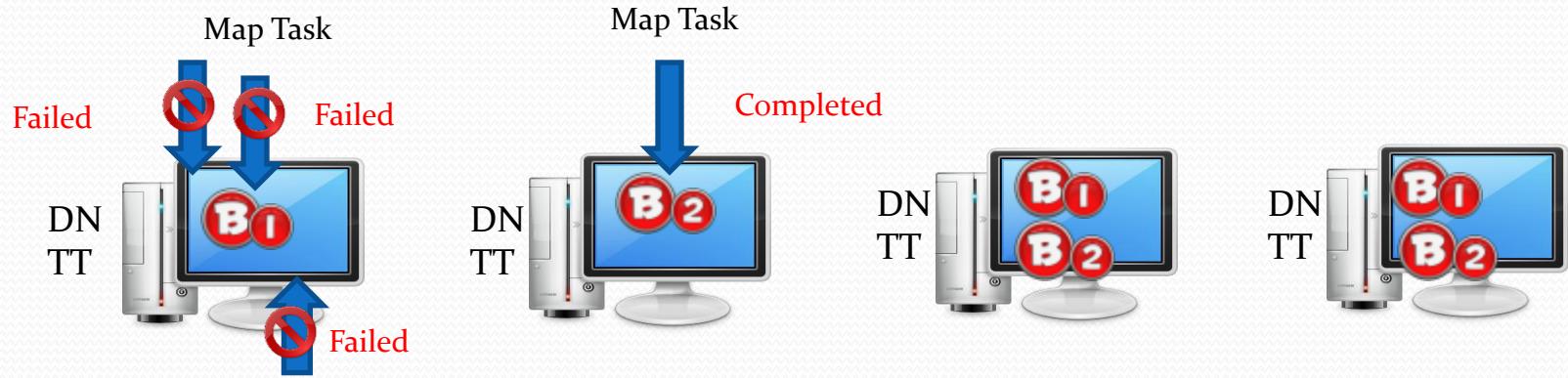
- Task failed (N/w failure, machine failure)
- First attempt failed
- Second re-attempt failed
- Third re-attempt will be done

Task Attempt



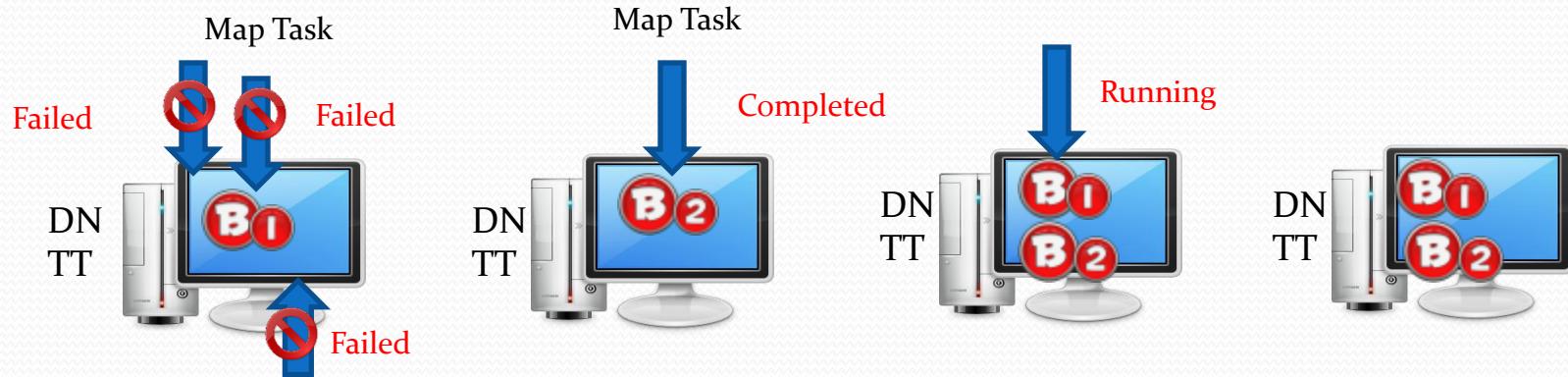
- Task failed (N/w failure, machine failure)
- First attempt failed
- Second re-attempt failed
- Third re-attempt will be done

Task Attempt



- Task failed (N/w failure, machine failure)
- First attempt failed
- Second re-attempt failed
- Third re-attempt failed
- Fourth and final re-attempt will be done

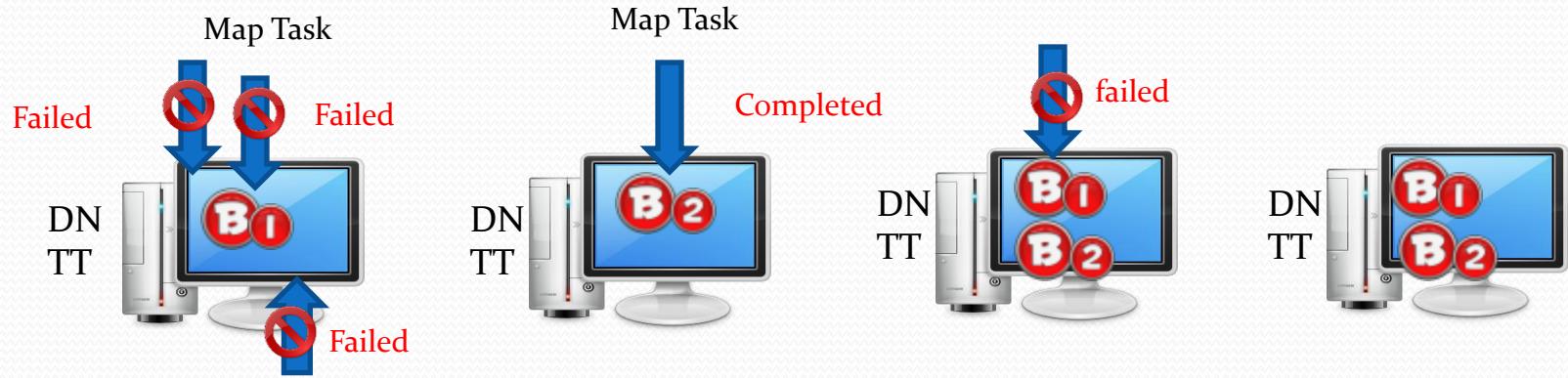
Task Attempt



- Task failed (N/w failure, machine failure)
- First attempt failed
- Second re-attempt failed
- Third re-attempt failed
- Fourth and final re-attempt will be done

By default 4 attempts will be attempted to process a block of data

Task Attempt



- Task failed (N/w failure, machine failure)
- First attempt failed
- Second re-attempt failed
- Third re-attempt failed
- Fourth attempt failed

If any task fails by default 4 times, entire job fails and no output is generated

Terminology continued...

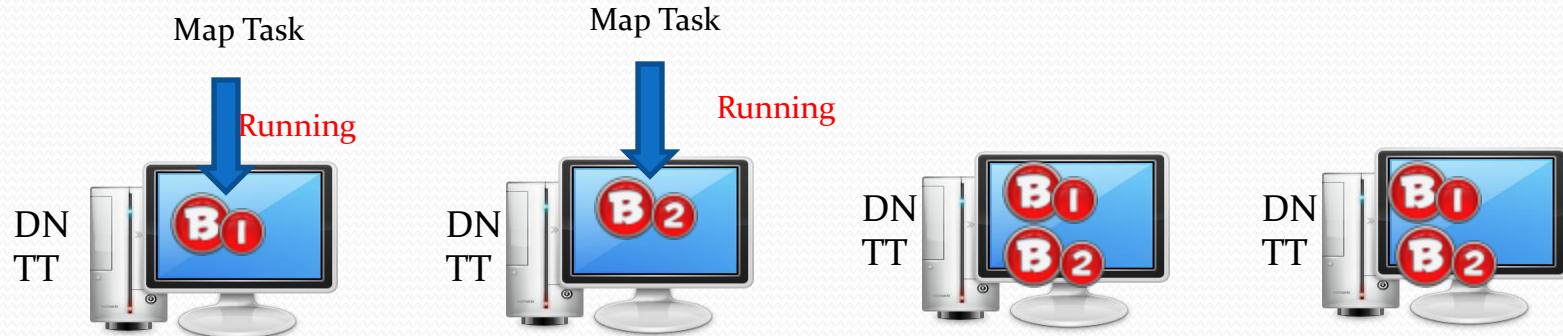
- How many tasks can run on portion of data?
 - Default 4
 - mapred.map.max.attempts & mapred.reduce.max.attempts
 - If “*speculative execution*” is ON, more task will run.
 - Speculative execution is a concept for improving the performance of a job, when few tasks are running slow.
- What is “*failed task*”?
 - Task can be failed due to exception, machine failure etc.
 - A failed task will be re-attempted again (4 times)
- What is “*killed task*”?
 - Task which runs as part of speculative execution are marked as killed and they are not re-attempted

Speculative Execution



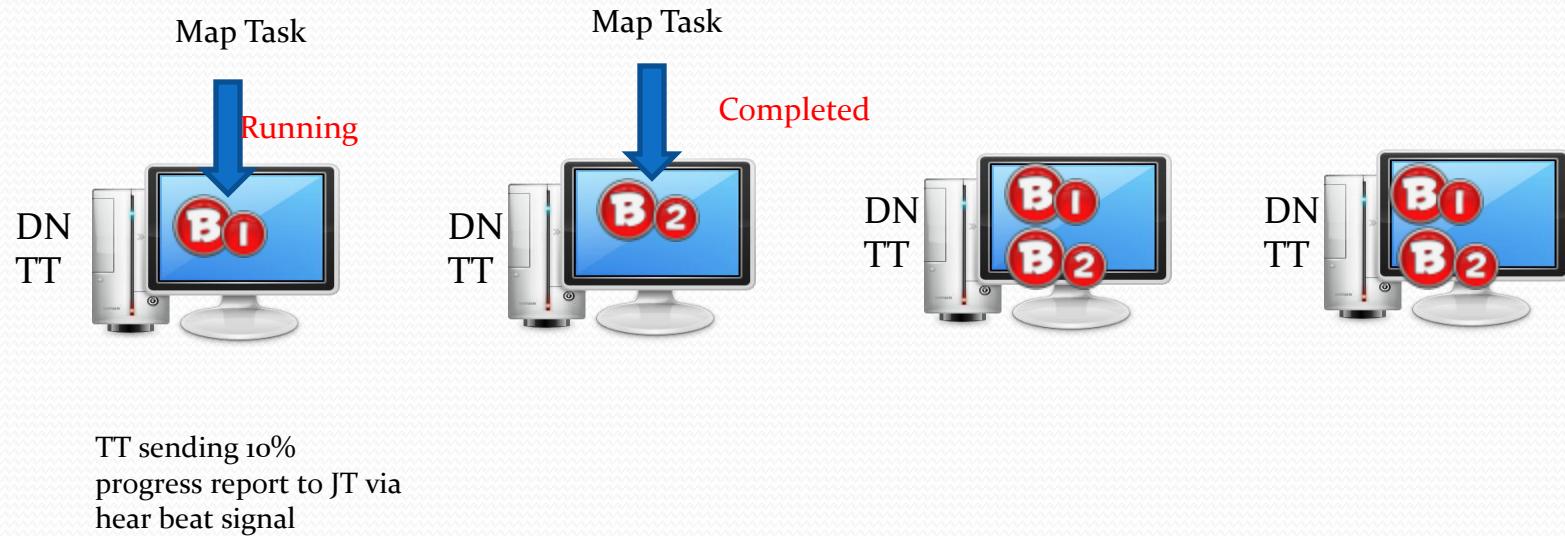
Number of blocks = 2
Number of map tasks = 2

Speculative Execution

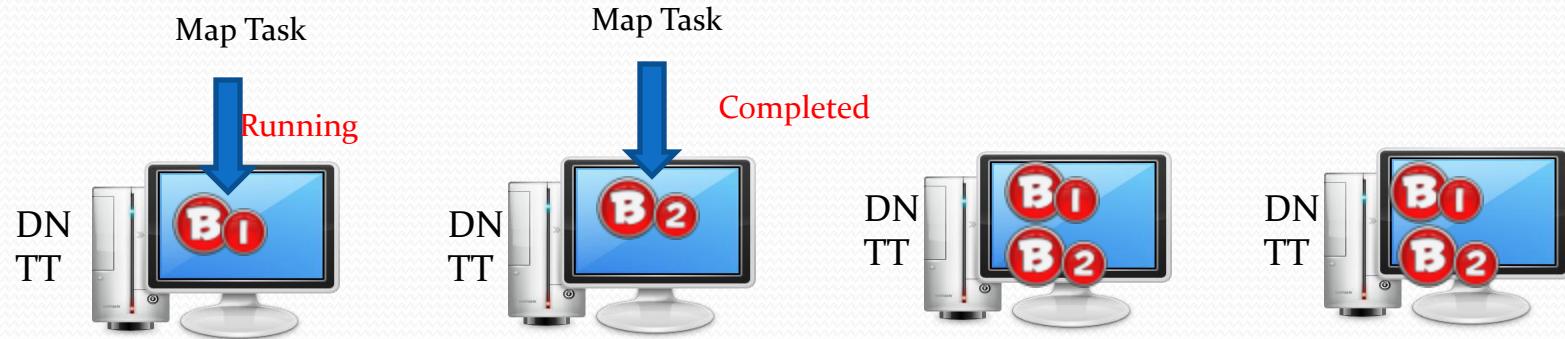


Tasks are running independently and processing block of data

Speculative Execution

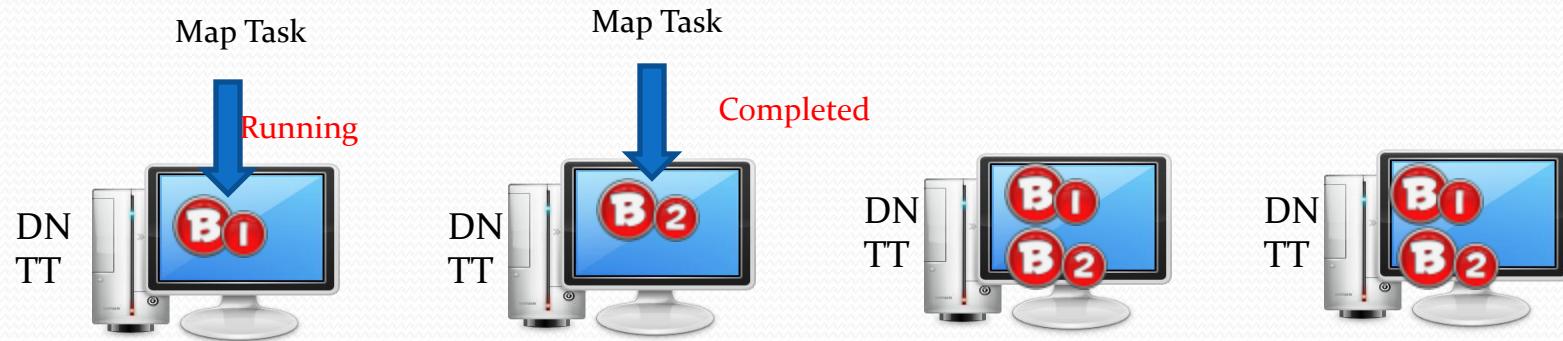


Speculative Execution



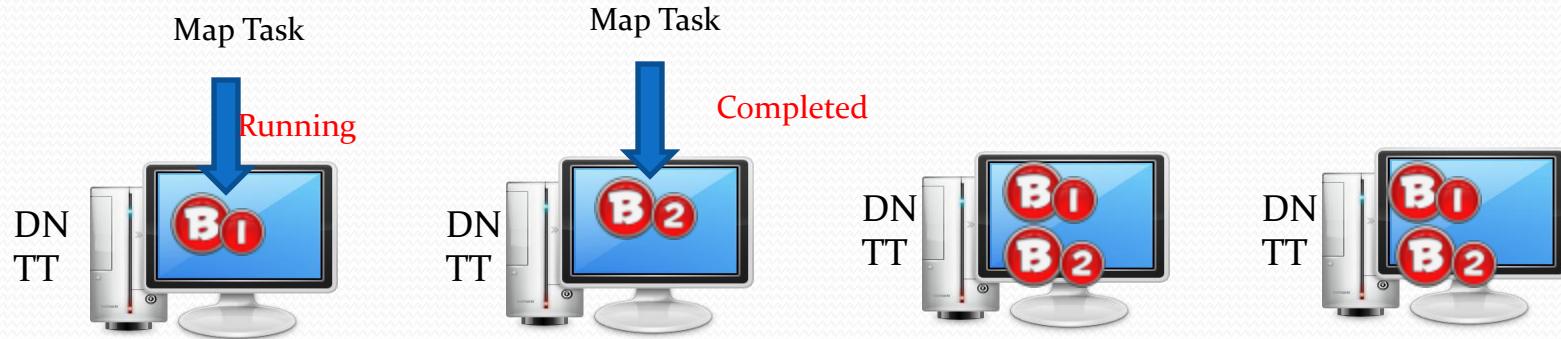
TT again sending 10% progress report to JT via hear beat signal

Speculative Execution



TT again sending 10% progress report to JT via hear beat signal

Speculative Execution

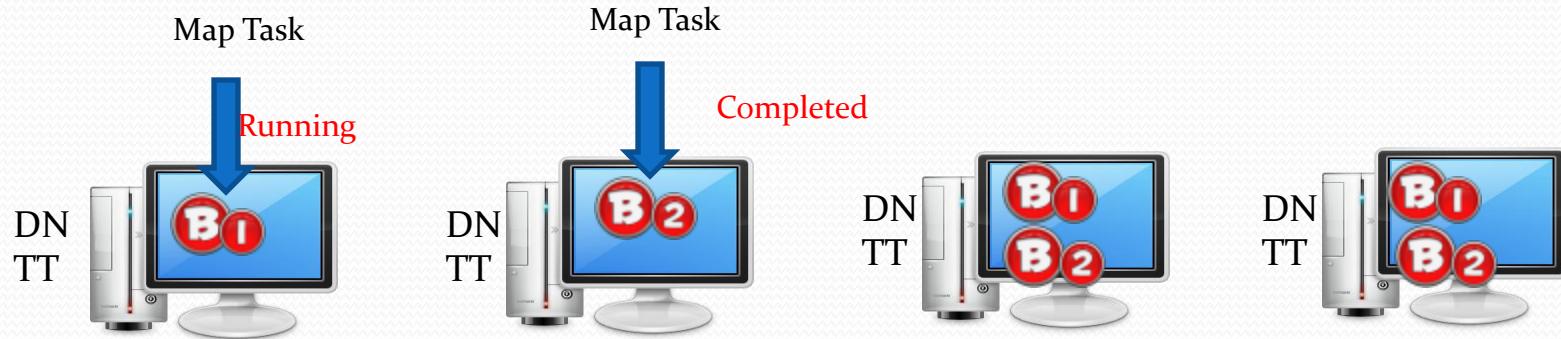


TT again sending 10% progress report to JT via hear beat signal

JT is receiving the same progress report for 3-4 hear beat signal.

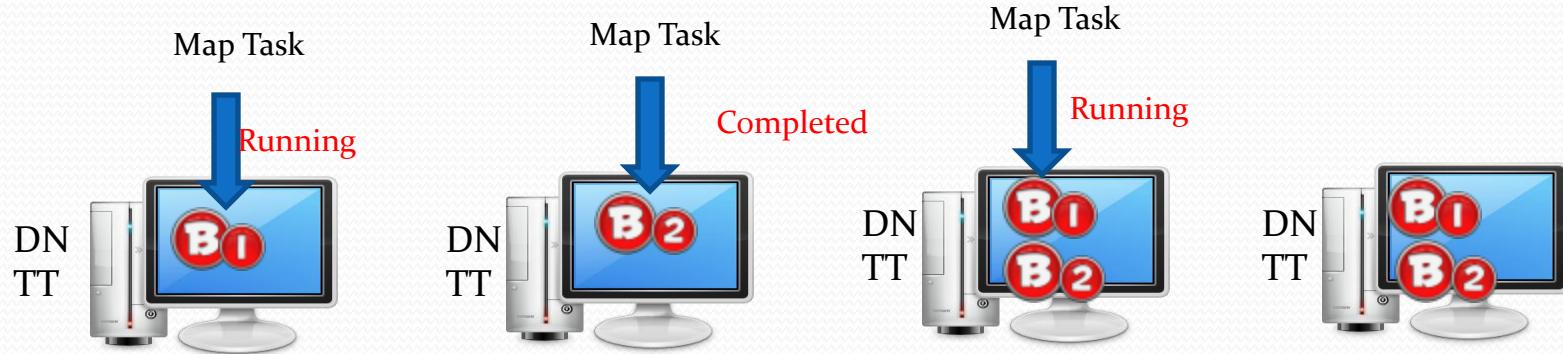
JT assumes task is running slow

Speculative Execution



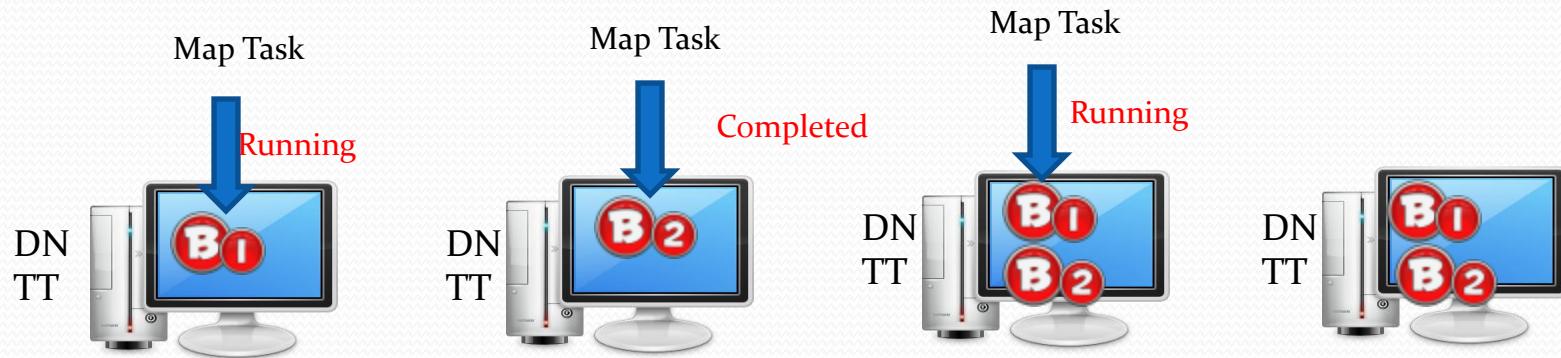
JT schedules another map task on the same block of data on some other machine

Speculative Execution



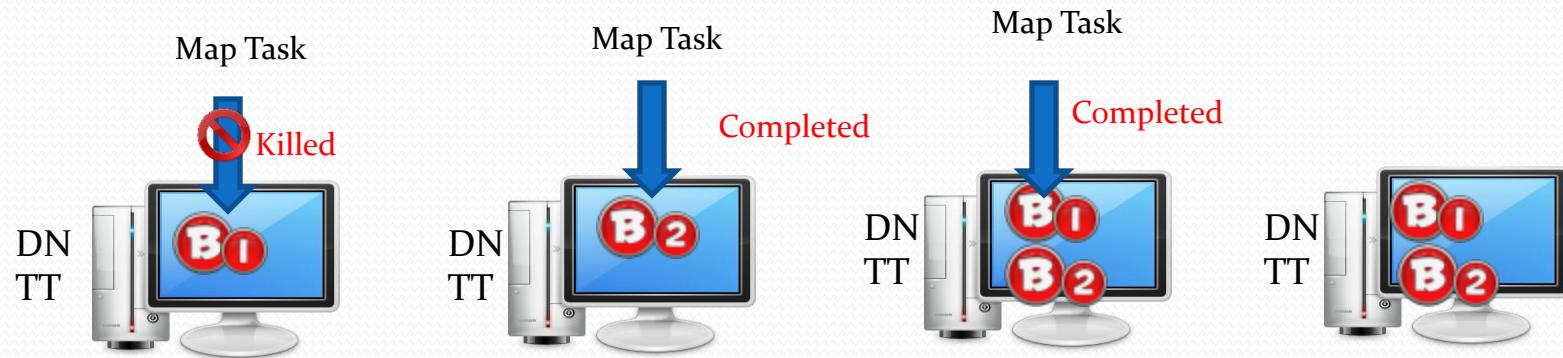
The two tasks are running parallelly on the same block of data. This is known as speculative execution

Speculative Execution



One of the task finishes first, the other task is killed and its output is rejected

Speculative Execution



Input Split

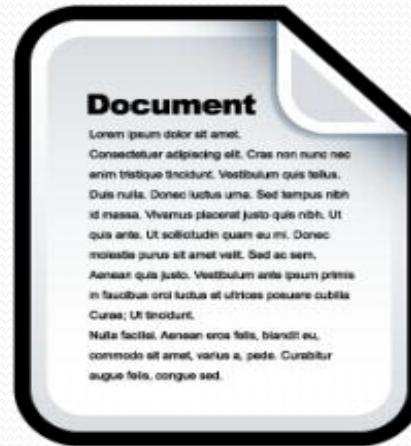
- Portion or chunk of data on which mapper operates
- Input split is just a reference to the data
- Number of map tasks = Number of input splits the data has
 - Number of input splits is determined by input split size
 - Default value of input split size is block size
- By changing the input split size, the number of input splits will change and hence the number of map tasks will change.

Input Split

File Size = 128MB

Block Size = 64MB

Replication Factor = 1

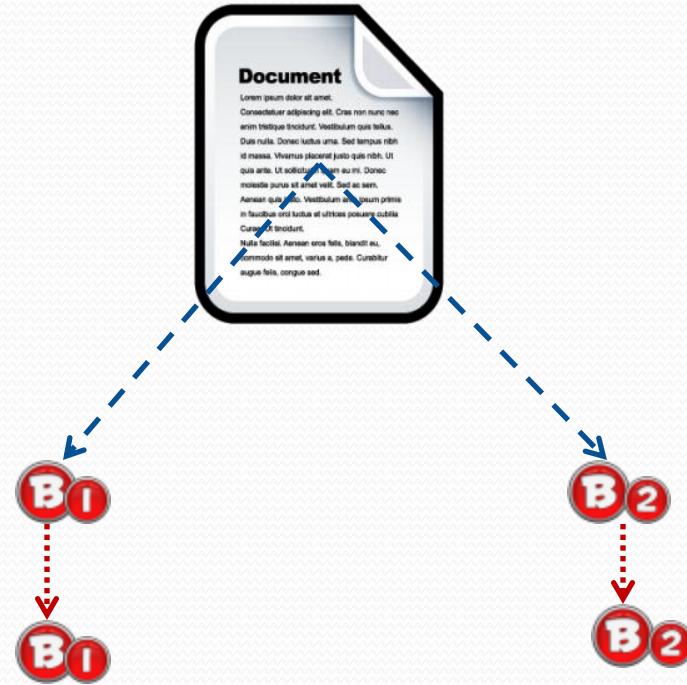


Input Split

File Size = 128MB

Block Size = 64MB

Replication Factor = 1

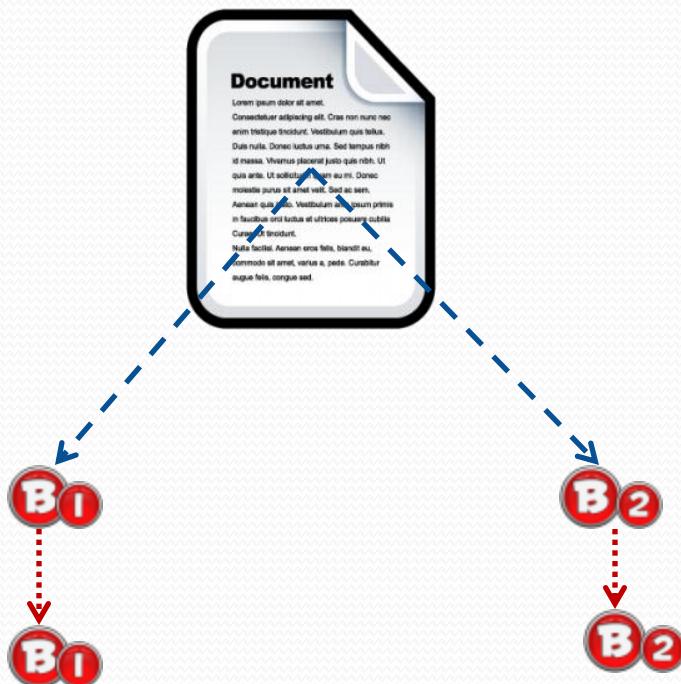


Input Split Size = Block Size

File Size = 128MB

Block Size = 64MB

Replication Factor = 1



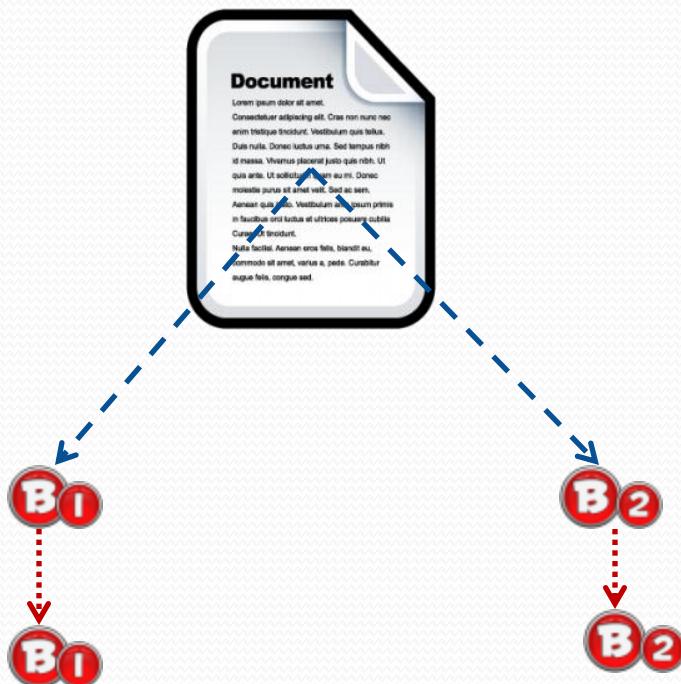
How many input splits?

Input Split Size = Block Size

File Size = 128MB

Block Size = 64MB

Replication Factor = 1



Number of input splits = 2

Hence number of map tasks = 2

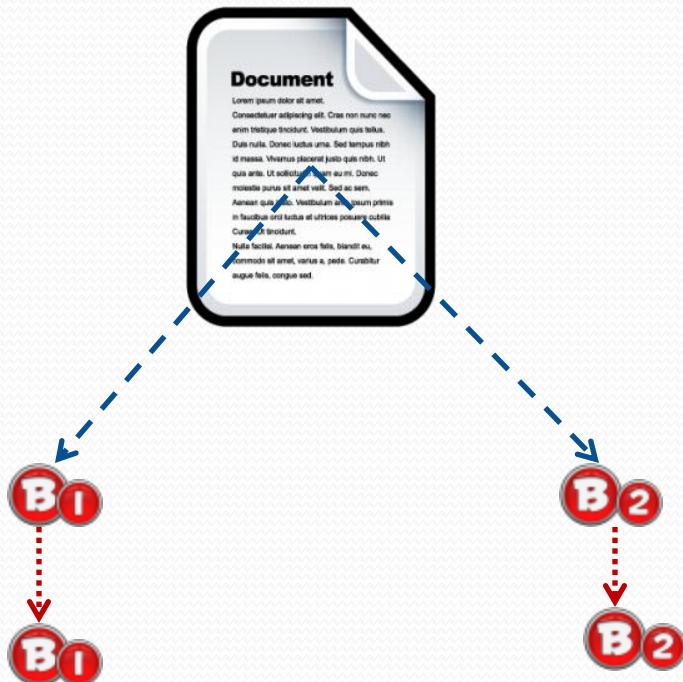
Each map task is going to process 64MB of data

Input Split Size = Block Size

File Size = 128MB

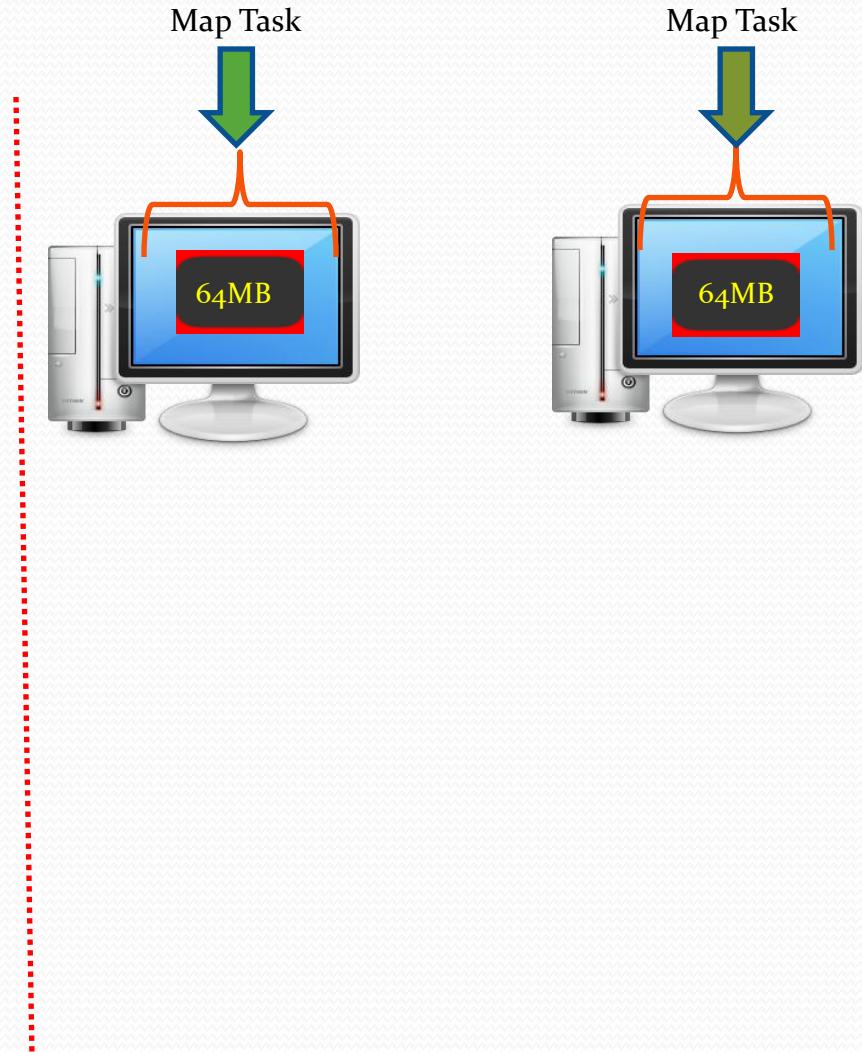
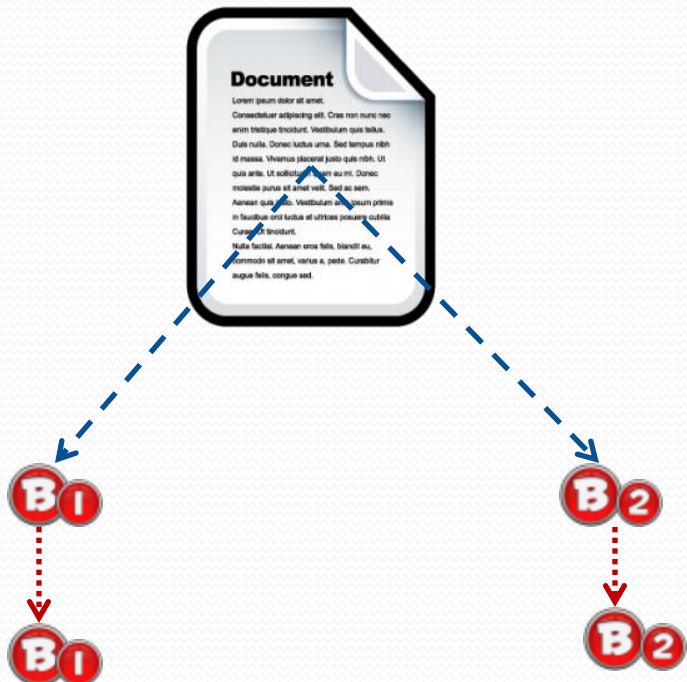
Block Size = 64MB

Replication Factor = 1



Input Split Size = Block Size

File Size = 128MB
Block Size = 64MB
Replication Factor = 1

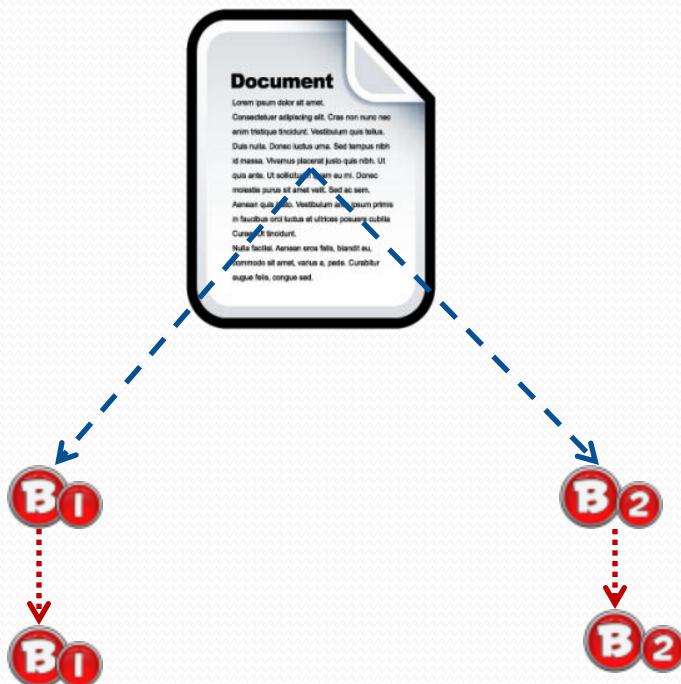


Input Split Size = 32MB

File Size = 128MB

Block Size = 64MB

Replication Factor = 1



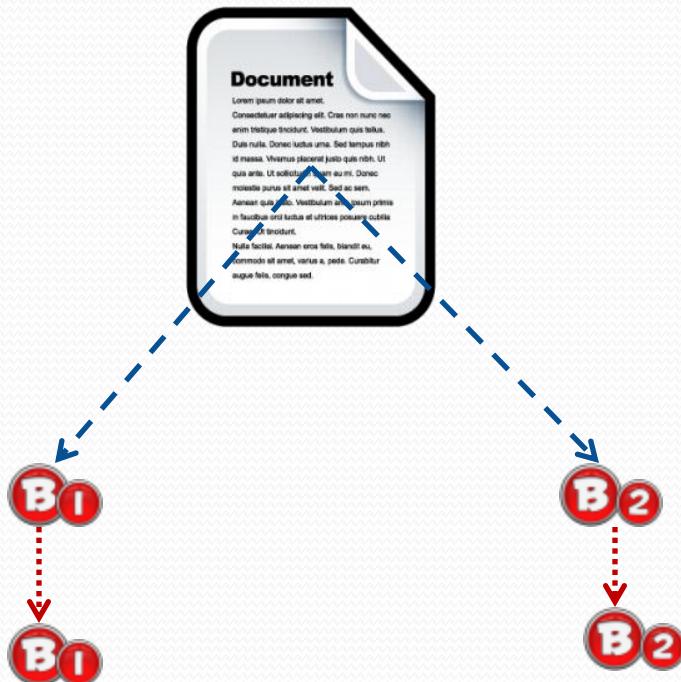
How many input splits?

Input Split Size = 32MB

File Size = 128MB

Block Size = 64MB

Replication Factor = 1



Number of input splits = 4

Hence number of map tasks = 4

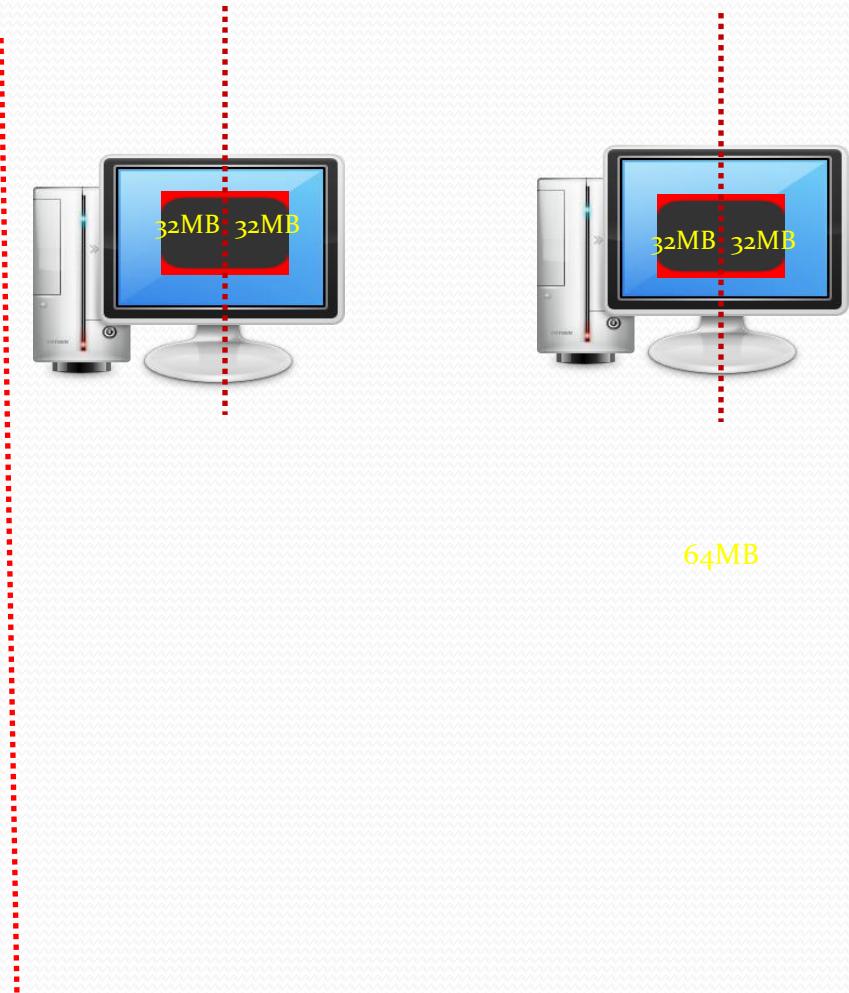
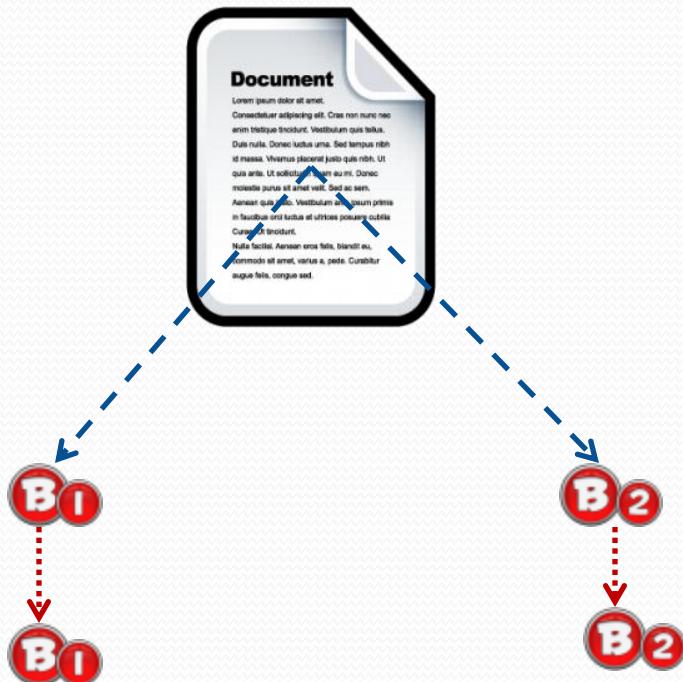
Each map task is going to process 32MB of data

Input Split Size = 32MB

File Size = 128MB

Block Size = 64MB

Replication Factor = 1

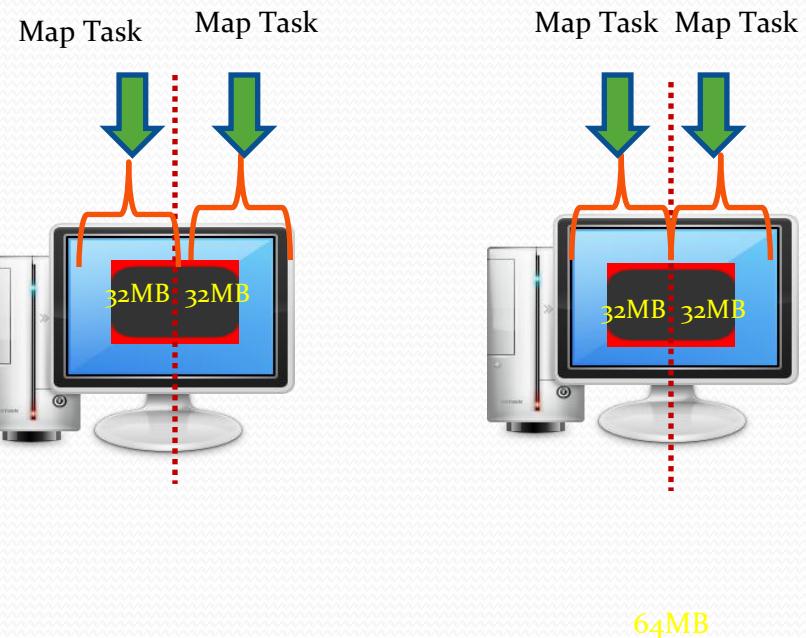
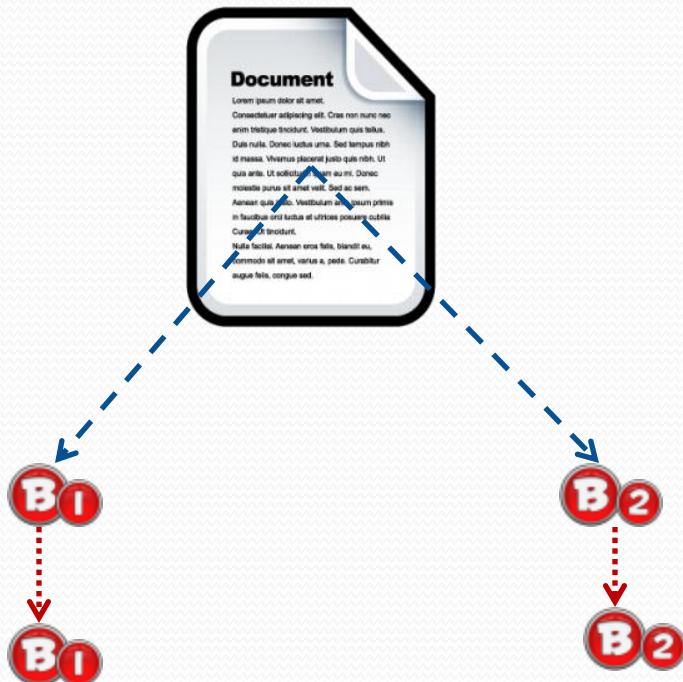


Input Split Size = 32MB

File Size = 128MB

Block Size = 64MB

Replication Factor = 1



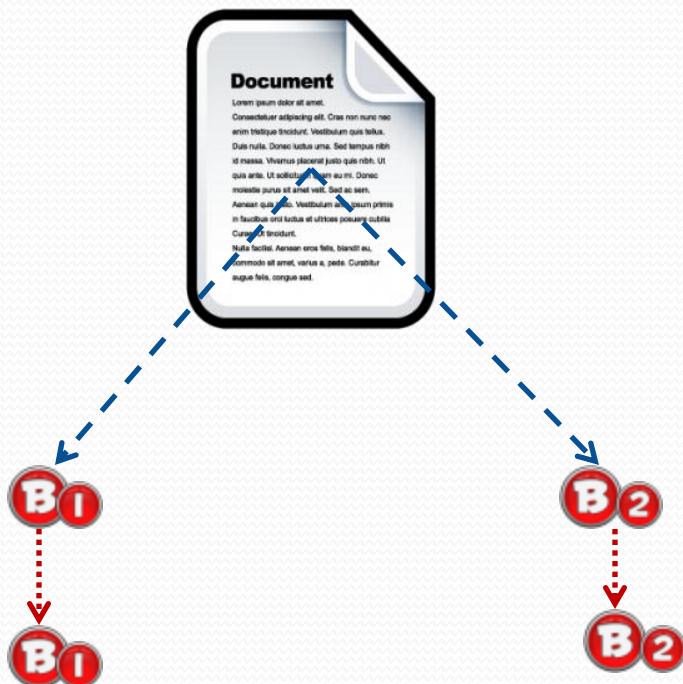
64MB

Input Split Size = 128 MB (Equal to File size)

File Size = 128MB

Block Size = 64MB

Replication Factor = 1



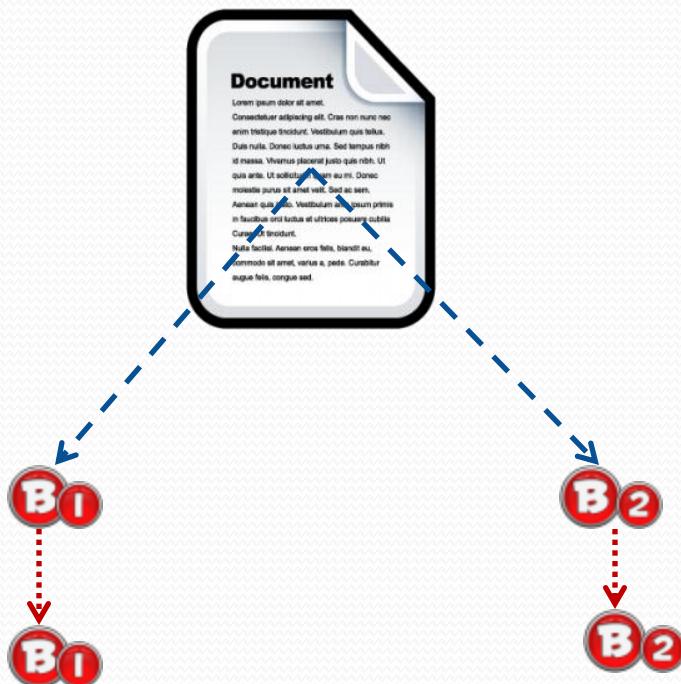
How many input splits?

Input Split Size = 128MB

File Size = 128MB

Block Size = 64MB

Replication Factor = 1



Number of input splits = 1

Hence number of map tasks = 1

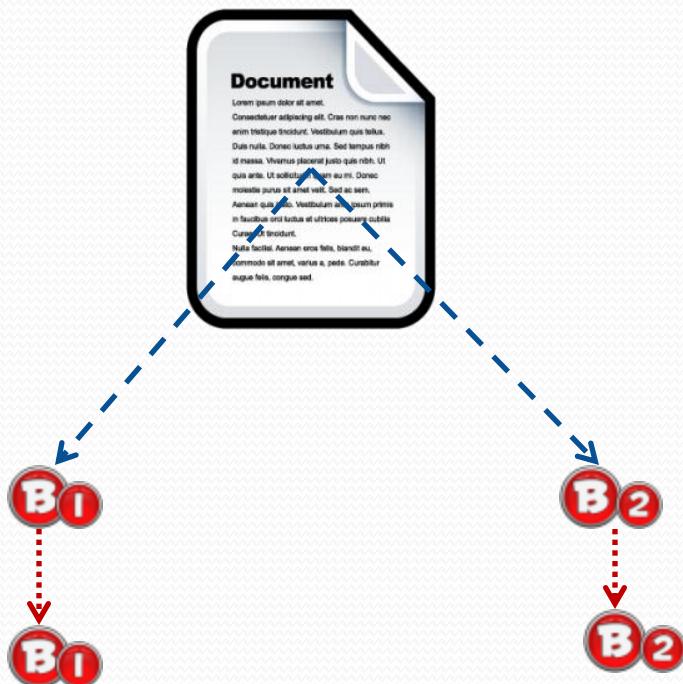
Each map task is going to process 128MB of data

Input Split Size = 128MB

File Size = 128MB

Block Size = 64MB

Replication Factor = 1



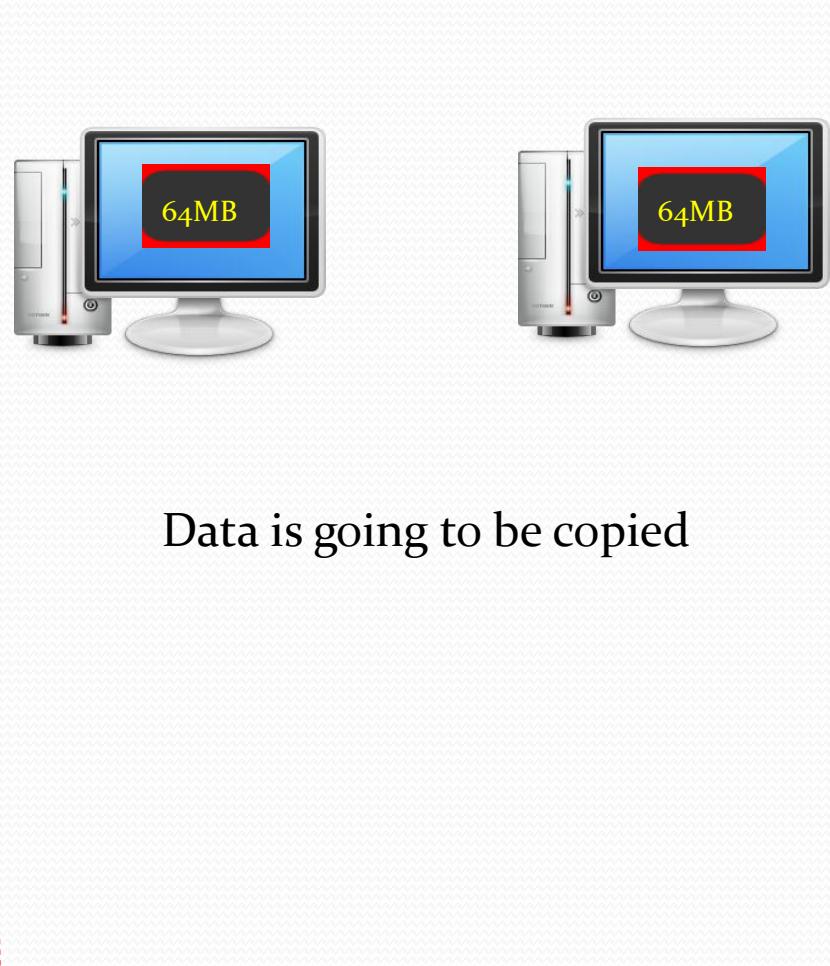
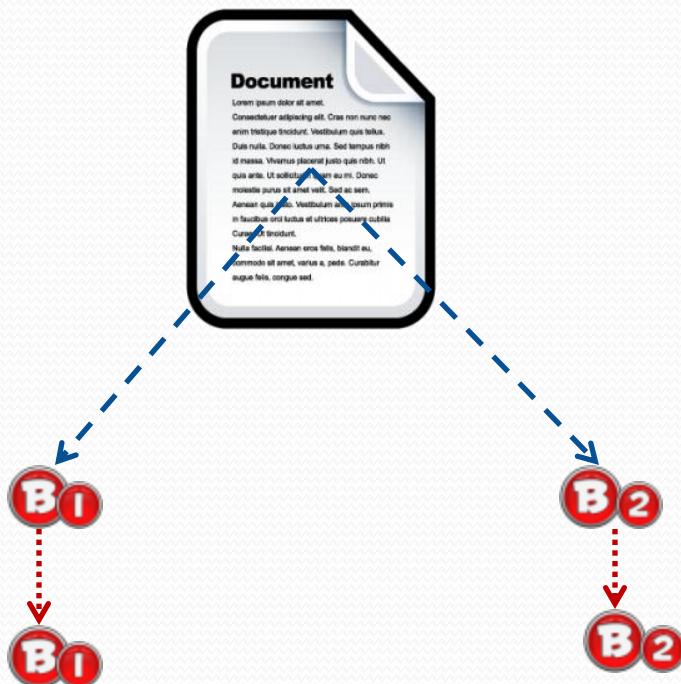
Doesn't have 128MB blocks

Input Split Size = 128MB

File Size = 128MB

Block Size = 64MB

Replication Factor = 1

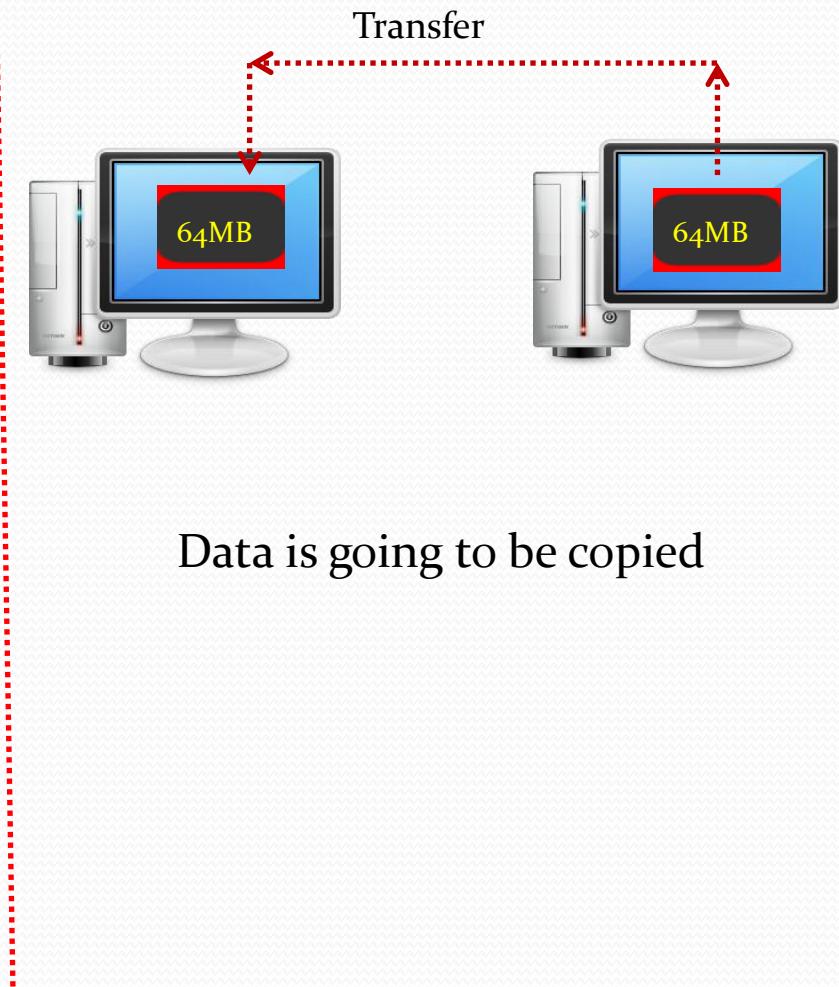
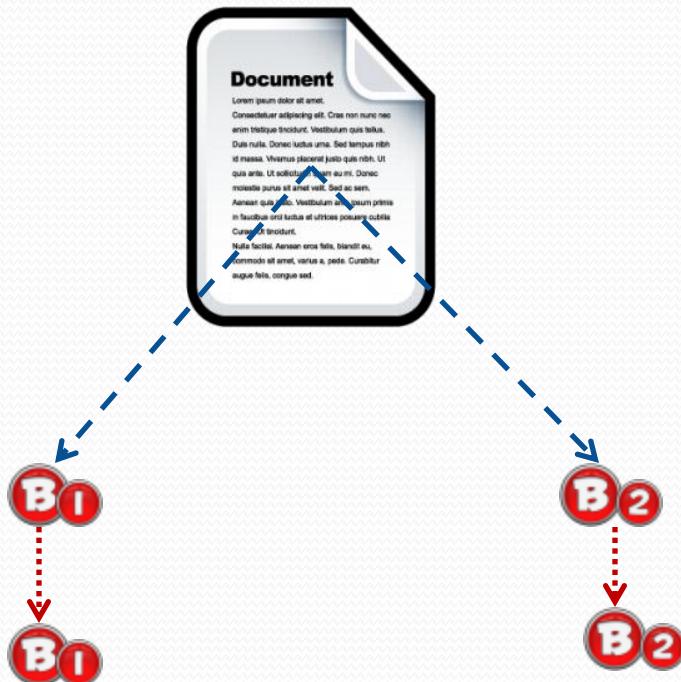


Input Split Size = 128MB

File Size = 128MB

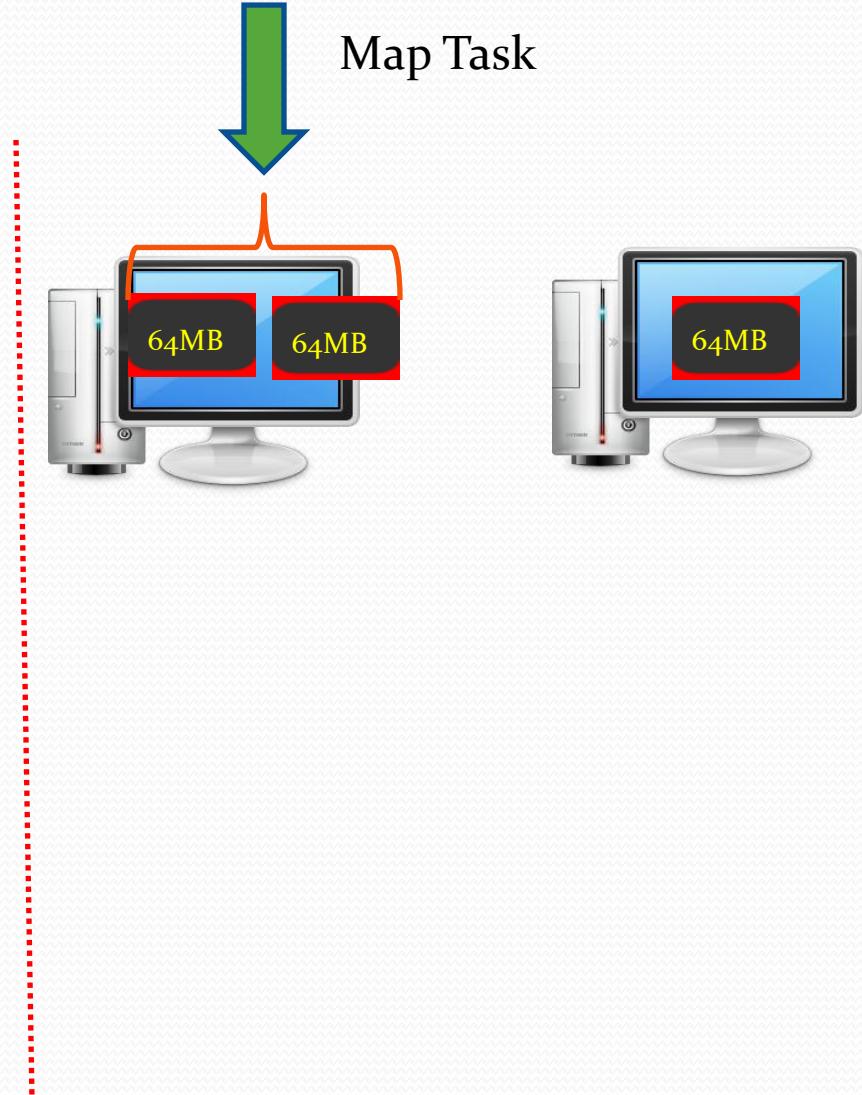
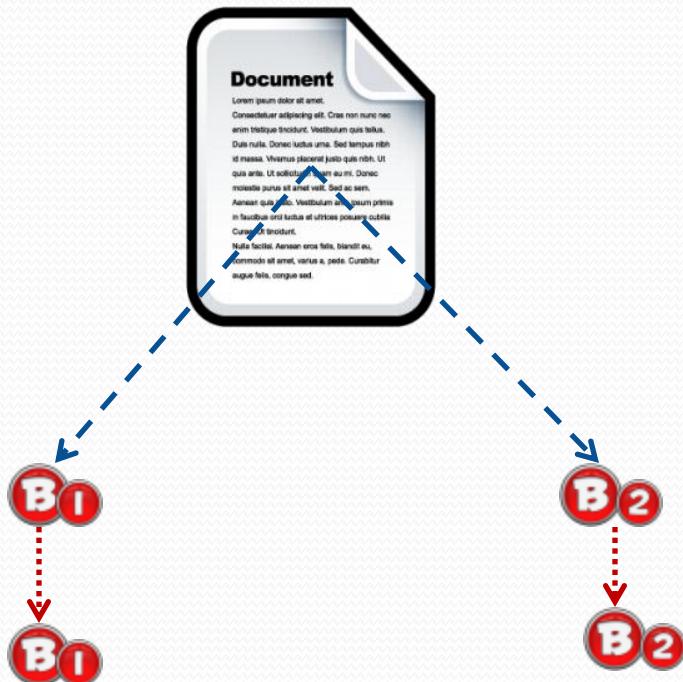
Block Size = 64MB

Replication Factor = 1



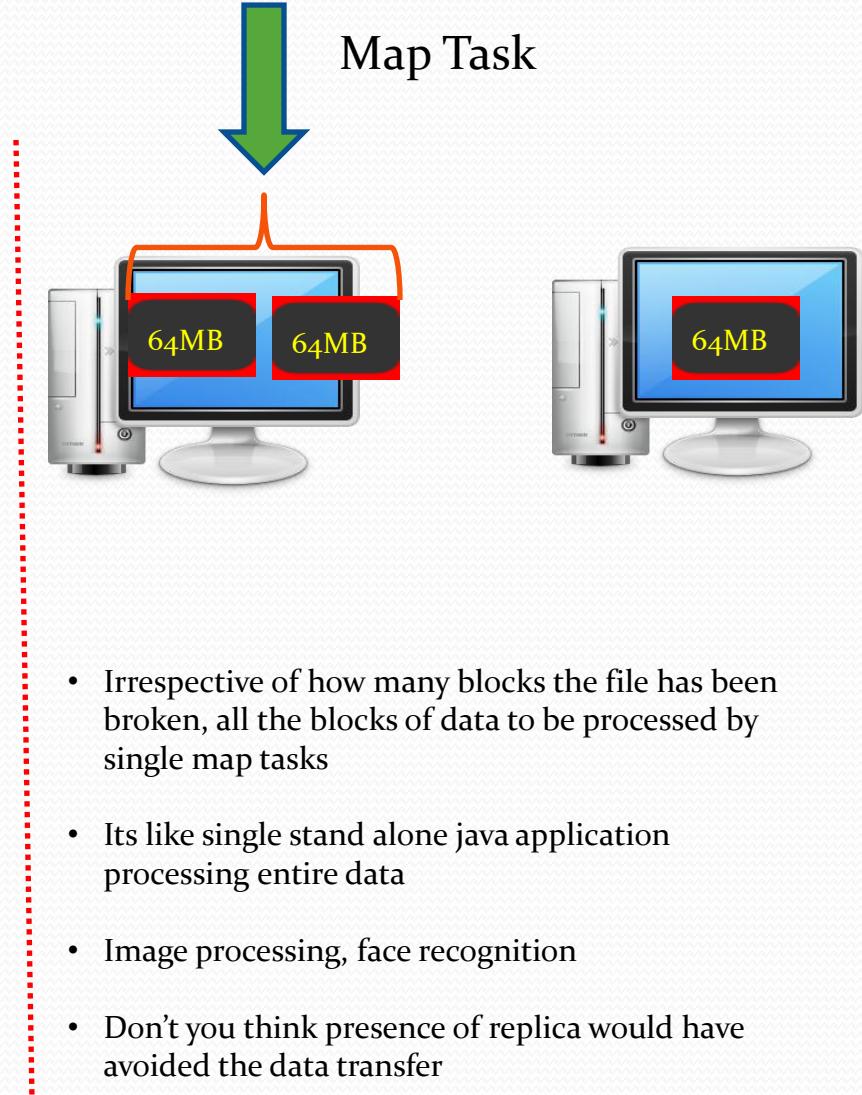
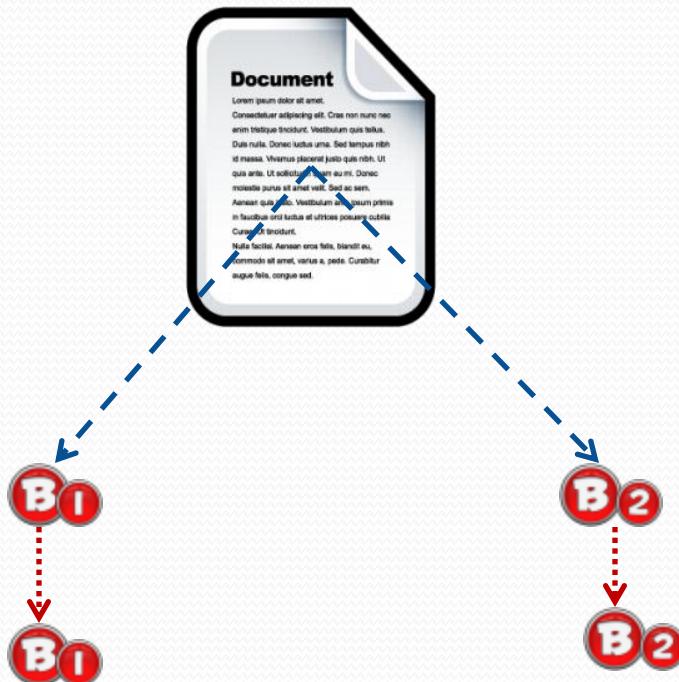
Input Split Size = 128MB

File Size = 128MB
Block Size = 64MB
Replication Factor = 1



Input Split Size = 128MB

File Size = 128MB
Block Size = 64MB
Replication Factor = 1



Input Split cont'd

- Each mapper works only on one input split
- Input split size is controlled by the following properties
 - `mapred.min.split.size` (default 1 byte)
 - `mapred.max.split.size` (default `LONG.MAX_VALUE` which is $2^{63} - 1$)
 - `dfs.block.size` (default 64 MB which is $64 * 1024 * 1024$ bytes)

Input Split size = max(minSplitSize, min(maxSplitSize, blockSize))

Input Split cont'd

$$\text{InputSplitSize} = \text{Max}(\text{minSplitSize}, \text{min}(\text{maxSplitSize}, \text{blockSize}))$$

Min Split Size	Max Split Size	block size	Input Split Size
1	LONG.MAX_VALUE	64	64
1	32	64	32
128	LONG.MAX_VALUE	64	128

Understanding MapReduce Job

Hi, how are you?
I am fine.
How are you doing?
Apple is orange.
Orange is apple.
How you are finding the
weather?

Problem Statement:

Count the number of words starting with vowels

Output:

Word	Count
are	3
I	1
Apple	2
Orange	2

Understanding MapReduce Job



Hi, how are you?
I am fine.
How are you doing?
Apple is orange.
Orange is apple.
How you are finding the
weather?

Approach

1. Take a single line at a time
2. Start reading the words from the line
3. If the word is starting with a vowel, take up the word and write 1 next to it.
 1. If the word is already existing, then just add 1 to it
4. Ignore other words and extra special characters.
5. Read the next line and go to step 2
6. Once all the lines are done, add the number of 1's to get the final count

Understanding MapReduce Job



Approach

1. Take a single line at a time

Hi, how are you?

Understanding MapReduce Job



Hi, how are you?
I am fine.
How are you doing?
Apple is orange.
Orange is apple.
How you are finding the
weather?

Approach

1. Take a single line at a time

Hi, how are you?

2. Read each word and see if it is starting with vowel.
3. Ignore words which are not starting with vowel and extra special characters
4. Only consider words which are starting with vowel
5. Output from this line would be

are 1

Understanding MapReduce Job

Hi, how are you?
I am fine.
How are you doing?
Apple is orange.
Orange is apple.
How you are finding the
weather?

Output once you are done reading all the lines

Are	1,1,1
I	1
Apple	1,1
Orange	1,1
Is	1,1

Understanding MapReduce Job

Hi, how are you?
I am fine.
How are you doing?
Apple is orange.
Orange is apple.
How you are finding the
weather?

Finally add the 1's to get the desired output

Are	1,1,1	1+1+1=3
I	1	1
Apple	1,1	1+1=2
Orange	1,1	1+1=2
Is	1,1	1+1=2

Understanding MapReduce Job

Hi, how are you?
I am fine.
How are you doing?
Apple is orange.
Orange is apple.
How you are finding the
weather?

Are	1,1,1	$1+1+1=3$
I	1	1
Apple	1,1	$1+1=2$
Orange	1,1	$1+1=2$
Is	1,1	$1+1=2$

What is Mapper?

- Mapper is the first phase of MapReduce job
- Works typically on one block of data (`dfs.block.size`)
- MapReduce framework always try to ensure that map task run closer to the data to avoid transfer of data
 - It always not possible. Why?
 - Several map tasks runs parallel on different machines and each working on different portion (block) of data
- Mapper reads key/value pairs and emits key/value pair
 - Plain key value pair.
 - Key is always associated with value.

Understanding Map Tasks

File Size = 128MB

Block Size = 64MB

Replication Factor = 3



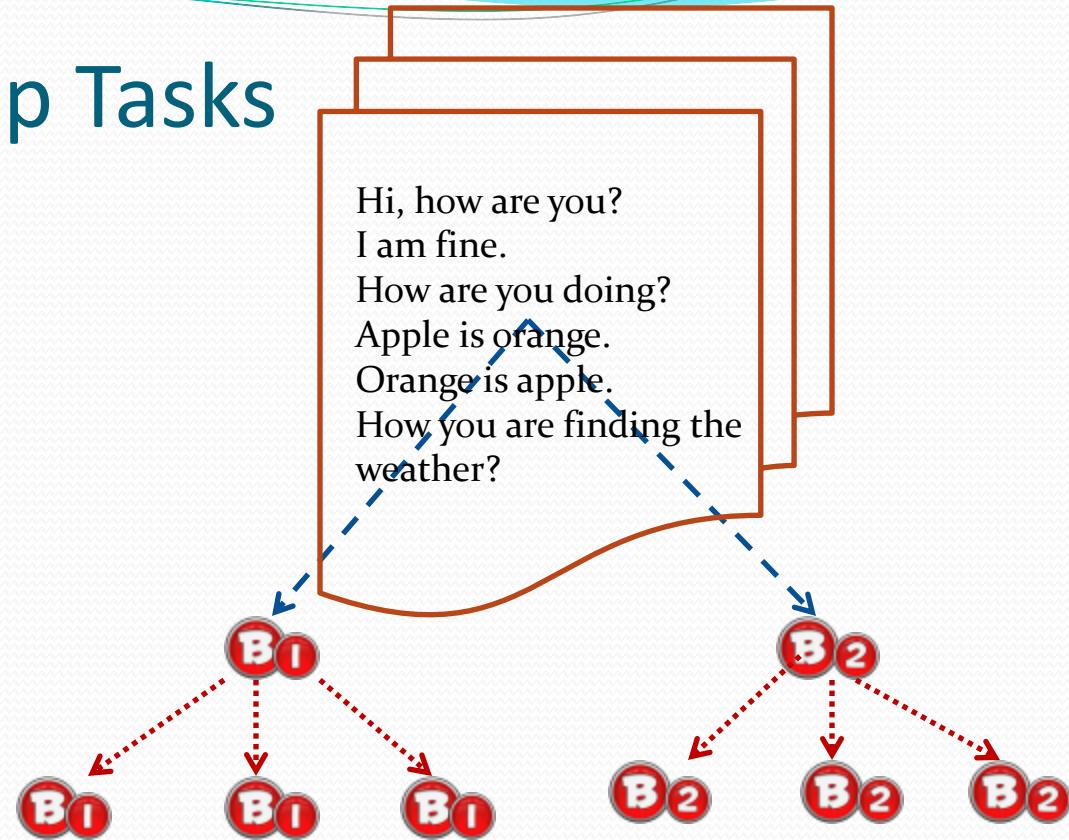
Hi, how are you?
I am fine.
How are you doing?
Apple is orange.
Orange is apple.
How you are finding the
weather?

Understanding Map Tasks

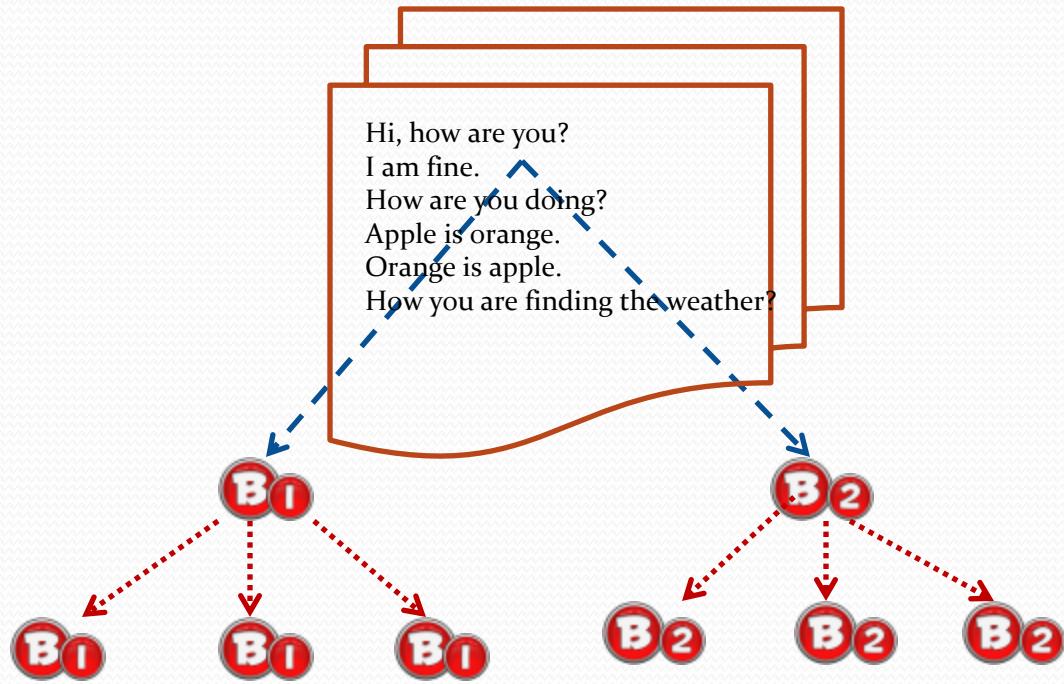
File Size = 128MB

Block Size = 64MB

Replication Factor = 3

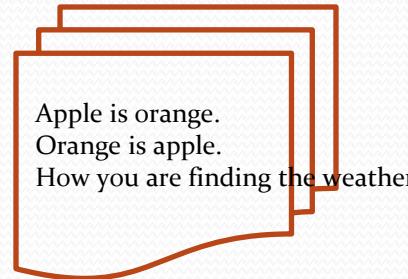


Understanding Map Tasks



Replica are distributed across several slave machines

Understanding Map Tasks



Understanding Map Tasks



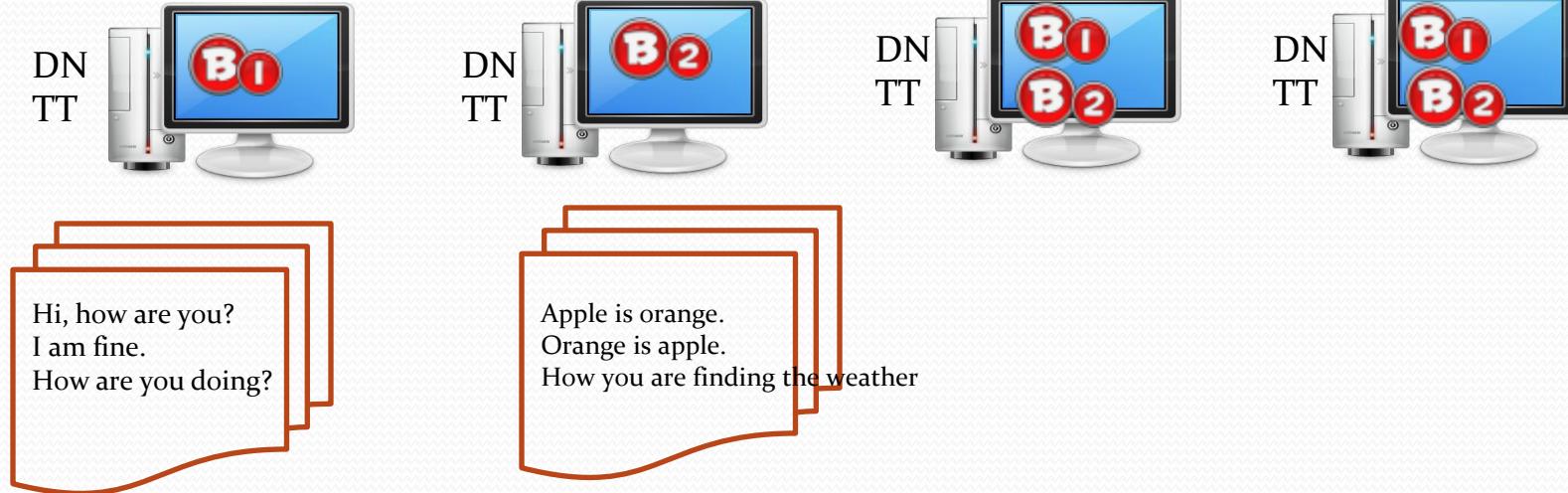
Hi, how are you?
I am fine.
How are you doing?

Apple is orange.
Orange is apple.
How you are finding the weather

How many number of blocks?



Understanding Map Tasks

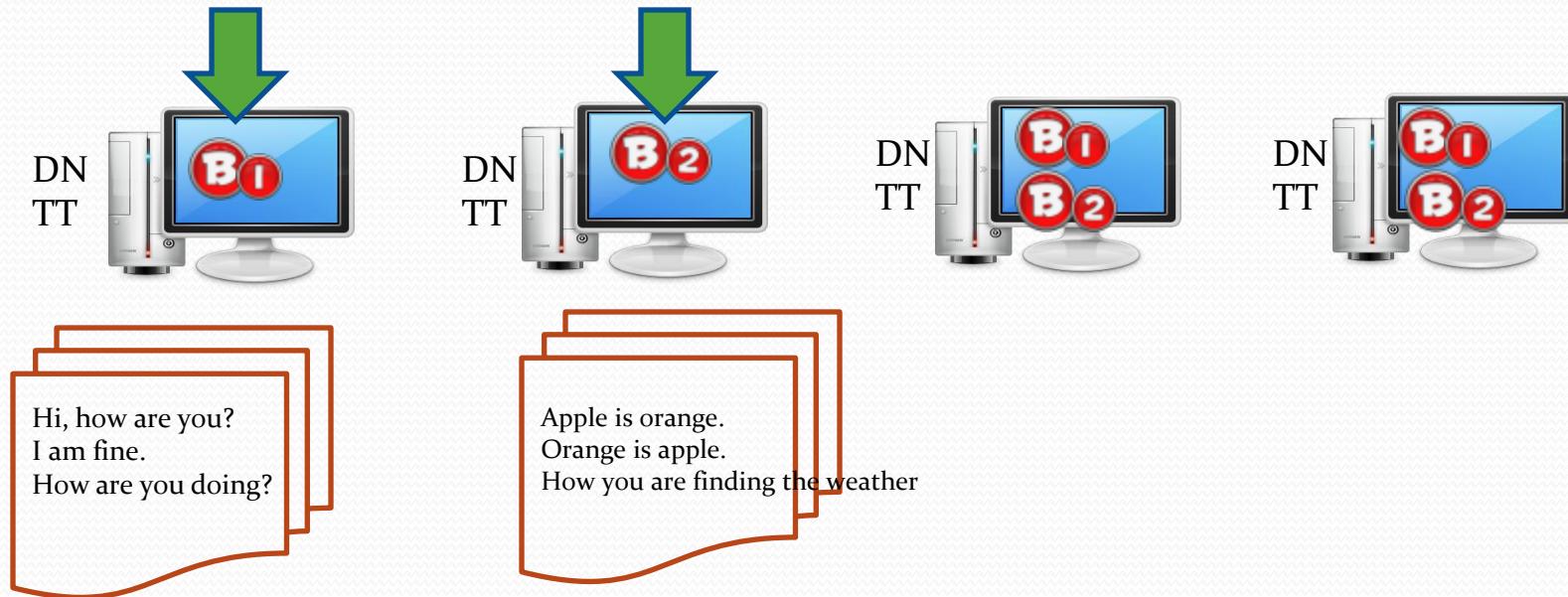


Number of blocks = 2

Number of map tasks = 2



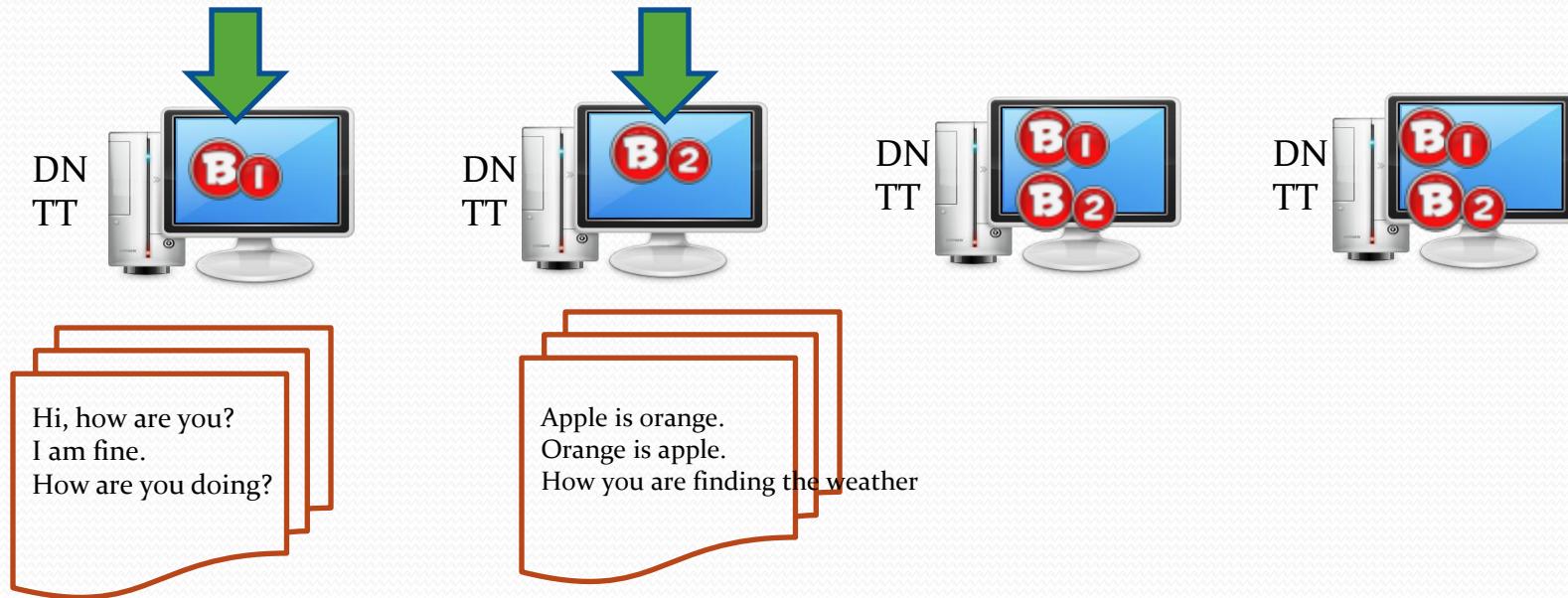
Understanding Map Tasks



Each map task is processing different blocks of data parallelly and independently



Understanding Map Tasks



Each Map Task is associated with map function which receives single record from an input split or block at a time



Understanding Map Tasks

```
map( input_key, input_value)
{
    As a developer you need to process input_key and input_value
    and generate output_key and output_value
}
```

You need to write only one map() which will be running across all the blocks of data

What is input_key and input_value?

Understanding Map Tasks

```
map( input_key, input_value)
{
    As a developer you need to process input_key and input_value
    and generate output_key and output_value
}
```

What is input_key and input_value?

A RECORD is converted into key value pair and sent to map function.

What is RECORD?

Understanding Map Tasks

```
map( input_key, input_value)
{
    As a developer you need to process input_key and input_value
    and generate output_key and output_value
}
```

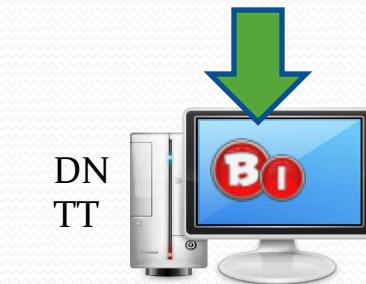
What is input_key and input_value?

A RECORD is converted into key value pair and sent to map function.

What is RECORD?

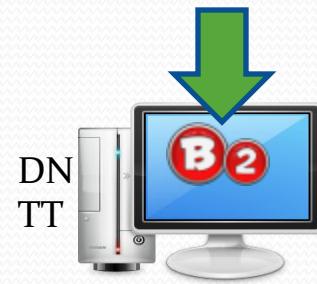
A RECORD is one complete line terminated by new line character.
Each and every RECORD is converted into key value pair and sent to map function one by one

Understanding Map Tasks



Hi, how are you?
I am fine.
How are you doing?

3 records in this input split



Apple is orange.
Orange is apple.
How you are finding the weather

3 records in this input split



How many times the map function will be invoked on both of these input splits



Understanding Map Tasks

```
map( input_key, input_value)
{
    As a developer you need to process input_key and input_value
    and generate output_key and output_value
}
```

NOTE:

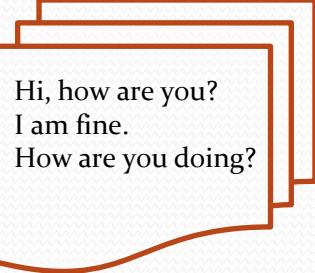
- While writing the map() keep only single record in mind.
- The logic should be written in such a way that it can handle all kinds of records
- Please notice that few records have 3 words, few records have 4 words, few records are empty etc. along with special characters like comma, question mark etc.

Understanding Map Tasks

Assumption:

A record or complete line is converted into key value pair with

KEY= line number & VALUE = record



3 records in this input split

Records converting into (key,value) pair



(0,"Hi,how are you?")
(1,"I am fine.")
(2,"How are you doing?")

Understanding Map Tasks

Hi, how are you?
I am fine.
How are you doing?

Records converting into (key,value) pair



(0,"Hi,how are you?")

(1,"I am fine.")

(2,"How are you doing?")

3 records in this input split

```
map( o, "Hi, How are you?" )  
{  
    -----Some Logic-----  
}
```

Understanding Map Tasks

Hi, how are you?
I am fine.
How are you doing?

Records converting into (key,value) pair



(0,"Hi,how are you?")
(1,"I am fine.")
(2,"How are you doing?")

3 records in this input split

```
map( 1,"I am fine."  
{  
    -----Some Logic-----  
}
```

Understanding Map Tasks

Hi, how are you?
I am fine.
How are you doing?

Records converting into (key,value) pair



(0,"Hi,how are you?")
(1,"I am fine.")
(2,"How are you doing?")

3 records in this input split

```
map( 1,"How are you doing?")
{
    -----Some Logic-----
}
```

Understanding Map Tasks

```
map( input_key = line number, input_value = complete line)
{
    Break the input_value into individual words and collect them in array

    Iterate over the array element
    {
        if( word is starting with vowel)
        {
            emit(word, 1) //output_key= word which is starting with vowel and output_value=1
        }
    }
}
```

Understanding Map Tasks

Hi, how are you?
I am fine.
How are you doing?

Records converting into (key,value) pair



(0,"Hi,how are you?")

(1,"I am fine.")

(2,"How are you doing?")

3 records in this input split

```
map( input_key = 0, input_value = "Hi,how are you?" )  
{  
    Break the input_value into individual words and collect them in  
    array  
  
    Iterate over the array element  
    {  
        if( word is starting with vowel)  
        {  
            emit(word, 1) //output_key= word which is starting  
            with vowel and output_value=1  
        }  
    }  
}
```

Output key value pair for this line



(are,1)

Understanding Map Tasks

Hi, how are you?
I am fine.
How are you doing?

Records converting into (key,value) pair



(0,"Hi,how are you?")

(1,"**I am fine.**")

(2,"How are you doing?")

3 records in this input split

```
map( input_key = 1, input_value = "I am fine"
{
    Break the input_value into individual words and collect them in
    array

    Iterate over the array element
    {
        if( word is starting with vowel)
        {
            emit(word, 1) //output_key= word which is starting
            with vowel and output_value=1
        }
    }
}
```

Output key value pair for this line



(I,1)
(am,1)

Understanding Map Tasks

Hi, how are you?
I am fine.
How are you doing?

Records converting into (key,value) pair



(0,"Hi,how are you?")
(1,"I am fine.")
(2,"How are you doing?")

3 records in this input split

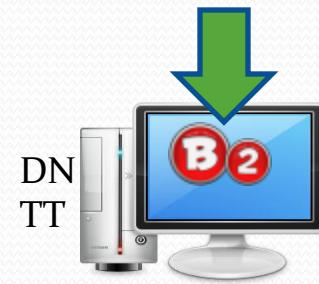
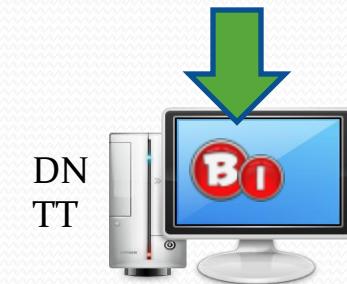
```
map( input_key = 1, input_value = "How are you doing?" )  
{  
    Break the input_value into individual words and collect them in  
    array  
  
    Iterate over the array element  
    {  
        if( word is starting with vowel)  
        {  
            emit(word, 1) //output_key= word which is starting  
            with vowel and output_value=1  
        }  
    }  
}
```

Output key value pair for this line



(are,1)

Understanding Map Tasks



Hi, how are you?
I am fine.
How are you doing?

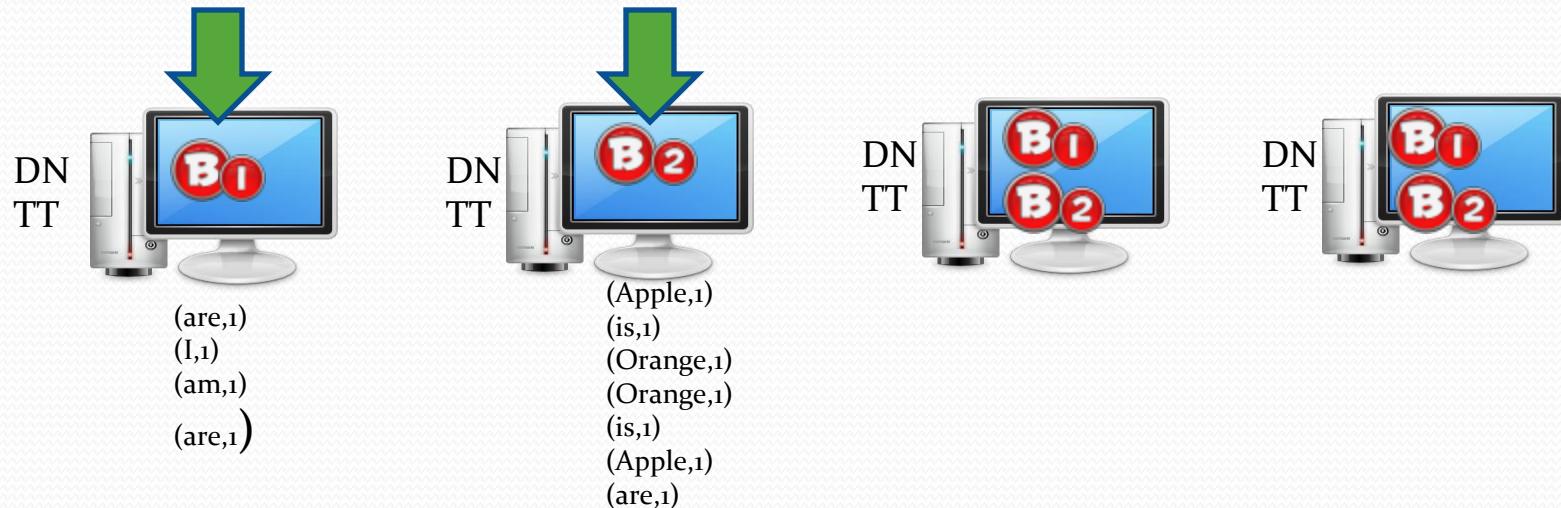
(are,1)
(I,1)
(am,1)
(are,1)

Apple is orange.
Orange is apple.
How you are finding the weather

(Apple,1)
(is,1)
(Orange,1)
(Orange,1)
(is,1)
(Apple,1)
(are,1)



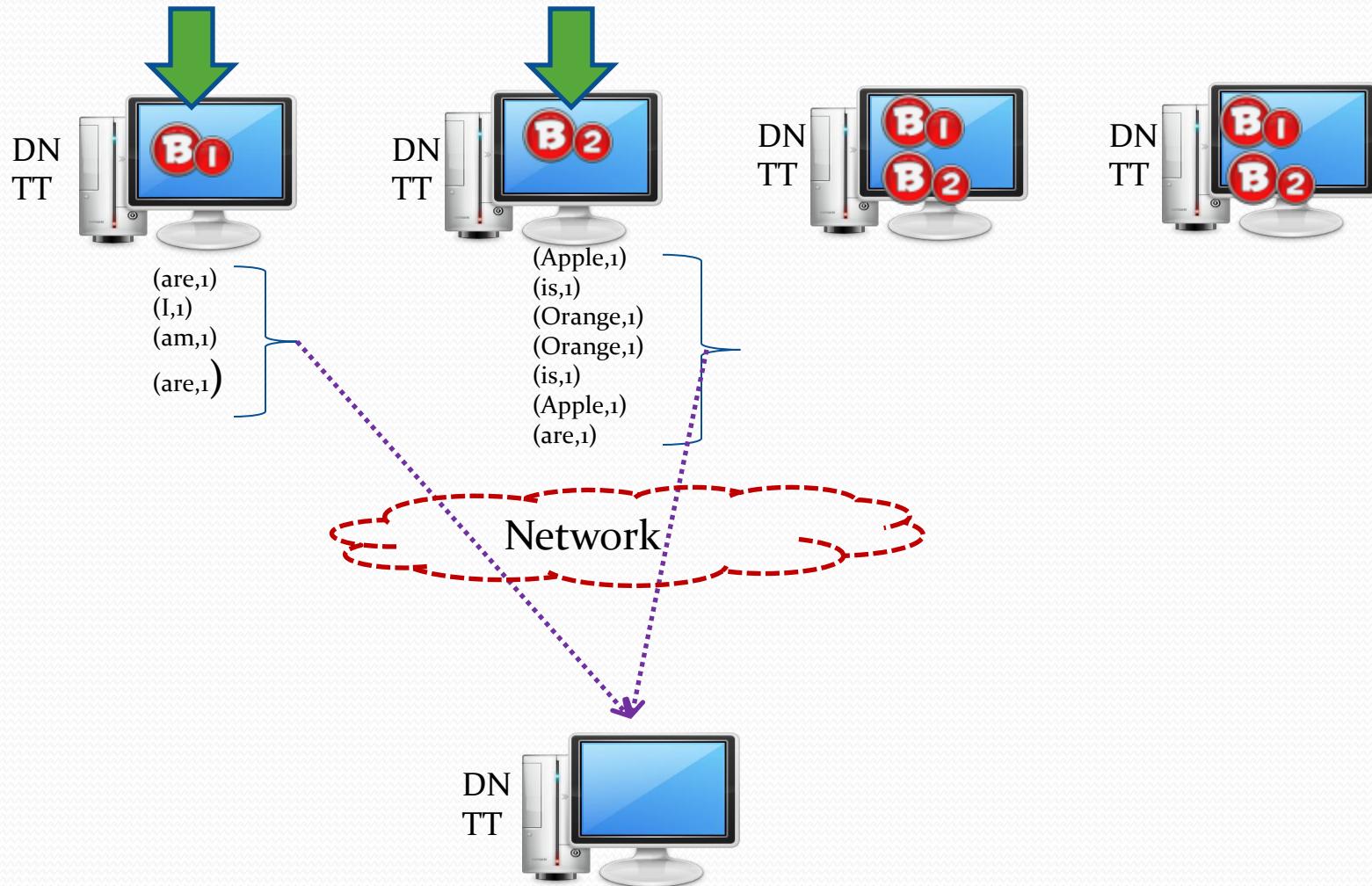
Understanding Map Tasks



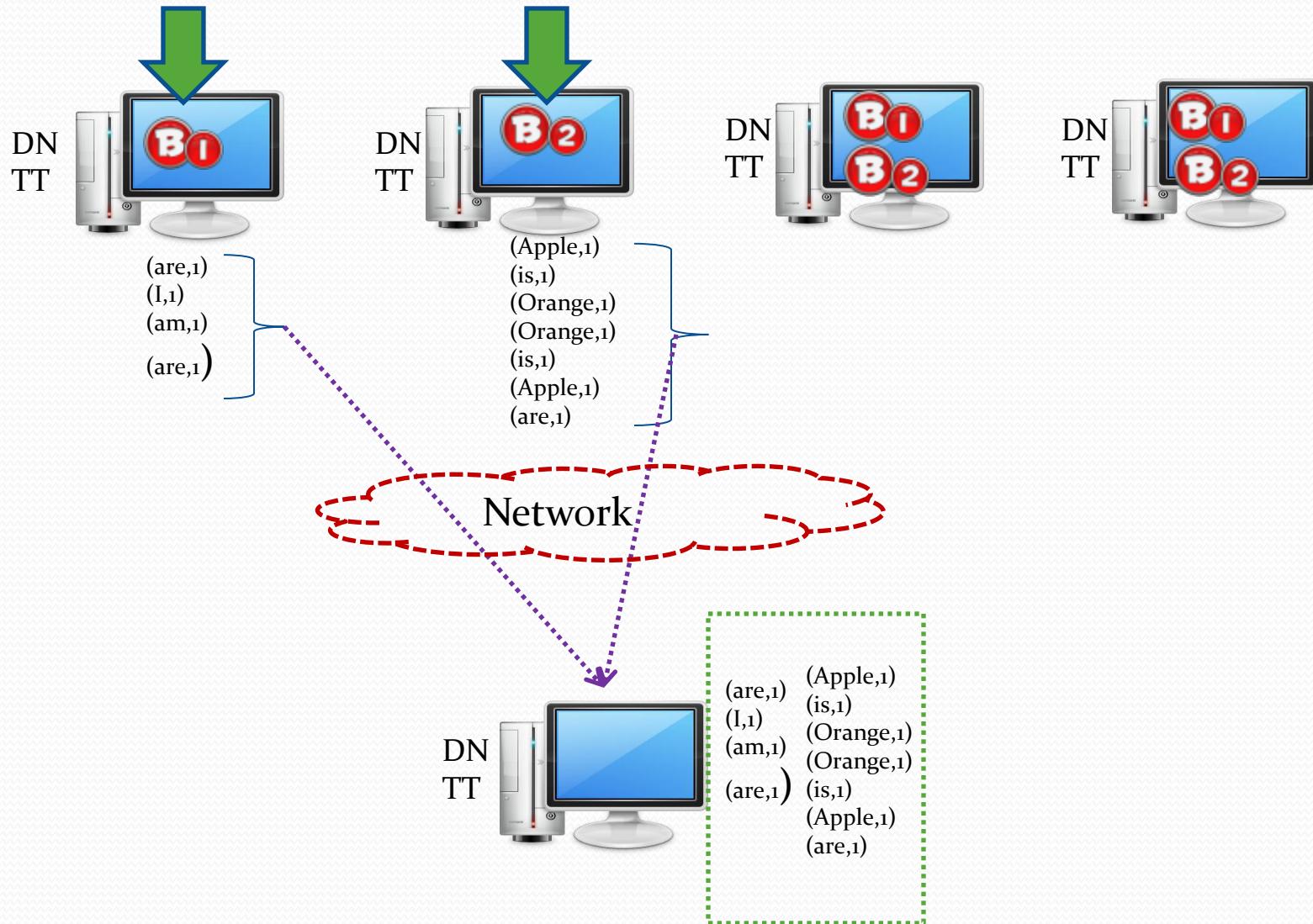
After the map task is over, the output of the map task is transferred to the machine where reducer task is running



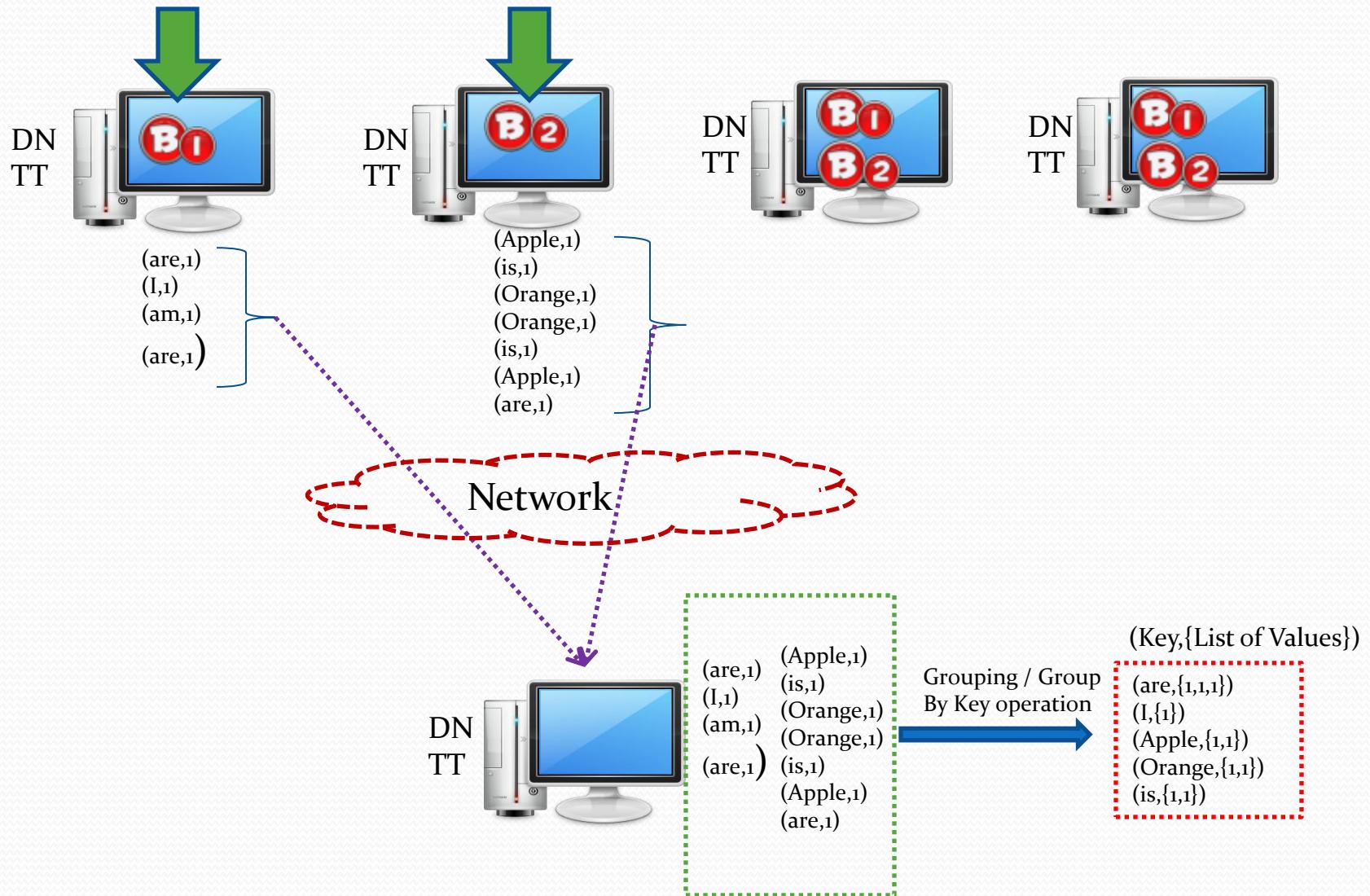
Understanding Reduce Tasks



Understanding Reduce Tasks



Understanding Reduce Tasks



Understanding Reduce Tasks

- As Map Tasks are associated with map function, Reduce task are associated with reduce function
- Output of the grouping or group by key operation is the input to reduce function

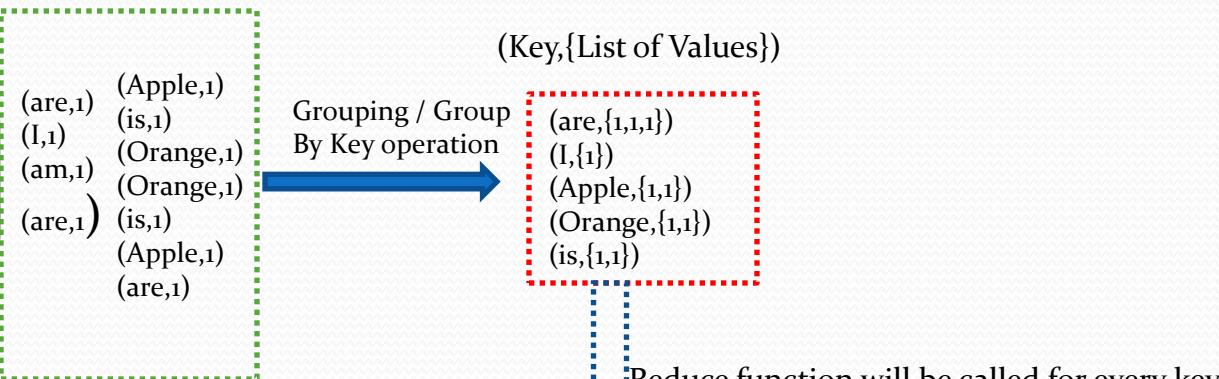
```
reduce(input_key, list_of_values)
{
```

Developer needs to write the logic to process input_key and the list of values and generate final output key and output value

```
}
```

Understanding Reduce Tasks

DN
TT



```
reduce(input_key, list_of_values)
{
```

Developer needs to write the logic to process input_key and the list of values and generate final output key and output value

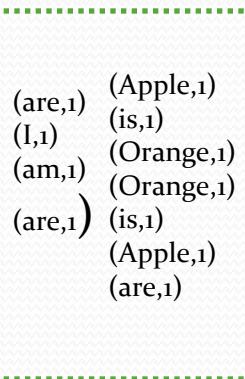
```
}
```

Understanding Reduce Tasks

```
reduce(input_key, list_of_values)
{
    sum = 0;
    Iterate over the list of values
    {
        sum = sum + val;
    }
    emit(input_key,sum); //output_key = input_key
}
```

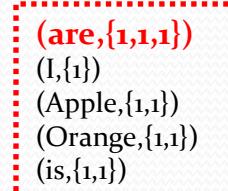
Understanding Reduce Tasks

DN
TT



Grouping / Group By Key operation

(Key,{List of Values})

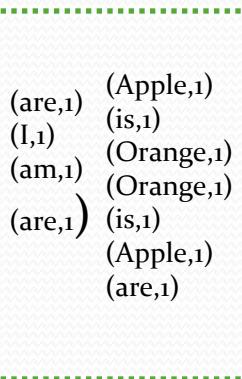


```
reduce(are, {1,1,1})  
{  
  
sum = 0;  
Iterate over the list of values  
{  
    sum = sum + val;  
}  
emit(input_key,sum); //output_key = input_key  
}
```

(are,3)

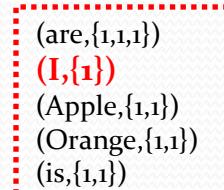
Understanding Reduce Tasks

DN
TT



Grouping / Group By Key operation

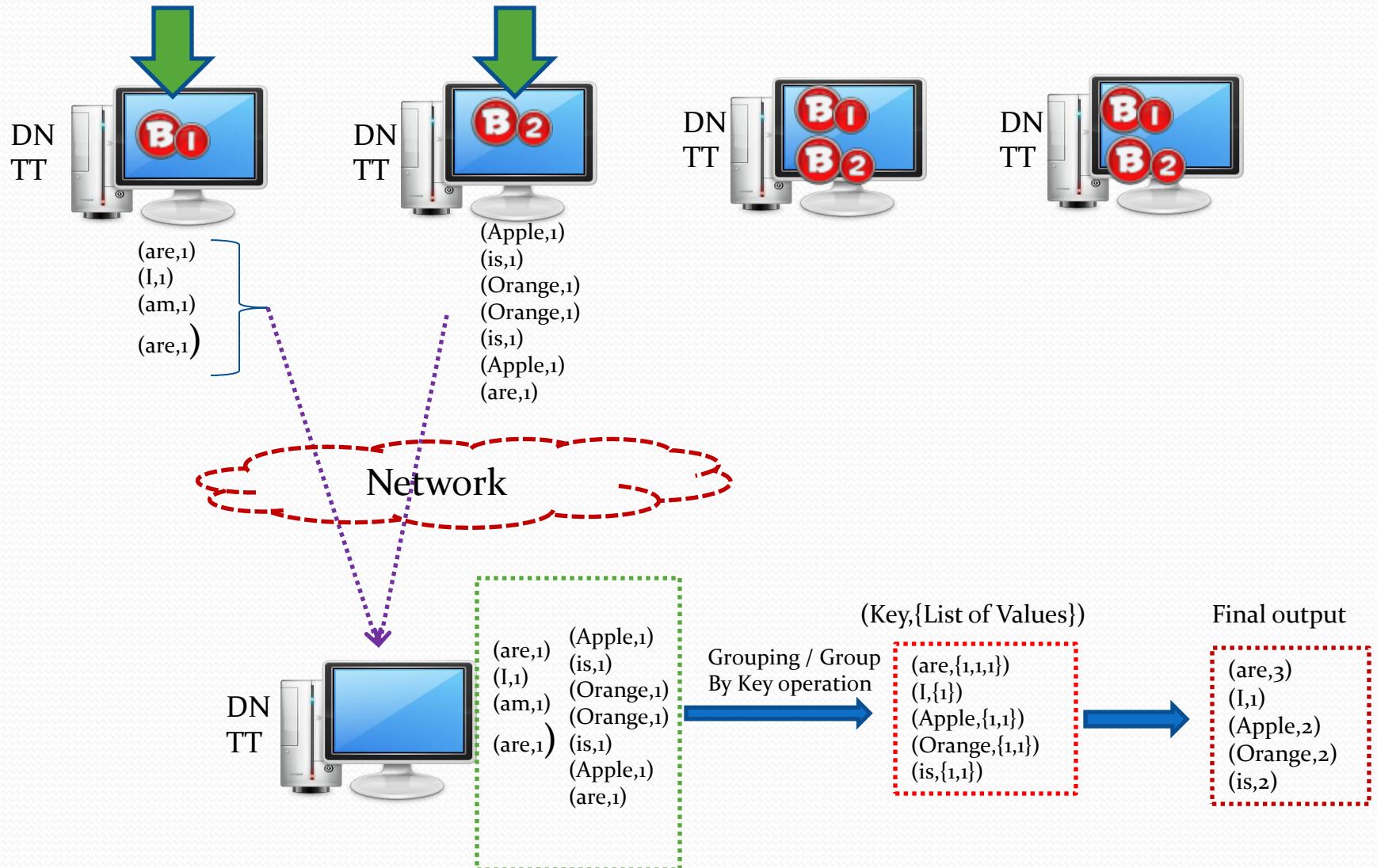
(Key,{List of Values})



```
reduce(I, {1})  
{  
  
    sum = 0;  
    Iterate over the list of values  
    {  
        sum = sum + val;  
    }  
    emit(input_key,sum); //output_key = input_key  
}
```

(I,1)

Understanding Reduce Tasks



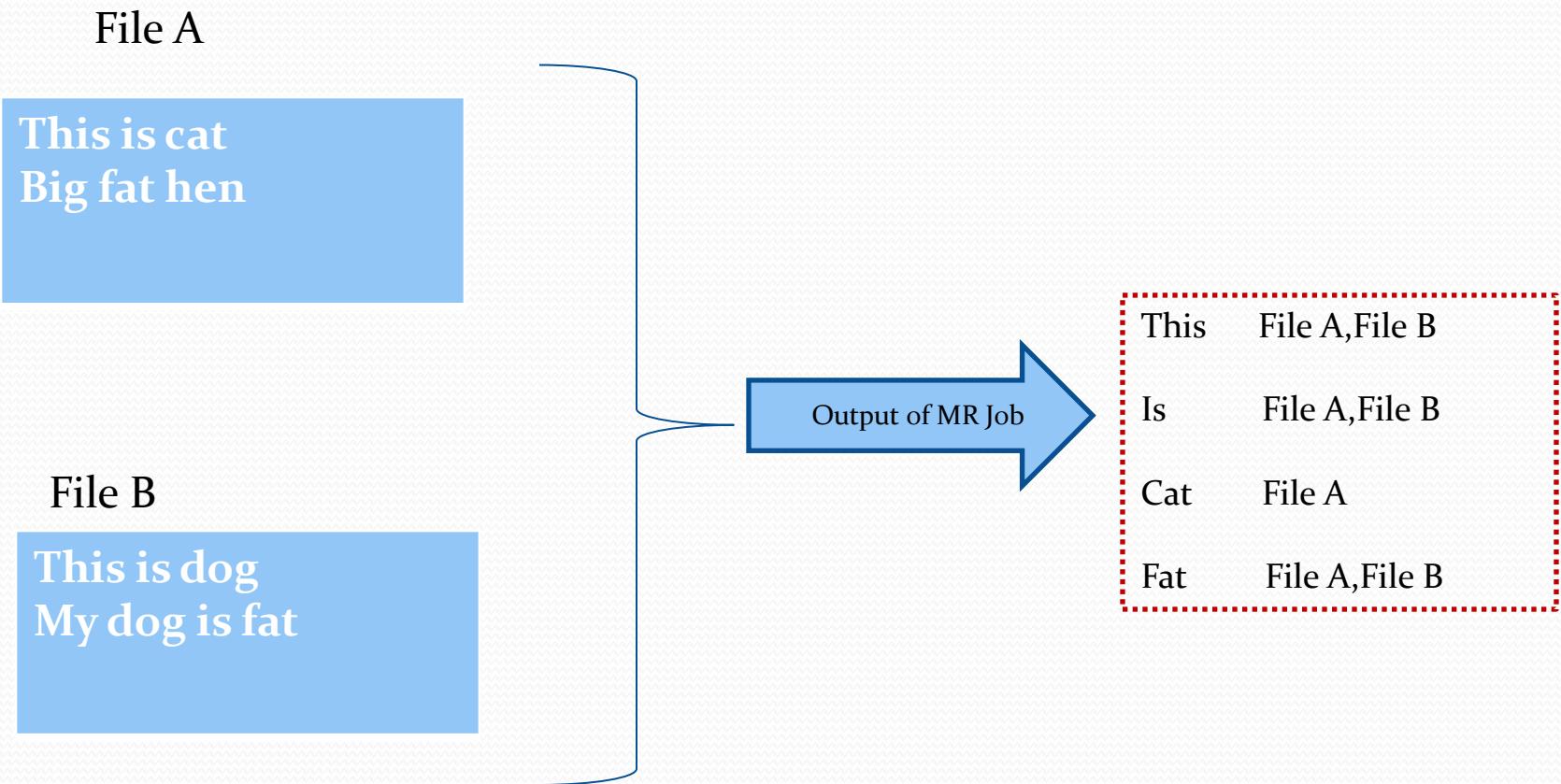
How to solve MapReduce Job?

- Two Step process
 - Always first decide the Map output key-value pair
 - Since the map function is going to be called for every record
 - Draw a quick diagram and confirm whether the grouping operation is going to give you appropriate result or not
 - Since the output of the grouping operation is going to decide what logic needs to go inside the reduce function

Inverted Index Problem

- Used for faster look up
 - Example: Indexing done for keywords in a book
- Problem statement:
 - From a list of files or documents map the words to the list of files in which it has appeared
- Output
 - *word* = List of documents in which this *word* has appeared

Indexing Problem



Indexing Problem

File A

This is cat
Big fat hen

File B

This is dog
My dog is fat



Need to run MR job on both these data sets

Decide Map output key value pair

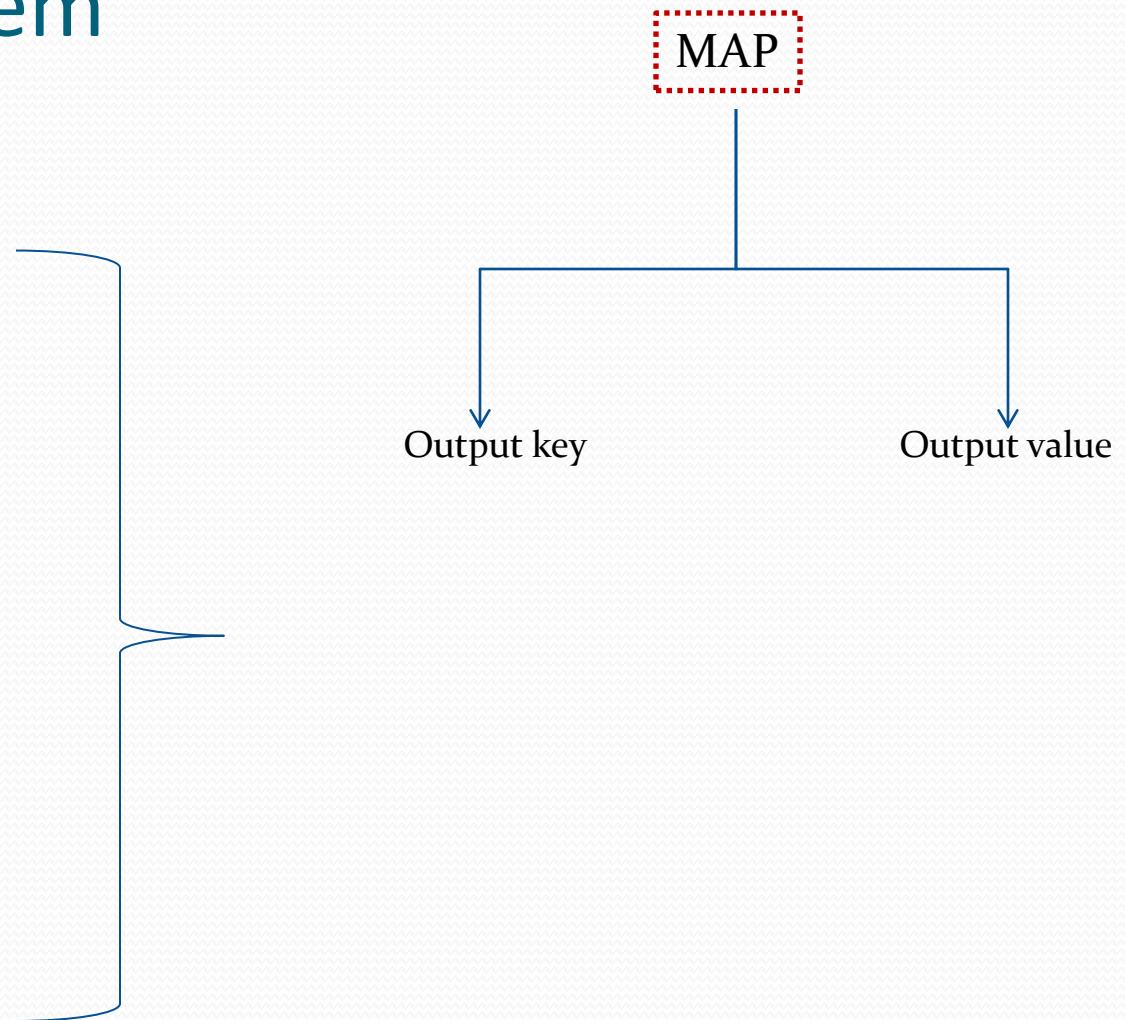
Indexing Problem

File A

This is cat
Big fat hen

File B

This is dog
My dog is fat



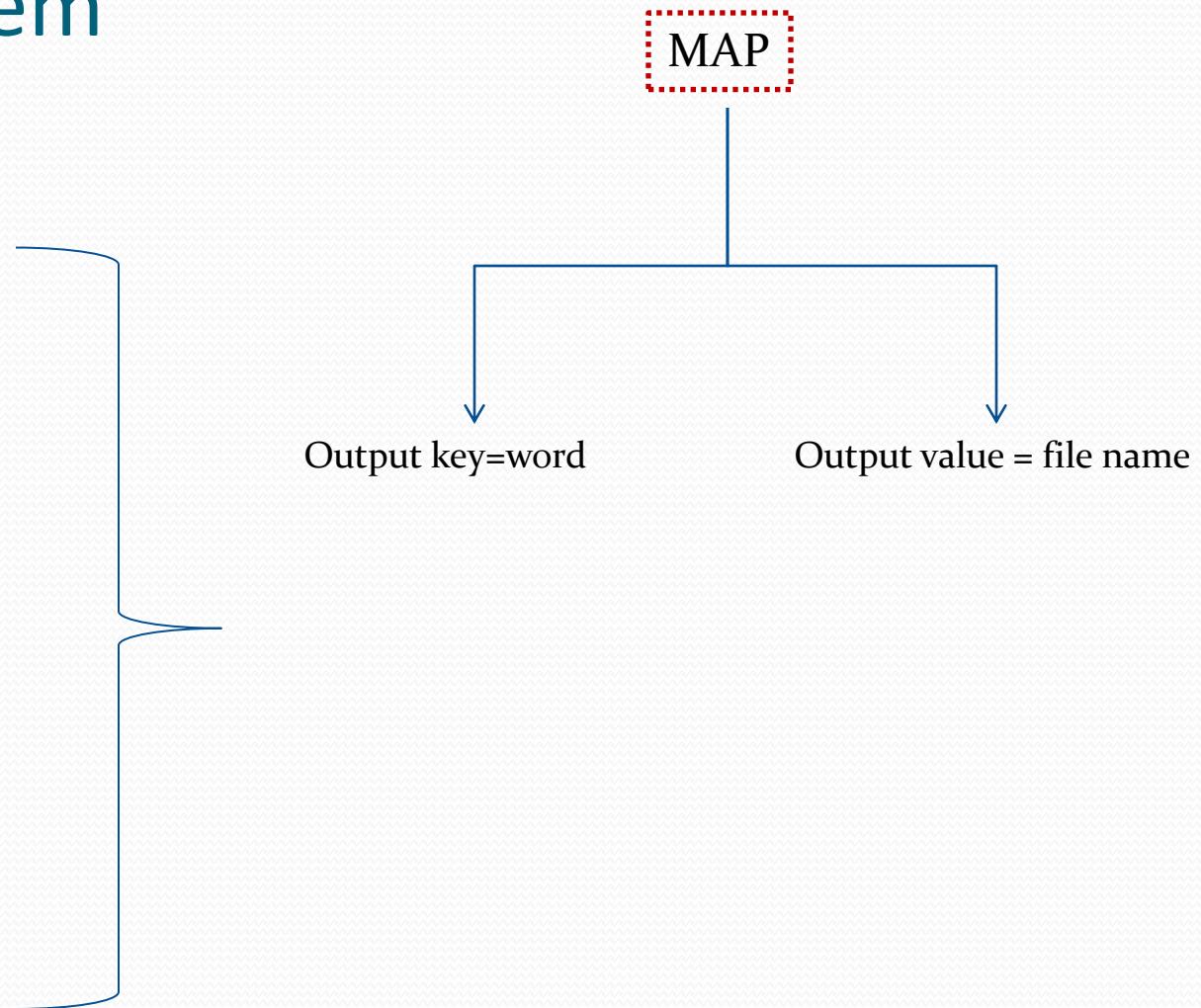
Indexing Problem

File A

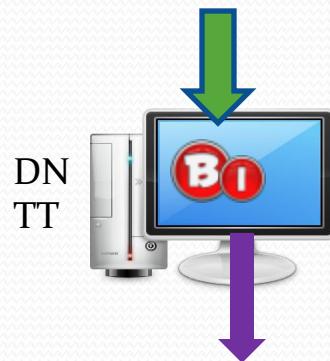
This is cat
Big fat hen

File B

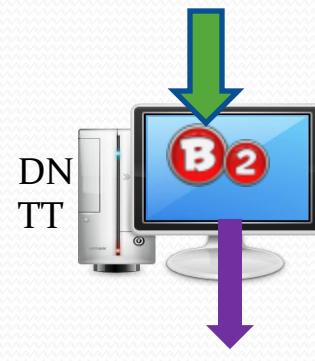
This is dog
My dog is fat



Inverted Indexing



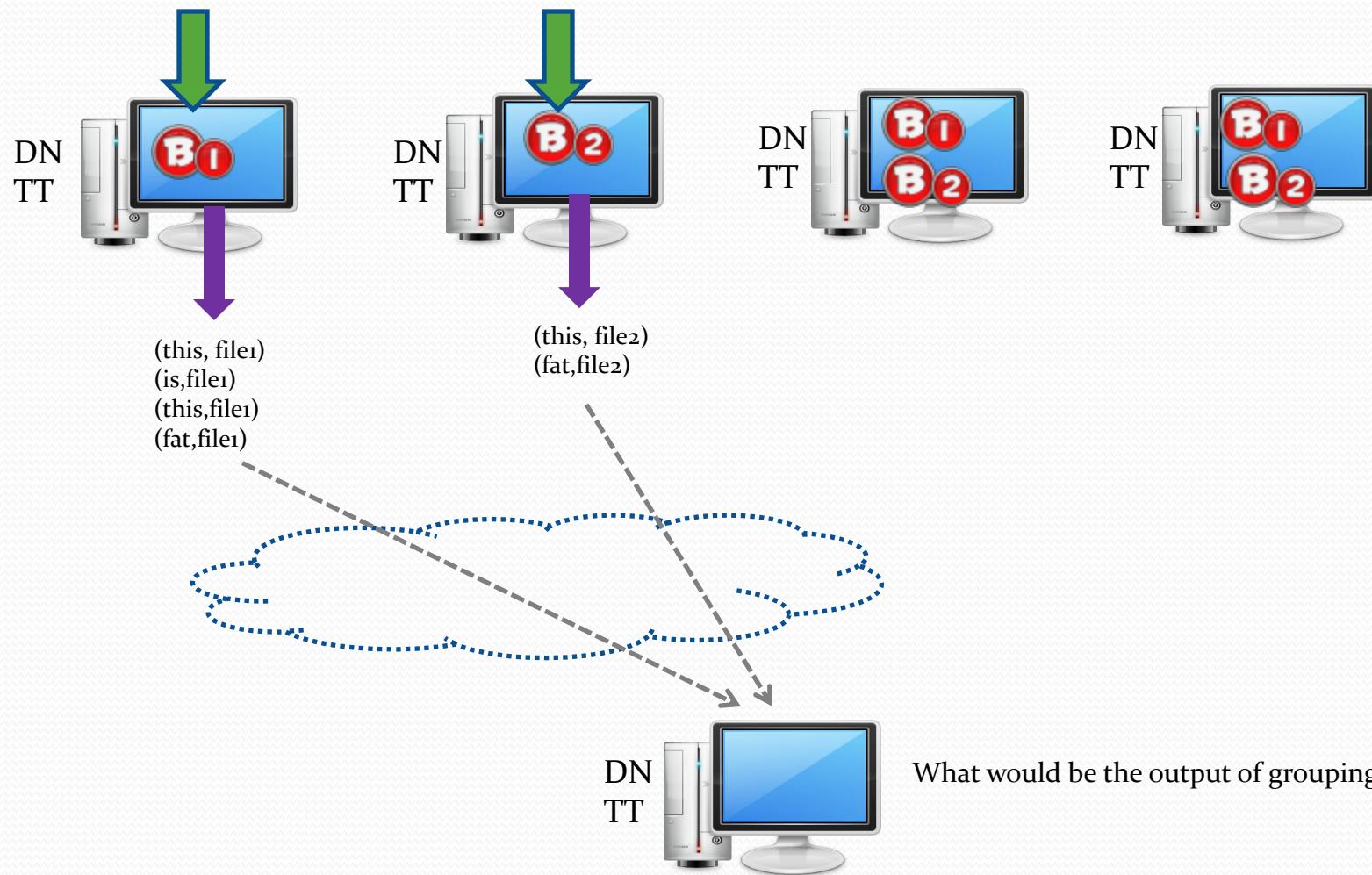
(this, file1)
(is,file1)
(this,file1)
(fat,file1)



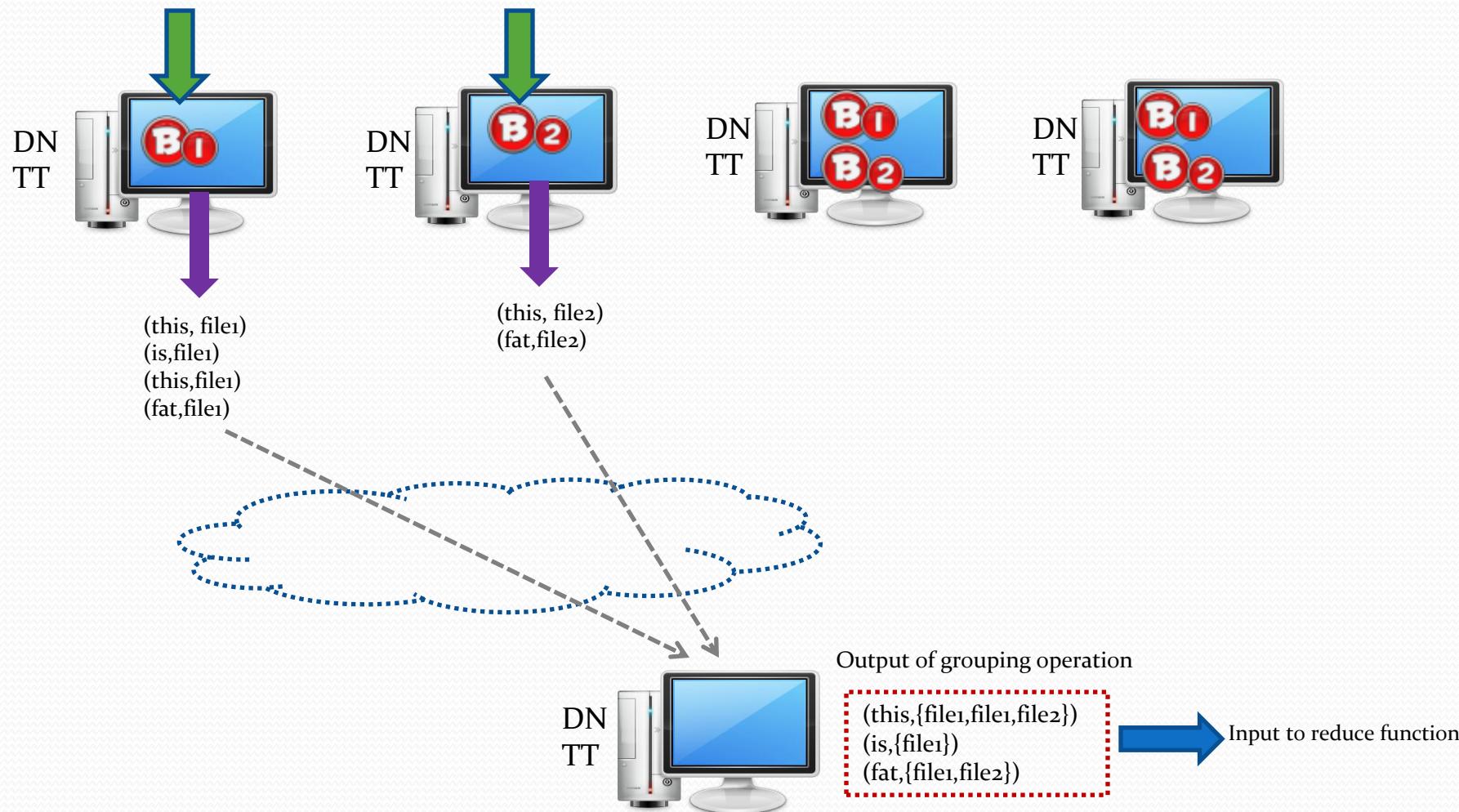
(this, file2)
(fat,file2)



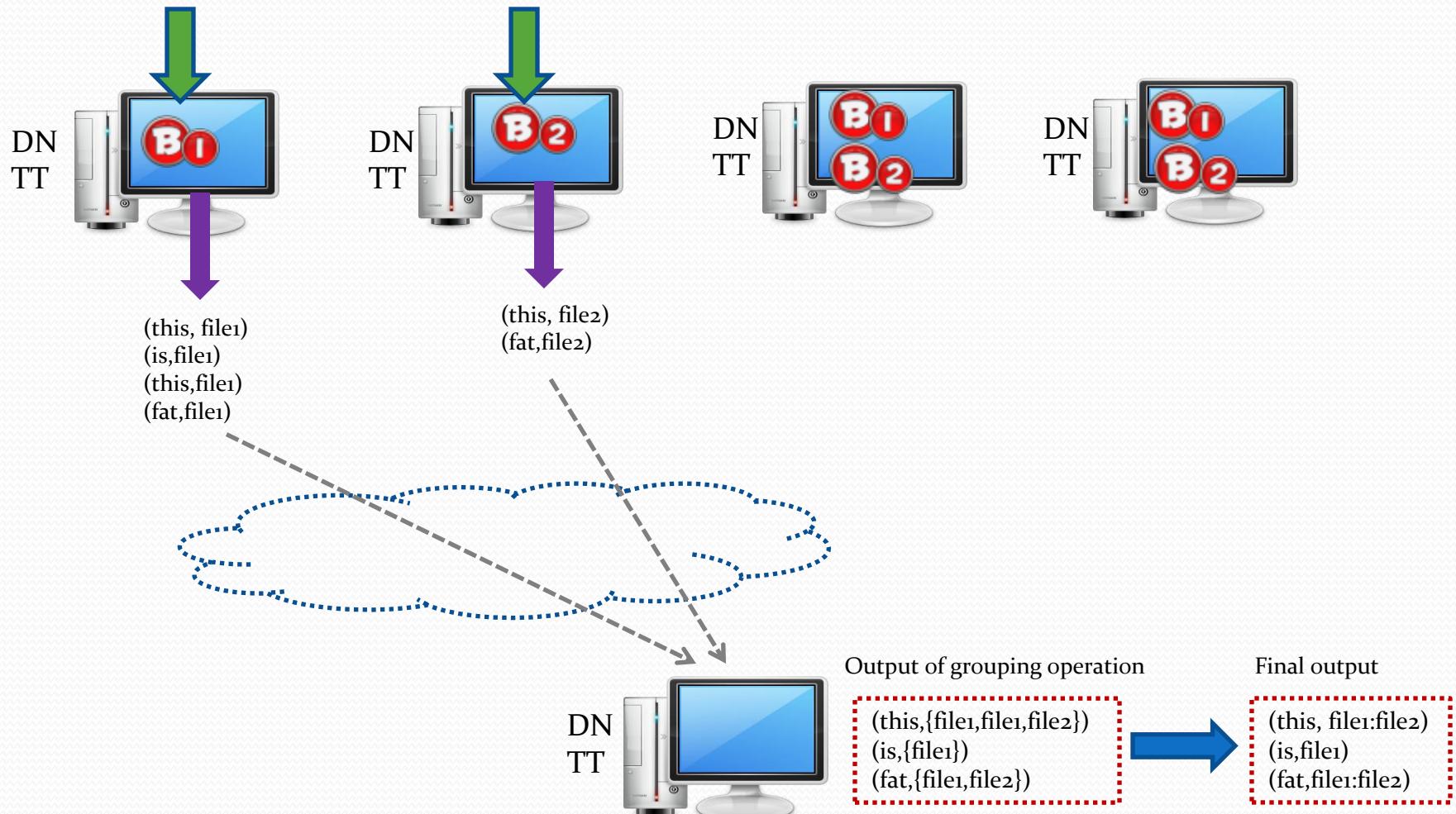
Inverted Indexing



Inverted Indexing



Inverted Indexing



Indexing problem cont'd

- Mapper

For each word in the line, $\text{emit}(\text{word}, \text{file_name})$

- Reducer

- Remember for word , all the file names list will be coming to the reducer
- $\text{Emit}(\text{word}, \text{file_name_list})$

Average Word Length Problem

- Consider the record in a file

Hey How is Henry these days?

- Problem Statement
 - Calculate the average word length for each character

$$\text{Average word length of character ch} = \frac{\text{Sum of the lengths of words starting with ch}}{\text{Total number of words starting with ch}}$$

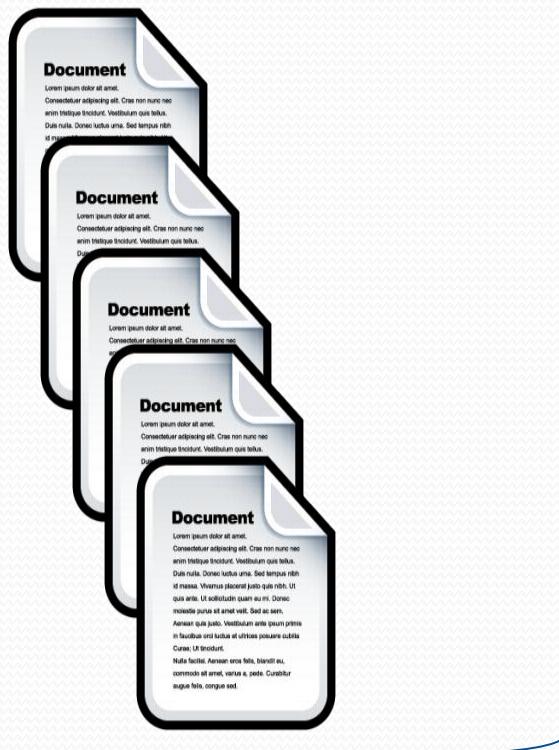
$$\text{Average word length of character 'H'} = \frac{(3+3+5)}{3} = 3.66$$

Average Word Length Problem

Hey How is Henry these days?

Character	Average word length
H	$(3+3+5)/3 = 3.66$
I	$2/1 = 2$
T	$5/1 = 5$
D	$4/1 = 4$

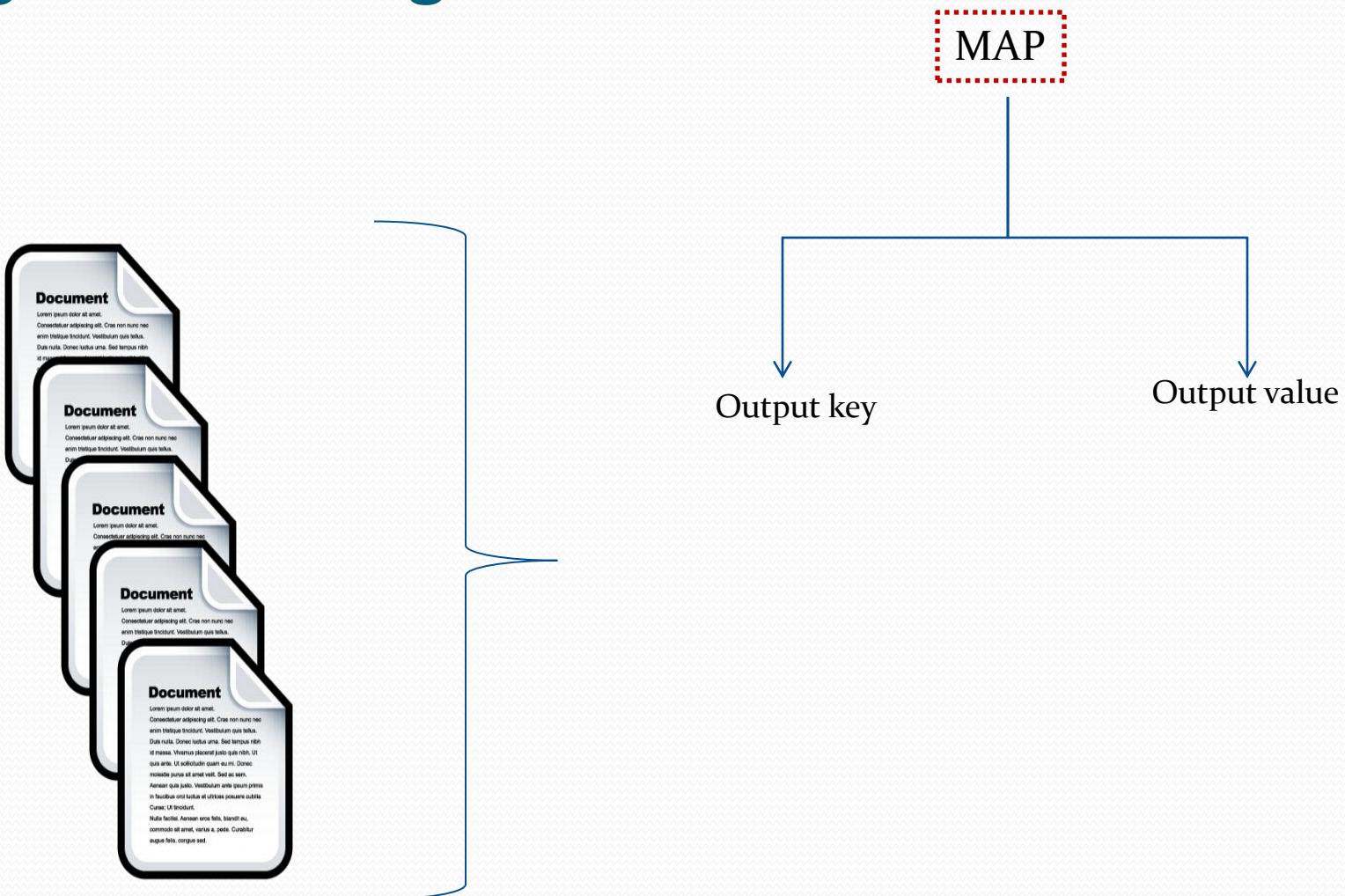
Average Word Length Problem



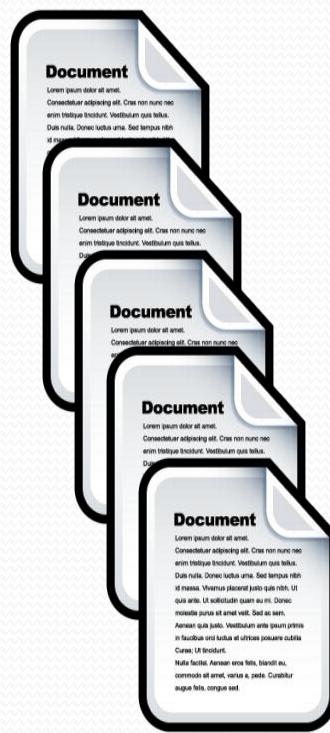
- Need to run MR job on both these data sets

Decide Map output key value pair

Average Word Length Problem



Average Word Length Problem

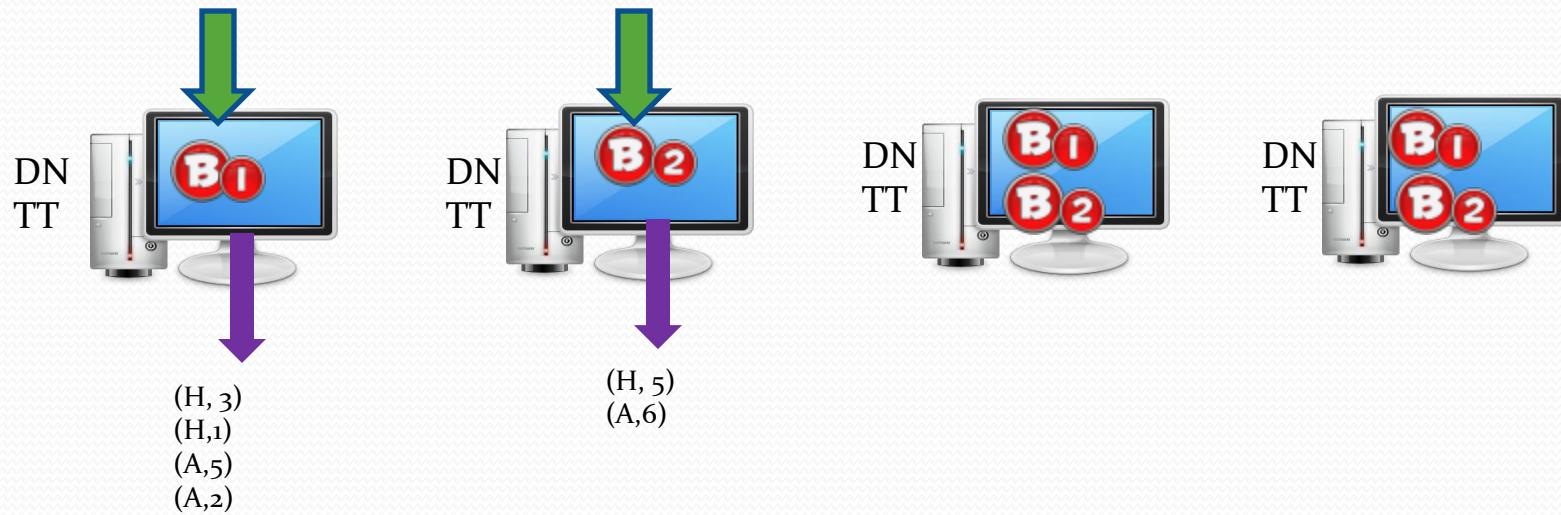


MAP

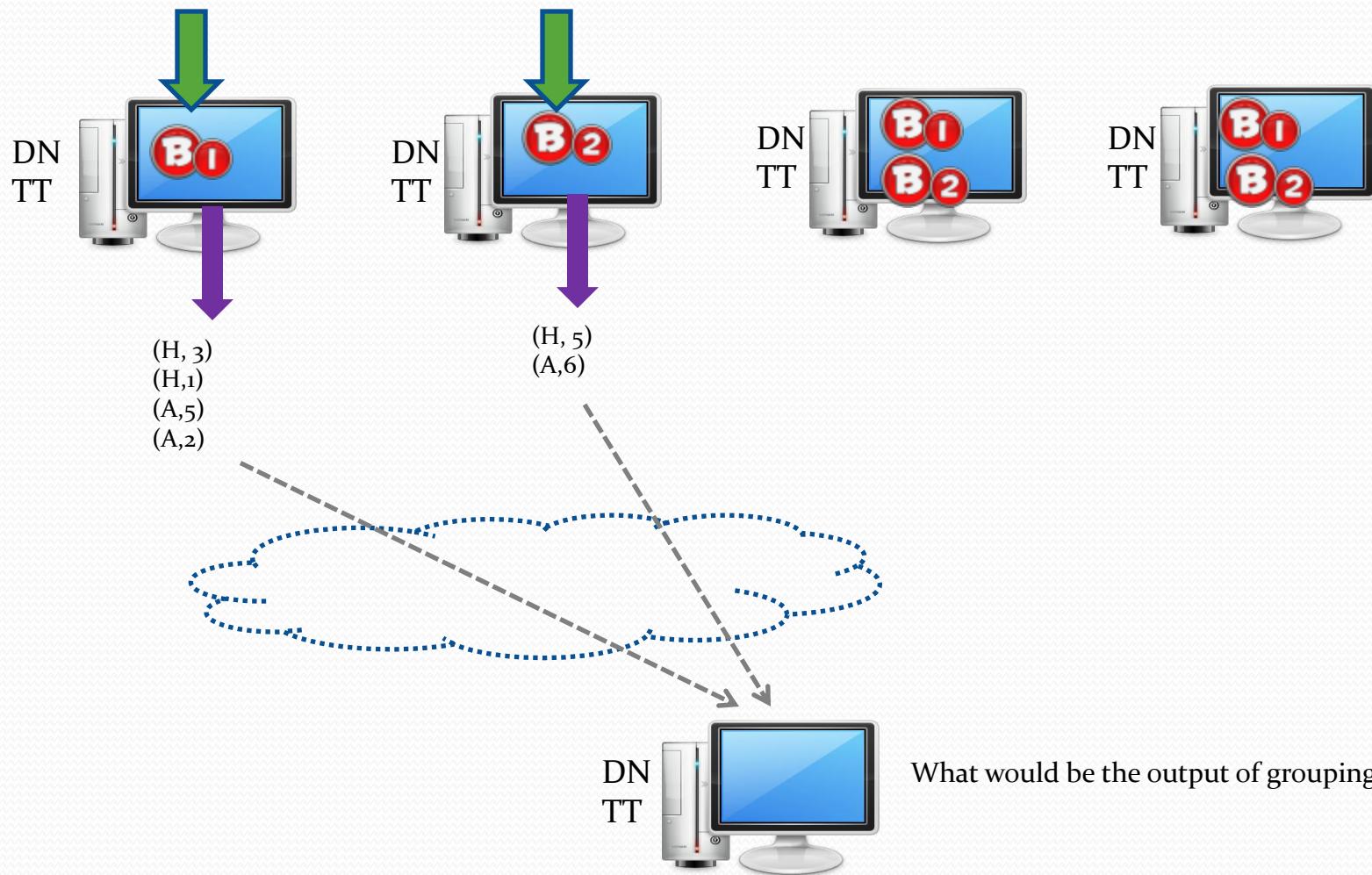
Output key=First character of word

Output value = Length of the word

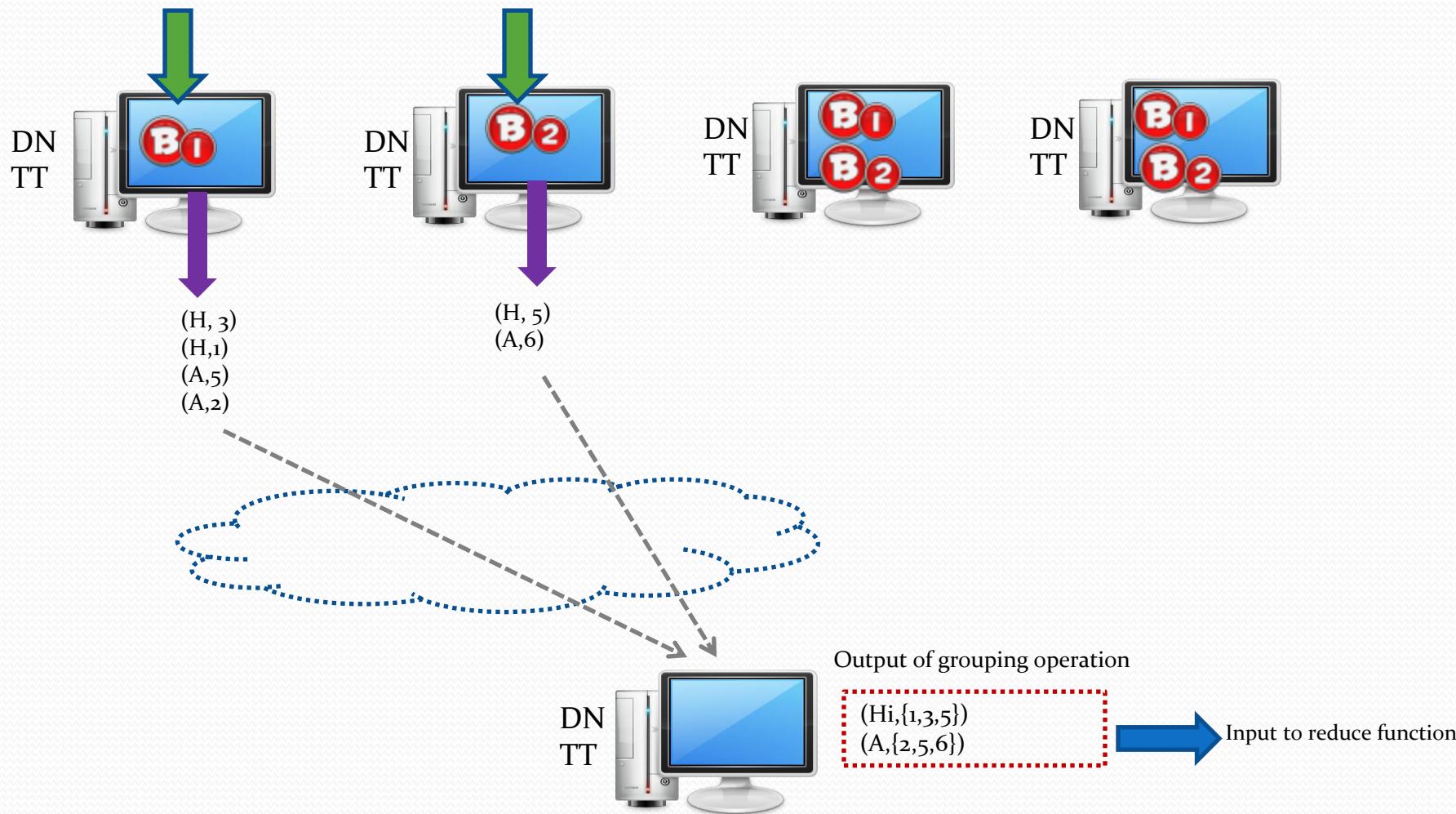
Average Word Length Problem



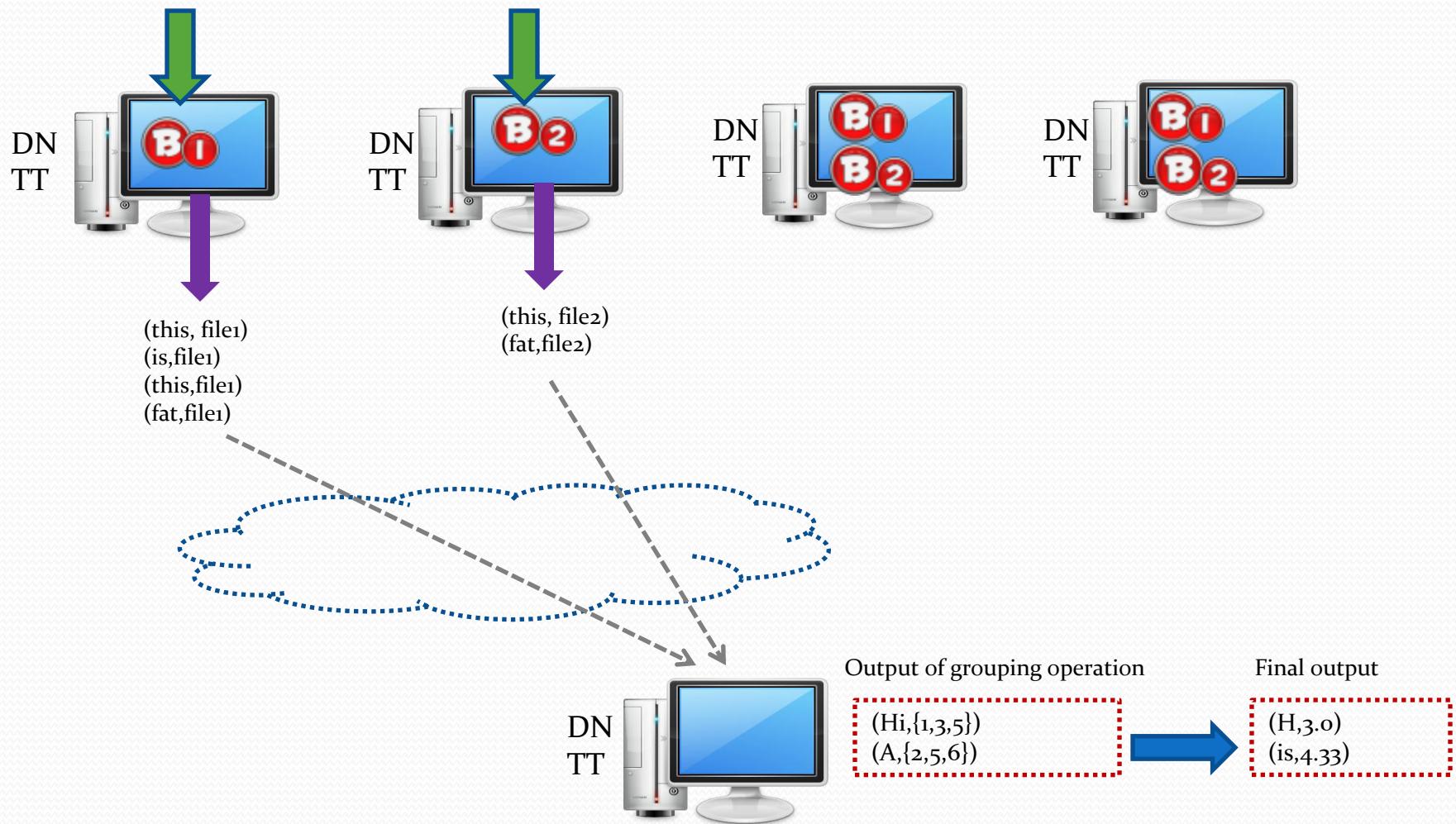
Average Word Length Problem



Average Word Length Problem



Average Word Length Problem



Average Word Length cont'd

- *Mapper*
 - For each word in a line,
emit(firstCharacterOfWord,lengthOfWord)
- *Reducer*
 - You will get a character as key and list of values corresponding to length
 - For each value in *listOfValue*
 - Calculate the sum and also count the number of values
 - *Emit(character,sum/count)*

Module 10

MapReduce Framework – Technical Details

In this module you will learn

- Hadoop primitive data types
- What are the various input formats, and what kind of key values they provide to the mapper function
- Seeing TextInputFormat in detail
- How input split is processed by the mapper?
- Understanding the flow of word count problem

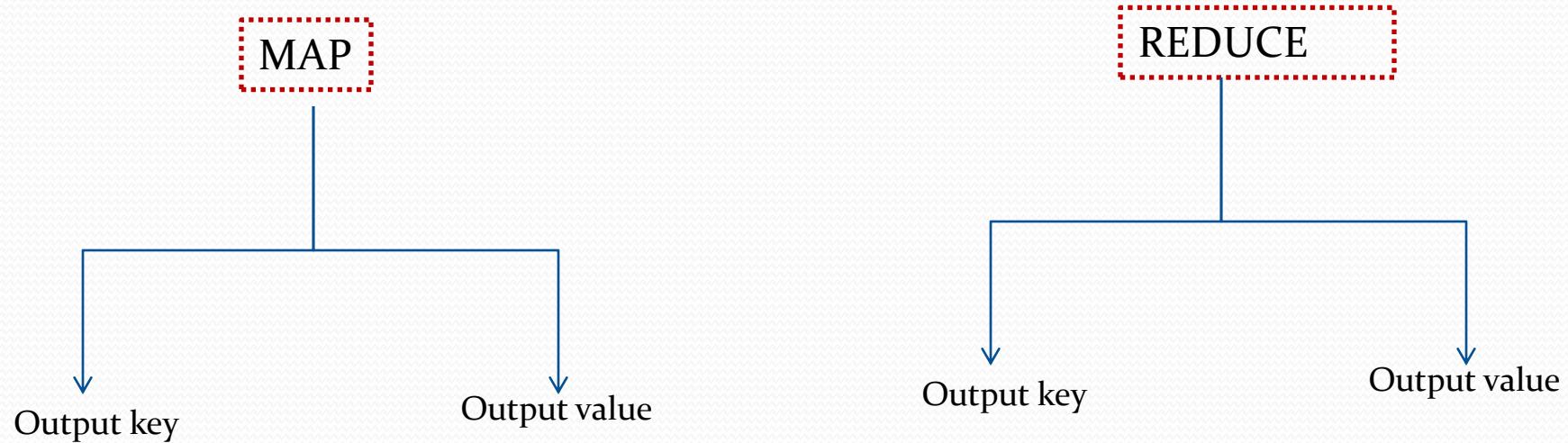
Hadoop Primitive Data types

- Even though Hadoop is written in Java it does not uses java serialization
- Hadoop has its own set of data types for representing Key-Value pairs
 - For efficient serialization over the network
 - While implementing mapper and reducer functionality you need to emit/write ONLY Hadoop Primitive data types OR your custom class extending from Writable or WritableComparable interface(More on this later)

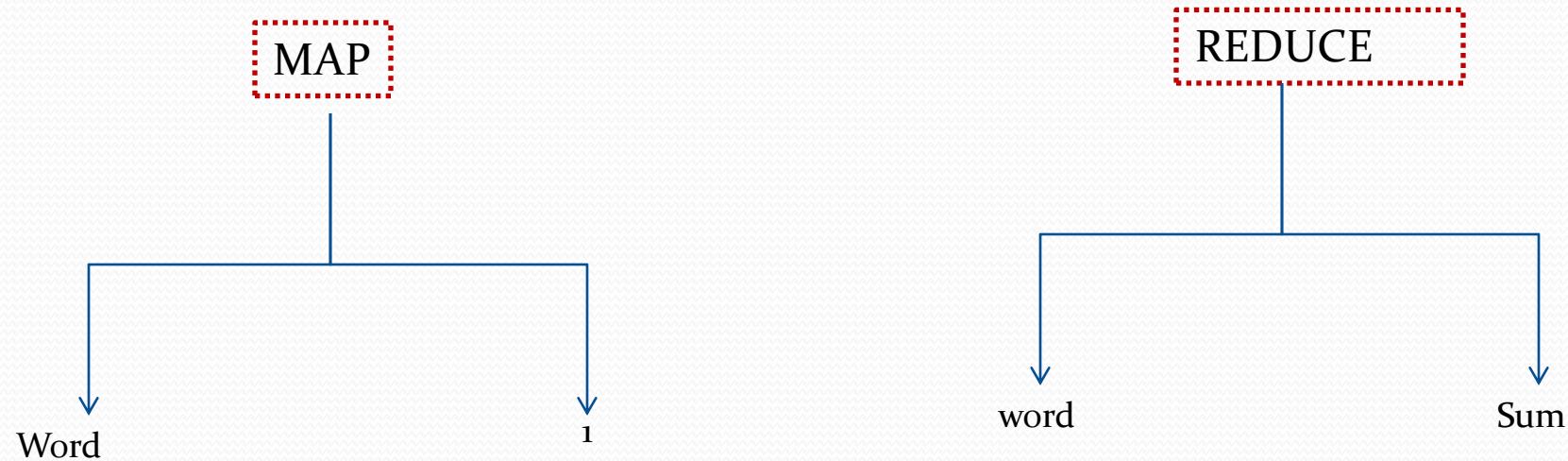
Hadoop Primitive Data types cont'd

Java Primitive(Box) Data types	Hadoop Primitive (Box) Data Types
Integer	IntWritable
Long	LongWritable
Float	FloatWritable
Byte	ByteWritable
String	Text
Double	DoubleWritable
Null	NullWritable
Boolean	BooleanWritable

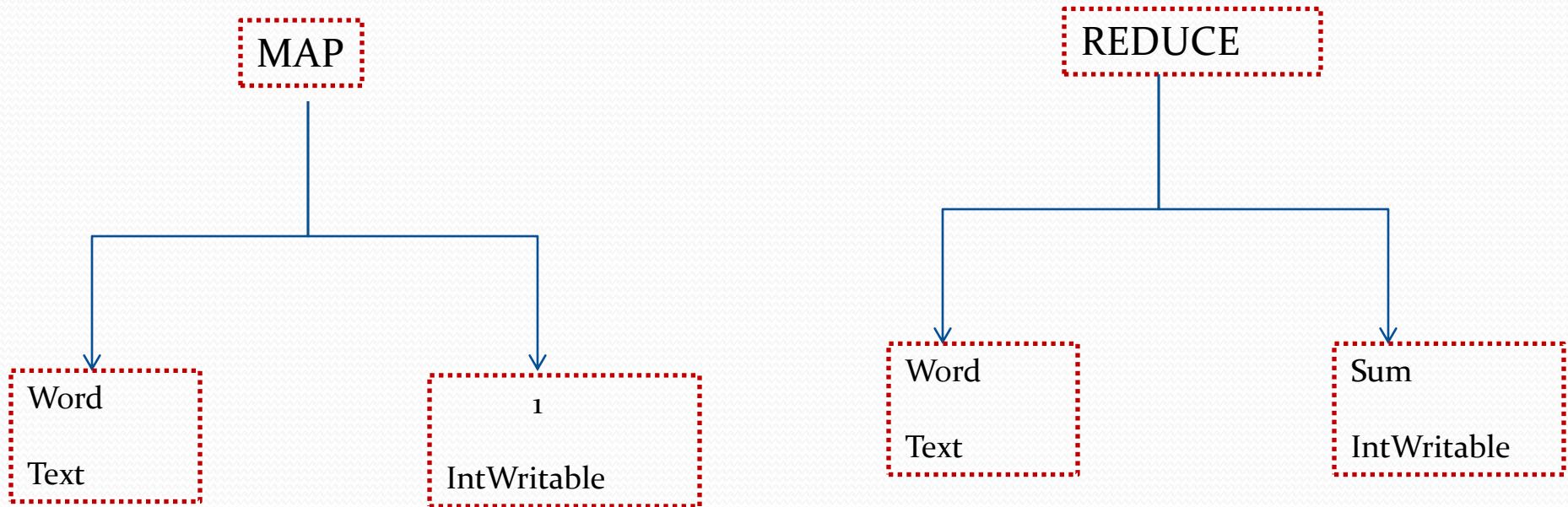
Word Count Problem



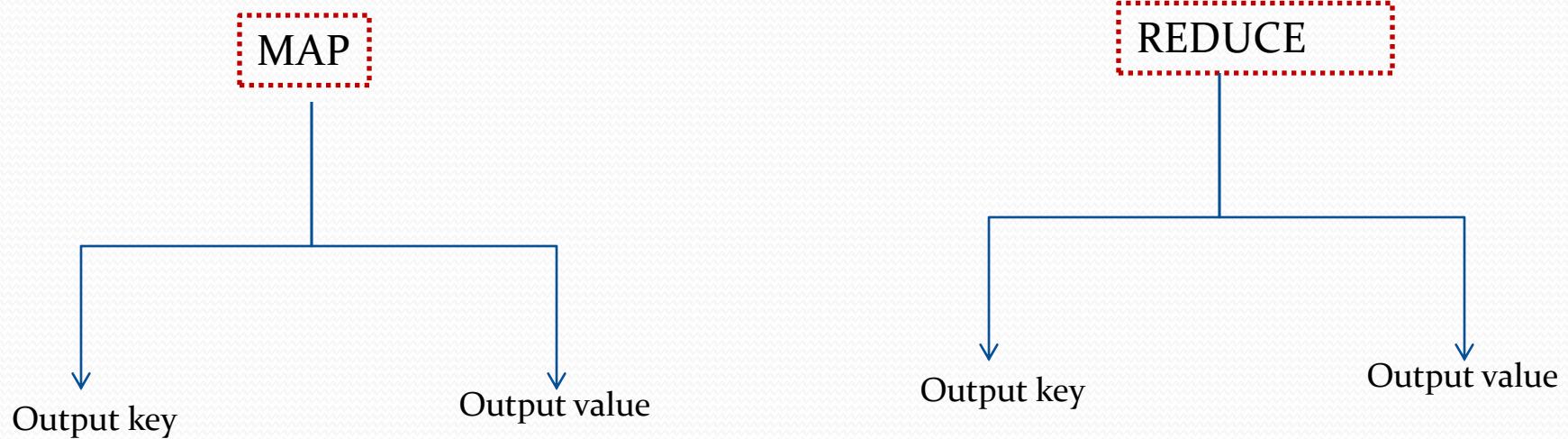
Word Count Problem



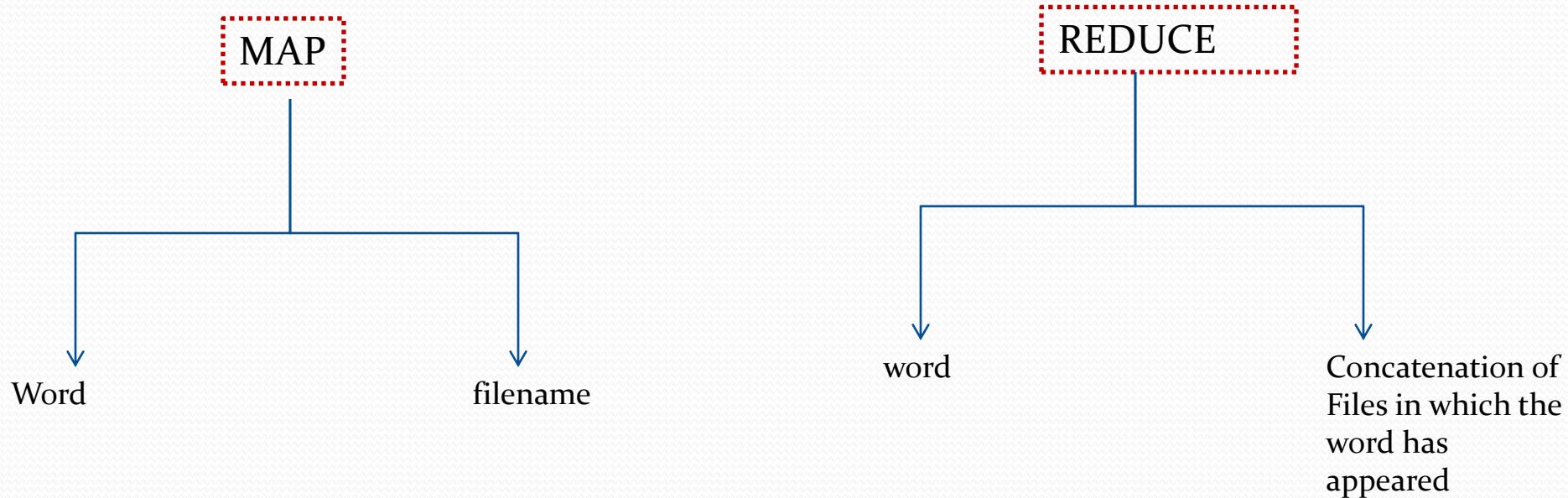
Word Count Problem



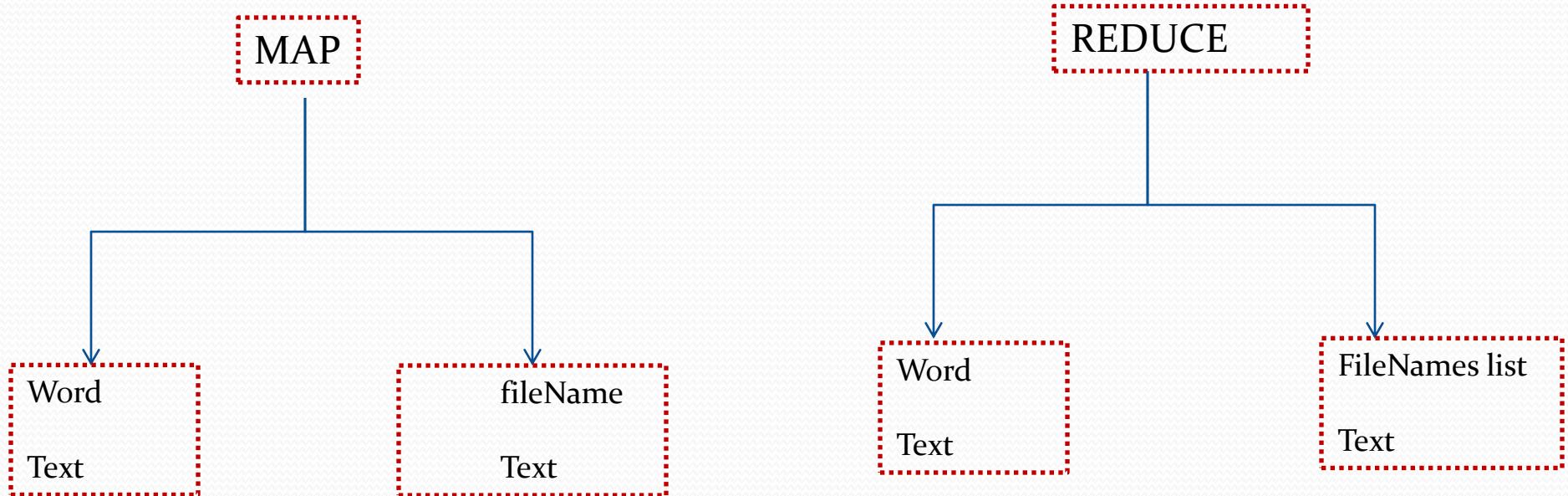
Inverted Indexing Problem



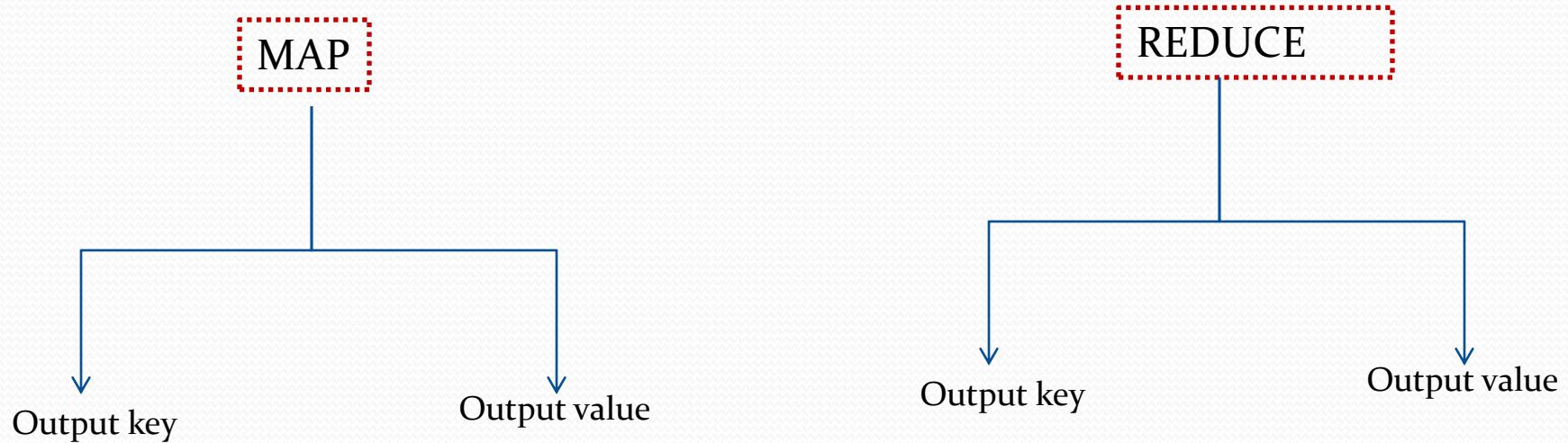
Inverted Indexing Problem



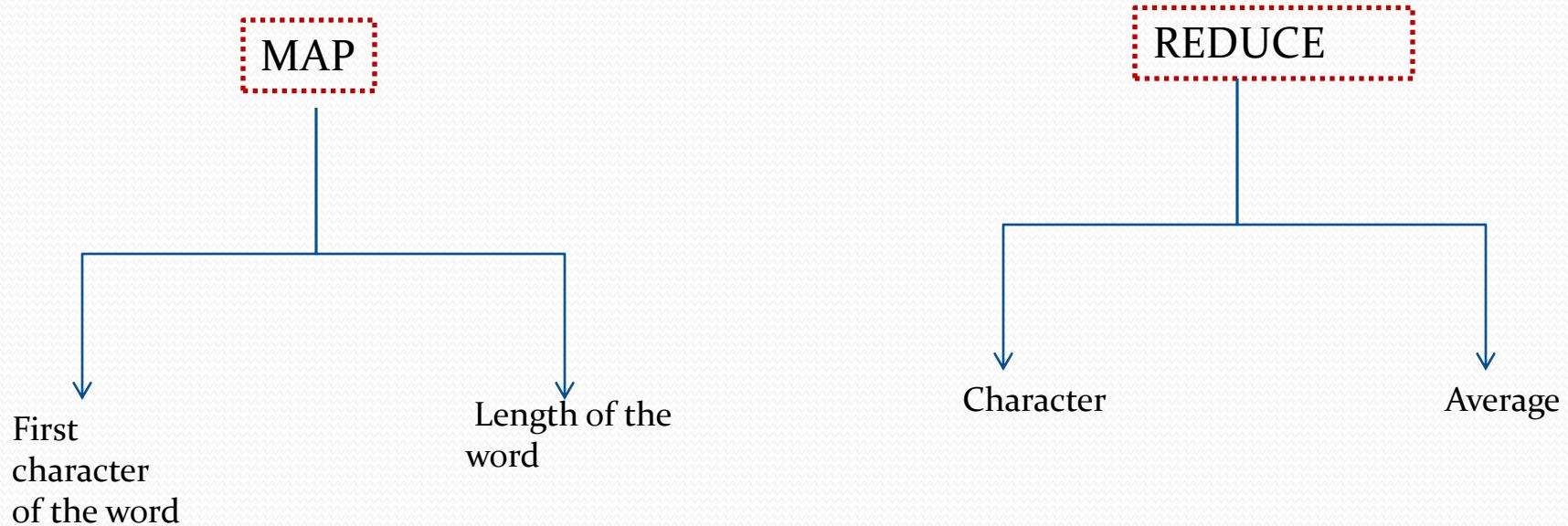
Inverted Indexing Problem



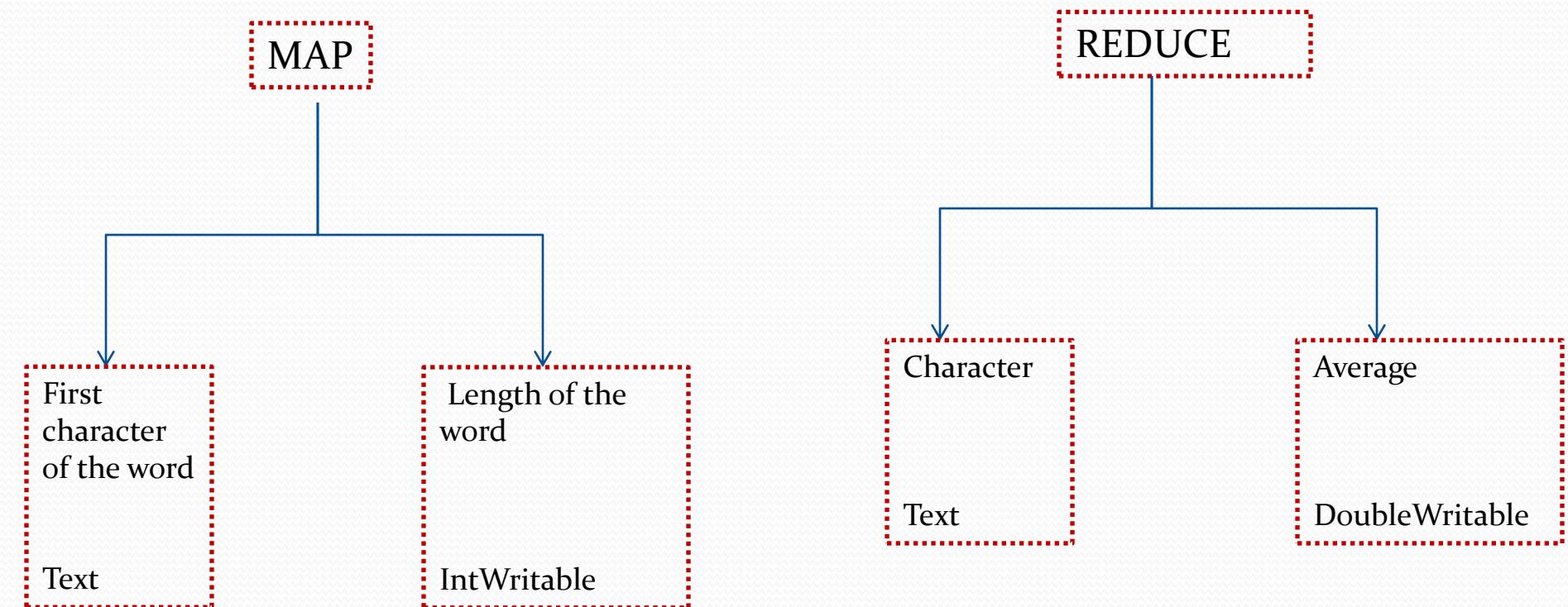
Average Word length



Average Word length



Average Word length



Input Format

Assumption:

A record or complete line is converted into key value pair with

KEY= line number & VALUE = record

Hi, how are you?
I am fine.
How are you doing?

Records converting into (key,value) pair

(0,"Hi,how are you?")
(1,"I am fine.")
(2,"How are you doing?")

```
map( input_key = line number, input_value = complete line)
{
    Break the input_value into individual words and collect them in array

    Iterate over the array element
    {
        if( word is starting with vowel)
        {
            emit(word, 1) //output_key= word which is starting with vowel and output_value=1
        }
    }
}
```

Input Format

- Input format will tell how a record will be converted into key value pair
- The input key type and input value type to the map function is determined by the input format

Input Format cont'd

- Base class is FileInputFormat

Input Format	Key	Value
Text Input Format	Offset of the line within a file	Entire line till "\n" as value
Key Value Text Input Format	Part of the record till the first delimiter	Remaining record after the first delimiter
Sequence File Input Format	Key needs to be determined from the header	Value needs to be determined from the header

Input Format cont'd

Input Format	Key Data Type	Value Data Type
Text Input Format	LongWritable	Text
Key Value Text Input Format	Text	Text
Sequence File Input Format	ByteWritable	ByteWritable

Input Format cont'd

Assuming TextInputFormat is used in word count problem, the map function will look like as follows:

```
public void map(LongWritable offset, Text value, Context context)
{
}

map( input_key = line number, input_value = complete line)
{
    Break the input_value into individual words and collect them in array

    Iterate over the array element
    {
        if( word is starting with vowel)
        {
            emit(word, 1) //output_key= word which is starting with vowel and output_value=1
        }
    }
}
```

Input Format cont'd

```
map( input_key = line number, input_value = complete line)
{
    Break the input_value into individual words and collect
    them in array

    Iterate over the array element
    {
        if( word is starting with vowel)
        {
            emit(word, 1) //output_key= word which is
            starting with vowel and output_value=1
        }
    }
}
```

```
:
public void map(LongWritable offset, Text value,
Context context)
{

String str = value.toString();
String[] splits = str.split("\\W+");

for(String w:splits)
{
    context.write(new Text(word),new
    IntWritable(1));
}

}
```

Word Count Problem

- Counting the number of times a word has appeared in a file
- Example:

Hello, how are you?

Hello, I am fine? How about you?

I am solving word count problem

Word Count Problem cont'd

Hello, how are you?

Hello, I am fine? How about you?

I am solving word count problem

Output of the Word Count problem

Key	Value
Hello	2
you	2
I	2

Word Count Mapper

- Assuming one input split

```
1 Hello, how are you?  
2 Hello, I am fine? How about you?  
3 |I am solving word count problem
```

- Input format is Text Input Format
 - Key = Offset which is of type LongWritable
 - Value = Entire Line as Text
- Remember map function will be called 3times
 - Since there are only 3 records in this input split

Word Count Mapper cont'd

- Map function for this record

1 Hello, how are you?

- Map(1>Hello, how are you?) ==>

Input to the map function

(Hello,1)
(how,1)
(are,1)
(you,1)

Intermediate key value pairs from
the mapper

Word Count Mapper cont'd

- Map function for this record

```
Hello, I am fine? How about you?
```

- Map(2, Hello, I am fine? How about you?)====>

Input to the mapper

(Hello,1)
(I,1)
(am,1)

⋮
⋮
⋮

Intermediate key value pair from the
mapper

Word Count Mapper cont'd

Pseudo Code

```
Map (inputKey, inputValue)
{
    Break the inputValue into individual words;
    For each word in the individual words
    {
        write (word ,1)
    }
}
```

Word Count Reducer

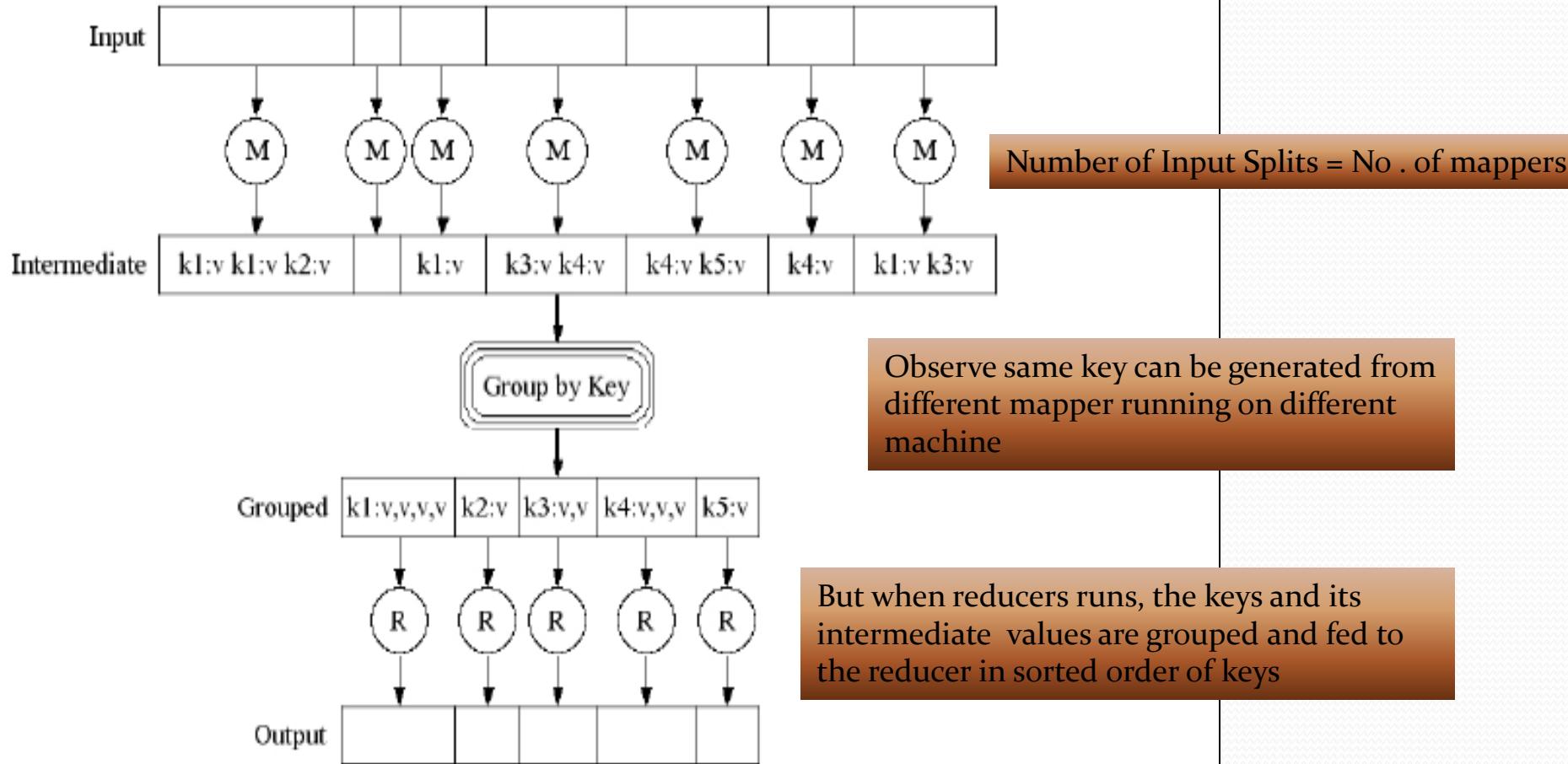
- Reducer will receive the intermediate key and its list of values
- If a word “*Hello*” has been emitted 4 times in the map phase, then input to the reducer *will be*
 - $(Hello, \{1, 1, 1, 1\})$
- To count the number of times the word “*Hello*” has appeared in the file, just add the number of 1’s

Word Count Reducer cont'd

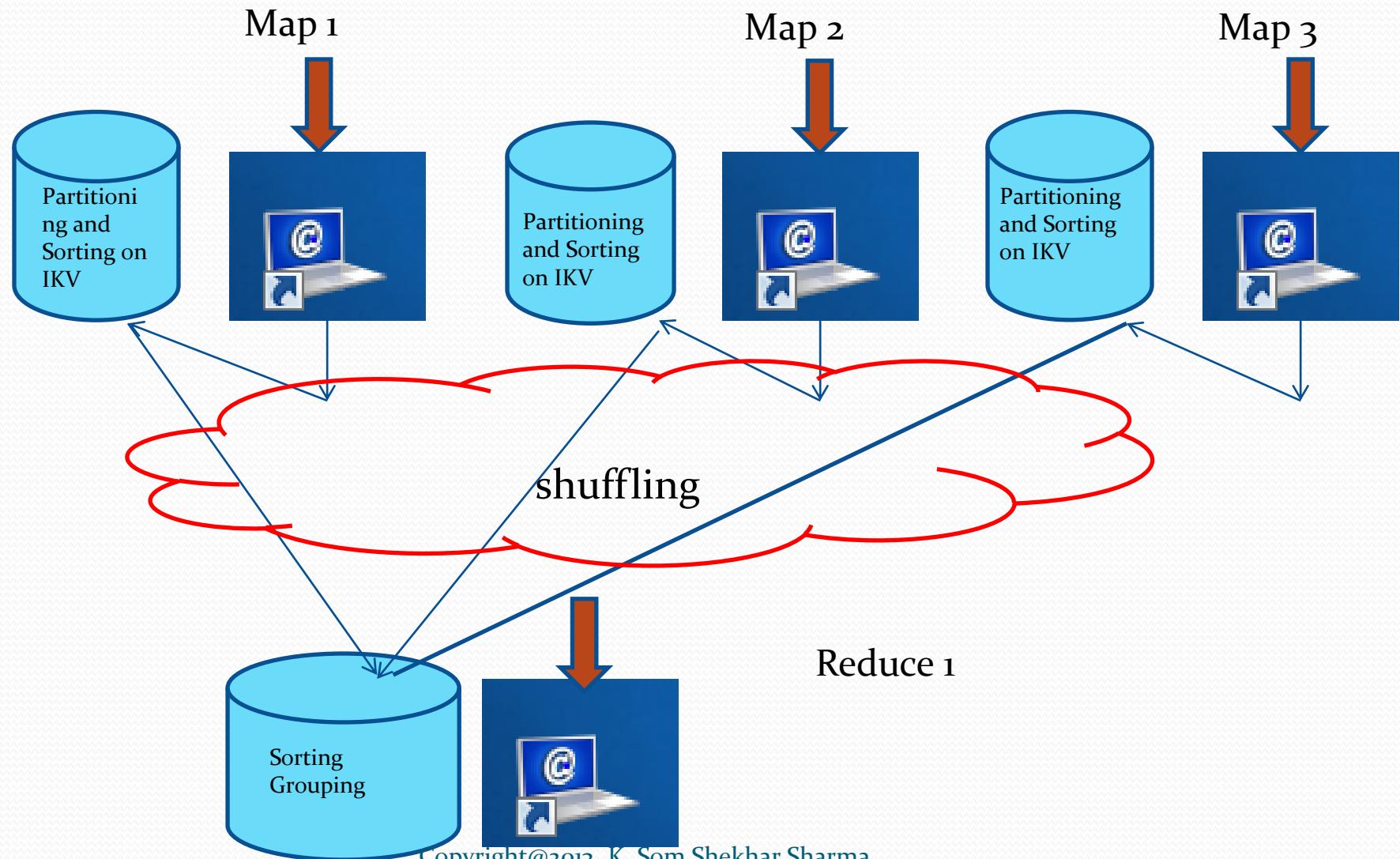
Pseudo Code

```
Reduce (inputKey, listOfValues)
{
    sum = 0;
    for each value in the listOfValues
    {
        sum = sum + value;
    }
    write(inputKey,sum)
}
```

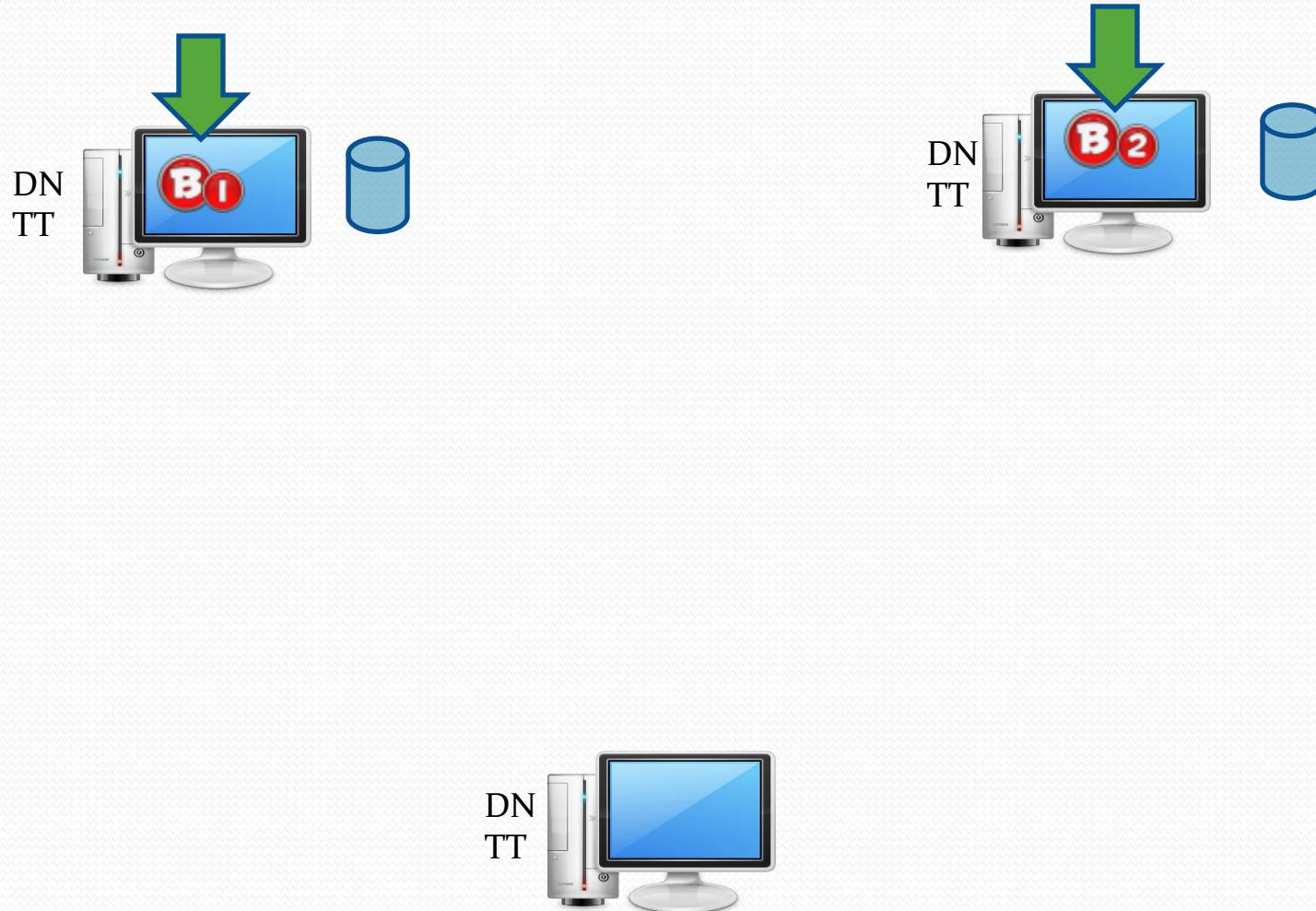
MapReduce Flow-1



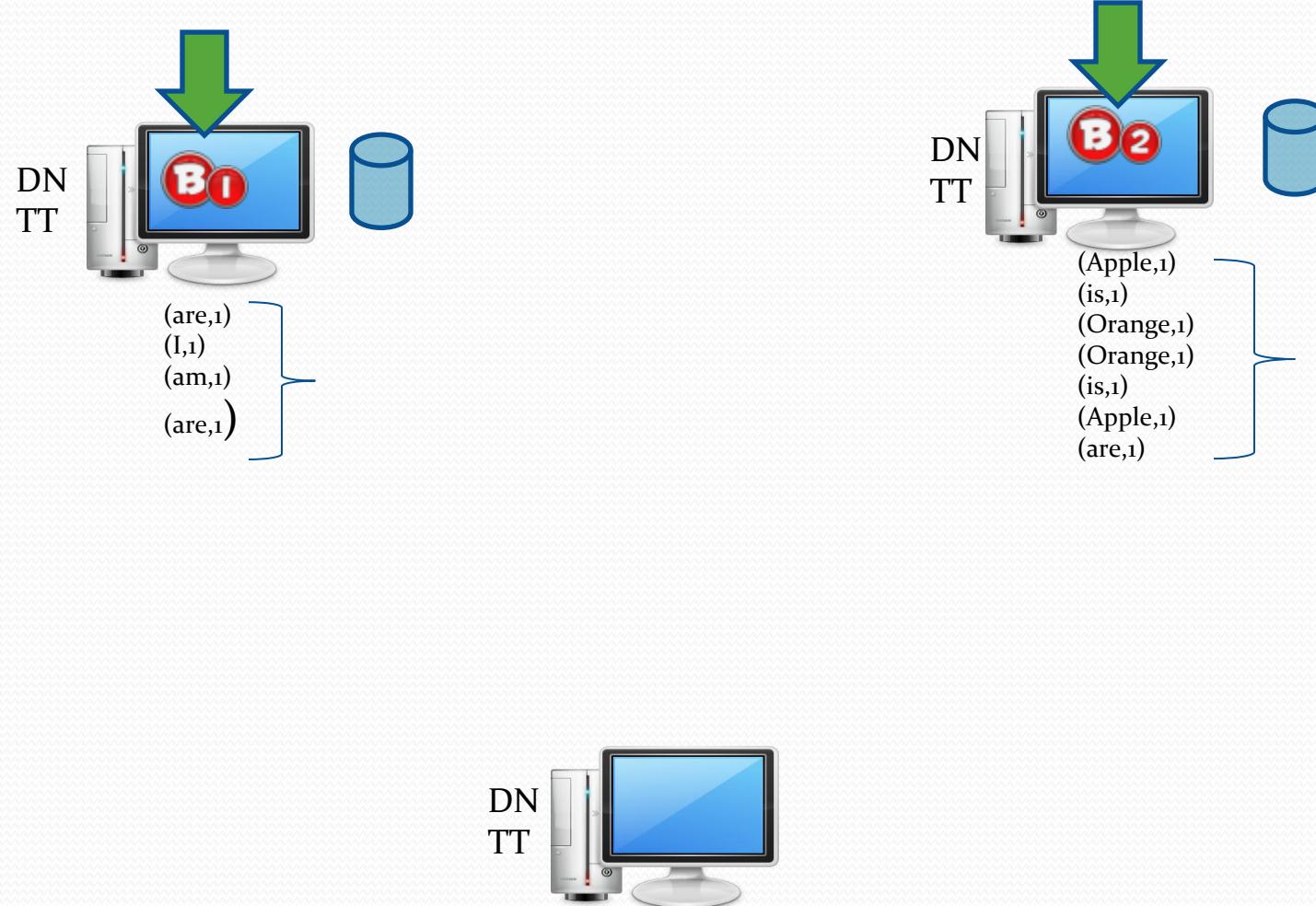
MapReduce – Data Flow



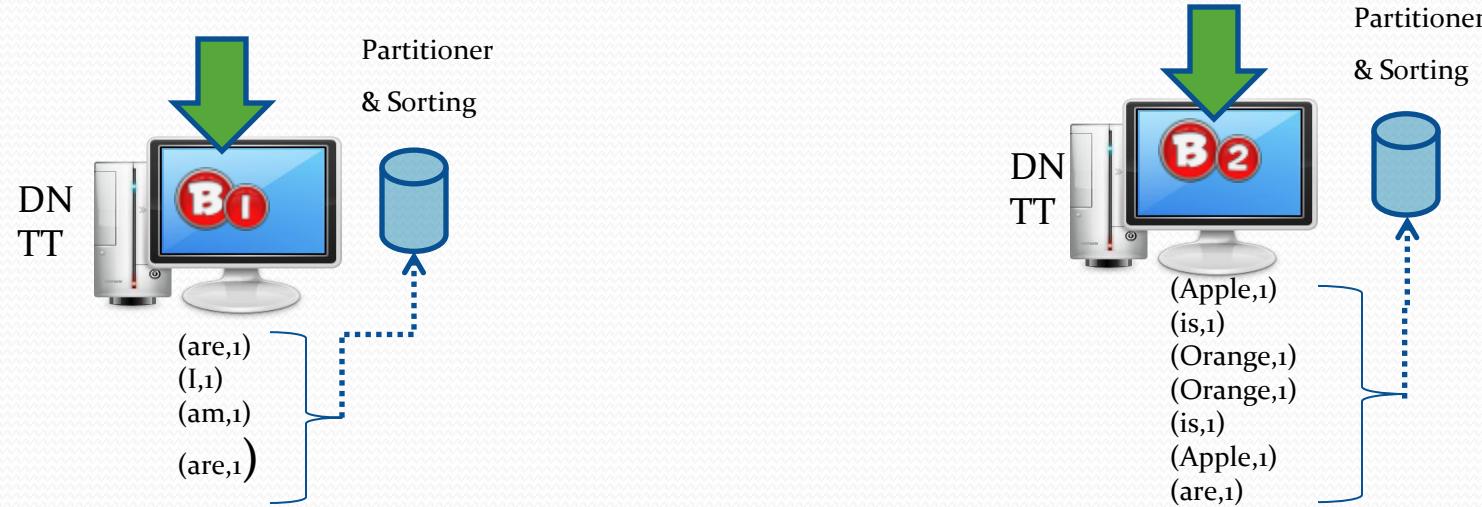
Map Reduce –Data Flow



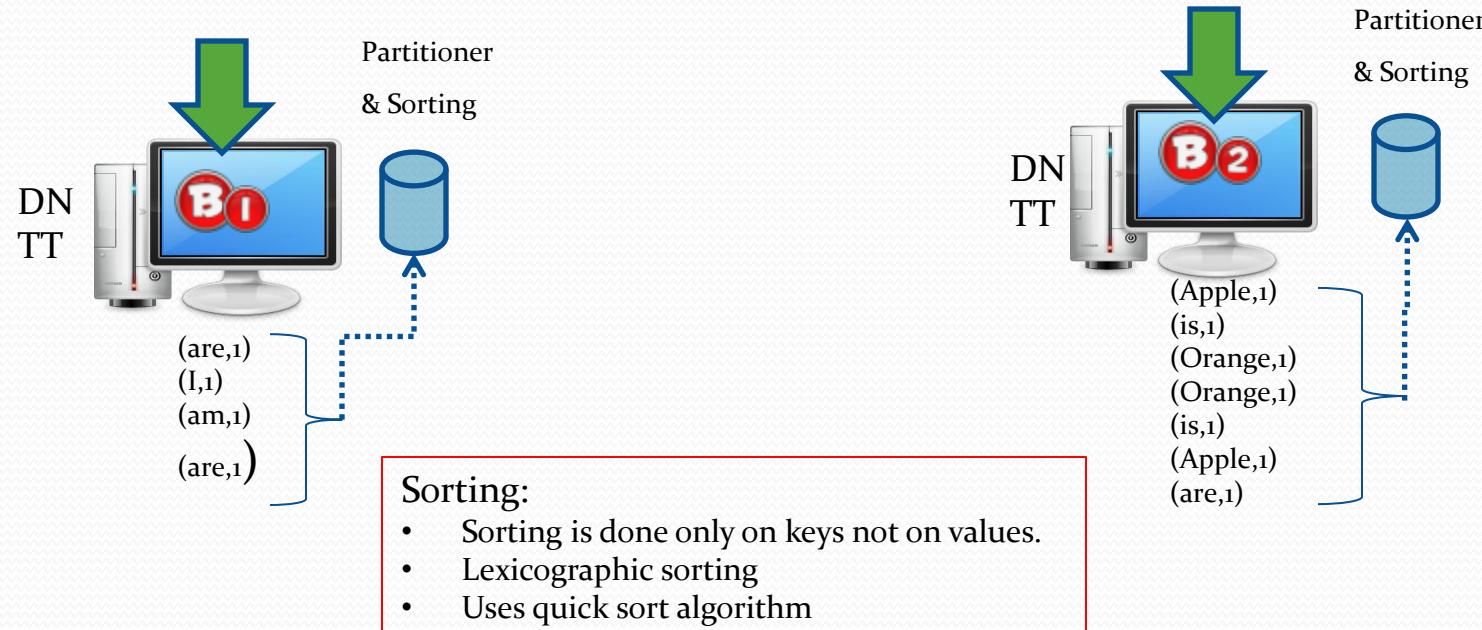
Map Reduce –Data Flow



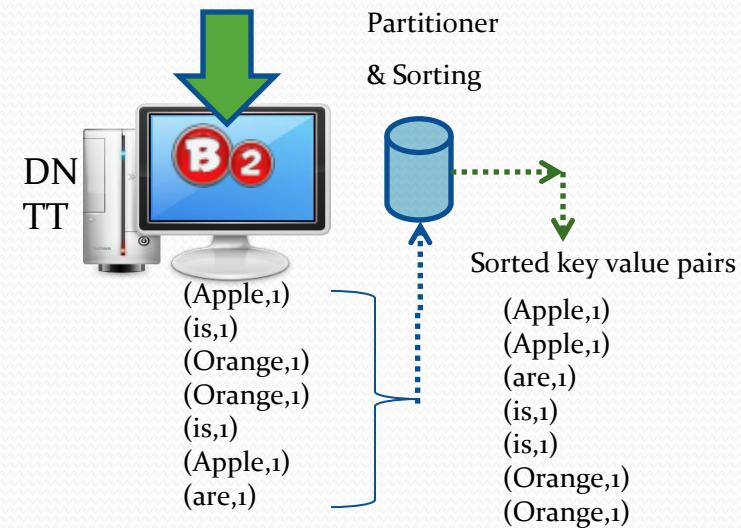
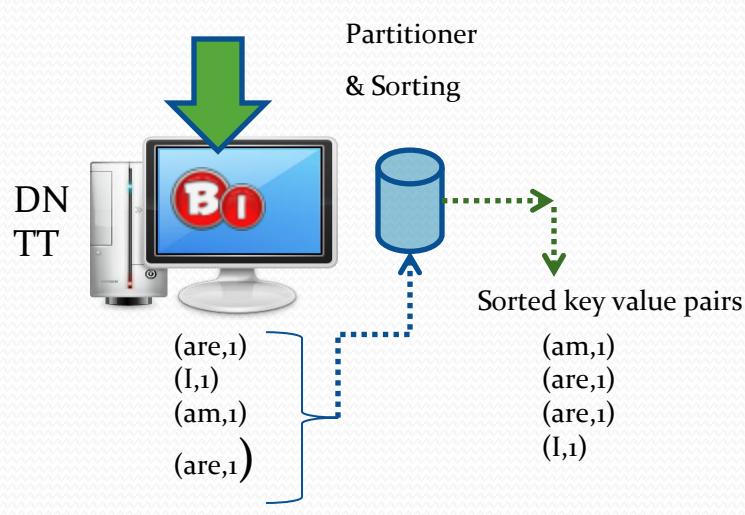
Map Reduce –Data Flow



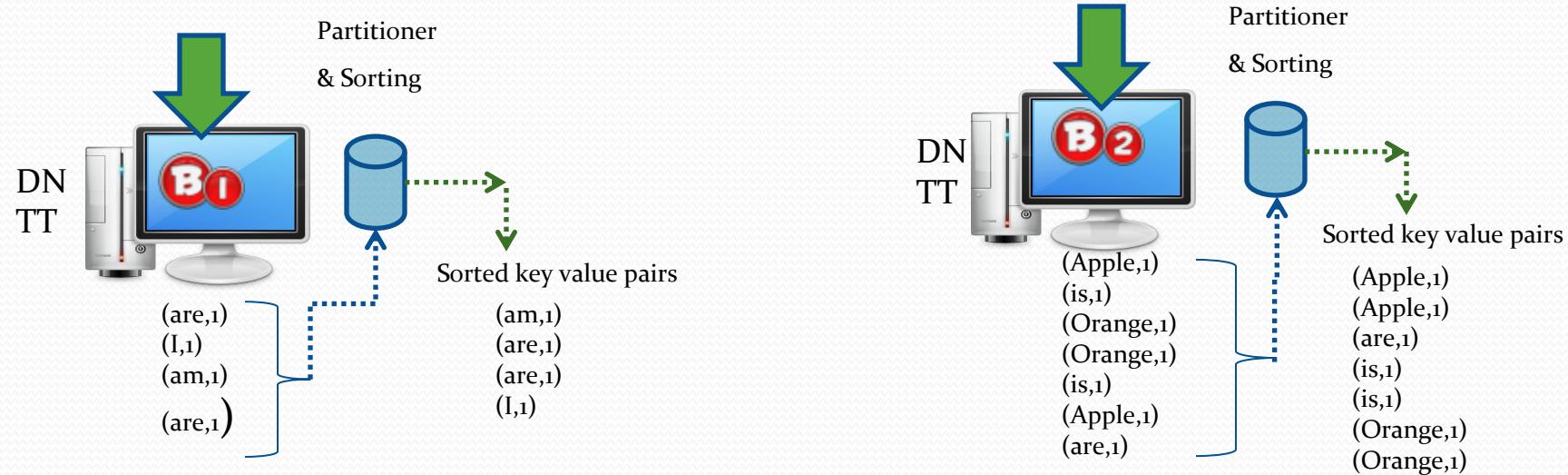
Map Reduce –Data Flow



Map Reduce –Data Flow



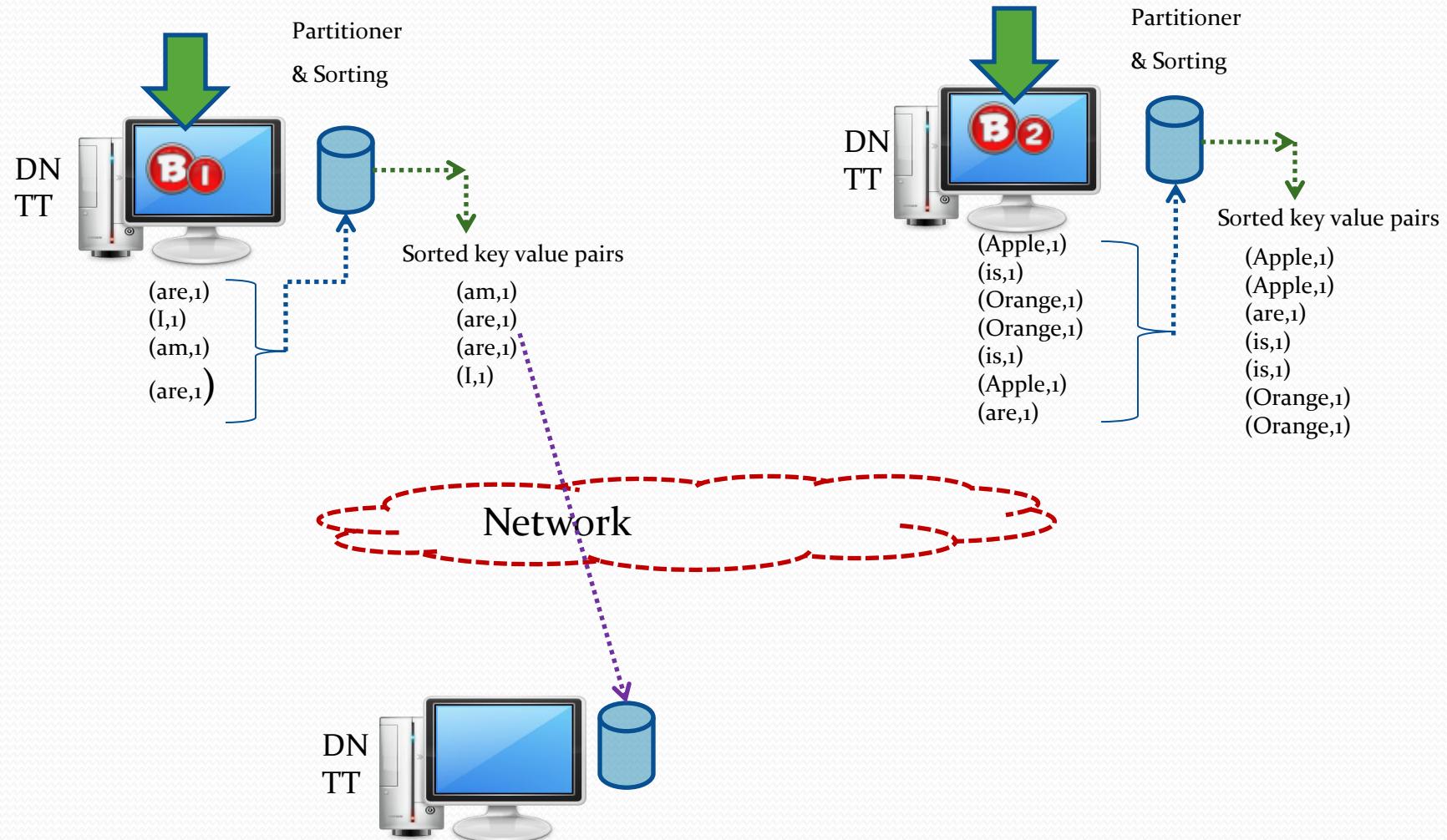
Map Reduce –Data Flow



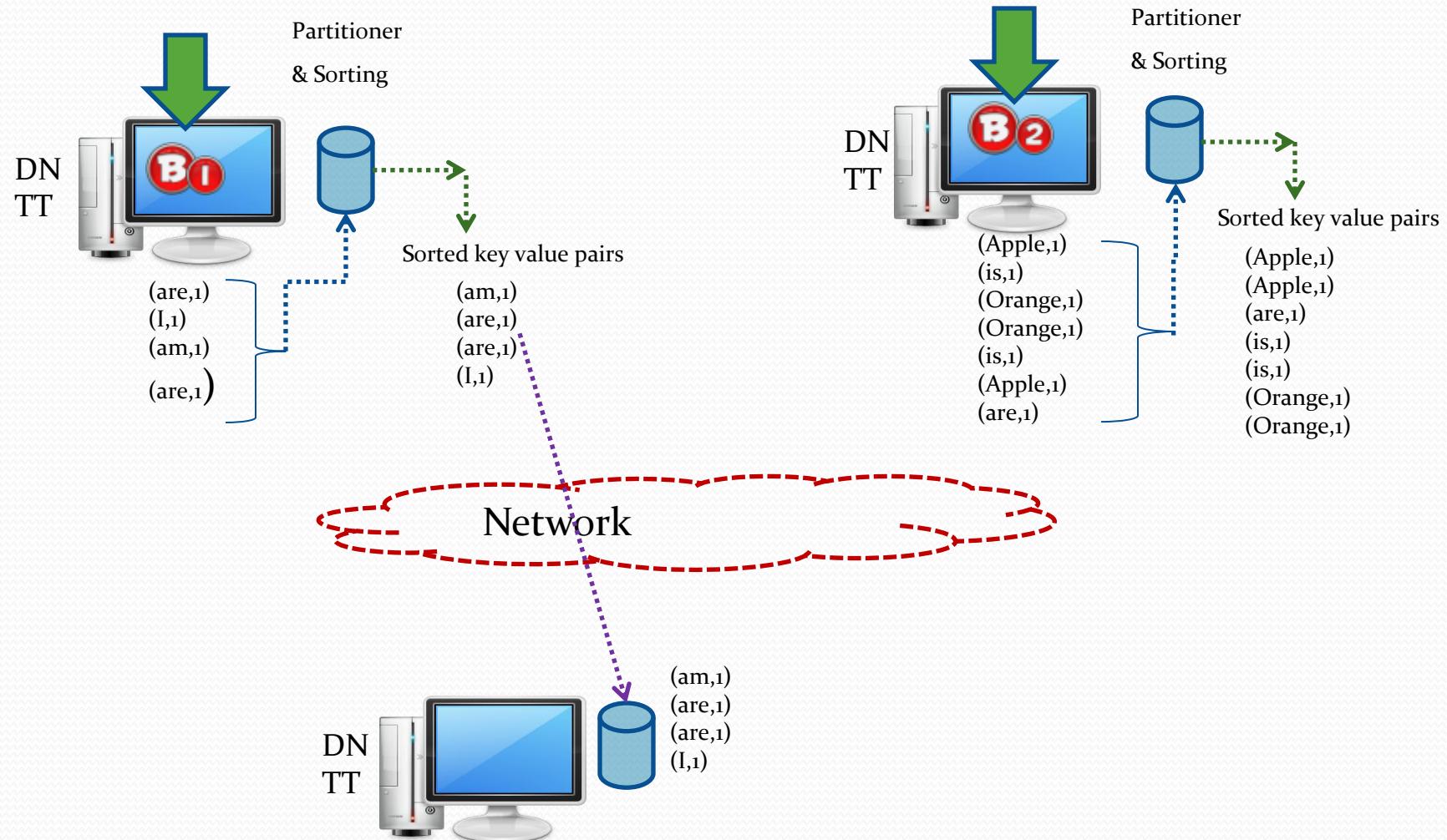
Shuffling process will start as soon as any of the map tasks gets completed



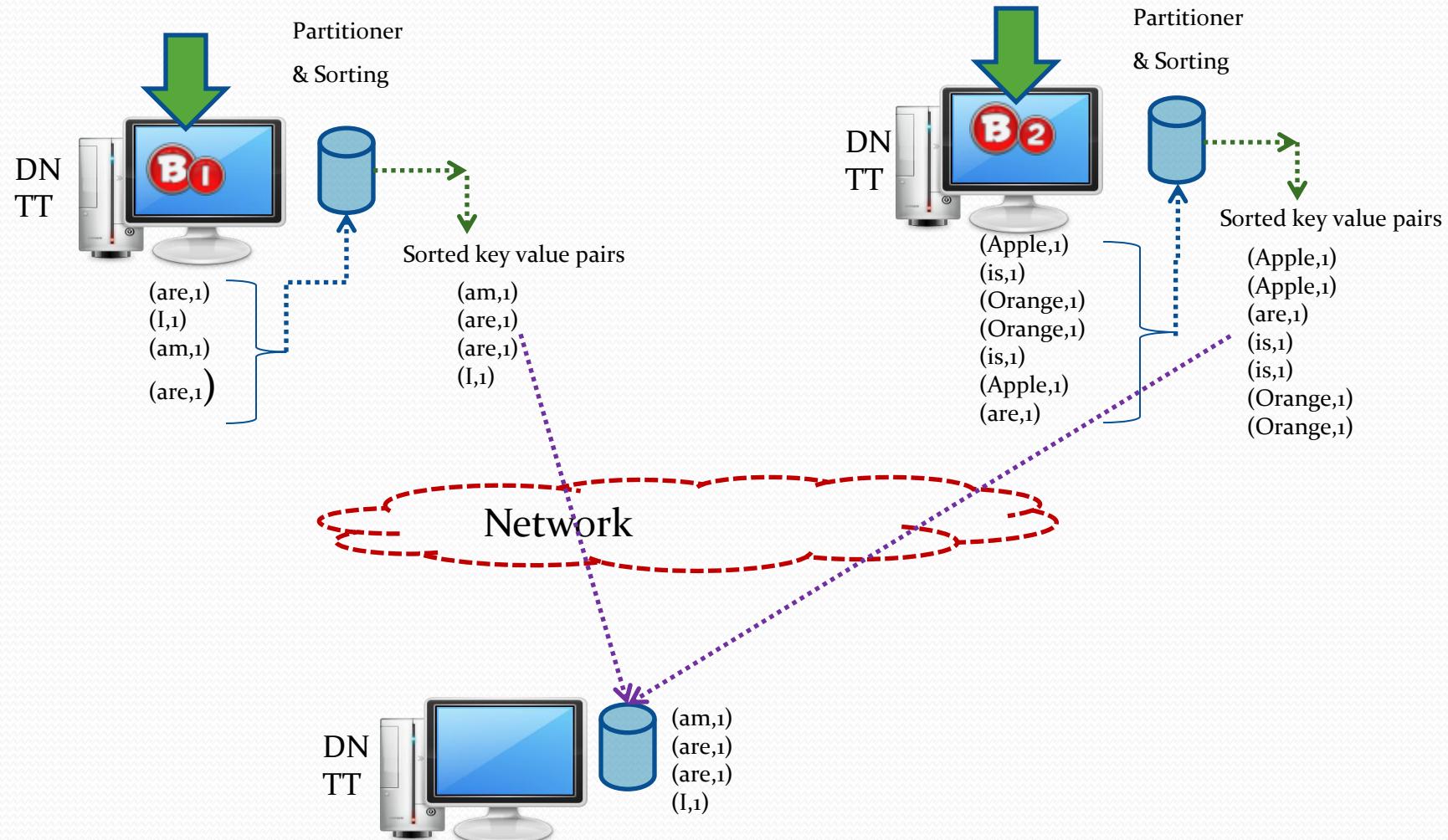
Map Reduce –Data Flow



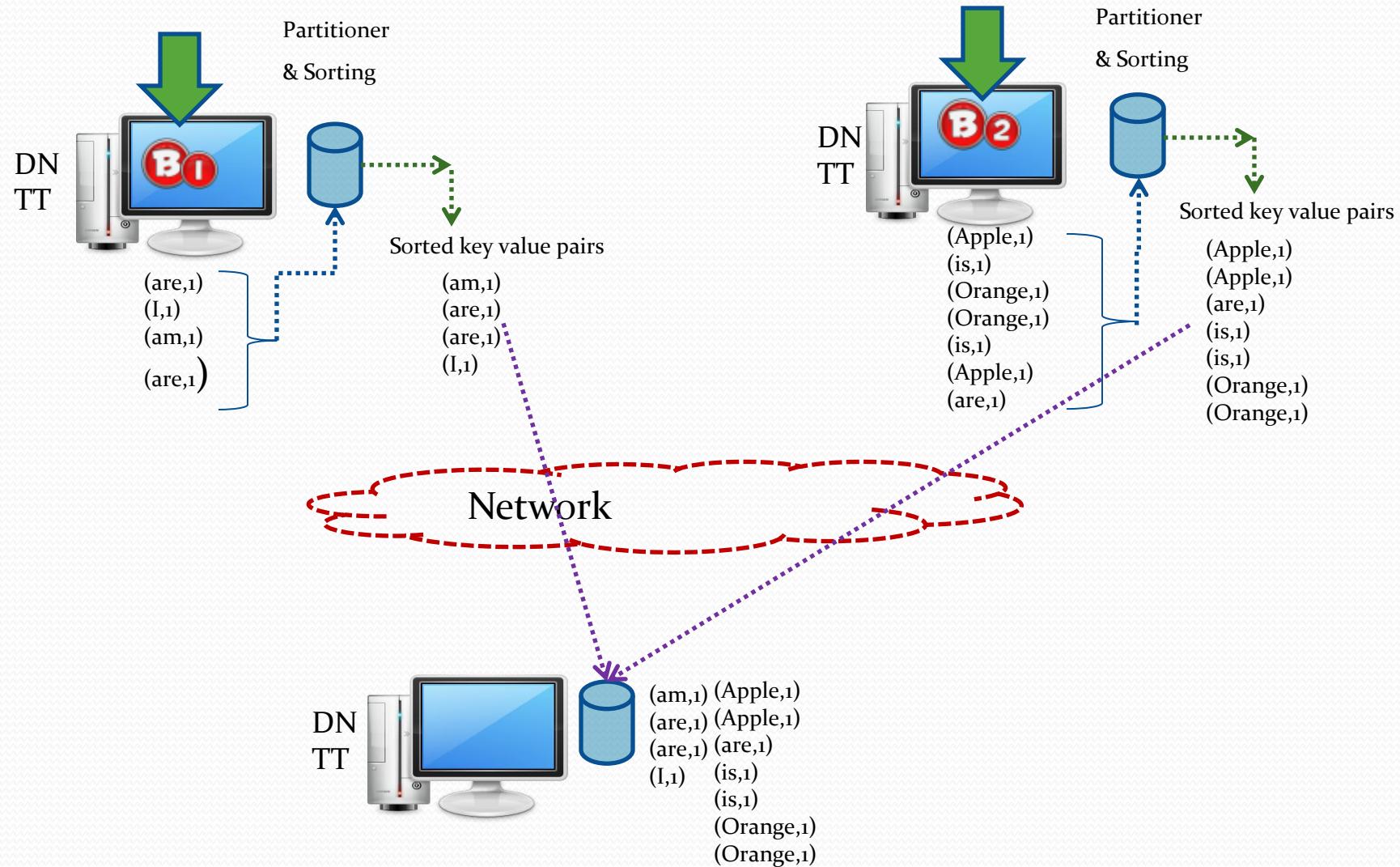
Map Reduce –Data Flow



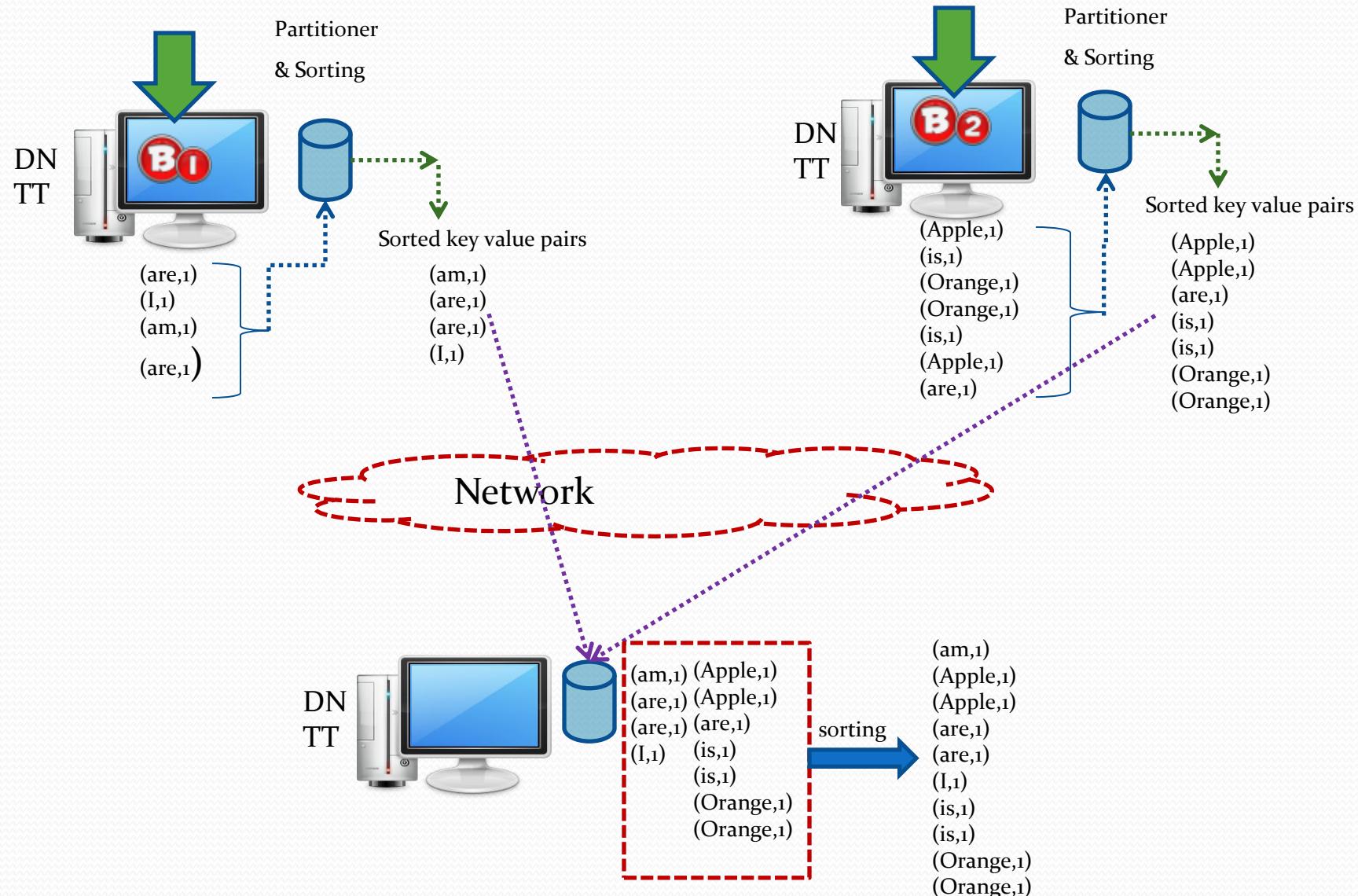
Map Reduce –Data Flow



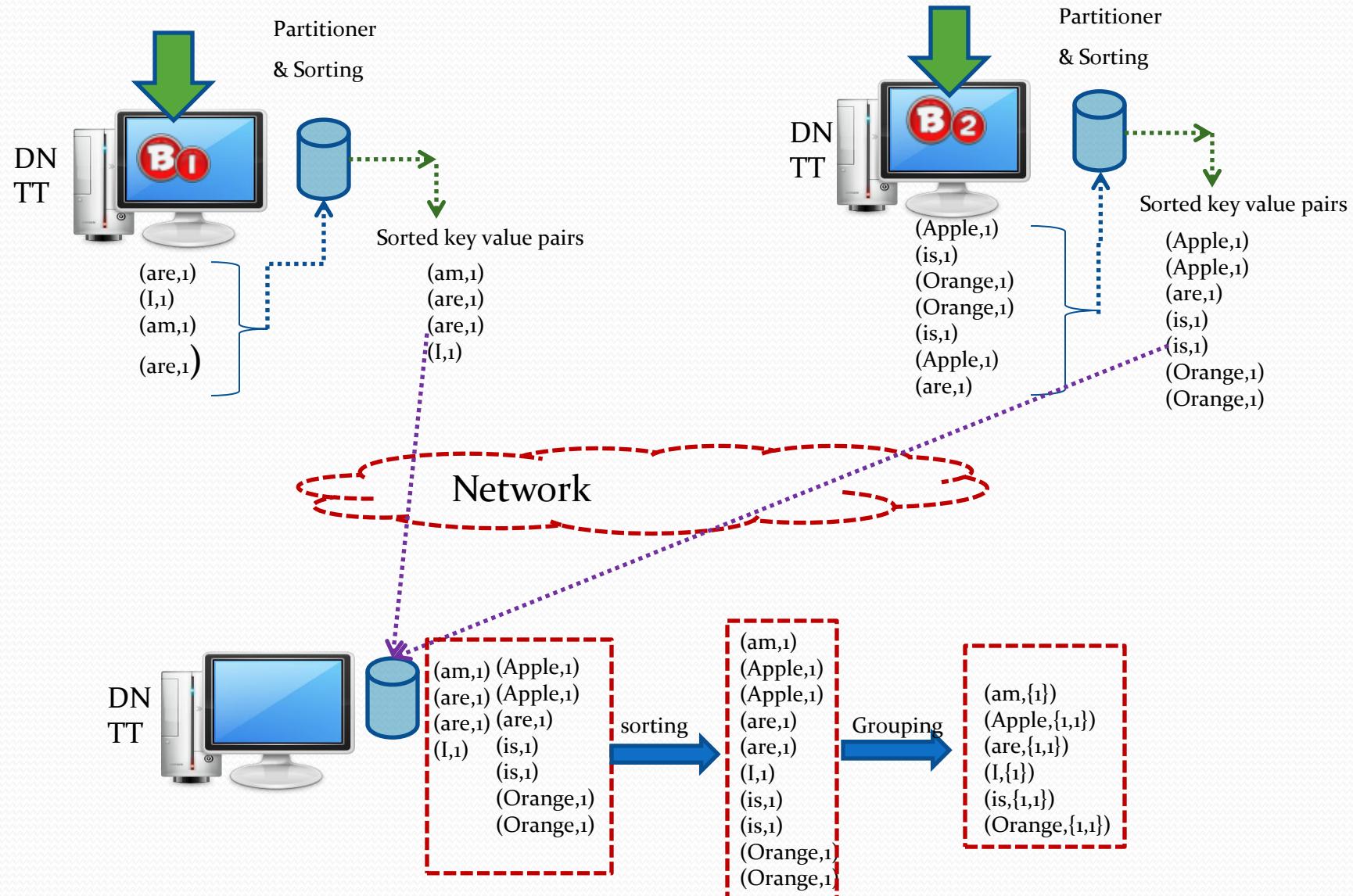
Map Reduce –Data Flow



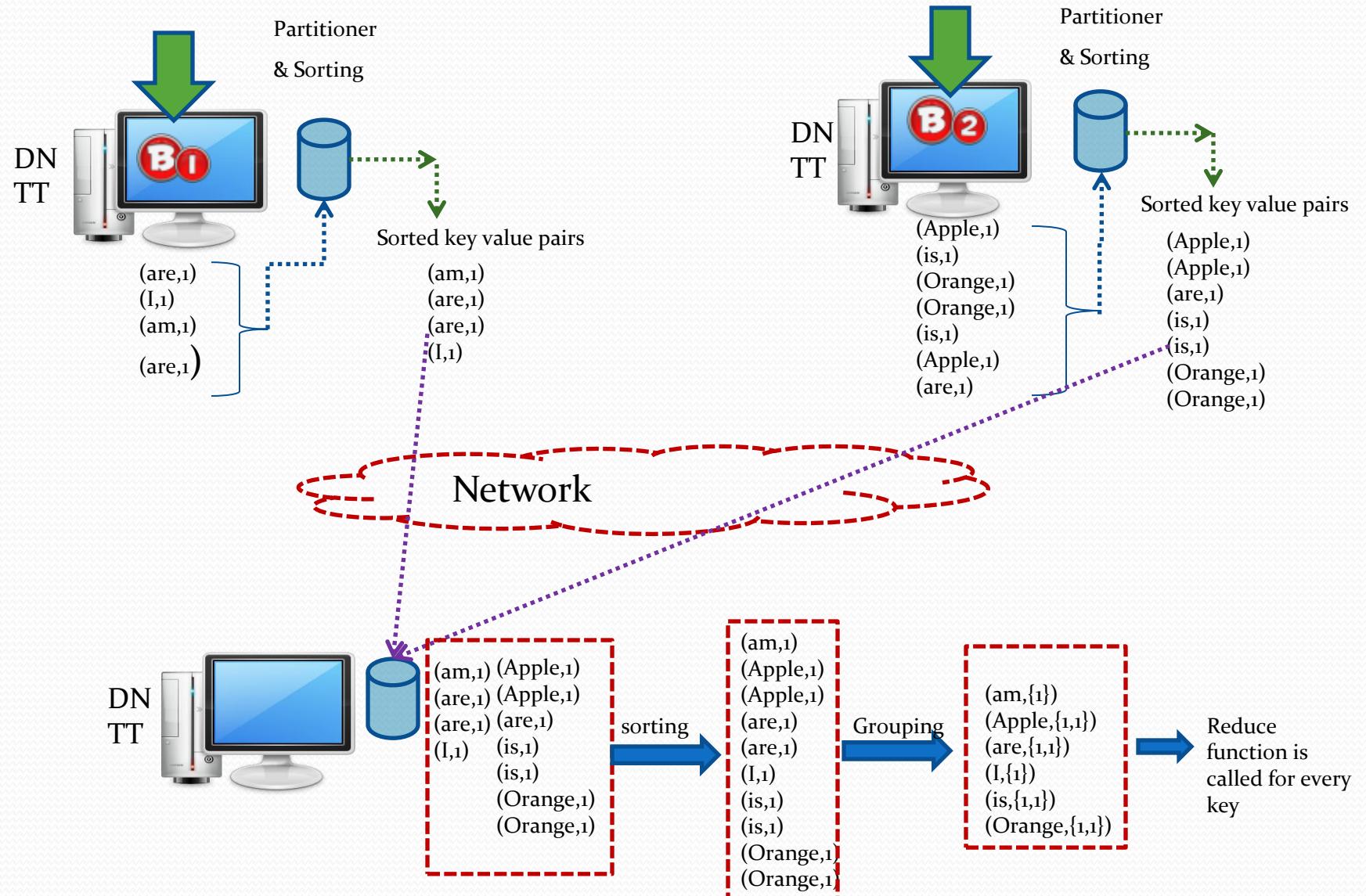
Map Reduce –Data Flow



Map Reduce –Data Flow



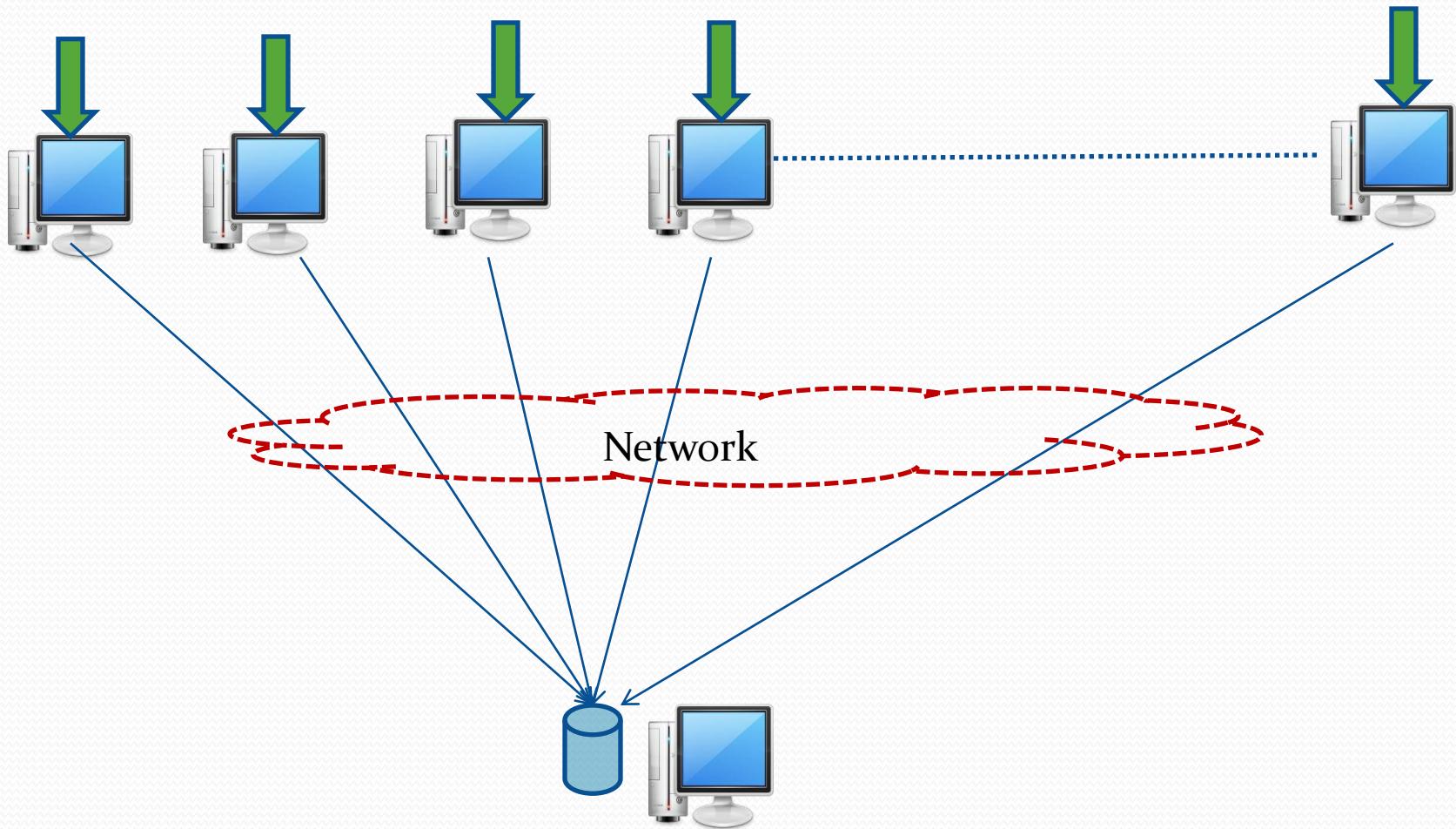
Map Reduce –Data Flow



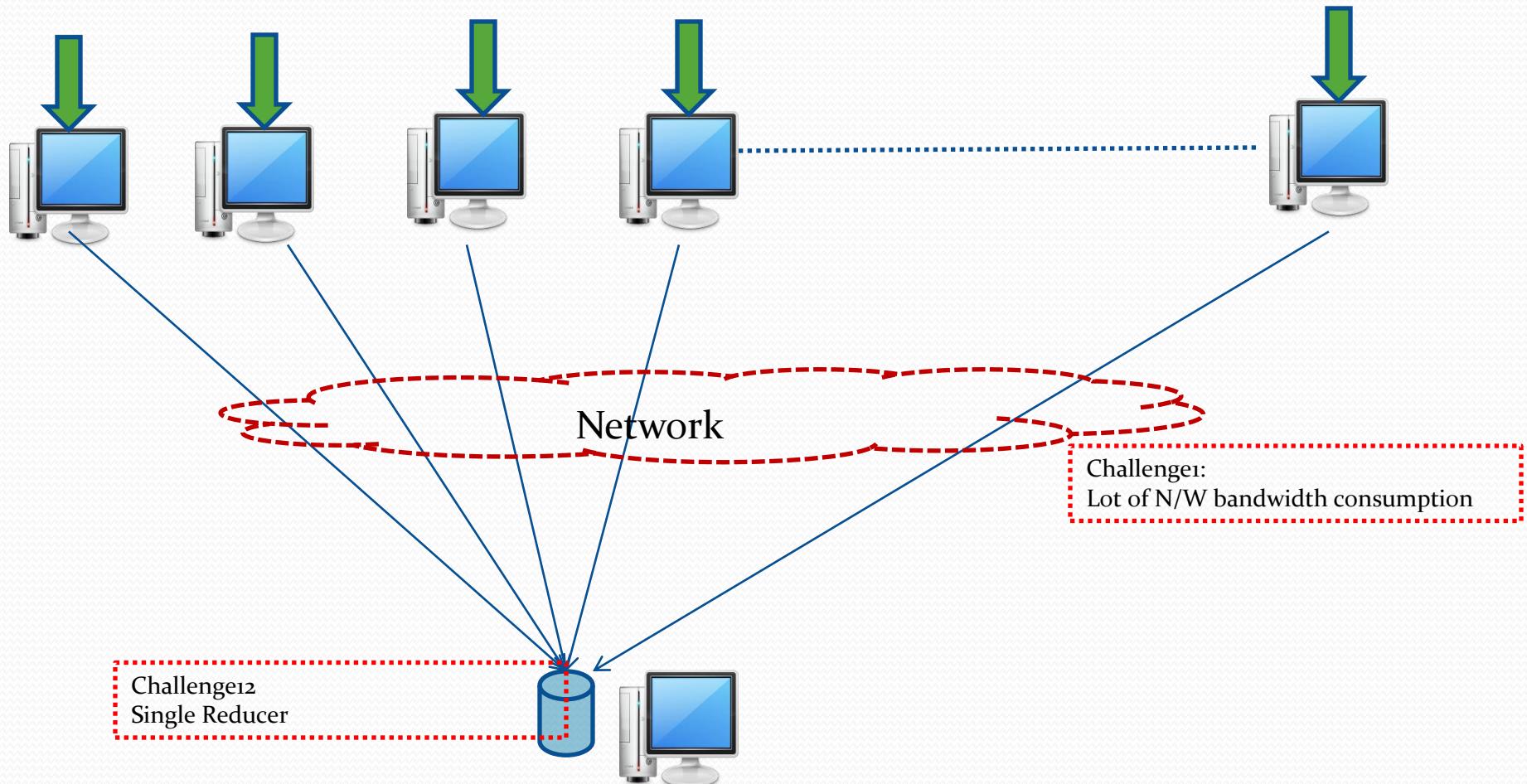
Important feature of map reduce job

- Intermediate output key and value generated by the map phase is stored on local file system
 - It is stored as sequence file (hadoop-temp/mapred)
 - It is kept on the local file system till the job is completed. Once completed this intermediate output is deleted
- Data localization is not applicable for reducer
- Sorting happens only on keys not on values
 - Values will be arbitrarily ordered and the ordering may vary for every run of same map reduce job
- SORTING on keys happens both after the mapper phase and before the reducer phase
 - Sorting at mapper phase is just an optimization

Challenges in map reduce data flow



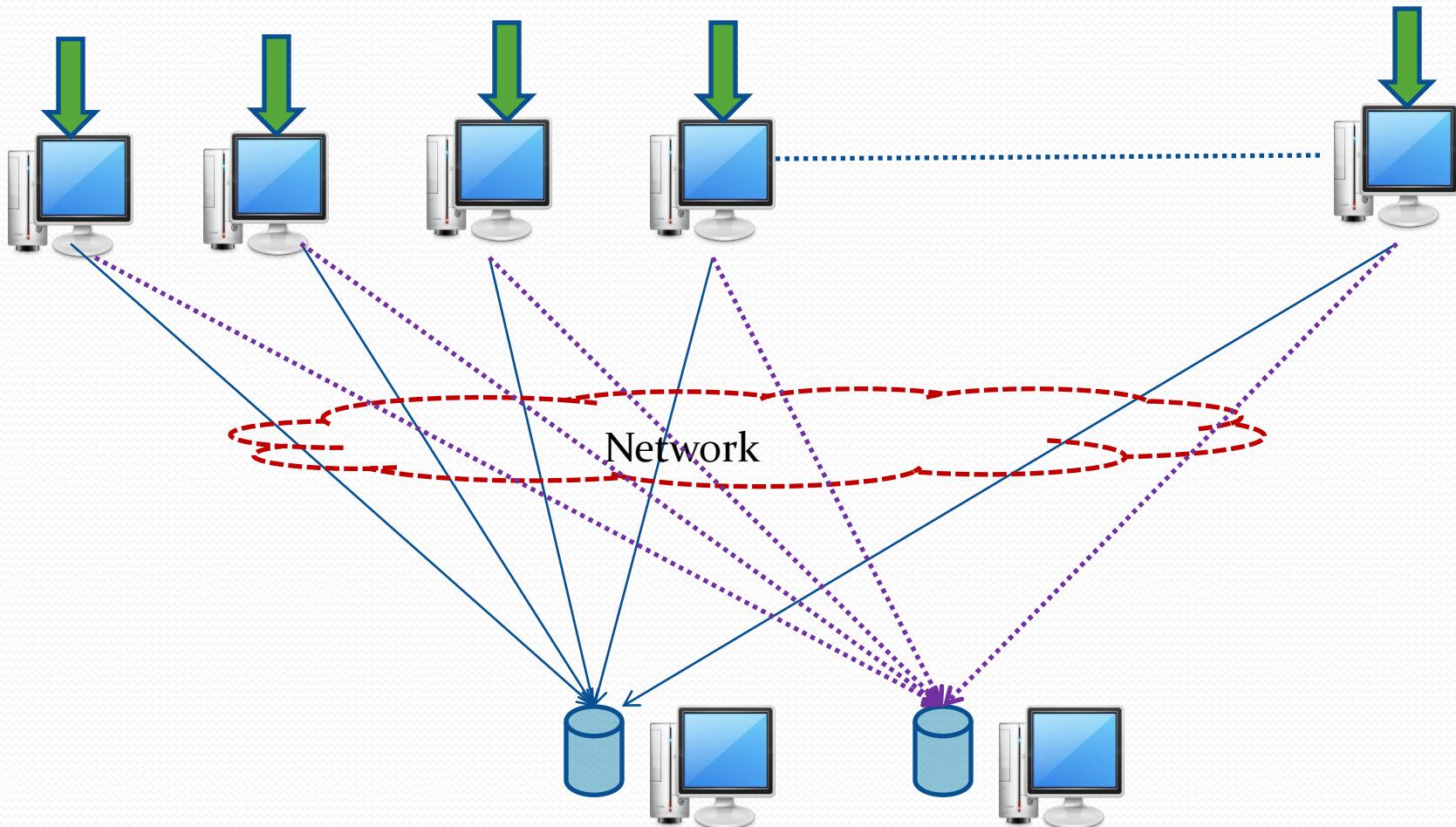
Challenges in map reduce data flow



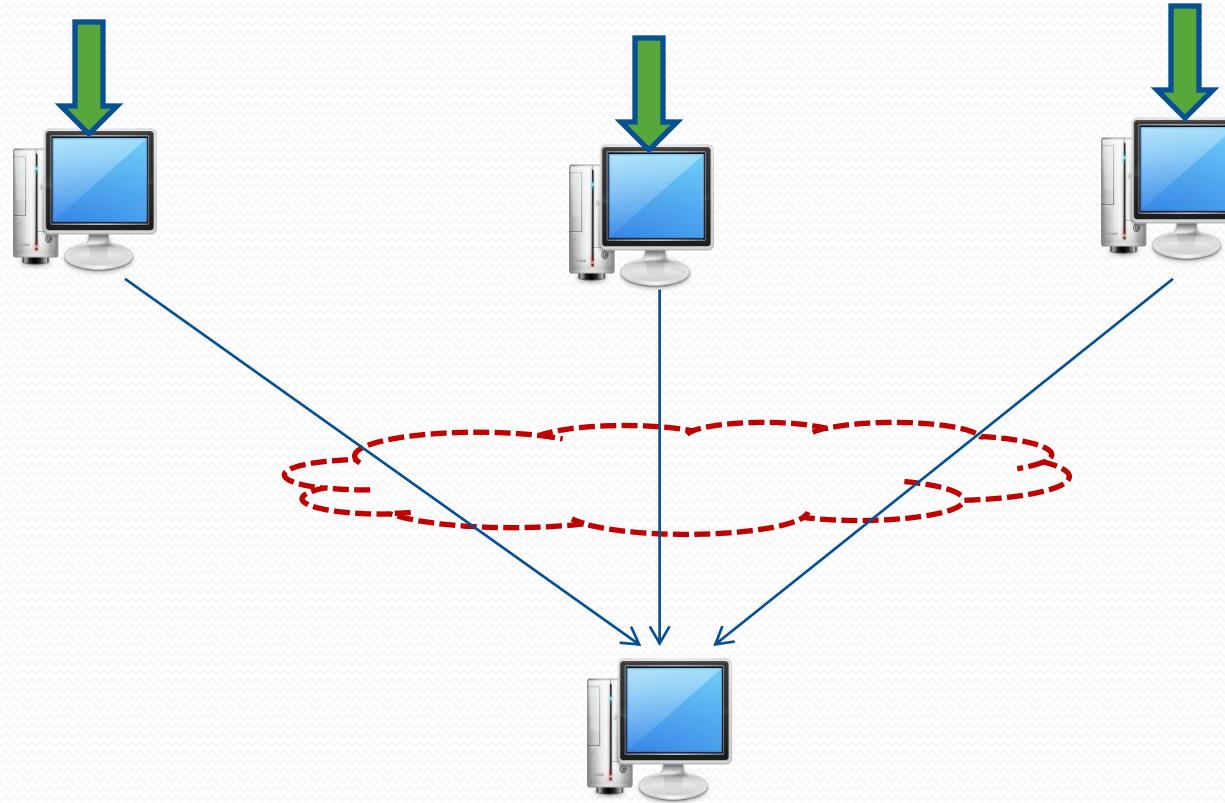
Single Reducer Problem

- All the key-value pairs need to be copied to a single reducer machine
 - Copying will be time consuming
 - Sorting will be time consuming
 - High chances that single reducer machine might not be able to handle that much of data
 - The size of key value pairs that are getting copied might exceed the hard disk capacity of the reducer machine
- Solution: In this case you might want to load balance the key value pairs by running more than 1 reducer

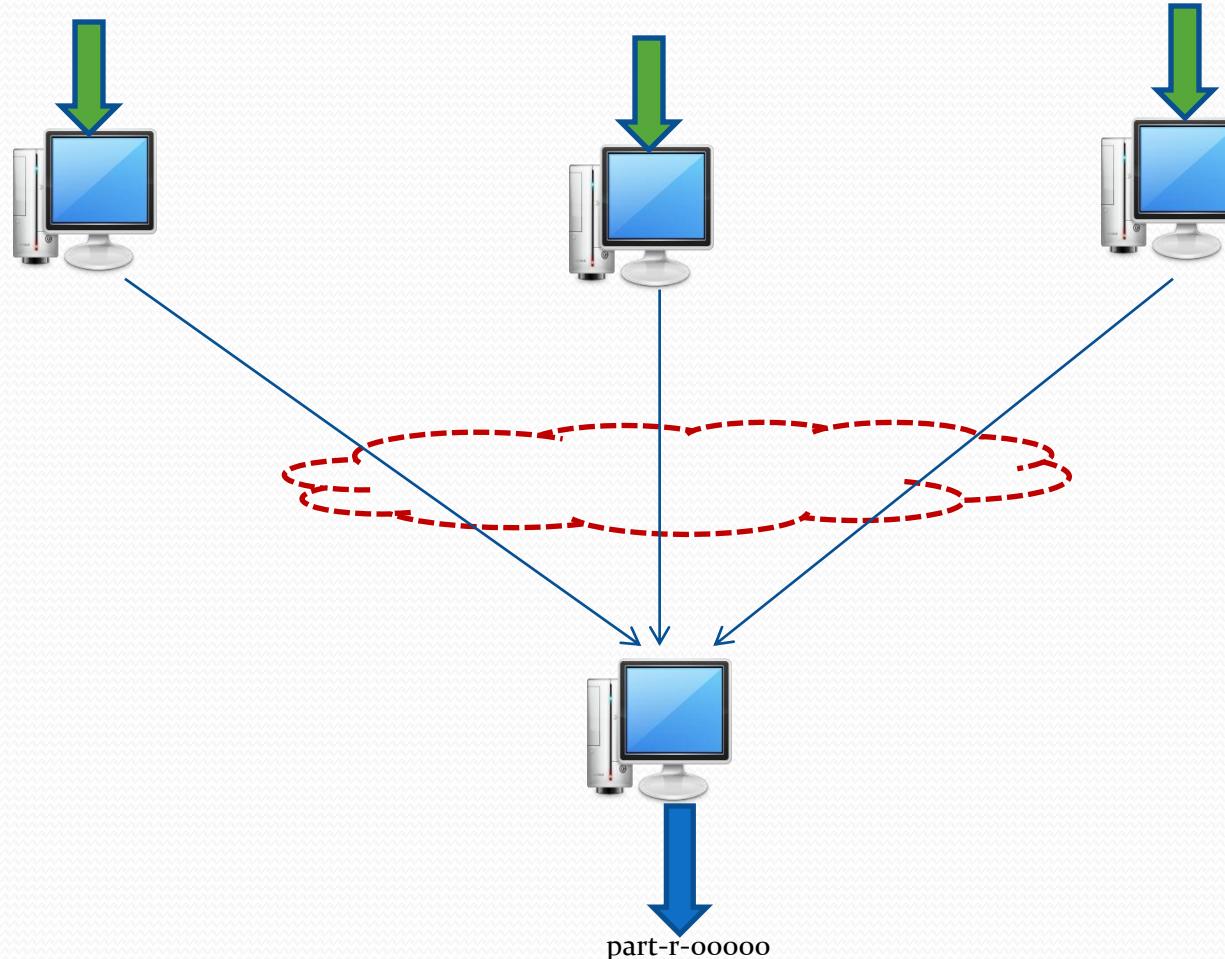
Mitigating Single Reducer Problem



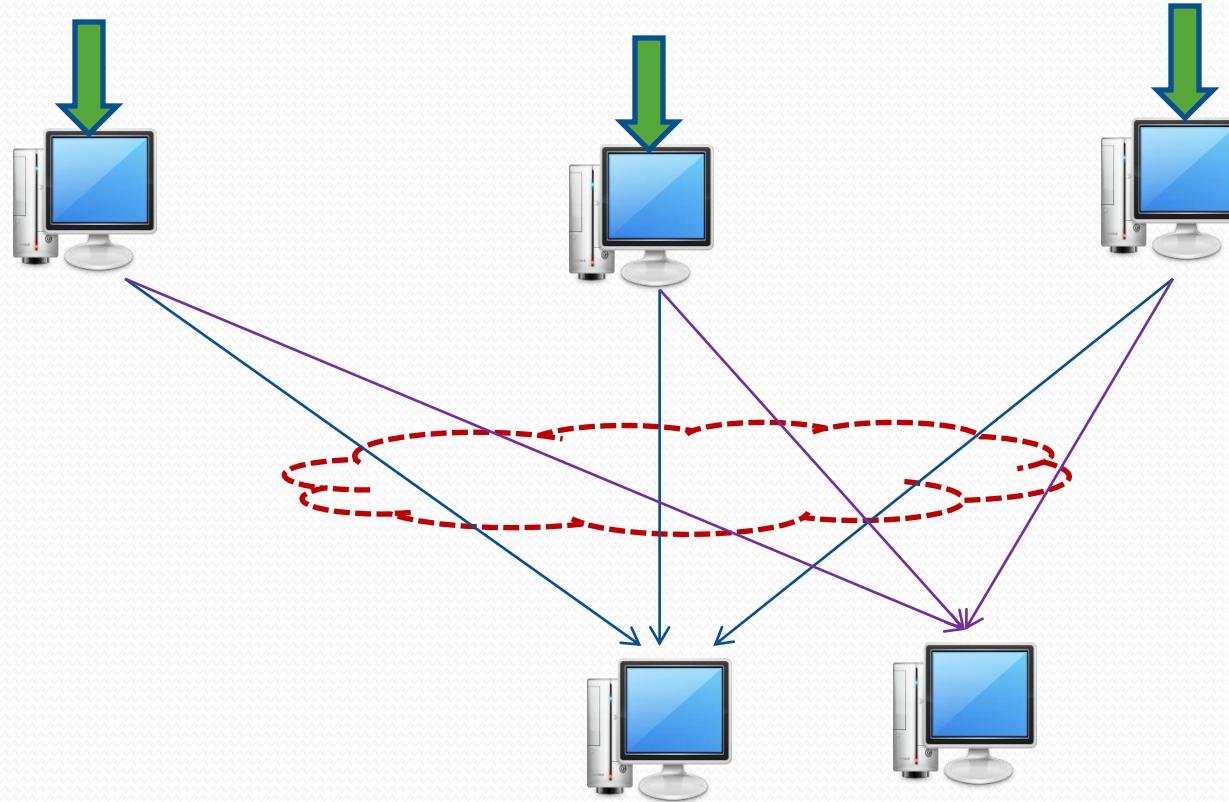
Number of output Files – Single Reducer



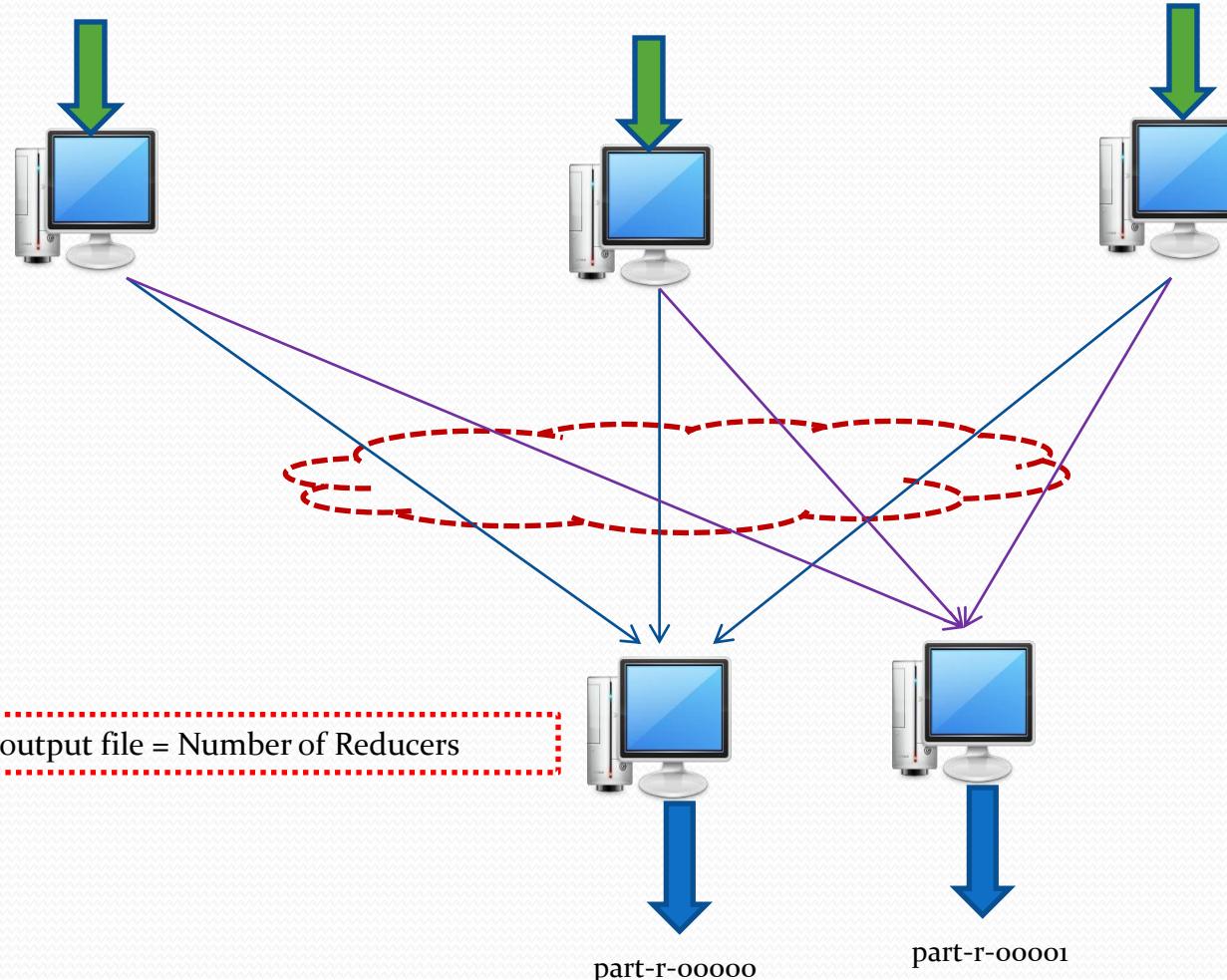
Number of output Files – Single Reducer



Number of output Files – Two Reducer



Number of output Files – Two Reducer



Number of output Files – Zero Reducer



Number of output Files – Zero Reducer



Number of output file = Number of map tasks when the number of reducers are zero

Module 11

MapReduce Framework- Writing Java Code

In this module you will learn

- How will you write mapper class?
- How will you write reducer class?
- How will you write driver class?
- How to use ToolRunner?
- Launching your MapReduce job

Writing Mapper Class

```
public class WordCountMapper extends Mapper<LongWritable, Text, Text,  
IntWritable>{  
  
    @Override  
    public void map(LongWritable inputKey, Text inputVal, Context context)  
    {  
        String line = inputVal.toString();  
        String[] splits = line.split("\\W+");  
        for(String outputKey:splits)      {  
            context.write(new Text(outputKey), new IntWritable(1));  
        }  
    }  
}
```

Writing Mapper Class

```
public class WordCountMapper extends Mapper<LongWritable, Text, Text, IntWritable>{
```

- Your Mapper class should extend from Mapper class
- Mapper<LongWritable, Text, Text, IntWritable>
 - First TypeDef : Input key Type given by input format you use
 - Second TypeDef: Input value Type given by input format
 - Third TypeDef: Output key Type which you emit from mapper
 - Fourth TypeDef: Output value Type which you emit from mapper

Writing Mapper Class

```
public class WordCountMapper extends Mapper<LongWritable, Text, Text,  
IntWritable>{
```

@Override

```
public void map(LongWritable inputKey, Text inputVal, Context context)  
{
```

- Override the map function
- First argument: Input key to your mapper
- Second argument: Input value to your mapper
- Third argument: Using this context object you will emit output key value pair

Writing Mapper Class

```
public class WordCountMapper extends Mapper<LongWritable, Text, Text,  
IntWritable>{  
  
    @Override  
    public void map(LongWritable inputKey, Text value, Context context)  
    {  
        String line = value.toString();  
        String[] splits = line.split("\\W+");  
        for(String outputKey : splits)  {  
            context.write(new Text(outputKey), new IntWritable(1));  
        }  
    }  
}
```

Step 1: Take the String object from the input value

Step 2: Splitting the string object obtained in step 1, split them into individual words and take them in array

Step 3: Iterate through each words in the array and emit individual word as key and emit value as 1, which is of type IntWritable

Writing Reducer Class

```
public class WordCountReducer extend Reducer <Text, IntWritable, Text,  
IntWritable > {  
  
    public void reduce(Text key, Iterable<IntWritable> values, Context output)  
        throws IOException, InterruptedException {  
  
        int sum = 0;  
        for (IntWritable val : values) {  
            sum += val.get();  
        }  
  
        output.write(key, new IntWritable(sum));  
    }  
}
```

Writing Reducer Class

```
public class WordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
```

- Your Reducer class should extend from Reducer class
- Reducer<Text,IntWritable,Text,IntWritable>
 - First TypeDef : Input key Type given by the output key of map output
 - Second TypeDef: Input value Type given by output value of map output
 - Third TypeDef: Output key Type which you emit from reducer
 - Fourth TypeDef: Output value Type which you emit from reducer

Writing Reducer Class

```
public class WordCountReducer extends Reducer<Text, IntWritable, Text,  
IntWritable> {  
  
    public void reduce(Text key, Iterable<IntWritable> values, Context output)  
        throws IOException, InterruptedException {  
  
        • Reducer will get key and list of values  
  
            • Example: Hello {1,1,1,1,1,1,1,1,1}  
  
    }  
}
```

Writing Reducer Class

```
public class WordCountReducer extend Reducer <Text, IntWritable, Text,  
IntWritable > {
```

```
    public void reduce(Text key, Iterable<IntWritable> values, Context output)  
        throws IOException, InterruptedException {
```

```
        int sum = 0;  
        for (IntWritable val : values) {  
            sum += val.get();  
        }
```

- Iterate through list of values and add the values

```
}
```

Writing Driver Class

- Step1 :Get the configuration object, which tells you where the namenode and job tracker are running
- Step2 :Create the job object
- Step3: Specify the input format. by default it takes the TextInputFormat
- Step4: Set Mapper and Reducer class
- Step5:Specify the mapper o/p key and o/pvalue class
i/p key and value to mapper is determined by the input format. So NO need to specify
- Step6: Specify the reducer o/p key and o/p value class
i/p key and value to reducer is determined by the map o/p key and map o/p value class respectively. So NO need to specify
- Step7: Provide the input and output paths
- Step8: Submit the job

Driver Class cont'd

```
Job job = new Job(getConf(),"Basic Word Count Job");
job.setJarByClass(WordCountDriver.class);

job.setMapperClass(WordCountMapper.class);
job.setReducerClass(WordCountReducer.class);

job.setInputFormatClass(TextInputFormat.class);

job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(IntWritable.class);
```

Driver Class cont'd

```
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));

job.waitForCompletion(true);
```

Launching the MapReduce Job

- Create the jar file
 - Either from eclipse or from command line
- Run the following command to launch

```
hadoop jar <jarName.jar> <DriverClassName> <input> <output>
```

- <input> could be a file or directory consisting of files on HDFS
- <output> Name should be different for every run.
- If the <output> directory with the same name is present, exception will be thrown
 - It's a directory which will be created on HDFS.
 - Output file name :part-r-ooooo OR part-m-ooooo
 - Demo for map only jobs showing intermediate output

Hands On

- Refer the hands-on document

Older and Newer API

Old API	New API
<pre>import org.apache.hadoop.mapred.*</pre> <p>Driver code:</p> <pre>JobConf conf = new JobConf(conf, Driver.class); conf.setSomeProperty(...); ... JobClient .runJob(conf);</pre>	<pre>import org.apache.hadoop.mapreduce.*</pre> <p>Driver code:</p> <pre>Configuration conf = new Configuration(); Job job = new Job(conf); job.setJarByClass(Driver.class); job.setSomeProperty(...); ... job.waitForCompletion(true);</pre>
<p>Mapper:</p> <pre>public class MyMapper extends MapReduceBase implements Mapper { public void map(Keytype k, Valuetype v, OutputCollector o, Reporter r) { ... o.collect(key, val); } }</pre>	<p>Mapper:</p> <pre>public class MyMapper extends Mapper { public void map(Keytype k, Valuetype v, Context c) { ... c.write(key, val); } }</pre>

Notes

- All the programs in this training uses Newer API

Module 12

Use Cases

In this module you will learn

- Word Co-Occurrence problem
- Average Word Length Problem
- Inverted Index problem
- Searching
- Sorting
- Hands on

Word Co-Occurrence Problem

- Measuring the frequency with which two words appearing in a set of documents
- Used for recommendation like
 - “*You might like this also*”
 - “*People who choose this also choose that*”
 - *Examples:*
 - Shopping recommendations
 - Identifying people of interest
- Similar to word count but two words at a time

Inverted Index Problem

- Used for faster look up
 - Example: Indexing done for keywords in a book
- Problem statement:
 - From a list of files or documents map the words to the list of files in which it has appeared
- Output
 - *word* = List of documents in which this *word* has appeared

Indexing Problem cont'd

File A

This is cat
Big fat hen

File B

This is dog
My dog is fat

Output from
the mapper

This: File A
is:File A
cat:File A
Big:File A
fat:File A
hen:File A

Final Output

This:File A,File B
is:File A,File B
cat:File A
fat: File A,File B

This:File B
is:File B
dog:File B
My: File B
dog:File B
is:File B
fat: File B

Indexing problem cont'd

- Mapper

For each word in the line, $\text{emit}(\text{word}, \text{file_name})$

- Reducer

- Remember for word , all the file names list will be coming to the reducer
- $\text{Emit}(\text{word}, \text{file_name_list})$

Average Word Length Problem

- Consider the record in a file

Hey How is Henry these days?

- Problem Statement
 - Calculate the average word length for each character
- Output:

Character	Average word length
H	$(3+3+5)/3 = 3.66$
I	$2/1 = 2$
T	$5/1 = 5$
D	$4/1 = 4$

Average Word Length cont'd

- Mapper
 - For each word in a line,
emit(firstCharacterOfWord,lengthOfWord)
- Reducer
 - You will get a character as key and list of values corresponding to length
 - For each value in listOfValue
 - Calculate the sum and also count the number of values
 - *Emit(character,sum/count)*

Hands On

- Refer hands-on document

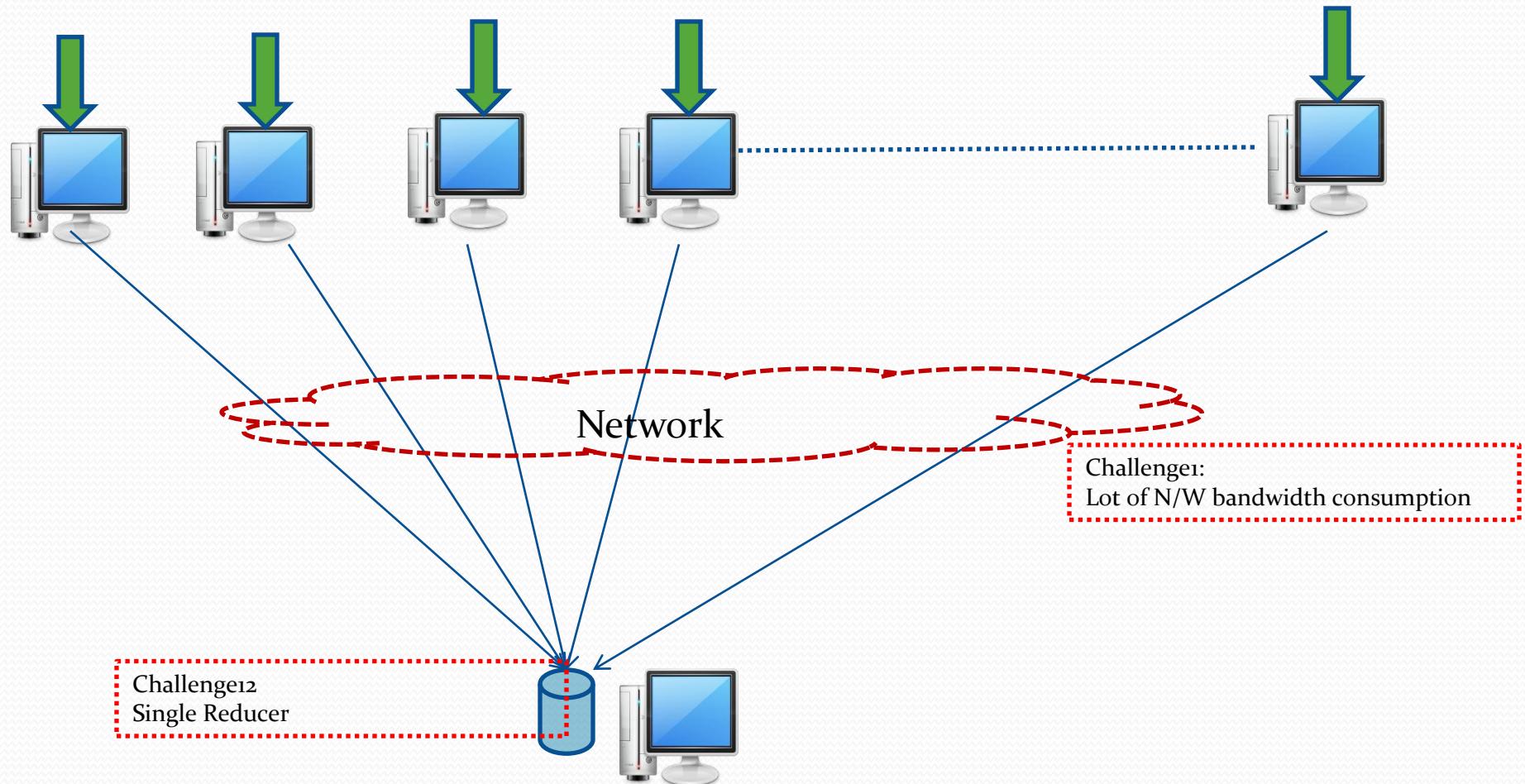
Module 13

Advance MapReduce Concepts – Part 1

In this module you will learn

- What is combiner and how does it work?
- How Partitioner works?
- Local Runner and Tool Runner
- setup/cleanup method in mapper / reducer
- Passing the parameters to mapper and reducer
- Distributed cache
- Counters
- Hands On

Motivation



Challenge1: N/W band width consumption

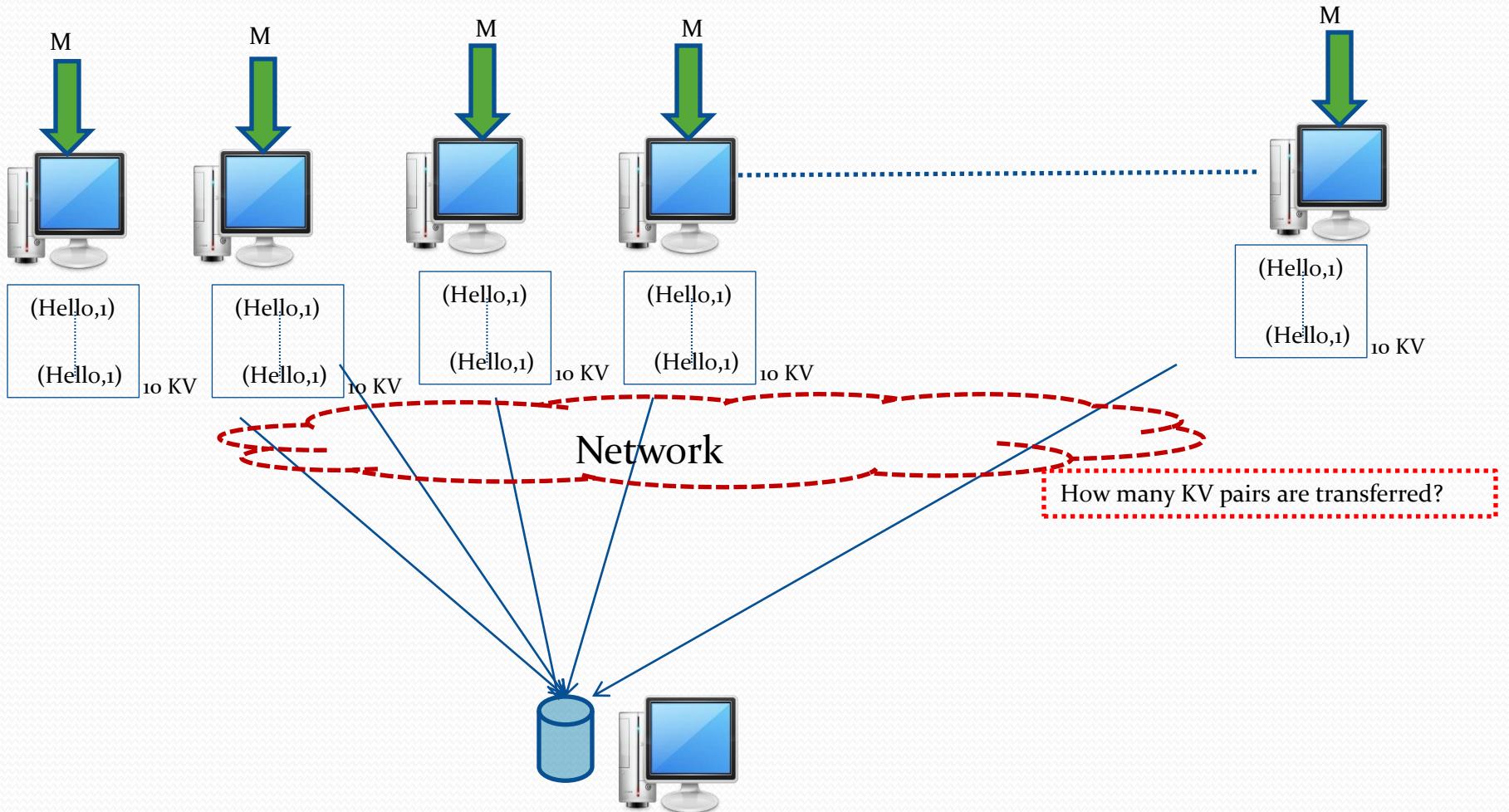
What is the source of n/w band width consumption?

Challenge1: N/W band width consumption

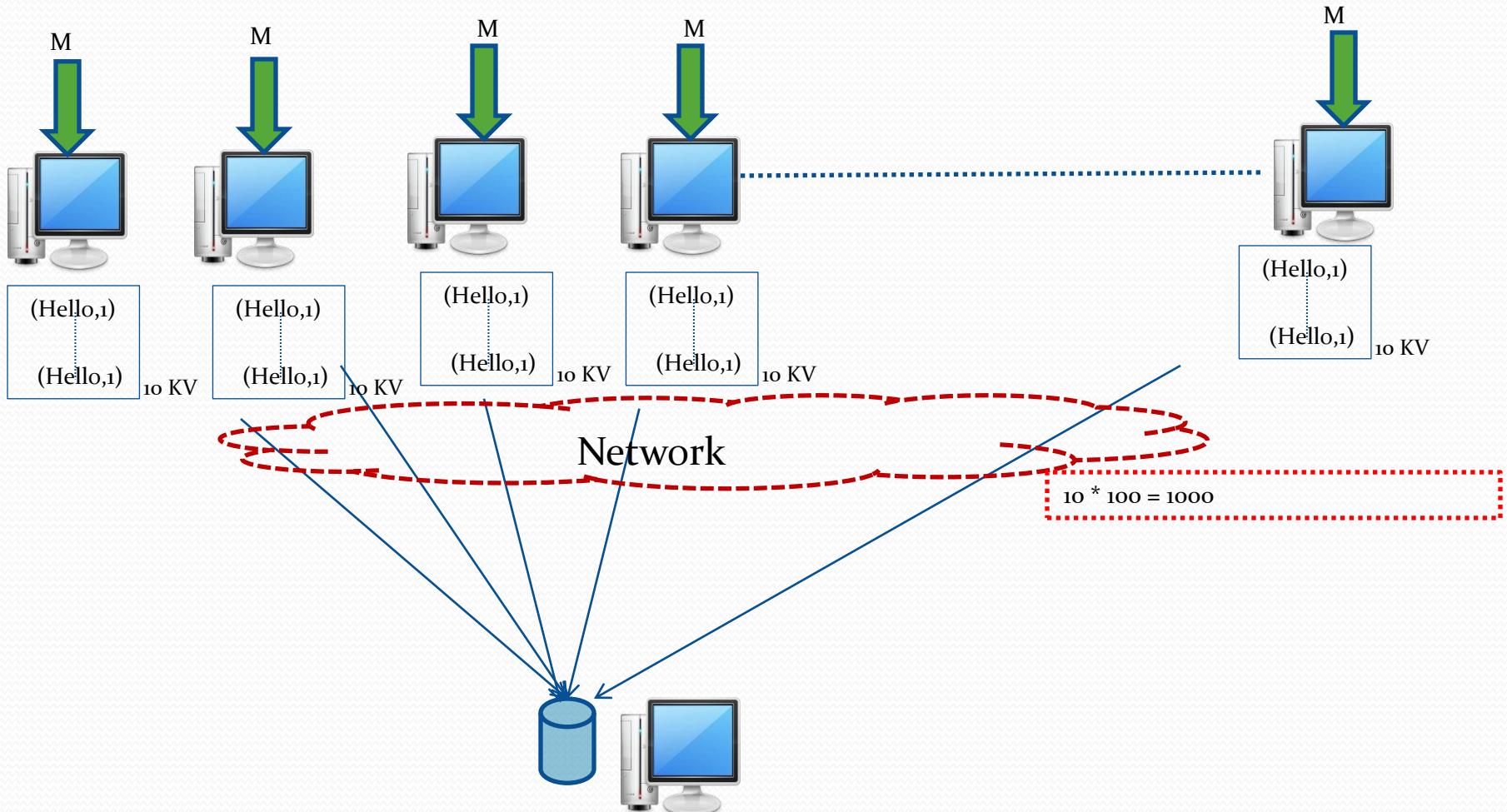
What is the source of n/w band width consumption?

Lot of key value pairs that are transferred across the network

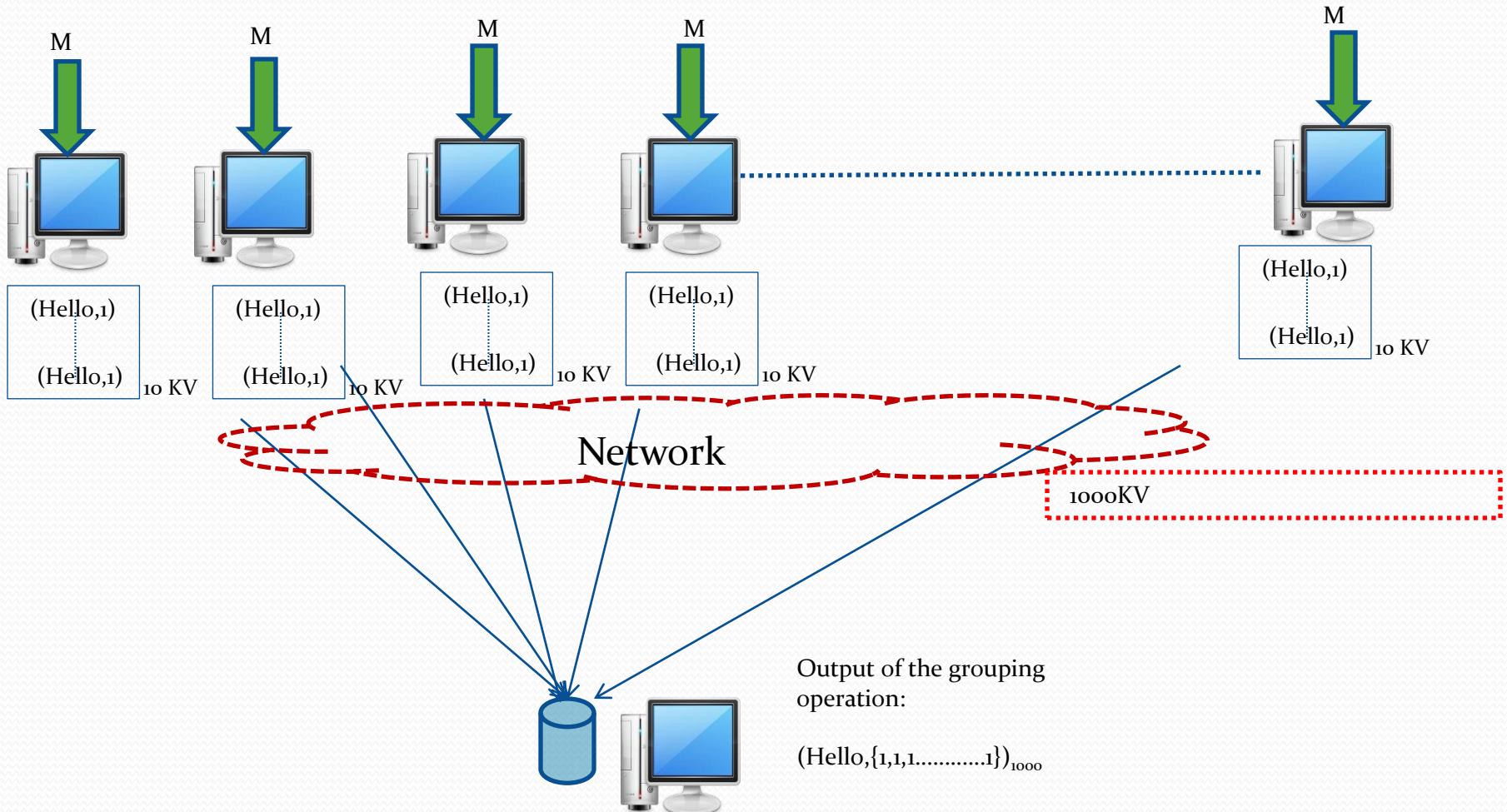
Challenge1: N/W band width consumption



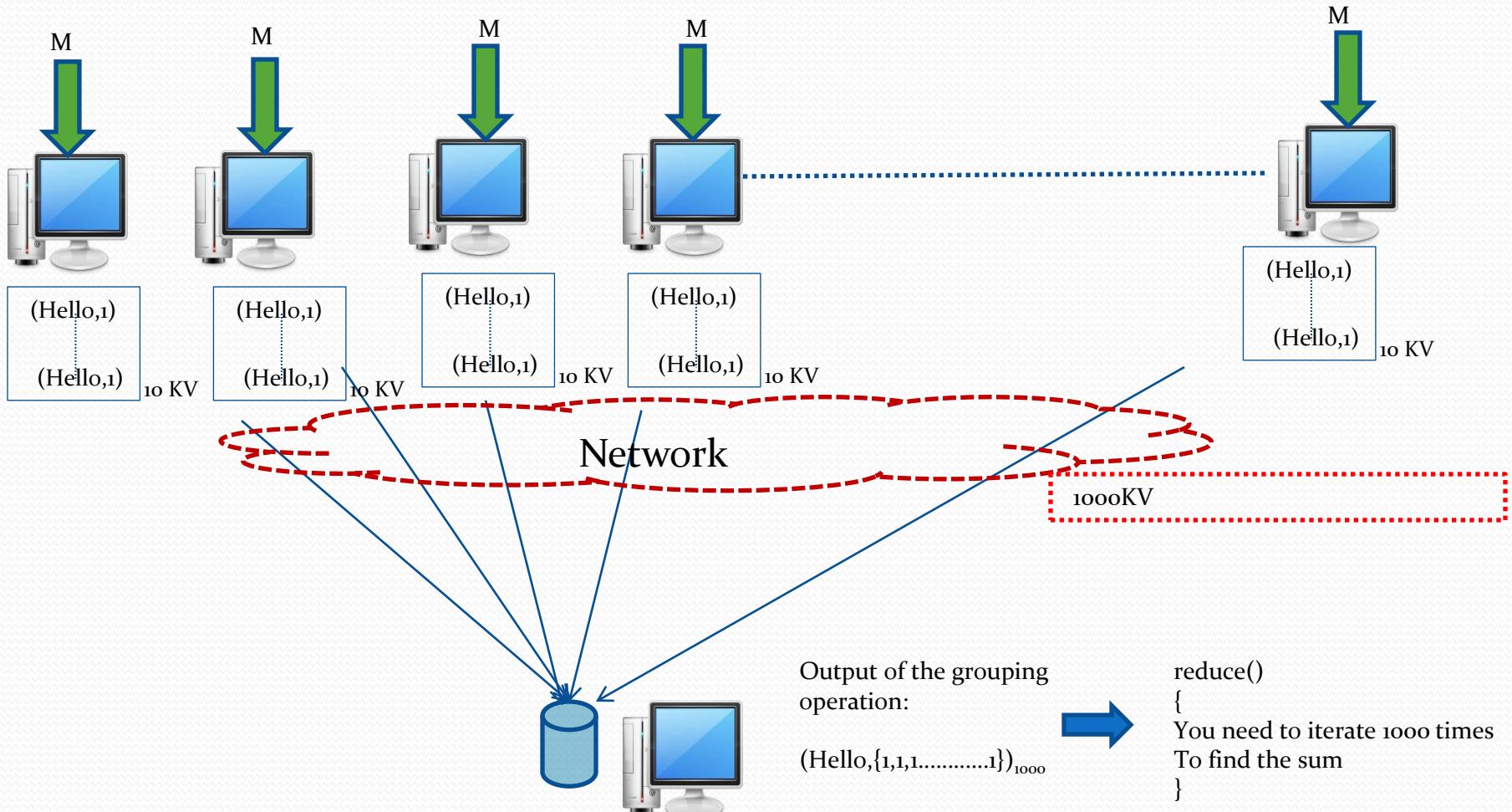
Challenge1: N/W band width consumption



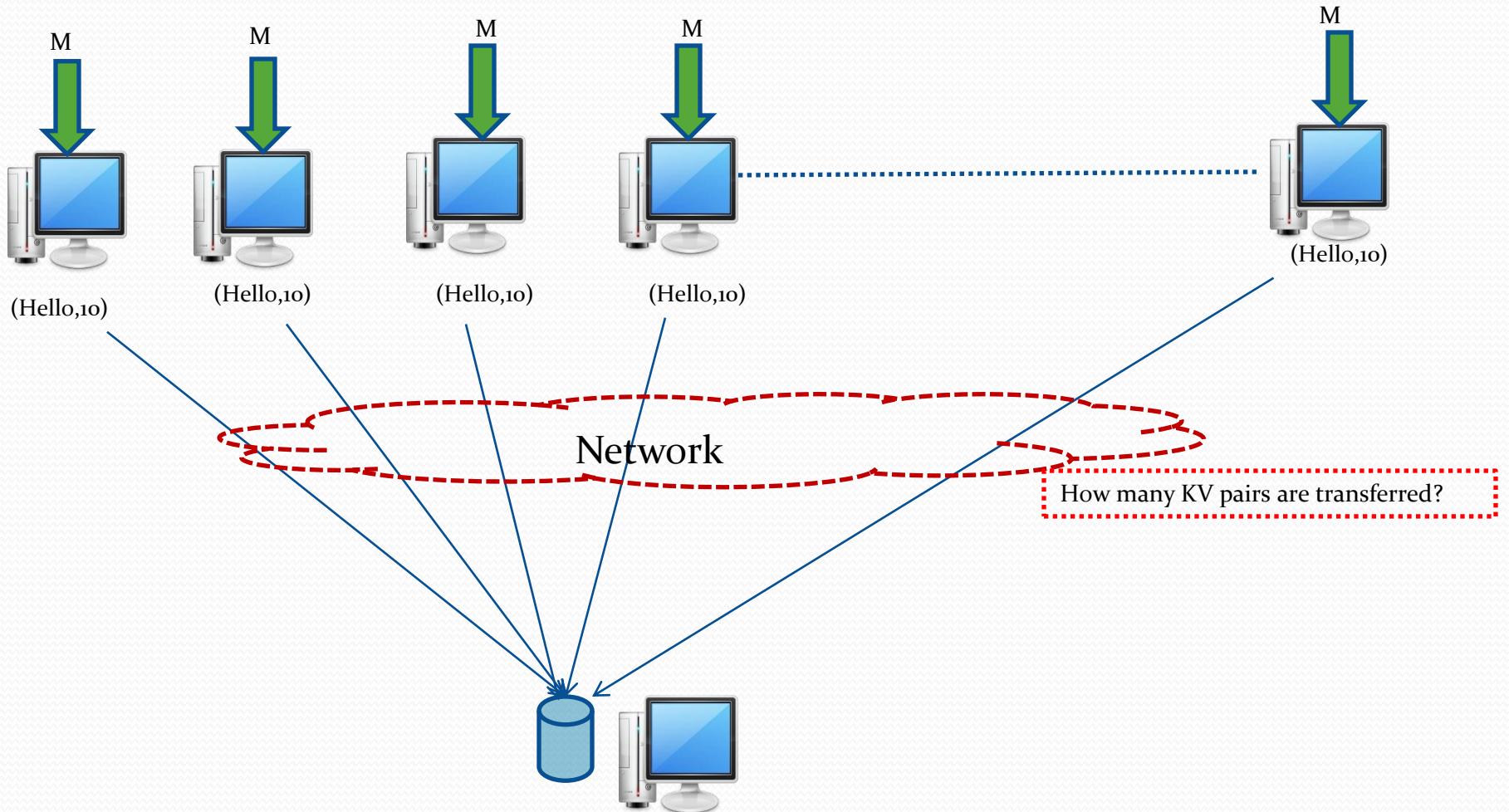
Challenge1: N/W band width consumption



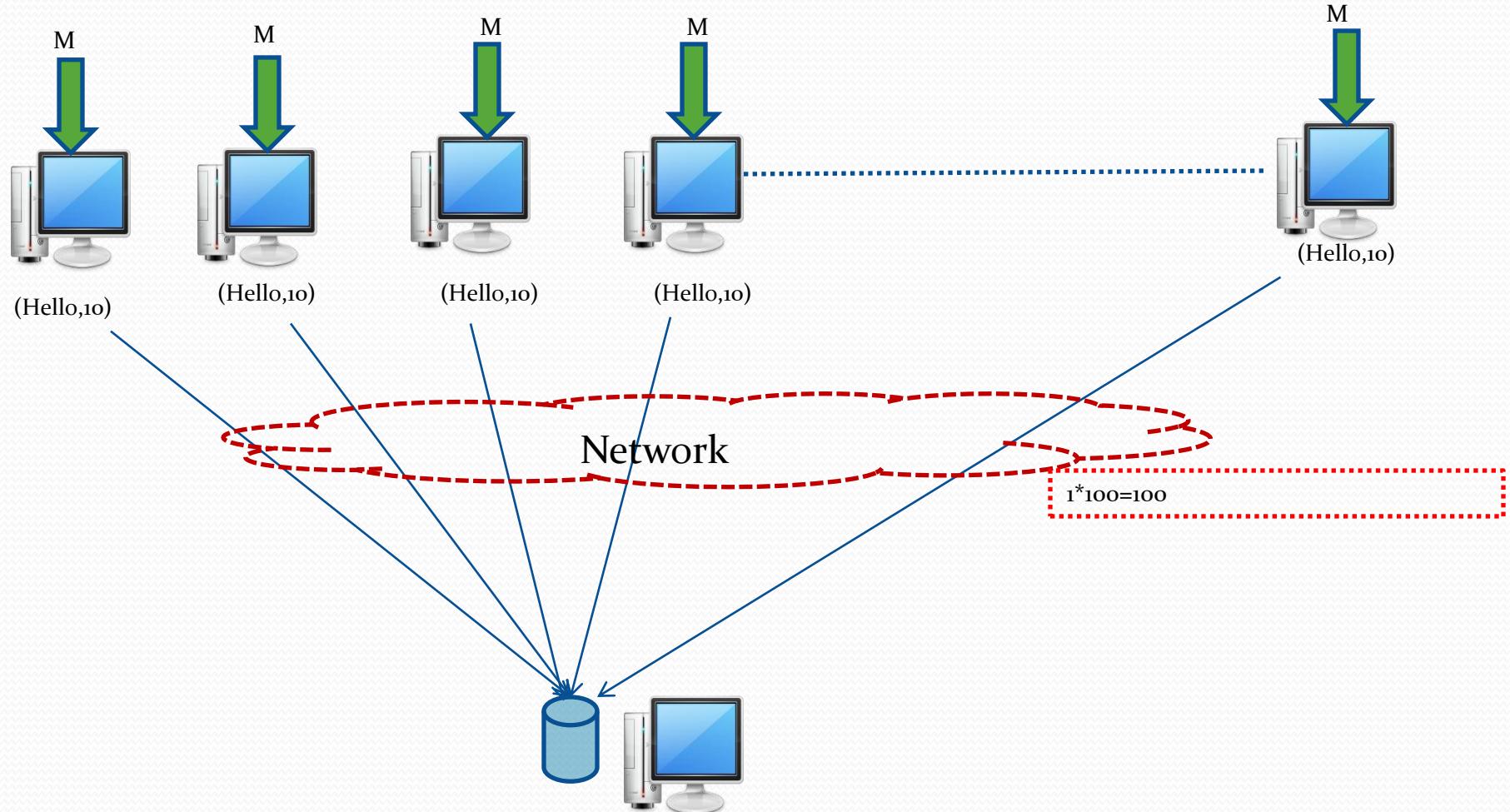
Challenge1: N/W band width consumption



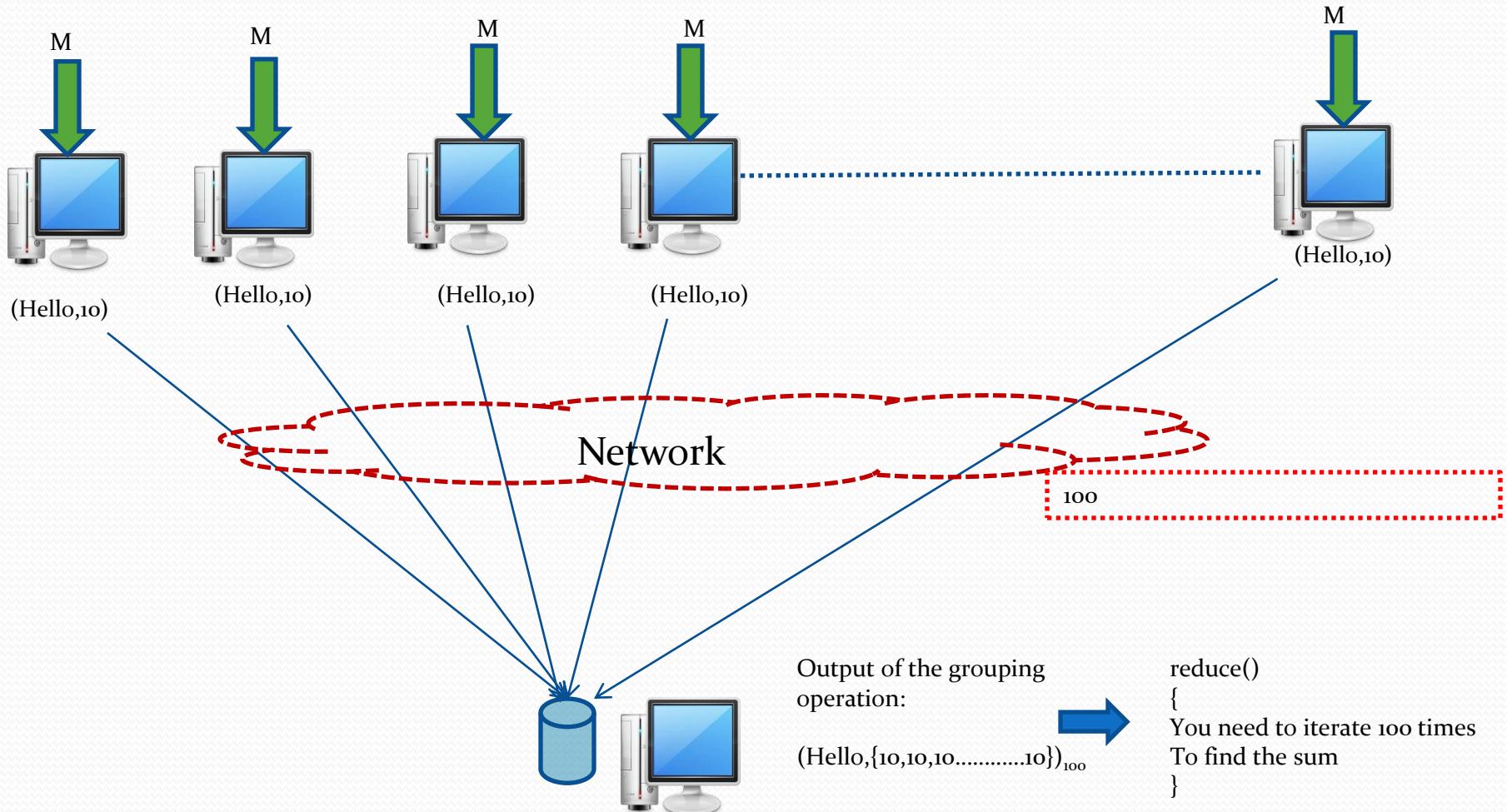
Challenge1: N/W band width consumption



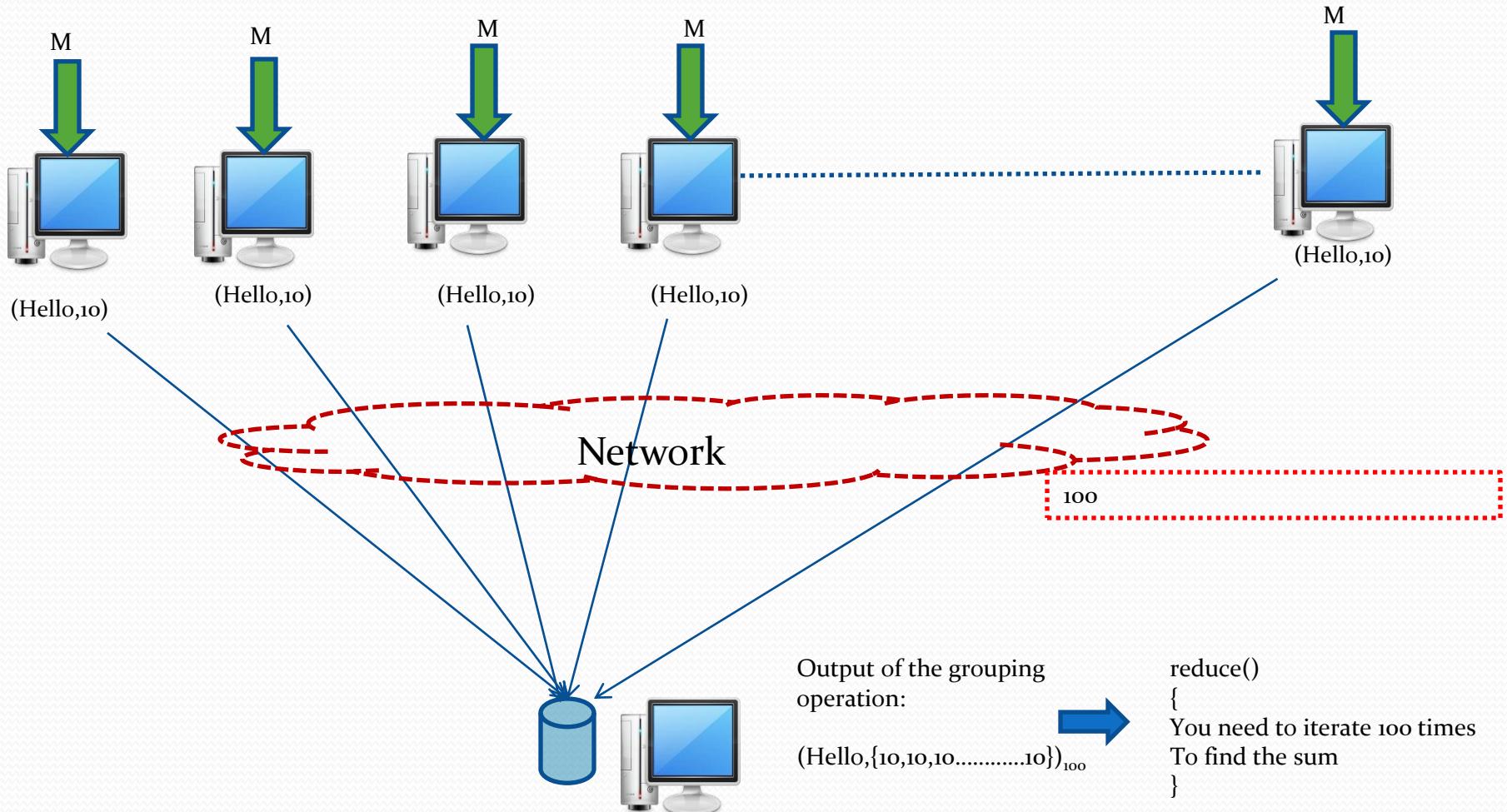
Challenge1: N/W band width consumption



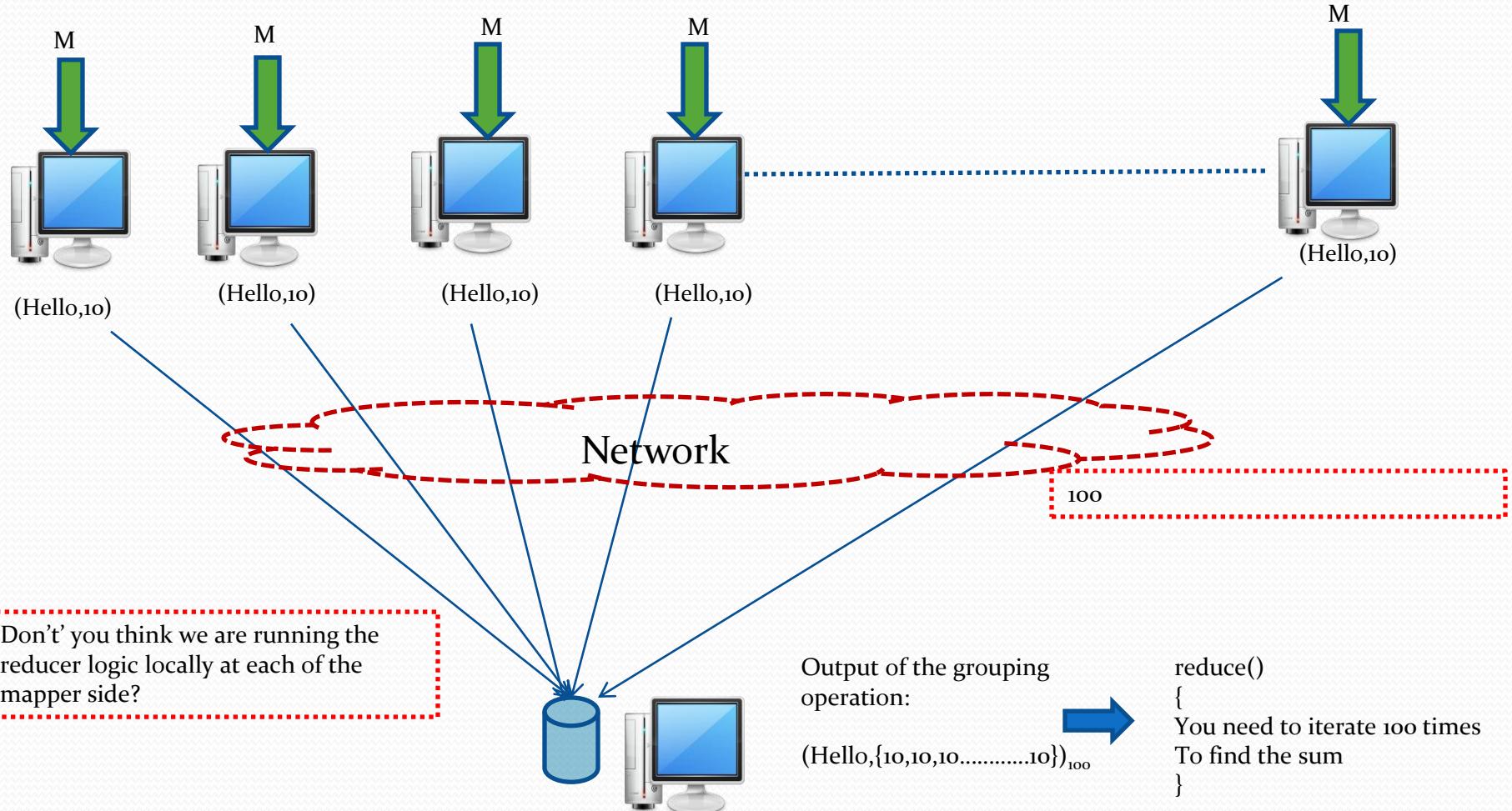
Challenge1: N/W band width consumption



Challenge1: N/W band width consumption



Challenge1: N/W band width consumption



Combiner cont'd

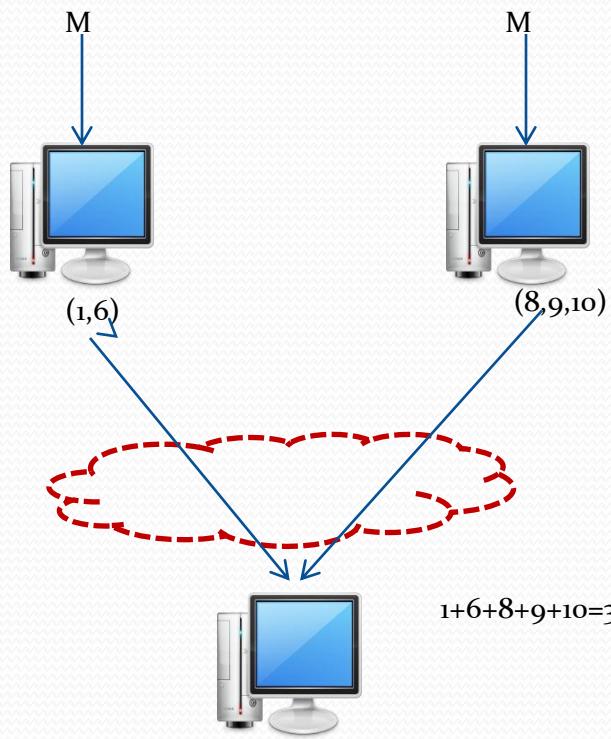
- Known as local reducer
 - Runs on the same machine as the mapper task
 - Runs the reducer code on the intermediate output of the mapper
 - Thus minimizing the intermediate key-value pair
- Advantages
 - Minimize the data transfer across the network
 - Local aggregation is done on the mapper side and thus minimizing the key value pairs
 - Speed up the execution
 - Since less key value pairs are generated and transferred
 - Reduces the burden on reducer
 - Less number of iteration needs to be done in the reduce function

Combiner cont'd

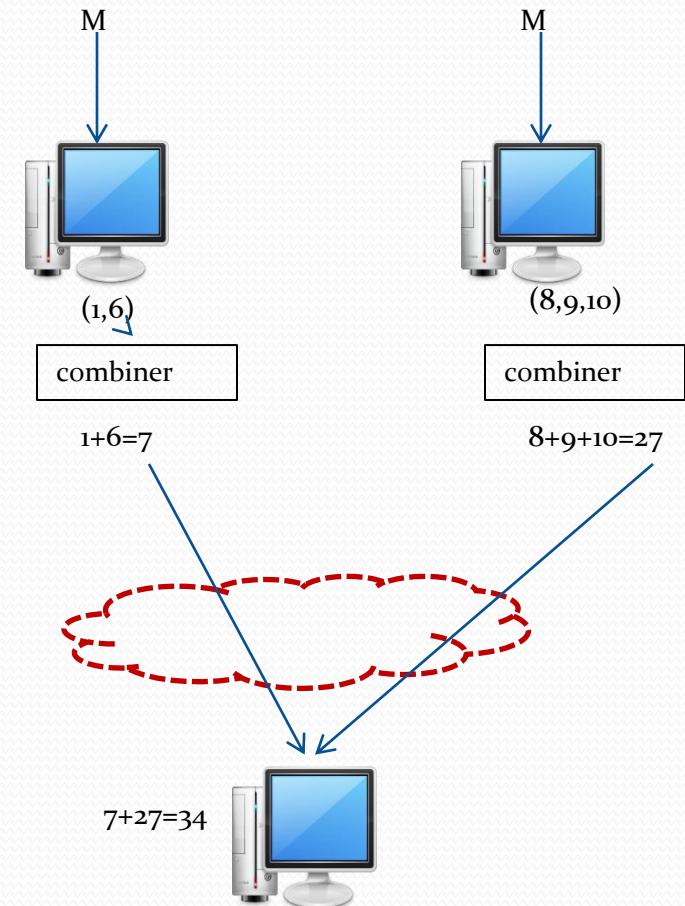
- Generally if you want to write your combiner logic you need to extend your class from Reducer class and override reduce method
- Generally you will make your existing reducer to run as combiner
 - The operation is associative or commutative in nature
 - Example: Sum, Multiplication, min, max
 - Average operation cannot be used

Reducer is performing sum operation

Without Combiner

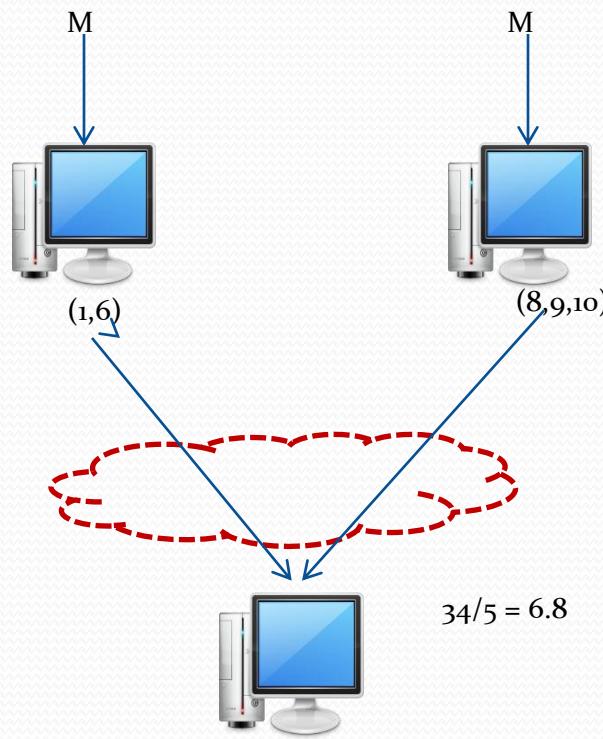


With Combiner

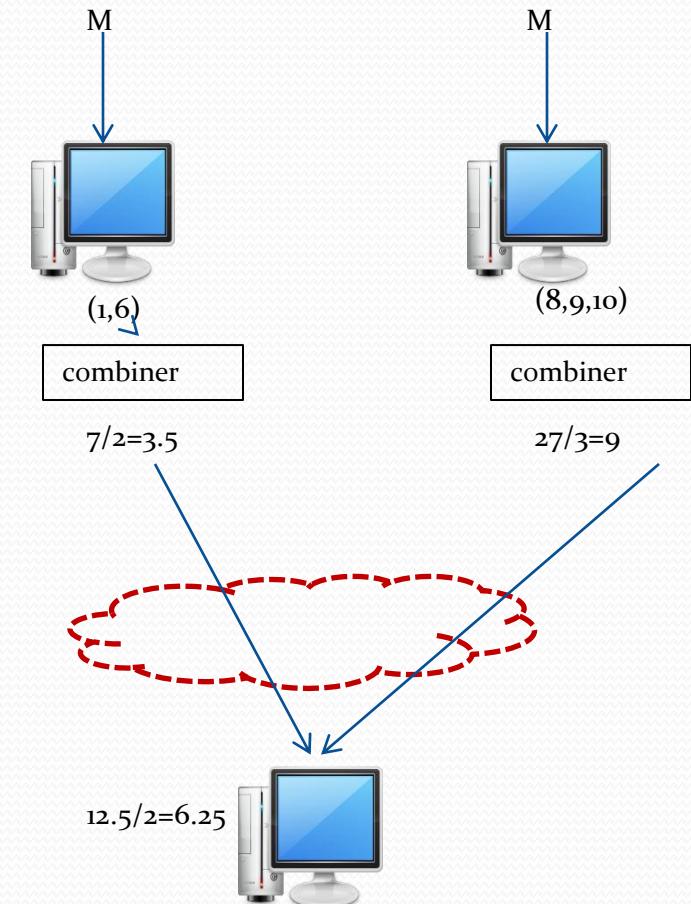


Reducer is performing Average operation

Without Combiner



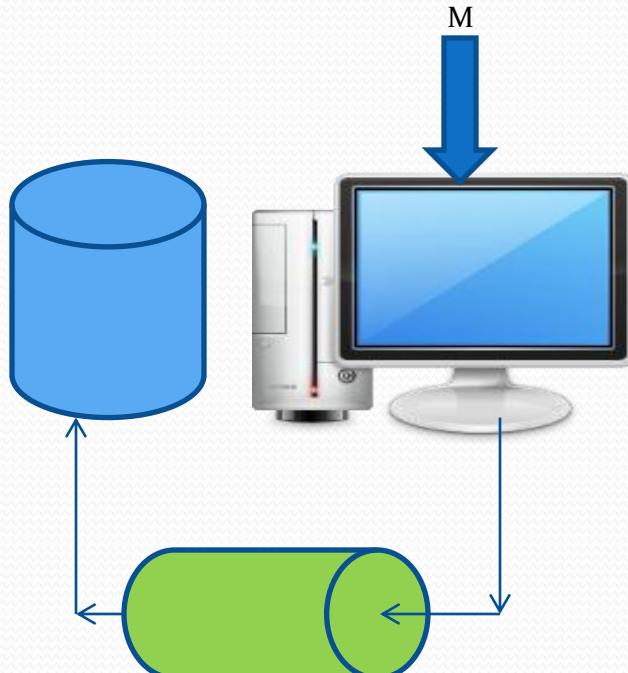
With Combiner



Combiner cont'd

- In the driver class specify
 - `Job.setCombinerClass(<ReducerClassName>.class);`
- NOTE:
 - On a single mapper task, combiner may run, may not run, and it might run more than once also
 - Depends on two properties
 - `io.sort.mb=100MB` (Specifies the size of circular buffer memory)
 - `io.sort.factor=10` (Specify number of files to be merged)

MR data flow

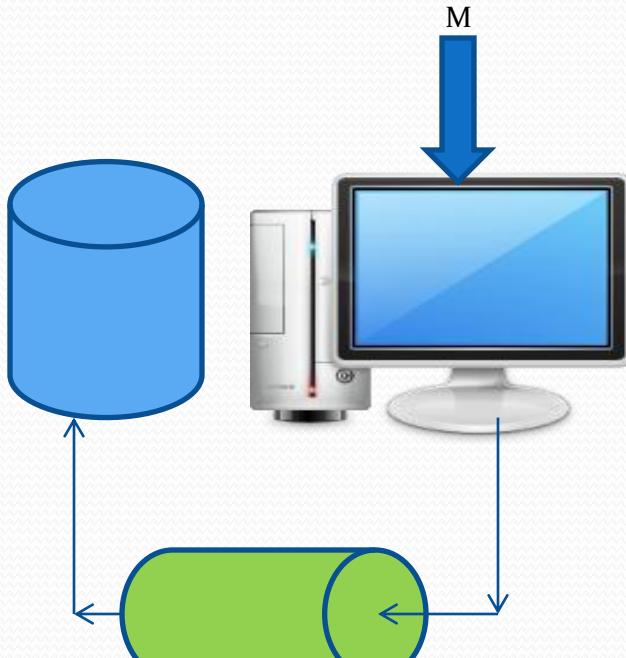


Circular buffer memory=100MB

IKV are continuously generated and are kept in circular buffer memory.

If the IKV size exceeds the circular buffer memory, then they are spilled on to the local file system

Scenario 1: Combiner not running at all



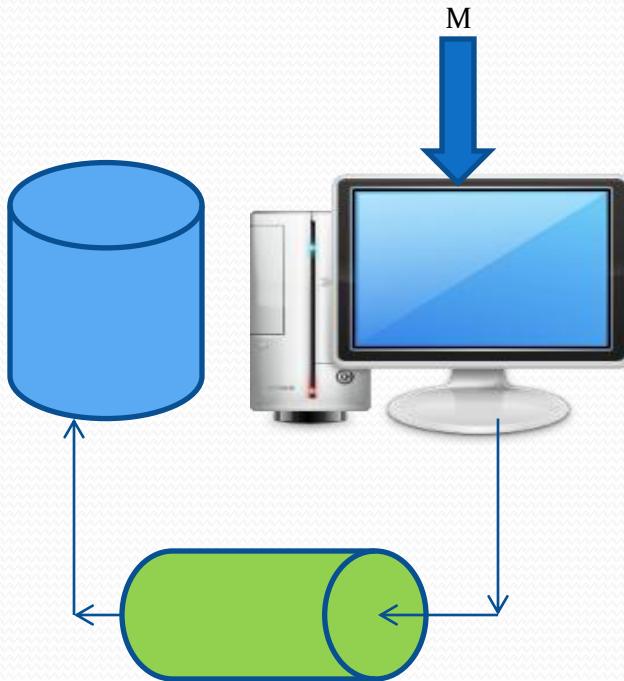
Assumption:

If (Hello,1) is emitted 10 times, then it has exceeded the circular buffer memory

- 5 (Hello,1) are generated
- IKV that are generated does not exceed the circular buffer memory.
- Combiner will not run at all
- So 5 KV pairs will be transmitted

Circular buffer memory=100MB

Scenario 2: Combiner running only once



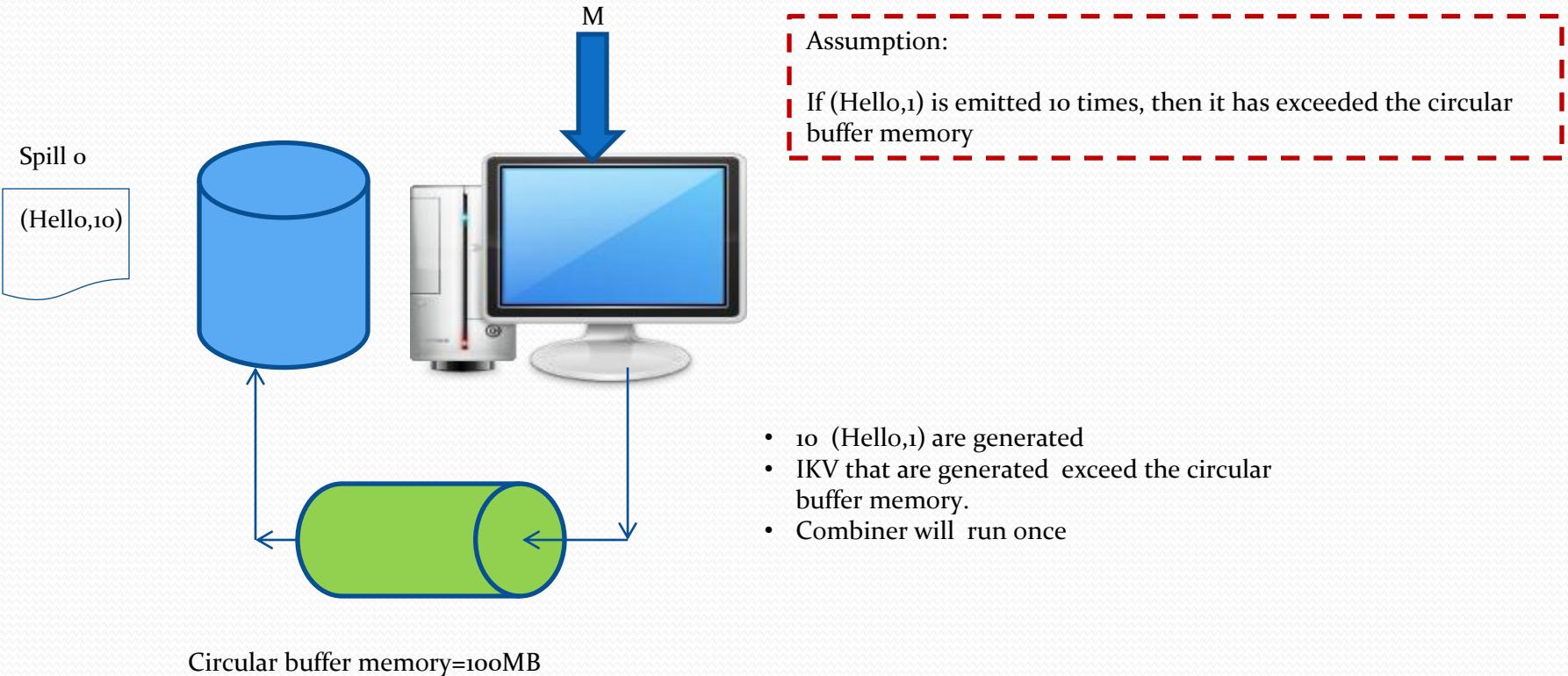
Circular buffer memory=100MB

Assumption:

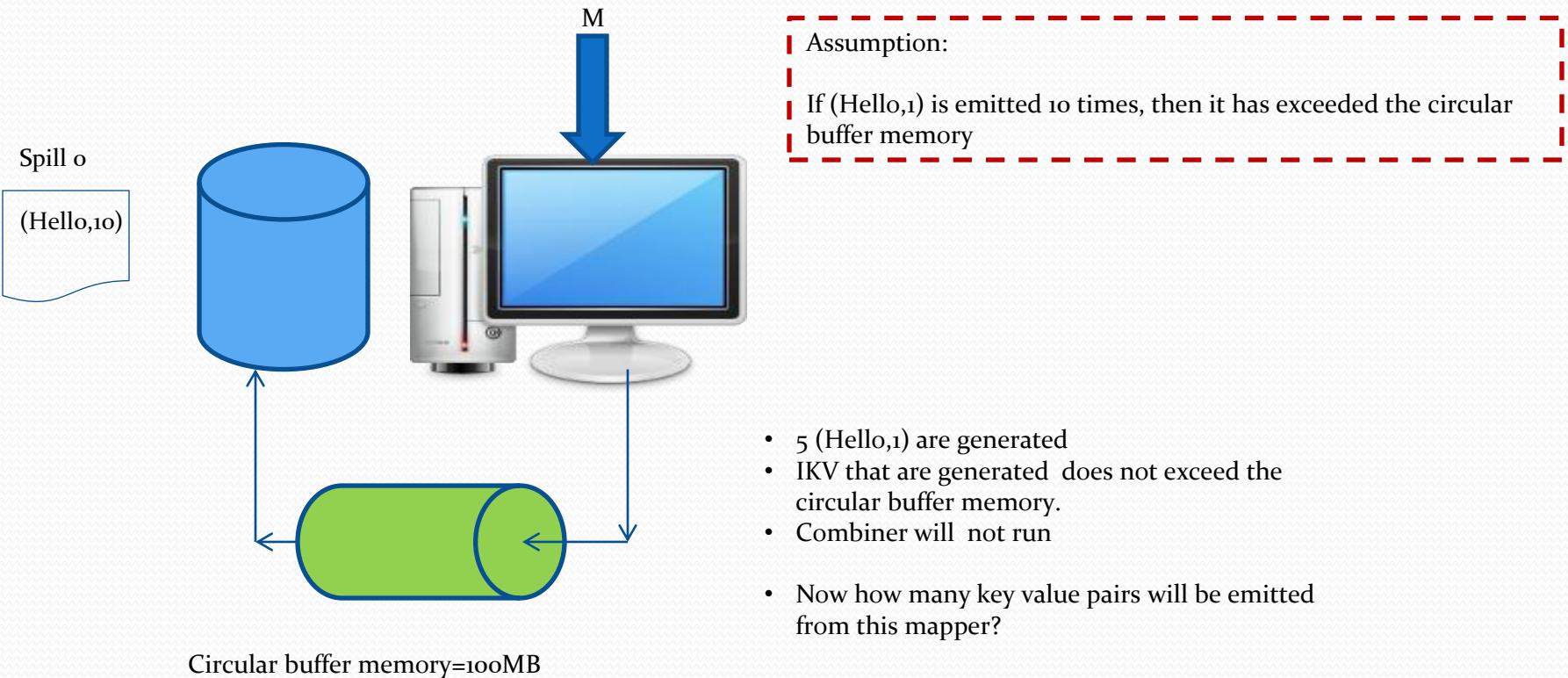
If (Hello,1) is emitted 10 times, then it has exceeded the circular buffer memory

- 10 (Hello,1) are generated
- IKV that are generated exceed the circular buffer memory.
- Combiner will run once

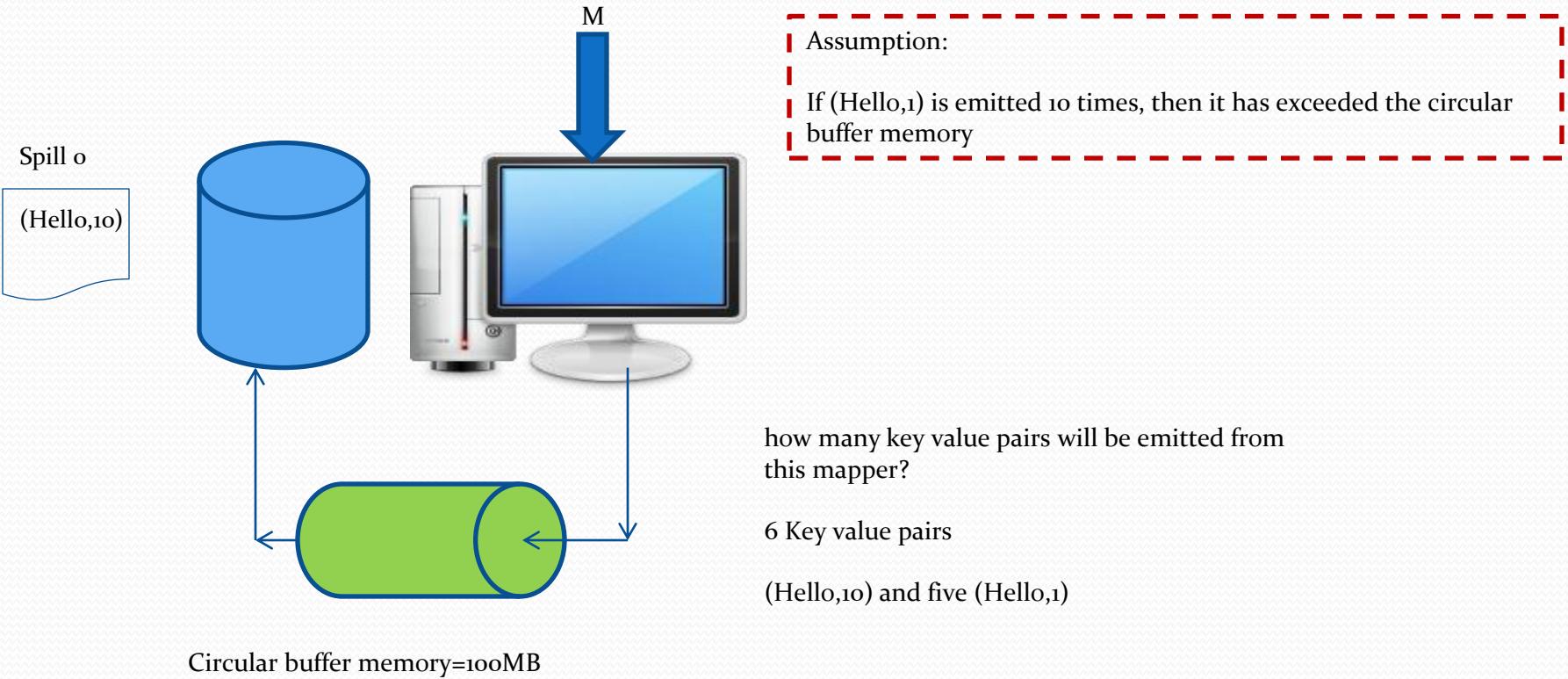
Scenario 2: Combiner running only once



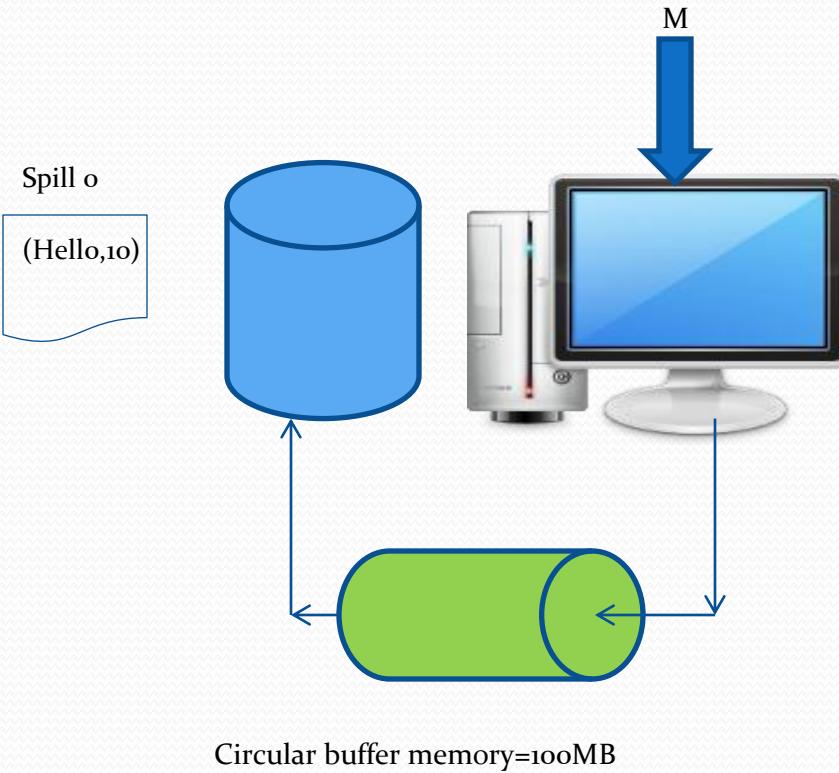
Scenario 2: Combiner running only once



Scenario 2: Combiner running only once



Scenario 2: Combiner running more than once



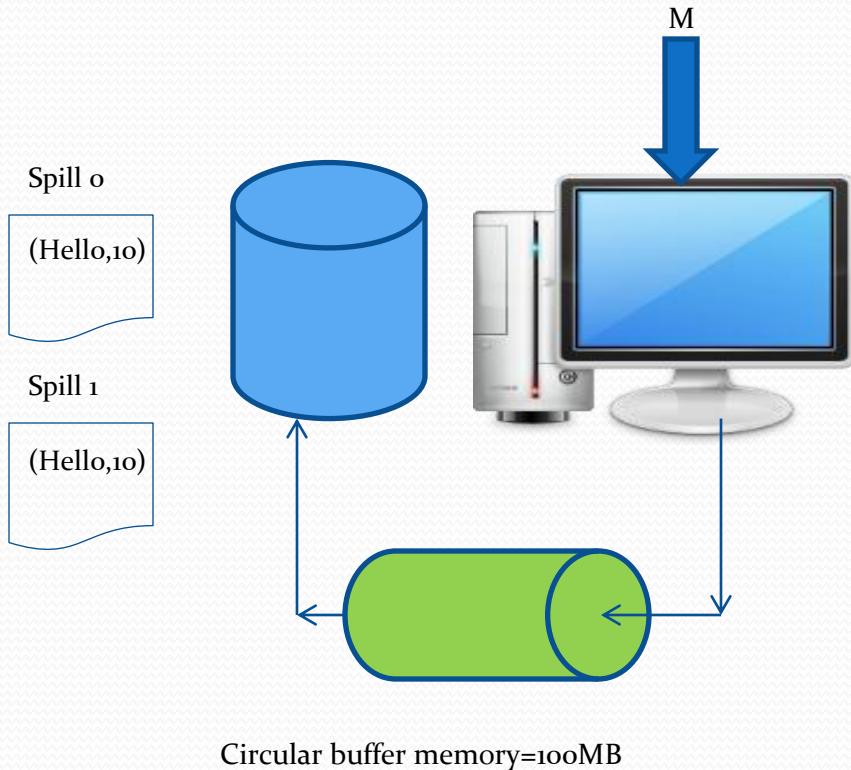
Assumption:

If (Hello,1) is emitted 10 times, then it has exceeded the circular buffer memory

Assuming combiner already ran once and now it has generated ten (Hello,1) more

Combiner will run again and it will generate one spill file

Scenario 2: Combiner running more than once



Assumption:

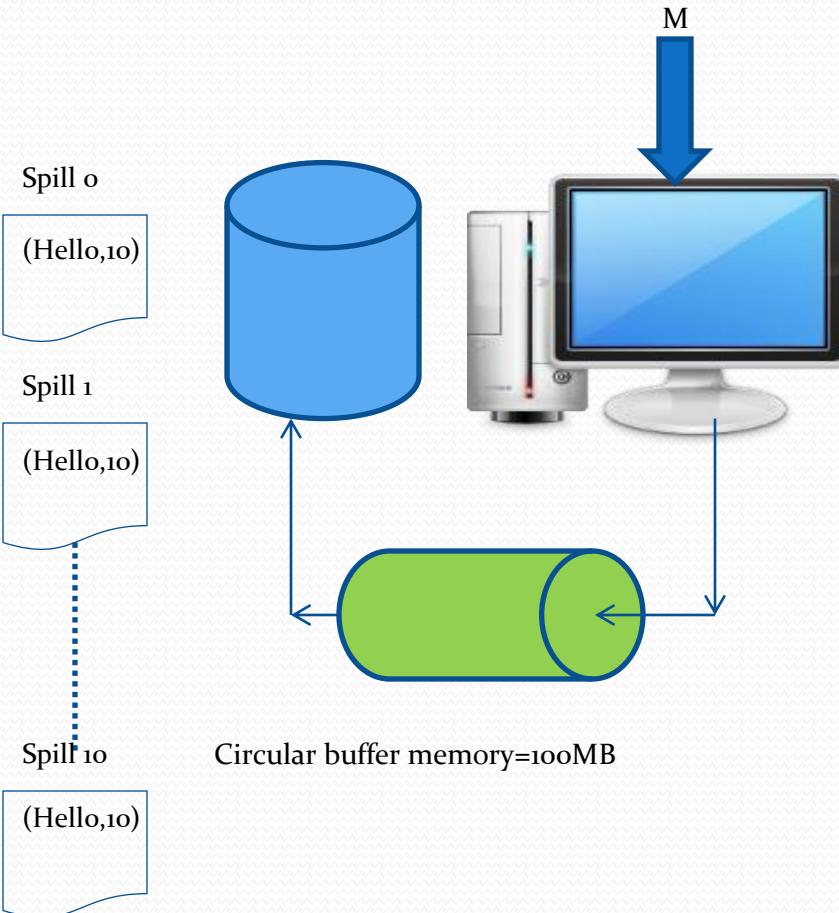
If (Hello,1) is emitted 10 times, then it has exceeded the circular buffer memory

Assuming combiner already ran once and now it has generated ten (Hello,1) more

Combiner will run again and it will generate one more spill file

If no more key value pairs are generated then two key value pairs will be emitted from this map task

Scenario 2: Combiner running more than once

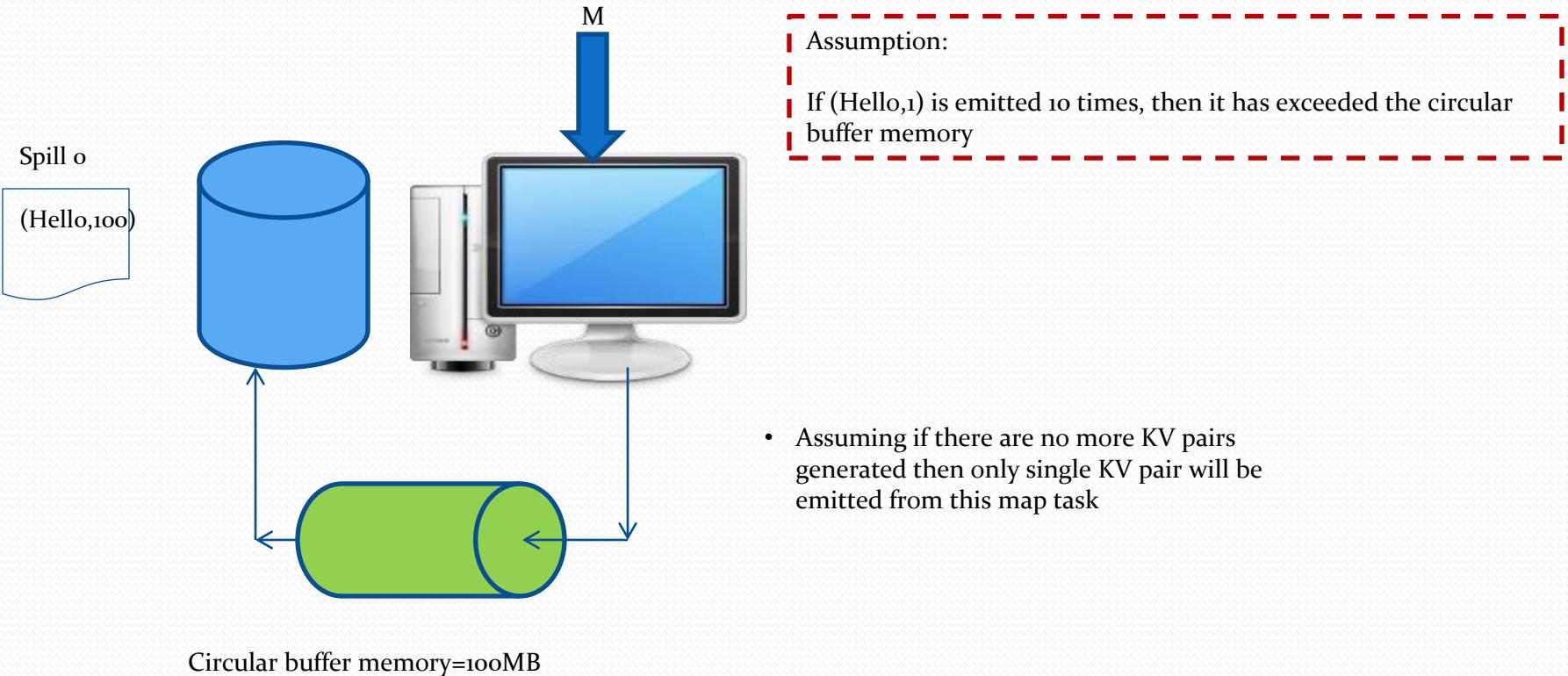


Assumption:

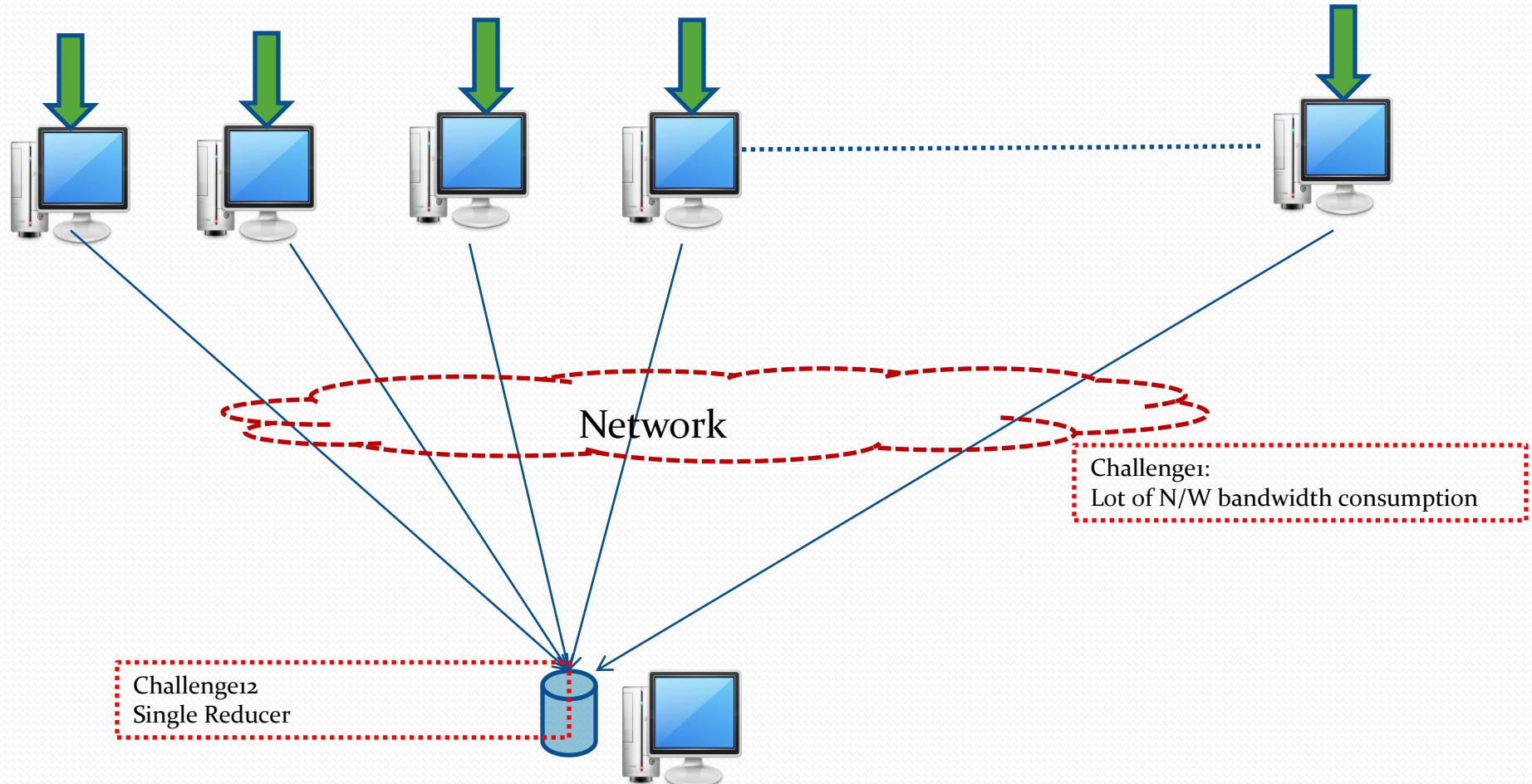
If $(\text{Hello}, 1)$ is emitted 10 times, then it has exceeded the circular buffer memory

- Assuming there are 10 spills happened
- $\text{io.sort.factor}=10$, so combiner will run again on these 10 spill files and generate single file

Scenario 2: Combiner running more than once



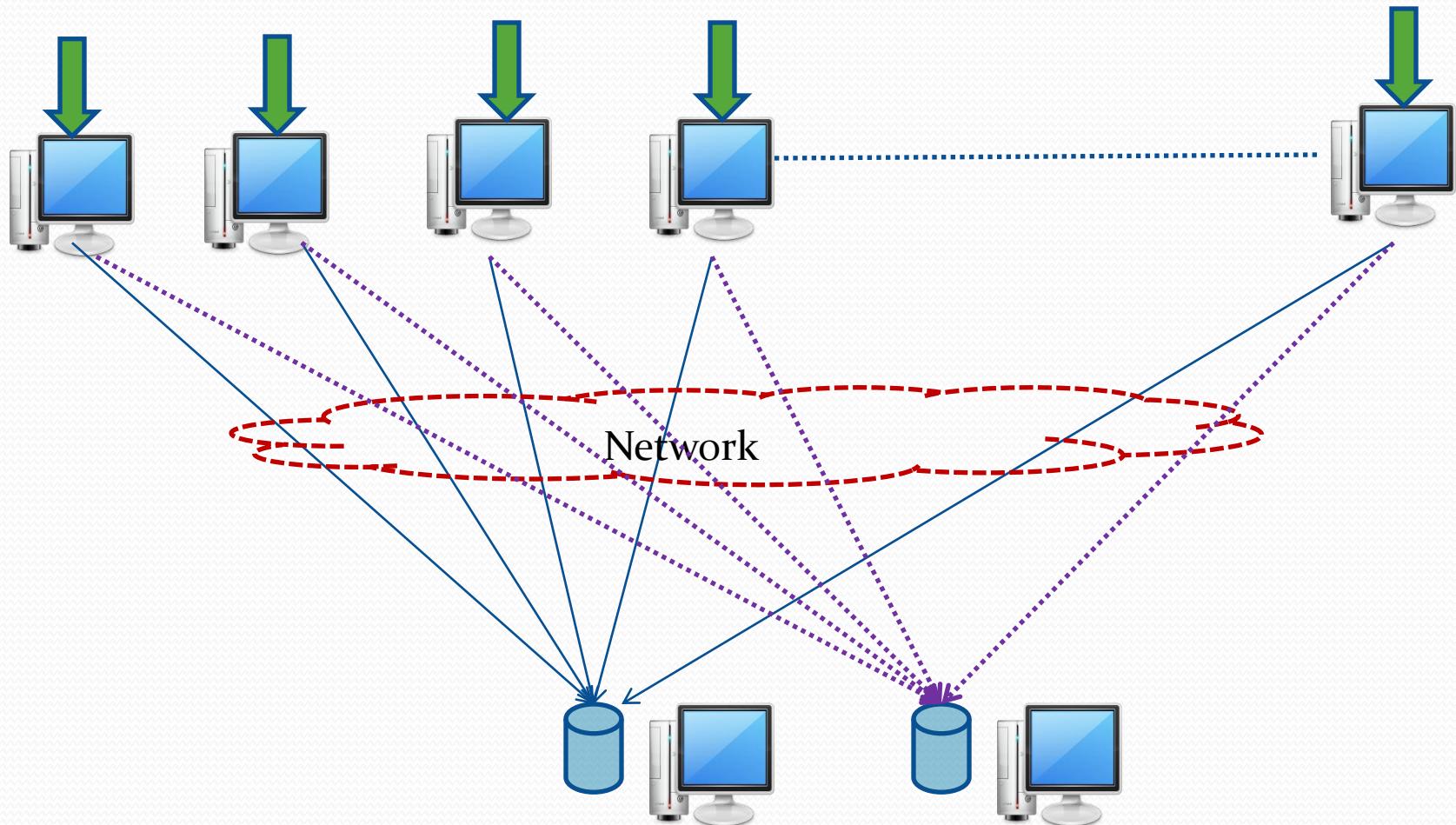
MapReduce Data flow



Challenge 2: Single Reducer Problem

- All the key-value pairs needs to be copied to a single reducer machine
 - Copying will be time consuming
 - Sorting will be time consuming
 - High chances that single reducer machine might not be handle that much of data
 - The size of key value pairs that are getting copied might exceed the hard disk capacity of the reducer machine
- Solution: In this case you might want to load balance the key value pairs by running more than 1 reducer

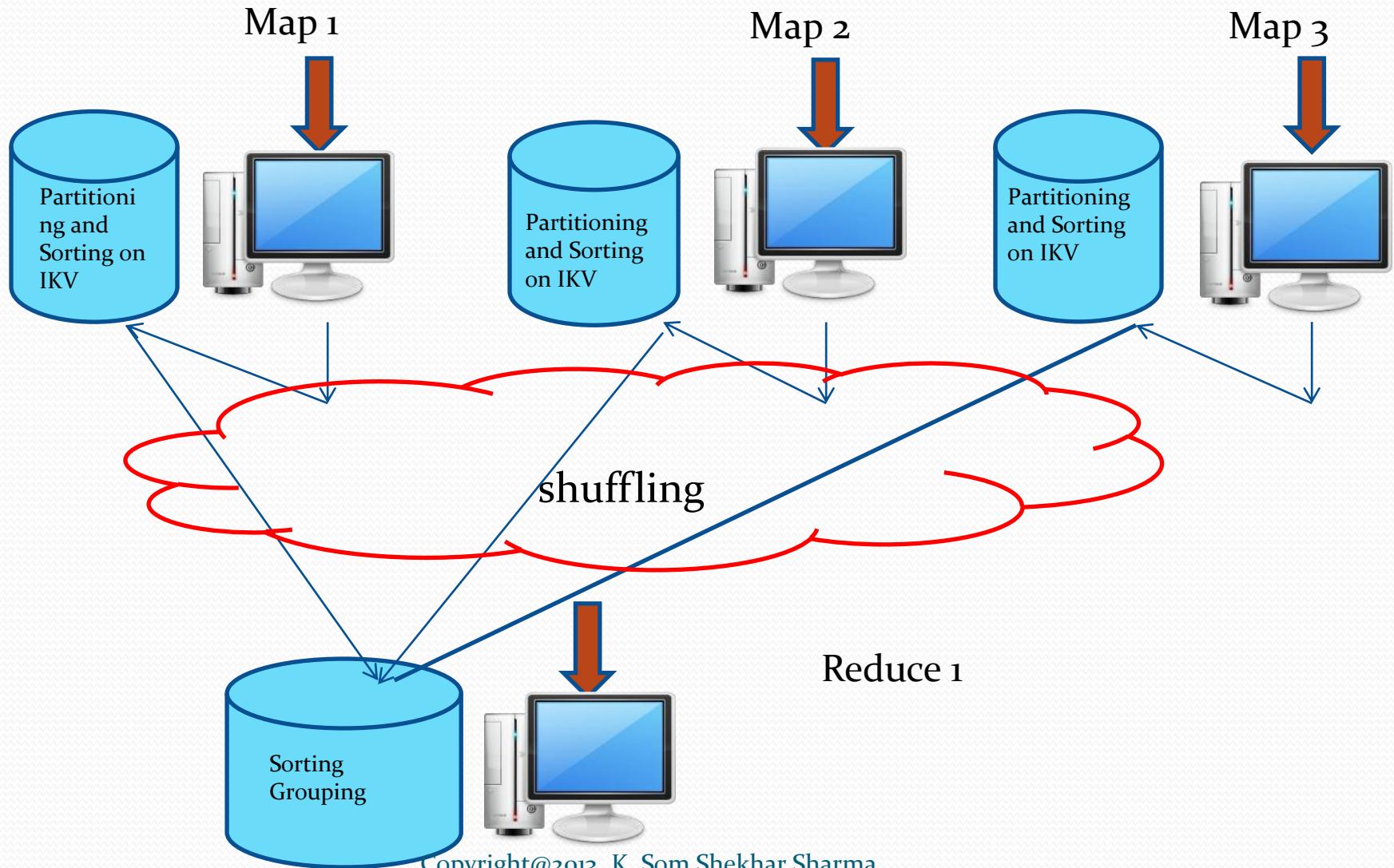
Mitigating Single Reducer Problem



Running multiple reducer

- Partitioner running on the mapper side decides which key value pairs should go to which reducer
 - Partitioner must send the same keys to the same reducer.

MapReduce – Data Flow



Partitioner

- Large number of mappers running will generate large amount of data
 - And If only one reducer is specified, then all the intermediate key and its list of values goes to a single reducer
 - Copying will take lot of time
 - Sorting will also be time consuming
 - Whether single machine can handle that much amount of intermediate data or not?
- Solution is to have more than one reducer
- Partitioner works on the key value pairs emitted from the map tasks

Partitioner cont'd

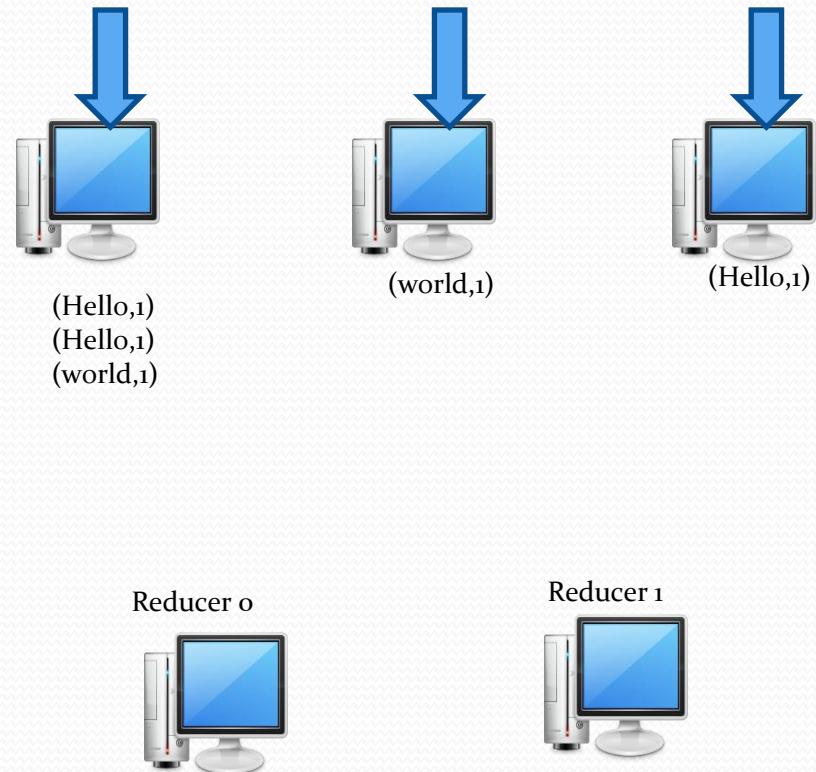
- Default is “*Hash Partitioner*”
 - Calculates the “*hashcode*” and do the modulo operator with total number of reducers which are running

Hash code of key %numOfReducer = Reducer number where this KV will go

The above operation returns between ZERO and (numOfReducer – 1)

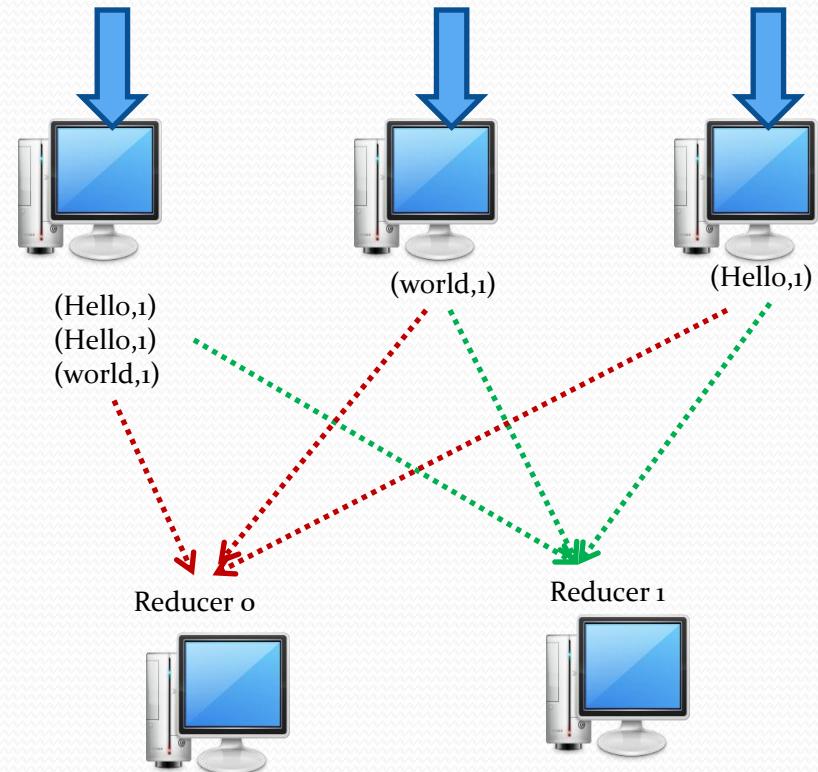
Partitioner cont'd

- Example:
 - Number of reducers = 2
 - Key = “Hello”
 - Hash code = 30 (let’s say)
 - The key “Hello” and its value will go to $30 \% 2 = 0^{\text{th}}$ reducer
 - Key = “World”
 - Hash Code = 31 (let’s say)
 - The key “world” and its value will go to $31 \% 2 = 1^{\text{st}}$ reducer



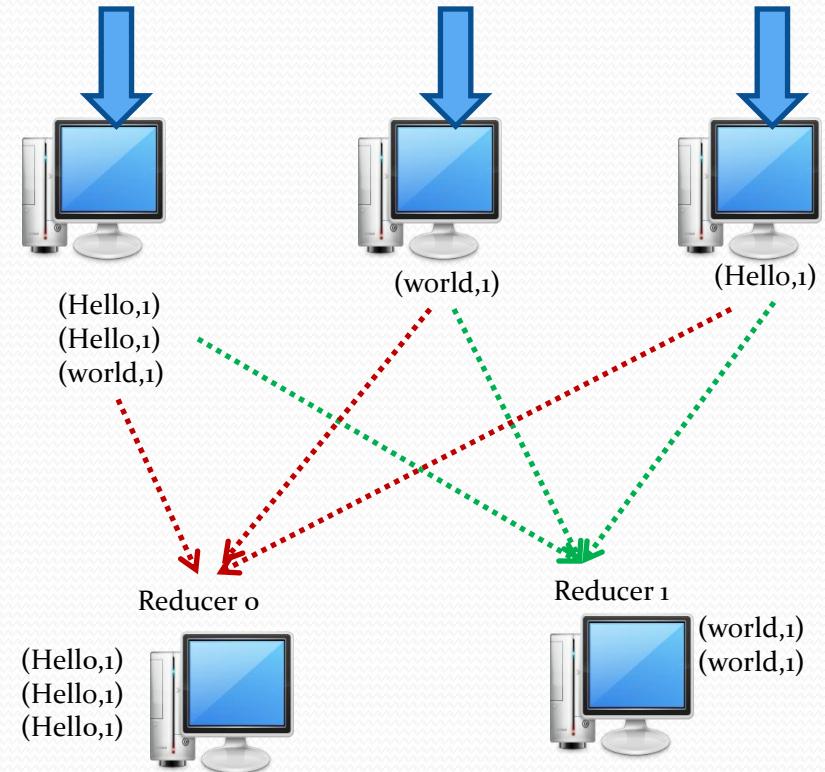
Partitioner cont'd

- Example:
 - Number of reducers = 2
 - Key = "Hello"
 - Hash code = 30 (let's say)
 - The key "Hello" and its value will go to $30 \% 2 = 0^{\text{th}}$ reducer
 - Key = "World"
 - Hash Code = 31 (let's say)
 - The key "world" and its value will go to $31 \% 2 = 1^{\text{st}}$ reducer



Partitioner cont'd

- Example:
 - Number of reducers = 2
 - Key = "Hello"
 - Hash code = 30 (let's say)
 - The key "Hello" and its value will go to $30 \% 2 = 0^{\text{th}}$ reducer
 - Key = "World"
 - Hash Code = 31 (let's say)
 - The key "world" and its value will go to $31 \% 2 = 1^{\text{st}}$ reducer



Hands-on

- Refer hands-on document

Local Runner

Local Runner	Cluster Mode
Local File System	HDFS
Map and reducer tasks runs on single JVM	Every map and reduce tasks runs on different jvm.
Only one reducer	Reducers can be more than 1
No Distributed Cache facility is available	Can use distributed cache facility
<code>fs.default.name=file:///</code>	<code>fs.default.name=hdfs://IP:PORT</code>
<code>mapred.job.tracker=local</code>	<code>mapred.job.tracker=IP:PORT</code>

Motivation of Tool Runner

```
public class WordCountDriver
{
    public static void main(String[] args)
    {
        //Job Code
    }
}
```

What is the problem in writing the driver code like this?

Motivation of Tool Runner

```
public class WordCountDriver  
{  
    public static void main(String[] args)  
    {  
        //Job Code  
    }  
}
```

What is the problem in writing the driver code like this?

- If you want to change some Hadoop properties, you need to change the code, recreate jar and run it.
 - If your project is really very big, then compilation will be time consuming
- Very inefficient way, if you would like to tune your map reduce job

Motivation of Tool Runner

```
public class WordCountDriver
{
    public static void main(String[] args)
    {
        //Job Code
    }
}
```

Best Option

- We should be able to provide the Hadoop properties via the command line while submitting the job.
 - It will eliminate the need of changing the driver code

Motivation of Tool Runner

```
public class WordCountDriver
{
    public static void main(String[] args)
    {
        //Job Code
    }
}
```

```
public class WordCountDriver extends Configured implements Tool
{
    @Override
    public int run(String[] args)
    {
        //Your Job Code
    }
    public static void main(String[] args)
    {
        ToolRunner.run(new WordCountDriver(),args);
    }
}
```

Motivation of Tool Runner

```
public class WordCountDriver extends Configured implements Tools
{
    @Override
    public int run(String[] args)
    {
        //Your Job Code
    }
    public static void main(String[] args)
    {
        ToolRunner.run(new WordCountDriver(),args);
    }
}
```

Advantages:

- Configured Class provides getConf() method which will read the properties from xml files present on the client machine.
- You can provide Hadoop/custom properties from the command line using “-D” switch

Motivation of Tool Runner

```
public class WordCountDriver extends Configured implements Tools
{
    @Override
    public int run(String[] args)
    {
        Job job = new Job(getConf());
    }
    public static void main(String[] args)
    {
        ToolRunner.run(new WordCountDriver(),args);
    }
}
```

BEWARE:

- Only the default properties can be changed. Properties which are explicitly specified in the driver code, can't be changed from the command line.

Default properties

Property Name	Default Value	Description
mapred.reduce.tasks	1	Number of reducer tasks
mapreduce.Map.class	IdentityMapper	If you don't explicitly specify in the driver code, Identity Map function will be called
mapreduce.Reduce.class	IdentityReducer	If you don't explicitly specify in the driver code, Identity reducer function will be called
	TextInputFormat	
InputSplitSize	dfs.block.size	
	TextOutputFormat	
mapred.job.name	Name of the jar	Job Name

Tool Runner

```
hadoop jar WordCount.jar WordCountDriver -D mapred.job.name=WordCount <input> <output>
```

```
hadoop jar WordCount.jar WordCountDriver -D mapred.job.name=WordCount -D  
mapred.reduce.tasks=o <input> <output>
```

```
hadoop jar WordCount.jar WordCountDriver -D mapred.job.name=WordCount -D  
mapred.reduce.tasks=o -D mapred.min.split.size=1048576 <input> <output>
```

```
hadoop jar WordCount.jar WordCountDriver -D mapred.job.name=WordCount -D  
mapred.reduce.tasks=o -D mapred.min.split.size=1048576 -D fs.default.name=file:/// -D  
mapred.job.tracker=local <input> <output>
```

BEWARE:

You should not have specified any of the properties in the driver code. If it is specified then the above commands will not have any effect

Tool Runner

- List of available options

- -D property=value
- -conf fileName [For overriding the default properties]
- -fs uri [-D fs.default.name=<NameNode URI>]
- -jt host:port [-D mapred.job.tracker=<JobTracker URI>]
- -files file1,file2 [Used for distributed cache]
- -libjars jar1, jar2, jar3 [Used for distributed cache]
- -archives archive1, archive2 [Used for distributed cache]

Tool Runner

```
hadoop jar WordCount.jar WordCountDriver -D mapred.job.name=WordCount -D  
mapred.reduce.tasks=0 -D mapred.min.split.size=1048576 -D fs.default.name=file:/// -D  
mapred.job.tracker=local <input> <output>
```

Providing the properties on the command line is cumbersome.

Setup/cleanup Method in Mapper and Reducer

- 3 functions can be overridden by your map/reduce class
 - *setup* method (Optional)
 - *map/reduce* method
 - *cleanup* method (Optional)
- *setup* method is called only once before calling the *map* function
 - If anything extra (parameter, files etc) is required while processing data in map/reduce task, it should be initialized or kept in memory in the *setup* method
 - Can be done in *map* function, but mapper will be called for every record

setup/cleanup method cont'd

- clean up method is called after the map/reduce function is over
 - Can do the cleaning like, if you have opened some file in the *setup* method, you can close the file

Setup/cleanup method cont'd

```
public class WordCountMapper extends Mapper <LongWritable, Text, Text,  
Intwritable> {  
  
    String searchWord;  
    public void setup(Context context)  {  
        searchWord = context.getConfiguration().get("WORD");  
    }  
  
    public void map(LongWritable inputKey,Text inputVal, Context context) {  
        //mapper logic goes here  
    }  
  
    public void cleanup()  {  
        //clean up the things  
    }  
}
```

Setup/cleanup method cont'd

```
public class WordCountMapper extends Mapper<LongWritable,Text,TextIntwritable>
{
    String searchWord;
    public void setup(Context context)
    {
        searchWord = context.getConfiguration().get("WORD");
    }
}
```

You can send the value of the property “WORD” from command line as follows

hadoop jar myJar MyDriver -D WORD=“hello”

In the *setup* method you can access the value of the property

Setup/cleanup method cont'd

Once you get the *searchWord* in the *setup* method. You can use this in your mapper

```
public void map(LongWritable inputKey, Text inputVal, Context context)
{
    //mapper logic goes here
    if(searchWord.equals(inputVal.toString()))
    {
        //do something
    }
}
```

Hands-on

- Refer hands-on document

Motivation-Distributed Cache

Employee

X,Y,UK
A,B,IN
A,T,US

Location

UK, United Kingdom
IN, India
US, United States

Problem Statement:

Replace the location id of every employee record with their corresponding location name.

Output

X, Y, United Kingdom
A, B, India
A, T, United States

Motivation-Distributed Cache

Employee

X,Y,UK
A,B,IN
A,T,US

Location

UK, United Kingdom
IN, India
US, United States

Problem Statement:

Replace the location id of every employee record with their corresponding location name.

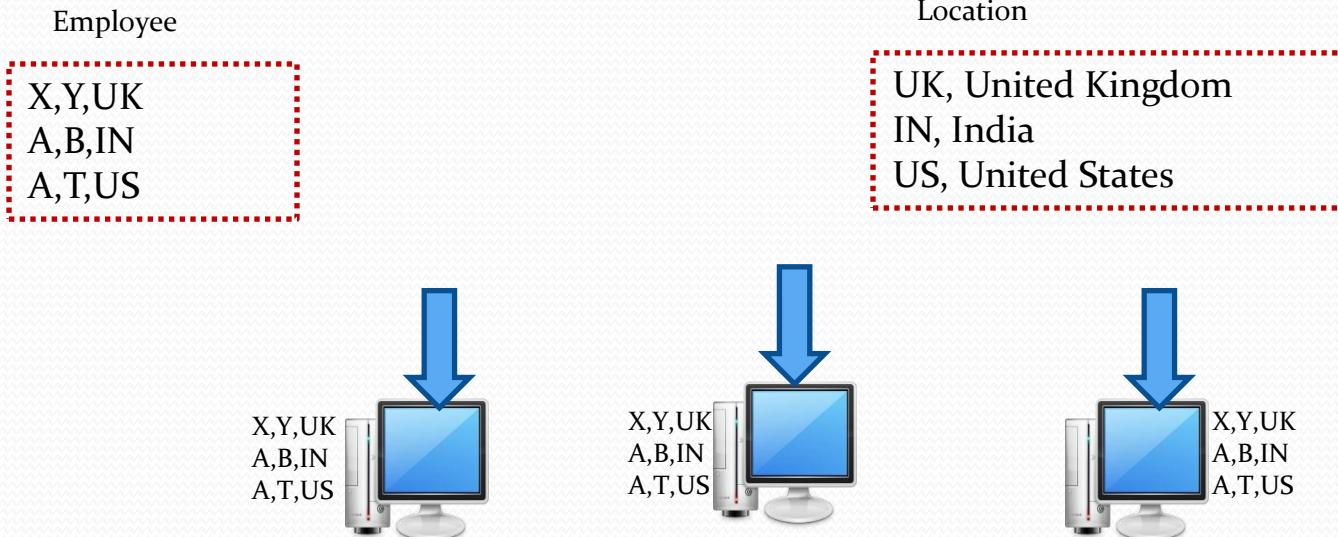
Output

X, Y, United Kingdom
A, B, India
A, T, United States

Decide the following things:

- On which data set you need to run map reduce job?
- Which is your extra data?

Motivation-Distributed Cache



- While map task be processing the employee records, and to replace the location id of every employee record, they need the entire location data
- The entire location data needs to be present with each map task
- How to make location data available to each of the map task?

Motivation-Distributed Cache

- While map task be processing the employee records, and to replace the location id of every employee record, they need the entire location data
- The entire location data needs to be present with each map task
- How to make location data available to each of the map task?
 - Place the location data on HDFS .

Motivation-Distributed Cache

Assuming the location data is on HDFS. How will you write your map function now?

Motivation-Distributed Cache

```
public void setup(Context context)
{
```

Step1: Using HDFS API, read the location data which is present on HDFS

Step2: Form look up table or HashMap<LocId,LocName>

```
}
```

```
public void map (LongWritable offset, Text value, Context context)
```

```
{
```

Step1: Extract the location id from the employee record.

Step2: Get the corresponding location name from the hash map which is formed in the setup method

Step 3: Replace the location id of the employee record with the location name

```
}
```

Motivation-Distributed Cache

What is the problem with this approach?

Motivation-Distributed Cache

What is the problem with this approach?

Reading the file from HDFS involves 3 steps:

Step1: Contacting NN

Step2: Getting the blocks and their location

Step3: Reading the blocks and merging

This process would be very slow, if you have lot of map tasks running

What is the solution?

Motivation-Distributed Cache

What is the solution?

If the location data is read from the local file system of the respective machines rather than HDFS. Then reading would be very faster

How to place the files on the local file system of all those machines where the tasks are going to run?

Motivation-Distributed Cache

How to place the files on the local file system of all those machines where the tasks are going to run?

Using Distributed cache.

- When a file is cached using distributed cache, then the files are placed on the local file system of all those machines where the task is going to run.
- And once the job is finished, the file is deleted from the local file system. Hence it's a per job configuration
- The files are placed under “hadoop-temp/mapred/DistributedCache” folder

Distributed Cache cont'd

- Following can be cached
 - Text data
 - Jar files
 - .zip files
 - .tar.gz files
 - .tgz files

Distributed Cache cont'd

- First Option: From the driver class
- Second Option: From the command line

Using Distributed Cache-First Option

- Place the files which you would like to cache on HDFS.
 - In our case, we are going to cache location data, so first we need to put on to HDFS
- You cannot cache files present on local file system

Configuring Distributed Cache

```
Job job = new Job();
DistributedCache.addCacheFile(new URI("/myapp/lookup.dat"), job);
DistributedCache.addFileToClassPath(new Path("/myapp/mylib.jar"), job);
DistributedCache.addCacheArchive(new URI("/myapp/map.zip"), job);
DistributedCache.addCacheArchive(new URI("/myapp/mytar.tar"), job);
DistributedCache.addCacheArchive(new URI("/myapp/mytgz.tgz"), job);
DistributedCache.addCacheArchive(new URI("/myapp/mytargz.tar.gz"), job);
```

Configuring Distributed Cache cont'd

- In *setup* of mapper/reducer , you can read the file

```
public class WordCountDistributedCacheReducer extends Reducer  
<Text, IntWritable, Text, IntWritable> {  
  
    private URI[] files;  
    HashMap<String, Integer> weightedAverages = new  
        HashMap<String, Integer>();  
  
    @Override  
    public void setup(Context context) throws IOException {  
        this.files = DistributedCache.  
            getCacheFiles(context.getConfiguration());  
        Path path = new Path(files[0]);  
        //do something  
    }  
}
```

Using Distributed Cache-Second Option

- You can send the files from the command line
 - Your driver should have implemented *ToolRunner*

```
hadoop jar MyJar.jar MyDriver -files file1,file2,file3
```

- *-files* option is for text data
- *-libjars* option is used for adding third party jar
- *-archives* option is used for tar, tar.gz, Tgz file
 - These files automatically gets unarchived on the machines where the tasks will run.

Advantage of second option

- Files need not to be present on HDFS
 - Can cache local file
- Once File is copied to the required machine
 - Read the file as if it is normal file

Hands-on

- Refer hands-on document

Counters

- Counters provides a way for mapper and reducer to pass aggregate values back to the driver code after the job has finished
 - Example: You can use counter to count the number of invalid and valid (good and bad) records
- Counters are like name and value pair
 - Value can be incremented with in the code
- Counters are collected into groups (enum)
- For example: Let's say we have group of counters by name RecordType
 - Names : validRecord, invalidRecord
 - Appropriate counter will be incremented as each record is read

Counter's Cont'd

- Counter's are also available from the job tracker's web UI

Hands On

- Refer hands-on document

Module 14

Advance MapReduce Concepts – Part 2

In this module you will learn

- How to create your own custom keys and values?
- How to create your own custom partitioner?
- How to write custom input format?

Motivation 1- Custom Keys and values

Employee

X,Y,UK,2000.50,01
A,B,IN,3000.25,02
A,T,US,15000.00,03

Location

UK, United Kingdom
IN, India
US, United States

You would like to run map reduce job on both these data sets together

Motivation 1- Custom Keys and values

Employee

X,Y,UK,2000.50,01
A,B,IN,3000.25,02
A,T,US,15000.00,03

Location

UK, United Kingdom
IN, India
US, United States

You can represent every thing as Text and whenever you required to convert to double or int, you cast it.

Not an efficient way

Motivation 1- Custom Keys and values

Employee

X,Y,UK,2000.50,01
A,B,IN,3000.25,02
A,T,US,15000.00,03

Location

UK, United Kingdom
IN, India
US, United States



```
public class myDataStructure
{
    Text fName;
    Text lName;
    Text locId;
    Text locName;
    DoubleWritable salary;
    IntWritable id;
}
```

Motivation 2- Custom Keys and values

lastName	firstName
Sharma	X
Sharma	B
Mishra	Z
Sharma	A
Mishra	A
Mishra	B

Last names are grouped together

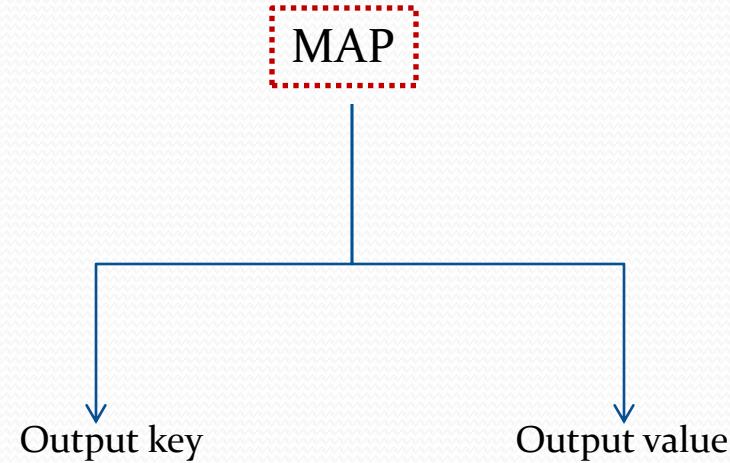
Problem Statement:
Group all the records having the same last name together and while writing the output the first names should be sorted in ascending order



lastName	firstName
Mishra	A
Mishra	B
Mishra	Z
Sharma	A
Sharma	B
Sharma	X

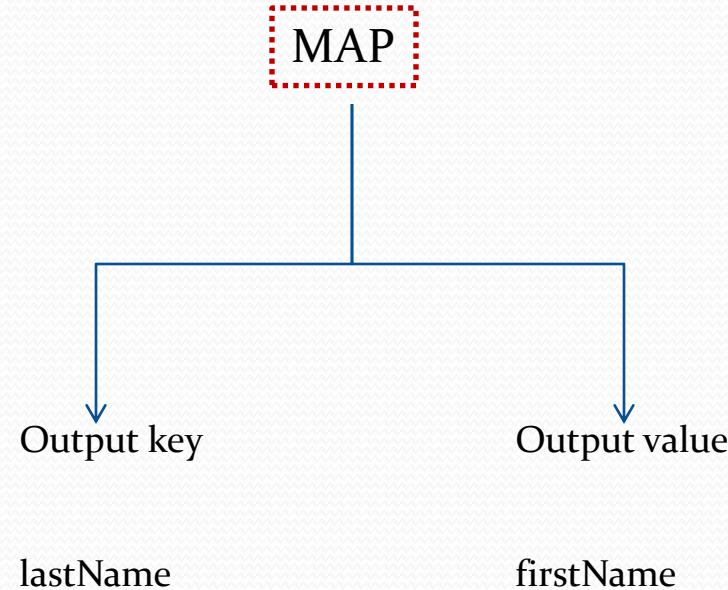
Motivation 2- Custom Keys and values

lastName	firstName
Sharma	X
Sharma	B
Mishra	Z
Sharma	A
Mishra	A
Mishra	B

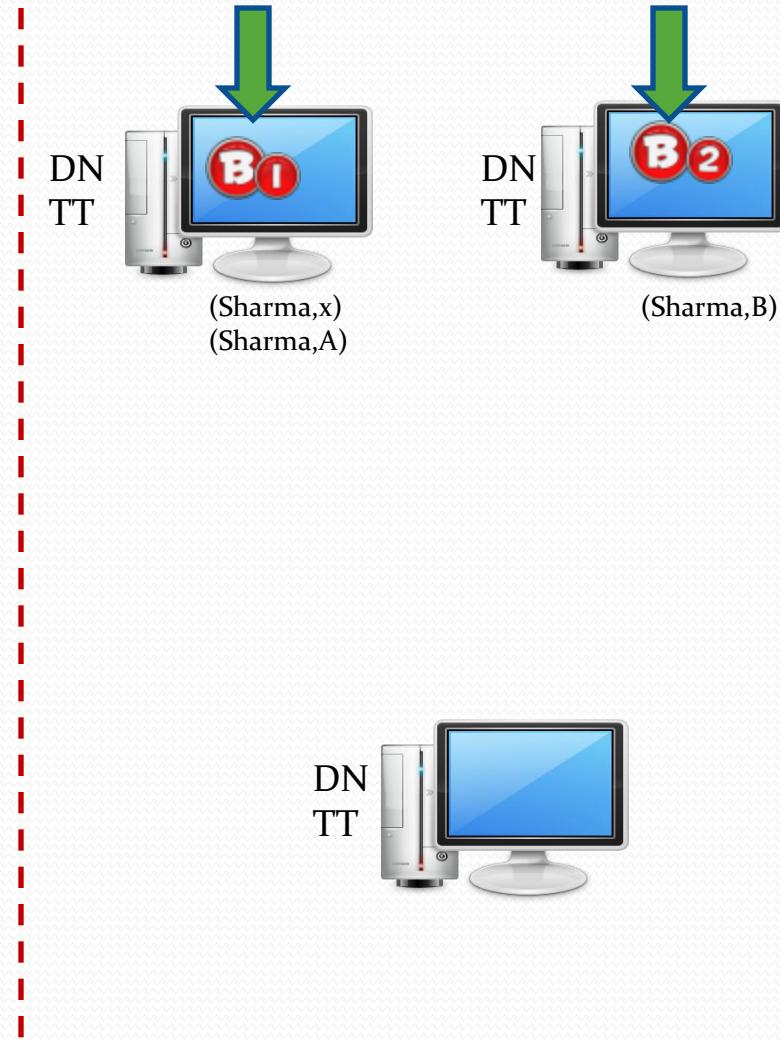
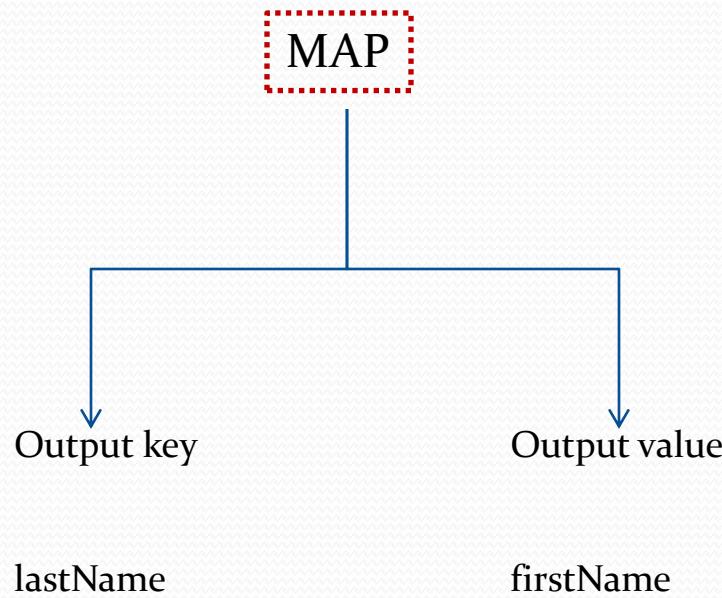


Motivation 2- Custom Keys and values

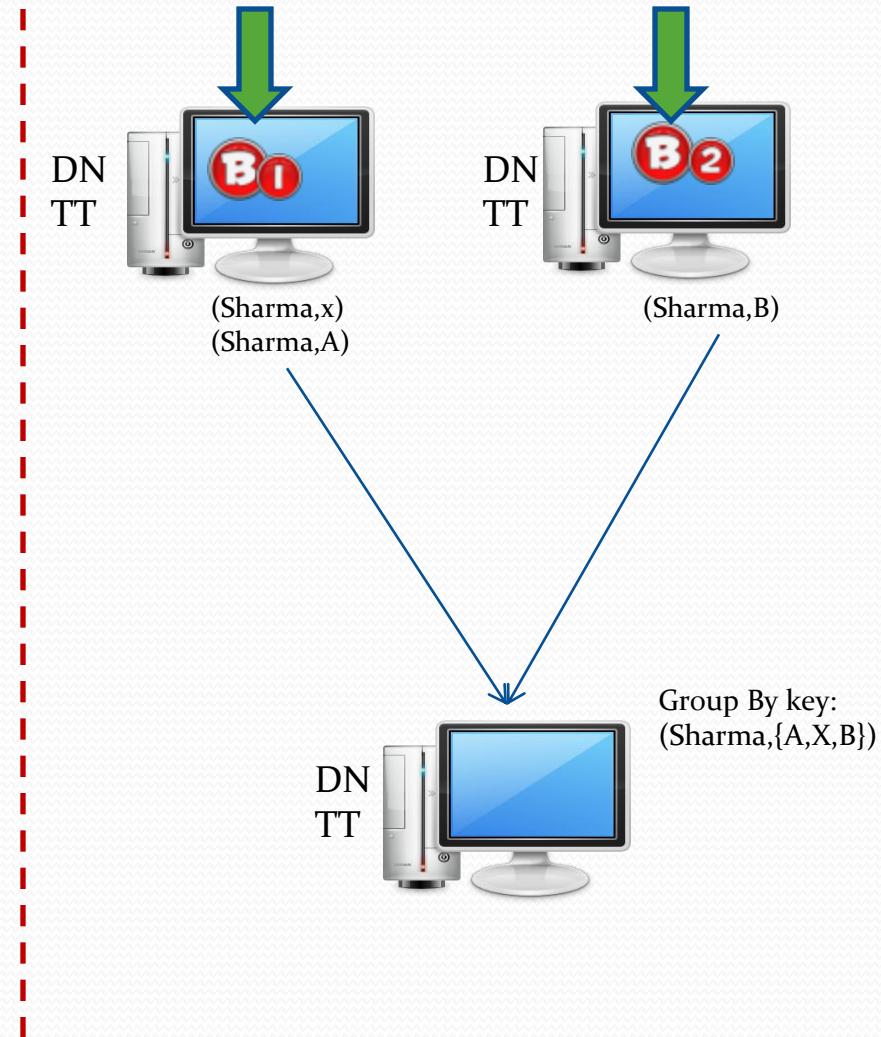
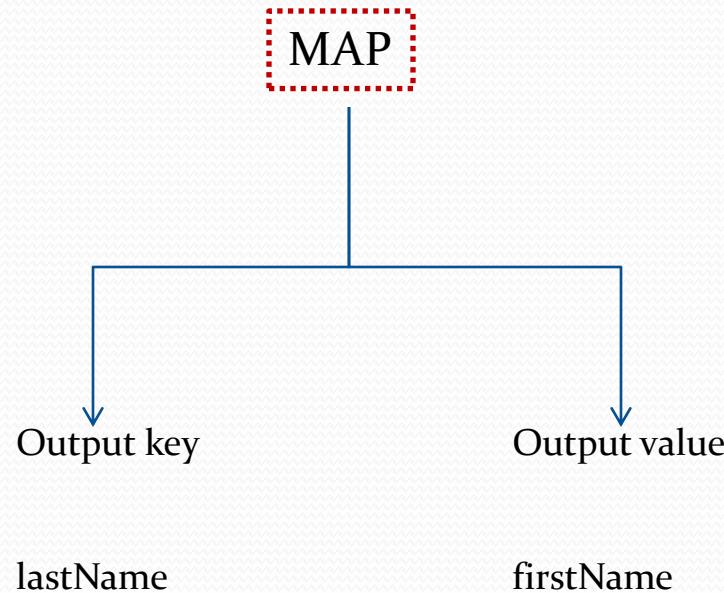
lastName	firstName
Sharma	X
Sharma	B
Mishra	Z
Sharma	A
Mishra	A
Mishra	B



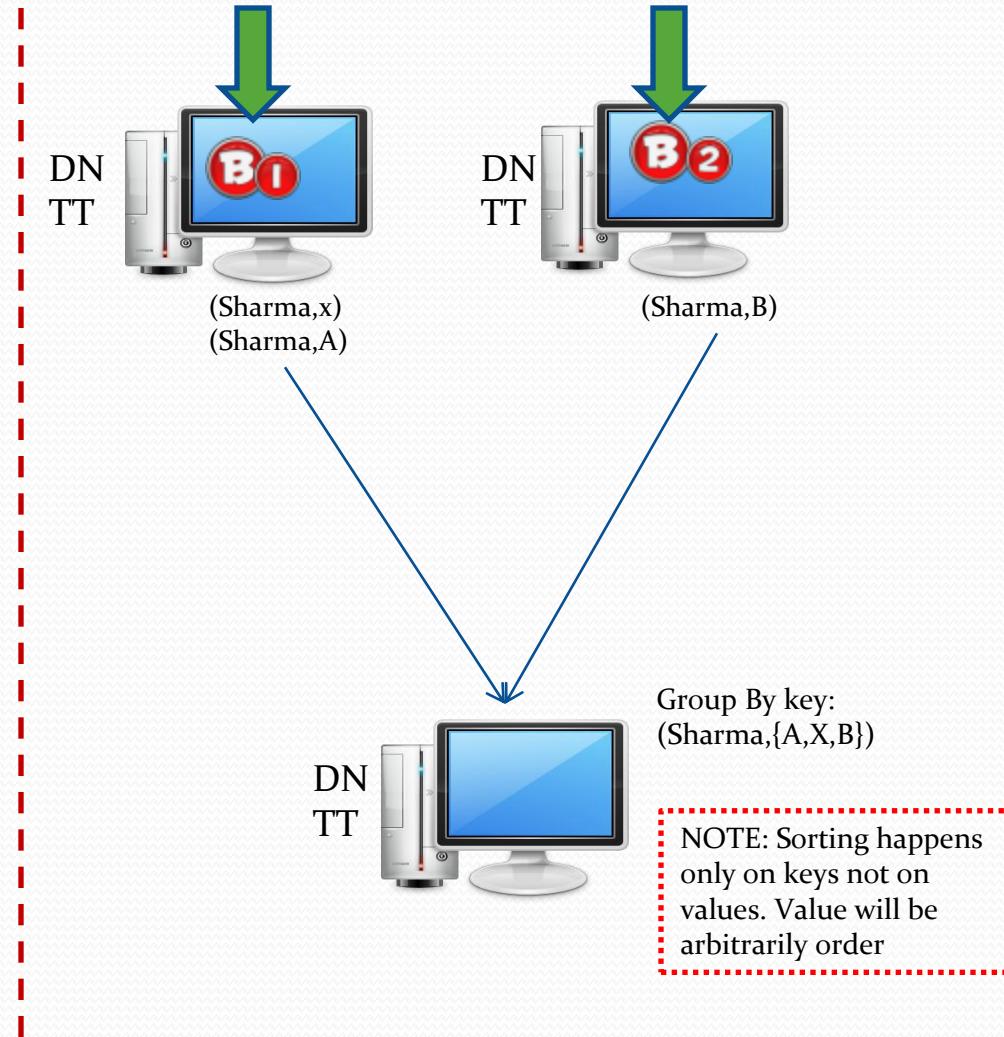
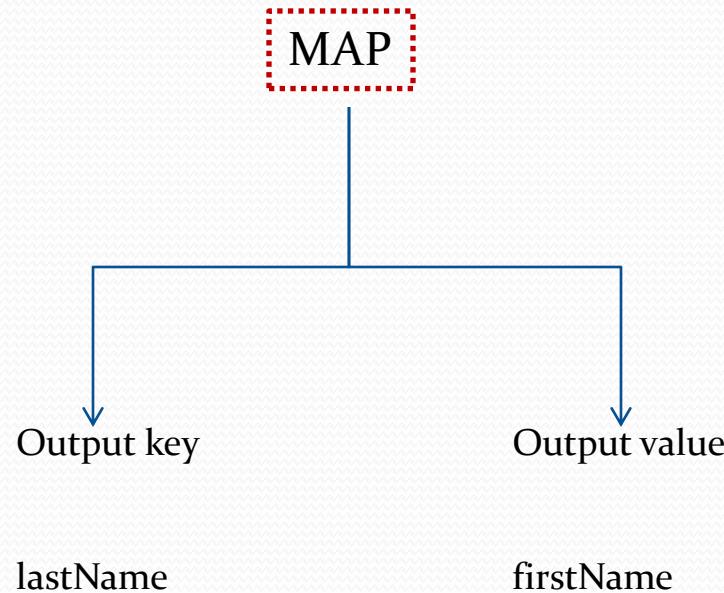
Motivation 2- Custom Keys and values



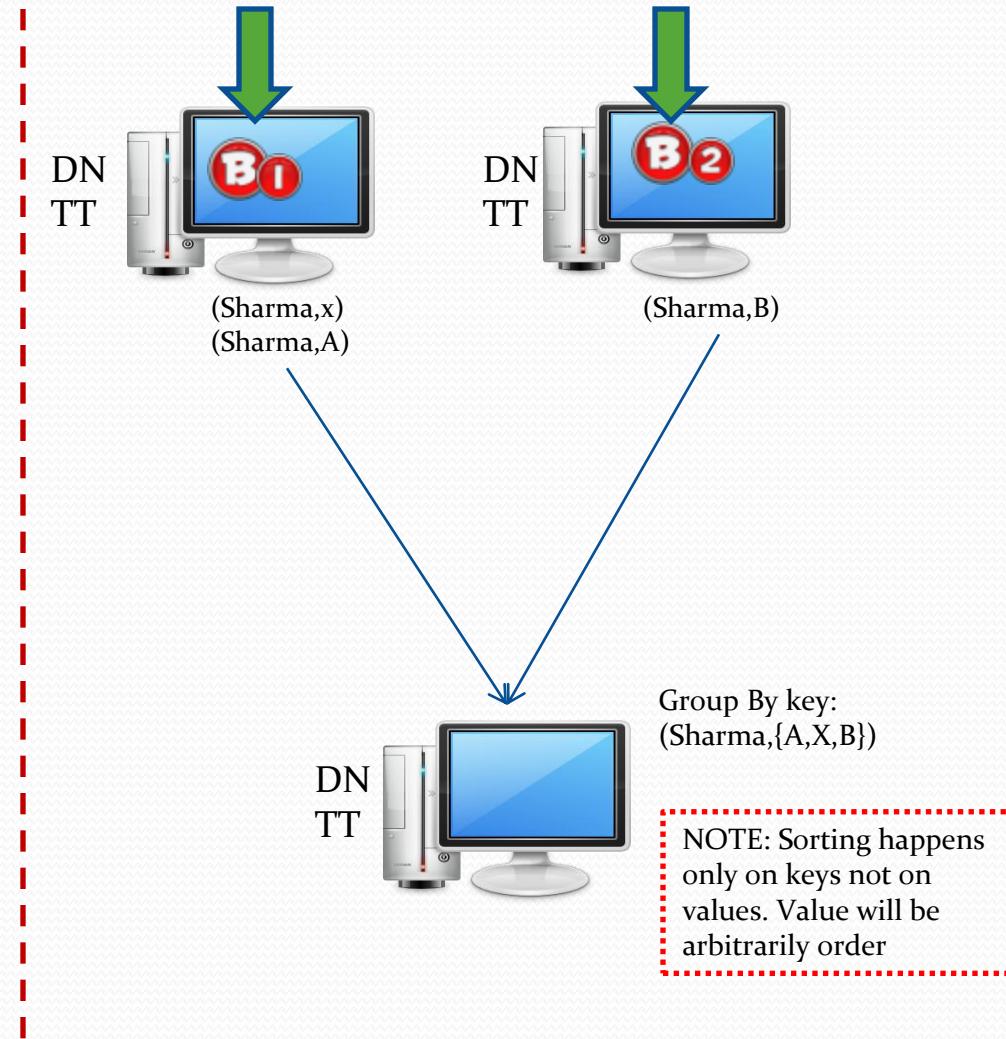
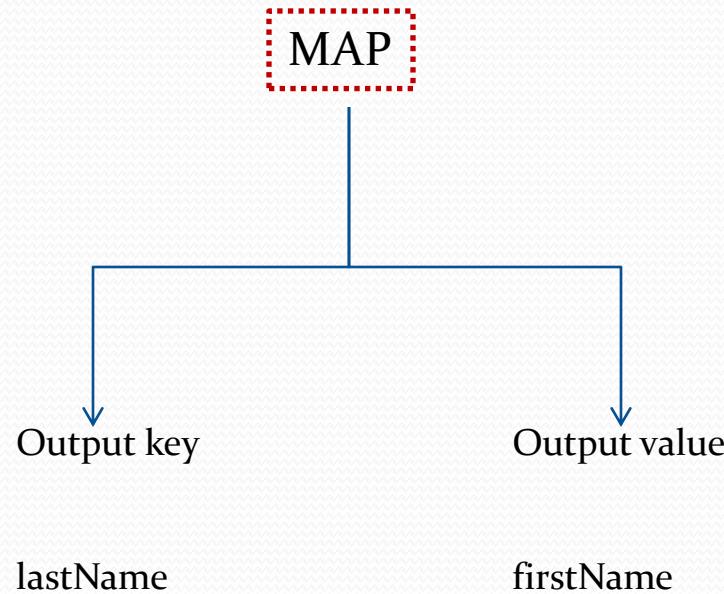
Motivation 2- Custom Keys and values



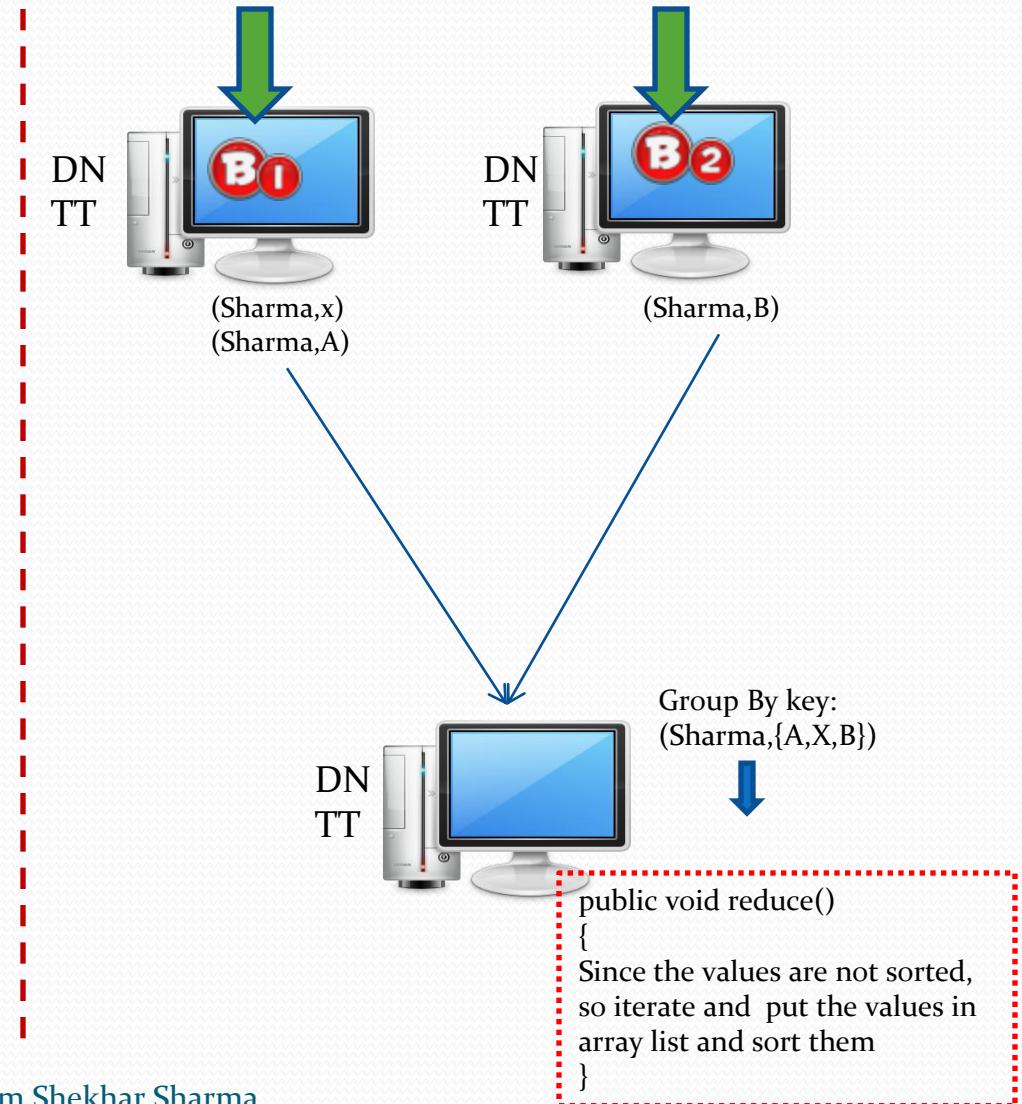
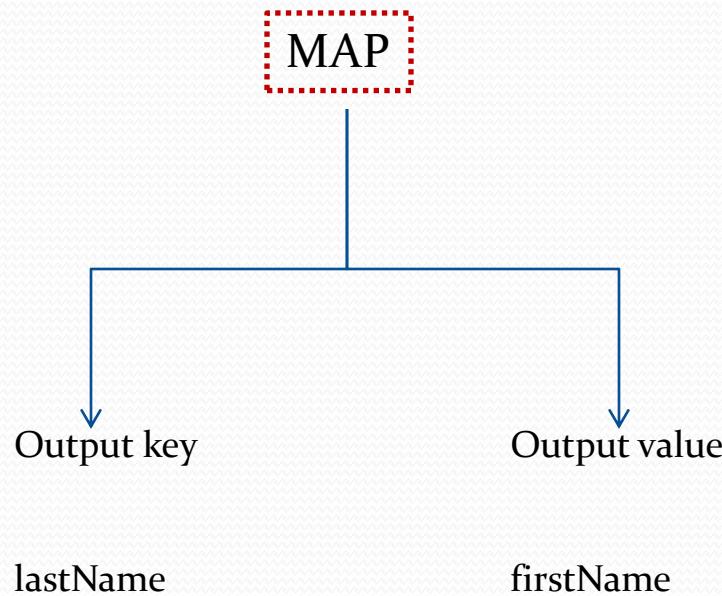
Motivation 2- Custom Keys and values



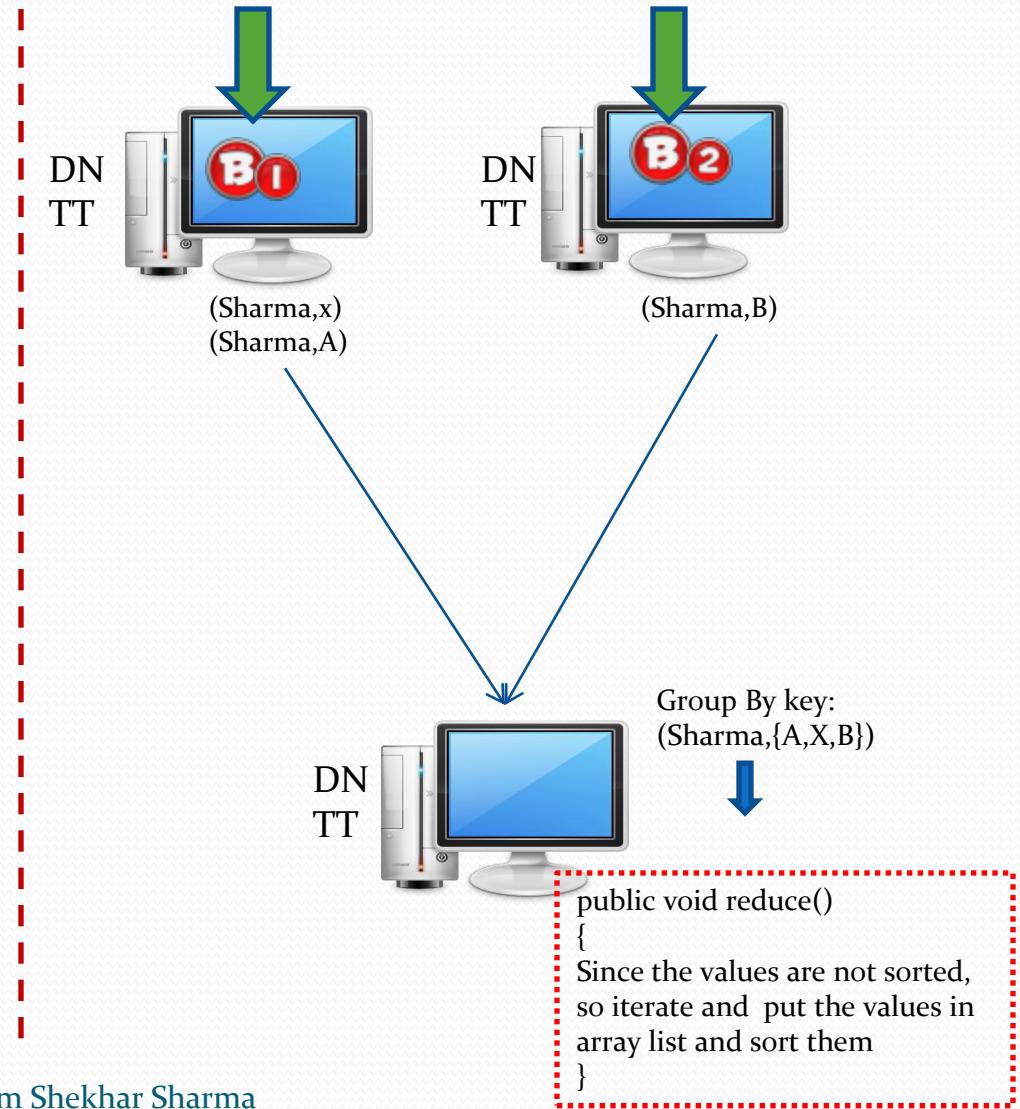
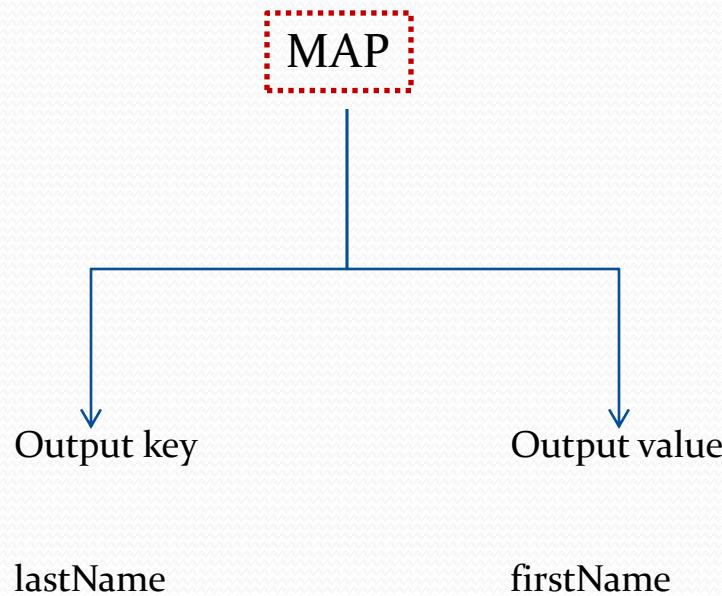
Motivation 2- Custom Keys and values



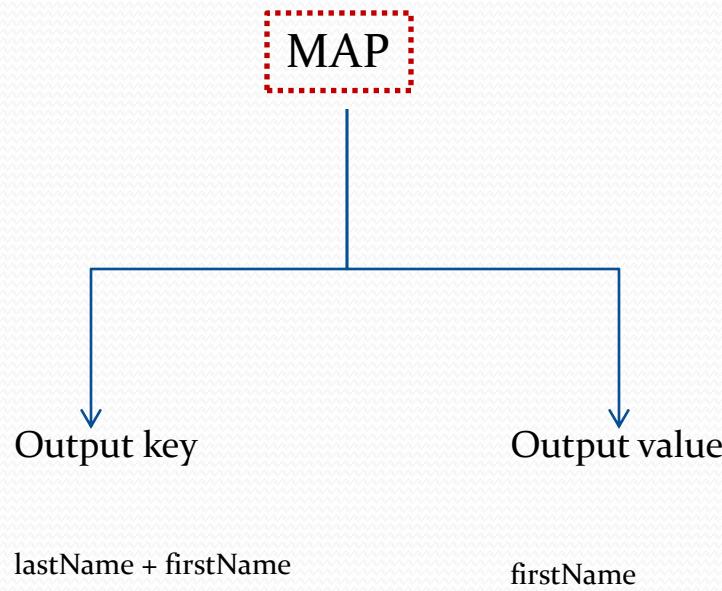
Motivation 2- Custom Keys and values



Motivation 2- Custom Keys and values



Motivation 2- Custom Keys and values



Implementing custom keys and values

- Working with Hadoop primitive data types does not offer much flexibility over grouping and sorting
- Custom keys/values will be useful for forming composite keys or complex data structure
- Example:
 - Data has 3 fields only *lastName*,*firstName*, and *empId*
 - You would like to group the data by *lastName* and sorting should be done on both *lastName* and *firstName*

Writable and WritableComparable

- Hadoop uses its own serialization mechanism for transferring the intermediate data over the network
 - Fast and compact
 - Hadoop does not use Java serialization
- Implement Writable interface for custom values
- Implement WritableComparable interface for custom keys
 - Since the keys will be compared during sorting phase, so provide the implementation for *compareTo()* method

Implementing Custom Values

```
public class PointWritable implement Writable {  
    private IntWritable xcoord;//x coordinate  
    private IntWritable ycoord;//y coordinate  
  
    //Provide necessary constructors and getters and setters  
    @Override  
    public void readFields(DataInput in) throws IOException {  
        xcoord.readFields(in);  
        ycoord.readFields(in);  
    }  
    @Override  
    public void write(DataOutput out) throws IOException {  
        xcoord.write(out);  
        ycoord.write(out);  
    }  
}
```

Implementing Custom Values cont'd

```
public class PointWritable implements Writable {  
    private IntWritable xcoord;//x coordinate  
    private IntWritable ycoord;//y coordinate  
  
    //Provide necessary constructors and getters and setters
```

- Your custom value class should implement Writable interface
- Provide the necessary constructors to initialize the member variables
- Provide setters and getters method for member variables
- Provide *equals()* and *hashCode()* method

Implementing Custom Values cont'd

```
public class PointWritable implements Writable {  
    private IntWritable xcoord;//x coordinate  
    private IntWritable ycoord;//y coordinate  
  
    //Provide necessary constructors and getters and setters
```

@Override

```
public void readFields(DataInput in) throws IOException {  
    xcoord.readFields(in);  
    ycoord.readFields(in);
```

Read the fields in the same order you have defined

```
xcoord.write(out);  
ycoord.write(out);  
}  
}
```

Implementing Custom Values cont'd

```
public class PointWritable implements Writable {  
    private IntWritable xcoord;//x coordinate  
    private IntWritable ycoord;//y coordinate  
  
    //Provide necessary constructors to initialize the member variables  
    @Override  
    public void readFields(DataInput in) throws IOException {  
        Write the fields in the same order you have defined  
    }  
    @Override  
    public void write(DataOutput out) throws IOException {  
        xcoord.write(out);  
        ycoord.write(out);  
    }  
}
```

Implementing Custom Keys

```
public class Person implements WritableComparable<Person>
{
    private Text lastName;
    private Text firstName;

    @Override
    public void readFields(DataInput in) throws IOException {
        lastName.readFields(in);
        firstName.readFields(in);
    }

    @Override
    public void write(DataOutput out) throws IOException {
        lastName.write(out);
        firstName.write(out);
    }
    @Override
    public int compareTo(Person other) {

        int cmp = lastName.compareTo(other.getLastName());

        if(cmp != 0) {
            return cmp;
        }

        return firstName.compareTo(other.getFirstName());
    }
}
```

Implementing Custom Keys cont'd

```
public class Person implements WritableComparable<Person> {  
    private Text lastName;  
    private Text firstName;  
  
    @Override  
    public void readFields(DataInput in) throws IOException {  
        ....  
    }  
  
    @Override  
    public void write(DataOutput out) throws IOException {  
        ...  
    }  
}
```

- Writable Comparable interface extends Writable interface
- Must implement compareTo() method, because keys needs to be compared during sorting phase
- Developers should provide *equals()* and *hashCode()* method

Implementing Custom Keys cont'd

```
@Override  
public int compareTo(Person other) {  
    int cmp = lastName.compareTo(other.getLastName());  
    if(cmp != 0) {  
        return cmp;  
    }  
  
    return firstName.compareTo(other.getFirstName());  
}  
}
```

- Compare the fields
- Instead of comparing both the fields, you can compare single fields

Hands-on

- Refer hands-on document

Implementing Custom Partitioner

- Recap:
 - For better load balancing of key value pairs, you should consider more than 1 reducer
 - Partitioner decides which key value pair should go to which reducer
 - Default partitioner takes the entire key to decide which key value pair should go to which reducer

Custom Partitioner cont'd

- If using composite key(ex- firstName:lastName then it would be difficult to achieve better load balancing

Custom key (Person) [Last Name & First name]

Smith John

Smith Jacob

Smith Bob

Smith Doug

- What would you do if you want all the records having the same *lastName* should be processed by single reducer?

Custom Partitioner cont'd

Custom key (Person) [Last Name & First name]

Smith John

Smith Jacob

Smith Bob

Smith Doug

- Default *HashPartitioner* will not work since it will take the full KEY to determine which reducer it should go
 - Records having different first name will go to different reducer
 - *Smith John* key is different than *Smith Jacob*

Custom Partitioner cont'd

Custom key (Person) [Last Name & First name]

Smith John

Smith Jacob

Smith Bob

Smith Doug

- To solve this problem implement custom partitioner which partitions the key w.r.t the *lastName*
 - All the records with same *lastName* will go to same reducer

Custom Partitioner cont'd

```
public class PersonPartitioner extends Partitioner<Person, Text>{  
  
    @Override  
    public int getPartition(Person outputKey, Text outputVal, int numReducer) {  
  
        //making sure that keys having same last name goes to the same reducer  
        //because partition is being done on last name  
  
        return Math.abs(outputKey.getLastName().hashCode() * 127) % numReducer;  
    }  
}
```

Custom Partitioner cont'd

```
public class PersonPartitioner extends Partitioner<Person, Text>{
```

- Custom Partitioner should extend from Partitioner class
- Input Arguments to Partitioner class represents mapper output key and mapper output value class
- In the current scenario the map output key is custom key
 - Custom key implements WritableComparable interface

```
}
```

Custom Partitioner cont'd

```
public class PersonPartitioner extends Partitioner<Person, Text>{  
  
    @Override  
    public int getPartition(Person outputKey, Text outputVal, int numReducer) {
```

- Override the getPartition method
- First argument is map output key
- Second argument is map output value
- Third argument is number of reducer which is set either through
 - job.setNumReduceTasks()
 - Command line [-D mapred.reduce.tasks]

Custom Partitioner cont'd

```
public class PersonPartitioner extends Partitioner<Person, Text>{
```

- Note the partition is done on the *lastName*
- Doing this way, it will send all the keys with the same *lastName* to a single reducer

```
    return Math.abs(outputKey.getLastName().hashCode() * 127)  
        %numOfReducer;
```

```
}
```

Using Custom Partitioner in Job

- *Job.setPartitionerClass(CustomPartitioner.class)*

Hands-On

- Refer hands-on document

Assignment

- Modify your WordCount MapReduce program to generate 26 output files for each character
 - Each output file should consist of words starting with that character only

Implementing Custom Input Format

- In built input formats available
 - Text Input format
 - Key value text input format
 - Sequence file input format
 - Nline input format etc
- In build input formats gives you pre-defined key value pairs to your mapper
- Custom input format provides better control over the input key values processed by mapper

Custom Input Format cont'd

- Input format is responsible for
 - Processing data splits (input splits)
 - Reading records from the input splits and sending record by record to the map function in key value pair form
- Hence custom input format should implement record reader

Implementing Custom Input Format

- Assume the data is

```
en yahoo 12 1456  
en google 13 234  
aa.b fb 13 1453  
dd rediff 14 1897
```

Fields are separated by space delimiter

- Data is similar to wikipedia hourly data set
- First column is project name
- Second column is page name
- Third column is page count
- Fourth column is page size

Implementing custom input format cont'd

- Let's assume that mapper should receive the records only consisting of project name as “en”
 - Basically we are filtering the records consist of “en” project name
 - You can run map reduce job to filter the records
- Another option you can implement custom input format
 - While reading the records implement your logic

Implementing custom input format cont'd

```
public class WikiProjectInputFormat extends  
FileInputFormat<Text,IntWritable>{  
  
    @Override  
    public RecordReader<Text, IntWritable> createRecordReader(InputSplit  
        input, TaskAttemptContext arg1) throws IOException, InterruptedException  
    {  
        return new WikiRecordReader();  
    }  
}
```

Implementing custom input format cont'd

```
public class WikiProjectInputFormat extends  
FileInputFormat<Text,IntWritable>{
```

- Extend your class from FileInputFormat
- FileInputFormat<Text,IntWritable>
 - *Text* will be the input key given to the map function
 - *IntWritable* will be the input value given to the map function

```
}
```

Implementing custom input format cont'd

```
public class WikiProjectInputFormat extends  
FileInputFormat<Text,IntWritable>{  
  
    @Override  
    public RecordReader<Text, IntWritable> createRecordReader(InputSplit  
        input, TaskAttemptContext arg1) throws IOException,  
    InterruptedException
```

- Over ride the createRecordReader method
 - You need to provide your own record reader to specify how to read the records
- Note the InputSplit as one of the argument

```
}
```

Implementing custom input format cont'd

```
public class WikiProjectInputFormat extends  
FileInputFormat<Text,IntWritable>{
```

- WikiRecordReader is the custom record reader which needs to be implemented

```
{  
    return new WikiRecordReader();  
}  
}
```

Implementing Record Reader

```
public class WikiRecordReader extends RecordReader<Text, IntWritable>{  
  
    private LineRecordReader lineReader;  
    private Text lineKey;  
    private IntWritable lineValue;  
  
    public WikiRecordReader() {  
        lineReader = new LineRecordReader();  
    }  
  
    @Override  
    public void initialize(InputSplit input, TaskAttemptContext context)  
        throws IOException, InterruptedException {  
        lineReader.initialize(input, context);  
    }  
}
```

Implementing Record Reader cont'd

```
@Override  
public boolean nextKeyValue() throws IOException, InterruptedException {  
    if(!lineReader.nextKeyValue()) {  
        return false;  
    }  
  
    Text value = lineReader.getCurrentValue();  
    String[] splits = value.toString().split(" ");  
    if(splits[0].equals("en")) {  
        lineKey = new Text(splits[1]);  
        lineValue = new IntWritable(Integer.parseInt(splits[2]));  
    } else {  
        lineKey = null;  
        lineValue=null;  
    }  
    return true;  
}
```

Implementing Record Reader cont'd

```
public class WikiRecordReader extends RecordReader<Text, IntWritable>{
```

```
    private LineRecordReader lineReader;  
    private Text lineKey;  
    private IntWritable lineValue;
```

- Extend your class from RecordReader class
- Using existing LineRecordReader class
 - Read line by line record
 - Provides offset as key
 - And entire line as value
- *lineKey* will be the input key to the map function
- *lineValue* will be the input value to the map function

Implementing Record Reader cont'd

```
public class WikiRecordReader extends RecordReader<Text, IntWritable>{
```

```
    private LineRecordReader lineReader;  
    private Text lineKey;  
    private IntWritable lineValue;
```

- Initialize the *lineReader*
 - *lineReader* takes the input split

```
@Override
```

```
public void initialize(InputSplit input, TaskAttemptContext context)  
throws IOException, InterruptedException  
{  
    lineReader.initialize(input, context);  
}
```

Implementing Record Reader cont'd

```
@Override
```

```
public boolean nextKeyValue() throws IOException, InterruptedException  
{
```

- This function provides the input key values to mapper function one at a time

```
    lineKey = new Text(splits[1]);  
    lineValue = new IntWritable(Integer.parseInt(splits[2]));  
} else {  
    lineKey = null;  
    lineValue=null;  
}  
return true;  
}
```

Implementing Record Reader cont'd

```
@Override  
public boolean nextKeyValue() throws IOException, InterruptedException  
{  
    if(!lineReader.nextKeyValue()) {  
        return false;  
    }  
}
```

```
Text value = lineReader.getCurrentValue();  
String[] splits = value.toString().split(" ");  
if(splits[0].equals("en")) {  
    lineKey = new Text(splits[1]);  
    lineValue = new IntWritable(Integer.parseInt(splits[2]));  
}
```

- *lineReader* gives the offset as key and entire line as value
- Once the value is taken
 - It can be split on space
 - Make the split[1] as key and split[2] as value

Using Custom Input Format in Job

- `job.setInputFormat(CustomInputFormat.class)`
- Note while using custom input format, map input key and value arguments should match with the key and value the custom input format provides

Custom Input Format features

- Can override *isSplittable()* function to return false
 - Then files will not be splitted
 - And entire data set will be operated by single mapper

Hands on

- Refer hands-on document

Module 15

Advance MapReduce Concepts – Part 3

In this module you will learn

- Implementing Custom comparator
- Secondary sorting

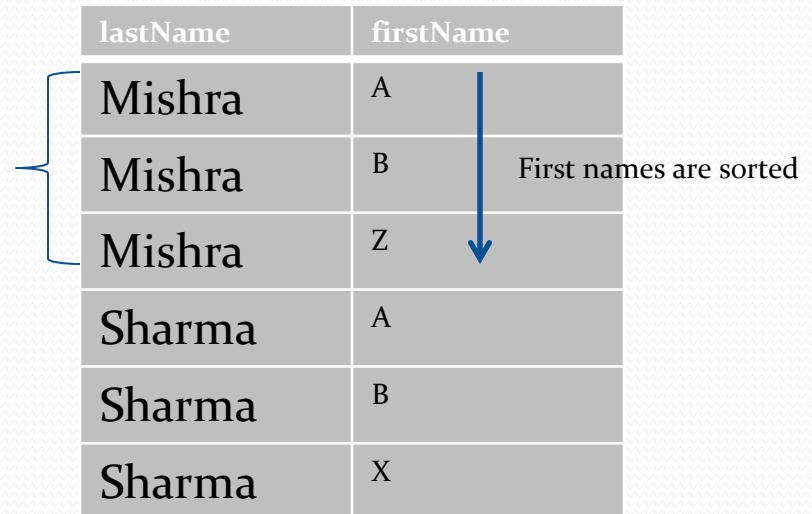
Problem Statement

lastName	firstName
Sharma	X
Sharma	B
Mishra	Z
Sharma	A
Mishra	A
Mishra	B

Last names are grouped together

Problem Statement:

Group all the records having the same last name together and while writing the output the first names should be sorted in ascending order

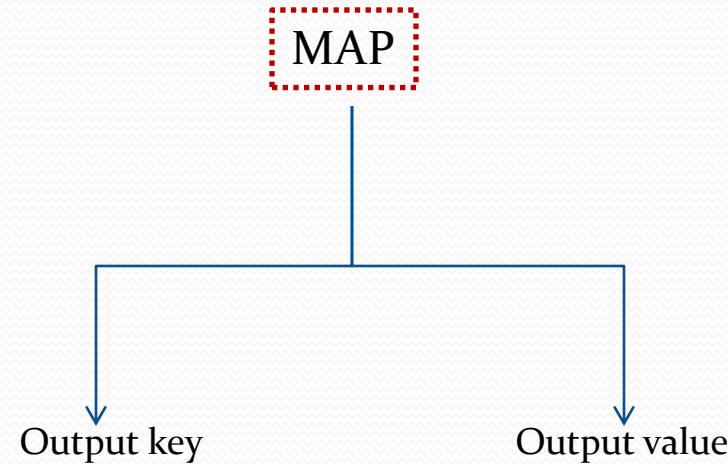


lastName	firstName
Mishra	A
Mishra	B
Mishra	Z
Sharma	A
Sharma	B
Sharma	X

First names are sorted

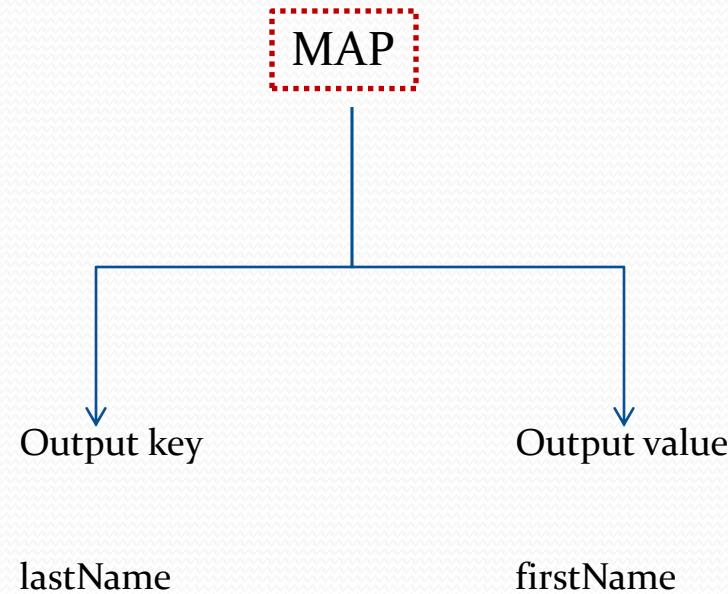
Solution Approach

lastName	firstName
Sharma	X
Sharma	B
Mishra	Z
Sharma	A
Mishra	A
Mishra	B

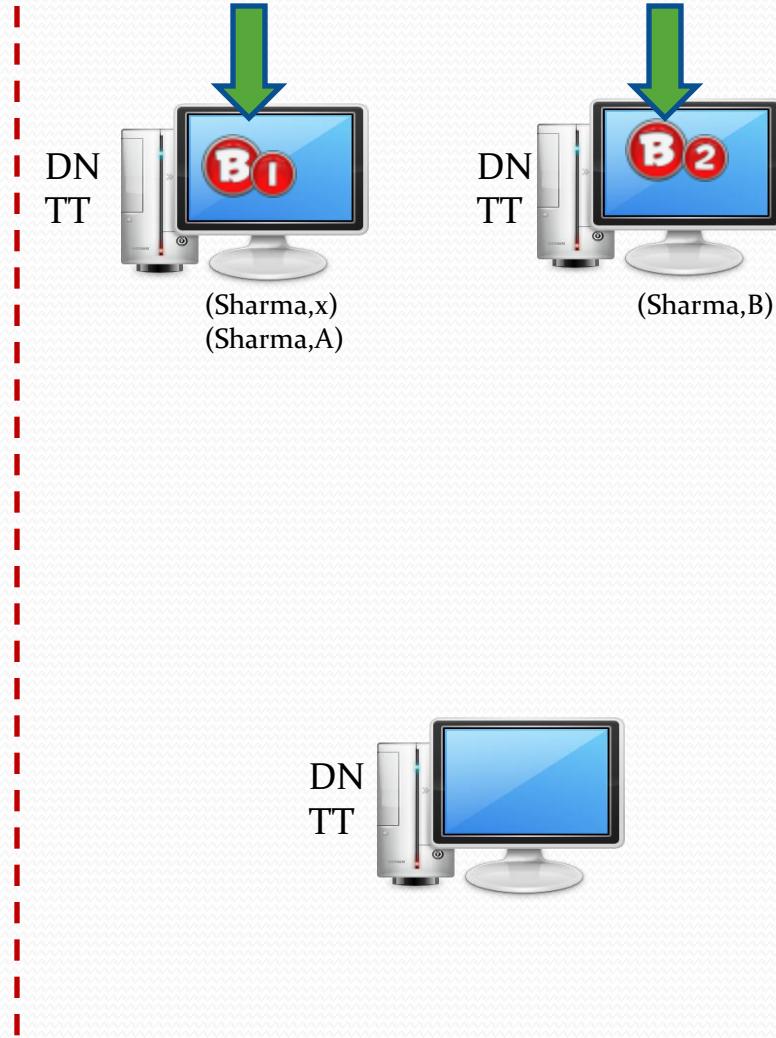
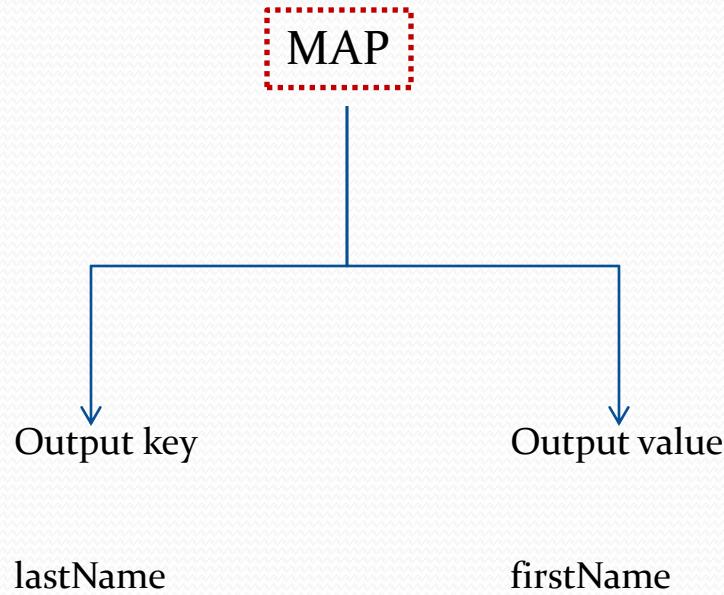


Solution Approach

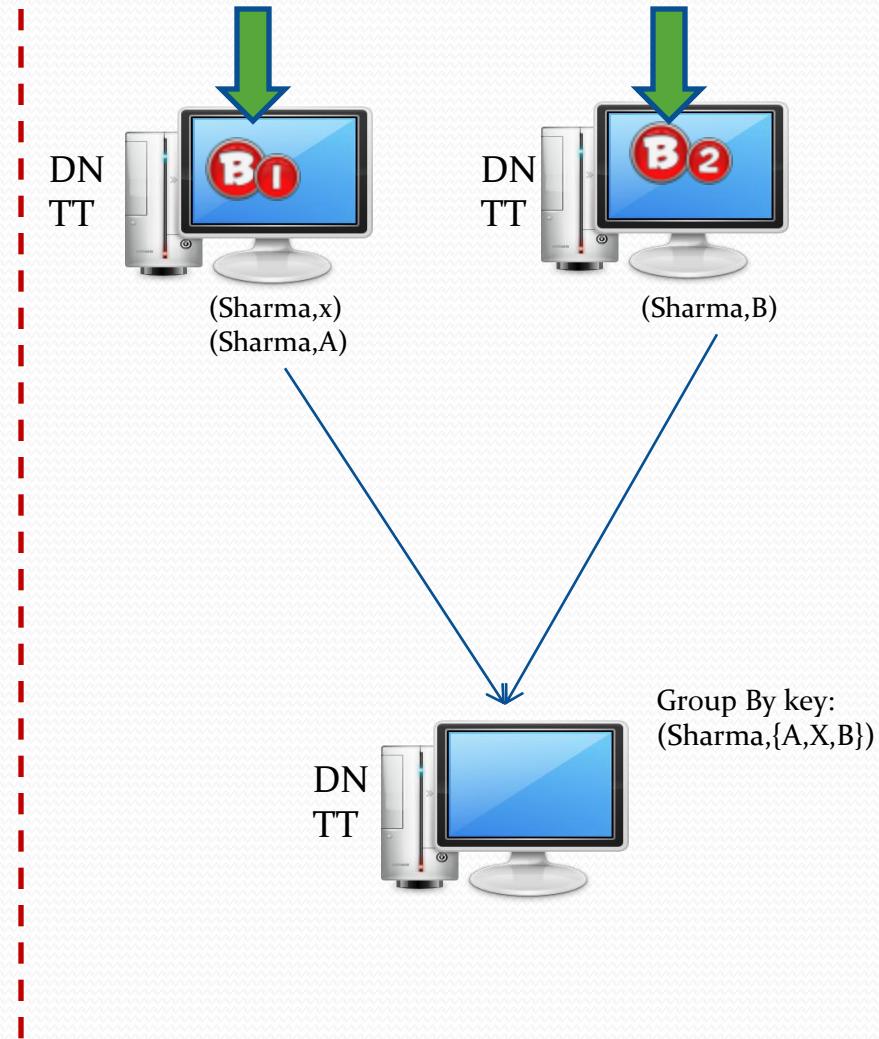
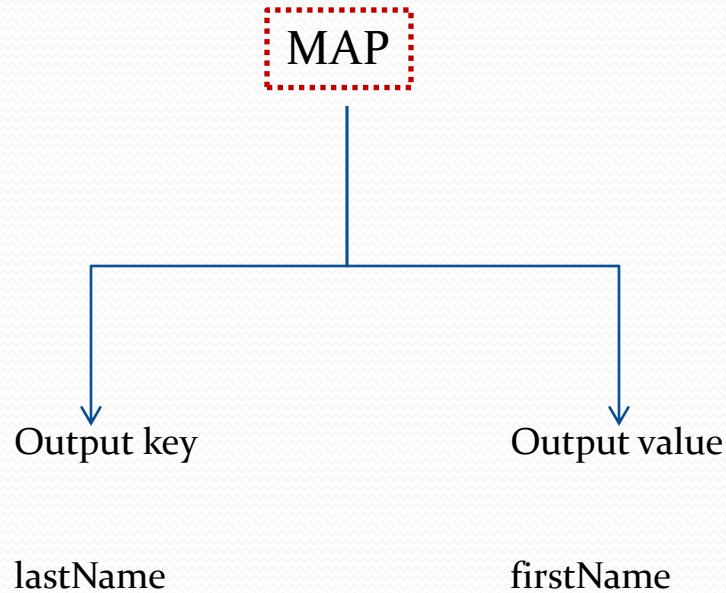
lastName	firstName
Sharma	X
Sharma	B
Mishra	Z
Sharma	A
Mishra	A
Mishra	B



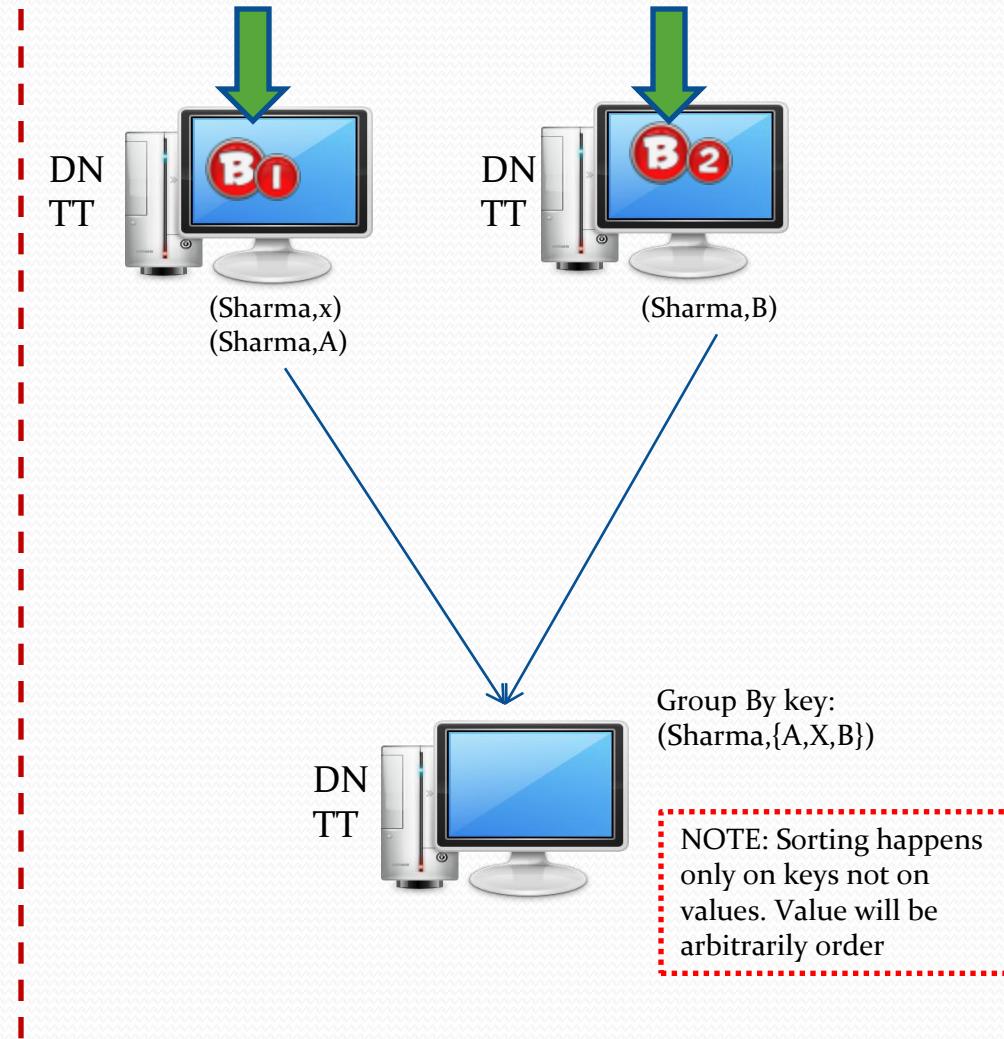
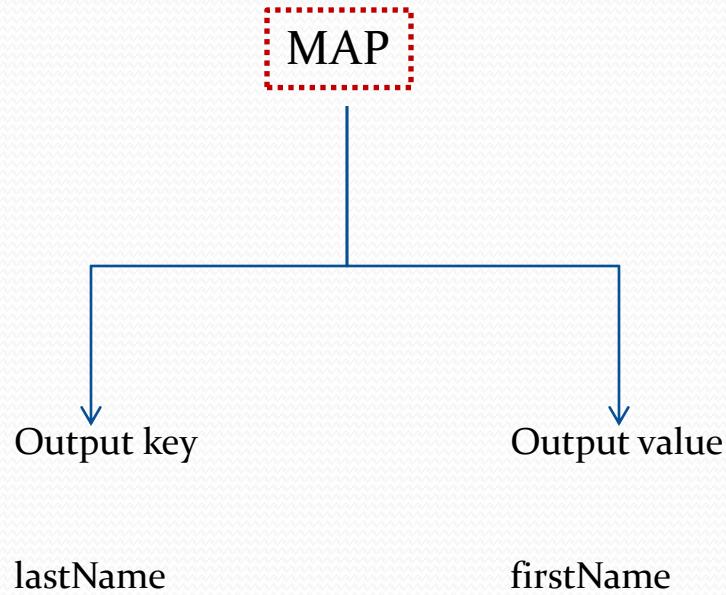
Solution Approach



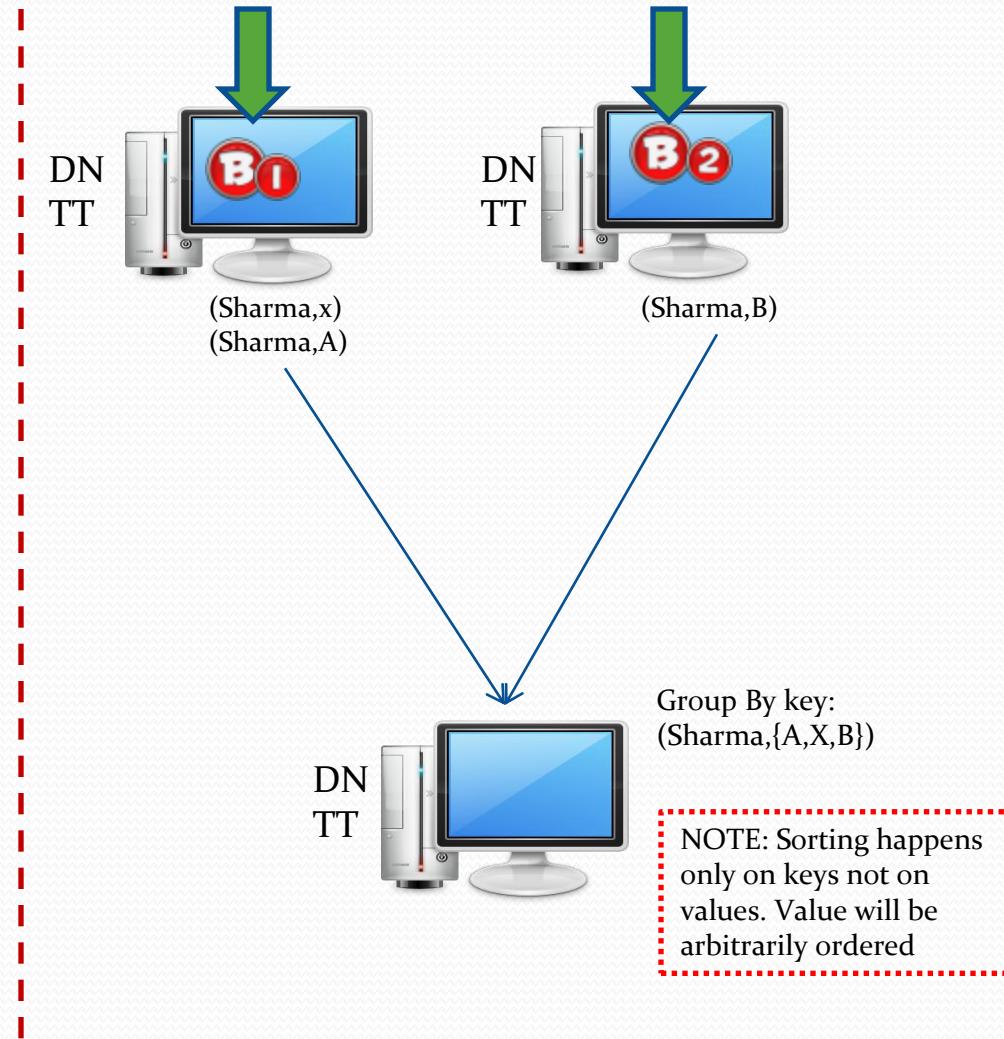
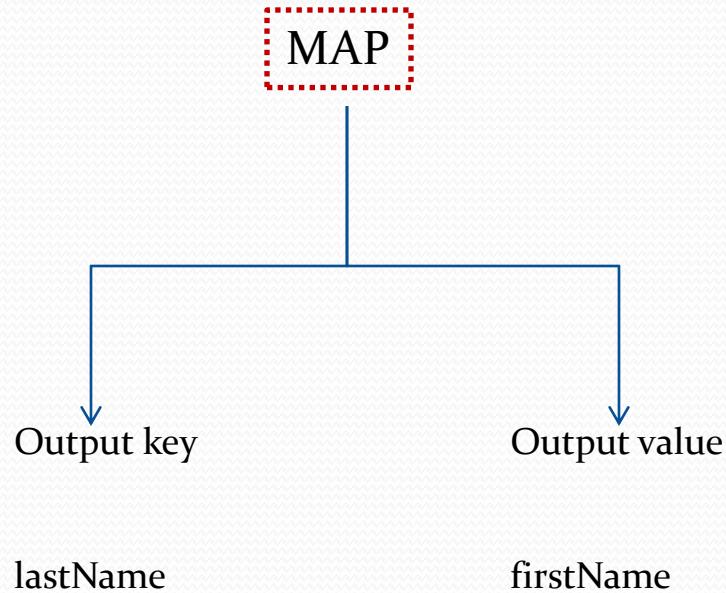
Solution Approach



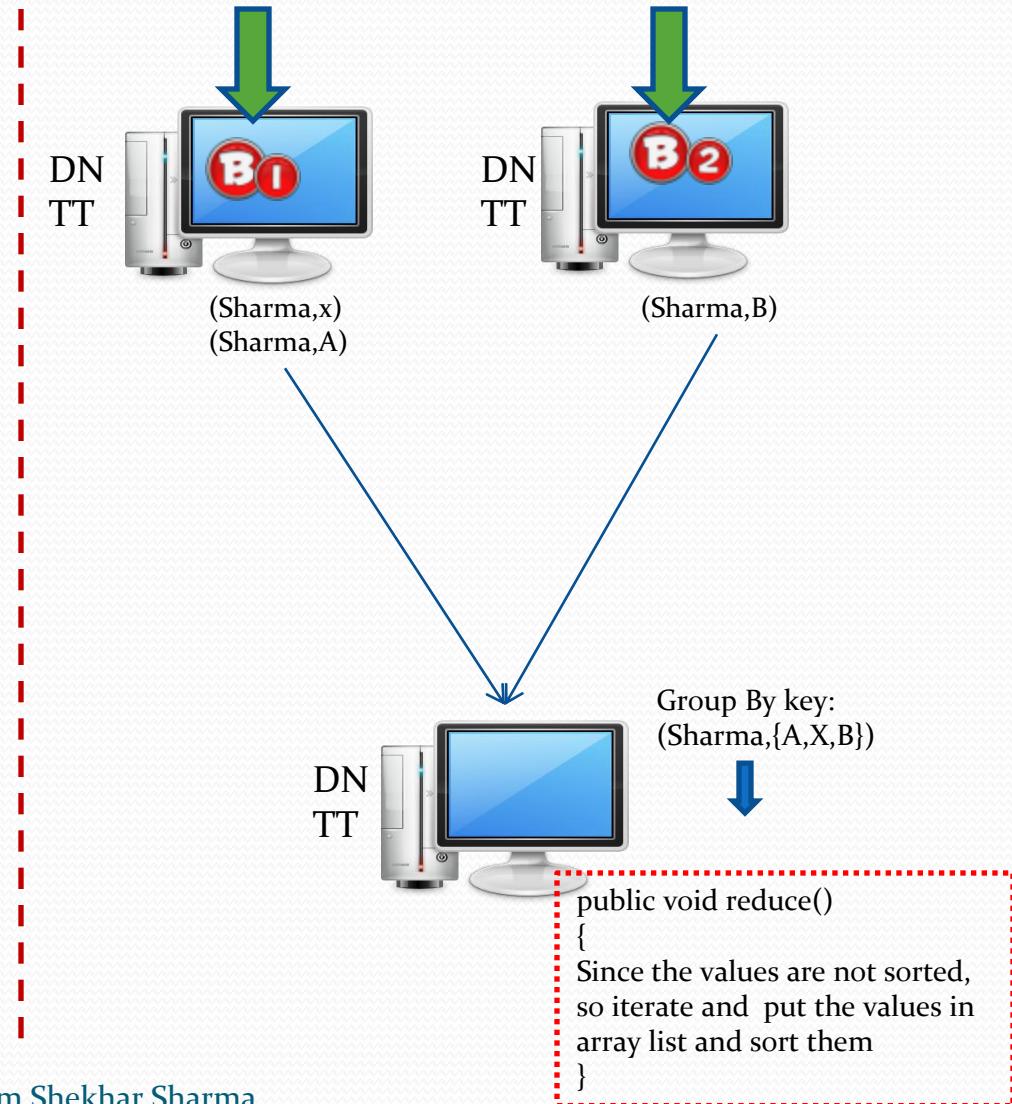
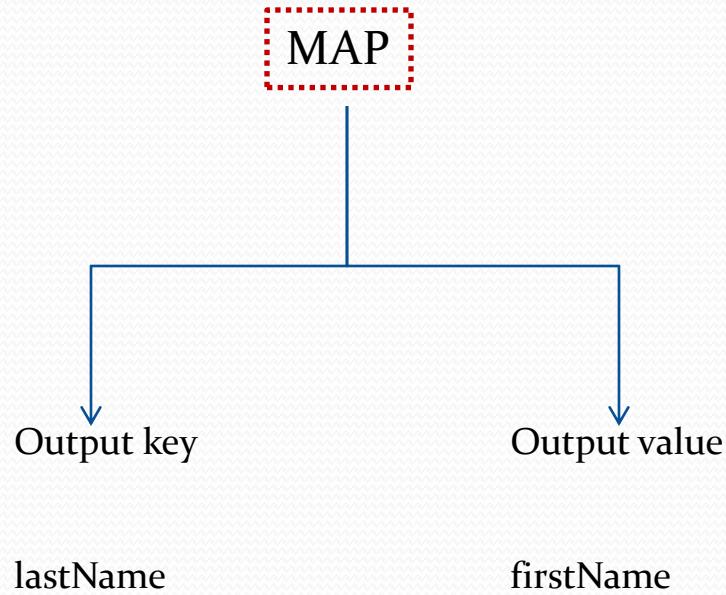
Solution Approach



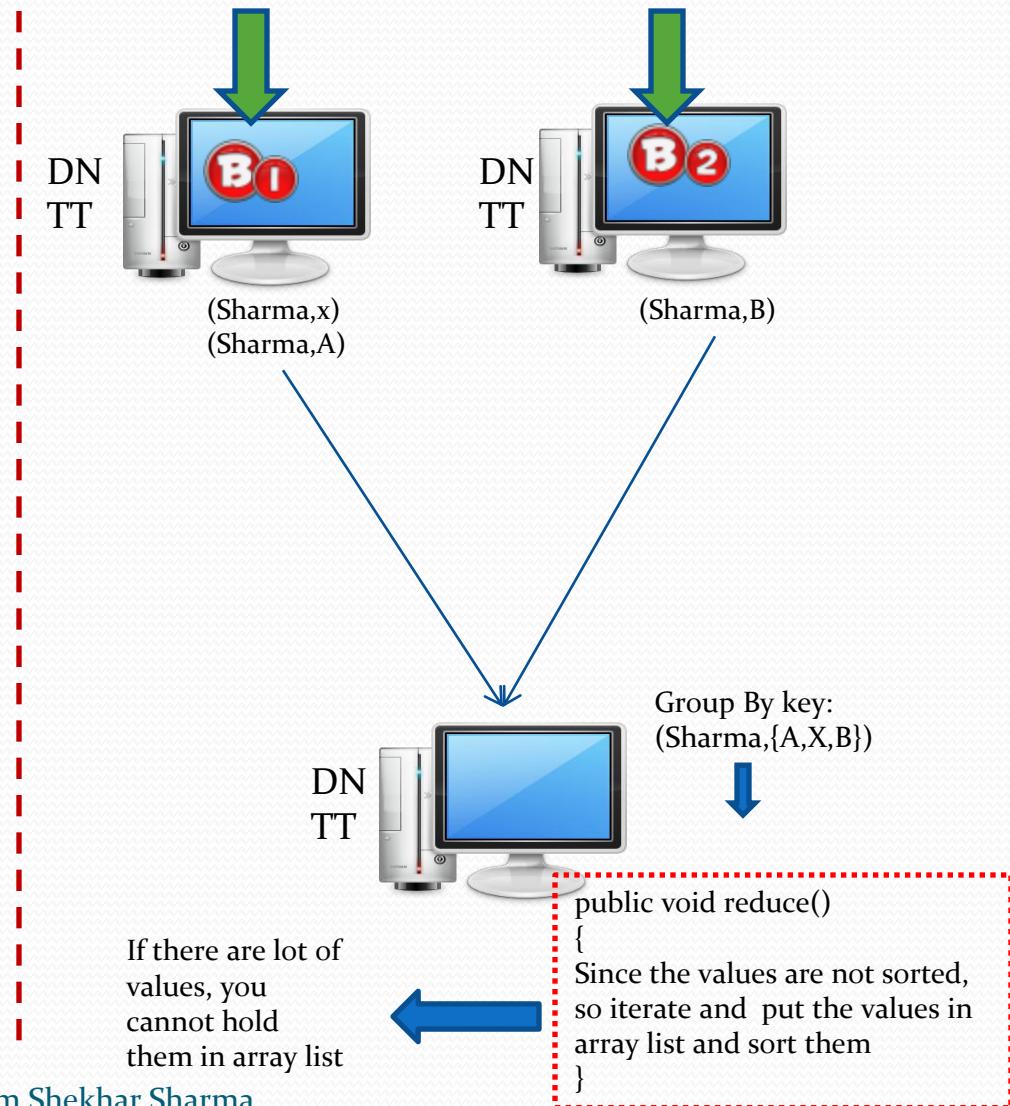
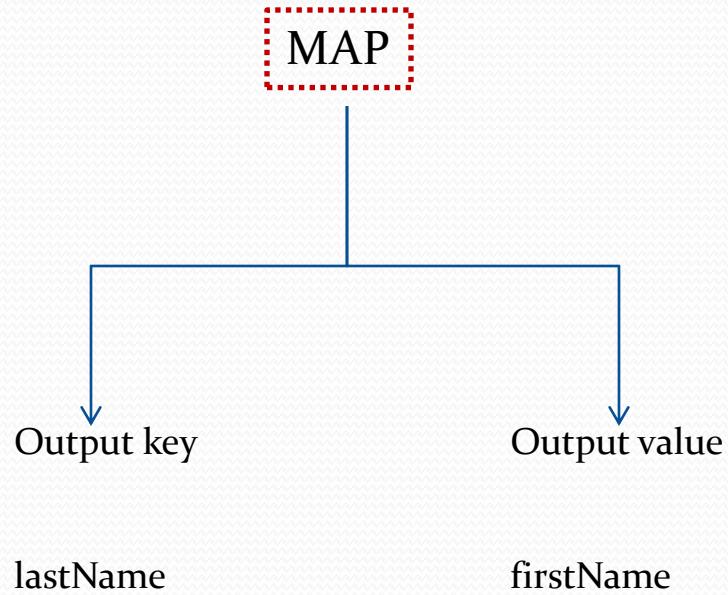
Solution Approach



Solution Approach



Solution Approach

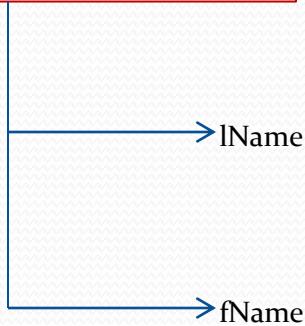


Solution Approach

- Iterating and storing the values in an array list and sorting is not a feasible solution
- Better approach is when we iterate through the list of values we get the values in sorted order
 - The approach eliminates the storing of the values in array list
- Are we not talking about sorting the values?
 - But sorting in MapReduce happens only on keys not on values.?
 - How we are going to sort the values

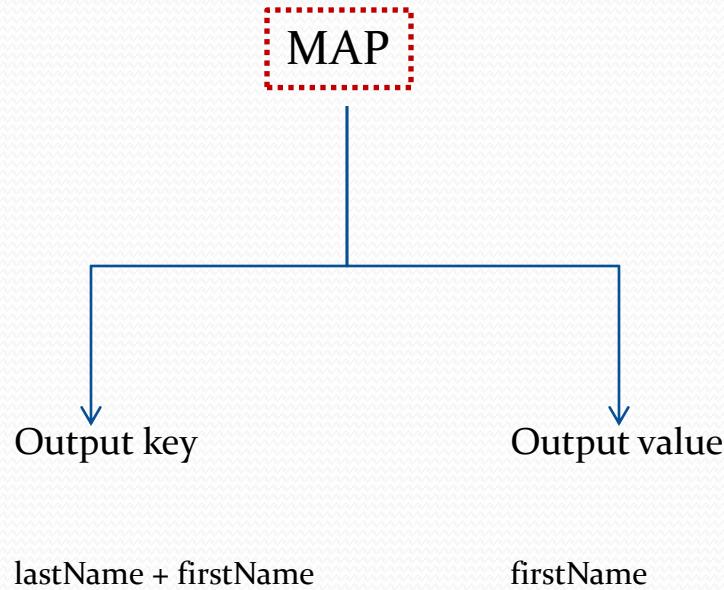
Solution Approach

Custom / Composite Key

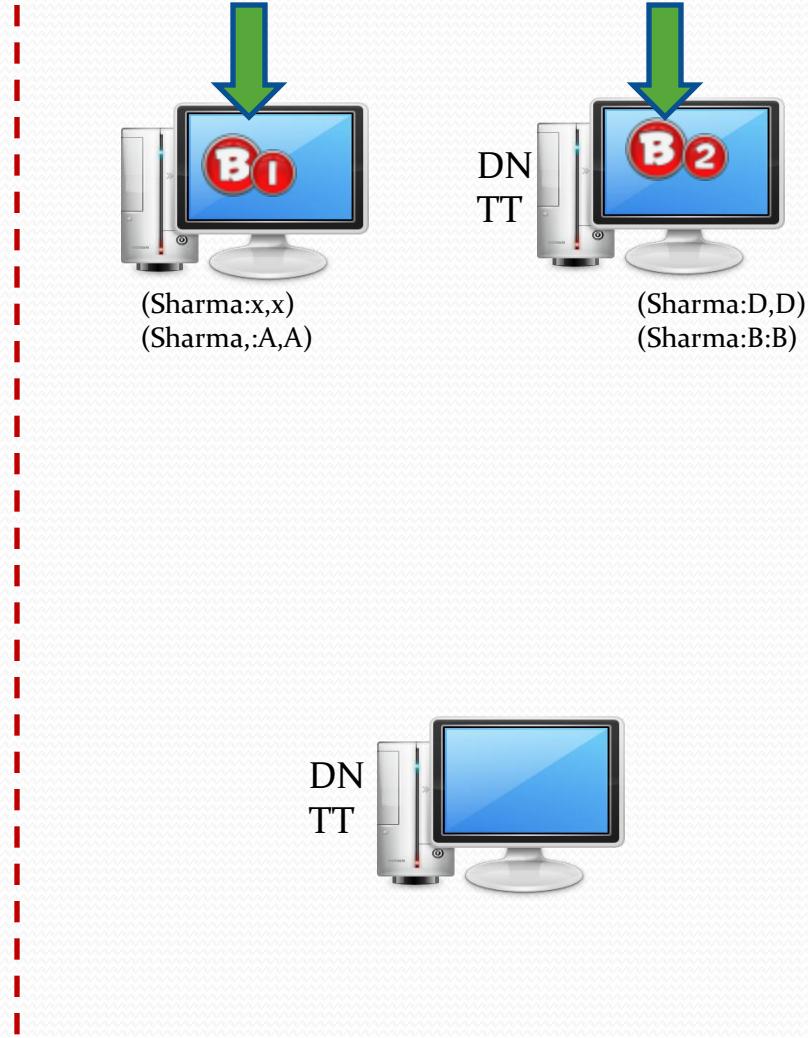
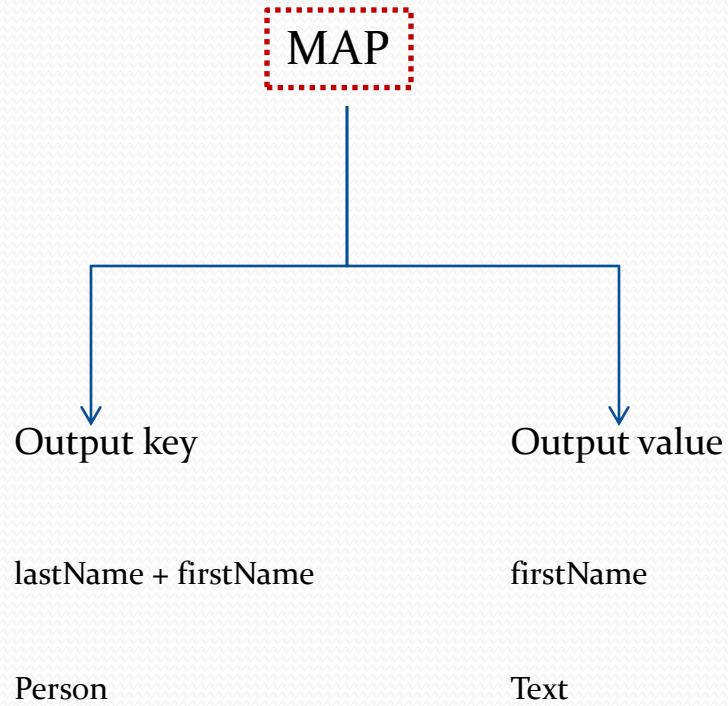


- Write a custom key class by name person
- Create two member variables lName and fName of data type Text which represents last name and first name of the employee respectively
- Define a constructor
 - Initialize the member variables
- Override readFields, write, compareTo methods
 - Leave the compareTo method implementation now
- Define setters and getters method for the two member variables

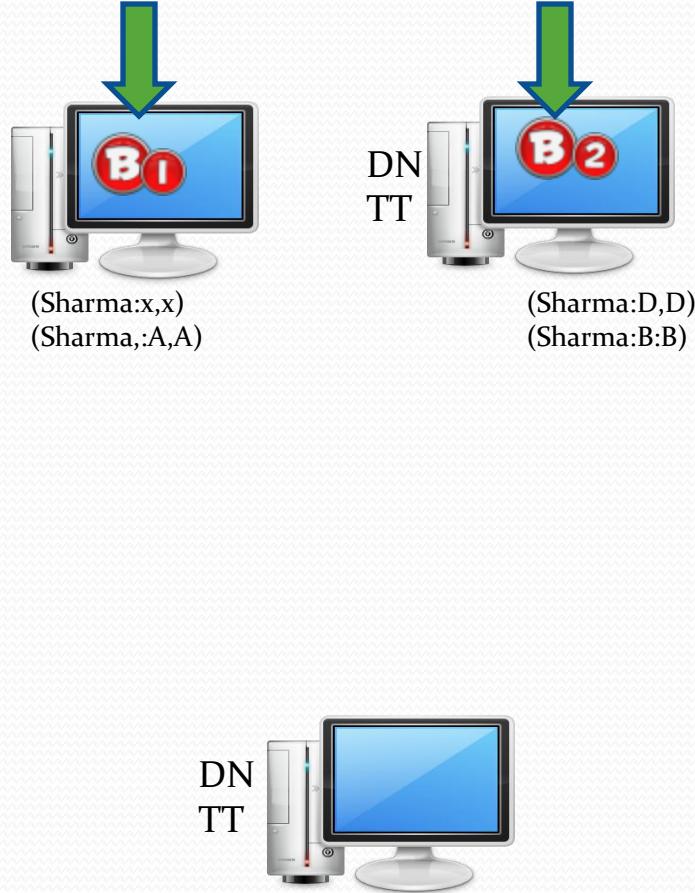
Solution Approach



Solution Approach

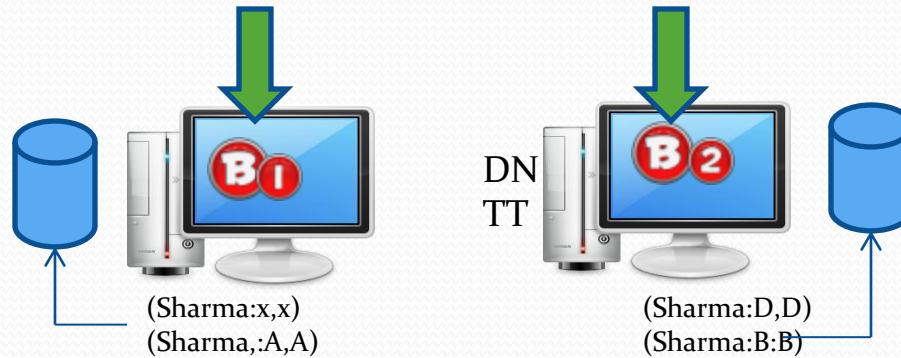


Solution Approach

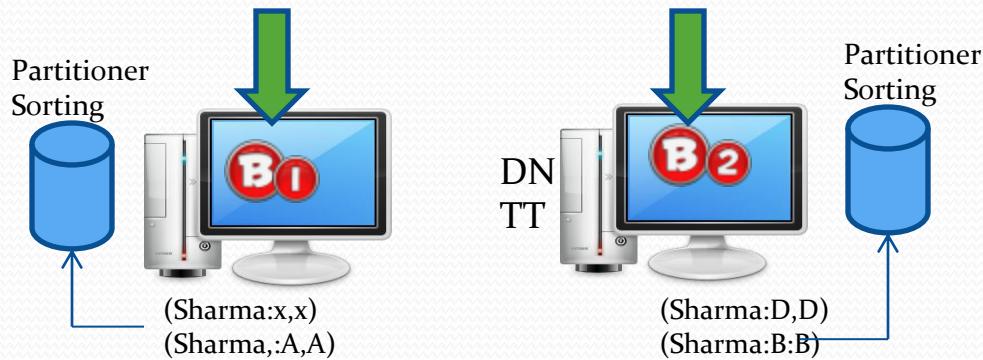


- We will be using KeyValueTextInputFormat
 - Key type= Text and Value Type= Text
 - Key represents last name and value represents first name
- Write a mapper class
 - Input key type = Text
 - Input value type = Text
 - Output key type = Person
 - Output value type = Text
- Override the map function
 - Set the first name and last names to the Person object
 - Emit Person as output key and fName as output value

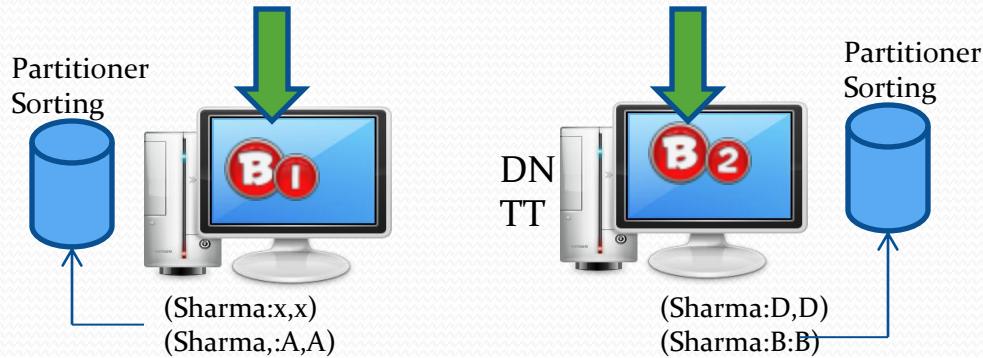
Solution Approach



Solution Approach



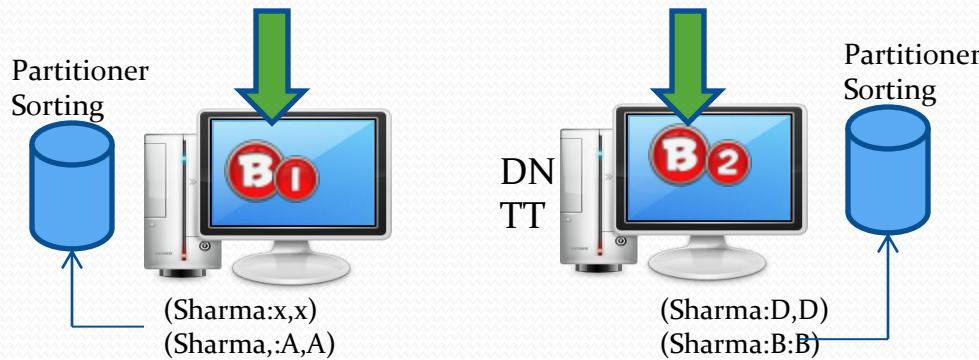
Solution Approach



Will the default Hash partitioner is going to work?



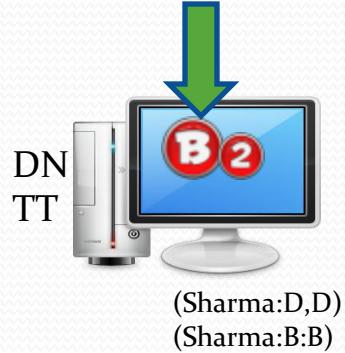
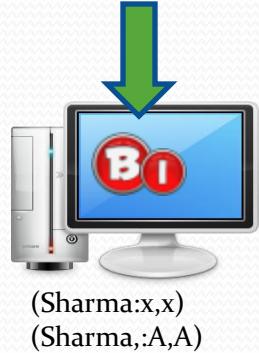
Solution Approach



Write a custom partitioner which partitions on the last name, so that records having the same last name goes to the same reducer

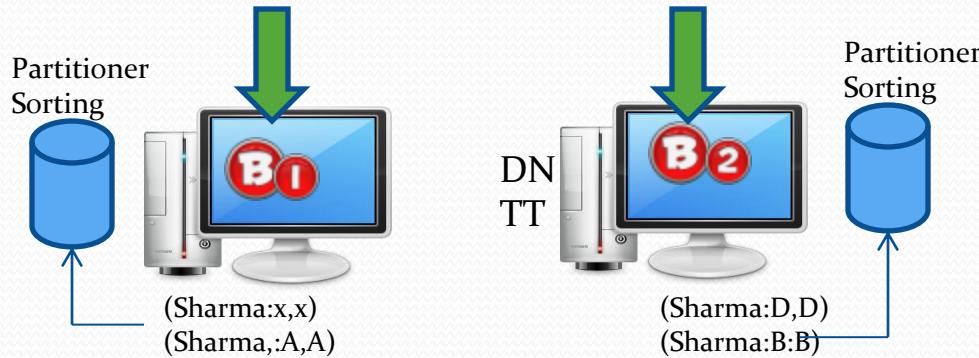


Solution Approach



- Create a class PersonPartitioner
- Override getPartition method
 - Write a logic to partition on last name

Solution Approach

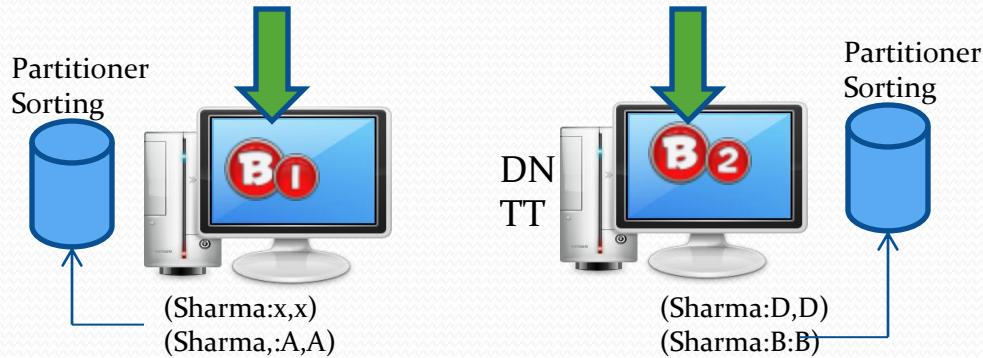


Problem statement: When writing the first names, the first names should be sorted in ascending order

- When (Mishra:z,z) and (Sharma:A,A) are compared, then (Mishra:z,z) should be placed first as “M” comes before “S”
- When (Sharma:x,x) and (Sharma:A,A) key value pairs needs to be arranged, the key value (Sharma:A,A) should be placed first



Solution Approach



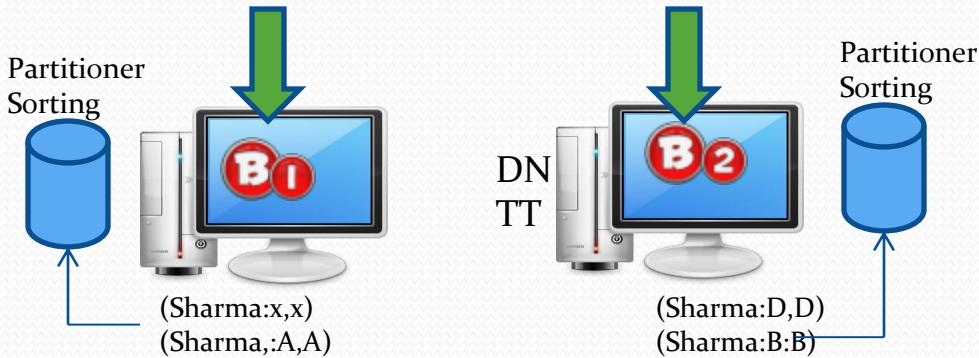
Sorting Logic:

When comparing two keys,

- Compare their last names first
- If the last names are same, then compare their first names



Solution Approach



Sorting Logic:

When comparing two keys,

- Compare their last names first
- If the last names are same, then compare their first names

Custom Key compareTo method implementation:

```
public int compareTo(Person other)
{
    int cmp=this.getLastName().compareTo(other.getLastName());

    if(cmp == 0)
    {
        cmp = this.getFirstName().compareTo(other.getFirstName());
    }

    return cmp;
}
```



Custom comparator

- Comparable & Comparator are interfaces in java – to provide sorting in a collection.
- WritableComparable
 - Hadoop's equivalent interface as Comparable
 - Enable to define only one way of sorting – Ex: Ascending order.

Motivation Custom comparator

Custom Key compareTo method implementation:

```
public int compareTo(Person other)
{
    int cmp=this.getLastName().compareTo(other.getLastName());

    If(cmp == o)
    {
        cmp = this.getFirstName().compareTo(other.getFirstName());
    }

    return cmp;
}
```

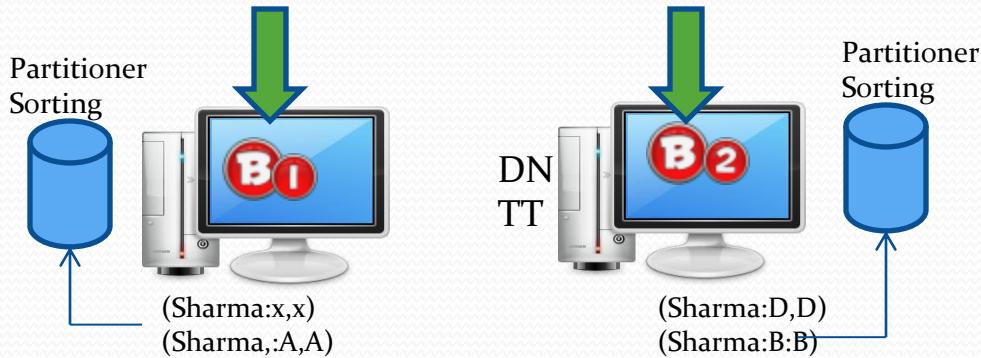
- Objects are de-serialized and then comparison is done
 - If keys are big, then deserialization will be time consuming
- Motivation:
 - We need some mechanism by which we can compare the keys in binary form without de - serializing them to the original object.
 - We also need some mechanism to enable different sorting strategies

Custom Sorting comparator

```
public class PersonSortComparator extends WritableComparator {  
  
    protected PersonSortComparator () {  
        super(Person.class,true);  
    }  
  
    @Override  
    public int compare(WritableComparable o1, WritableComparable o2) {  
        Person e1 = (Person ) o1;  
        Person e2 = (Person ) o2;  
        int comparison = e1.getLastName().compareTo(e2.getLastName());  
        if(comparison == 0)  
        {  
            comparison = e1.getFirstName().compareTo(e2.getFirstName());  
        }  
        return comparison;  
    }  
}
```

To use this comparator, in the driver code specify : job.setSortComparator(PersonSortComparator.class);

Solution Approach



Sorting Logic:

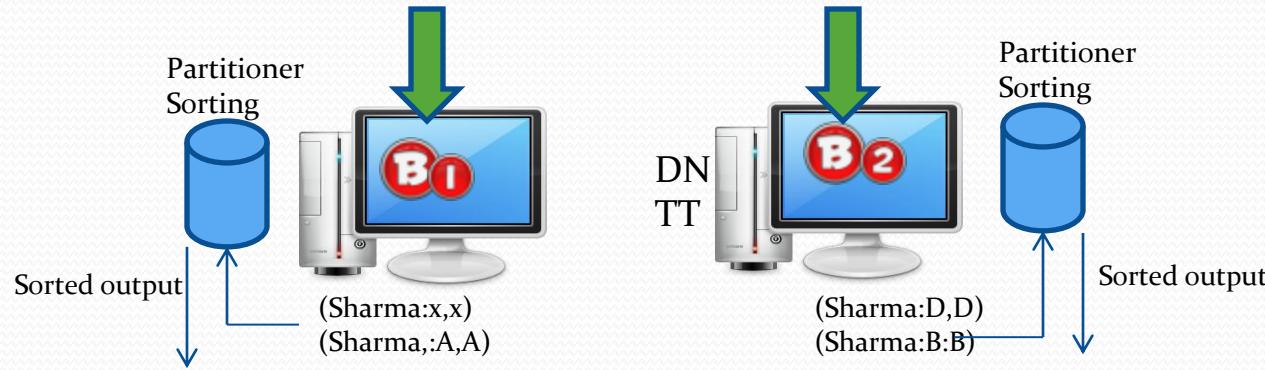
When comparing two keys,

- Compare their last names first
- If the last names are same, then compare their first names

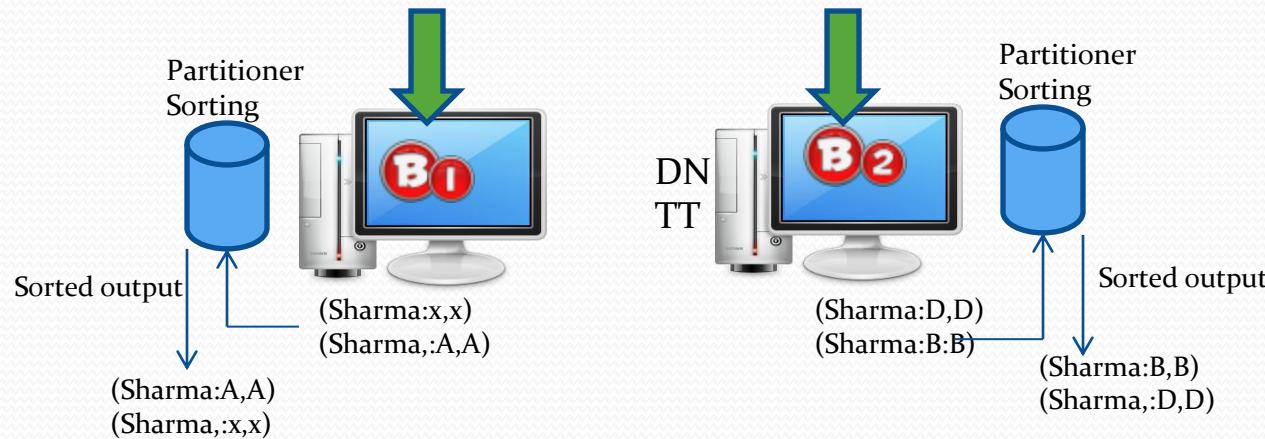
Write the comparator class, as defined earlier and embed the logic



Solution Approach



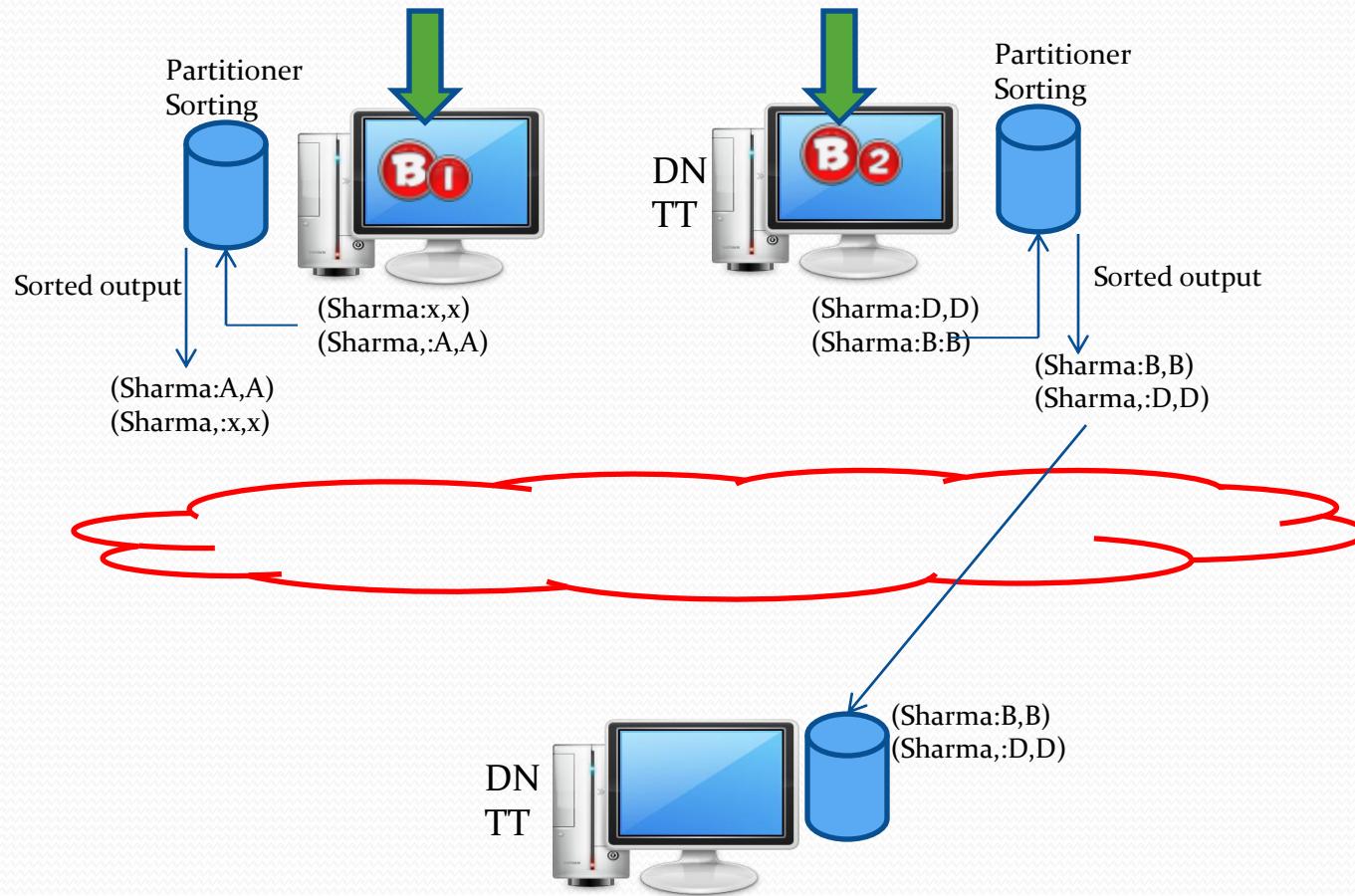
Solution Approach



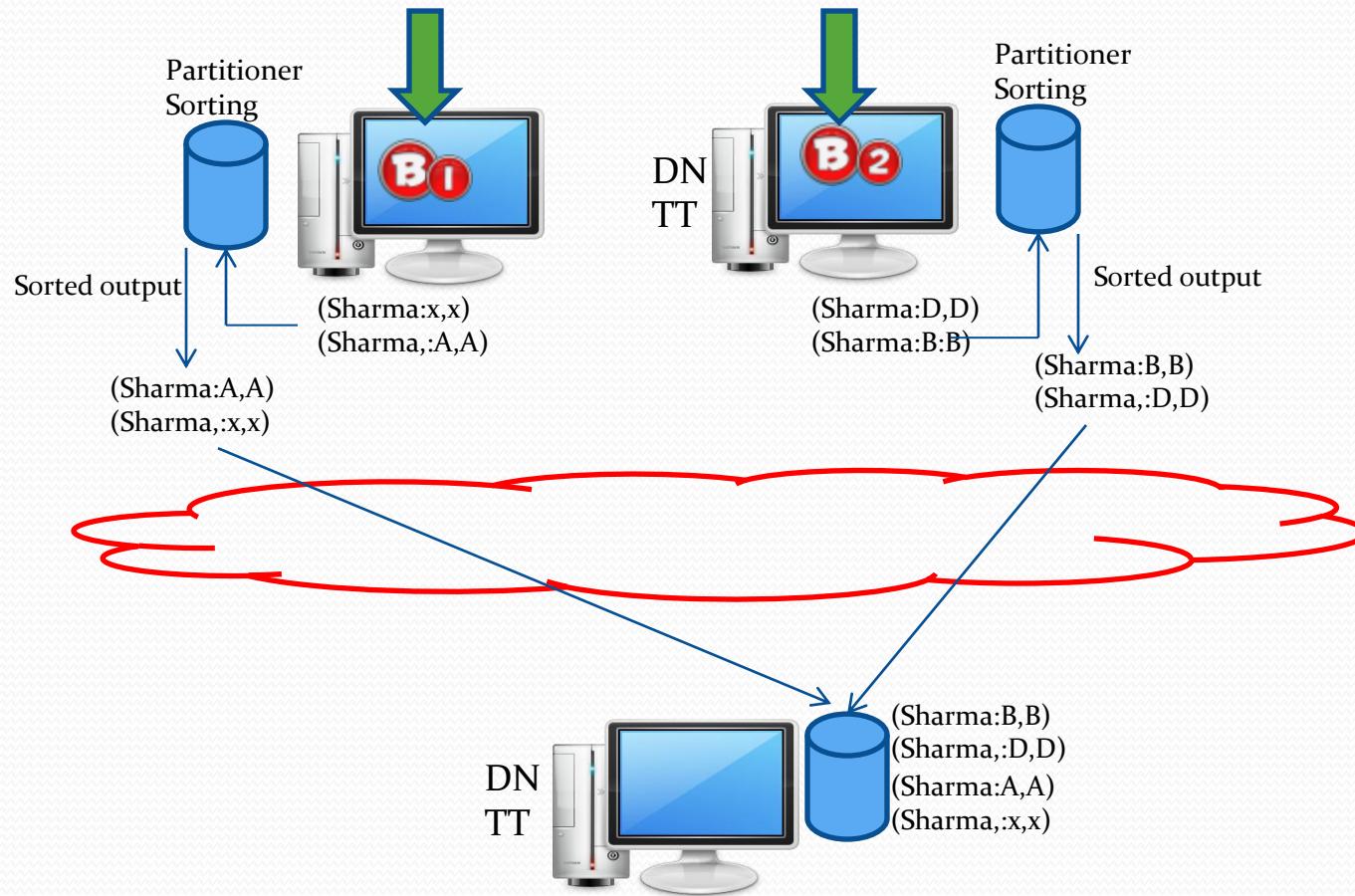
Once any of the map task is over shuffling process will start



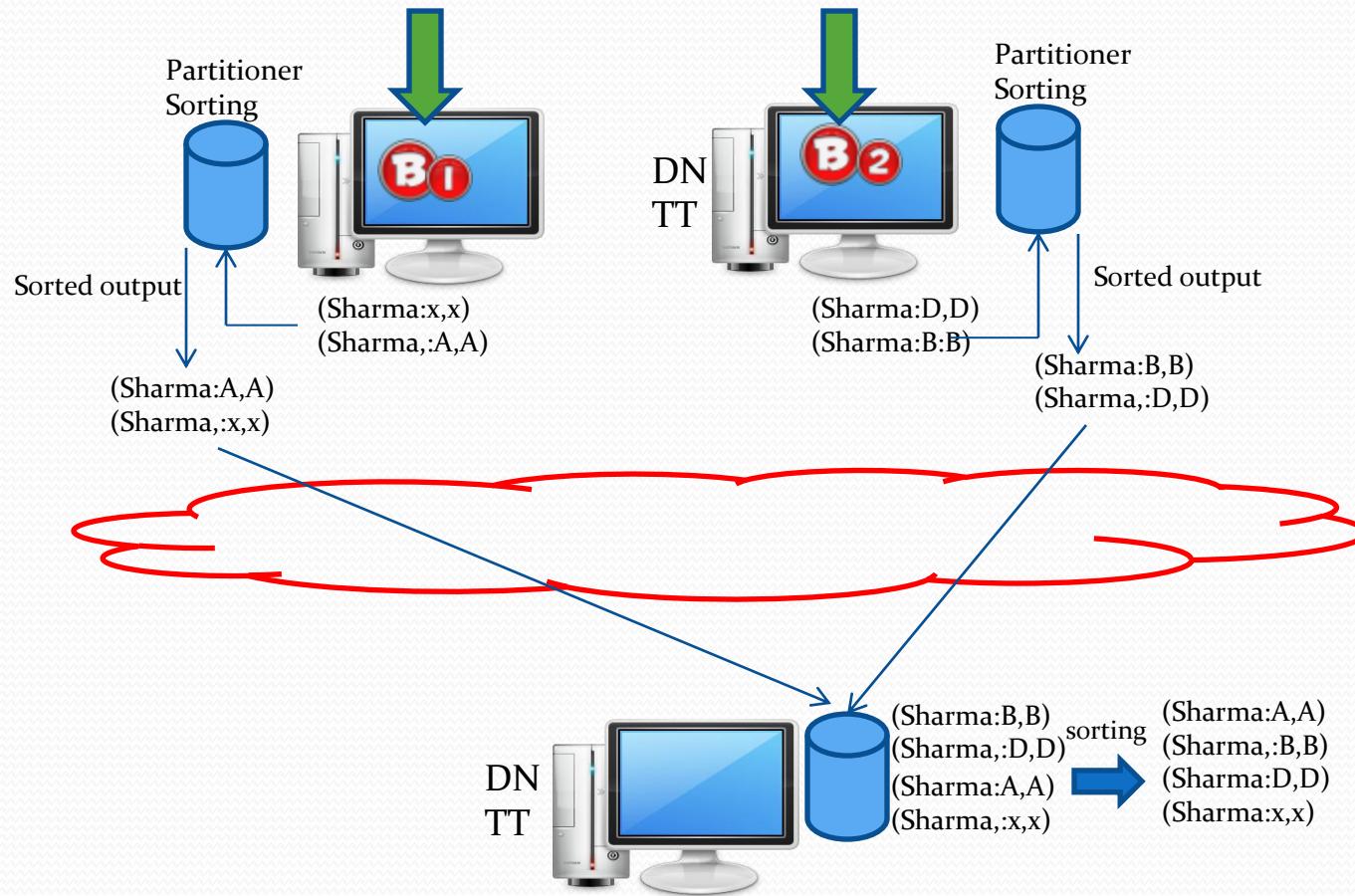
Solution Approach



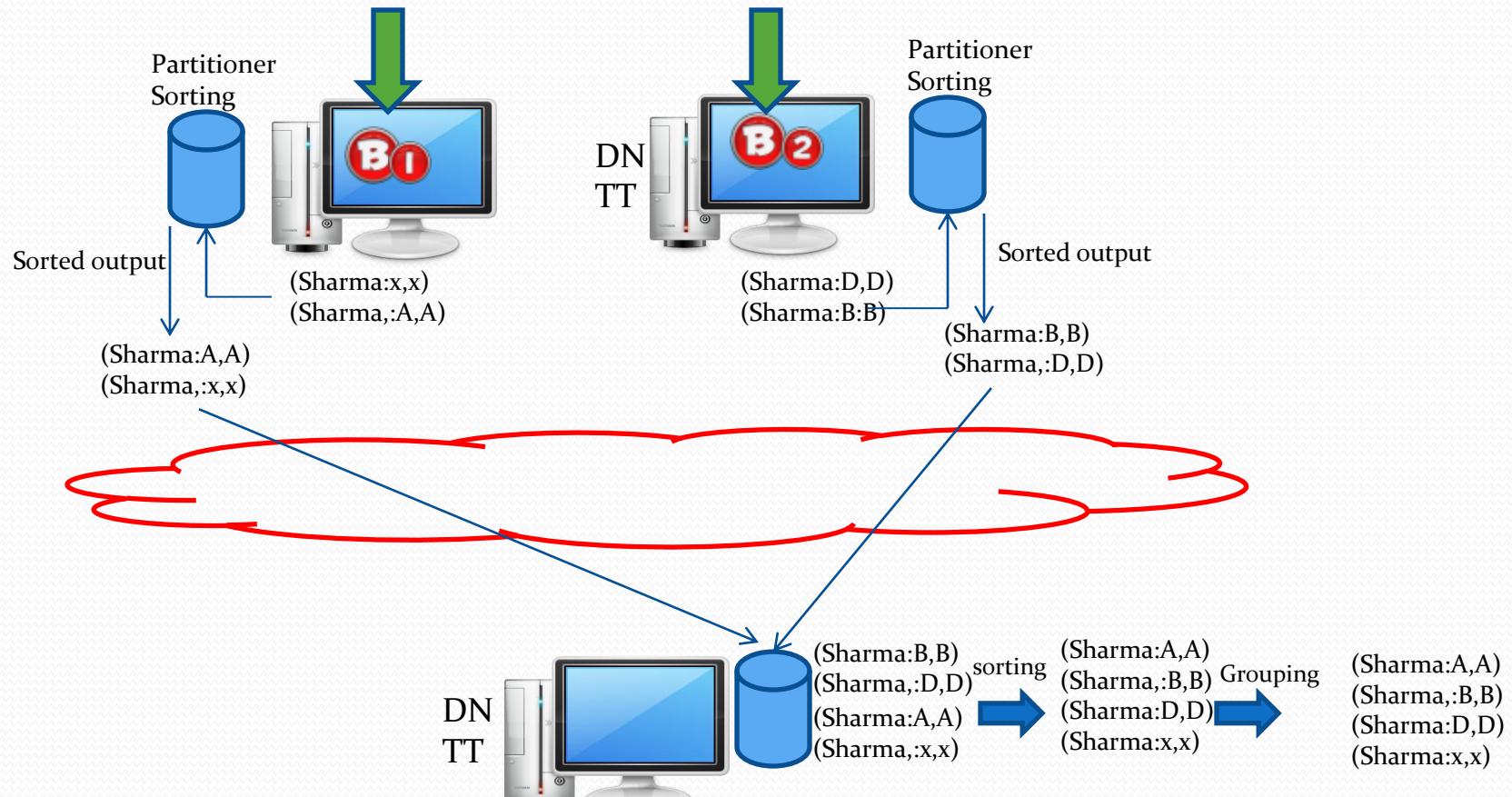
Solution Approach



Solution Approach



Solution Approach



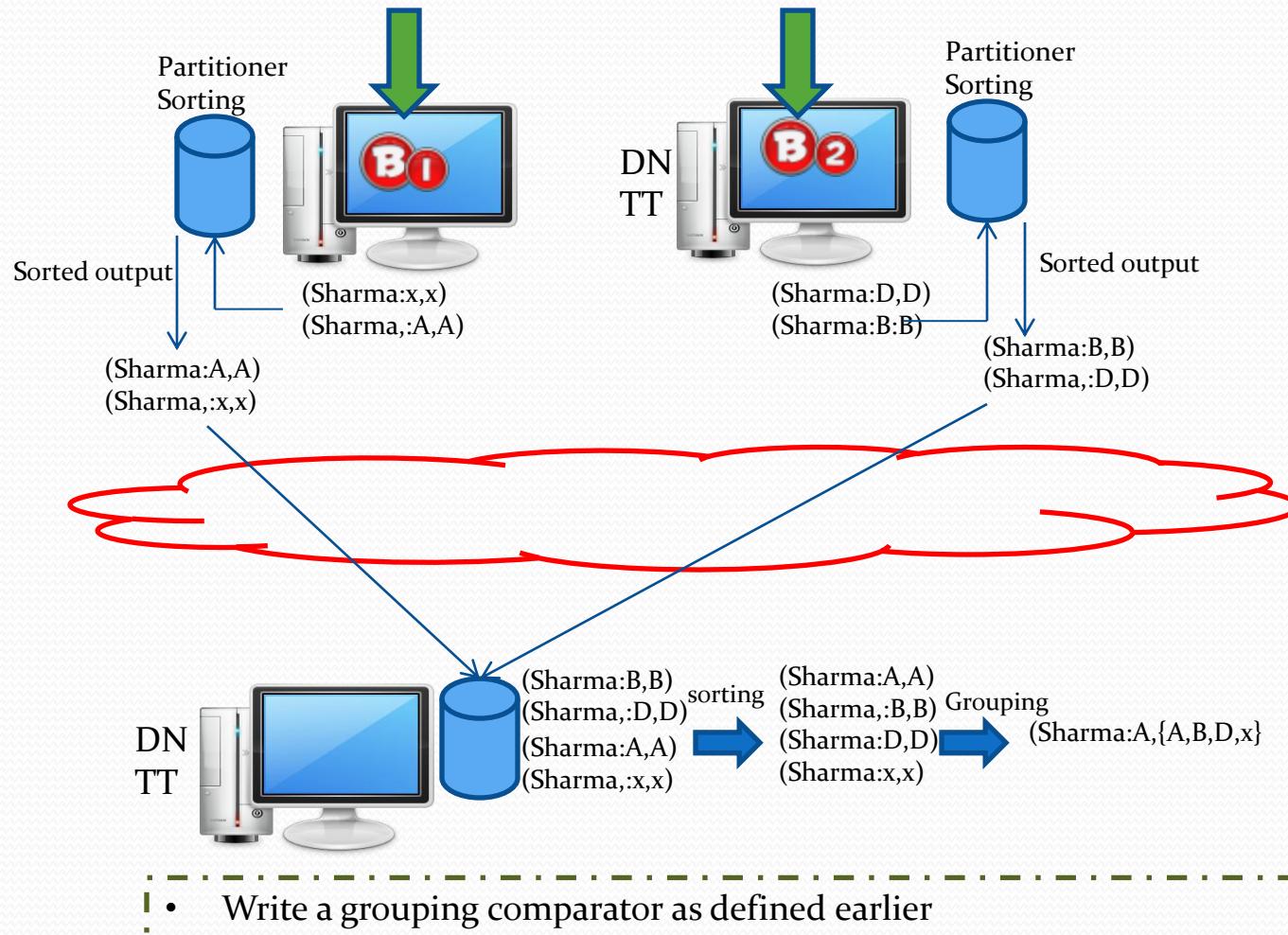
- Do we need the output of grouping operation like this?
- Or do we need $(\text{Sharma}:\{\text{A,B,D,X}\})$?

Custom Grouping comparator

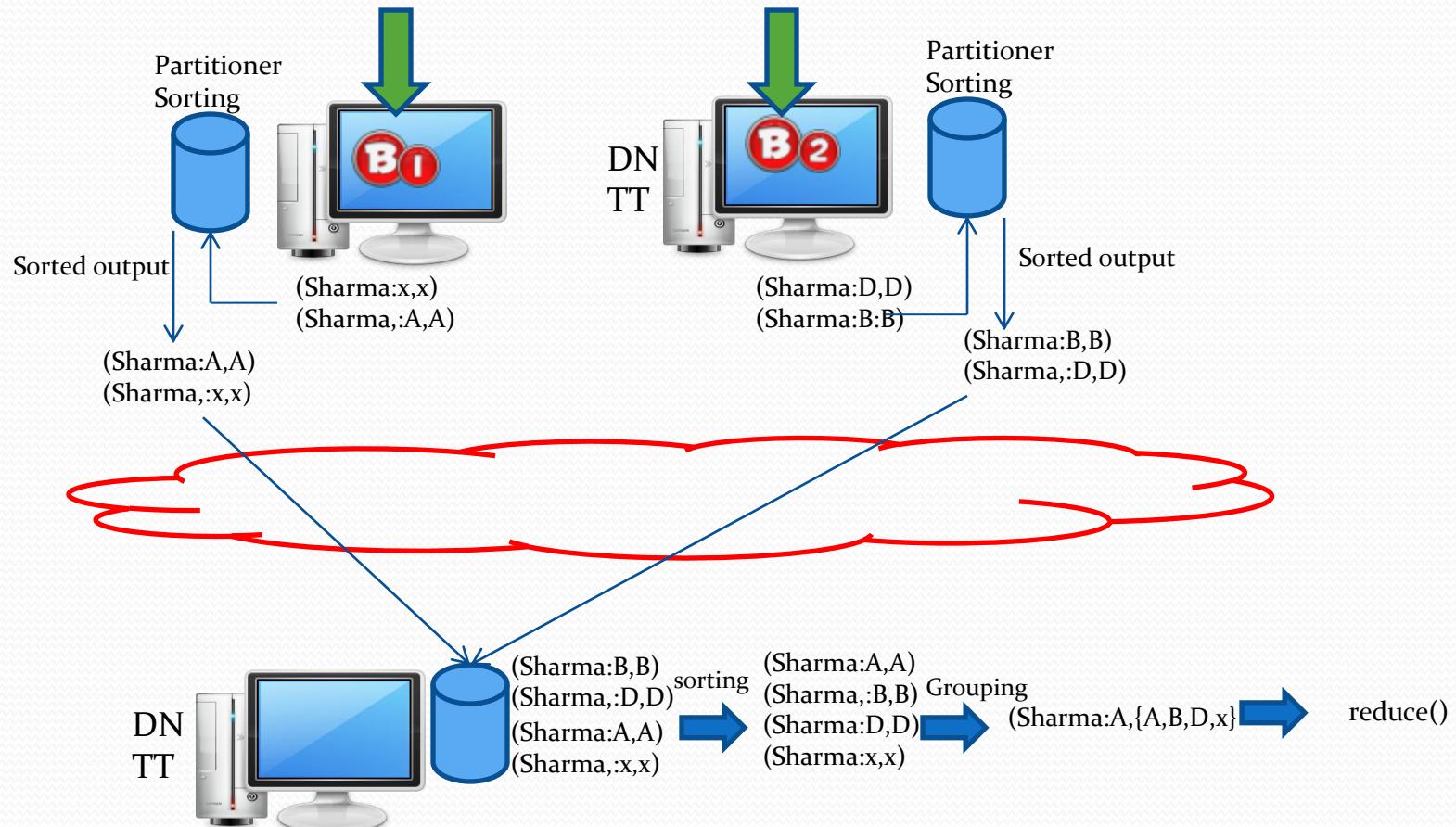
```
public class PersonGroupingComparator extends WritableComparator {  
    protected PersonGroupingComparator () {  
        super(Person.class,true);  
    }  
  
    @Override  
    public int compare(WritableComparable o1, WritableComparable o2) {  
        Person e1 = (Person ) o1;  
        Person e2 = (Person ) o2;  
        int comparison = e1.getLastName().compareTo(e2.getLastName());  
        return comparison;  
    }  
}
```

To use this grouping comparator, in the driver code specify :
job.setGroupingComparator(PersonGroupingComparator.class);

Solution Approach



Solution Approach



- Write a reducer class that just emits the last name as key and first name as value
- Observe the output that first names are sorted

Secondary Sorting

- Method for sorting the values
- Following steps are required
 - Custom key implementation
 - Custom partitioner implementation
 - Custom sorting comparator implementation
 - Custom grouping comparator implementation

Secondary Sorting- Hands On

- Custom key implementation
 - Person class
- Custom partitioner implementation
 - PersonPartitioner class
- Custom sorting comparator implementation
 - PersonSortingComparator class
- Custom grouping comparator implementation
 - PersonGroupingComparator class
- PersonMapper class
- PersonReducer class
- PersonDriver class

Hands On

- Refer hands-on document

Module 16

Joining Datasets in Map Reduce

In this module you will learn

- What are joins?
- What is map side join?
- What is reduce side join?
- Hands on

What are joins?

- Joining means joining two data sets on a common key
- Employee Table

Name	ID	Country Code
James	01	AUS
Siddharth	11	IN
Suman	23	US

Country Table

CountryCode	Country Name
AUS	Australia
IN	India
US	United States

Joins cont'd

- Your job is to replace the country code in the employee table with country names
- Output should be

Name	ID	Country
James	01	Australia
Siddharth	11	India
Suman	23	UnitedStates

Joins cont'd

- MapReduce provides two types of join
 - Map Side join
 - Reduce Side join
- Both kind of joins does the join but differs the way it is done
 - Map Side join is done at mapper phase. No Reducer is required
 - Reduce side join requires to have reducer

Map Side Join

- Use this join, if one of the data sets you are joining can be fit into memory
 - In the previous example if “*CountryTable*” can be fit into memory, then do the joining during mapper phase
- Map Side join is done in memory, and hence it is fast
- Use Distributed Cache facility and cache the “*CountryTable*” and use this in map function

Map Side Join cont'd

```
Public class MapSideMapper extends Mapper<>
{
    public void setup(Context context)
    {
        Step1: Get the CountryTable from the distributed cache
        Step2: You can populate a hash table where key is country code
               and value is country name
    }
    public void map(...)
    {
        Step 1: Read the record and get the country code
        Step 2: For the given country code get the country name from the
               hash table you populated in set up method above and
               replace in the record
        Step 3: Write the modified record using context object
    }
}
```

Map Side join cont'd

- This join is faster
 - Hash table is in memory and for every record just a look up is required
 - Use when dataset is small. (Can fit in memory)
- It is not scalable
 - Memory is constraint
 - What if the *CountryTable* size is 100GB?

Reduce side join

- Use case : Joining two datasets sharing one common field in reducer phase.
- Motivation : Map side join relies heavily on memory, so when datasets are very large, Reduce side join is the only option.
- Example:

Employee		
EMPID	Name	Location
42	John	13



Location	
LocationId	LocationName
13	New York



EMPID	Name	Location	LocationName
42	John	13	New York

Reduce side join contd..

- Steps:
 - Run mapper on both datasets and emit key which is the common fields and value as the entire record. Ex:

```
M1 { 13 : 42 John 13 }
M2 { 13 : New York
```

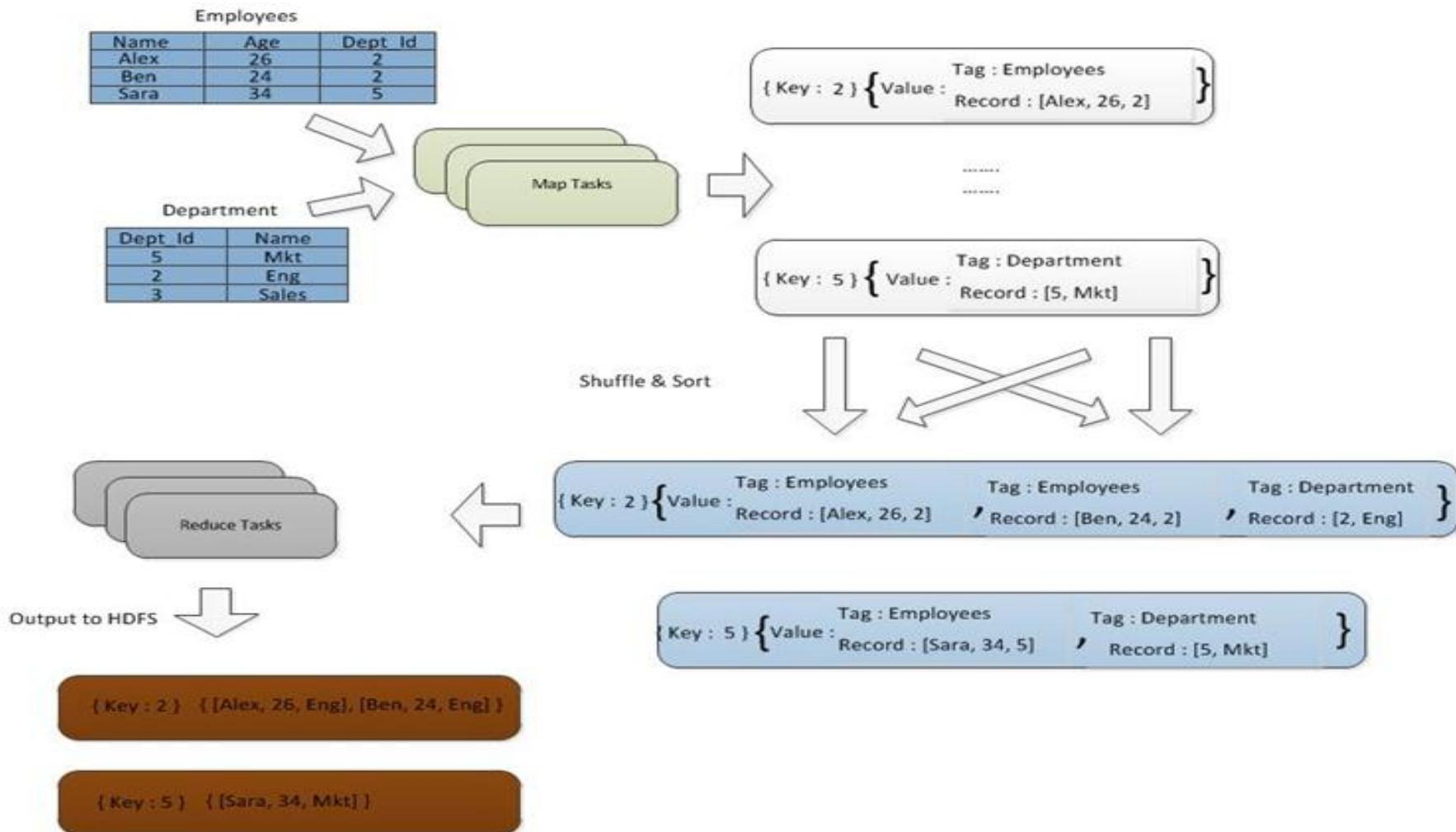
- Now, join the data in Reducer (all values for the same key will be passed to the same reducer)
- Is there any problem here ?

Reduce side join contd..

- Problem: For larger datasets one value need to be joined with several other values (and values are not sorted!). Ex: location with employee

Hint – use secondary sort to make sure that the joining key appears first on reducer

Reduce Side Join cont'd



Hands On

- Refer hands-on document

Module 17

Practical Development Tips

In this module you will learn

- Deciding number of mappers and reducers
- Map only jobs
- Compression

Deciding Number of Mappers

- By default total number of mappers running will be equal to total number of blocks
- If you have 1000 blocks then 1000 map task will run
 - If you have 50 machines and each machine is dual core then total of 100 (50 machines * 2 cores) tasks can run parallelly.
 - Scheduling 1000 maps will be time consuming
 - You can merge the files and make it a bigger files

NOTE: Lot of smaller files means lot of tasks

Deciding Number of Reducer

- By default only one reducer
- Running one reducer could be very performance bottleneck
 - Copying will take time
 - Sorting will take time
 - Reducer will be slow
- Can run more than one reduce, but the output will not

Compression

- Compressing the intermediate key value pairs from the mapper
- Use LZO / Snappy compression

Module 18

Use Case – Data Mining on Wikipedia Data Set

In this module you will learn

- How to perform simple data mining using Map Reduce

Understanding Wikipedia data set

- Download wikipedia data set from
<http://dammit.lt/wikistats/>
- Its an hourly data set
- Structure data with four columns
 - Project Name
 - Page Name
 - Page Count
 - Page size

Wikipedia Data set cont'd

- Sample data in file

en	yahoo	12	1456
en	google	13	234
aa . b	fb	13	1453
dd	rediff	14	1897

- Page Name Appears only once in a file
- Example:
 - Yahoo page has been watched 12 times in that hour

- First column is project name
- Second column is page name
- Third column is page count
- Fourth column is page size

Analyzing Wikipedia data set- Part 1

- Find Top 10 English project from a set of Wikipedia files

Analyzing Wikipedia data set- Part 1 cont'd

- Approach:
- Use 2 MapReduce job
- Map Reduce Job 1
 - Use Text Input Format
 - In mapper filter the project belonging to english project
 - In reducer calculate the sum of page count
- Map Reduce job 2
 - Take the output of previous map reduce job
 - Do the sorting (secondary sorting) and arrange in descending order

Analyzing Wikipedia data set-Part 2

- Find top-10 English project sites?

Analyzing Wikipedia data set- Part 2 cont'd

- Approach:
- Use 2 MapReduce job
- Map Reduce Job 1
 - Use Custom input format and send only English project to your mapper
 - From mapper send page name and page hit/count
 - In reducer calculate the sum of page count
- Map Reduce job 2
- Take the output of previous map reduce job
- Do the sorting (secondary sorting) and arrange in descending order

Module 19

Hive & Pig

In this module you will learn

- What is the motivation for Hive and Pig?
- Hive basics
- Pig basics

In this module you will learn

- What is the motivation for Hive and Pig?
- Hive basics
- Pig basics

Motivation of using Hive and Pig

- Most of MapReduce programming is done in java
- So one who is solving big data problems must
 - Have good Java programming back ground
 - Understand MapReduce and its concepts very well
- Solving a word count problems needs 100 lines of code
- Solving data mining problem (on wikipedia data set, which we have seen earlier) require two MapReduce jobs to be implemented and run sequentially one after the other

Motivation cont'd

- Map Reduce in Java is very low level
 - Needs to deal with all the intricacies
- Not only programmer, many non programmers like Business Analyst, Data Scientist wants to analyze the data
- Requirement:
 - Need of higher level abstraction on top of MapReduce
 - Not to deal with low level stuff involved in MapReduce

In this module you will learn

- What is the motivation for Hive and Pig?
- **Hive basics**
- **Pig basics**

Hive

- Data ware housing tool on top of Hadoop
- SQL like interface
- Provides SQL like language to analyze the data stored on HDFS
- Can be used by people who know SQL
- Not all traditional SQL capabilities are supported
 - Example: Sub queries are not supported in hive

Hive cont'd

- Under the hood hive queries are executed as MapReduce jobs
- No extra work is required

Hive Components

- MetaStore
 - Its a data base consisting of table definitions and other metadata
 - By default stored on the local machine on derby data base
 - It can be kept on some shared machine like relational data base if multiple users are using

Hive components cont'd

- Query Engine
 - Hive-QL which gives SQL like query
 - Internally Hive queries are run as map reduce job

Installing Hive

- Download Hive (0.10.0)
- Create following environment in .bashrc file
 - export HIVE_HOME=<path to your hive home directory>
 - export HIVE_CONF_DIR=\$HIVE_HOME/conf
 - export HIVE_LIB=\$HIVE_HOME/lib
 - export PATH=\$PATH:\$HIVE_HOME/bin
 - export CLASSPATH=\$CLASSPATH:\$HIVE_LIB

Installing Hive

- Create `hive-site.xml` (if not present) under `$HIVE_HOME/conf` folder

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
  <property>
    <name>hive.metastore.warehouse.dir</name>
    <value>/user/hive/warehouse</value>
  </property>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:54310/</value>
  </property>
  <property>
    <name>mapred.job.tracker</name>
    <value>localhost:54311</value>
  </property>
</configuration>
```

Installing Hive

- Create two directories `/tmp` and `/user/hive/warehouse` on HDFS and give them full permission
 - `hadoop fs -mkdir /tmp`
 - `hadoop fs -mkdir /user/hive/warehouse`
 - `hadoop fs -chmod g+w /tmp`
 - `hadoop fs -chmod g+w /user/hive/warehouse`

Launching Hive Shell

- Type “*hive*” command

Hive Data Models

- Hive forms or layers table definitions on top of data residing on HDFS
- Databases
 - Name space that separates tables from other units from naming confliction
- Table
 - Homogenous unit of data having same schema
- Partition
 - Determines how the data is stored
 - Virtual columns, not part of data but derived from load
- Buckets / Cluster
 - Data in each partition can be further divided into buckets
 - Efficient for sampling the data

Hive data types

- **Primitive types**
 - – TINYINT (1byte)
 - – SMALLINT (2 byte)
 - – INT (4 bytes)
 - – BIGINT (8 bytes)
 - – FLOAT
 - – BOOLEAN
 - – DOUBLE
 - – STRING
 - – BINARY (recently included)
 - – TIMESTAMP(recently included)
- **Type constructors:**
 - – ARRAY < *primitive-type* >
 - – MAP < *primitive-type*, *data-type* >
 - – STRUCT < *col-name* : *data-type*, ... >

Basic Queries

```
hive> SHOW TABLES;
```

```
hive>SHOW DATABASES;
```

```
hive> CREATE TABLE sample(firstName STRING, lastName  
STRING, id INT)  
ROW FORMAT  
DELIMITED FIELDS  
TERMINATED BY ''  
LINES TERMINATED BY '\n'  
STORED AS TEXTFILE
```

```
hive> DESCRIBE sample;
```

Loading Data Into Table From HDFS

- Assuming data is already present on HDFS

```
LOAD DATA INPATH “sample_data” INTO TABLE sample;
```

- The “*sample_data*” on HDFS will be moved to “/user/hive/warehouse/*sample*” folder on HDFS
 - /user/hive/warehouse is metastore

Loading Data Into Table From Local File System

- Give the path to the data present on your local file system

```
LOAD DATA LOCAL INPATH “sample_local_data” INTO TABLE  
sample;
```

Basic Select Queries

```
SELECT * FROM sample;
```

```
SELECT * FROM sample WHERE id >100  
ORDER BY id ASC  
LIMIT 20
```

- The first query does not run map reduce job
 - Just queries the meta data and displays the entire data
- Second query executes map reduce job

Joins in Hive

- Joins are complex and trickier in map reduce job

```
SELECT A.id, B.id,A.firstName FROM  
sample A  
JOIN otherTable B  
ON  
(A.id = B.id)
```

Storing Results

- Create a new table and store the results into this table

```
CREATE TABLE output(sum int);
```

```
INSERT OVERWRITE INTO TABLE output  
SELECT sum(id) FROM TABLE sample;
```

- Results are stored in the table *output*
 - Result is just an output file which can be used in subsequent queries or in map reduce jobs

Data Mining on Wikipedia datasets

- Design a schema

```
CREATE EXTERNAL TABLE wiki(projectname STRING,pagename  
STRING,pageview INT,pagesize INT)  
ROW FORMAT  
DELIMITED FIELDS  
TERMINATED BY ''  
LINES TERMINATED BY '\n'  
STORED AS TEXTFILE LOCATION '/user/dev/wiki';
```

- External table will not move the data to
/user/hive/warehouse folder
 - It just point to that location
-

Data Mining on Wikipedia datasets

- Create another table to store the output

```
CREATE TABLE wiki1(pagename STRING,sum INT);
```

Data Mining on Wikipedia datasets

- Fire the actual query to calculate the sum of the page count

```
INSERT OVERWRITE TABLE wiki1
SELECT pagename,sum(pageview) FROM wiki
WHERE projectname='en' group by pagename;
```

Data Mining on Wikipedia datasets

- View the results on console in descending order

```
Select * from wiki1 order by sum DESC limit 20
```

Hive Limitations

- All the standard SQL queries are not supported
 - Sub queries are not supported
- No Support for UPDATE and DELETE operation
- Cannot insert single rows

Hands On

- Refer hands-on document

Installing Pig

- Download Pig (0.10.0)
- Create following environment variables in .bashrc file
 - `export PIG_HOME=<path to pig directory>`
 - `export PATH = $PATH:$PIG_HOME/bin`
 - `export PIG_HADOOP_VERSION=1` (Since we are using hadoop version of 1.0.3)
 - `export PIG_CLASSPATH=$HADOOP_HOME/conf/`
 - Specify `fs.default.name & mapred.job.tracker` in `pig.properties` file under `$PIG_HOME/conf` directory
- No need to install the pig in all the nodes.
 - Requires only where the namenode and job tracker are running

Launching Grunt Shell

- Run *pig* command

Pig basics

- High level platform for MapReduce programs
 - Uses PigLatin Language
 - Simple Syntax
 - PigLatin queries are converted into map reduce jobs
- No shared meta data is required as in Hive

Pig Terminology

- Tuple:
 - Rows / Records
- Bags:
 - Unordered Collection of tuples
- Relation:
 - Pig Operator.
 - Generally the output of operator is stored in a relation
- PigLatin scripts generally starts by loading one or more data sets in bag, and then creates new bags by modifying

Loading Data

- ‘LOAD’ keyword
- TAB is the default delimiter
 - If input file is not tab delimited, even though loading will not be a problem but subsequent operations will be erroneous
- Other delimiters can be specified by “*using PigStorage(‘ ’)*”
 - In this case space is the delimiter

Loading Data cont'd

```
inputData = LOAD 'wiki/input1' using PigStorage(' ') as  
(projectName:chararray,pageName:chararray,pageCount:int,pageSize:int);
```

- Data “*wiki/input1*” is stored on HDFS
- Here the fields are separated by space which is defined by “*USING PigStorage('')*”
- Note when you are defining the column name, you need to specify the data types of the column, else NULL will be stored
- “*inputData*” will consist of the output of the loading

Filtering

- Use *Filter* Keyword

```
Output = FILTER records by projectName=='en'
```

- “*records*” is the data set and “*pageName*” is the column name defined in the “*records*”

Grouping

- Use *GROUP* key word

```
grouped_records = GROUP filtered_records by pageName;
```

- Here grouping is done on the column “*pageName*”

Grouping cont'd

```
grouped_records = GROUP filtered_records by pageName;
```

- Here grouping is done on the column “*pageName*”
- Each Tuple in “*grouped_records*” has two parts
 - *group* element which is column name on which you are grouping
 - *filtered_records* is itself is a bag containing all the tuples from *filtered_records* with matching *pageName*

Joins

- Use *JOIN* keyword

```
result = JOIN persons BY personId, orders BY personId;
```

- Default is *INNER JOIN*
- Above example joins “*persons*” and “*orders*” data on “*personId*” column

Joins cont'd

- Left outer join

```
result = JOIN persons BY personId LEFT OUTER, orders BY personId;
```

- For doing left outer join use “*LEFT OUTER*” key word

Joins cont'd

- Right outer join

```
result = JOIN persons BY personId RIGHT OUTER, orders BY  
personId;
```

Joins cont'd

- Full outer join

```
result = JOIN persons BY personId FULL OUTER, orders BY personId;
```

FOREACH...GENERATE

- Iterate over tuples in a bag

```
empID=FOREACH employee GENERATE id;
```

Other useful commands

- DESCRIBE
- EXPLAIN
- ILLUSTRATE
- REGISTER
- DEFINE



User Defined Functions (UDF)

DataMining: Top 10 English sites from wiki data

```
records = LOAD 'wiki/inputi'using PigStorage(' ') as  
(projectName:chararray,pageName:chararray,pageCount:int,pageSize:int);  
  
filtered_records = FILTER records by projectName == 'en';  
  
grouped_records = GROUP filtered_records by pageName;  
  
results = FOREACH grouped_records generate group,  
SUM(filtered_records.pageCount);  
  
sorted_result = ORDER results by $1 desc;  
  
STORE sorted_result INTO 'wikiOP' ;
```

Using Pig Scripts

- Can combine the pig statements in a script
 - Script name should end with “*.pig*” extension
- Can run the pig script by one of the three ways as described as follows:
 - *pig script_name.pig*
 - From the grunt shell
 - *run script_name.pig*
 - *exec script_name.pig*
 - From java (*Embedded Pig*)

Hands On

- Refer hands-on document

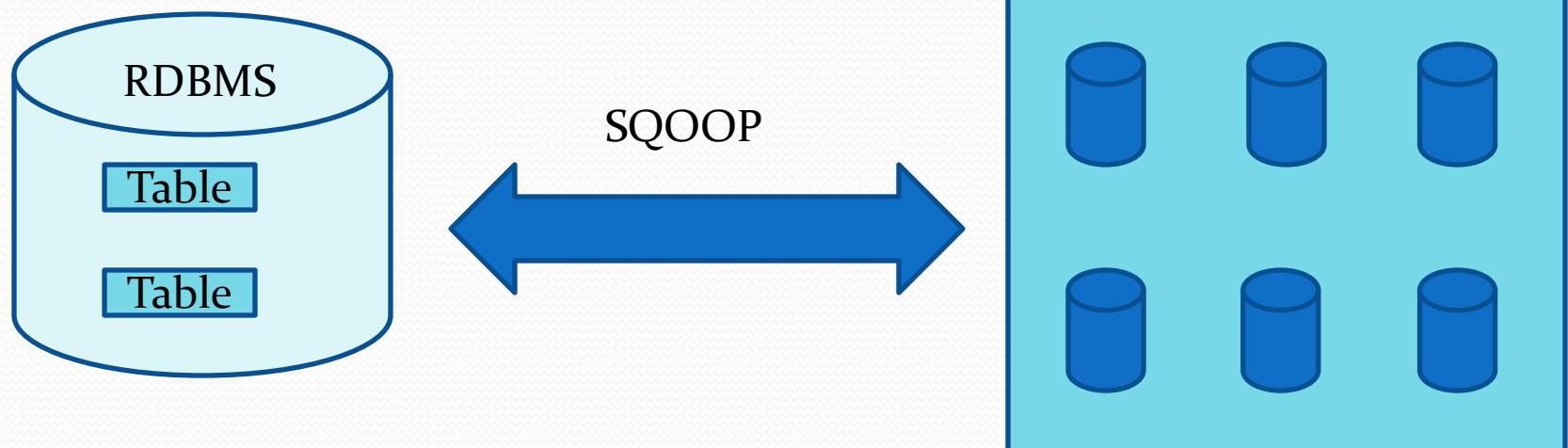
Module 20

Sqoop

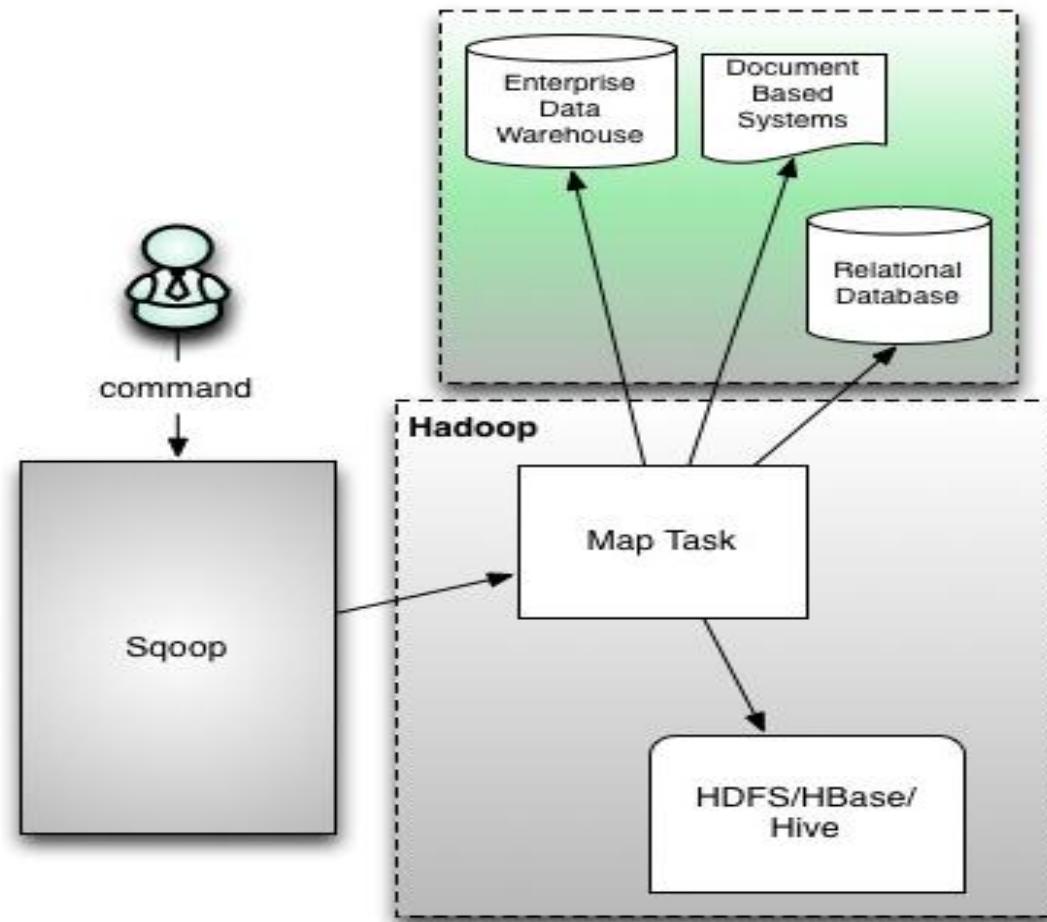
What is sqoop

- Applications use RDBMS to support OLTP
- Lots of data – legacy and current
- Sqoop designed to transfer data between RDMS and HDFS
 - Automate the process
 - Import data from RDMS to HDFS
 - Transfer the data in Hadoop MapReduce
 - Export data back to RDBMS
- Currently apache incubating project

SQOOP



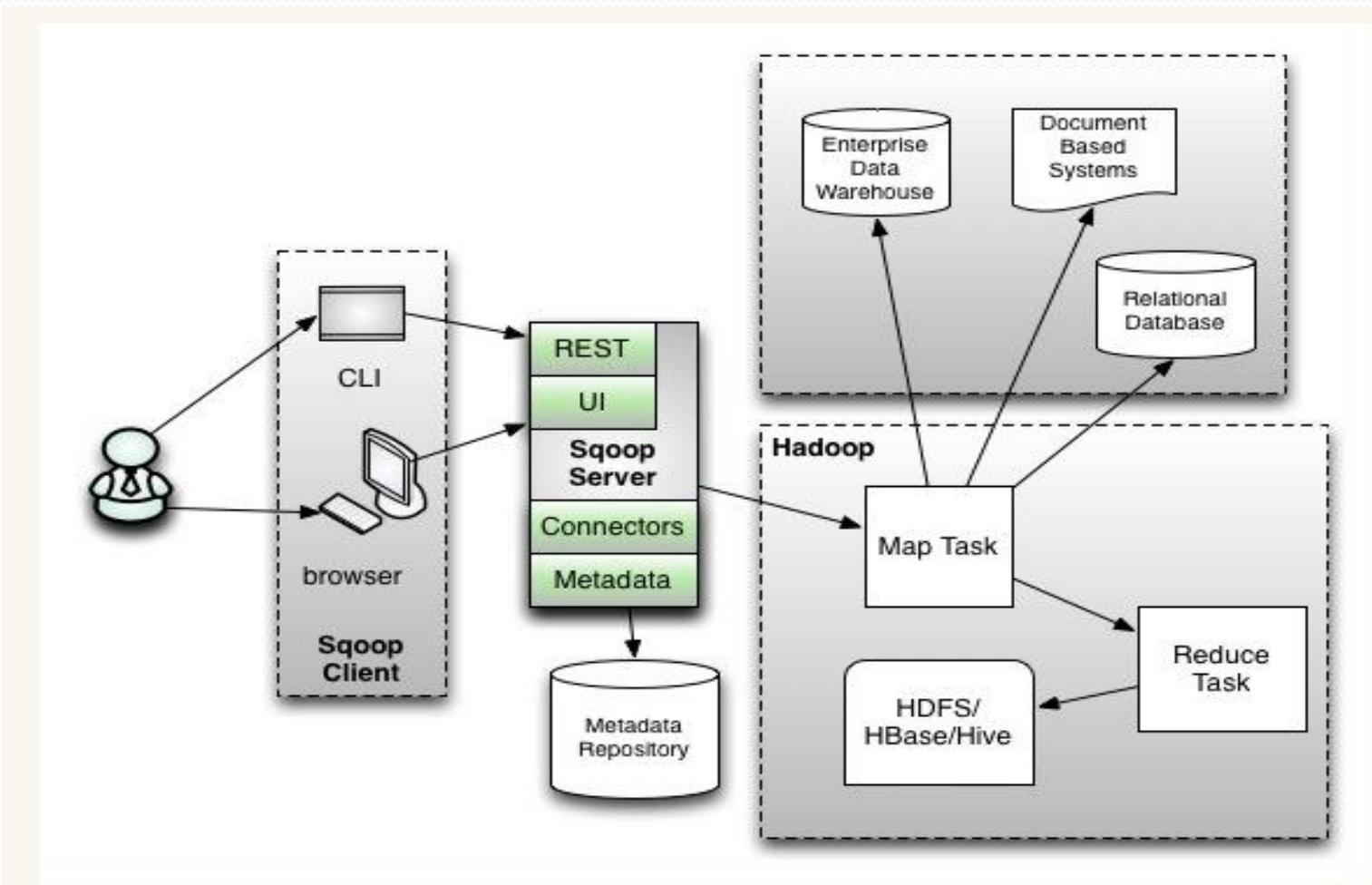
Sqoop Architecture



Challenges

- Cryptic command line arguments can lead to choose wrong connectors
- Tight coupling between data and serialization format
- Security concerns with open shared credentials
- Debugging the map jobs are limited to turning on the verbose flag
- Connectors are forced to follow the jdbc model regardless if it is applicable

Sqoop 2 Architecture



Sqoop 2 - Advantages

- Ease of use
 - Installed and configured on server side
 - All the connectors will be configured at one place
 - Client can connect either through command line or REST API's
- Ease of Extension
 - Connectors no more need to follow JDBC model

Features

- Transfer a table row-by-row into HDFS
- Parallelize the process by using four Mappers(configurable)
- Executes basically Map only job
- Output is a set of files depending on no of mappers
- Output file can be a delimited text files or binary Avro or SequenceFiles containing serialized record data

Features contd..

- Sqoop also gives a generated Java class which can encapsulate one row of the imported table
- This class can serialize and deserialize data to and from the SequenceFile format
- It can also parse the delimited-text form of a record
- Sqoop does reverse process for exporting data back into RDBMS after processing

Sqoop help

- help

```
$ sqoop help  
usage: sqoop COMMAND [ARGS]
```

Available commands:

codegen	Generate code to interact with database records
create-hive-table	Import a table definition into Hive
eval	Evaluate a SQL statement and display the results
.....	
.....	

See 'sqoop help COMMAND' for information on a specific command

- sqoop help import or sqoop import –help also give same

Command Aliases

- `sqoop (toolname)` can also be invoked as `sqoop-(toolname)`
 - `toolname` is one of the COMMAND like `import`, `codegen`, `eval` etc
 - `sqoop-import` is same as `sqoop import`

Choosing hadoop installation

- The sqoop command-line program is a wrapper which runs the bin/hadoop script shipped with Hadoop
- In case of multiple Hadoop installations, the \$HADOOP_HOME environment variable will be used to connect to correct cluster
- If \$HADOOP_HOME is not set, Sqoop will use the default installation location for Cloudera's Distribution for Hadoop,/usr/lib/hadoop

Generic and specific arguments

```
$ sqoop help import  
usage: sqoop import [GENERIC-ARGS] [TOOL-ARGS]
```

.....
.....

- generic arguments -conf, -D, and so on should be given after the tool name but **before** any tool-specific arguments (such as --connect)
- generic Hadoop arguments are preceded by a single dash character (-), whereas tool-specific arguments start with two dashes (--)

Options Files

- The command line options that do not change from invocation to invocation can be put in an options file for convenience
- An options file is a text file where each line identifies an option in the order that it appears otherwise on the command line
- Create an options file in a convenient location and pass it to the command line via --options-file argument

Options File ..

```
$ sqoop import --connect jdbc:mysql://localhost/db --username foo --table TEST
```

```
$ sqoop --options-file /users/homer/work/import.txt --table TEST
```

```
import  
--connect  
jdbc:mysql://localhost/db  
--username  
foo
```

sqoop import

- Imports an individual table from an RDBMS to HDFS
- Each row from a table is represented as a separate record in HDFS
- Records can be stored as text files (one record per line)
- Or in binary representation as Avro or SequenceFiles

Syntax:

```
$ sqoop import (generic-args) (import-args)
```

Note: Generic args must precede import-args
Import args have no order among each other

sqoop import

- Connecting to database server(Common arguments)
 - Use “*--connect <jdbc-uri>*”
 - Many databases supported
 - If a database(JDBC compliant) is not supported, put driver jar in \$SQOOP_HOME/lib and use “*--driver*”

```
$ sqoop import --connect jdbc:mysql://database.example.com/employees  
\\ --username aaron --password 12345
```

```
$ sqoop import --driver com.microsoft.jdbc.sqlserver.SQLServerDriver \\ -  
-connect <connect-string> ...
```

sqoop import

- Selecting the data
 - Sqoop typically imports data in a table-centric fashion
 - Use the "*--table*" argument to select the table
 - Subset can be specified by "*--columns*"
 - Rows also can be controlled by "*--where*"
- Free-form query imports
 - Sqoop can import using a sql statement using "*--query*"
 - Must specify a destination directory with "*--target-dir*"

sqoop import

- Example of query imports

```
$ sqoop import \ --query 'SELECT a.* , b.* FROM a JOIN b on (a.id == b.id)  
WHERE $CONDITIONS' \ --split-by a.id --target-dir /user/foo/joinresults
```

- “*split-by*” helps to run parallel mappers

```
$ sqoop import \ --query 'SELECT a.* , b.* FROM a JOIN b on (a.id == b.id)  
WHERE $CONDITIONS' \ -m 1 --target-dir /user/foo/joinresults
```

- “*-m or --num-mappers*” specify number of mappers

sqoop import

- Controlling parallelism
 - Use “*-m or --num-mappers*”
 - Sqoop uses a *splitting column* to split the workload.
Default is primary key column
 - Sqoop cannot currently split on multi-column indices
- Controlling import
 - “*--direct*”. Ex: with mysql , sqoop will use mysqldump
 - “*--target-dir*” specifies the HDFS directory

sqoop import

- Importing into Hive
 - Sqoop can also import the data into Hive by generating and executing a CREATE TABLE statement
 - Use “*--hive-import*”
 - Use “*--hive-overwrite*” when table already exists(it will replace the table!)
 - Sqoop supports importing data into partition

sqoop import

- Importing data into HBASE
 - Sqoop can also import records into a table in Hbase
 - By specifying “*--hbase-table*”
 - Each row of the input table will be transformed into an Hbase put operation to a row of the output table
 - Each output column will be placed in the same column family, which must be specified with “*--column-family*”

sqoop import

- Code generation control
 - A byproduct of importing a table to HDFS is a class which can manipulate the imported data
 - “*--package-name*” specifies package
 - “*--outdir*” specifies output directory for generated code
 - “*--bindir*” specifies output directory for compiled objects
 - “*--jar-file*” and “*--class-name*” options if code already present. To suppress new code

sqoop import-all-tables

- Imports a set of tables from an RDBMS to HDFS
- Conditions to be met:
 - Each table must have a single-column primary key
 - Must import all columns of each table
 - Must not intend to use non-default splitting column, nor impose any conditions via a WHERE clause
- Syntax:

```
$ sqoop import-all-tables (generic-args) (import-args)  
$ sqoop-import-all-tables (generic-args) (import-args)
```

sqoop import-all-tables

- Common database connection arguments; Import control arguments; Hive arguments & Code generation arguments are same as sqoop-import
- Doesn't support import to HBASE
- Example:

```
$ sqoop import-all-tables --connect jdbc:mysql://db.foo.com/corp
```

sqoop export

- Exports a set of files from HDFS back to an RDBMS
- The target table must already exist in the database
- By default, sqoop does INSERT operation
- In “update” mode, it will update

```
$ sqoop export (generic-args) (export-args)  
$ sqoop-export (generic-args) (export-args)
```

sqoop export

- “*--table*” is required.
- “*--export-dir*” is also required, specifies HDFS source directory
- This tool also supports “*--num-mappers*” or “*-m*” arguments for parallelism
- Provides “*--staging-table*” option which acts as an auxiliary table that is used to stage exported data
- Staging option is very useful where there is a possibility of failed exports

sqoop job

- Sqoop allows you to define *saved jobs*
- Useful for repeated imports and exports
- Very handy in incremental import
- Saved jobs remember the parameters used to specify a job, so they can be re-executed by invoking the job by its handle

```
$ sqoop job (generic-args) (job-args) [-- [subtool-name] (subtool-args)]  
$ sqoop-job (generic-args) (job-args) [-- [subtool-name] (subtool-args)]
```

sqoop job

- “--create <job-id>” creates the job. A second Sqoop command-line, separated by a -- should be specified; this defines the saved job
- “--delete <job-id>” deletes a saved job
- “--exec <job-id>” run a saved job
- “--show <job-id>” Show the parameters for a saved job
- “--list” list all jobs
- Either uses default metastore in \$HOME/.sqoop
- Or it can be shared, use “--meta-connect <jdbc-uri>”

sqoop job

- Examples

```
$ sqoop job --create myjob -- import --connect jdbc:mysql://example.com/db  
\\ --table mytable
```

```
$ sqoop job -list  
Available jobs:  
myjob
```

```
$ sqoop job --exec myjob  
10/08/19 13:08:45 INFO tool.CodeGenTool: Beginning code generation  
...
```

sqoop job

- Saved jobs and passwords
 - Sqoop does not store passwords in the metastore
 - It prompts each time at job invocation
 - But it can be overridden by setting `sqoop.metastore.client.record.password` to true
- Incremental imports
 - If an incremental import is run from the command line, “`--last-value`” need to be specified
 - If an incremental import is run from a saved job, this value will be retained in the saved job

sqoop codegen

- Codegen tool generates Java classes which encapsulate and interpret imported records
- If Java source is lost, it can be recreated using tool(import create source files)
- New versions of a class can be created
- Runs with common connection arguments, output line formatting arguments and input parsing arguments

```
$ sqoop codegen (generic-args) (codegen-args)  
$ sqoop-codegen (generic-args) (codegen-args)
```

sqoop codegen

- Also support hive arguments, generates file containing HQL statement to create a table and load data

```
$ sqoop codegen --connect jdbc:mysql://db.example.com/corp \ --table  
employees --outdir src/main/java --class-name com.example.sqoopimport
```

sqoop eval

- Eval tool allows users to quickly run simple SQL queries against a database
- Results are printed to the console; allowing user to do a dry run

```
$ sqoop eval (generic-args) (eval-args)  
$ sqoop-eval (generic-args) (eval-args)
```

- **-e,--query <statement>** Execute *statement* in SQL

```
$ sqoop eval --connect jdbc:mysql://db.example.com/corp \ --query "SELECT *\nFROM employees LIMIT 10"
```

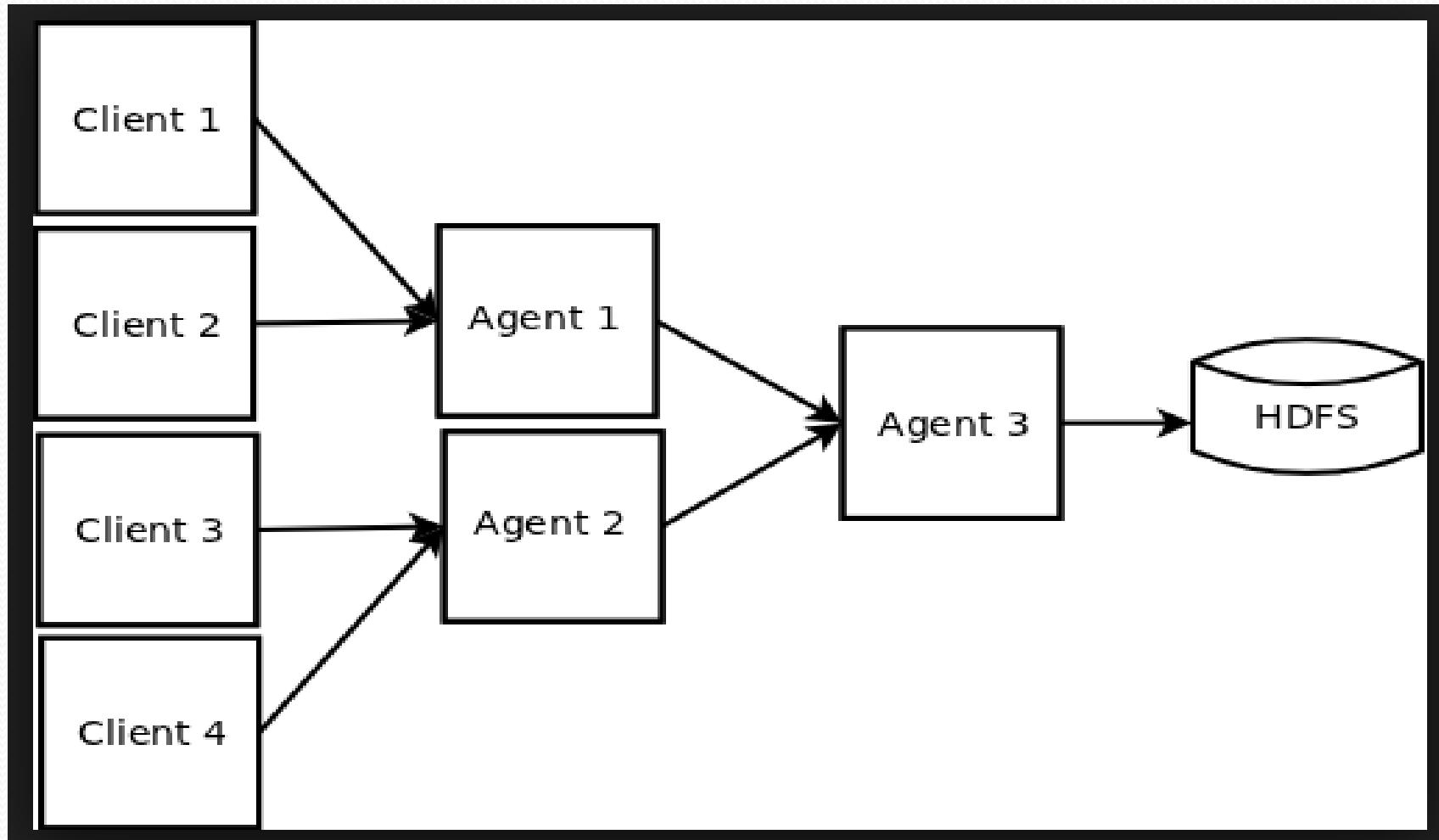
Module 21

Flume

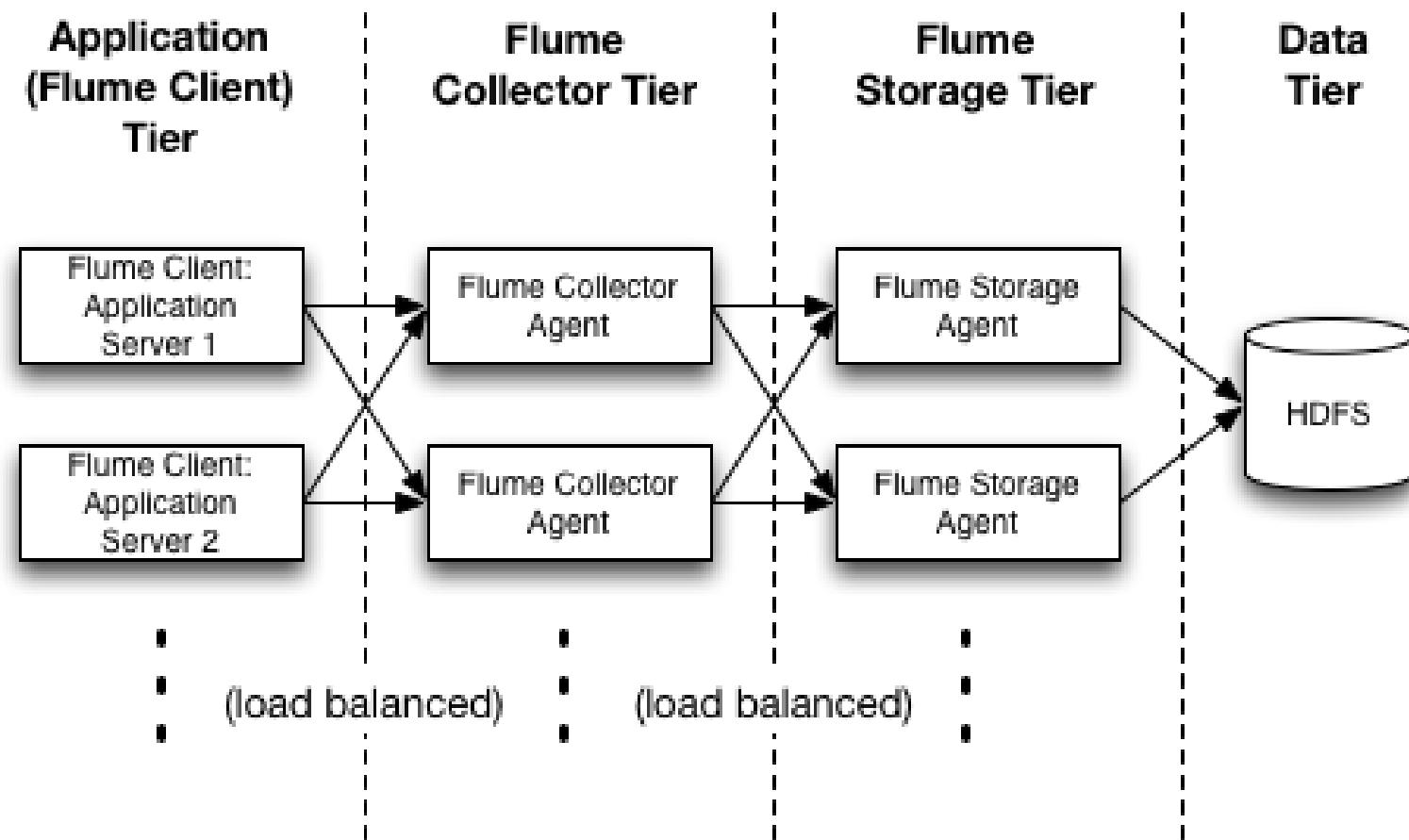
What is Flume?

- Apache Flume is distributed, reliable and available system
- Used for collecting, aggregating large amounts of data from various sources to a centralized data store

How Flume is used?



How Flume can be used?



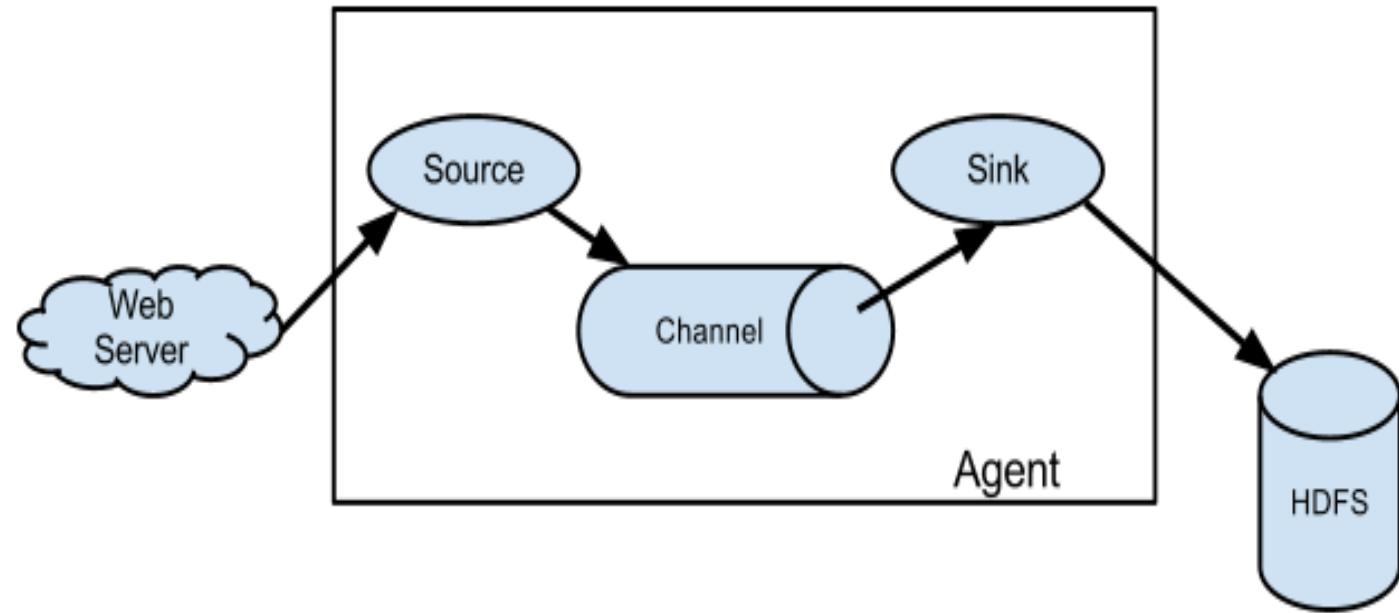
Flume Terminology

- Event: Single unit of data flow
 - Example: Single record of a web log
- Flume Agent: It's a process that hosts components through which event flows from external world to the next destination
 - A single flume agent consist of three components
 - Source
 - Channel
 - Sink

Flume Terminology cont'd

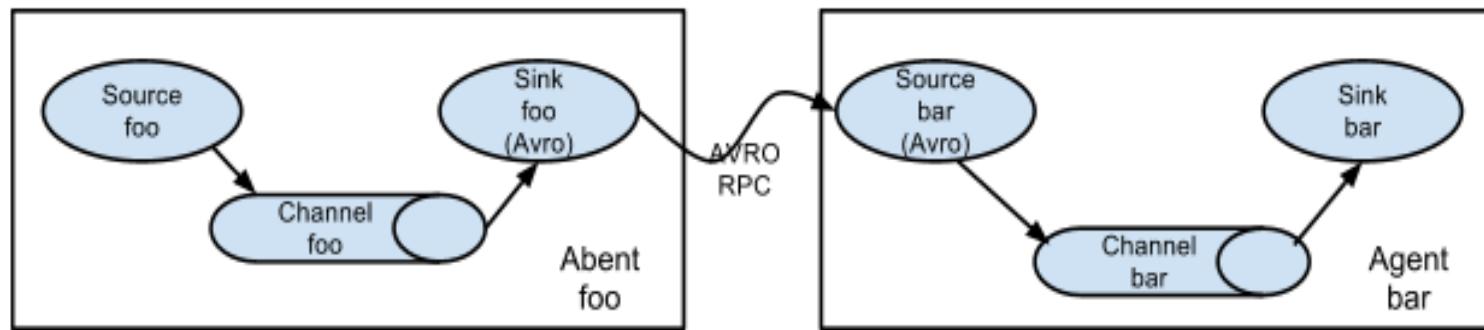
- Source: Consumes the events delivered by the external source
- Channel: Passive store where the events are pushed. Keeps the events till it is consumed by the sink
- Sink: Pull (Remove) the events from the channel and delivering to the next destination

Simple Data flow Model



Feature of Flume

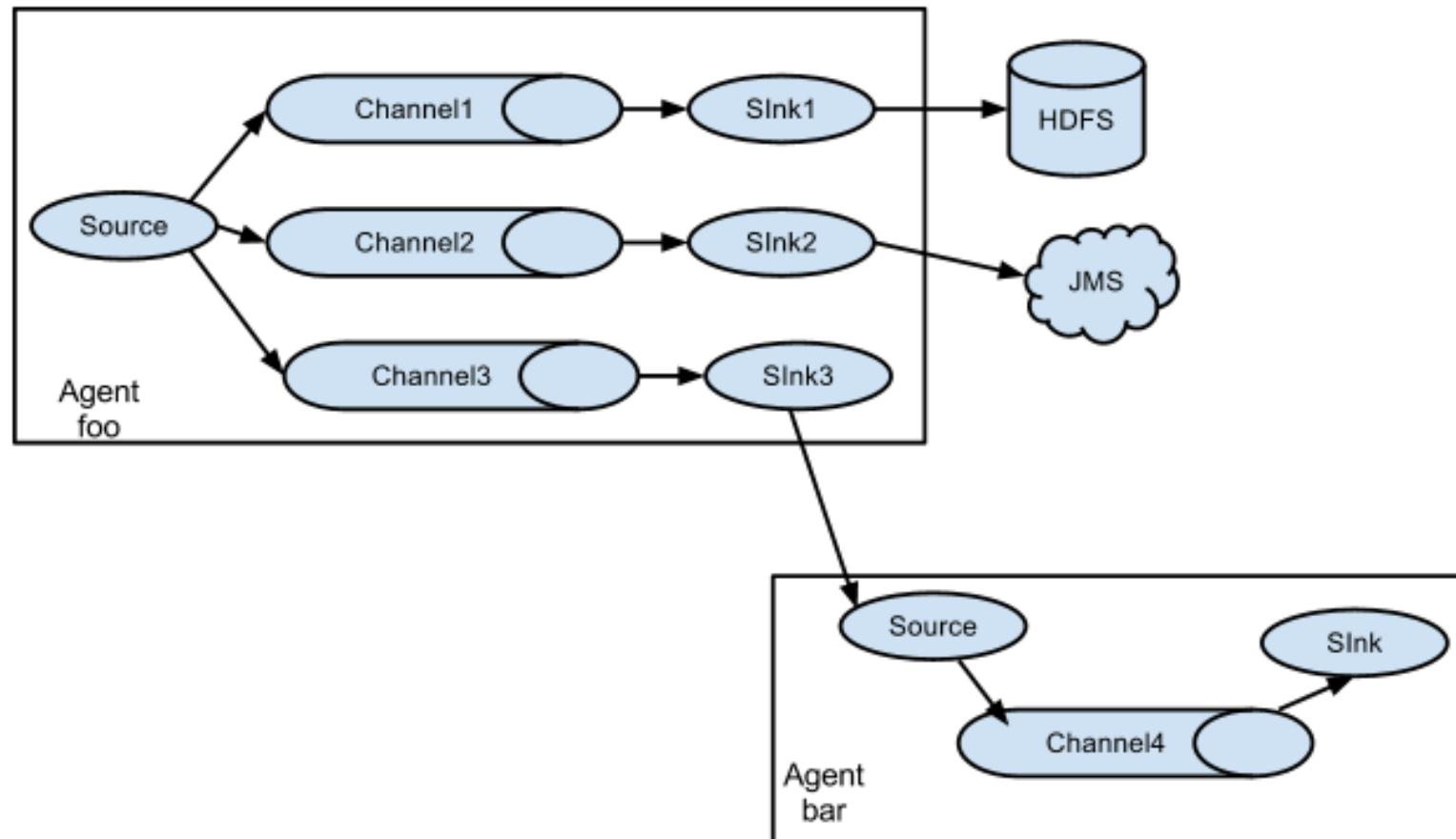
- Multi hop flows



- Events can travel through multiple agents before reaching the final destination

Features of Flume cont'd

- Multiplexing the flow



Features of Flume

- Contextual routing
 - Depending on the context of the event you are sending event to different channels
- Failover hops
 - If an agent fails, the event is delivered to another agent

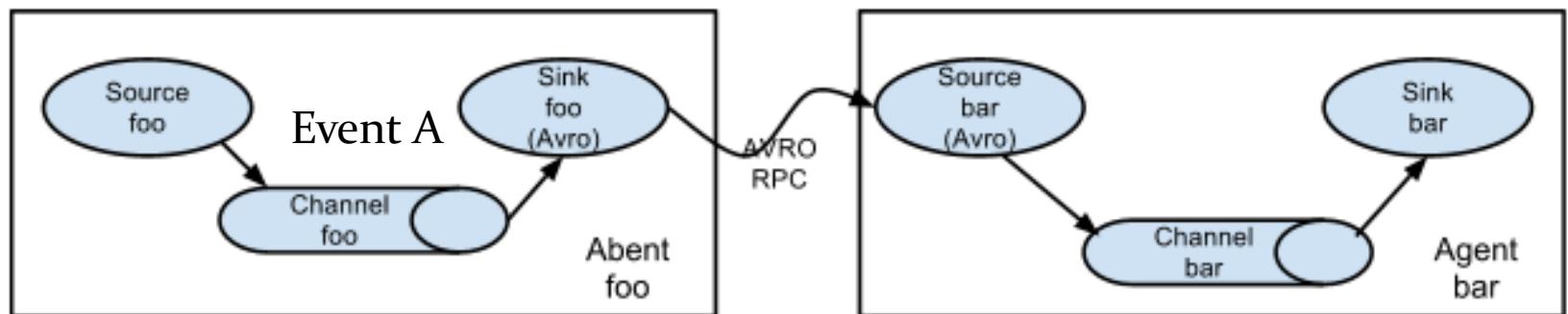
How Reliability is achieved?

- What is Reliability ?
 - Making sure that event has been delivered properly to the destination
- It uses a transactional approach to guarantee the reliable delivery of events
- Events are removed from the channel only after they are stored in the channel of next agent or terminal repository

How Reliability is achieved?

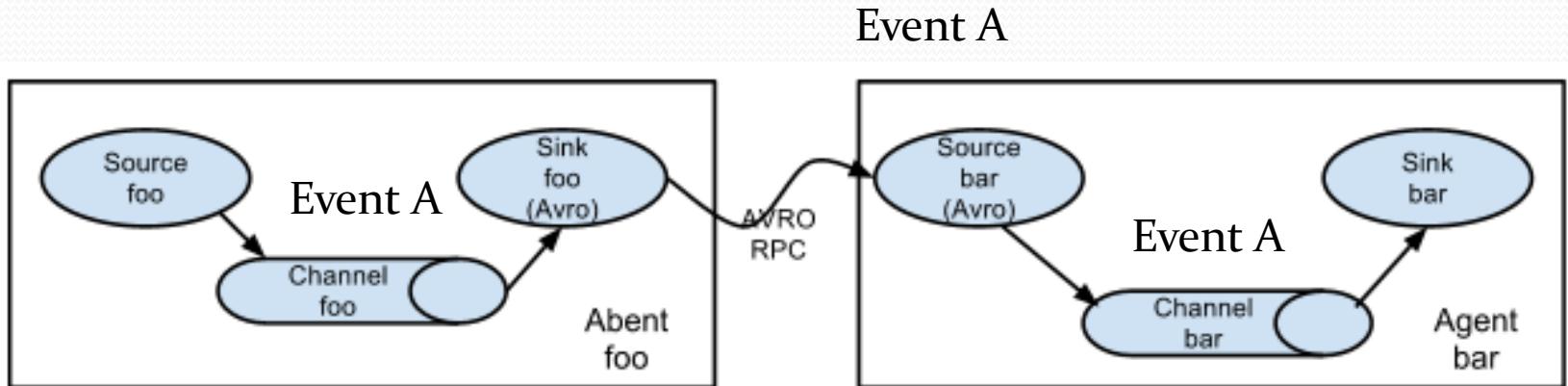
Event is taken from source

Event A

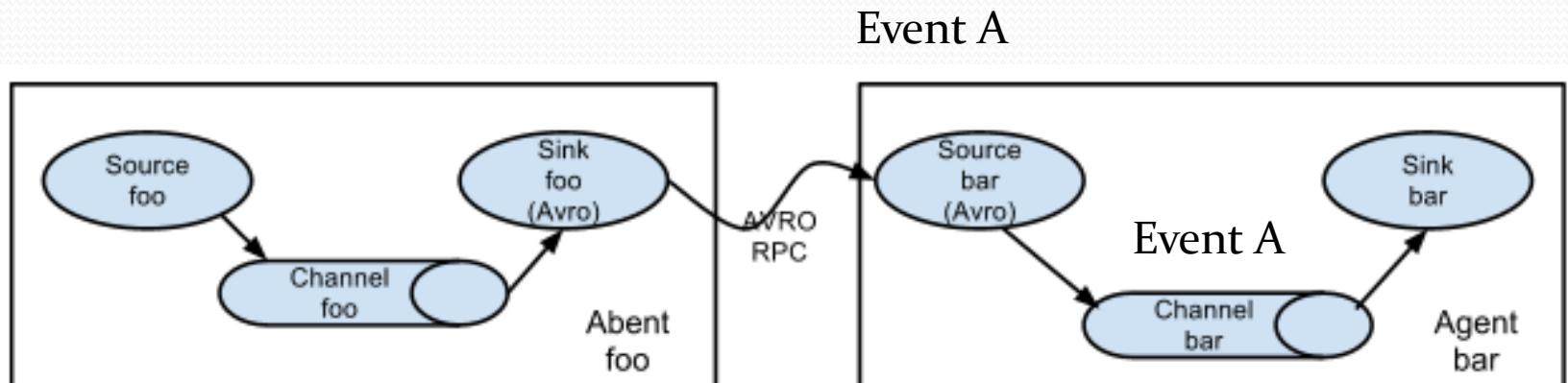


How Reliability is achieved?

Event is put into channel



How Reliability is achieved?



Event A is removed from
channel

How Recoverability is achieved?

- What is recoverability?
 - Recovering the events in case of flume agent failure
 - Making sure that events are not lost in case the machine is failed where flume agent is running
- Flume also supports a durable file channel which is backed by the file channel

Setting up Flume agent-

- Source
 - Type of the source
 - Address and port
 - Channel
- Sink
 - Type of the sink
 - Address and port
 - Channel
- Channel
 - Type of the channel
 - Other properties

Description

- We are going to configure an agent where
 - source is listening to a port 44444
 - Channel that buffers the event data in memory
 - And Sink that logs event data to console

Sample Flume Agent

```
sample.sources = netcatSource  
sample.channels = memoryChannel  
sample.sinks = loggerSink
```

#Source configuration

```
sample.sources.netcatSource.type = netcat  
sample.sources.netcatSource.bind = localhost  
sample.sources.netcatSource.port = 44444  
sample.sources.netcatSource.channels = memoryChannel
```

#Channels

```
sample.channels.memoryChannel.type = memory
```

#Sinks

```
sample.sinks.loggerSink.type = logger  
sample.sinks.loggerSink.channel = memoryChannel
```

Sample Flume Agent cont'd

```
sample.sources = netcatSource  
sample.channels = memoryChannel  
sample.sinks = loggerSink
```

- “sample” is the flume agent name
- “sample” agent has the source by name “netcatSource”
- “sample” agent has the sink by name “loggerSink”

```
sample.channels.memoryChannel.type = memory
```

```
#Sinks  
sample.sinks.loggerSink.type = logger  
sample.sinks.loggerSink.channel = memoryChannel
```

Sample Flume Agent

```
sample.sources = netcatSource  
sample.channels = memoryChannel  
sample.sinks = loggerSink
```

#Source configuration

```
sample.sources.netcatSource.type = netcat  
sample.sources.netcatSource.bind = localhost  
sample.sources.netcatSource.port = 44444  
sample.sources.netcatSource.channels = memoryChannel
```

- “sample” agent has the source by name “netcatSource” which is of type “netcat”
- This source is bind to your localhost address and at port 44444
- This source will put the events in the channel whose name is “memoryChannel”

Sample Flume Agent

```
sample.sources = netcatSource  
sample.channels = memoryChannel  
sample.sinks = loggerSink
```

#Source configuration

- “sample” agent has channel by name “memoryChannel” and this channel is of type “memory”

sample.channels.memoryChannel.type = memory

#Sinks

```
sample.sinks.loggerSink.type = logger  
sample.sinks.loggerSink.channel = memoryChannel
```

Sample Flume Agent

```
sample.sources = netcatSource  
sample.channels = memoryChannel  
sample.sinks = loggerSink
```

#Source configuration

“sample” agent is using the sink by name “loggerSink” and this sink is of type “logger”

This sink is using the same memory channel which is of type “memory”

#Sinks

```
sample.sinks.loggerSink.type = logger  
sample.sinks.loggerSink.channel = memoryChannel
```

Hands-On

- Refer the hands-on document

Ingesting data from external Source

- Using RPC mechanism
 - You can use Avro client that can send entire file to Flume avro source

RPC Mechanism Hands-On

- Refer the Hands-On Document

Using Exec source to send the events

- Can use unix commands like tail or cat to generate the events
- Exec source expects the data to be continuously produced on std out
- If the process exits for some reason, the source will also exit and it produces no further data
- Not reliable
 - Does not take guarantee that events are delivered
 - Use Spooling Directory instead

Exec Source Hands-on

- Refer the hands-on document

Disadvantage of using Exec Source

- Client will not know if the source has failed to put the events into the channel
- Data will be lost in such cases
- Example:
 - An application writing a log to a file, and using exec source you are sending the events, but if flume channel is filled and if Flume unables to send the event, there is no way to indicate the application regarding the failure

Using Spooling Directory Source

- Place the files in a spooling directory on a disk
- Source will watch the specified directory for new files, parses the events out of new files as they appear
- After the file has been fully read into the channel, the file is renamed to indicate completion

Precaution while using spooling directory

- File once placed into the spooling directory must not be written / changed
- Same file name should not be reused at later time
- Events might be sometimes duplicated if some downstream failure occurs
- Use newer version of flume (1.3) to use it

Using Sequence Generator Source

- Simple source that continuously generates events with counter starting from 0 and increment by 1
- Generally used for testing

Channels-Memory Channel

- Events are stored in in-memory queue with configurable size
 - fast
- Ideal where you need higher throughput
- Data might be lost in case of agent failure

Channels- JDBC Channel

- Events are stored in persisted storage that's backed by database
- Currently supports embedded derby

Channels-File Channel

- Stores the events in a file and keeps the check point

Fanning out the event flow

- Flume can fan out the flow from one source to multiple channels
- Two modes are available
 - Replicating
 - Event is sent to all the configured channels
 - Multiplexing
 - Event is sent to subset of qualifying channels

Module 21

Introduction to Complex Event Processing

Conquer the World of Big Data

Big Data Is No More BIG

