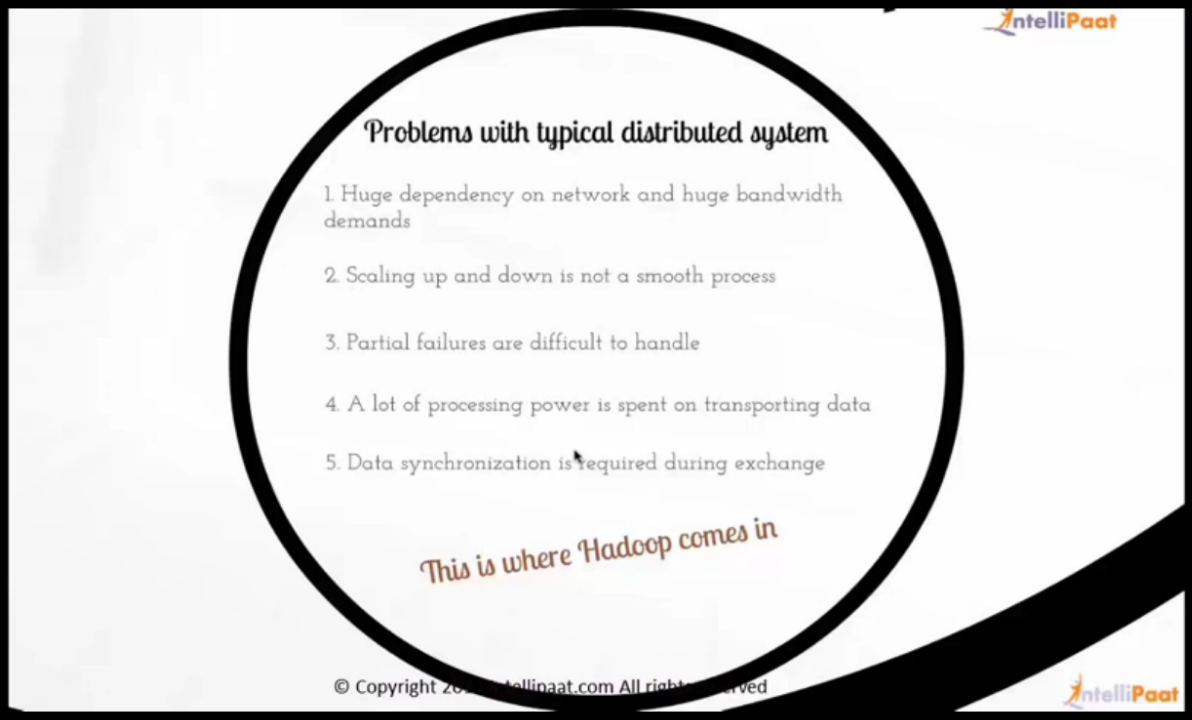
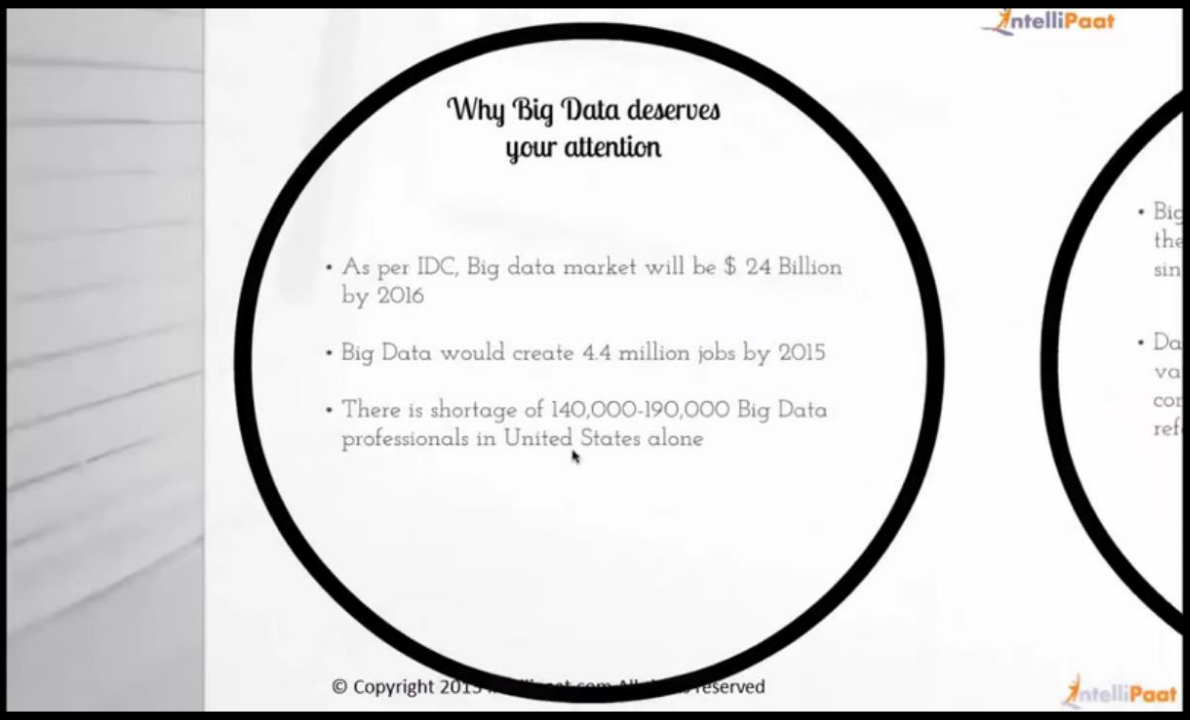
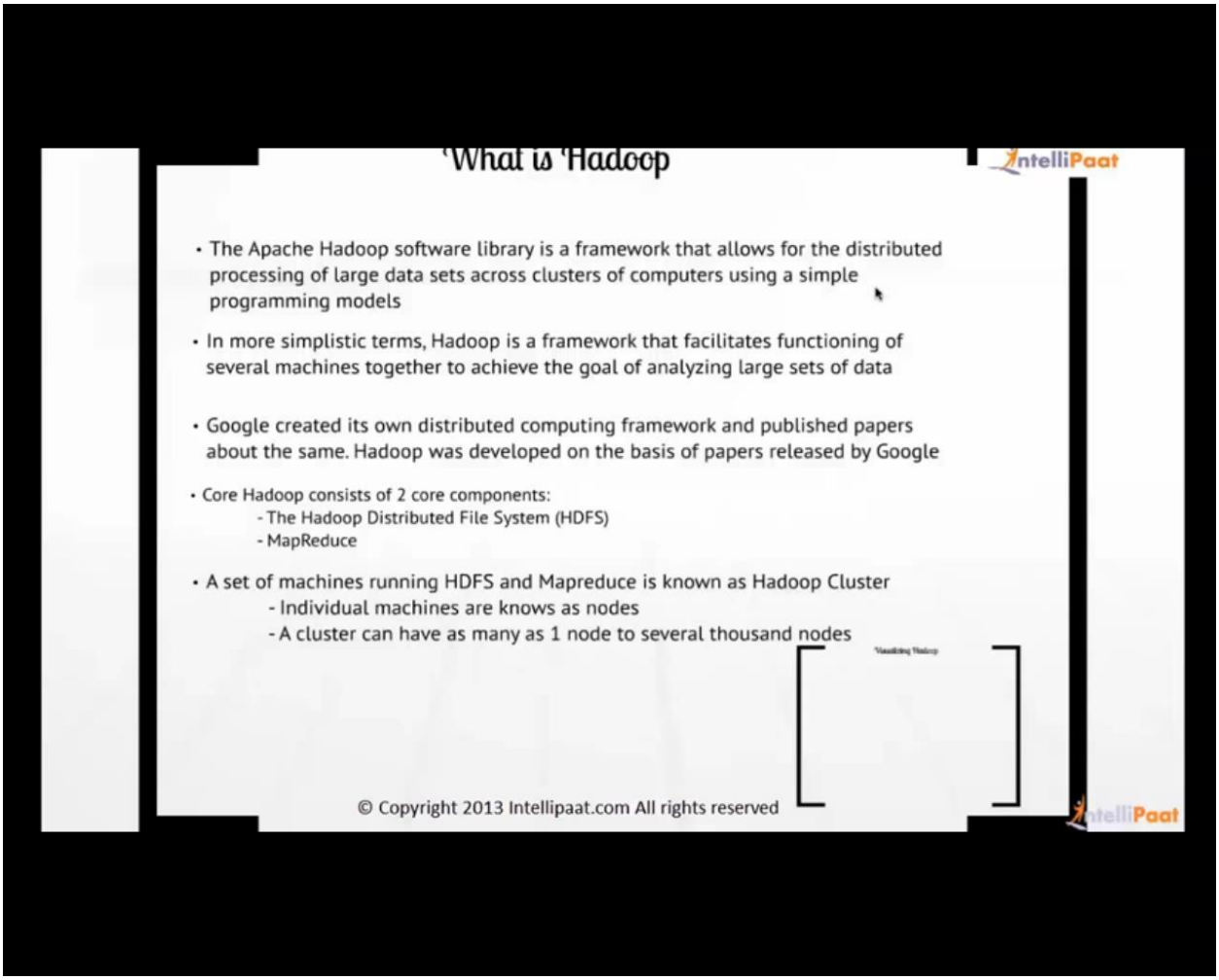
Hadoop Tutorial Videos | Hadoop Online Videos | Hadoop Videos | Intellipaat



11:24 / 1:01:04


IntelliPaat
Why Big Data deserves your attention
• As per IDC, Big data market will be $ 24 Billion by 2016
• Big Data would create 4.4 million jobs by 2015
• There is shortage of 140,000-190,000 Big Data professionals in United States alone
© Copyright 2015
reserved
IntelliPaat



MapReduce Flow Chart

WordCountJob :

---

file.txt (200MB)

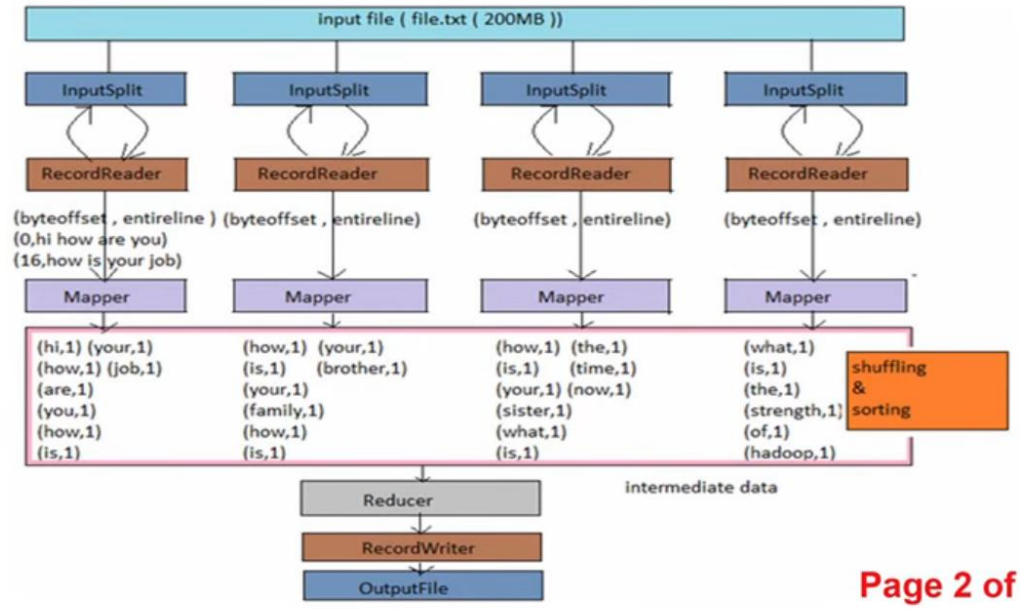
hi how are you how is your job	64MB
how is your family how is your brother	64MB
how is your sister what is the time now	64MB
what is the strength of hadoop	8MB

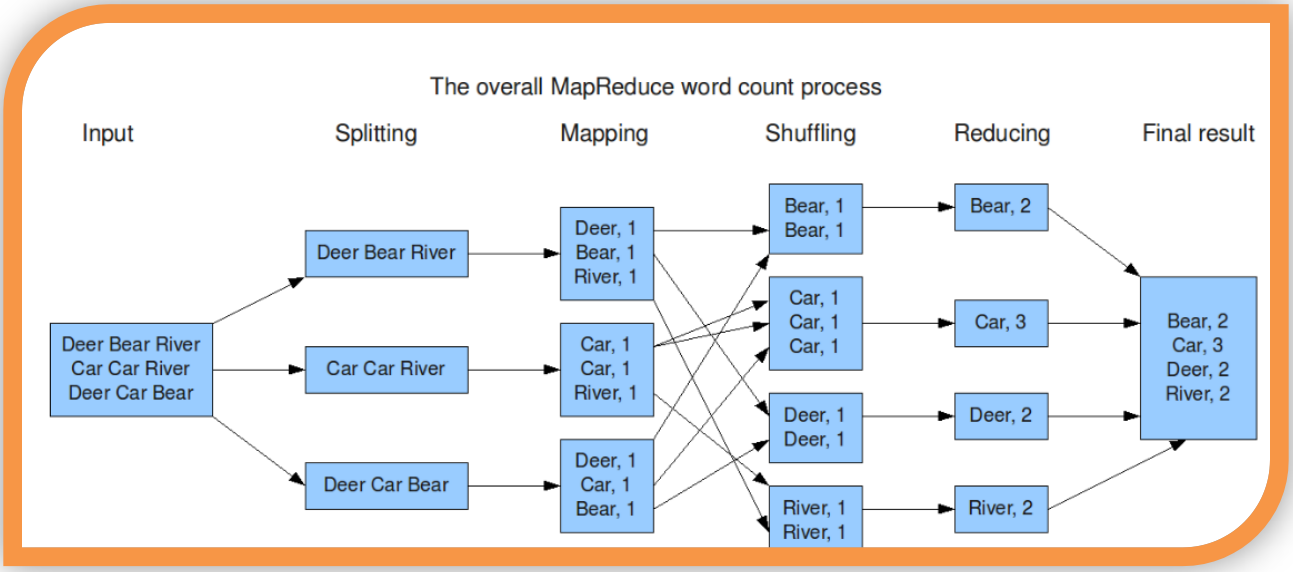
Input File Formats :

---

1. TextInputFormat
2. KeyValueTextInputFormat
3. SequenceFileInputFormat

MapReduce Flow Chart :





MapReduce Job Submission Flow

edureka!

Input data is distributed to nodes

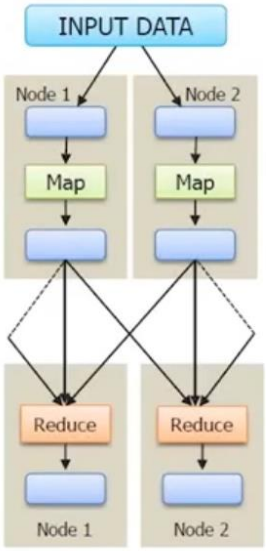
Each map task works on a "split" of data

Mapper outputs intermediate data

Data exchange between nodes in a "shuffle" process

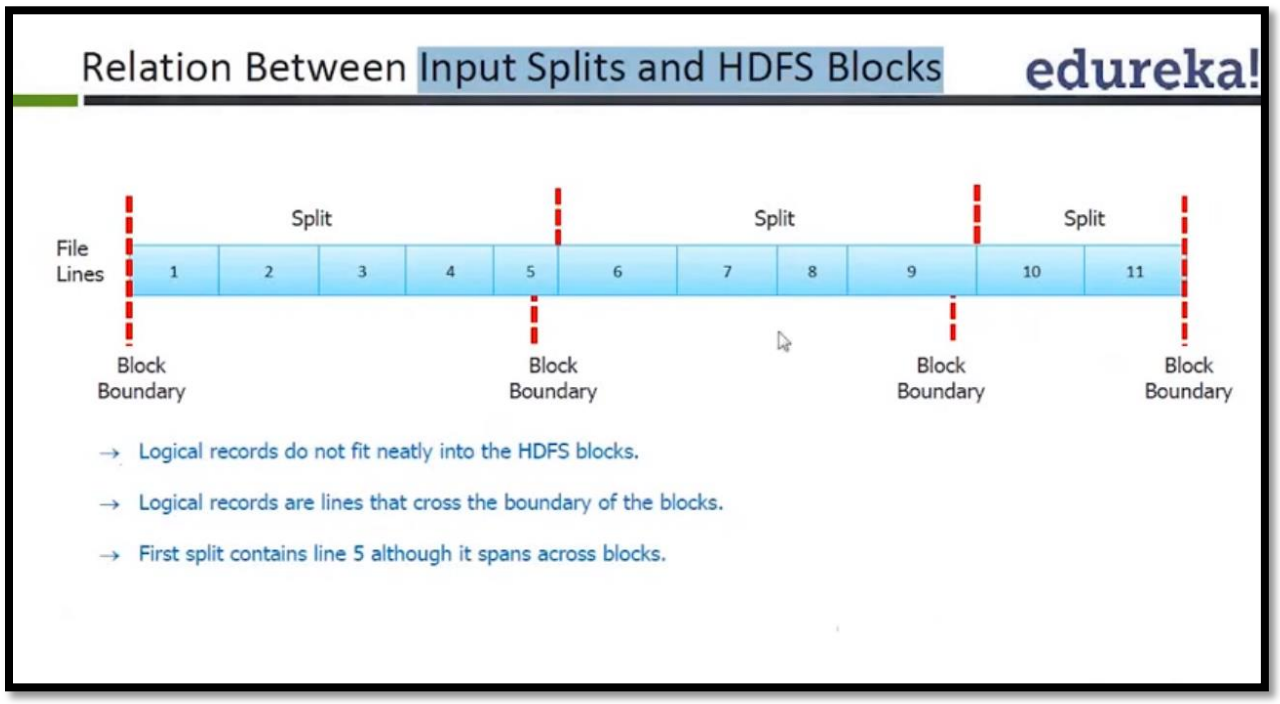
Intermediate data of the same key goes to the same reducer

Reducer output is stored



http://www.edureka.co/comprehensive-mapreduce

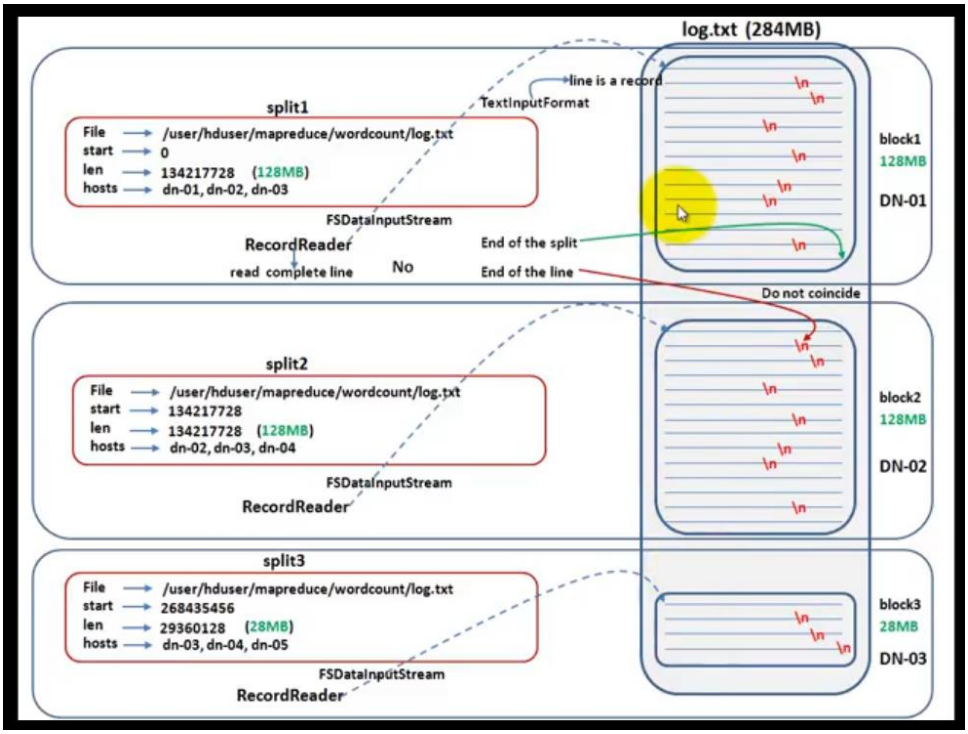
Input Split and HDFS Blocks



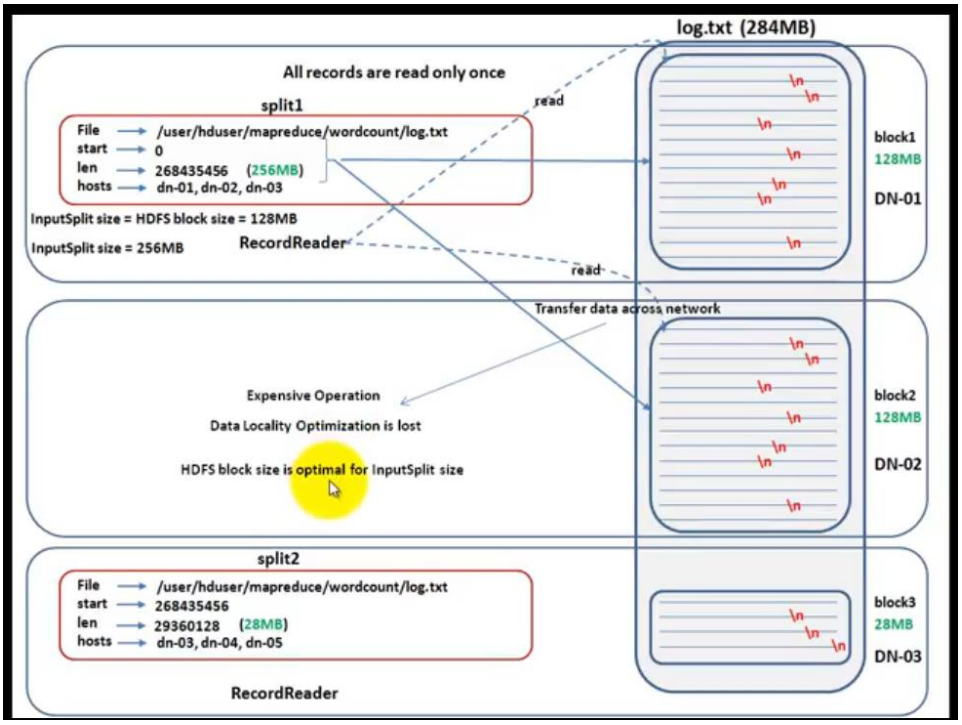
In case of record 5 and 9 (line 5 and 9) records are split in two blocks so record reader 1 will read the entire 5th line using FSDataInputStream via RPC and process the whole record. FSDataInputStream has all the information regarding file like file size, block location etc.

Generally input split is same as block size to achieve data locality, if we set input split size twice as block size than one mapper will process the data that is placed under two different blocks in two different nodes

Input split (FileSplit) creates splits, store lots of information like block location, start offset of the split, length of the split, size of the split etc.. that information is used by record reader to read the lines

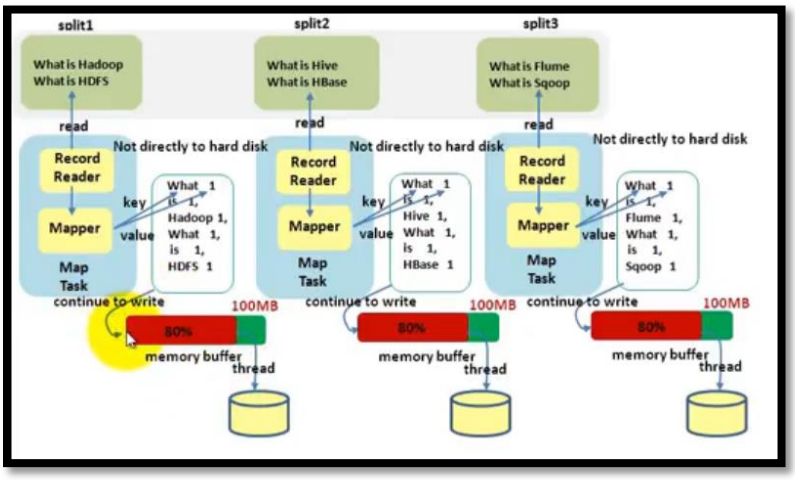

log.txt (284MB)
line is a record
TextInputFormat
split1
File → /user/hduser/mapreduce/wordcount/log.txt
start → 0
len → 134217728 (128MB)
hosts → dn-01, dn-02, dn-03
FSDataInputStream
RecordReader
read complete line
No
End of the split
End of the line
Do not coincide
block1
128MB
DN-01
split2
File → /user/hduser/mapreduce/wordcount/log.txt
start → 134217728
len → 134217728 (128MB)
hosts → dn-02, dn-03, dn-04
FSDataInputStream
RecordReader
block2
128MB
DN-02
split3
File → /user/hduser/mapreduce/wordcount/log.txt
start → 268435456
len → 29360128 (28MB)
hosts → dn-03, dn-04, dn-05
FSDataInputStream
RecordReader
block3
28MB
DN-03

If split size change to 256 MB and block size is 128 MB

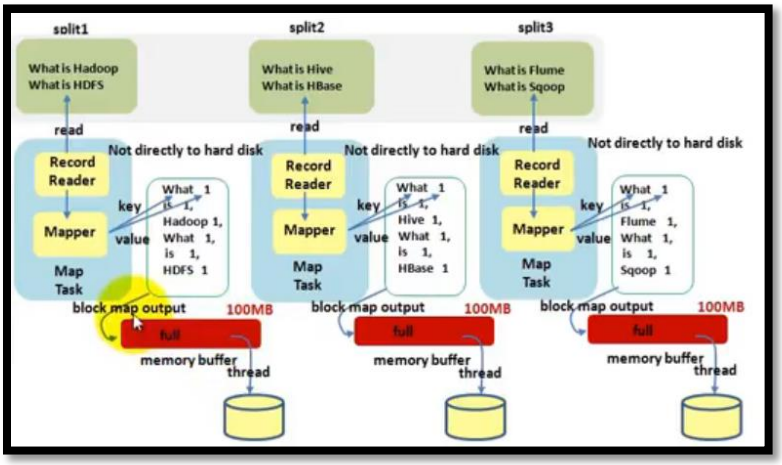


Mapper Reducer Shuffling Sorting

Processed data of map first write into map buffer by default its size is 100mb(mapreduce.task.io.sort.mb in Cloudera it is set to 250 MB) and it spill to disk after it reach to certain threshold by default the value is .8 or 80% (`mapreduce.map.sort.spill.percent`), the background thread spill the data into disk and mapper continue to write the data

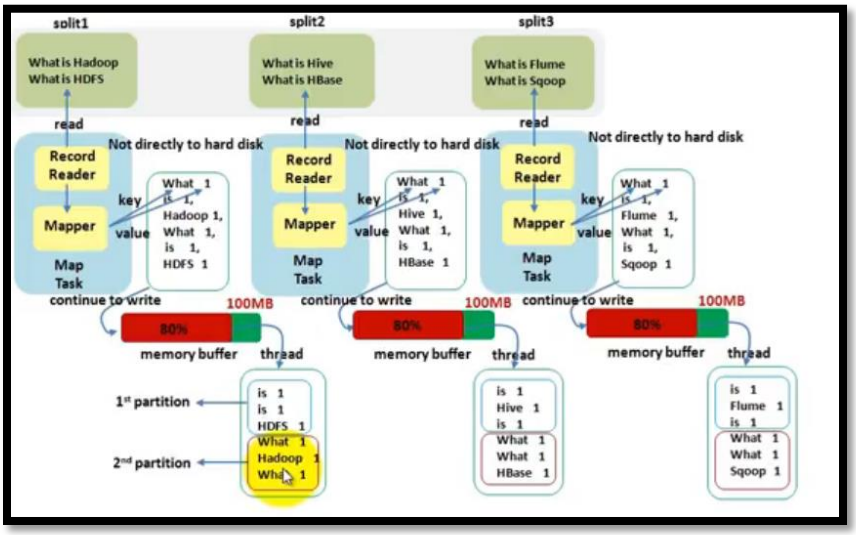


Map block if buffer is full till the time it is spilled to disk

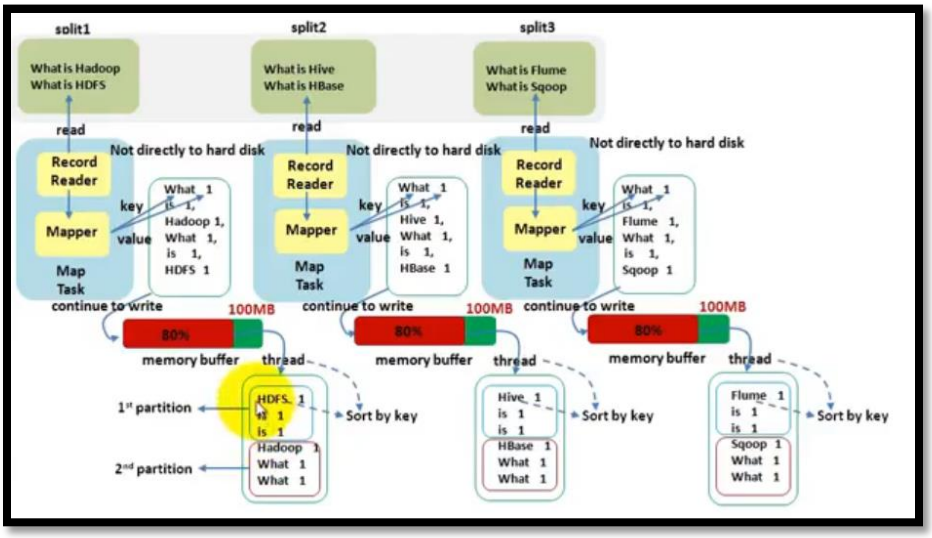


If we used partition in our mapreduce program, first background thread creates partition before writing onto the disk

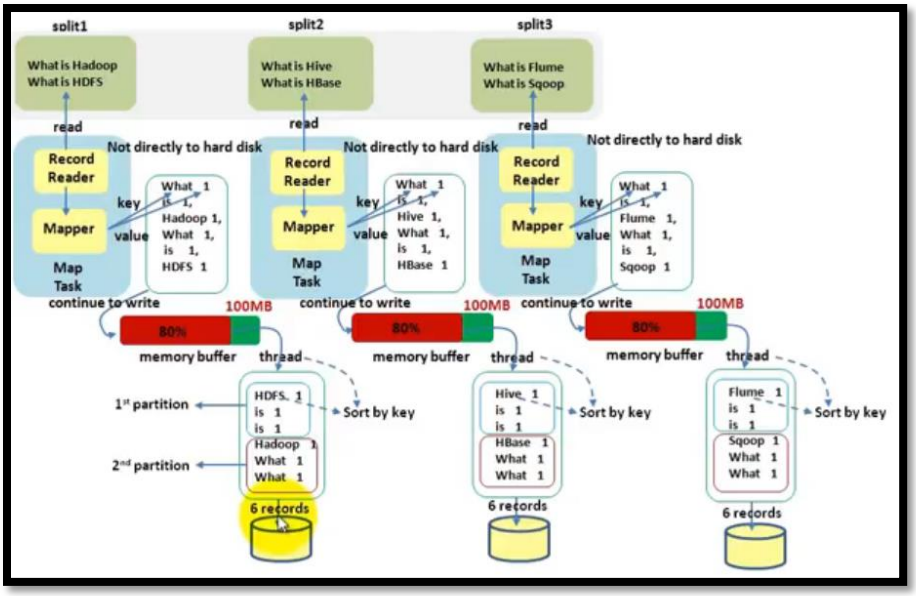
Here 2 partition created as we set 2 reduce task in program



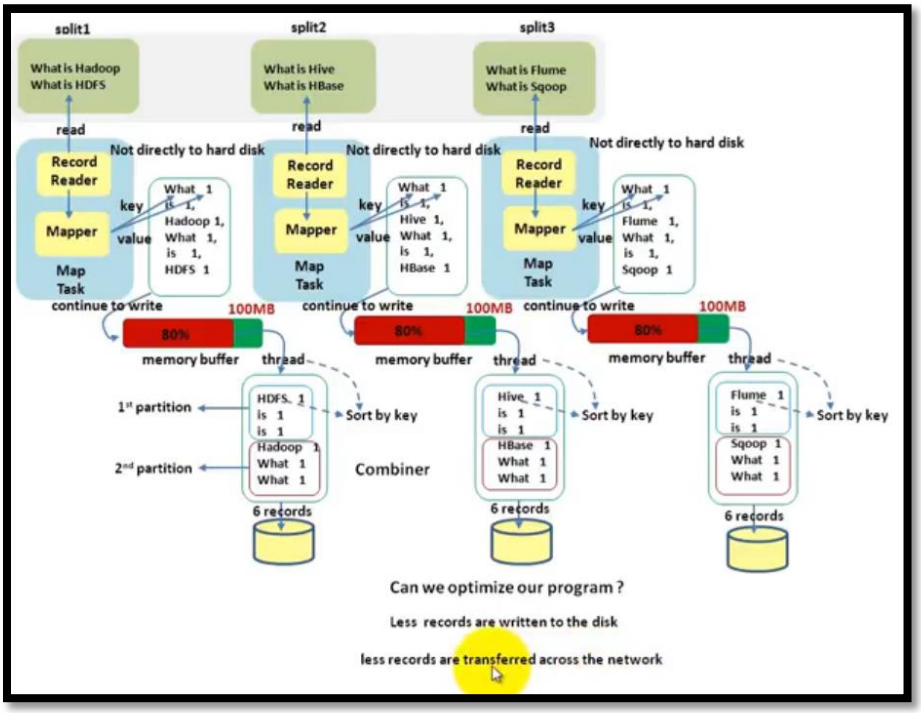
Background thread sort all the records by key



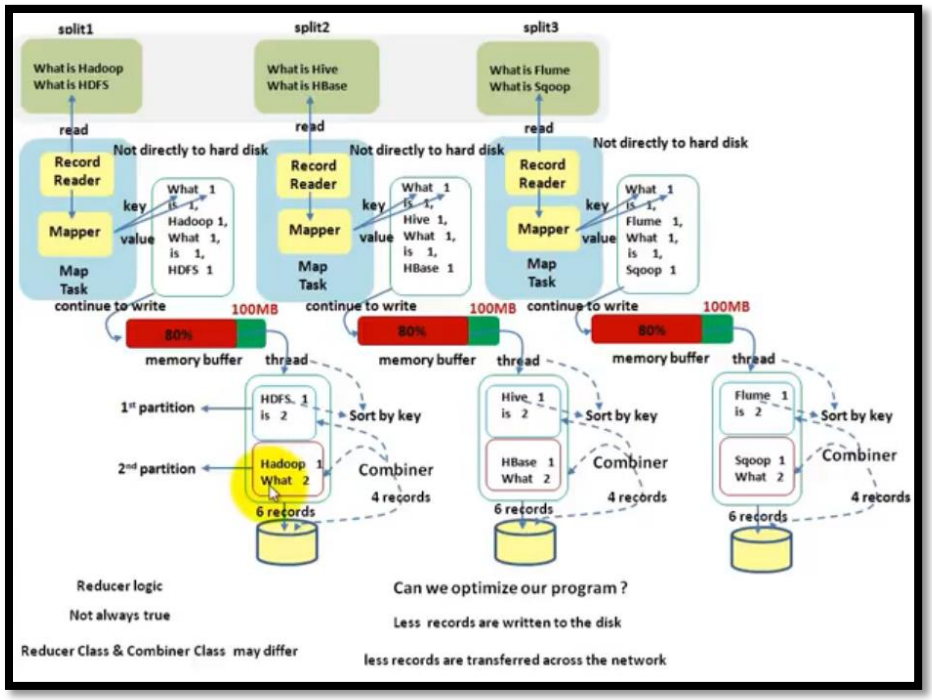
All these operation (partition +sorting) performed inside the memory buffer and later it will write into local filesystem disk



We can use combiner so that fewer records are written to the disk and fewer records are transferred across the network, so combiner process also performed inside memory buffer before writing data on to the disk

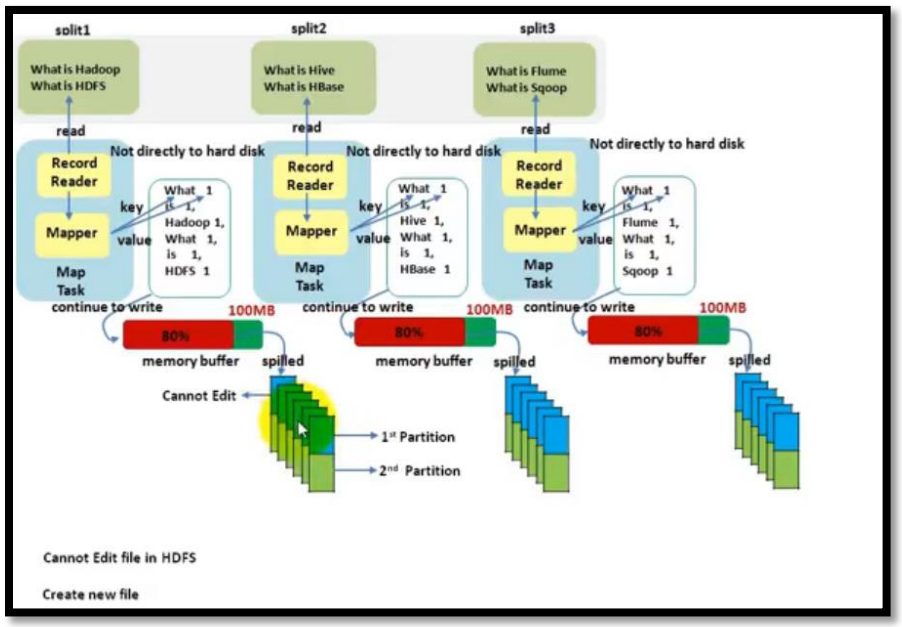


Instead of 6 now only 4 records are written to disk

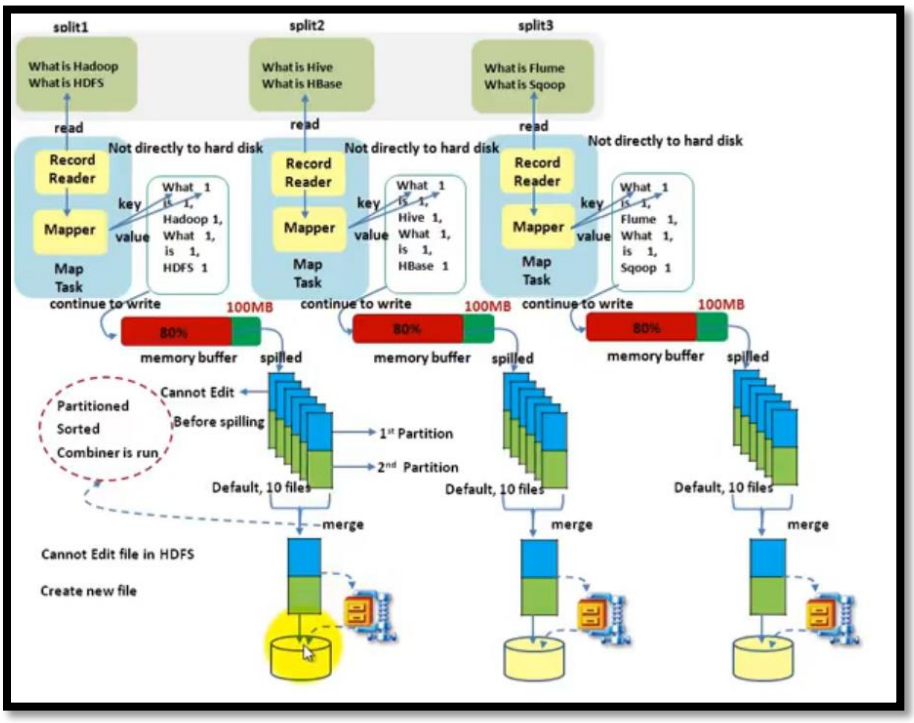


As we know background thread write the data onto the disk once buffer reach to threshold value so once it written data in one spilled files we cannot append the content on same file when threshold limit again reaches

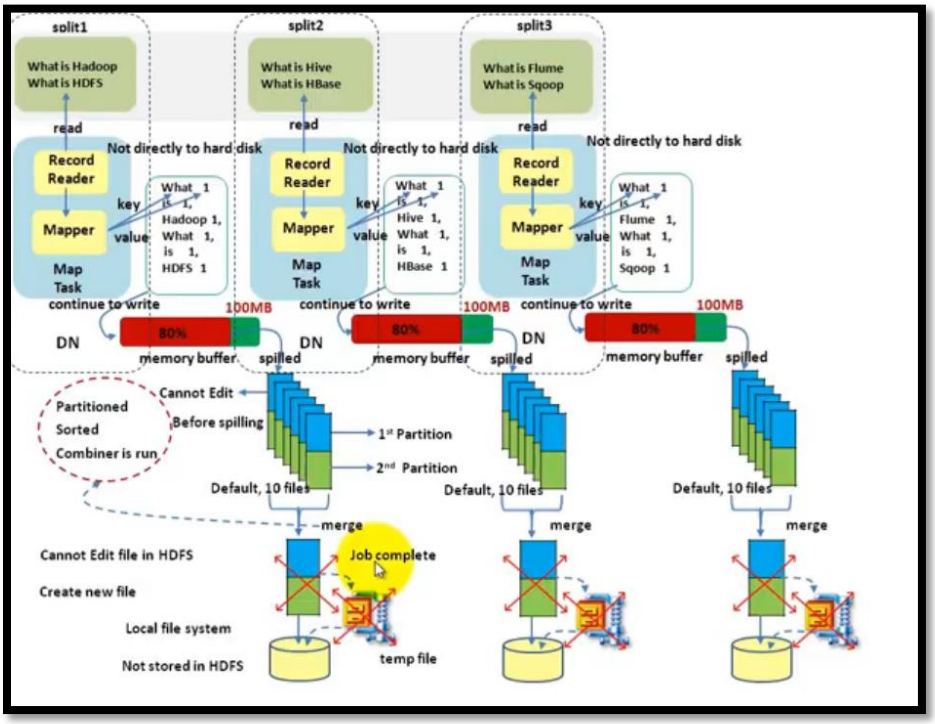
So for every threshold one new file is being created



Before the map task finishes spilled files are merged into a single file by default 10 files are merged at a time, while merging partition, sorting and combiner operation performed again on this file, you can also compressed the file and store it on to disk, finally we will have single merged file from each mapper

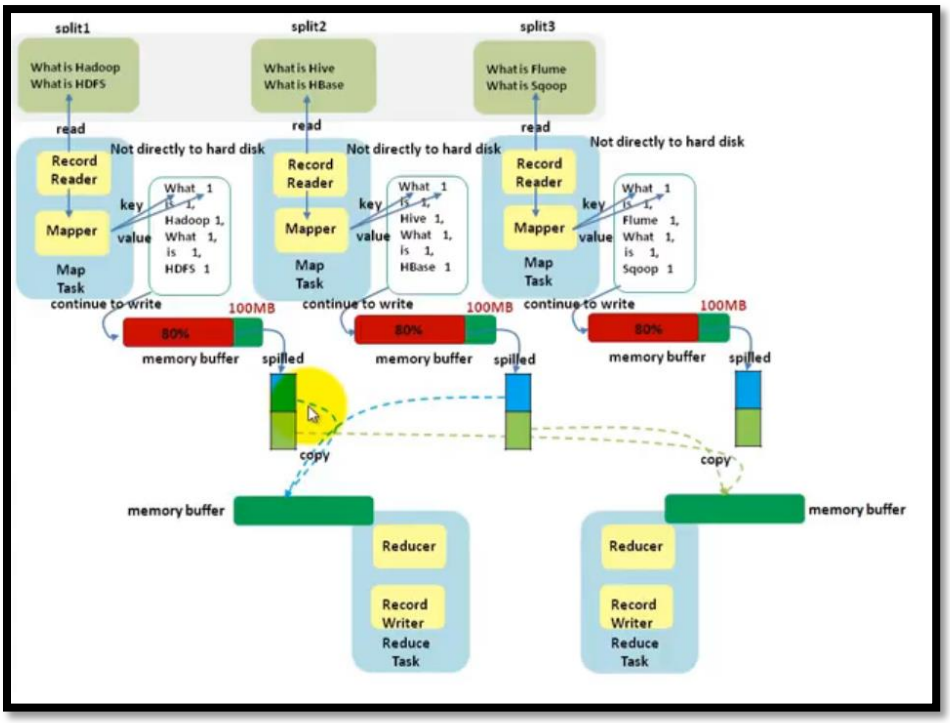


These files are stored on local file system **not on HDFS** of data node where the map task is running, these files are temp files and all are deleted once job finished

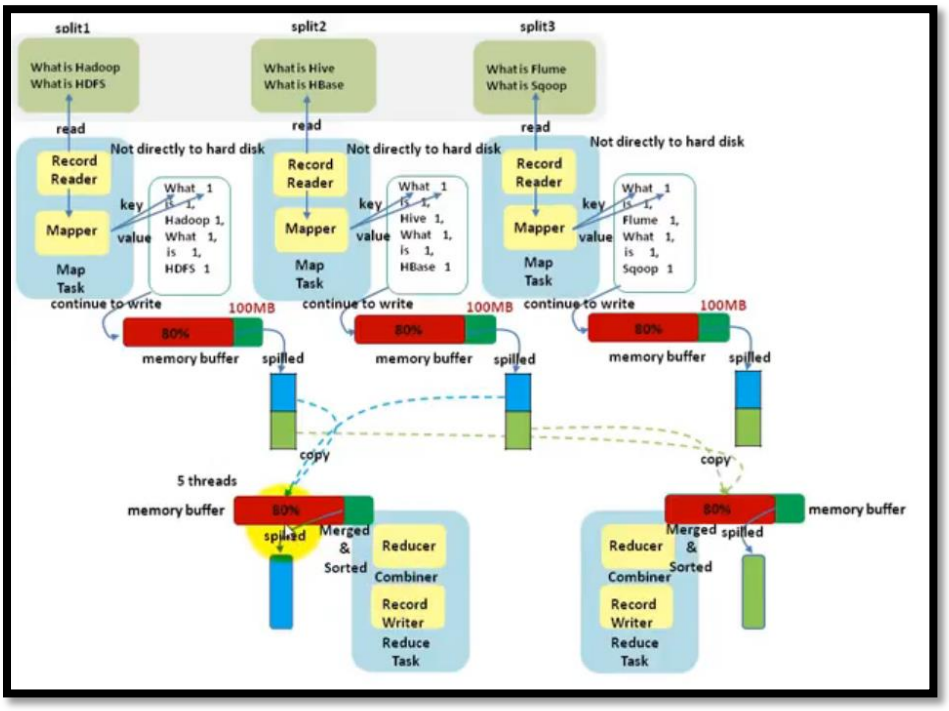


Reduce task starts copying the file from the map output, reduce task will not wait for all the map task to complete , as soon as one map task is complete reduce task start copying the file, that is why some time you see in log that reducer starts before all mapper finish, in that case reducer start copying the file.

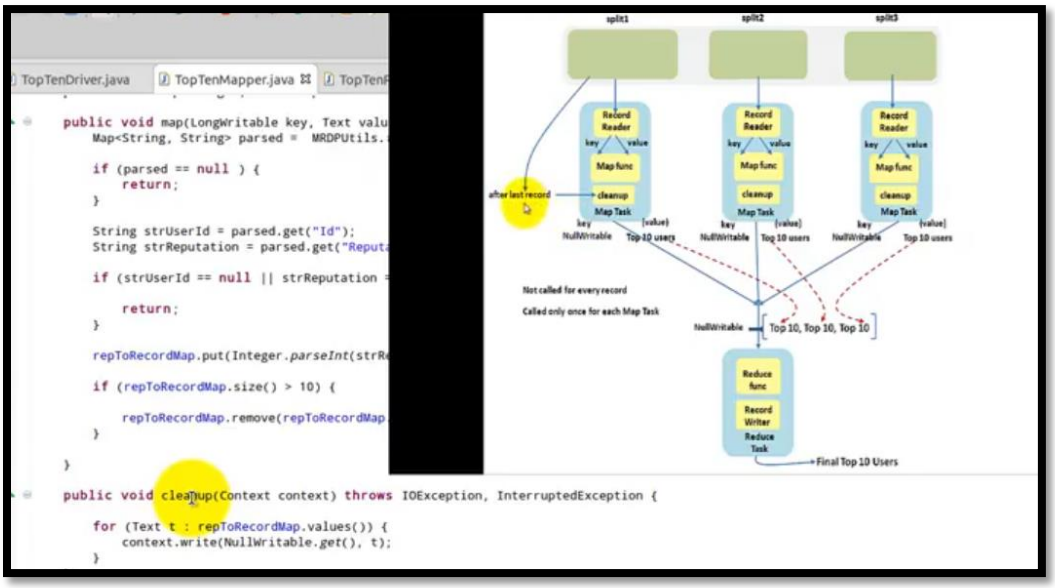
By default reduce task use 5 threads to copying data from map output in parallel



It copies the data to its memory buffer same as mapper when 80% of the buffer is consumed, records are spilled to the disk, before spilling it is merged and sorted again the combiner function run here so that lesser records are written to the disk

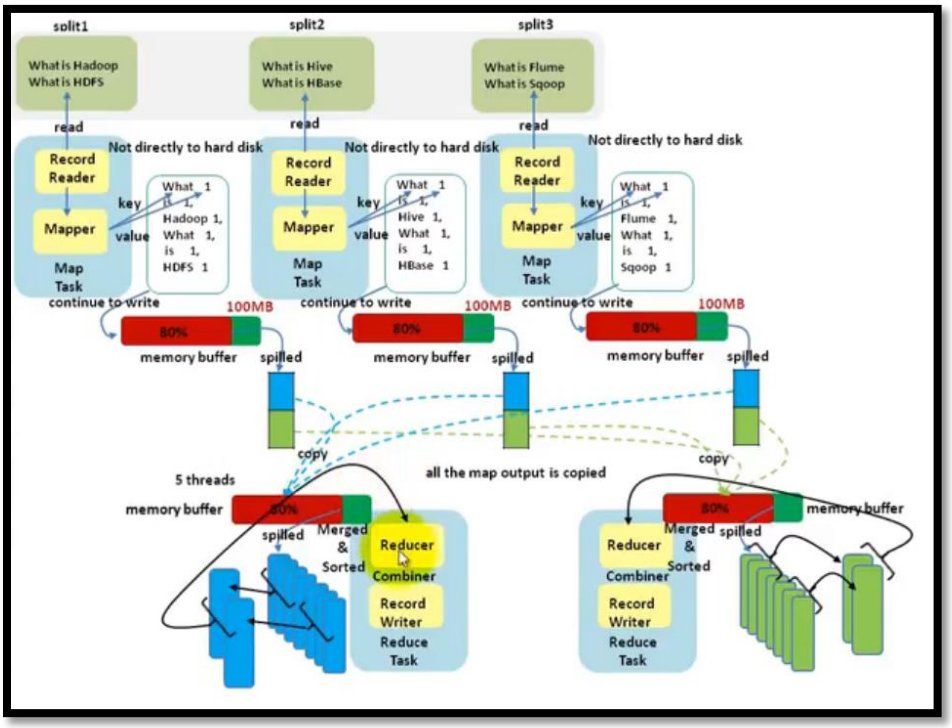


Cleanup function call at the end of map task when all records are processed

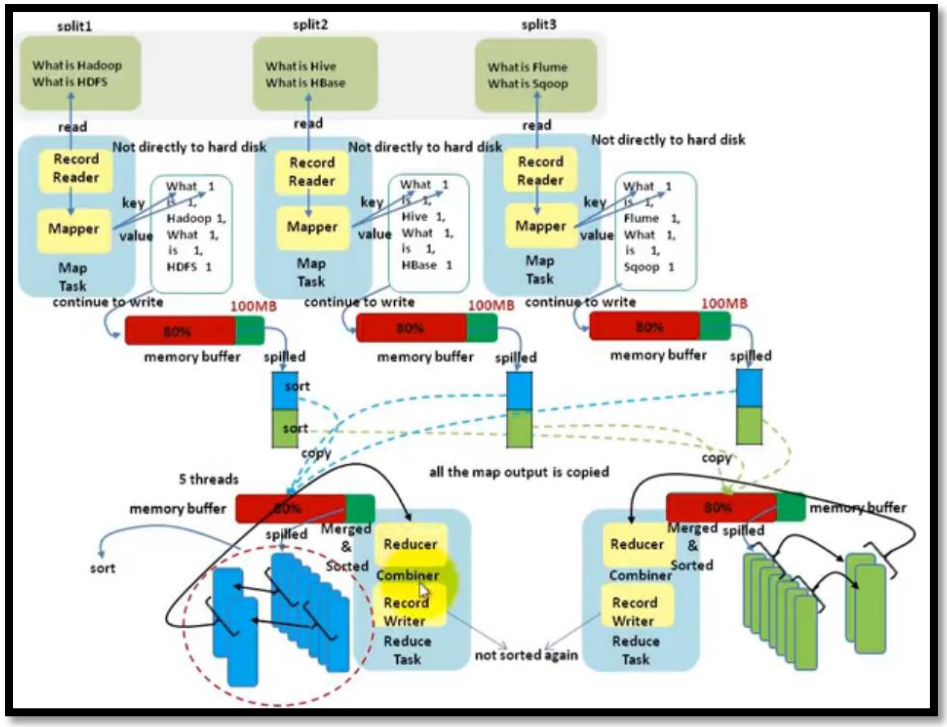


Same as map task here also we could have several spills files which will be merged into large sorted files from background thread

After all the map output is copied the remaining spills file are merged together and passed to the Reduce function

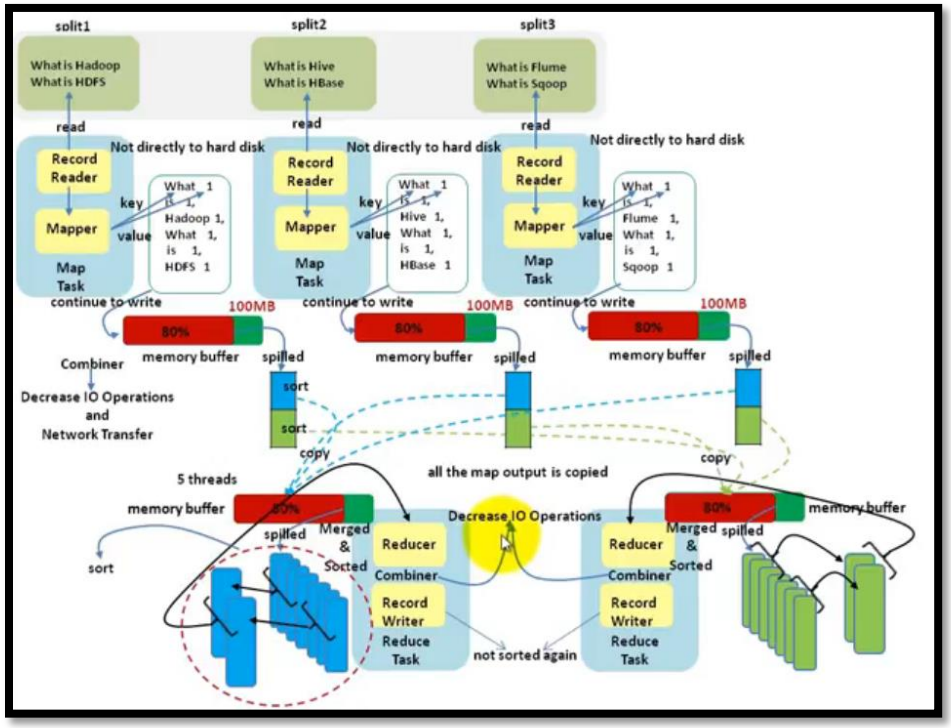


So sorting is done in the map phase as well as in reduce phase, in reduce phase it is done before the reduce function is called, reduce output is not sorted again



Apart from sorting combiner function used in the map phase as well as in the reduce phase

In the map phase it is used to decrease the IO operation and Network traffic while in reduce phase it is used to decrease the number of input output (IO) Operations

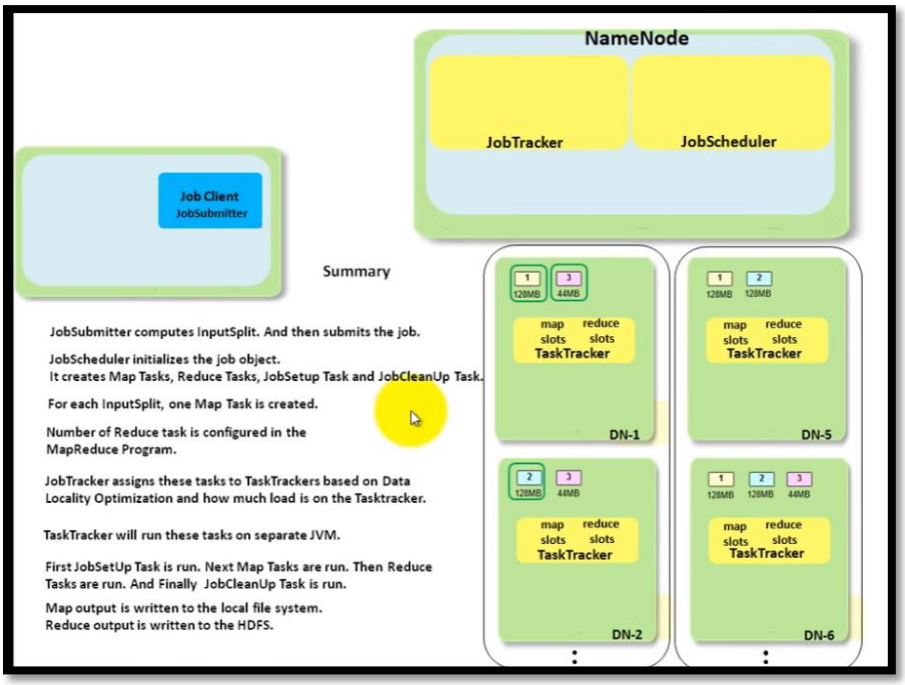


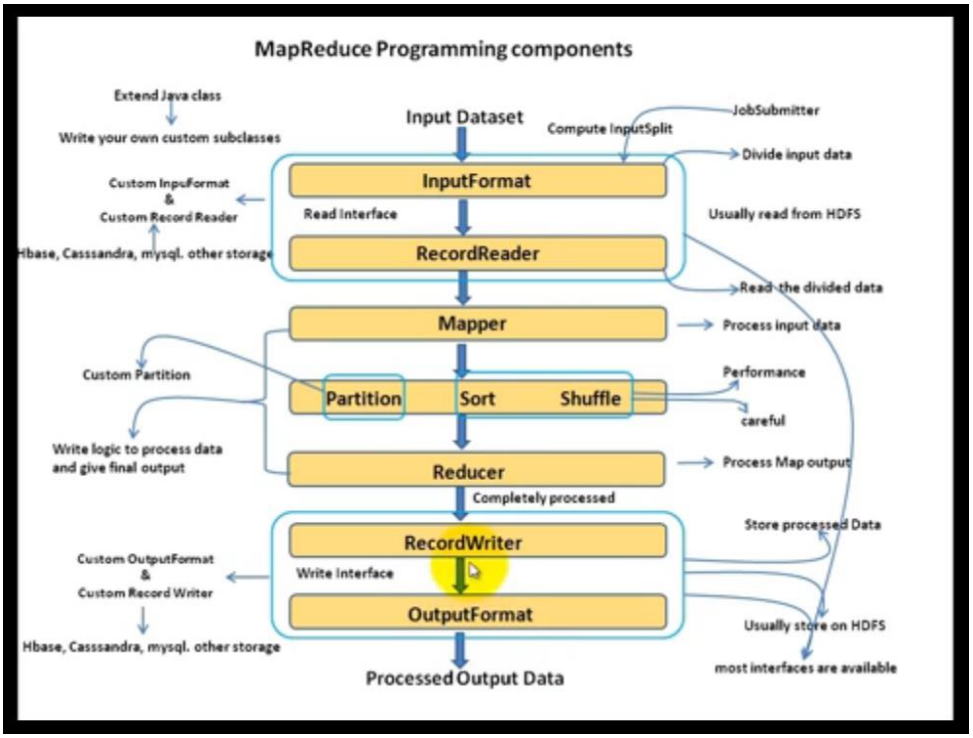
https://www.youtube.com/watch?v=F4Zc4S-8n0w

pramod narayana

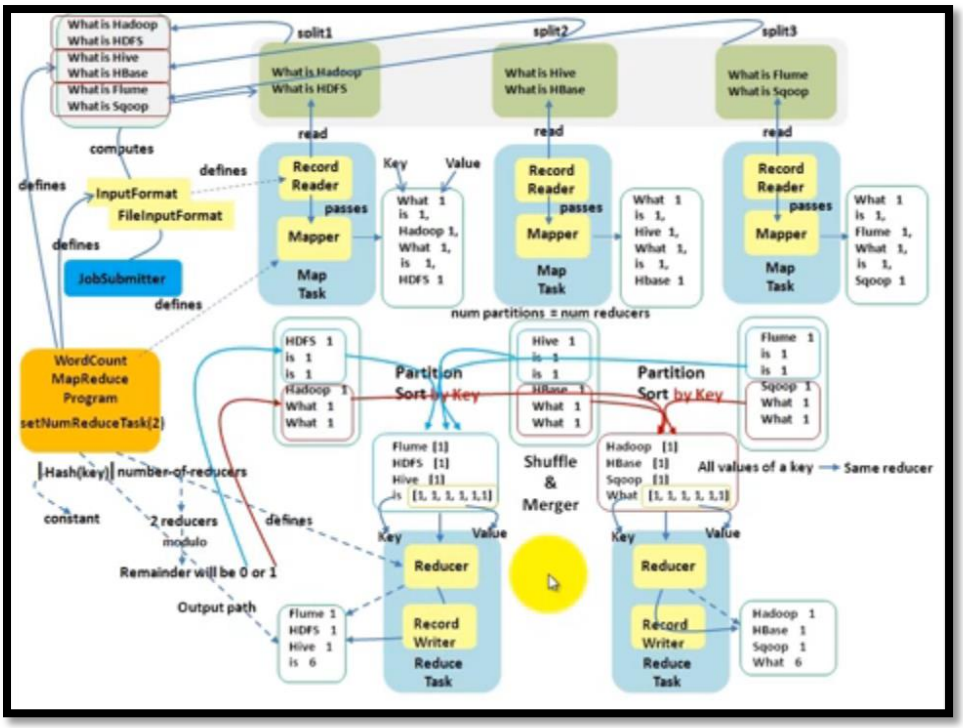
https://www.youtube.com/channel/UCDApOTqXw80sSxKdyQpcnUA

Map Reduce version 1 flow





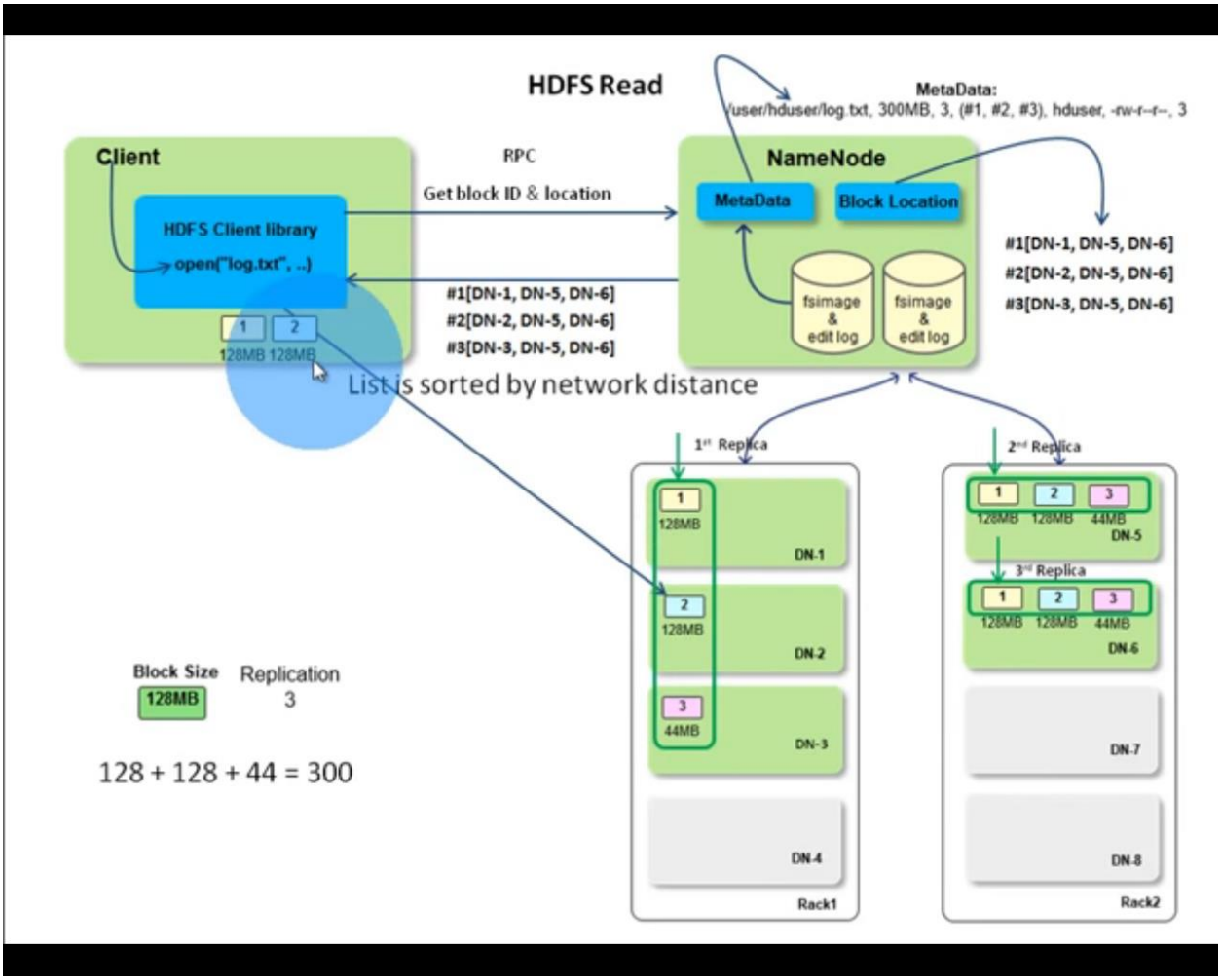
Word count flow



HDFS READ

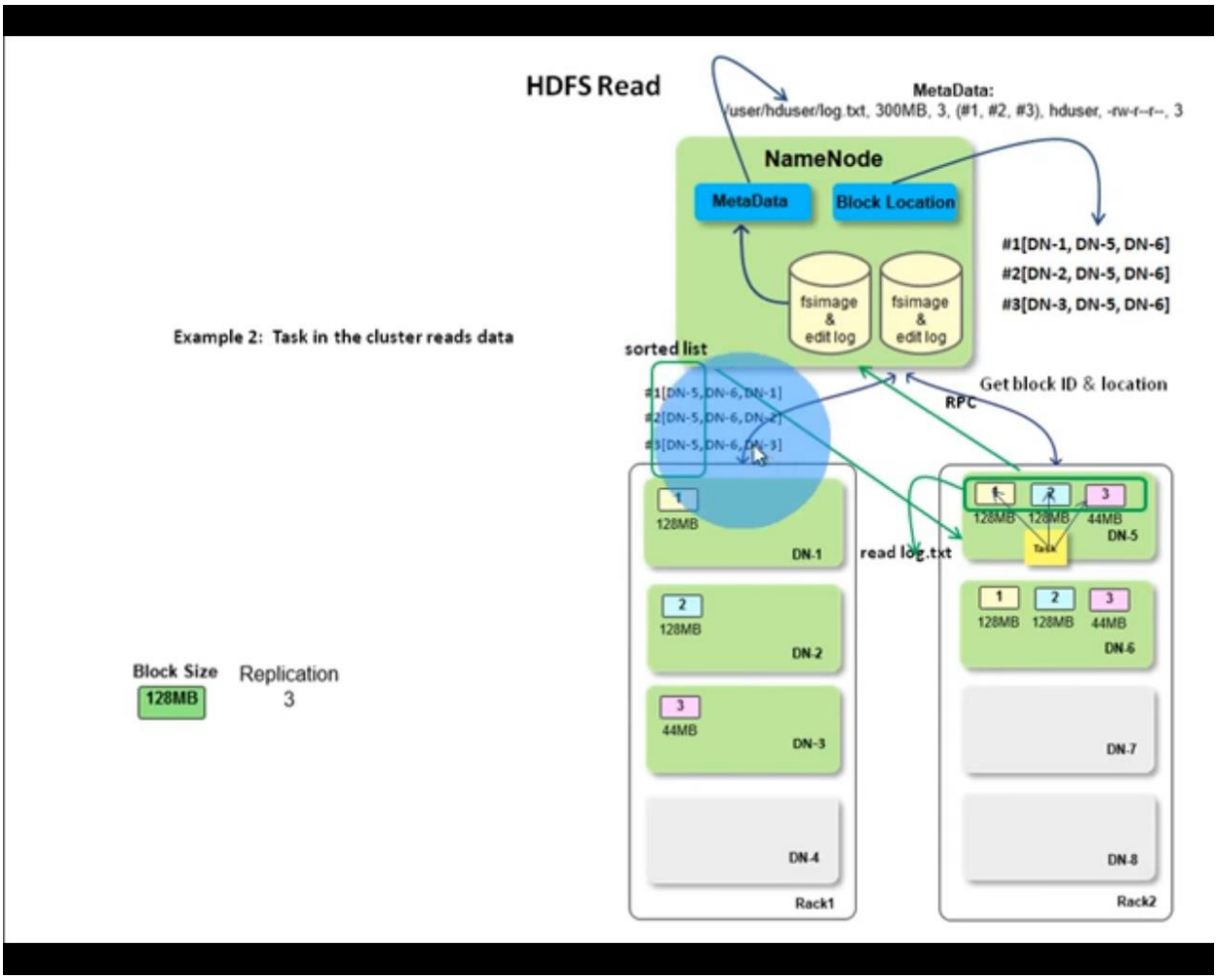
Client outside cluster

Block location returned is sorted distance from client

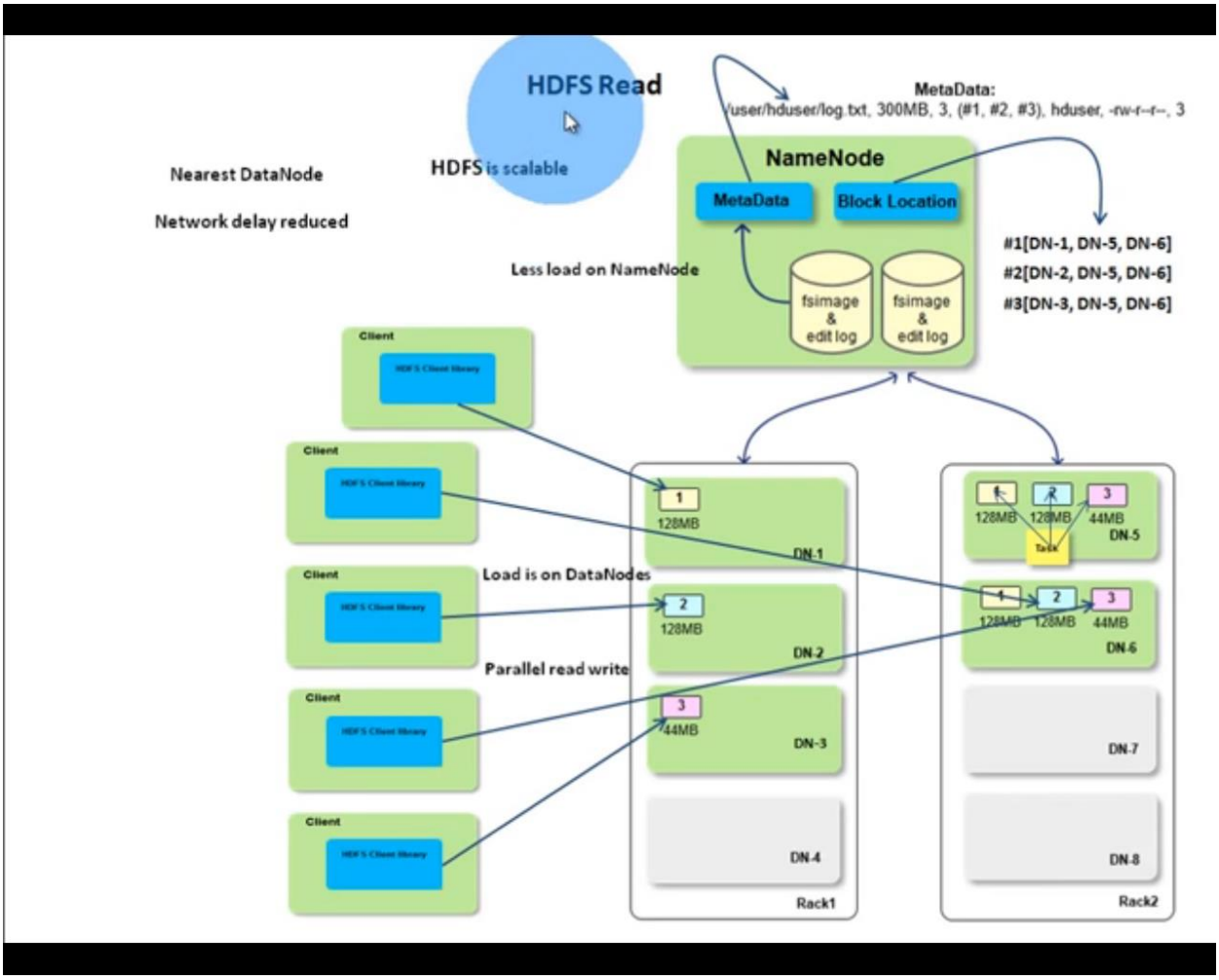


Client inside cluster

Here client is running in DN-5 and that's why now DN-5 comes first in block location

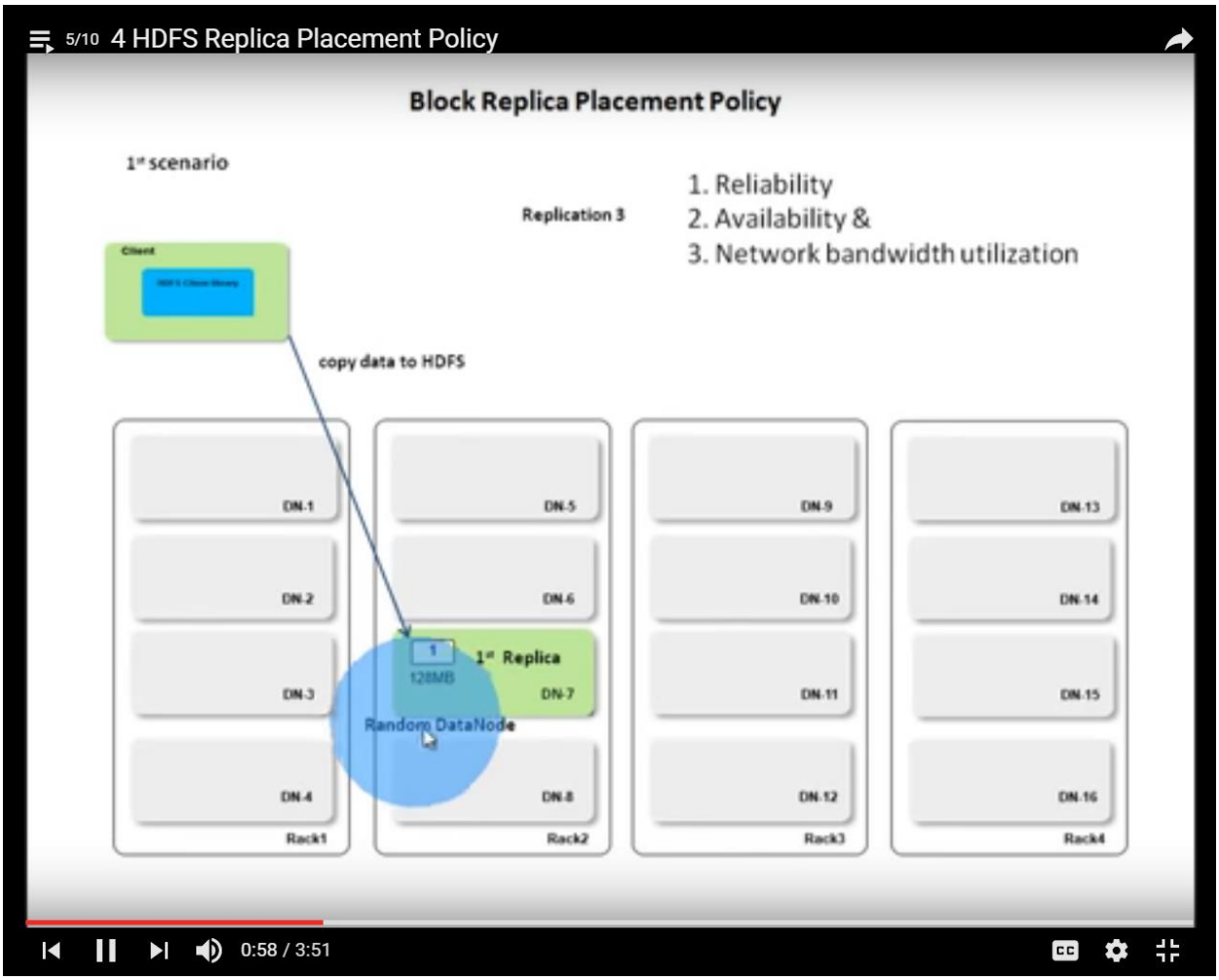


Client directly communicate with data node as per block location given by namenode and that is why name node is not overloaded



Data Replication policy

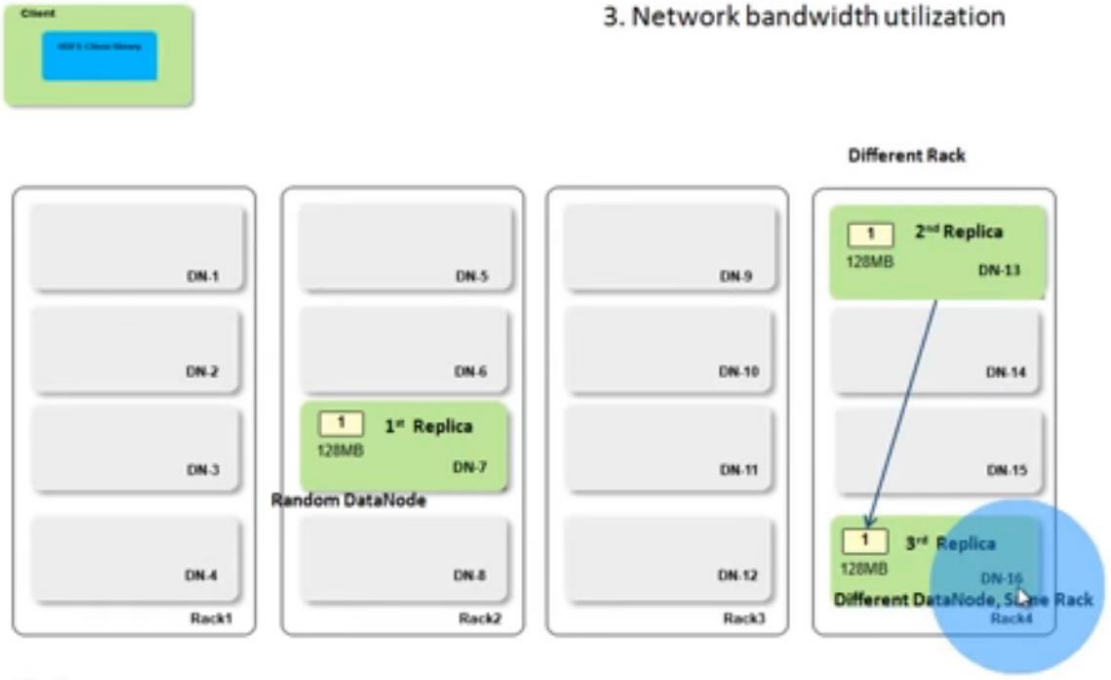
Client is outside cluster



Block Replica Placement Policy

Replication 3

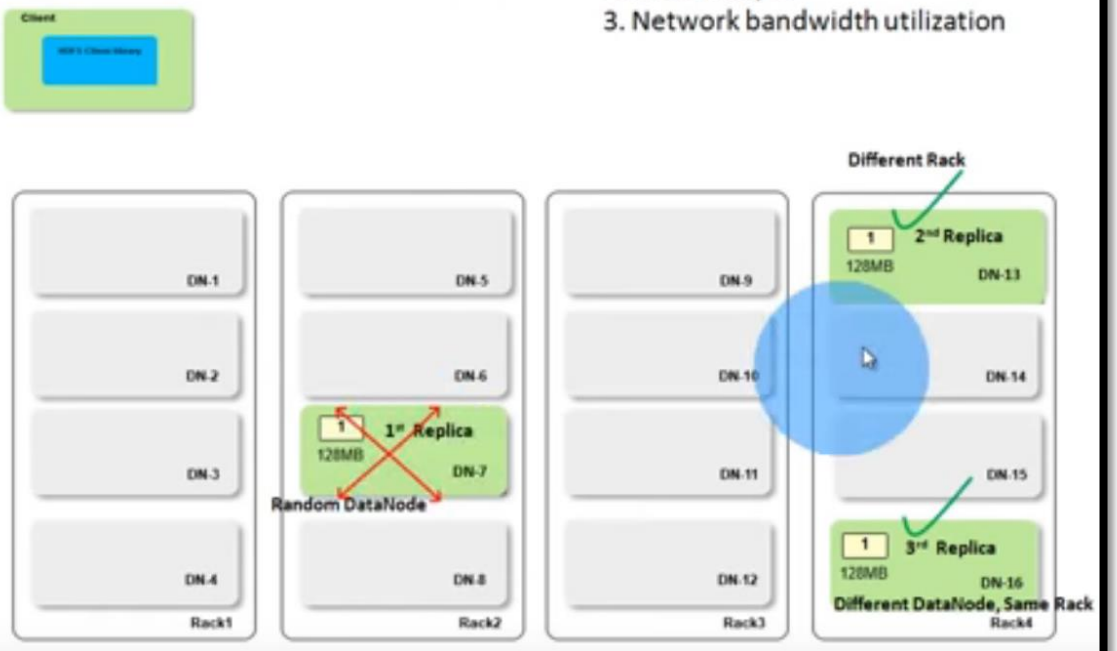
1. Reliability
2. Availability &
3. Network bandwidth utilization



Block Replica Placement Policy

Replication 3

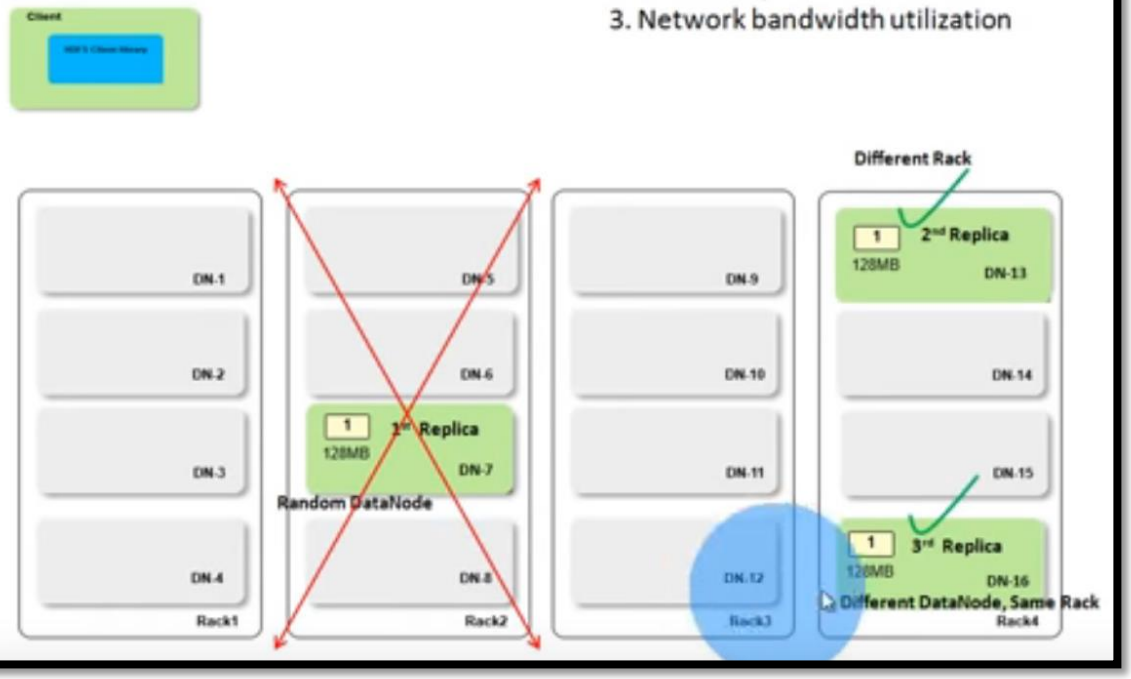
1. Reliability
2. Availability &
3. Network bandwidth utilization



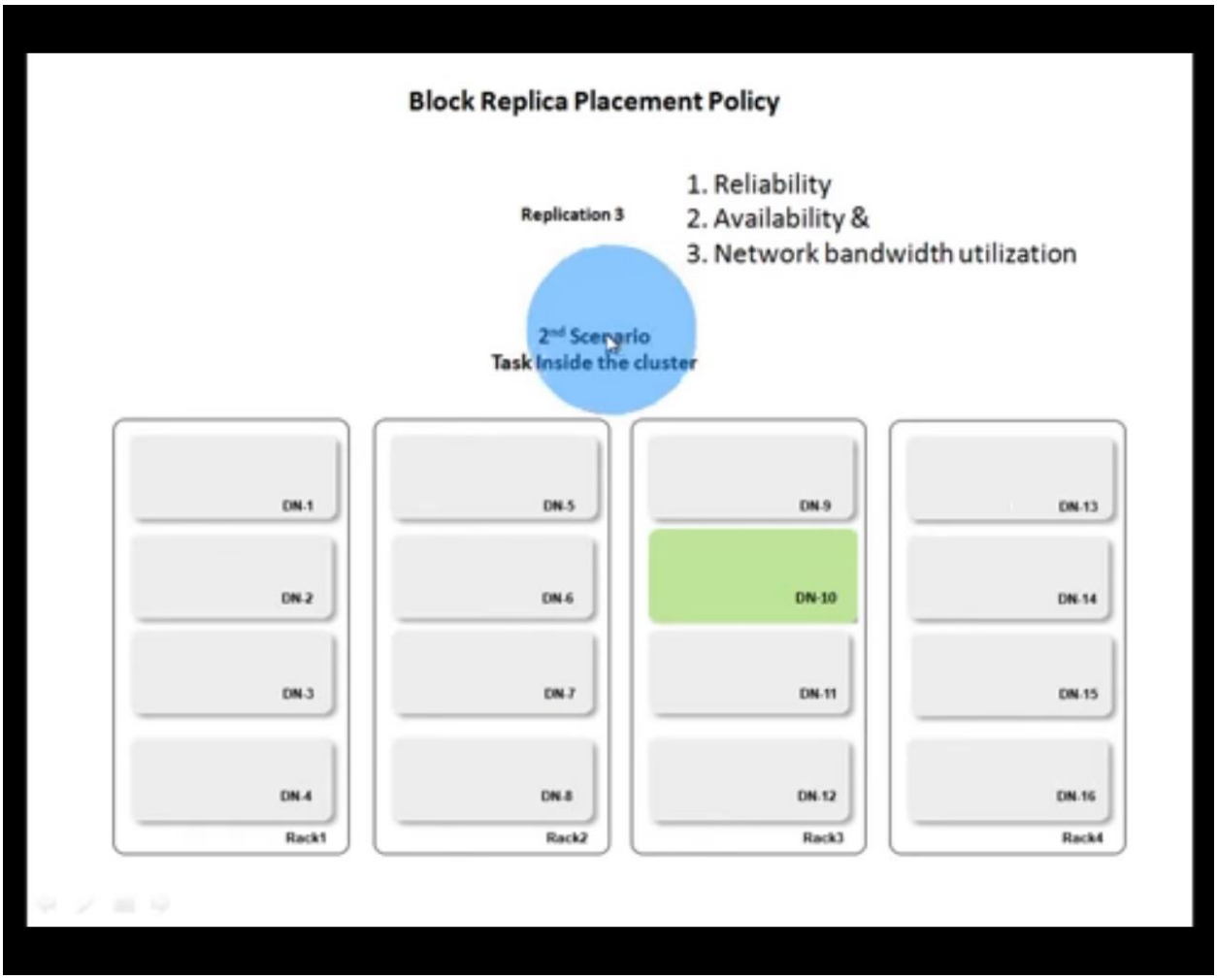
Block Replica Placement Policy

Replication 3

1. Reliability
2. Availability &
3. Network bandwidth utilization



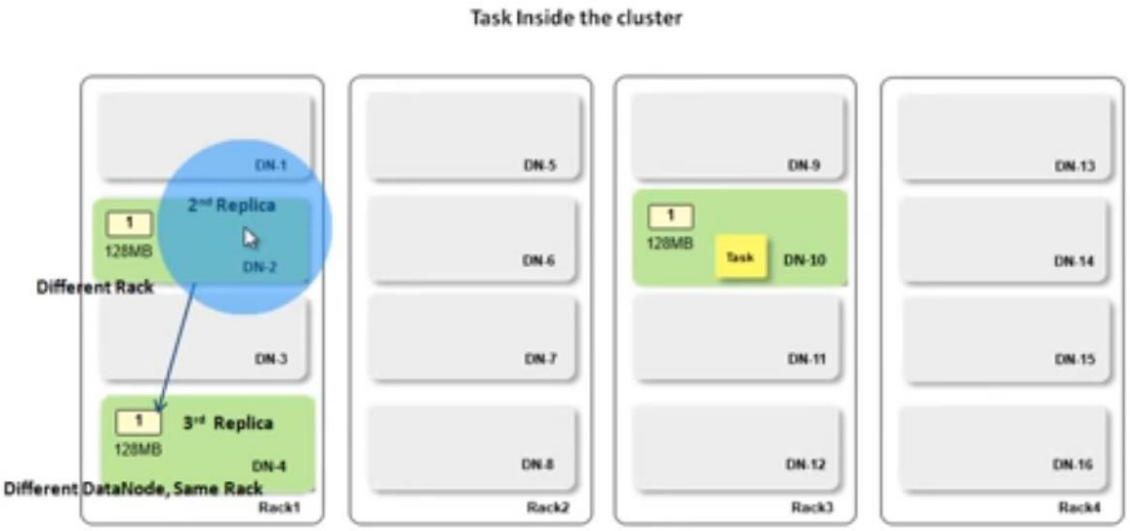
Client inside cluster



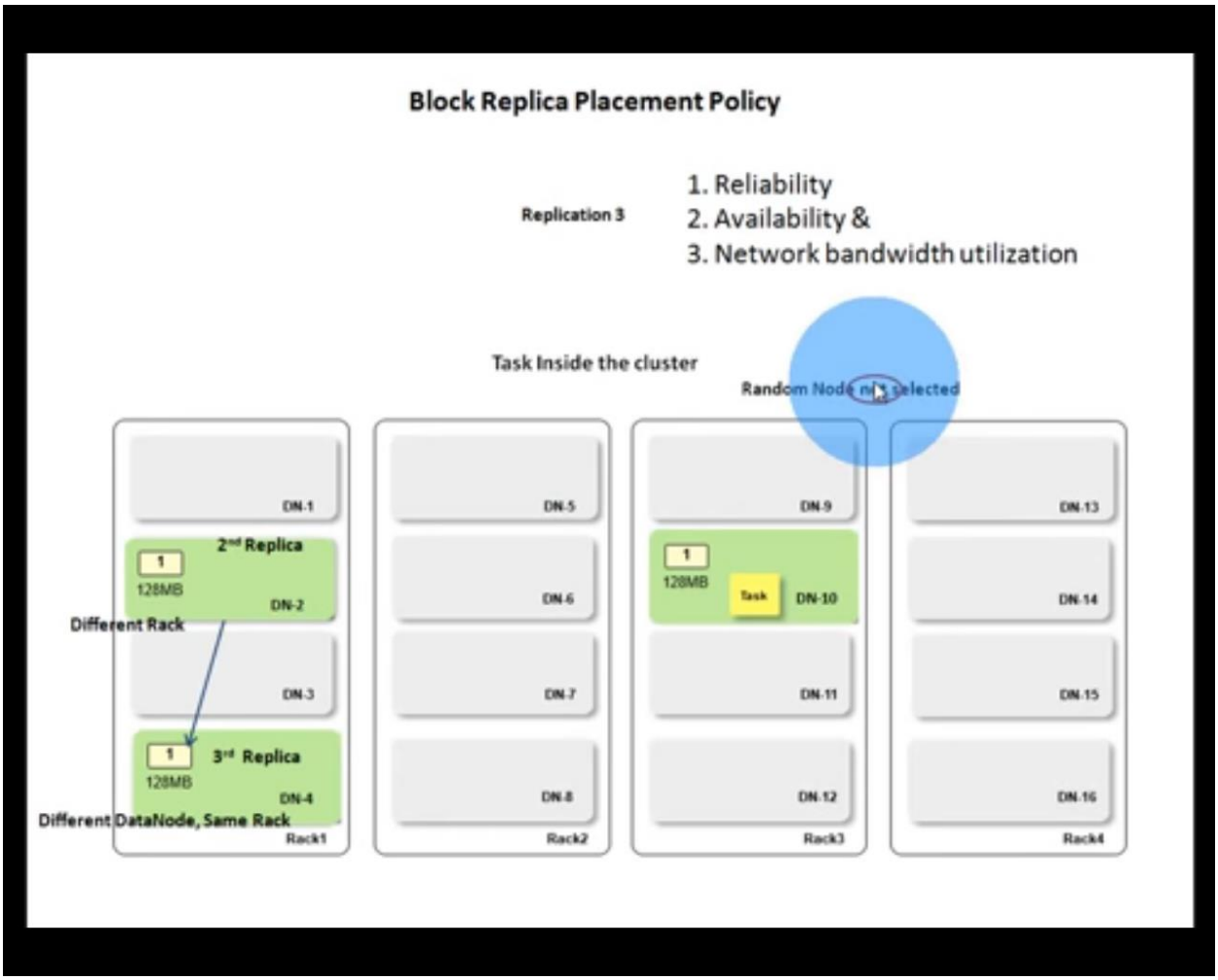
Block Replica Placement Policy

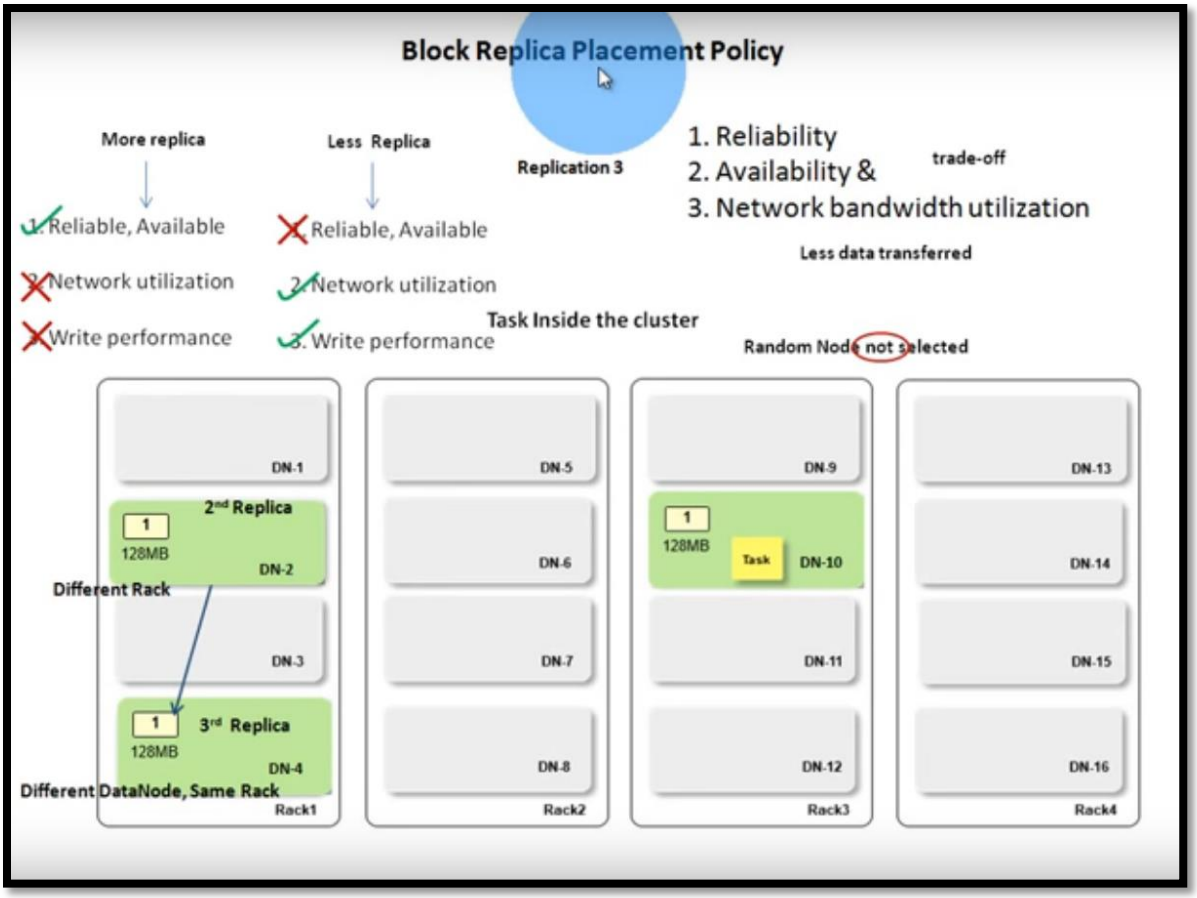
Replication 3

1. Reliability
2. Availability &
3. Network bandwidth utilization

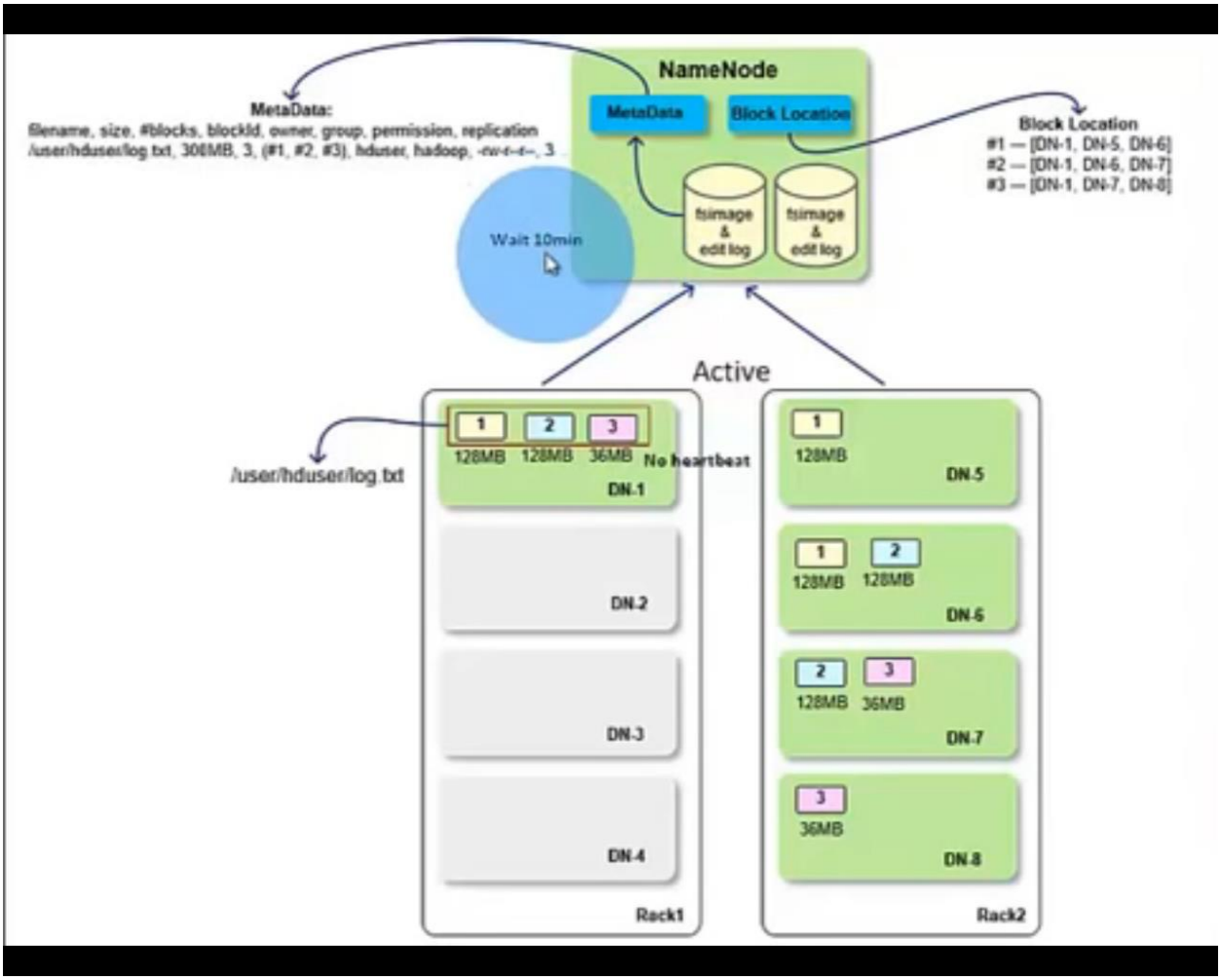


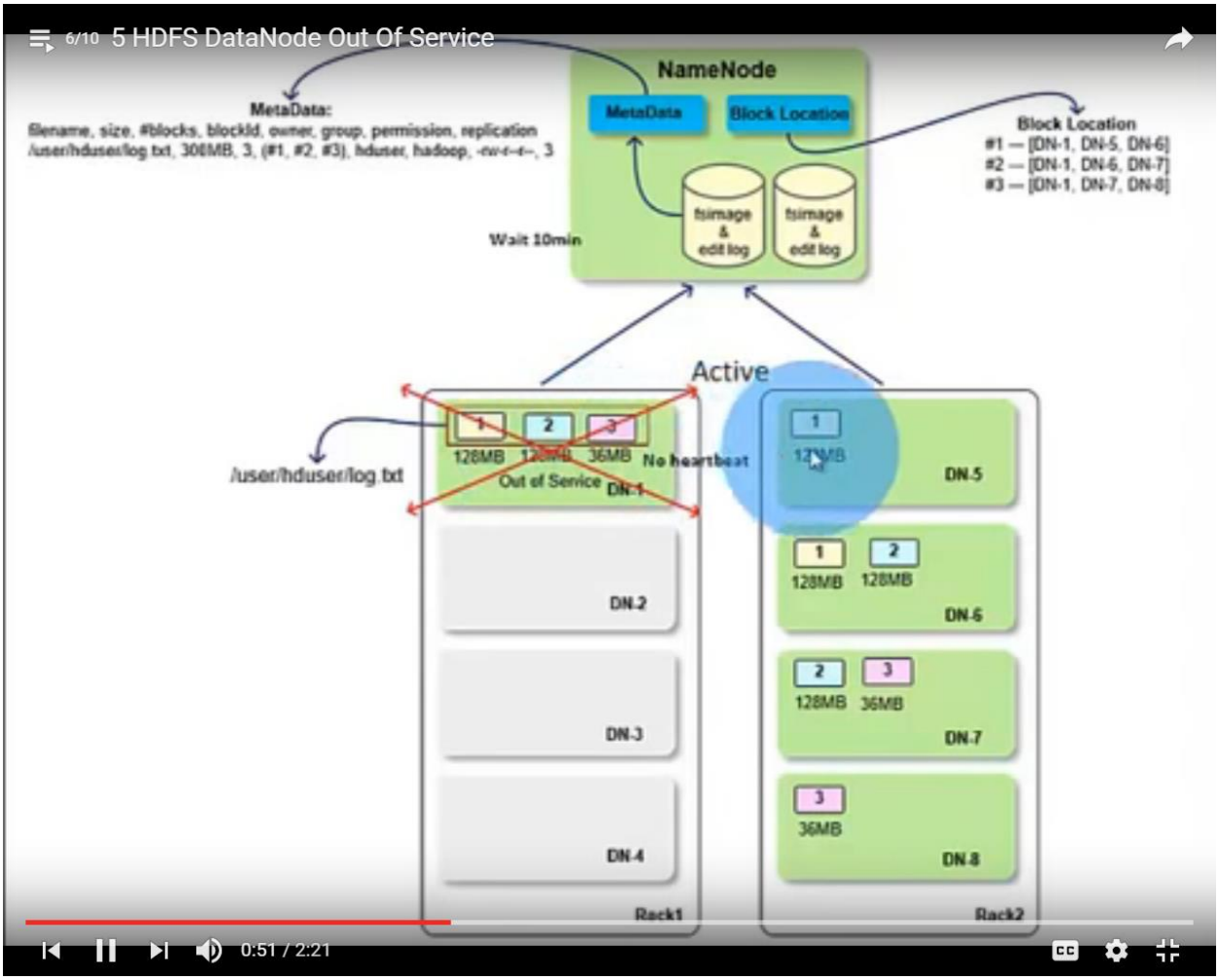
Written in same datanode where task is running

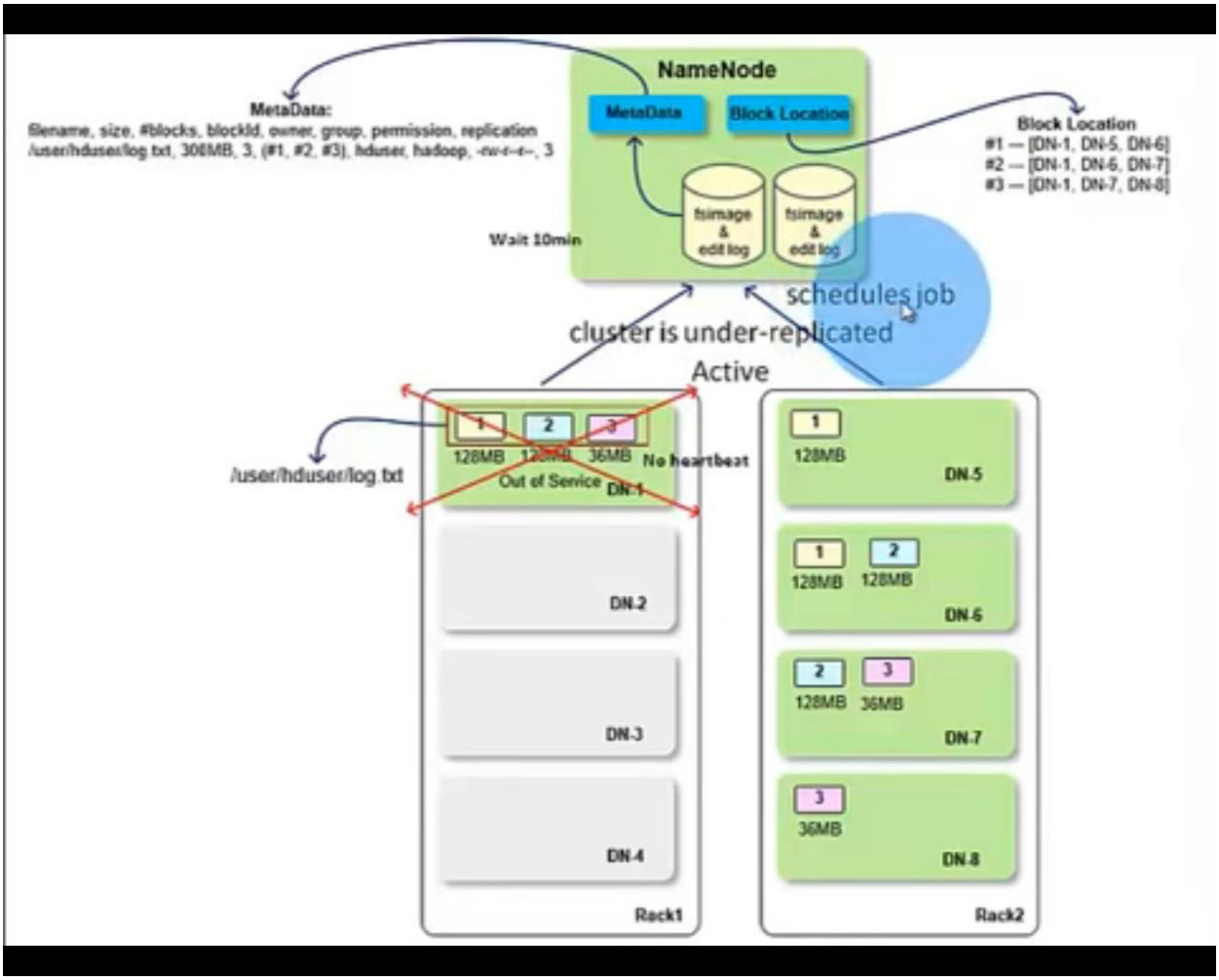


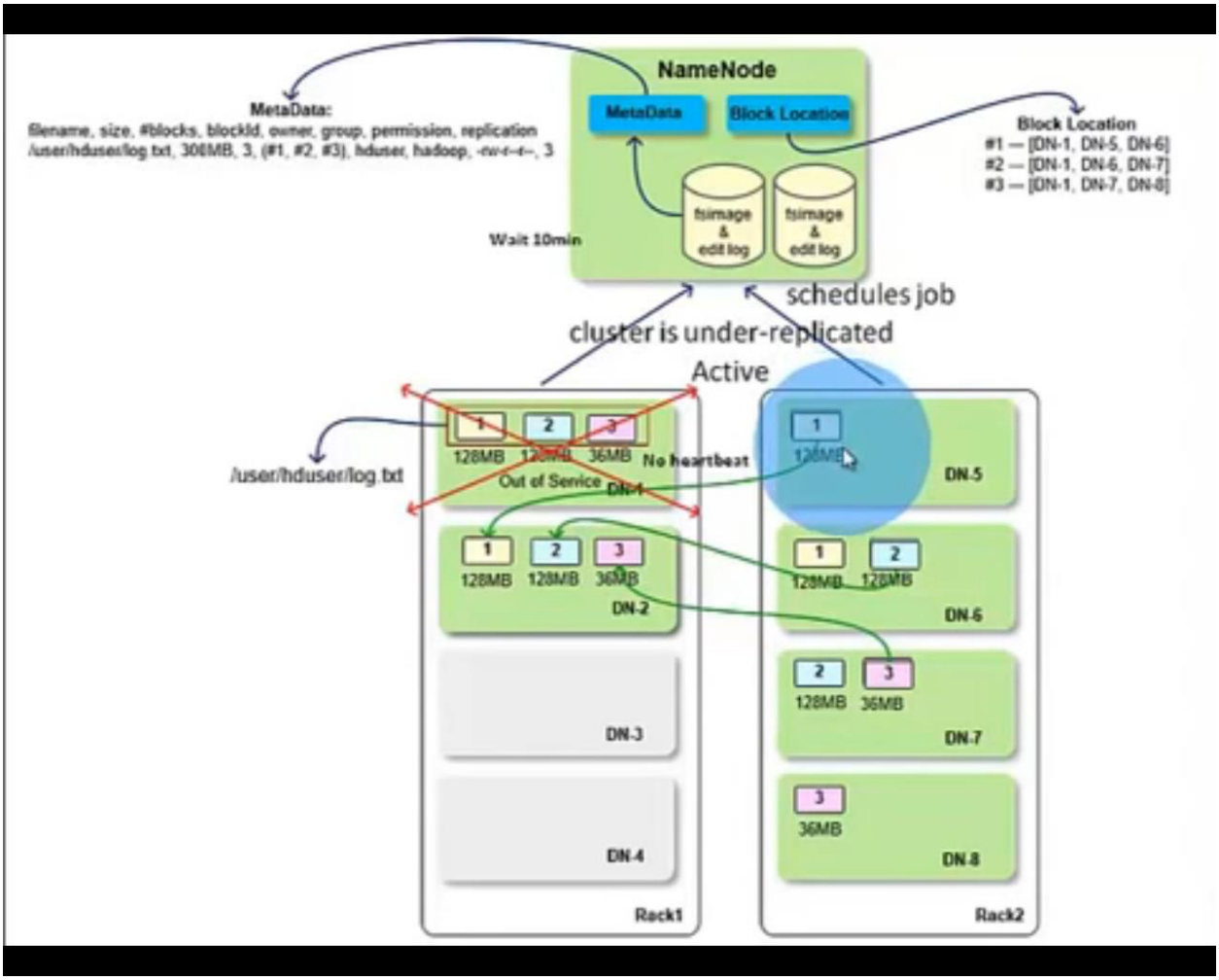

Block Replica Placement Policy
More replica
Less Replica
Replication 3
1. Reliability
2. Availability &
3. Network bandwidth utilization
trade-off
1. Reliable, Available
2. Network utilization
3. Write performance
1. Reliable, Available
2. Network utilization
3. Write performance
Less data transferred
Task Inside the cluster
Random Node not selected
DN-1
2nd Replica
1
128MB
DN-2
Different Rack
DN-3
1
3rd Replica
128MB
DN-4
Different DataNode, Same Rack
Rack1
DN-5
DN-6
DN-7
DN-8
Rack2
DN-9
1
128MB
Task
DN-10
DN-11
DN-12
Rack3
DN-13
DN-14
DN-15
DN-16
Rack4

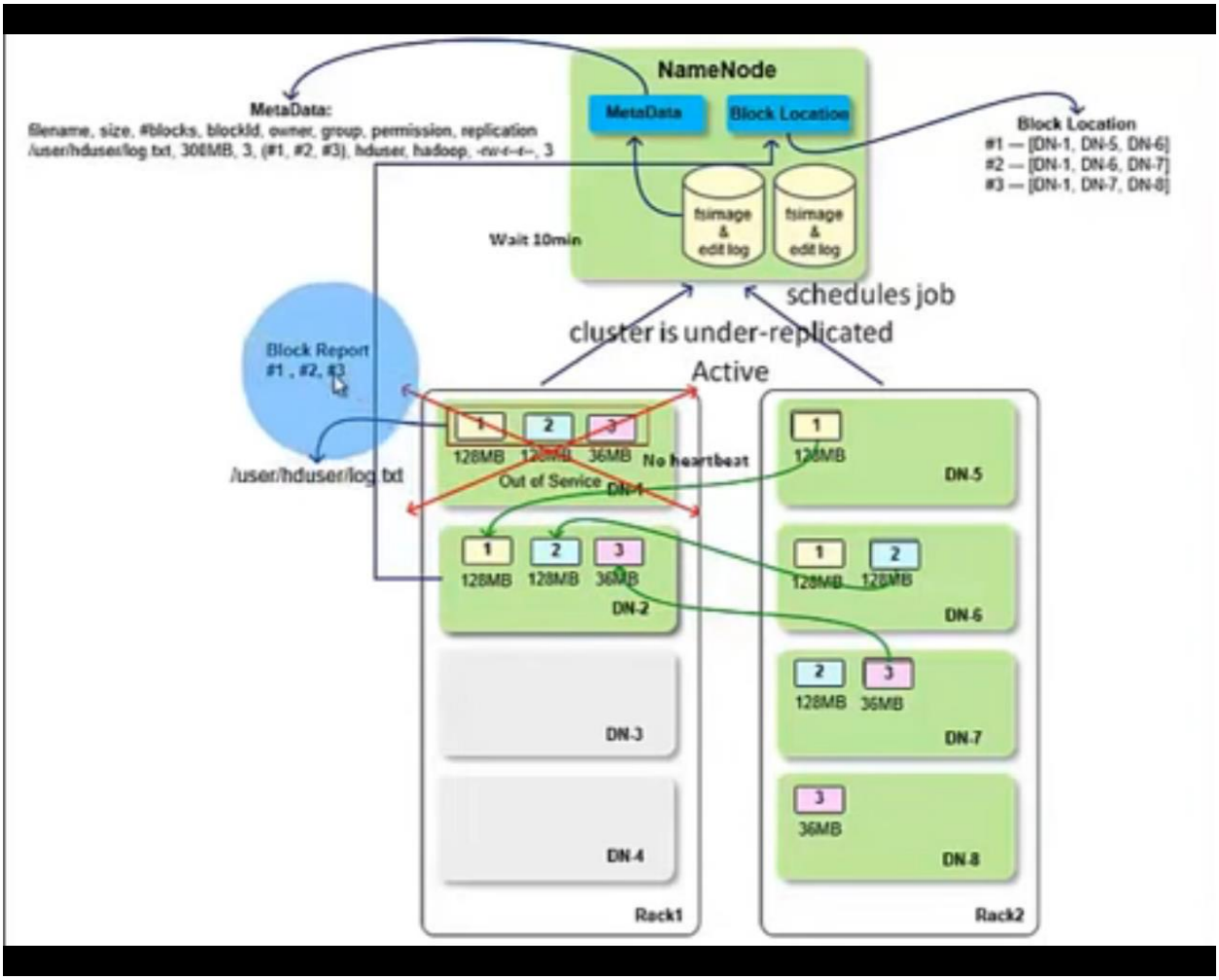
DataNode Out of Service

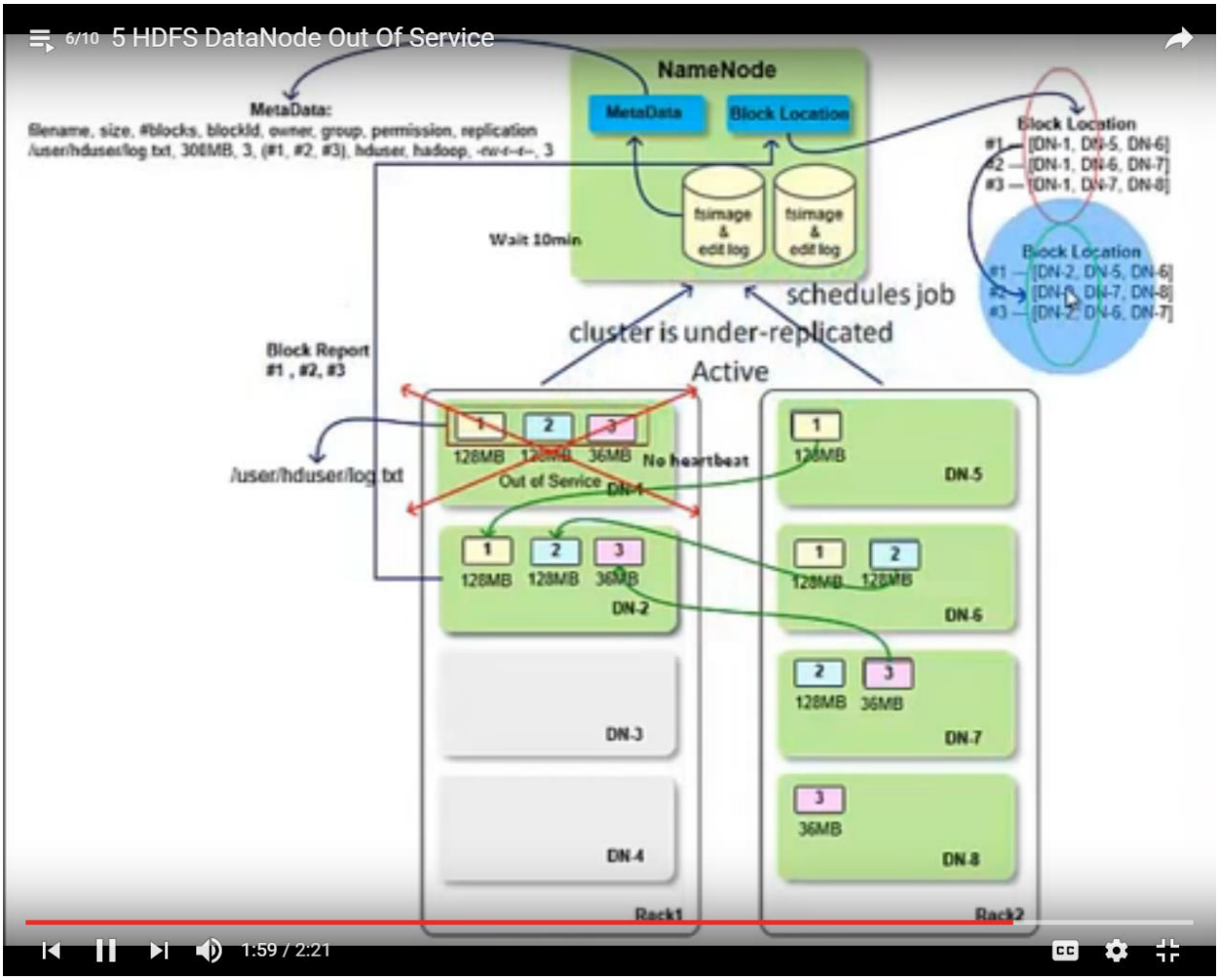



6/10 5 HDFS DataNode Out Of Service
NameNode
MetaData
Block Location
MetaData:
filename, size, #blocks, blockId, owner, group, permission, replication
/user/hduser/log.txt, 308MB, 3, (#1, #2, #3), hduser, hadoop, -rw-r--r--, 3
Block Location
#1 — [DN-1, DN-5, DN-6]
#2 — [DN-1, DN-6, DN-7]
#3 — [DN-1, DN-7, DN-8]
fsimage & edit log
fsimage & edit log
Wait 10min
Active
/user/hduser/log.txt
128MB
128MB
36MB
Out of Service
No heartbeat
DN-1
DN-2
DN-3
DN-4
DN-5
DN-6
DN-7
DN-8
Rack1
Rack2
0:51 / 2:21

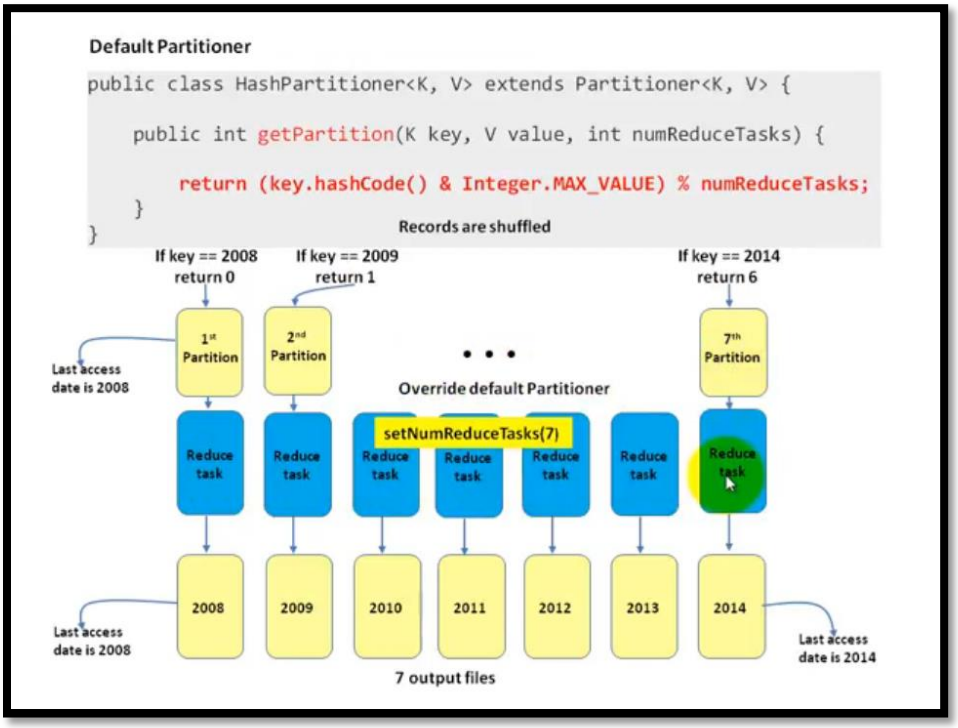

NameNode
MetaData
Block Location
MetaData:
filename, size, #blocks, blockId, owner, group, permission, replication
/user/hduser/log.txt, 308MB, 3, (#1, #2, #3), hduser, hadoop, -rw-r--r--, 3
Block Location
#1 — [DN-1, DN-5, DN-6]
#2 — [DN-1, DN-6, DN-7]
#3 — [DN-1, DN-7, DN-8]
fsimage & edit log
fsimage & edit log
Wait 10min
schedules job
cluster is under-replicated
Active
/user/hduser/log.txt
128MB
128MB
36MB
No heartbeat
Out of Service
DN-1
DN-2
DN-3
DN-4
Rack1
DN-5
DN-6
DN-7
DN-8
Rack2

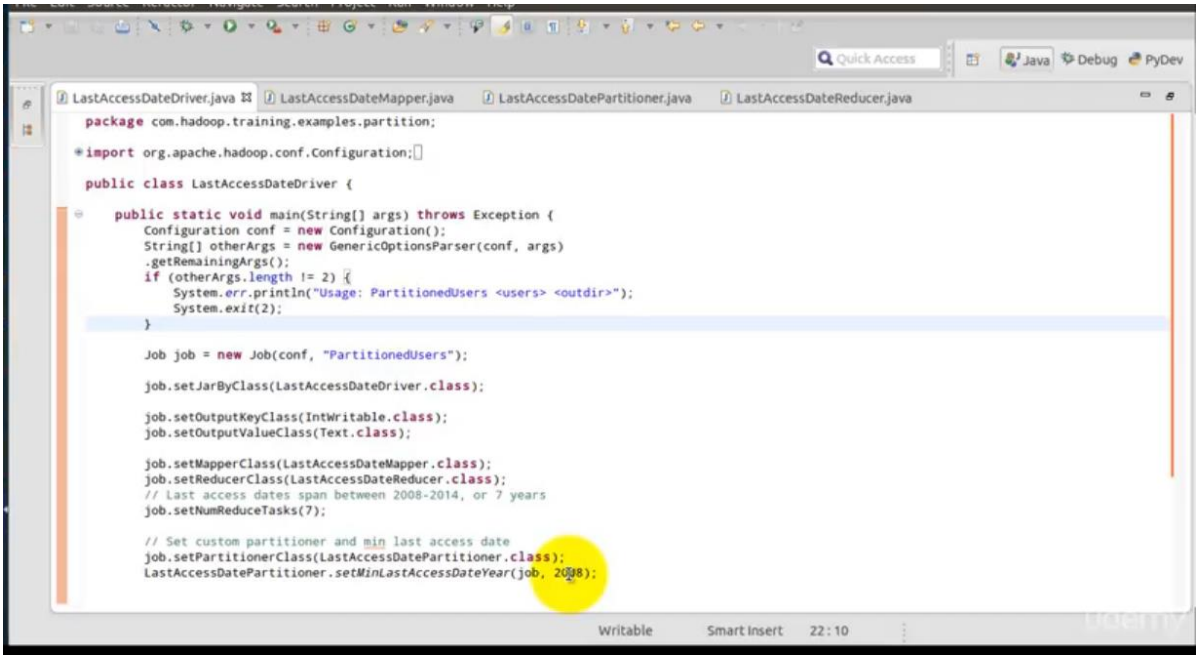

NameNode
MetaData
Block Location
MetaData:
filename, size, #blocks, blockId, owner, group, permission, replication
/user/hduser/log.txt, 306MB, 3, (#1, #2, #3), hduser, hadoop, -rw-r--r--, 3
Block Location
#1 — [DN-1, DN-5, DN-6]
#2 — [DN-1, DN-6, DN-7]
#3 — [DN-1, DN-7, DN-8]
fsimage & edit log
fsimage & edit log
Wait 10min
schedules job
cluster is under-replicated
Active
/user/hduser/log.txt
128MB
128MB
36MB
No heartbeat
Out of Service
DN-1
DN-2
DN-3
DN-4
DN-5
DN-6
DN-7
DN-8
Rack1
Rack2

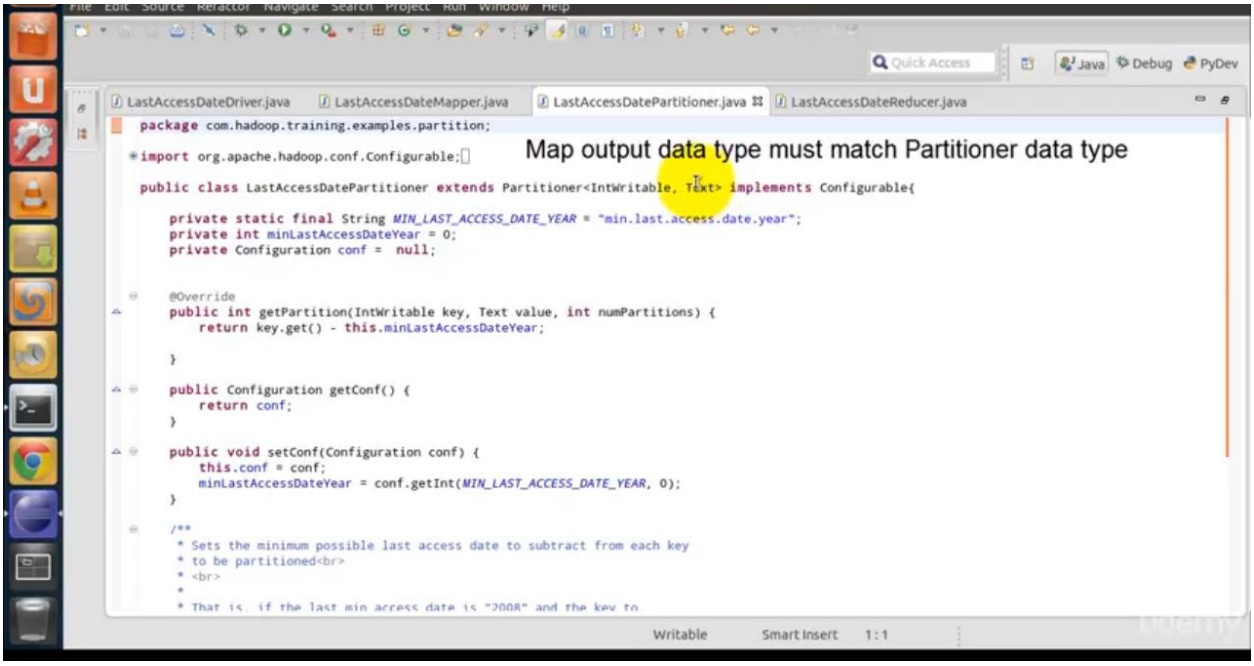

NameNode
MetaData
Block Location
MetaData:
filename, size, #blocks, blockId, owner, group, permission, replication
/user/hduser/log.txt, 306MB, 3, (#1, #2, #3), hduser, hadoop, -rw-r--r--, 3
Block Location
#1 — [DN-1, DN-5, DN-6]
#2 — [DN-1, DN-6, DN-7]
#3 — [DN-1, DN-7, DN-8]
fsimage & edit log
fsimage & edit log
Wait 10min
schedules job
cluster is under-replicated
Active
Block Report
#1 , #2, #3
/user/hduser/log.txt
128MB
128MB
36MB
No heartbeat
Out of Service
DN-1
DN-2
DN-3
DN-4
DN-5
DN-6
DN-7
DN-8
Rack1
Rack2

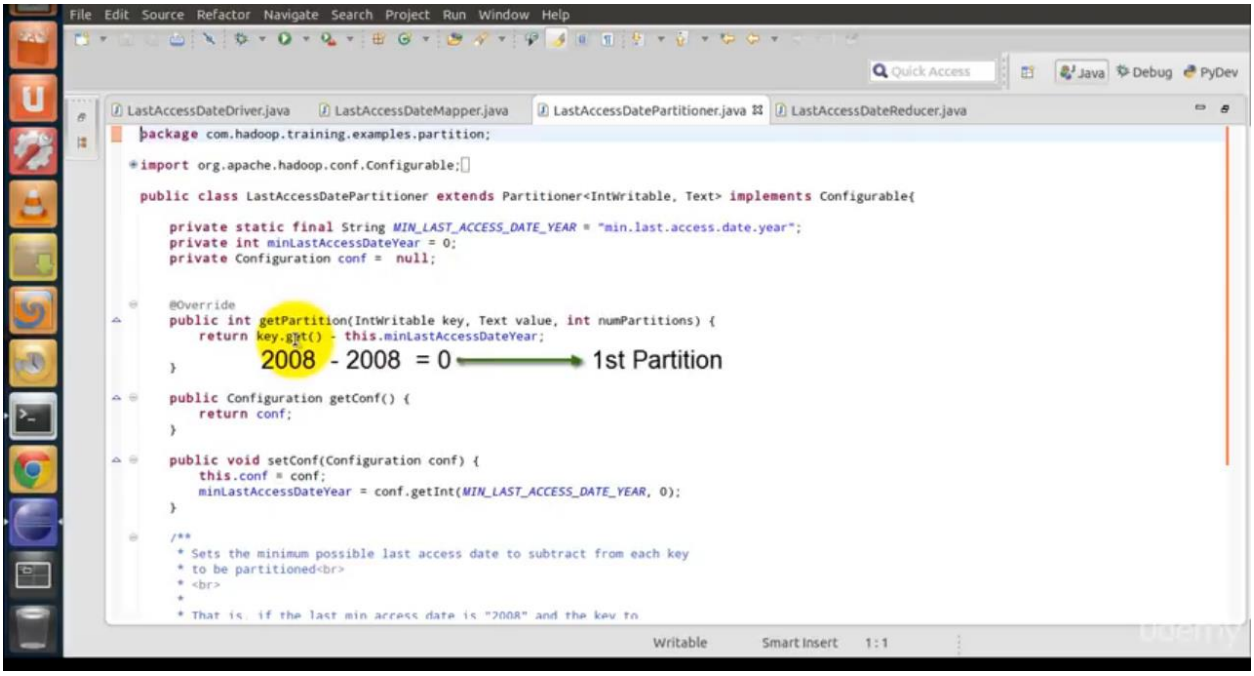

6/10 5 HDFS DataNode Out Of Service
NameNode
MetaData
Block Location
MetaData:
filename, size, #blocks, blockId, owner, group, permission, replication
/user/hduser/log.txt, 306MB, 3, (#1, #2, #3), hduser, hadoop, -rw-r--r--, 3
fsimage & edit log
fsimage & edit log
Wait 10min
Block Location
#1 — [DN-1, DN-5, DN-6]
#2 — [DN-1, DN-6, DN-7]
#3 — [DN-1, DN-7, DN-8]
Block Location
#1 — [DN-2, DN-5, DN-6]
#2 — [DN-2, DN-7, DN-8]
#3 — [DN-2, DN-6, DN-7]
schedules job
cluster is under-replicated
Active
Block Report
#1 , #2, #3
/user/hduser/log.txt
128MB
128MB
36MB
No heartbeat
Out of Service
DN-1
DN-2
DN-3
DN-4
DN-5
DN-6
DN-7
DN-8
Rack1
Rack2
1:59 / 2:21

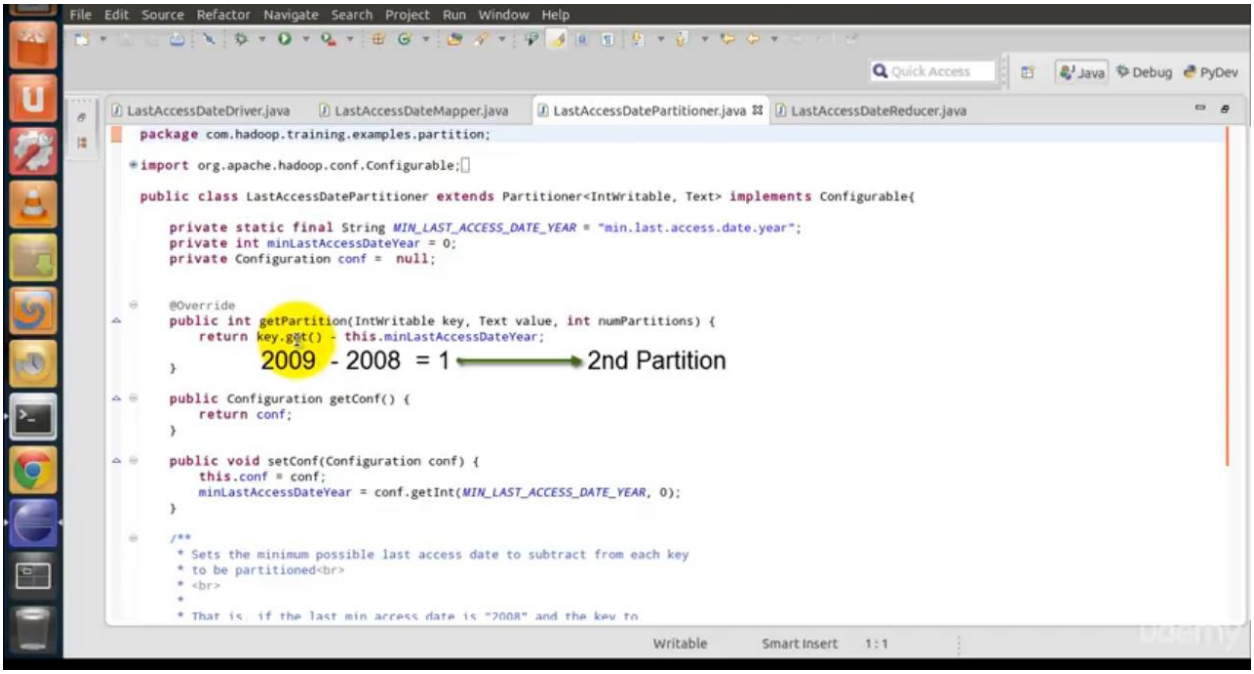
Custom Partition

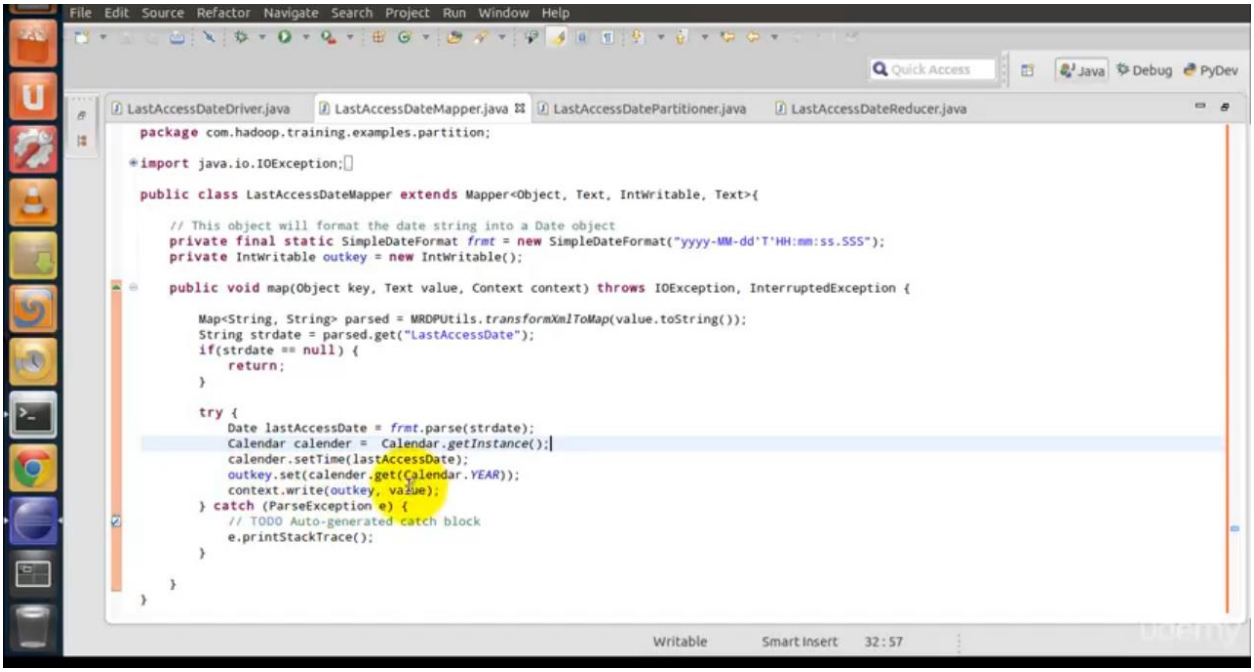


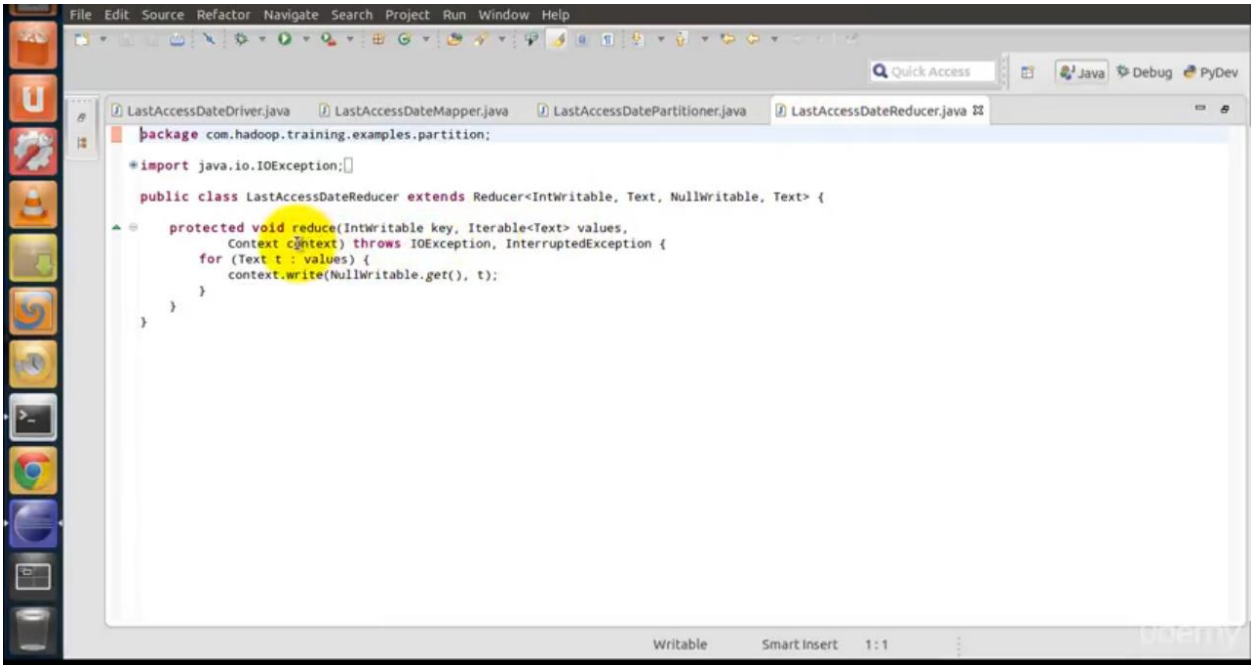


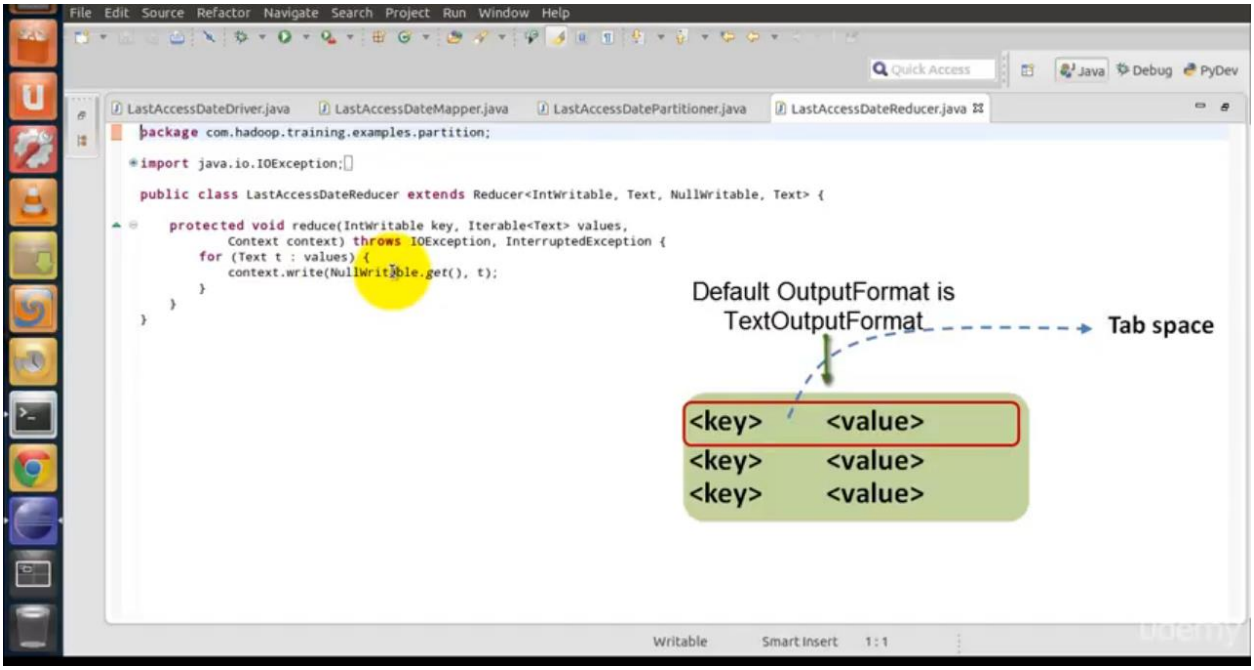

Map output data type must match Partitioner data type
LastAccessDateDriver.java
LastAccessDateMapper.java
LastAccessDatePartitioner.java
LastAccessDateReducer.java
package com.hadoop.training.examples.partition;
import org.apache.hadoop.conf.Configurable;
public class LastAccessDatePartitioner extends Partitioner<IntWritable, Text> implements Configurable{
private static final String MIN_LAST_ACCESS_DATE_YEAR = "min.last.access.date.year";
private int minLastAccessDateYear = 0;
private Configuration conf = null;
@Override
public int getPartition(IntWritable key, Text value, int numPartitions) {
return key.get() - this.minLastAccessDateYear;
}
public Configuration getConf() {
return conf;
}
public void setConf(Configuration conf) {
this.conf = conf;
minLastAccessDateYear = conf.getInt(MIN_LAST_ACCESS_DATE_YEAR, 0);
}
/**
* Sets the minimum possible last access date to subtract from each key
* to be partitioned
*
Writable
Smart Insert
1 : 1

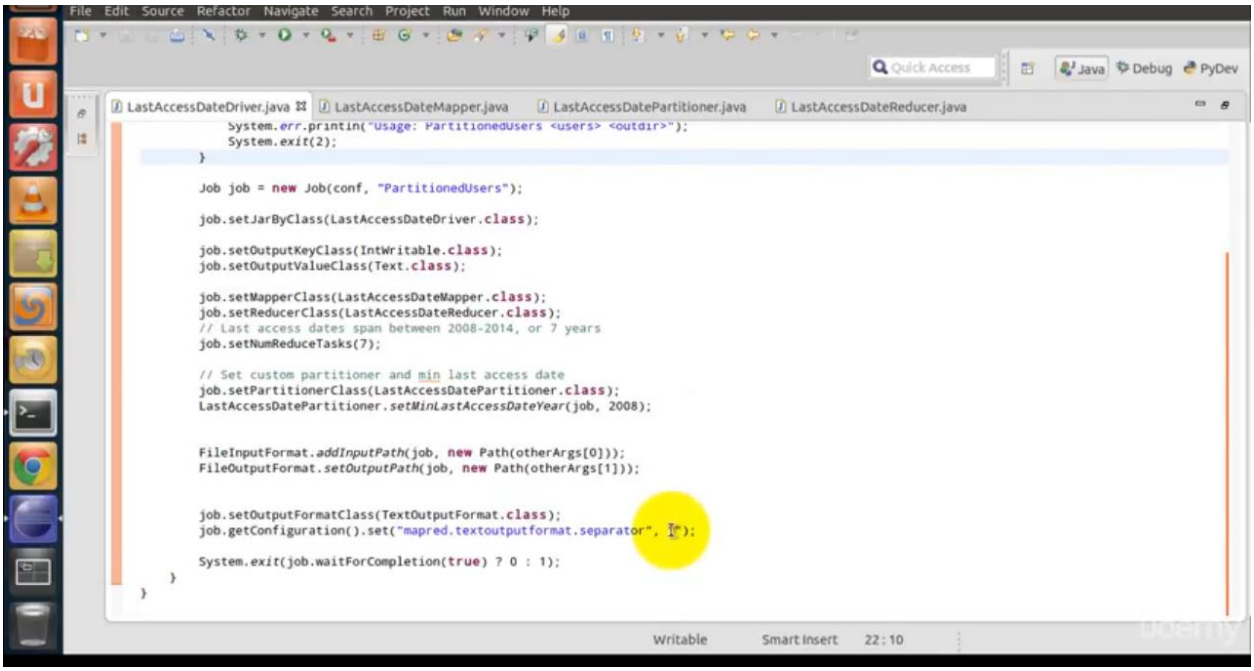

2008 - 2008 = 0
1st Partition
LastAccessDateDriver.java
LastAccessDateMapper.java
LastAccessDatePartitioner.java
LastAccessDateReducer.java
package com.hadoop.training.examples.partition;
import org.apache.hadoop.conf.Configurable;
public class LastAccessDatePartitioner extends Partitioner<IntWritable, Text> implements Configurable{
private static final String MIN_LAST_ACCESS_DATE_YEAR = "min.last.access.date.year";
private int minLastAccessDateYear = 0;
private Configuration conf = null;
@Override
public int getPartition(IntWritable key, Text value, int numPartitions) {
return key.get() - this.minLastAccessDateYear;
}
public Configuration getConf() {
return conf;
}
public void setConf(Configuration conf) {
this.conf = conf;
minLastAccessDateYear = conf.getInt(MIN_LAST_ACCESS_DATE_YEAR, 0);
}
/**
* Sets the minimum possible last access date to subtract from each key
* to be partitioned
*
Writable
Smart Insert
1 : 1


File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
PyDev
LastAccessDateDriver.java
LastAccessDateMapper.java
LastAccessDatePartitioner.java
LastAccessDateReducer.java
package com.hadoop.training.examples.partition;
import org.apache.hadoop.conf.Configurable;
public class LastAccessDatePartitioner extends Partitioner<IntWritable, Text> implements Configurable{
private static final String MIN_LAST_ACCESS_DATE_YEAR = "min.last.access.date.year";
private int minLastAccessDateYear = 0;
private Configuration conf = null;
@Override
public int getPartition(IntWritable key, Text value, int numPartitions) {
return key.get() - this.minLastAccessDateYear;
}
2009 - 2008 = 1
2nd Partition
public Configuration getConf() {
return conf;
}
public void setConf(Configuration conf) {
this.conf = conf;
minLastAccessDateYear = conf.getInt(MIN_LAST_ACCESS_DATE_YEAR, 0);
}
/**
* Sets the minimum possible last access date to subtract from each key
* to be partitioned
*
Writable
Smart Insert
1 : 1


File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
PyDev
LastAccessDateDriver.java
LastAccessDateMapper.java
LastAccessDatePartitioner.java
LastAccessDateReducer.java
package com.hadoop.training.examples.partition;
import java.io.IOException;
public class LastAccessDateMapper extends Mapper<Object, Text, IntWritable, Text>{
// This object will format the date string into a Date object
private final static SimpleDateFormat frmt = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSS");
private IntWritable outkey = new IntWritable();
public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
Map<String, String> parsed = MRDPUtils.transformXmlToMap(value.toString());
String strdate = parsed.get("LastAccessDate");
if(strdate == null) {
return;
}
try {
Date lastAccessDate = frmt.parse(strdate);
Calendar calender = Calendar.getInstance();
calender.setTime(lastAccessDate);
outkey.set(calender.get(Calendar.YEAR));
context.write(outkey, value);
} catch (ParseException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
}
Writable
Smart Insert
32 : 57

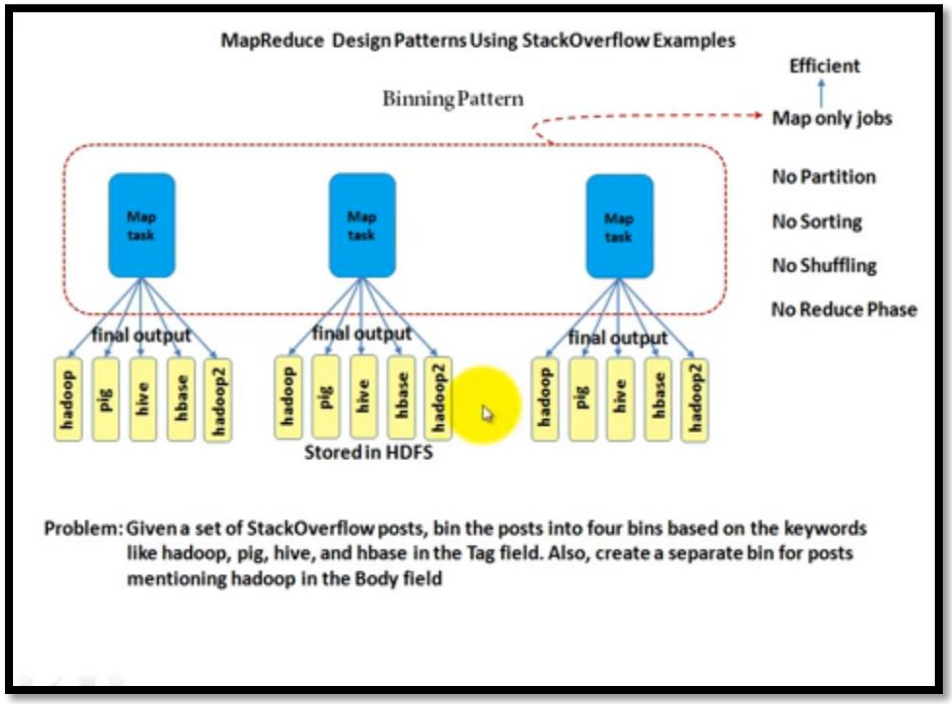

File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
PyDev
LastAccessDateDriver.java
LastAccessDateMapper.java
LastAccessDatePartitioner.java
LastAccessDateReducer.java
package com.hadoop.training.examples.partition;
import java.io.IOException;
public class LastAccessDateReducer extends Reducer<IntWritable, Text, NullWritable, Text> {
protected void reduce(IntWritable key, Iterable<Text> values,
Context context) throws IOException, InterruptedException {
for (Text t : values) {
context.write(NullWritable.get(), t);
}
}
}
Writable
Smart Insert
1 : 1

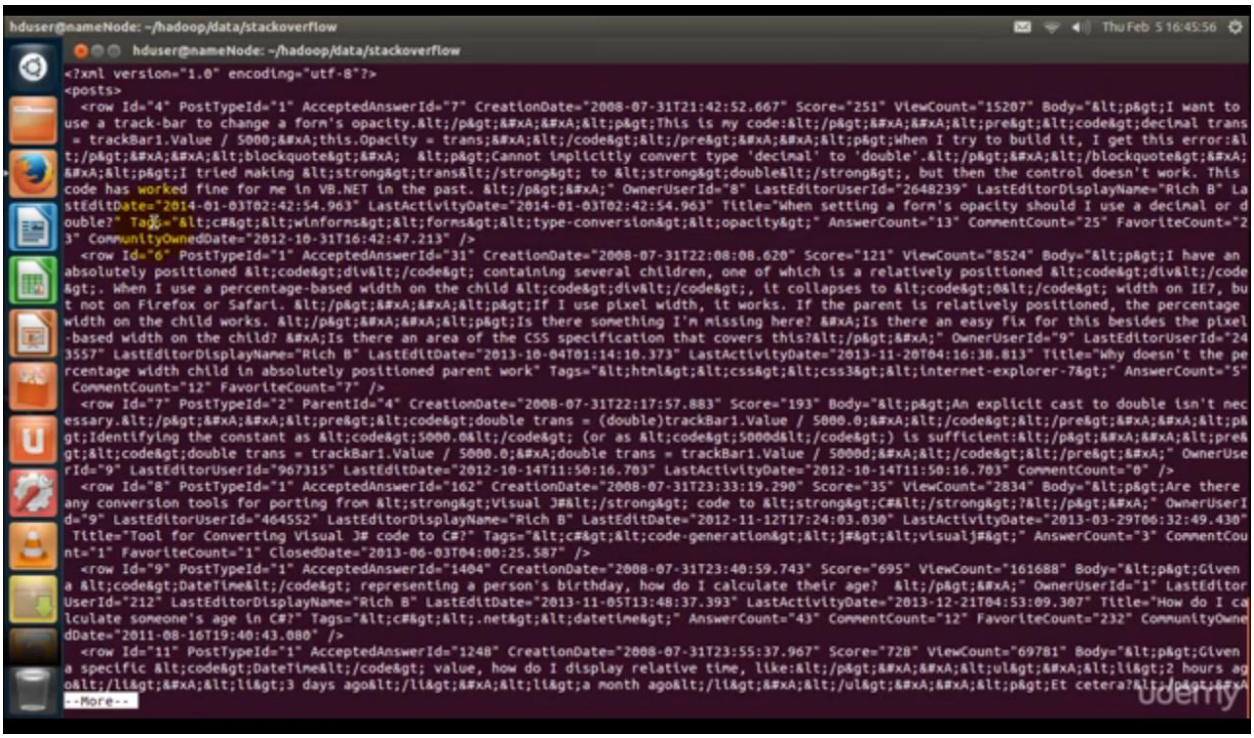

File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
PyDev
LastAccessDateDriver.java
LastAccessDateMapper.java
LastAccessDatePartitioner.java
LastAccessDateReducer.java
package com.hadoop.training.examples.partition;
import java.io.IOException;
public class LastAccessDateReducer extends Reducer<IntWritable, Text, NullWritable, Text> {
protected void reduce(IntWritable key, Iterable<Text> values,
Context context) throws IOException, InterruptedException {
for (Text t : values) {
context.write(NullWritable.get(), t);
}
}
}
Default OutputFormat is
TextOutputFormat
Tab space
<key> <value>
<key> <value>
<key> <value>
Writable
Smart Insert
1 : 1


File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
PyDev
LastAccessDateDriver.java
LastAccessDateMapper.java
LastAccessDatePartitioner.java
LastAccessDateReducer.java
System.err.println("Usage: PartitionedUsers <users> <outdir>");
System.exit(2);
}
Job job = new Job(conf, "PartitionedUsers");
job.setJarByClass(LastAccessDateDriver.class);
job.setOutputKeyClass(IntWritable.class);
job.setOutputValueClass(Text.class);
job.setMapperClass(LastAccessDateMapper.class);
job.setReducerClass(LastAccessDateReducer.class);
// Last access dates span between 2008-2014, or 7 years
job.setNumReduceTasks(7);
// Set custom partitioner and min last access date
job.setPartitionerClass(LastAccessDatePartitioner.class);
LastAccessDatePartitioner.setMinLastAccessDateYear(job, 2008);
FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
job.setOutputFormatClass(TextOutputFormat.class);
job.getConfiguration().set("mapred.textoutputformat.separator", ",");
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
Writable
Smart Insert
22 : 10

Binning Pattern

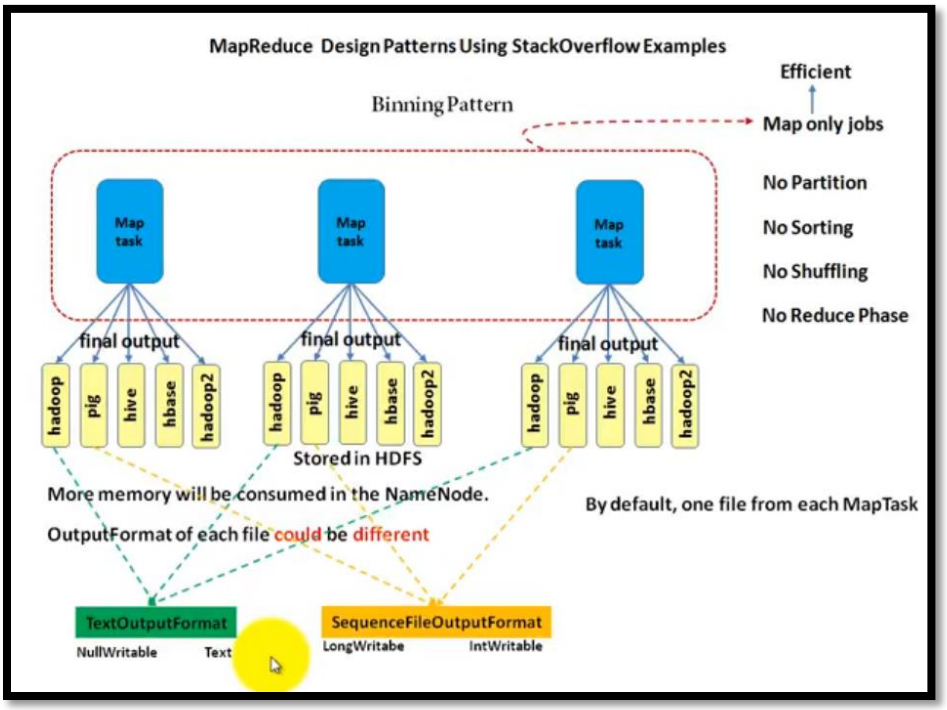
- Binning is similar to partition pattern
- group your data into different categories
- data is group together in map phase instead of doing in the partition phase
- map output itself is the final output
- No Reduce phase
- Called map only jobs
- Map output not be partitioned ,sorted and shuffled
- More efficient
- Final output of map stored on the hdfs instead of local disk
- Not good for namenode scalability as more file will be created



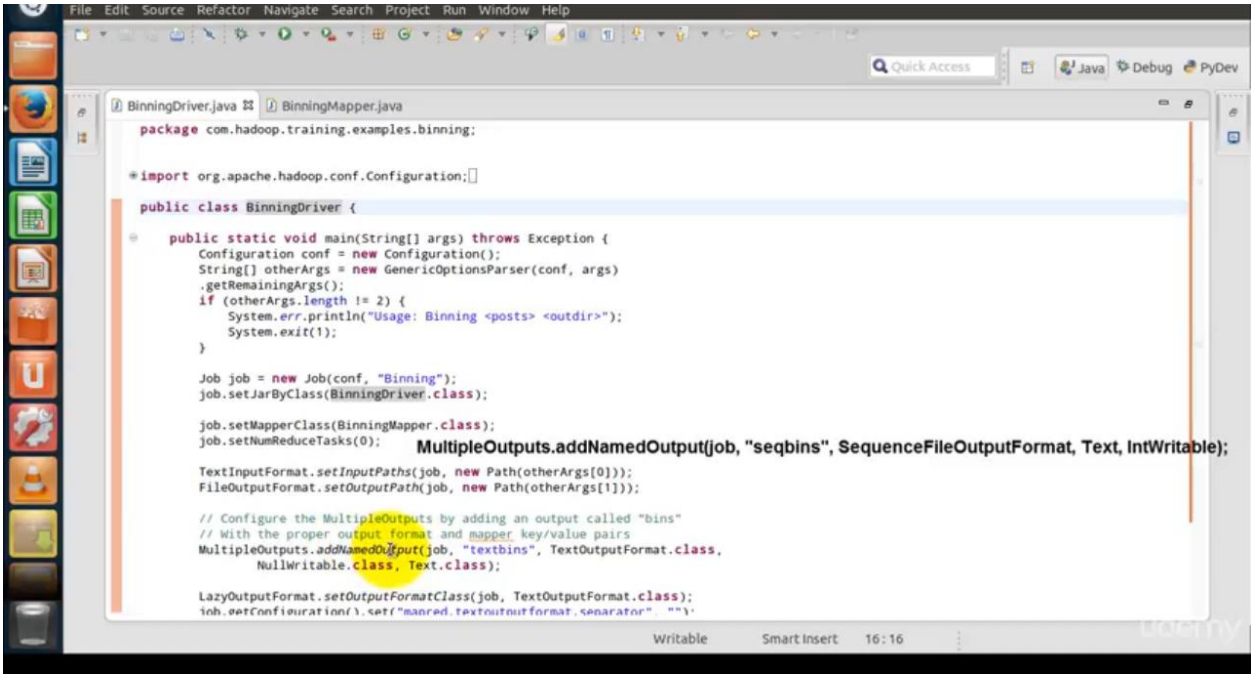


Search keywords in body and tag fields

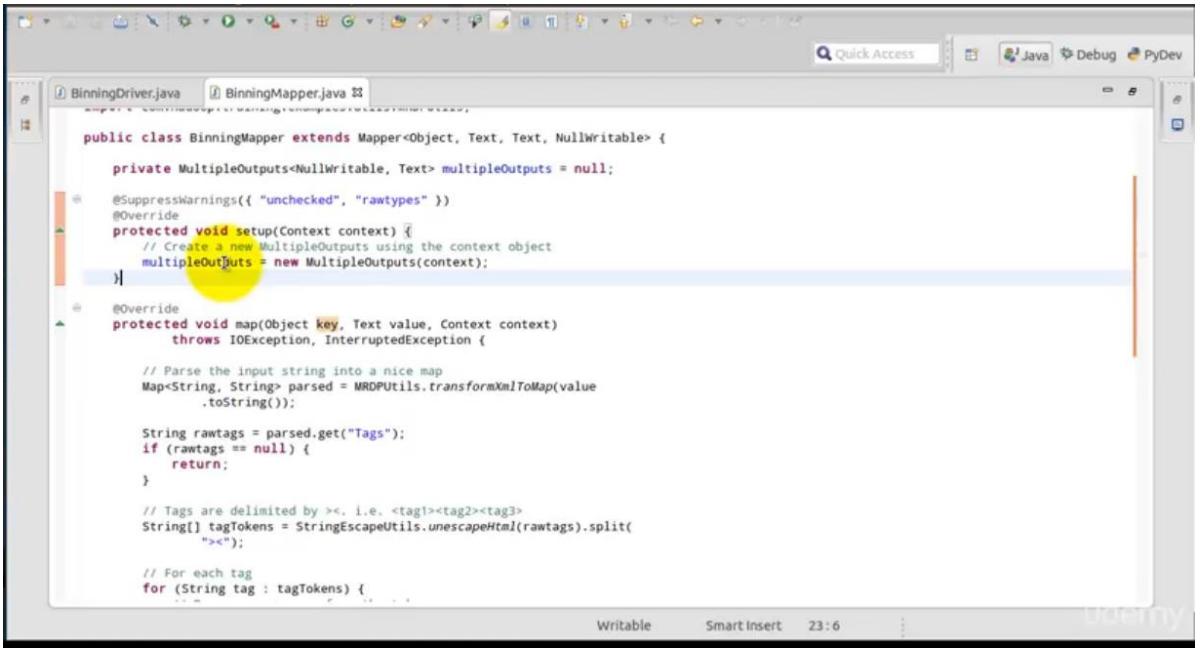
In this example we have to create five output files in each map tasks as per keywords



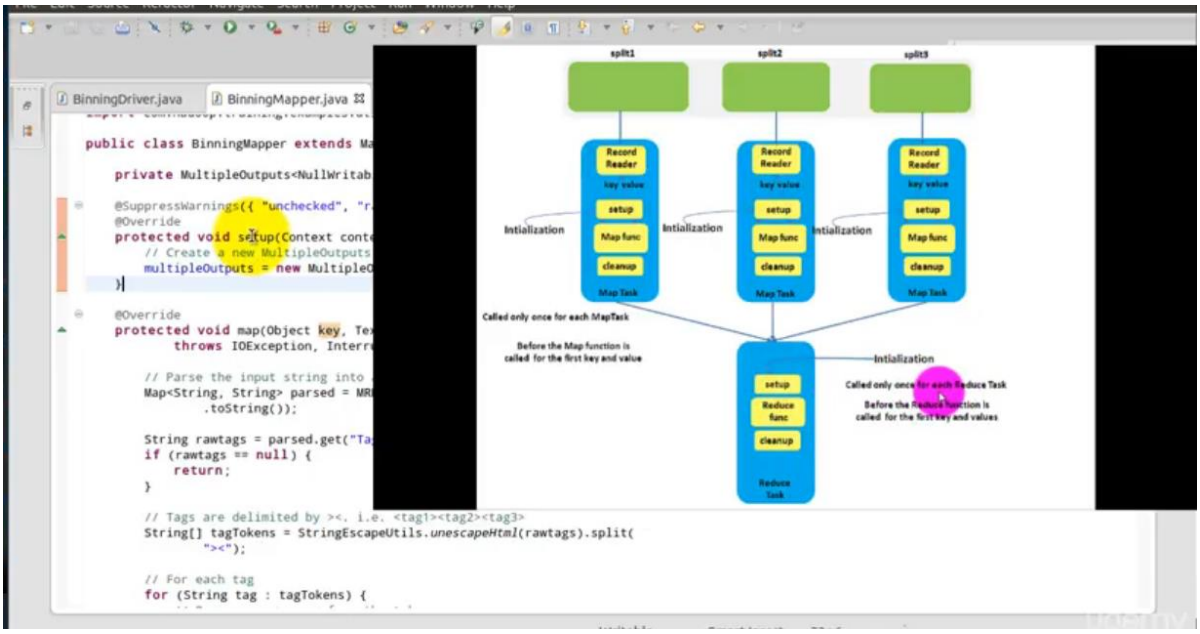
We can have different output format and different key value types for each file

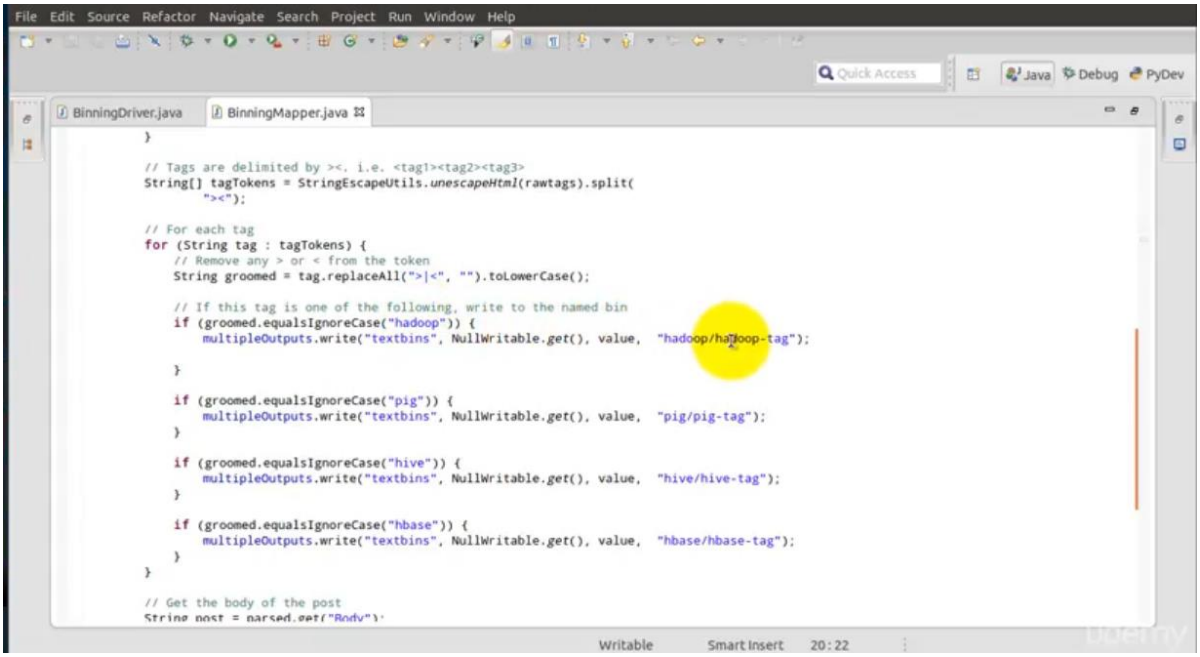


Number of reduce task must be set to 0

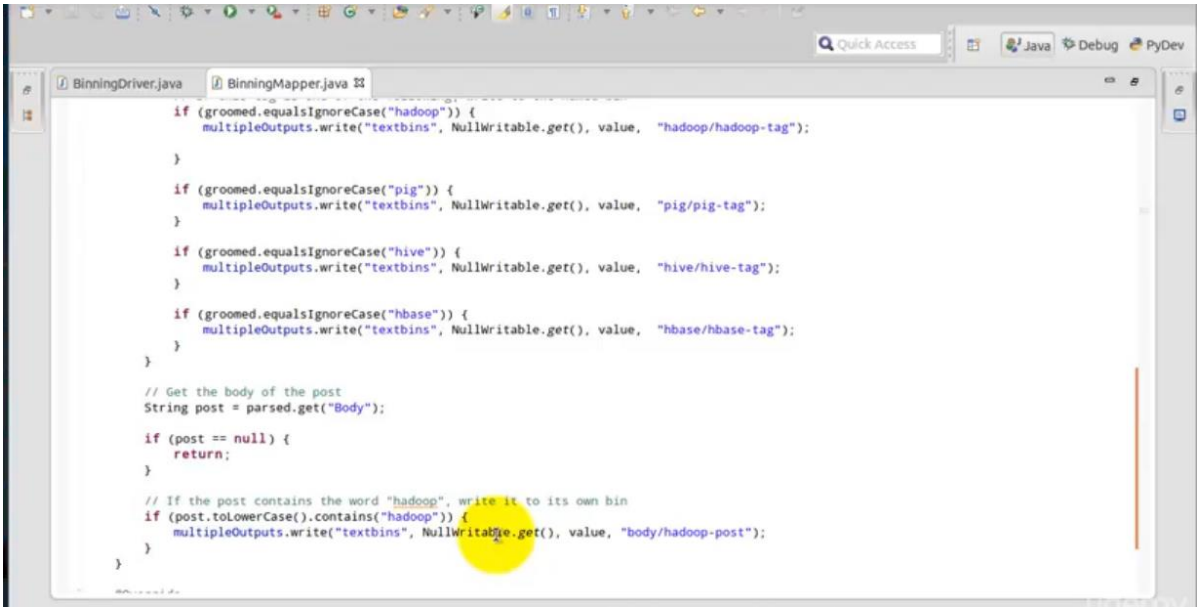


Setup function is function of mapper class which is used to do initialization before the map function is called, this function is called only once for each map task

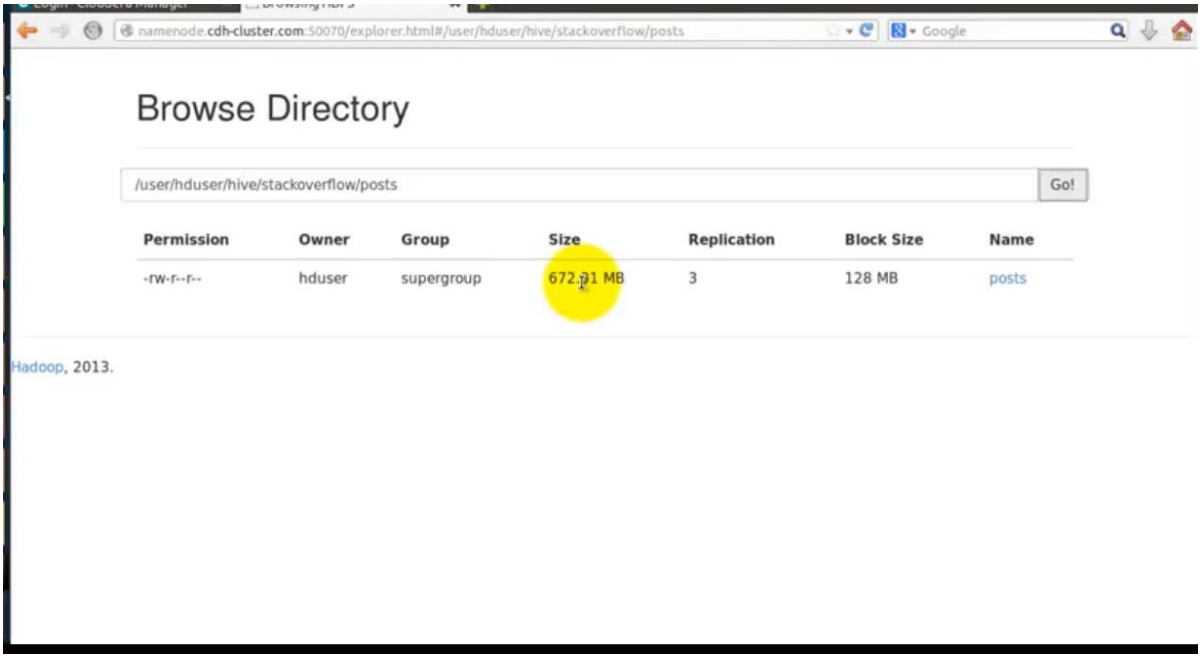




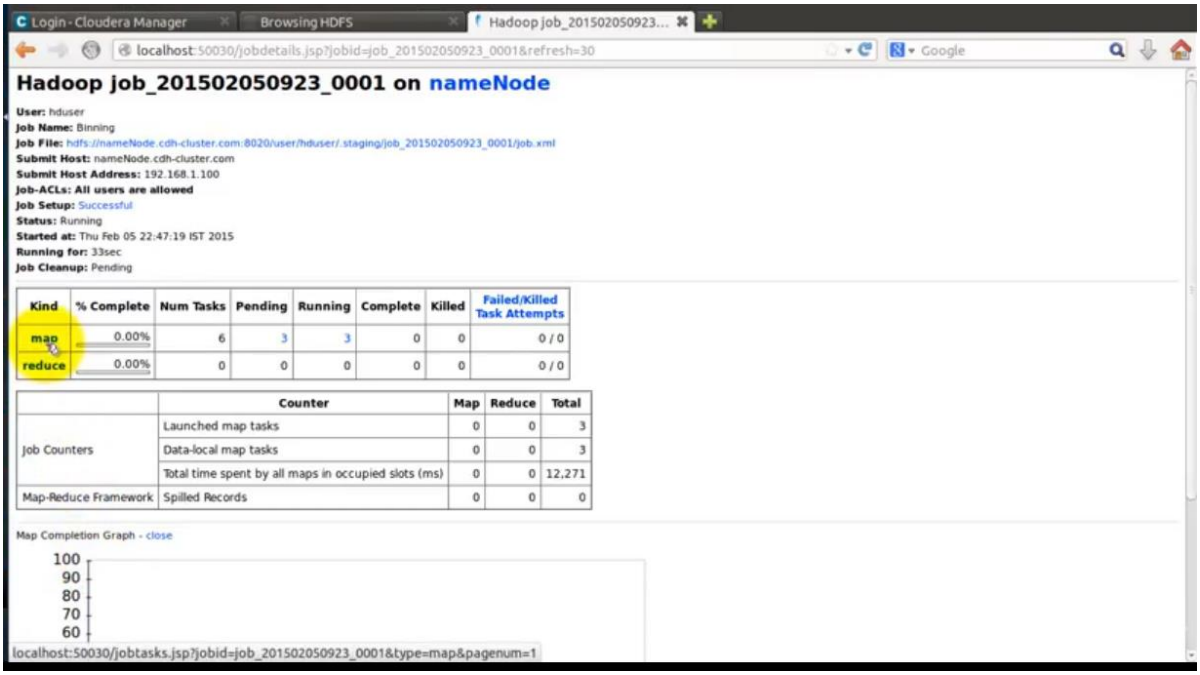
Hadoop file are created under Hadoop directory and pig file under pig folder and so on


Quick Access
Java
Debug
PyDev
BinningDriver.java
BinningMapper.java
if (groomed.equalsIgnoreCase("hadoop")) {
multipleOutputs.write("textbins", NullWritable.get(), value, "hadoop/hadoop-tag");
}
if (groomed.equalsIgnoreCase("pig")) {
multipleOutputs.write("textbins", NullWritable.get(), value, "pig/pig-tag");
}
if (groomed.equalsIgnoreCase("hive")) {
multipleOutputs.write("textbins", NullWritable.get(), value, "hive/hive-tag");
}
if (groomed.equalsIgnoreCase("hbase")) {
multipleOutputs.write("textbins", NullWritable.get(), value, "hbase/hbase-tag");
}
}
// Get the body of the post
String post = parsed.get("Body");
if (post == null) {
return;
}
// If the post contains the word "hadoop", write it to its own bin
if (post.toLowerCase().contains("hadoop")) {
multipleOutputs.write("textbins", NullWritable.get(), value, "body/hadoop-post");
}
}

Input file

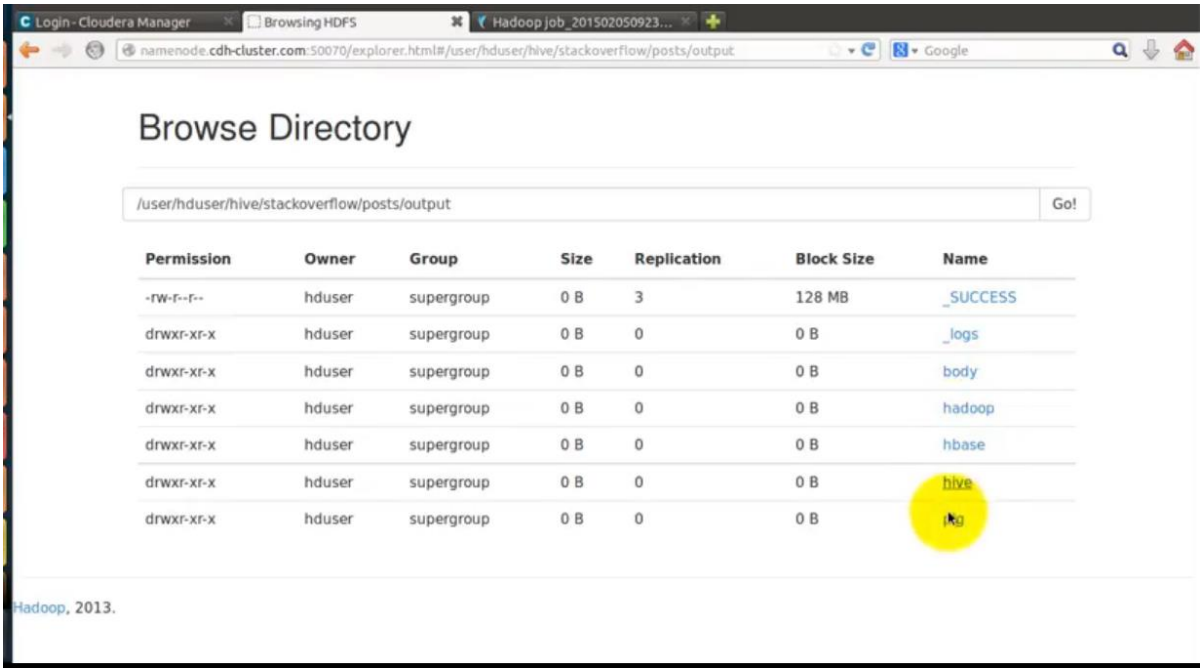


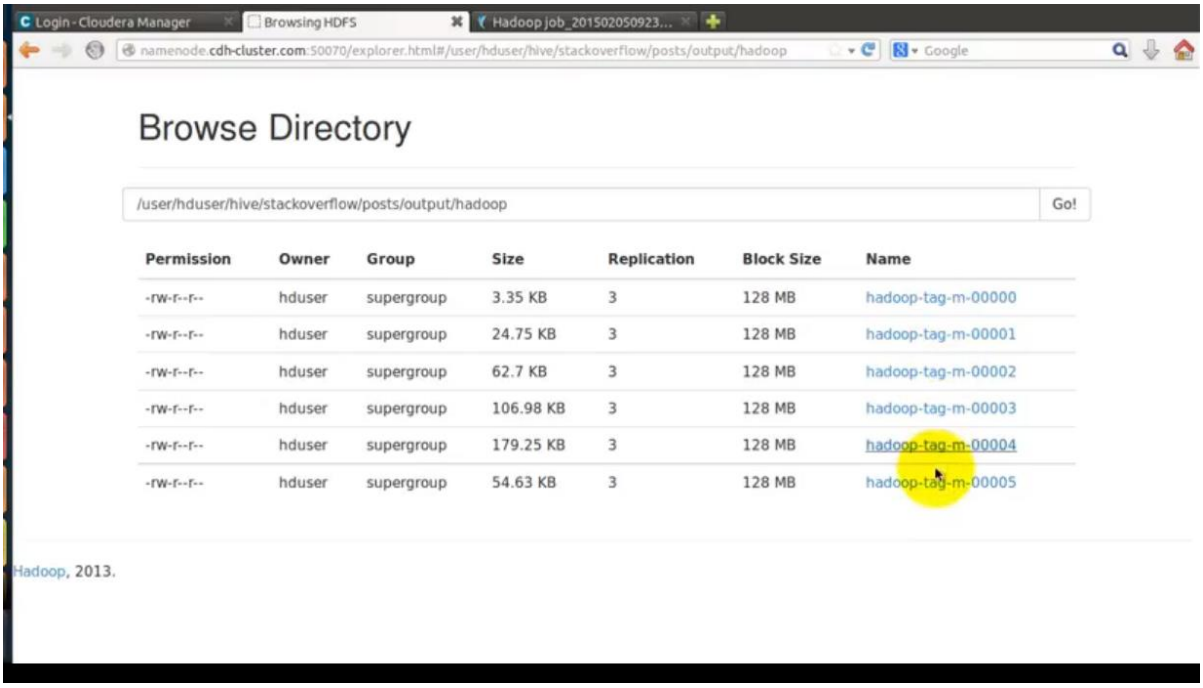
6 map task will be created as size is 672 MB and block size is 128 MB

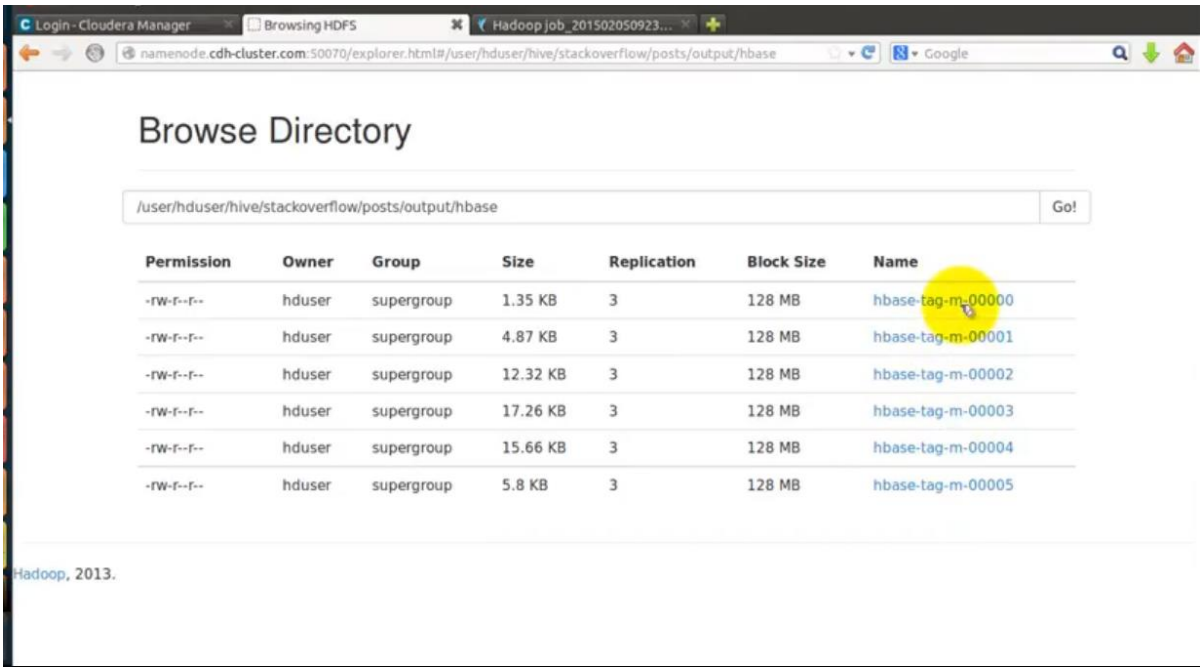


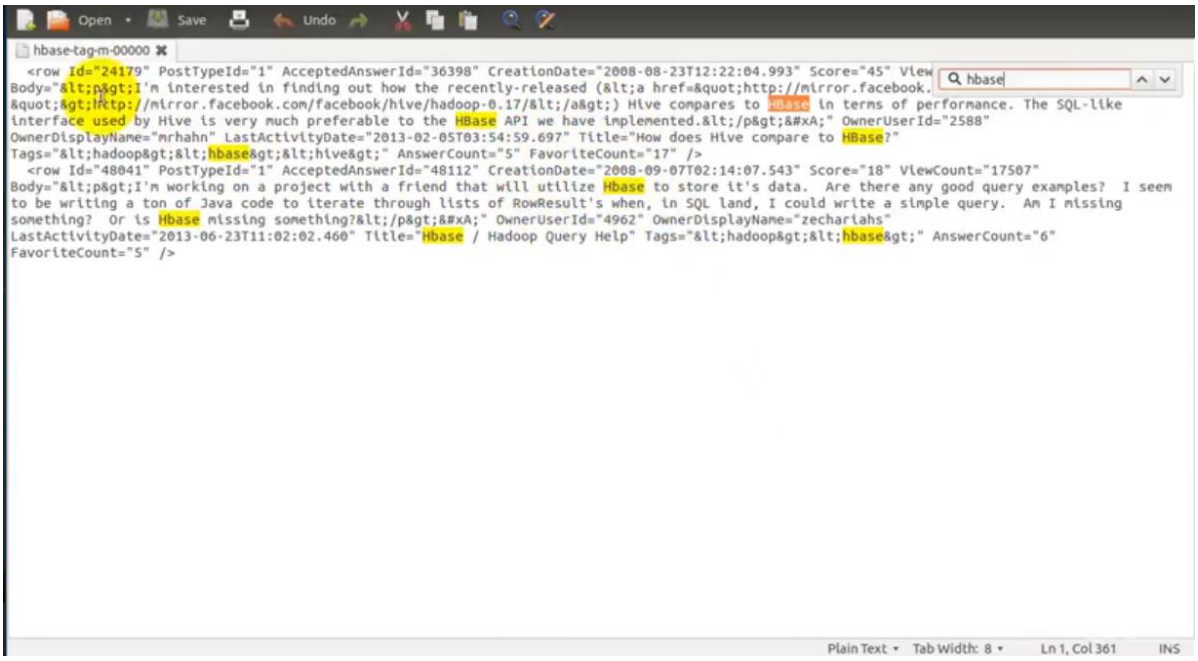
We can see 6 map tasks and 0 reduce task are created

Output file

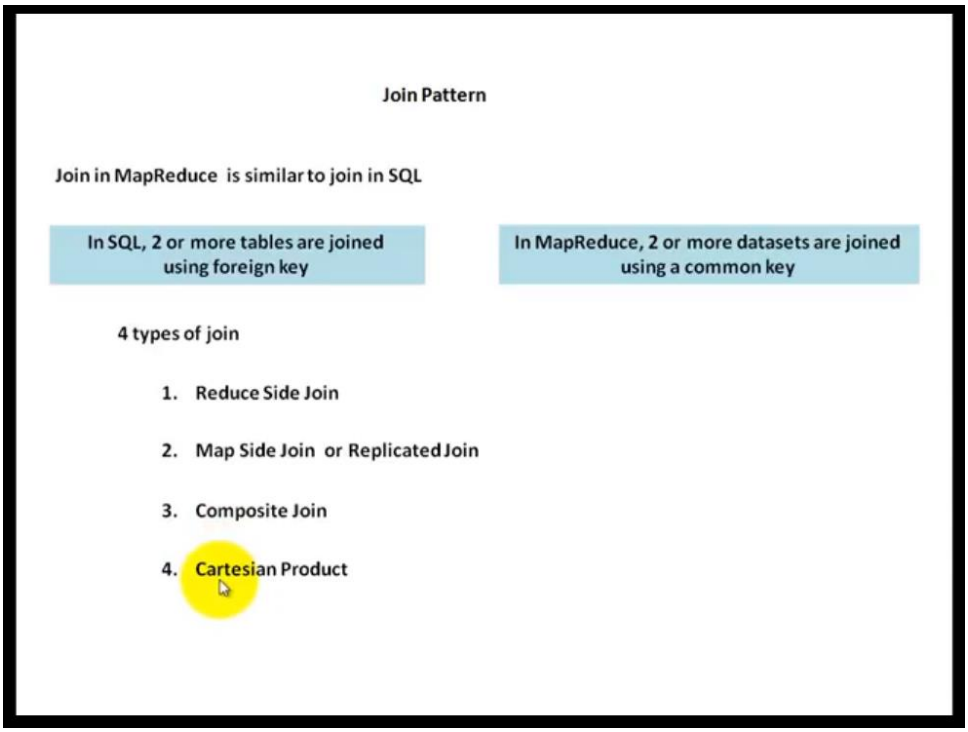

Login - Cloudera Manager
Browsing HDFS
Hadoop job_201502050923...
namenode.cdh-cluster.com:50070/explorer.html#/user/hduser/hive/stackoverflow/posts/output
Google
Browse Directory
/user/hduser/hive/stackoverflow/posts/output
Go!
Permission Owner Group Size Replication Block Size Name
-rw-r--r-- hduser supergroup 0 B 3 128 MB _SUCCESS
drwxr-xr-x hduser supergroup 0 B 0 0 B _logs
drwxr-xr-x hduser supergroup 0 B 0 0 B body
drwxr-xr-x hduser supergroup 0 B 0 0 B hadoop
drwxr-xr-x hduser supergroup 0 B 0 0 B hbase
drwxr-xr-x hduser supergroup 0 B 0 0 B hive
drwxr-xr-x hduser supergroup 0 B 0 0 B pig
Hadoop, 2013.

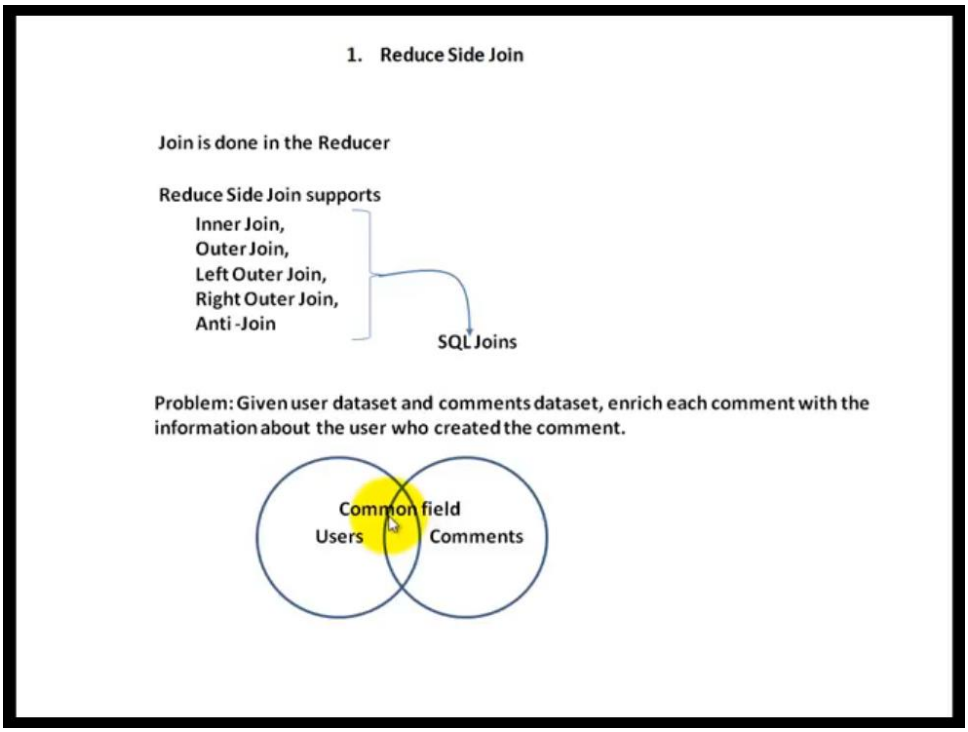

Login - Cloudera Manager
Browsing HDFS
Hadoop job_201502050923...
namenode.cdh-cluster.com:50070/explorer.html#/user/hduser/hive/stackoverflow/posts/output/hadoop
Google
Browse Directory
/user/hduser/hive/stackoverflow/posts/output/hadoop
Go!
Permission Owner Group Size Replication Block Size Name
-rw-r--r-- hduser supergroup 3.35 KB 3 128 MB hadoop-tag-m-00000
-rw-r--r-- hduser supergroup 24.75 KB 3 128 MB hadoop-tag-m-00001
-rw-r--r-- hduser supergroup 62.7 KB 3 128 MB hadoop-tag-m-00002
-rw-r--r-- hduser supergroup 106.98 KB 3 128 MB hadoop-tag-m-00003
-rw-r--r-- hduser supergroup 179.25 KB 3 128 MB hadoop-tag-m-00004
-rw-r--r-- hduser supergroup 54.63 KB 3 128 MB hadoop-tag-m-00005
Hadoop, 2013.

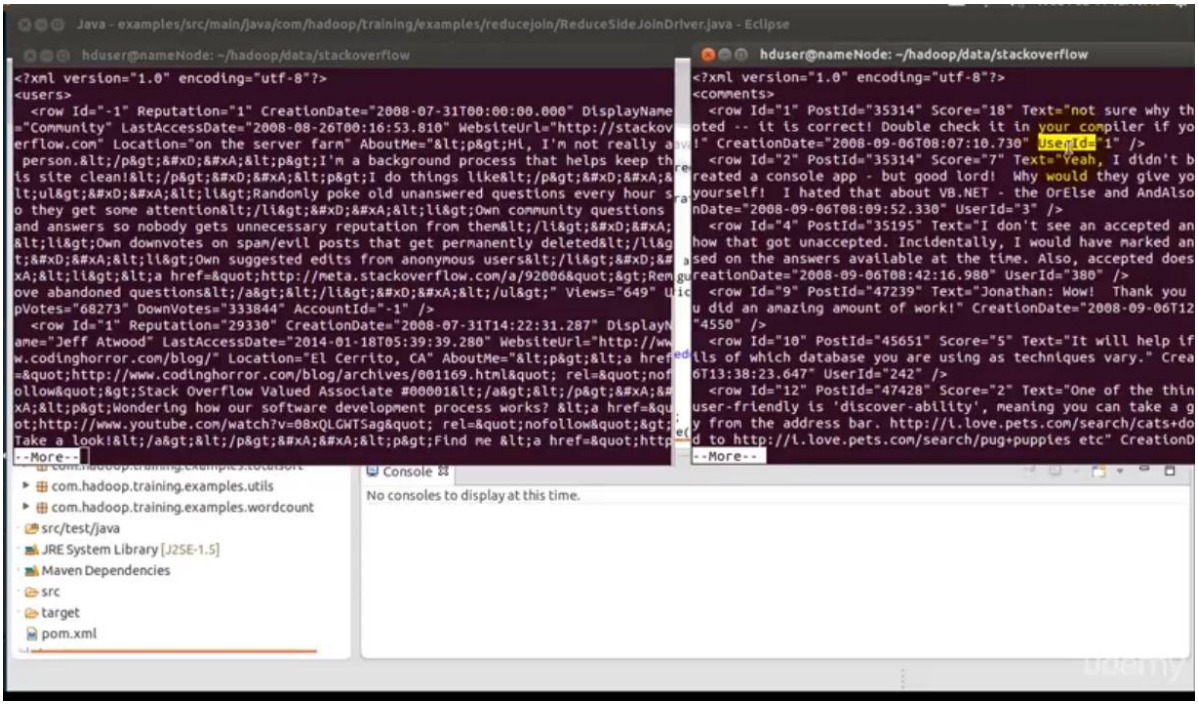

Login - Cloudera Manager
Browsing HDFS
Hadoop job_201502050923...
namenode.cdh-cluster.com:50070/explorer.html#/user/hduser/hive/stackoverflow/posts/output/hbase
Google
Browse Directory
/user/hduser/hive/stackoverflow/posts/output/hbase
Go!
Permission Owner Group Size Replication Block Size Name
-rw-r--r-- hduser supergroup 1.35 KB 3 128 MB hbase-tag-m-00000
-rw-r--r-- hduser supergroup 4.87 KB 3 128 MB hbase-tag-m-00001
-rw-r--r-- hduser supergroup 12.32 KB 3 128 MB hbase-tag-m-00002
-rw-r--r-- hduser supergroup 17.26 KB 3 128 MB hbase-tag-m-00003
-rw-r--r-- hduser supergroup 15.66 KB 3 128 MB hbase-tag-m-00004
-rw-r--r-- hduser supergroup 5.8 KB 3 128 MB hbase-tag-m-00005
Hadoop, 2013.


Open
Save
Undo
hbase-tag-m-00000
hbase
<row Id="24179" PostTypeId="1" AcceptedAnswerId="36398" CreationDate="2008-08-23T12:22:04.993" Score="45" ViewCount="17507" Body="<p>I'm interested in finding out how the recently-released (http://mirror.facebook.com/facebook/hive/hadoop-0.17/) Hive compares to HBase in terms of performance. The SQL-like interface used by Hive is very much preferable to the HBase API we have implemented.</p>
" OwnerUserId="2588" OwnerDisplayName="mrhahn" LastActivityDate="2013-02-05T03:54:59.697" Title="How does Hive compare to HBase?" Tags="<hadoop><hbase><hive>" AnswerCount="5" FavoriteCount="17" />
<row Id="48041" PostTypeId="1" AcceptedAnswerId="48112" CreationDate="2008-09-07T02:14:07.543" Score="18" ViewCount="17507" Body="<p>I'm working on a project with a friend that will utilize Hbase to store it's data. Are there any good query examples? I seem to be writing a ton of Java code to iterate through lists of RowResult's when, in SQL land, I could write a simple query. Am I missing something? Or is Hbase missing something?</p>
" OwnerUserId="4962" OwnerDisplayName="zechariahs" LastActivityDate="2013-06-23T11:02:02.460" Title="Hbase / Hadoop Query Help" Tags="<hadoop><hbase>" AnswerCount="6" FavoriteCount="5" />
Plain Text
Tab Width: 8
Ln 1, Col 361
INS

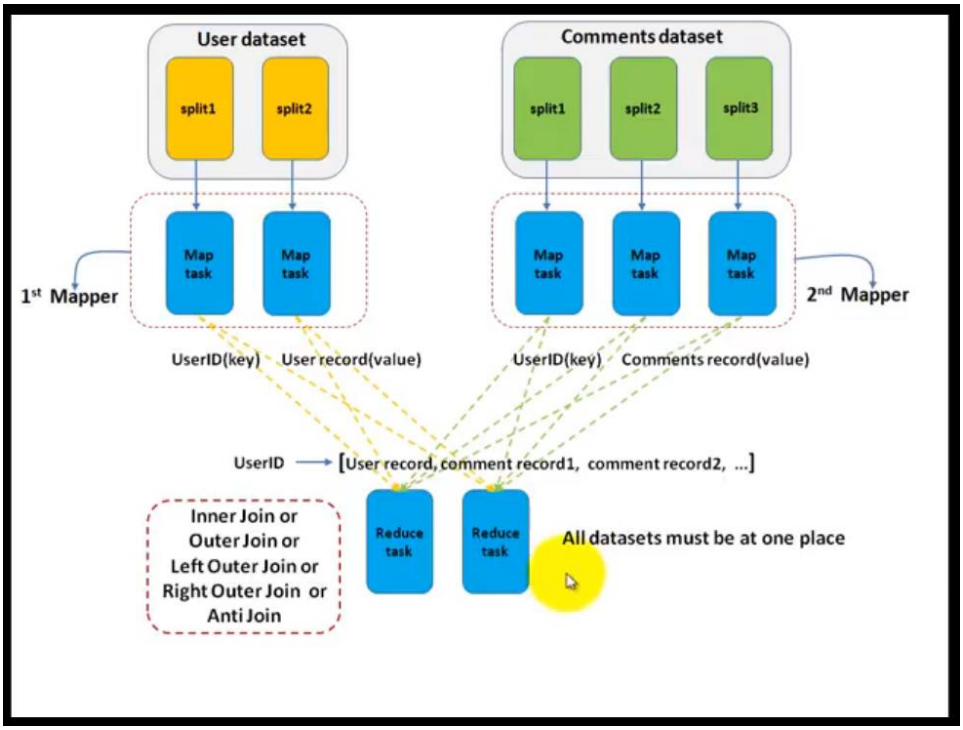
Reduce Side Join Pattern

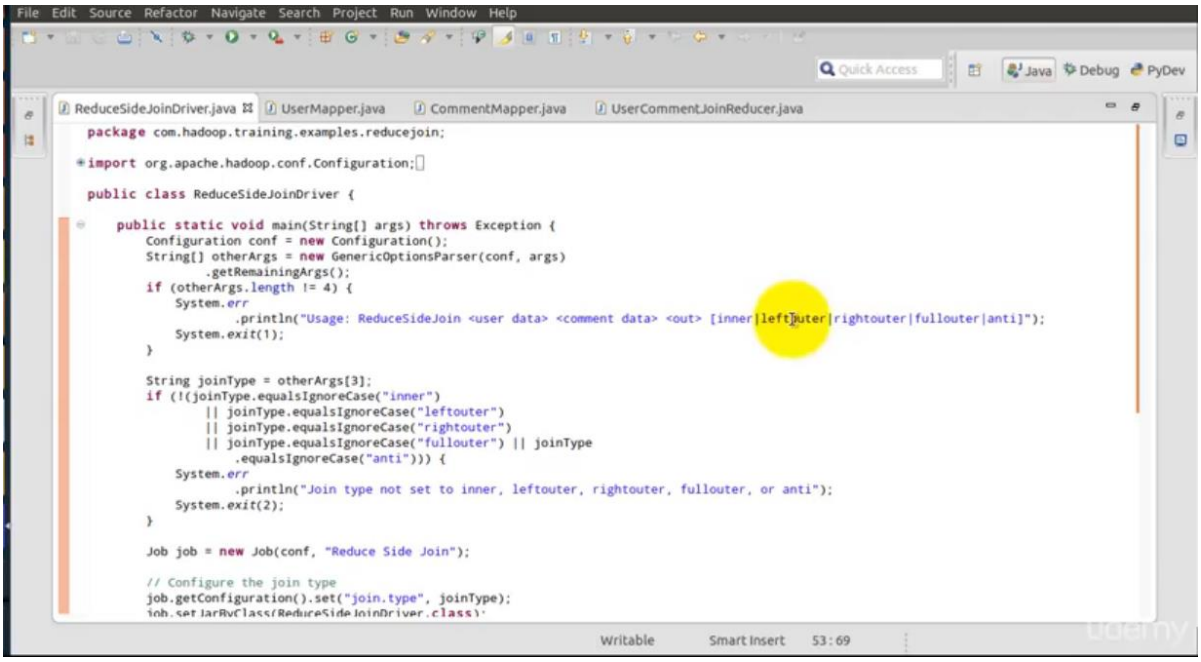


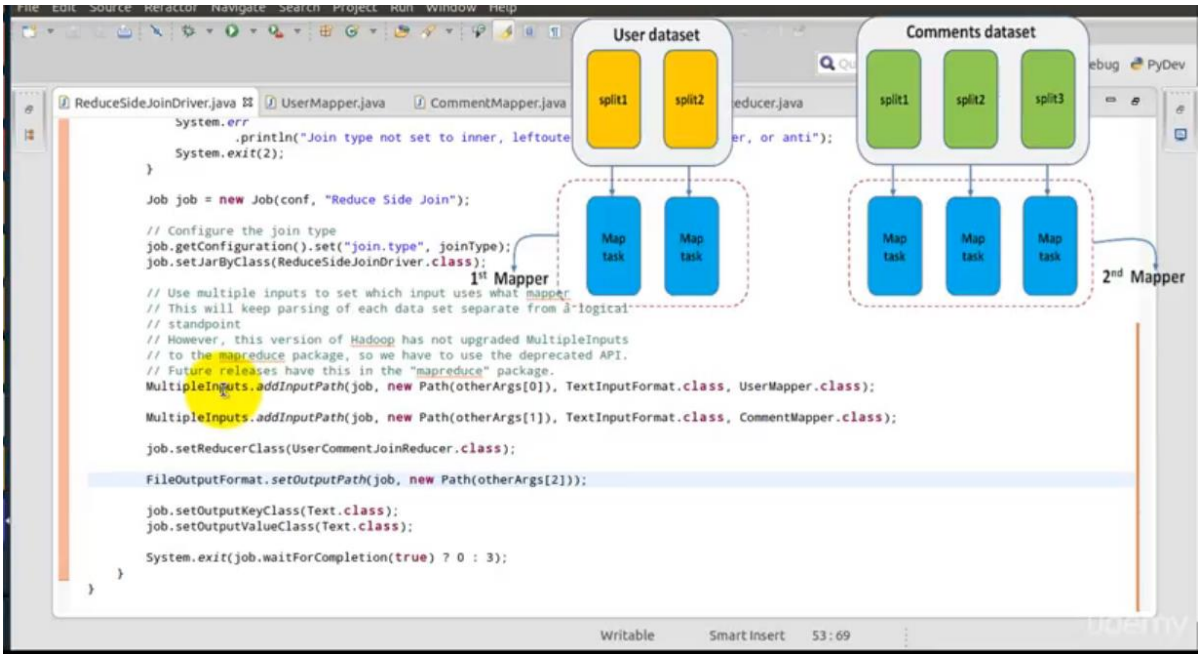


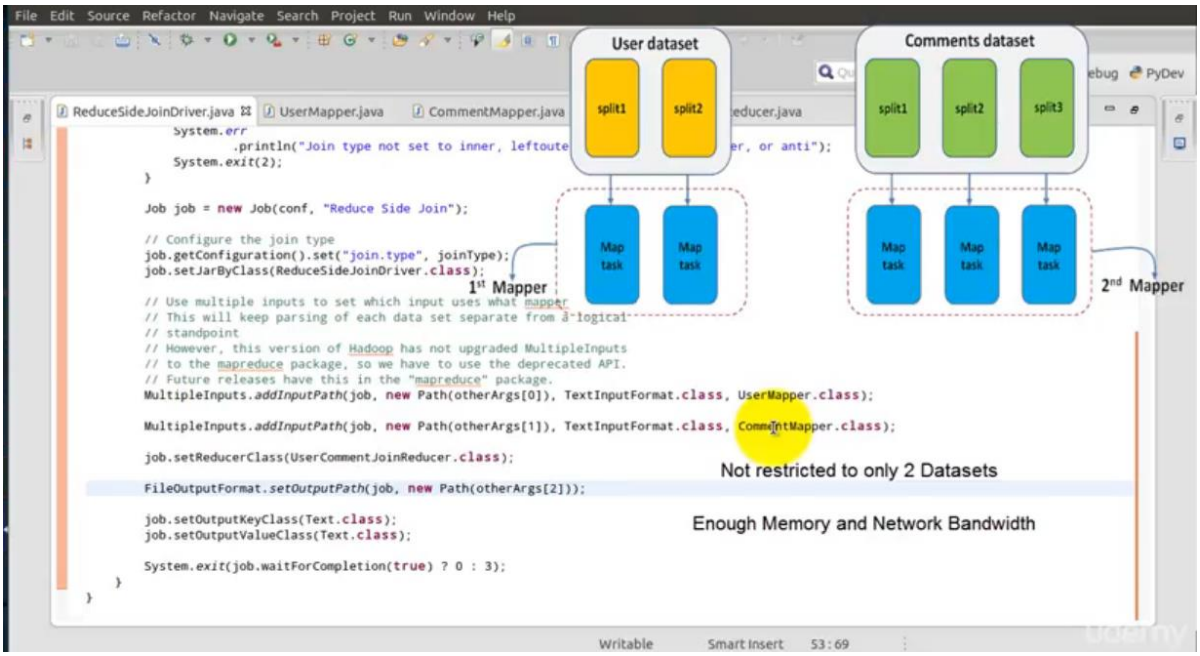


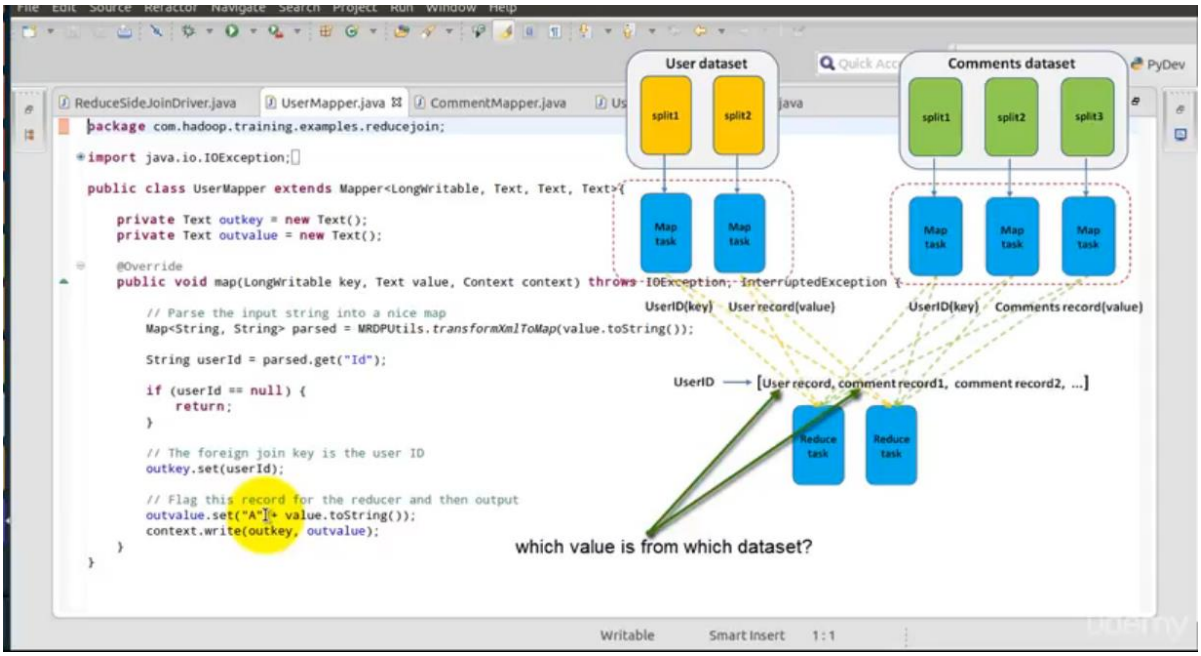
Users data set has id and comment dataset has userid both represent same

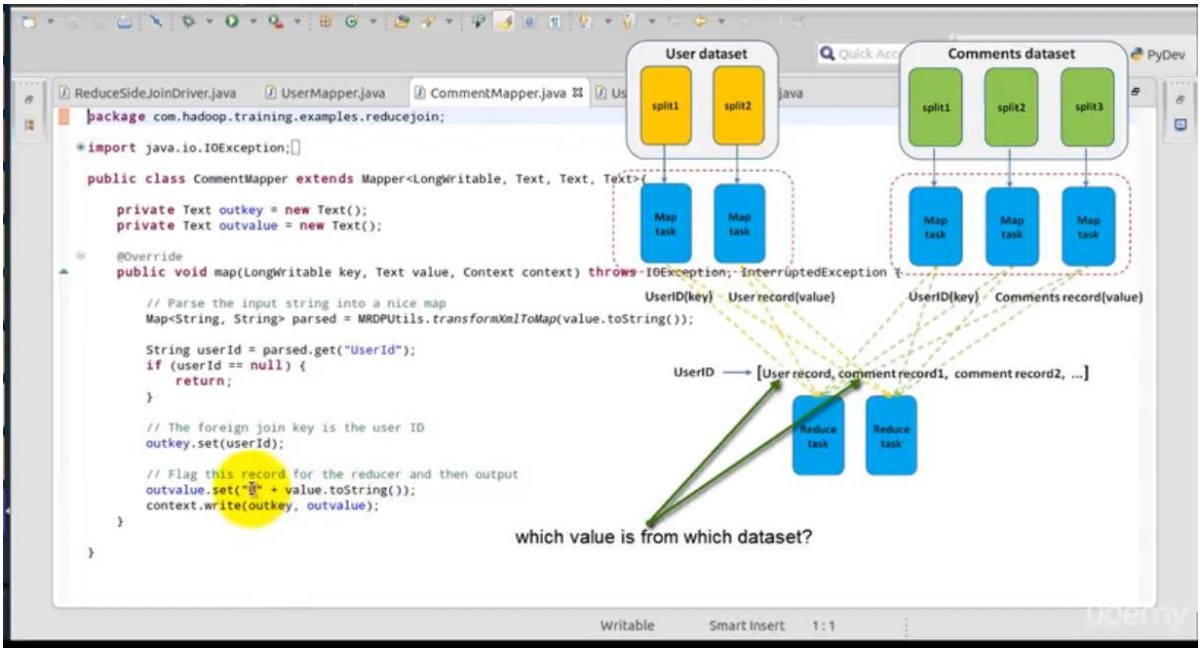


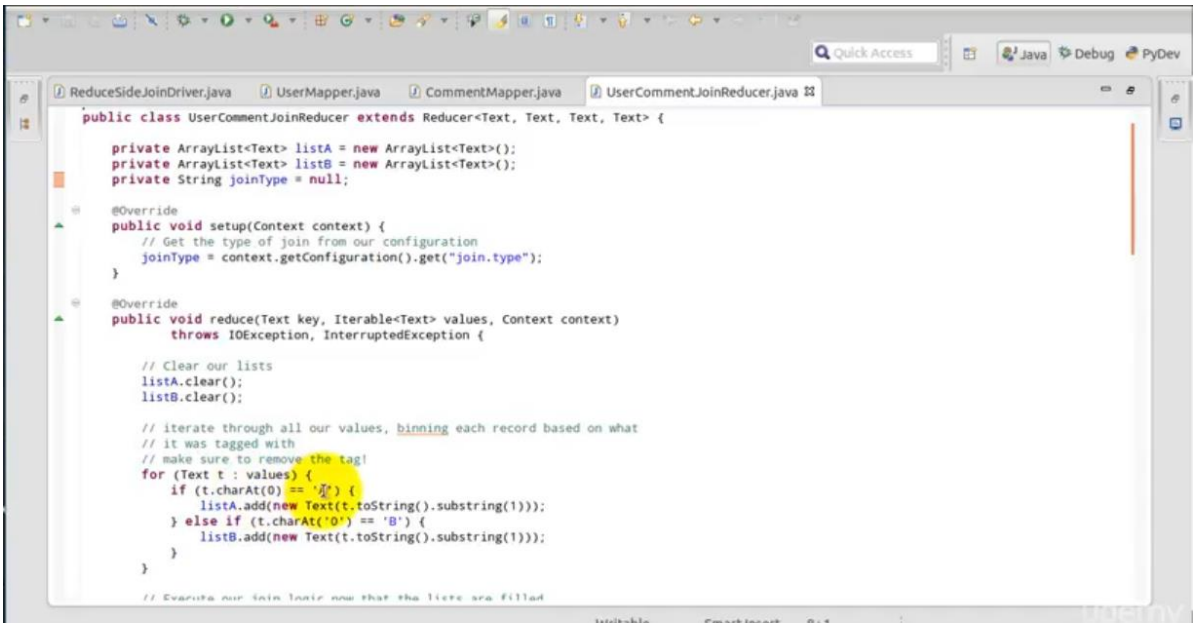

File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
PyDev
ReduceSideJoinDriver.java
UserMapper.java
CommentMapper.java
UserCommentJoinReducer.java
package com.hadoop.training.examples.reducejoin;
import org.apache.hadoop.conf.Configuration;
public class ReduceSideJoinDriver {
public static void main(String[] args) throws Exception {
Configuration conf = new Configuration();
String[] otherArgs = new GenericOptionsParser(conf, args)
.getRemainingArgs();
if (otherArgs.length != 4) {
System.err
.println("Usage: ReduceSideJoin <user data> <comment data> <out> [inner|leftouter|rightouter|fullouter|anti]");
System.exit(1);
}
String joinType = otherArgs[3];
if (!(joinType.equalsIgnoreCase("inner")
|| joinType.equalsIgnoreCase("leftouter")
|| joinType.equalsIgnoreCase("rightouter")
|| joinType.equalsIgnoreCase("fullouter") || joinType
.equalsIgnoreCase("anti"))) {
System.err
.println("Join type not set to inner, leftouter, rightouter, fullouter, or anti");
System.exit(2);
}
Job job = new Job(conf, "Reduce Side Join");
// Configure the join type
job.getConfiguration().set("join.type", joinType);
Writable
Smart Insert
53 : 69

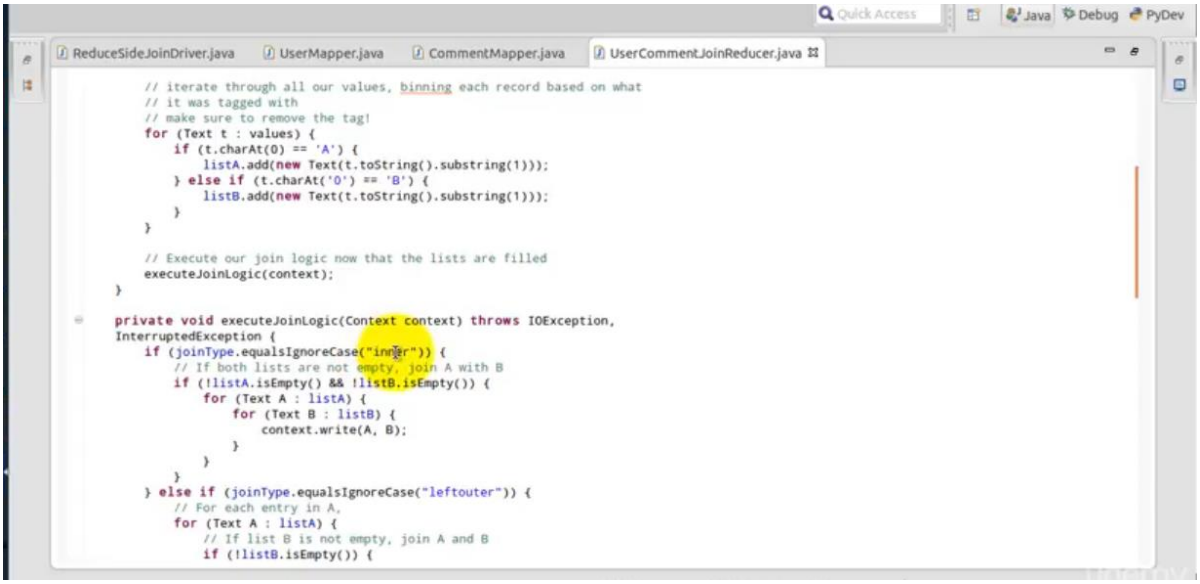

User dataset
split1
split2
Comments dataset
split1
split2
split3
Map task
Map task
Map task
Map task
Map task
1st Mapper
2nd Mapper
ReduceSideJoinDriver.java
UserMapper.java
CommentMapper.java
System.exit(2);
}
Job job = new Job(conf, "Reduce Side Join");
// Configure the join type
job.getConfiguration().set("join.type", joinType);
job.setJarByClass(ReduceSideJoinDriver.class);
// Use multiple inputs to set which input uses what mapper
// This will keep parsing of each data set separate from a 'logical'
// standpoint
// However, this version of Hadoop has not upgraded MultipleInputs
// to the mapreduce package, so we have to use the deprecated API.
// Future releases have this in the "mapreduce" package.
MultipleInputs.addInputPath(job, new Path(otherArgs[0]), TextInputFormat.class, UserMapper.class);
MultipleInputs.addInputPath(job, new Path(otherArgs[1]), TextInputFormat.class, CommentMapper.class);
job.setReducerClass(UserCommentJoinReducer.class);
FileOutputFormat.setOutputPath(job, new Path(otherArgs[2]));
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(Text.class);
System.exit(job.waitForCompletion(true) ? 0 : 3);
}
}
Writable
Smart Insert
53 : 69

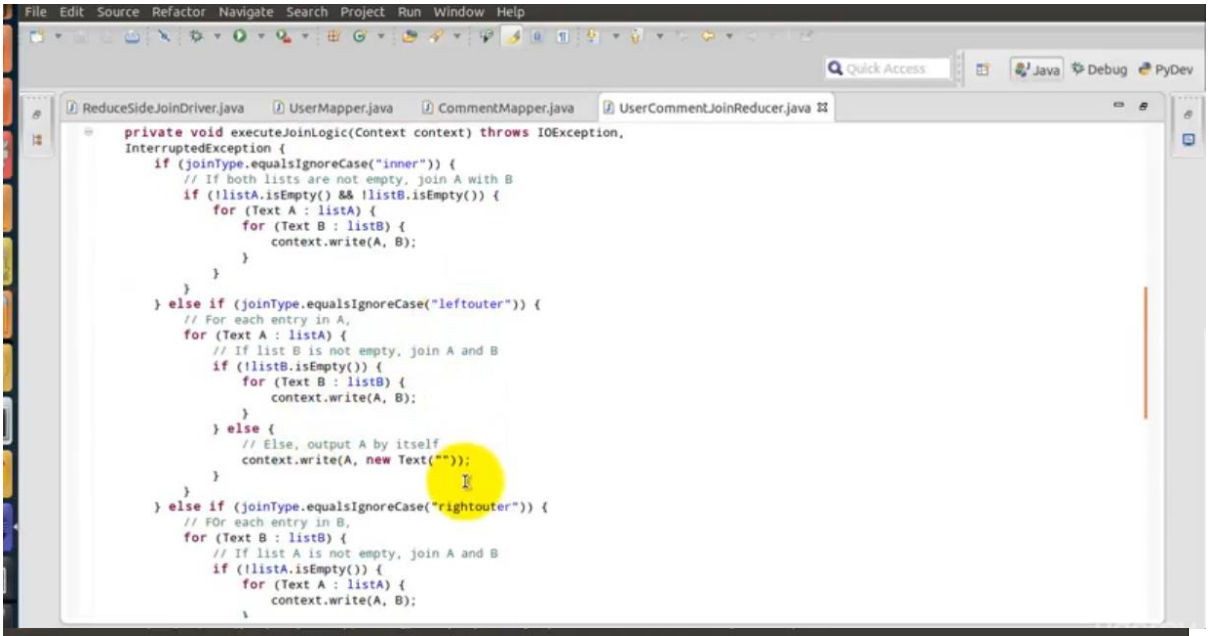

File Edit Source Refactor Navigate Search Project Run Window Help
ReduceSideJoinDriver.java
UserMapper.java
CommentMapper.java
System.err
.println("Join type not set to inner, leftoute
er, or anti");
System.exit(2);
}
Job job = new Job(conf, "Reduce Side Join");
// Configure the join type
job.getConfiguration().set("join.type", joinType);
job.setJarByClass(ReduceSideJoinDriver.class);
// Use multiple inputs to set which input uses what mapper
// This will keep parsing of each data set separate from a logical
// standpoint
// However, this version of Hadoop has not upgraded MultipleInputs
// to the mapreduce package, so we have to use the deprecated API.
// Future releases have this in the "mapreduce" package.
MultipleInputs.addInputPath(job, new Path(otherArgs[0]), TextInputFormat.class, UserMapper.class);
MultipleInputs.addInputPath(job, new Path(otherArgs[1]), TextInputFormat.class, CommentMapper.class);
job.setReducerClass(UserCommentJoinReducer.class);
FileOutputFormat.setOutputPath(job, new Path(otherArgs[2]));
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(Text.class);
System.exit(job.waitForCompletion(true) ? 0 : 3);
}
}
User dataset
split1
split2
Comments dataset
split1
split2
split3
Map task
Map task
Map task
Map task
Map task
1st Mapper
2nd Mapper
Not restricted to only 2 Datasets
Enough Memory and Network Bandwidth
Writable
Smart Insert
53 : 69

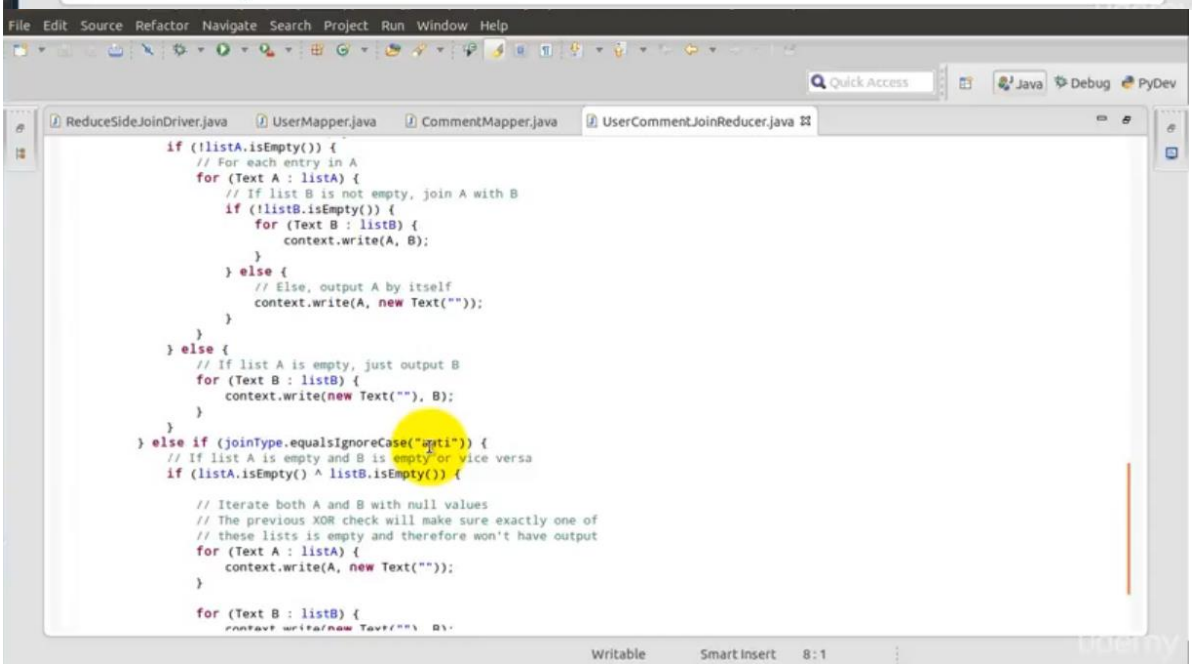

ReduceSideJoinDriver.java
UserMapper.java
CommentMapper.java
package com.hadoop.training.examples.reducejoin;
import java.io.IOException;
public class UserMapper extends Mapper<LongWritable, Text, Text, Text>{
private Text outkey = new Text();
private Text outvalue = new Text();
@Override
public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
// Parse the input string into a nice map
Map<String, String> parsed = MRDPUtils.transformXmlToMap(value.toString());
String userId = parsed.get("Id");
if (userId == null) {
return;
}
// The foreign join key is the user ID
outkey.set(userId);
// Flag this record for the reducer and then output
outvalue.set("A"+ value.toString());
context.write(outkey, outvalue);
}
}
User dataset
split1
split2
Comments dataset
split1
split2
split3
Map task
Map task
Map task
Map task
Map task
UserID(key)
User record(value)
UserID(key)
Comments record(value)
UserID
[User record, comment record1, comment record2, ...]
Reduce task
Reduce task
which value is from which dataset?
Writable
Smart Insert
1 : 1

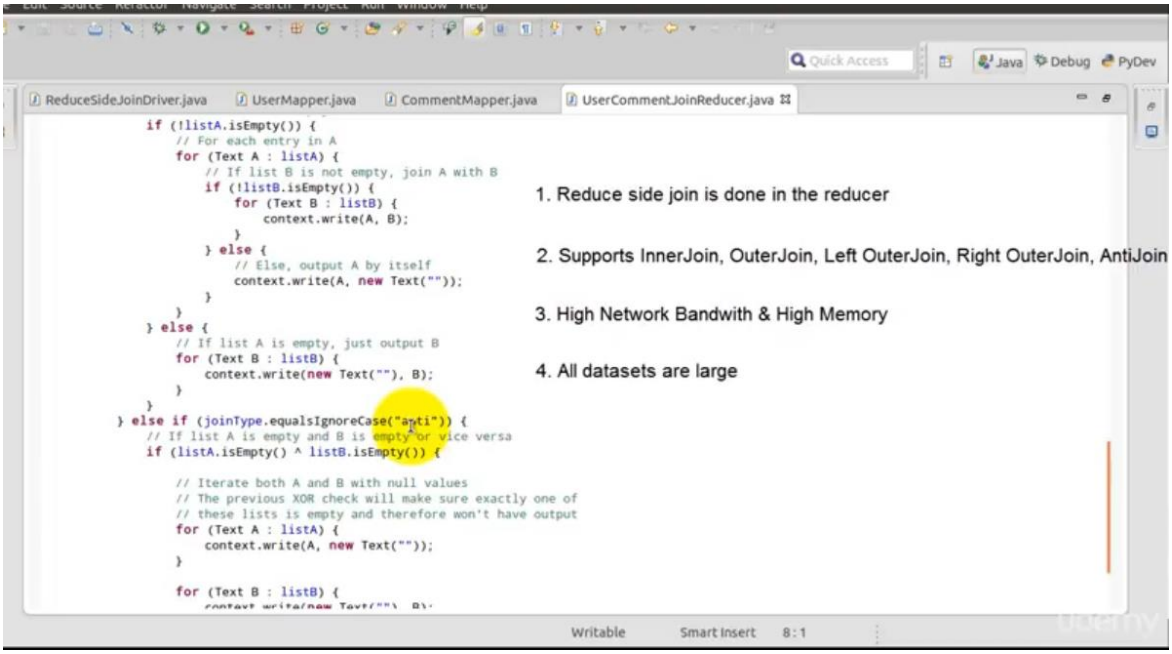

User dataset
split1
split2
Comments dataset
split1
split2
split3
Map task
Map task
Map task
Map task
Map task
UserID(key)
User record(value)
UserID(key)
Comments record(value)
UserID
[User record, comment record1, comment record2, ...]
Reduce task
Reduce task
which value is from which dataset?
ReduceSideJoinDriver.java
UserMapper.java
CommentMapper.java
package com.hadoop.training.examples.reducejoin;
import java.io.IOException;
public class CommentMapper extends Mapper<LongWritable, Text, Text, Text>{
private Text outkey = new Text();
private Text outvalue = new Text();
@Override
public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
// Parse the input string into a nice map
Map<String, String> parsed = MRDPUtils.transformXmlToMap(value.toString());
String userId = parsed.get("UserId");
if (userId == null) {
return;
}
// The foreign join key is the user ID
outkey.set(userId);
// Flag this record for the reducer and then output
outvalue.set("B" + value.toString());
context.write(outkey, outvalue);
}
}
Writable
Smart Insert
1 : 1


Quick Access
Java
Debug
PyDev
ReduceSideJoinDriver.java
UserMapper.java
CommentMapper.java
UserCommentJoinReducer.java
public class UserCommentJoinReducer extends Reducer<Text, Text, Text, Text> {
private ArrayList<Text> listA = new ArrayList<Text>();
private ArrayList<Text> listB = new ArrayList<Text>();
private String joinType = null;
@Override
public void setup(Context context) {
// Get the type of join from our configuration
joinType = context.getConfiguration().get("join.type");
}
@Override
public void reduce(Text key, Iterable<Text> values, Context context)
throws IOException, InterruptedException {
// Clear our lists
listA.clear();
listB.clear();
// iterate through all our values, binning each record based on what
// it was tagged with
// make sure to remove the tag!
for (Text t : values) {
if (t.charAt(0) == 'A') {
listA.add(new Text(t.toString().substring(1)));
} else if (t.charAt('0') == 'B') {
listB.add(new Text(t.toString().substring(1)));
}
}
// Execute our join logic now that the lists are filled

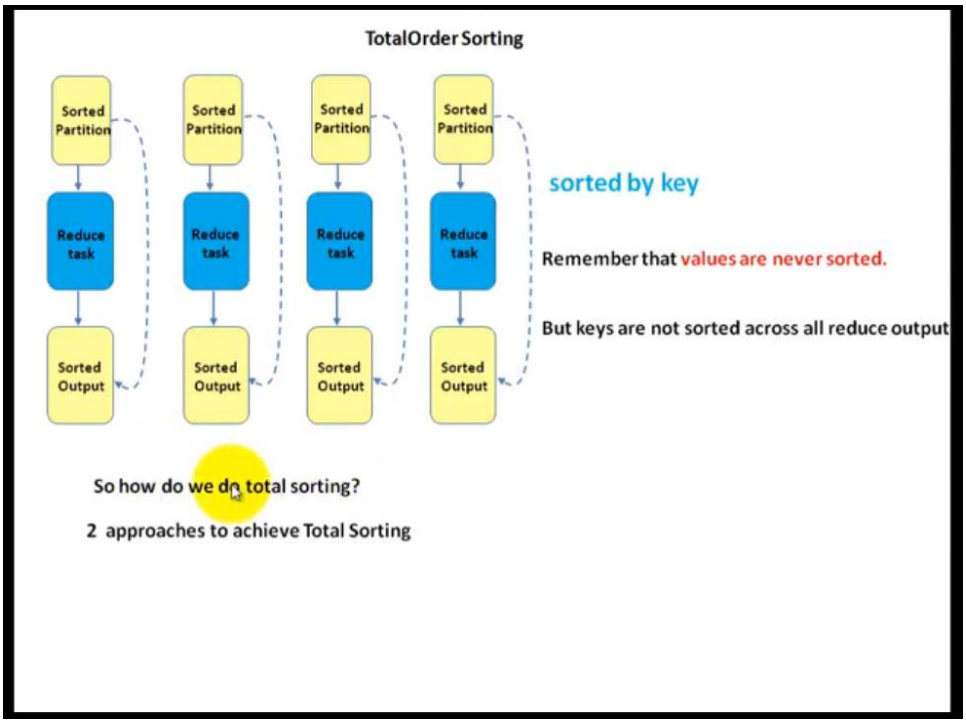

Quick Access
Java
Debug
PyDev
ReduceSideJoinDriver.java
UserMapper.java
CommentMapper.java
UserCommentJoinReducer.java
// iterate through all our values, binning each record based on what
// it was tagged with
// make sure to remove the tag!
for (Text t : values) {
if (t.charAt(0) == 'A') {
listA.add(new Text(t.toString().substring(1)));
} else if (t.charAt('0') == 'B') {
listB.add(new Text(t.toString().substring(1)));
}
}
// Execute our join logic now that the lists are filled
executeJoinLogic(context);
}
private void executeJoinLogic(Context context) throws IOException,
InterruptedException {
if (joinType.equalsIgnoreCase("inner")) {
// If both lists are not empty, join A with B
if (!listA.isEmpty() && !listB.isEmpty()) {
for (Text A : listA) {
for (Text B : listB) {
context.write(A, B);
}
}
}
} else if (joinType.equalsIgnoreCase("leftouter")) {
// For each entry in A,
for (Text A : listA) {
// If list B is not empty, join A and B
if (!listB.isEmpty()) {

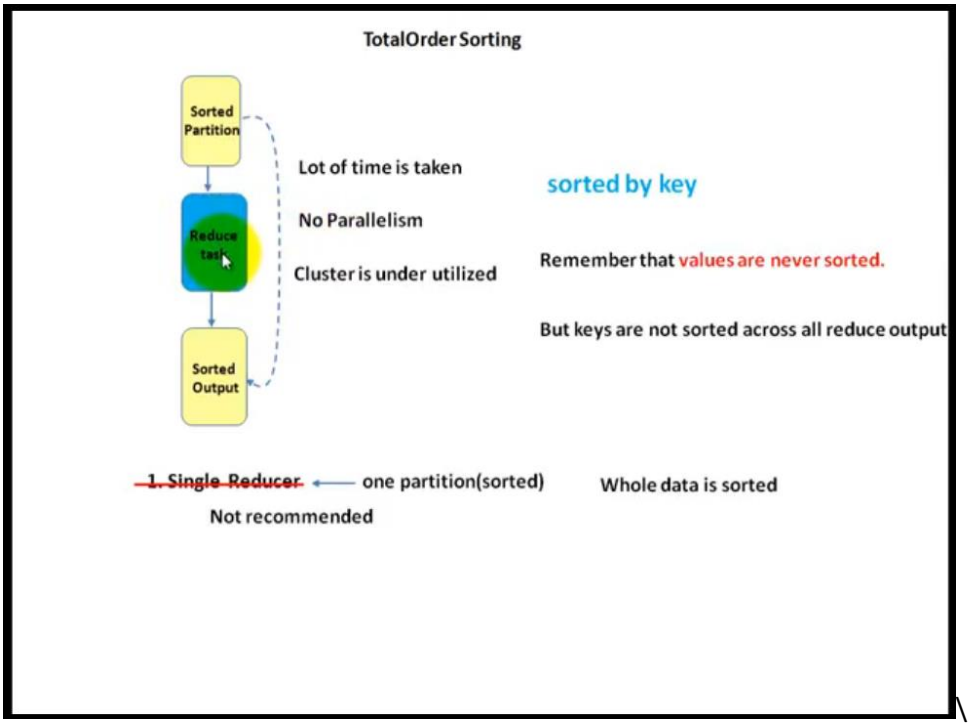

File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
PyDev
ReduceSideJoinDriver.java
UserMapper.java
CommentMapper.java
UserCommentJoinReducer.java
private void executeJoinLogic(Context context) throws IOException,
InterruptedException {
if (joinType.equalsIgnoreCase("inner")) {
// If both lists are not empty, join A with B
if (!listA.isEmpty() && !listB.isEmpty()) {
for (Text A : listA) {
for (Text B : listB) {
context.write(A, B);
}
}
}
} else if (joinType.equalsIgnoreCase("leftouter")) {
// For each entry in A,
for (Text A : listA) {
// If list B is not empty, join A and B
if (!listB.isEmpty()) {
for (Text B : listB) {
context.write(A, B);
}
} else {
// Else, output A by itself
context.write(A, new Text(""));
}
}
} else if (joinType.equalsIgnoreCase("rightouter")) {
// FOr each entry in B,
for (Text B : listB) {
// If list A is not empty, join A and B
if (!listA.isEmpty()) {
for (Text A : listA) {
context.write(A, B);

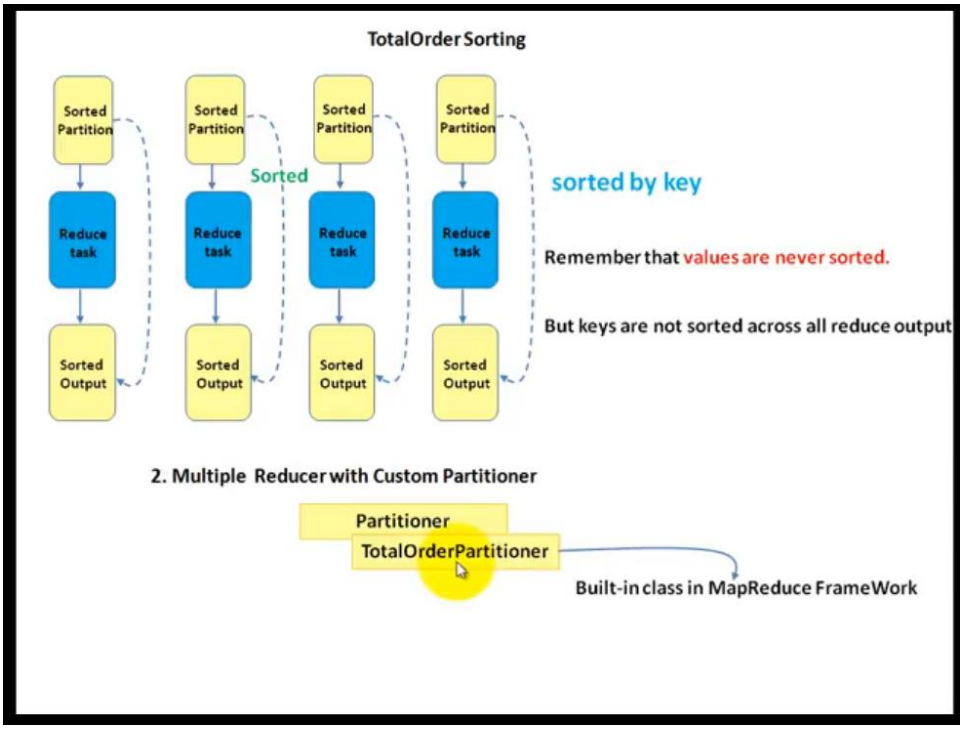

File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
PyDev
ReduceSideJoinDriver.java
UserMapper.java
CommentMapper.java
UserCommentJoinReducer.java
if (!listA.isEmpty()) {
// For each entry in A
for (Text A : listA) {
// If list B is not empty, join A with B
if (!listB.isEmpty()) {
for (Text B : listB) {
context.write(A, B);
}
} else {
// Else, output A by itself
context.write(A, new Text(""));
}
}
} else {
// If list A is empty, just output B
for (Text B : listB) {
context.write(new Text(""), B);
}
}
} else if (joinType.equalsIgnoreCase("outti")) {
// If list A is empty and B is empty or vice versa
if (listA.isEmpty() ^ listB.isEmpty()) {
// Iterate both A and B with null values
// The previous XOR check will make sure exactly one of
// these lists is empty and therefore won't have output
for (Text A : listA) {
context.write(A, new Text(""));
}
for (Text B : listB) {
Writable
Smart Insert
8 : 1

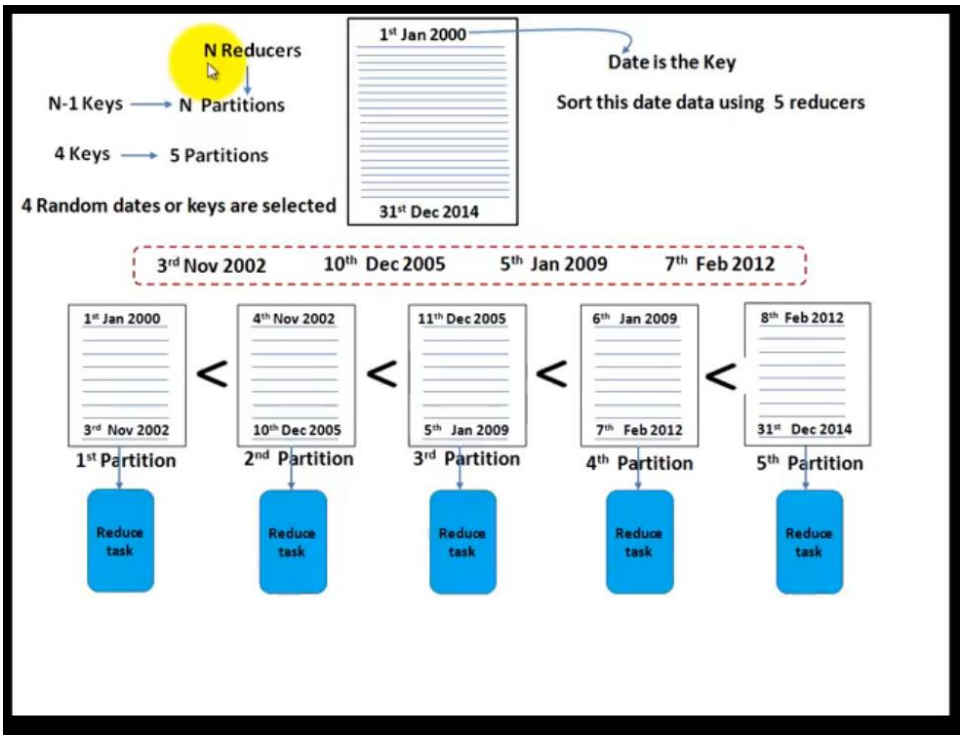

ReduceSideJoinDriver.java
UserMapper.java
CommentMapper.java
UserCommentJoinReducer.java
Quick Access
Java
Debug
PyDev
if (!listA.isEmpty()) {
// For each entry in A
for (Text A : listA) {
// If list B is not empty, join A with B
if (!listB.isEmpty()) {
for (Text B : listB) {
context.write(A, B);
}
} else {
// Else, output A by itself
context.write(A, new Text(""));
}
}
} else {
// If list A is empty, just output B
for (Text B : listB) {
context.write(new Text(""), B);
}
}
} else if (joinType.equalsIgnoreCase("anti")) {
// If list A is empty and B is empty or vice versa
if (listA.isEmpty() ^ listB.isEmpty()) {
// Iterate both A and B with null values
// The previous XOR check will make sure exactly one of
// these lists is empty and therefore won't have output
for (Text A : listA) {
context.write(A, new Text(""));
}
for (Text B : listB) {
1. Reduce side join is done in the reducer
2. Supports InnerJoin, OuterJoin, Left OuterJoin, Right OuterJoin, AntiJoin
3. High Network Bandwith & High Memory
4. All datasets are large
Writable
Smart Insert
8 : 1

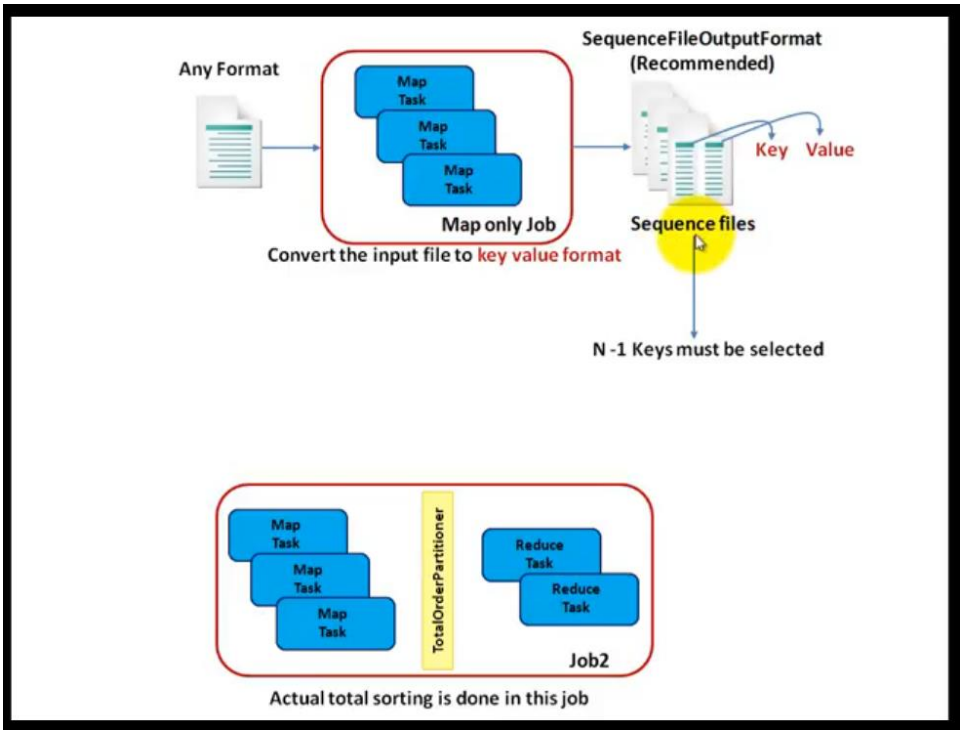
Total Order Sorting Pattern

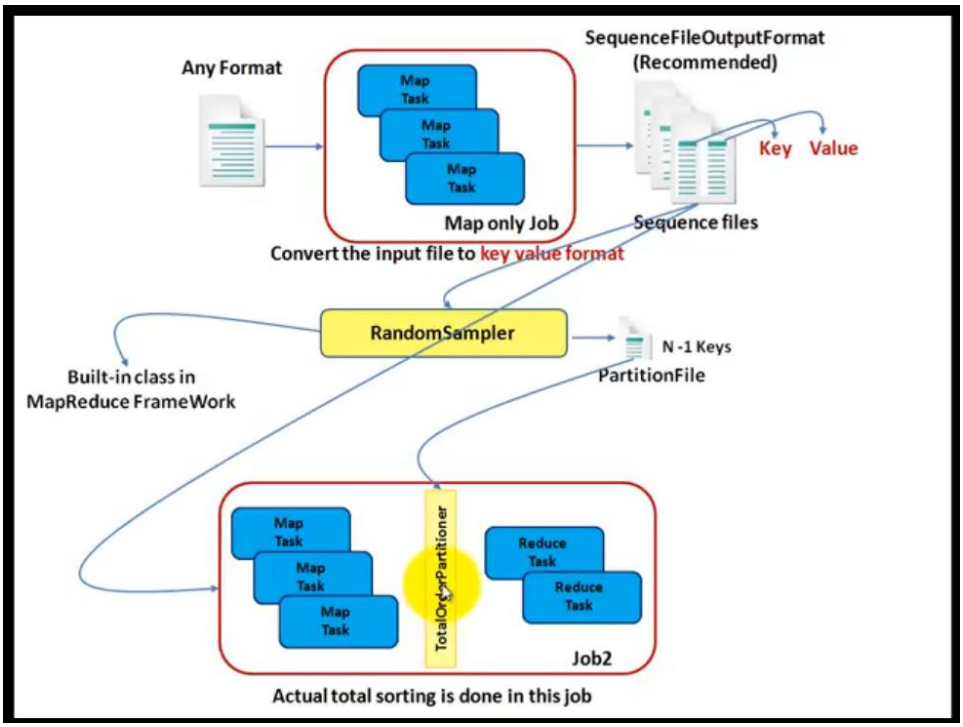





TotalOrder Sorting
Sorted Partition
Reduce task
Sorted Output
Sorted
sorted by key
Remember that values are never sorted.
But keys are not sorted across all reduce output
2. Multiple Reducer with Custom Partitioner
Partitioner
TotalOrderPartitioner
Built-in class in MapReduce FrameWork

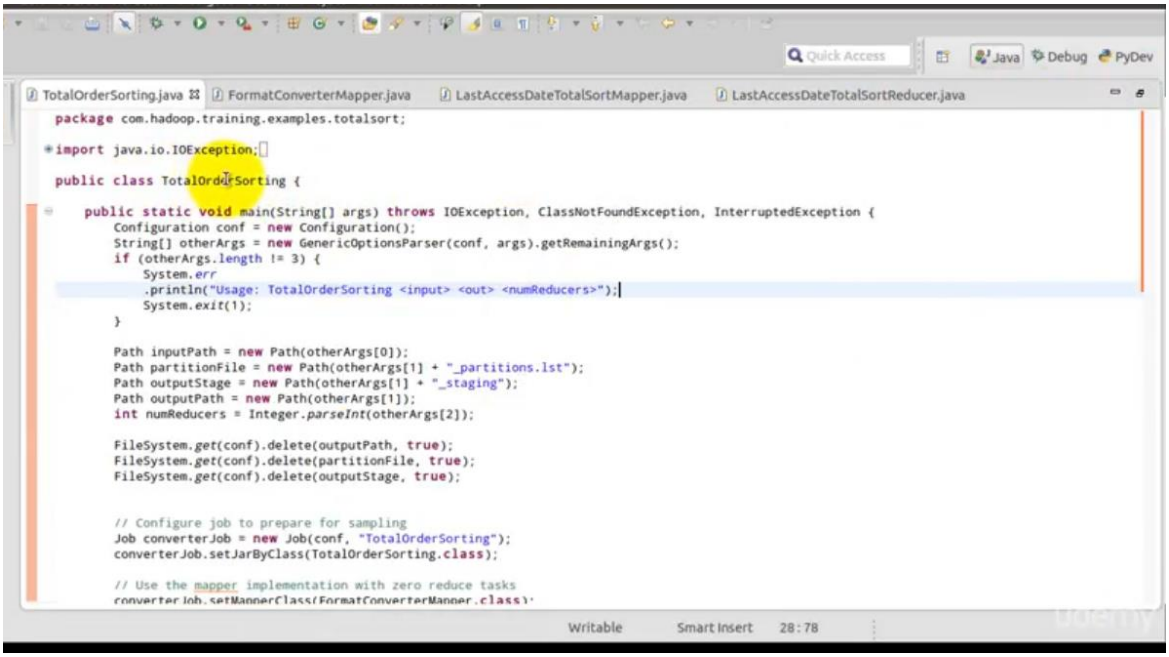

N Reducers
N-1 Keys → N Partitions
4 Keys → 5 Partitions
4 Random dates or keys are selected
1st Jan 2000
31st Dec 2014
Date is the Key
Sort this date data using 5 reducers
3rd Nov 2002
10th Dec 2005
5th Jan 2009
7th Feb 2012
1st Jan 2000
3rd Nov 2002
1st Partition
4th Nov 2002
10th Dec 2005
2nd Partition
11th Dec 2005
5th Jan 2009
3rd Partition
6th Jan 2009
7th Feb 2012
4th Partition
8th Feb 2012
31st Dec 2014
5th Partition
Reduce task

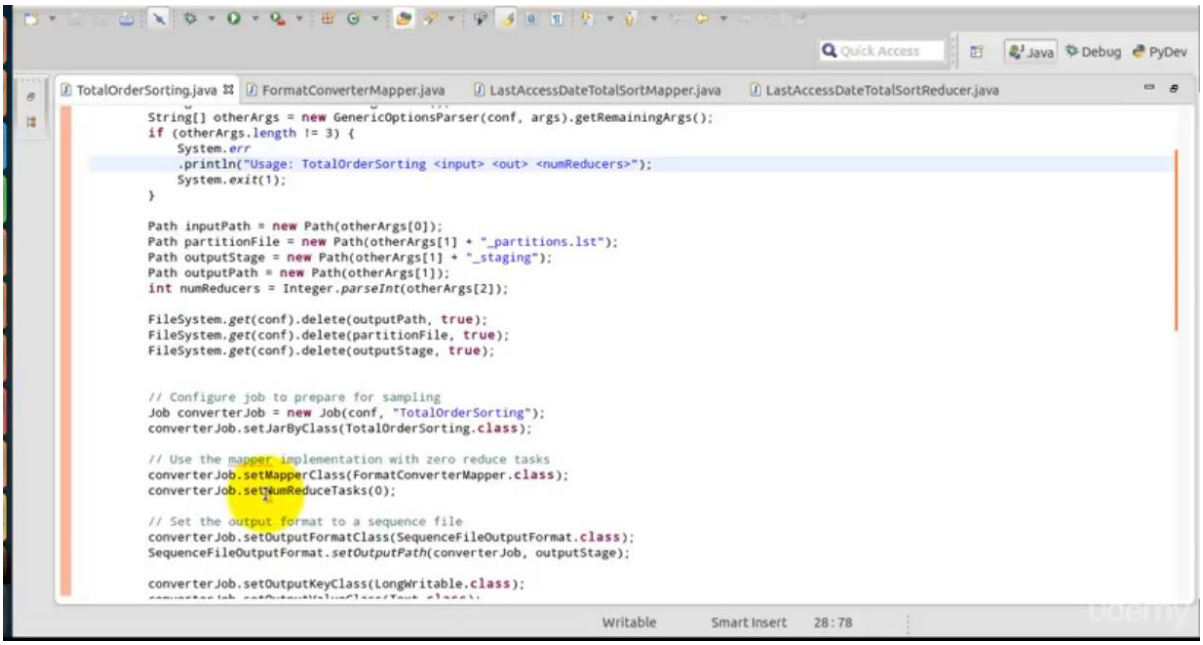

Any Format
Map Task
Map Task
Map Task
Map only Job
Convert the input file to key value format
SequenceFileOutputFormat
(Recommended)
Key
Value
Sequence files
N -1 Keys must be selected
Map Task
Map Task
Map Task
TotalOrderPartitioner
Reduce Task
Reduce Task
Job2
Actual total sorting is done in this job

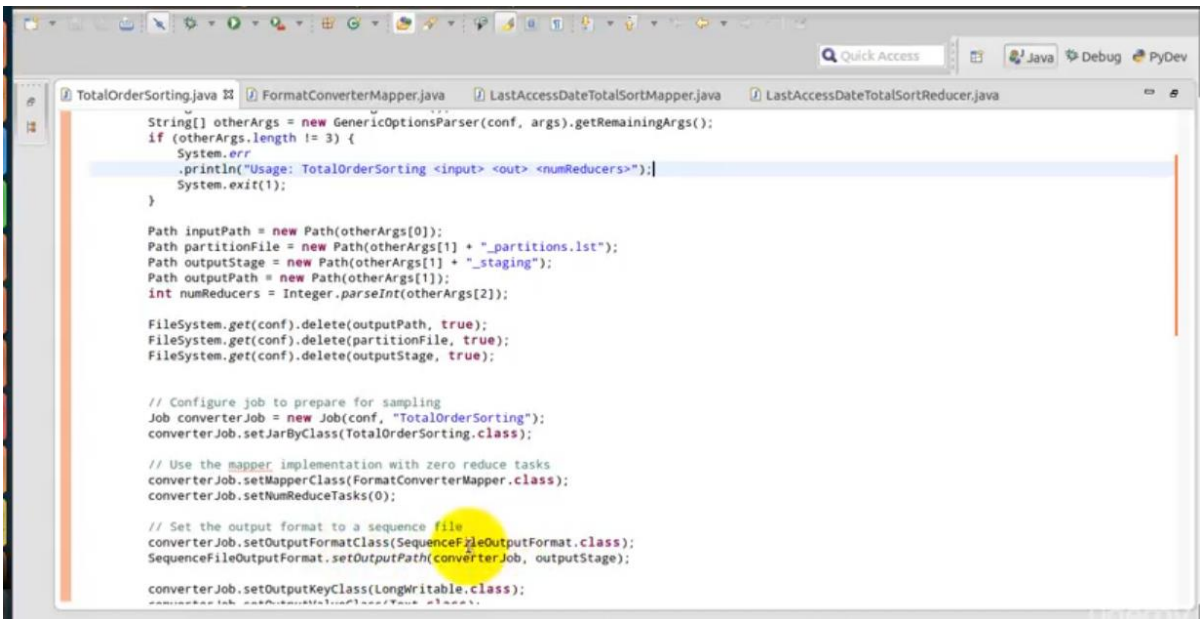

Any Format
Map Task
Map Task
Map Task
Map only Job
Convert the input file to key value format
SequenceFileOutputFormat
(Recommended)
Key
Value
Sequence files
RandomSampler
N -1 Keys
PartitionFile
Built-in class in
MapReduce FrameWork
Map Task
Map Task
Map Task
TotalOrderPartitioner
Reduce Task
Reduce Task
Job2
Actual total sorting is done in this job

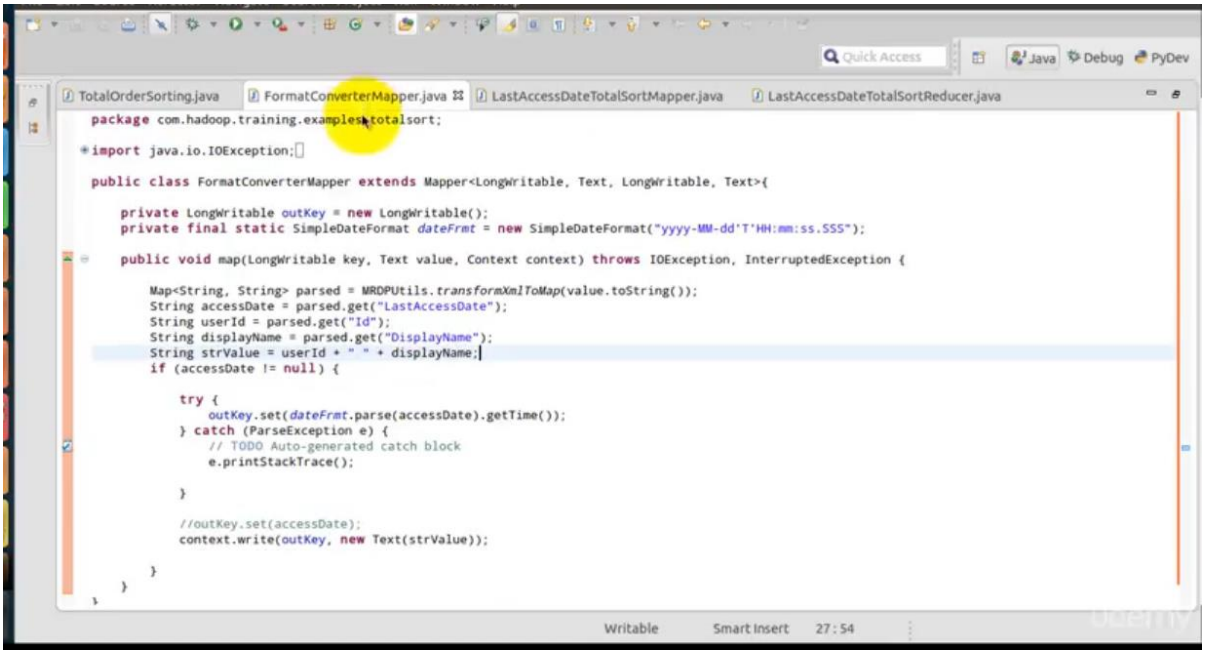
Problem: Sort user data by last access date

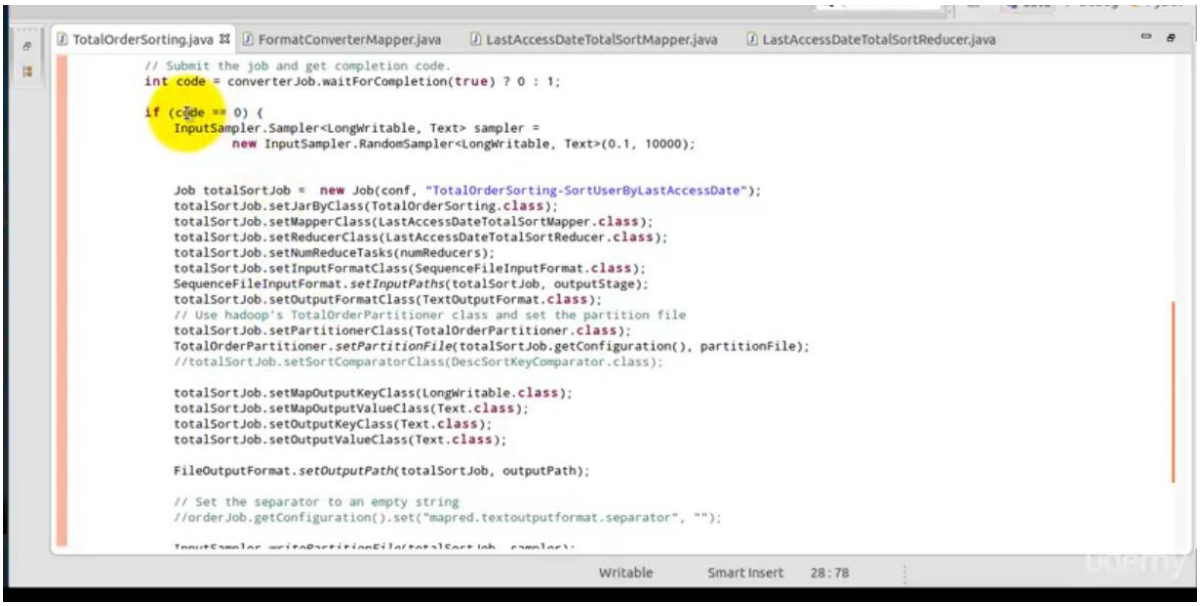
Currently it is sorted by CreationDate



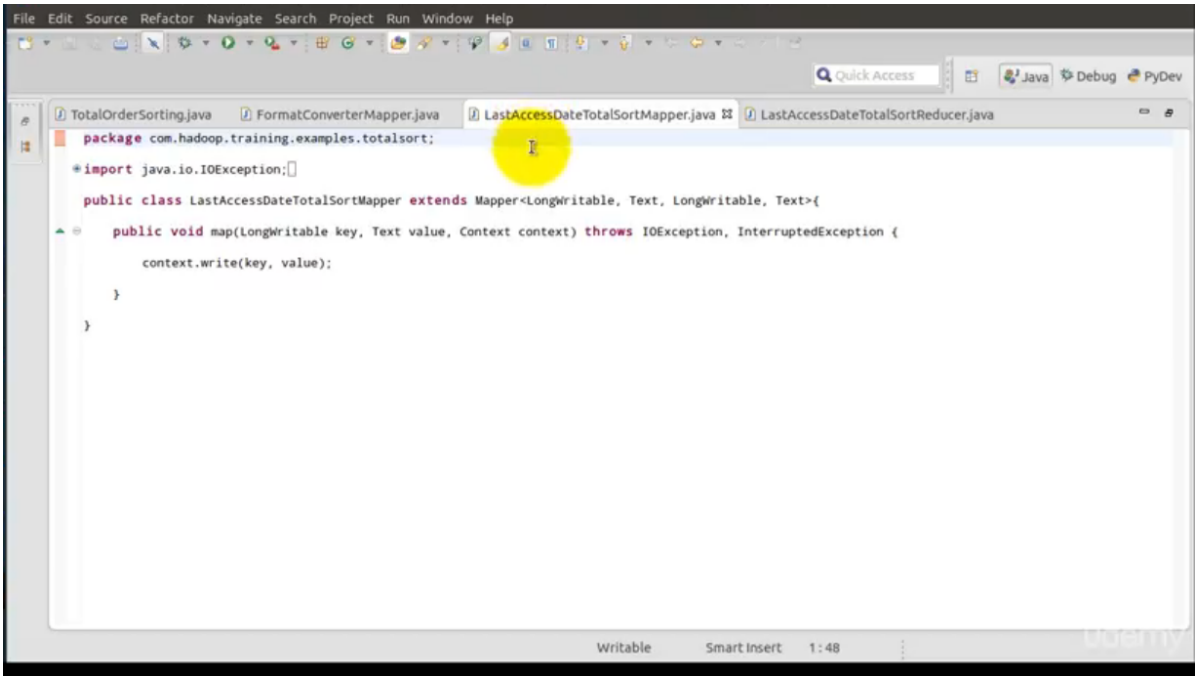

Quick Access
Java
Debug
PyDev
TotalOrderSorting.java
FormatConverterMapper.java
LastAccessDateTotalSortMapper.java
LastAccessDateTotalSortReducer.java
String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
if (otherArgs.length != 3) {
System.err
.println("Usage: TotalOrderSorting <input> <out> <numReducers>");
System.exit(1);
}
Path inputPath = new Path(otherArgs[0]);
Path partitionFile = new Path(otherArgs[1] + "_partitions.lst");
Path outputStage = new Path(otherArgs[1] + "_staging");
Path outputPath = new Path(otherArgs[1]);
int numReducers = Integer.parseInt(otherArgs[2]);
FileSystem.get(conf).delete(outputPath, true);
FileSystem.get(conf).delete(partitionFile, true);
FileSystem.get(conf).delete(outputStage, true);
// Configure job to prepare for sampling
Job converterJob = new Job(conf, "TotalOrderSorting");
converterJob.setJarByClass(TotalOrderSorting.class);
// Use the mapper implementation with zero reduce tasks
converterJob.setMapperClass(FormatConverterMapper.class);
converterJob.setNumReduceTasks(0);
// Set the output format to a sequence file
converterJob.setOutputFormatClass(SequenceFileOutputFormat.class);
SequenceFileOutputFormat.setOutputPath(converterJob, outputStage);
converterJob.setOutputKeyClass(LongWritable.class);
Writable
Smart Insert
28 : 78

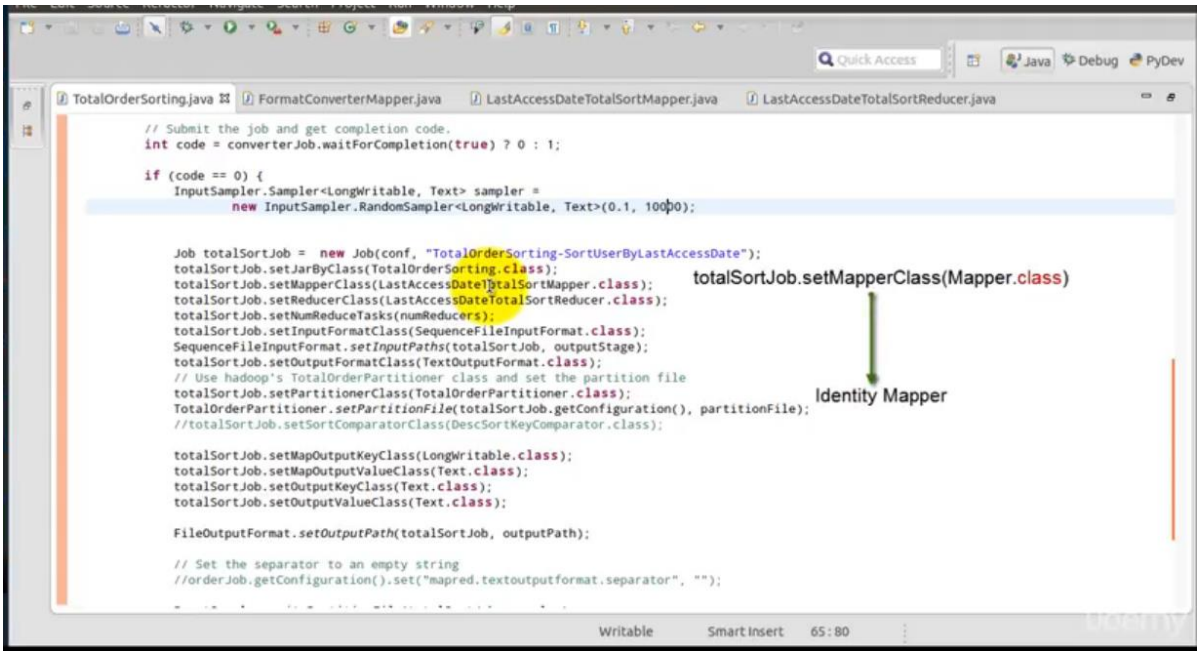

Quick Access
Java
Debug
PyDev
TotalOrderSorting.java
FormatConverterMapper.java
LastAccessDateTotalSortMapper.java
LastAccessDateTotalSortReducer.java
String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
if (otherArgs.length != 3) {
System.err
.println("Usage: TotalOrderSorting <input> <out> <numReducers>");
System.exit(1);
}
Path inputPath = new Path(otherArgs[0]);
Path partitionFile = new Path(otherArgs[1] + "_partitions.lst");
Path outputStage = new Path(otherArgs[1] + "_staging");
Path outputPath = new Path(otherArgs[1]);
int numReducers = Integer.parseInt(otherArgs[2]);
FileSystem.get(conf).delete(outputPath, true);
FileSystem.get(conf).delete(partitionFile, true);
FileSystem.get(conf).delete(outputStage, true);
// Configure job to prepare for sampling
Job converterJob = new Job(conf, "TotalOrderSorting");
converterJob.setJarByClass(TotalOrderSorting.class);
// Use the mapper implementation with zero reduce tasks
converterJob.setMapperClass(FormatConverterMapper.class);
converterJob.setNumReduceTasks(0);
// Set the output format to a sequence file
converterJob.setOutputFormatClass(SequenceFileOutputFormat.class);
SequenceFileOutputFormat.setOutputPath(converterJob, outputStage);
converterJob.setOutputKeyClass(LongWritable.class);



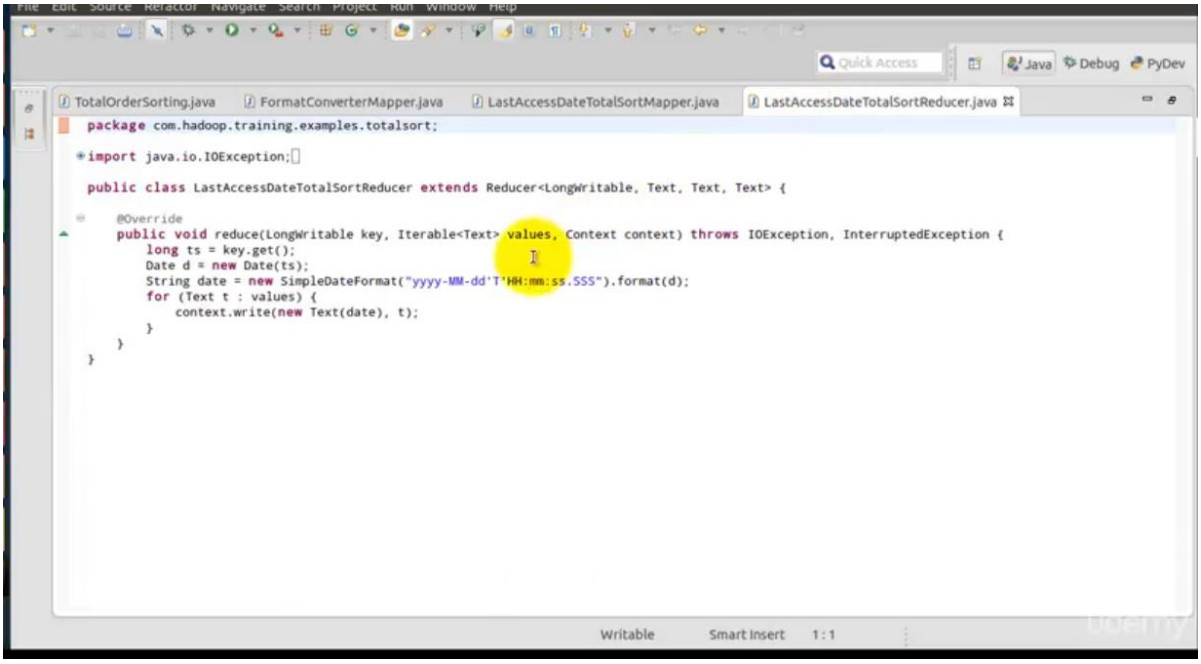


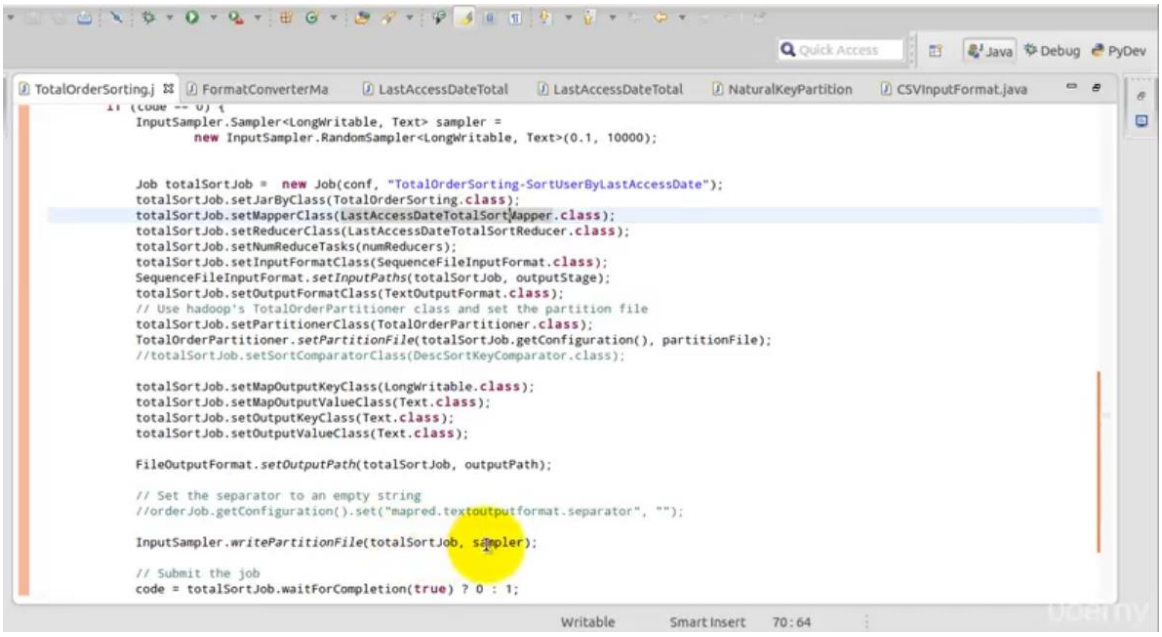
RandomSampler object select N-1 random keys and write it to the partition file it will pick 10000 records from the input data, probability of picking a record is 0.1 it means for every 10 records there is a probability that record will be picked, finally N-1 keys selected from 10000 records, then these records are sorted and returned to the partitioned file

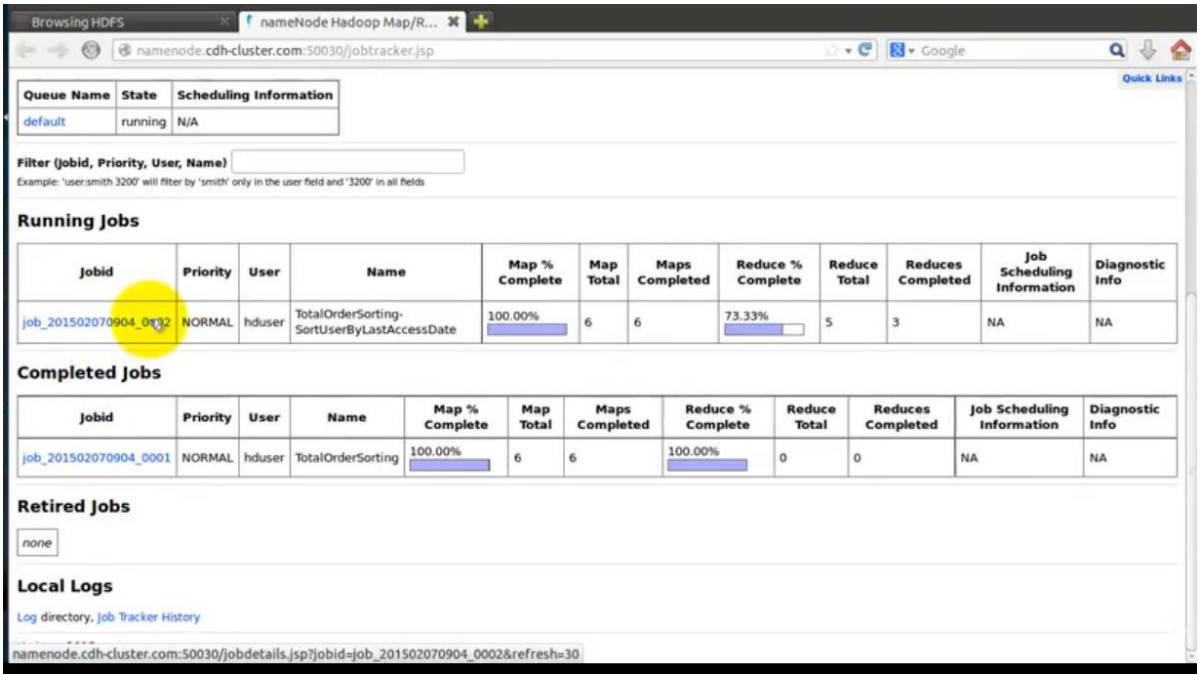




As we are not doing anything in custom mapper class so we can use Mapper.class provided by Hadoop framework which is also called Identity Mapper

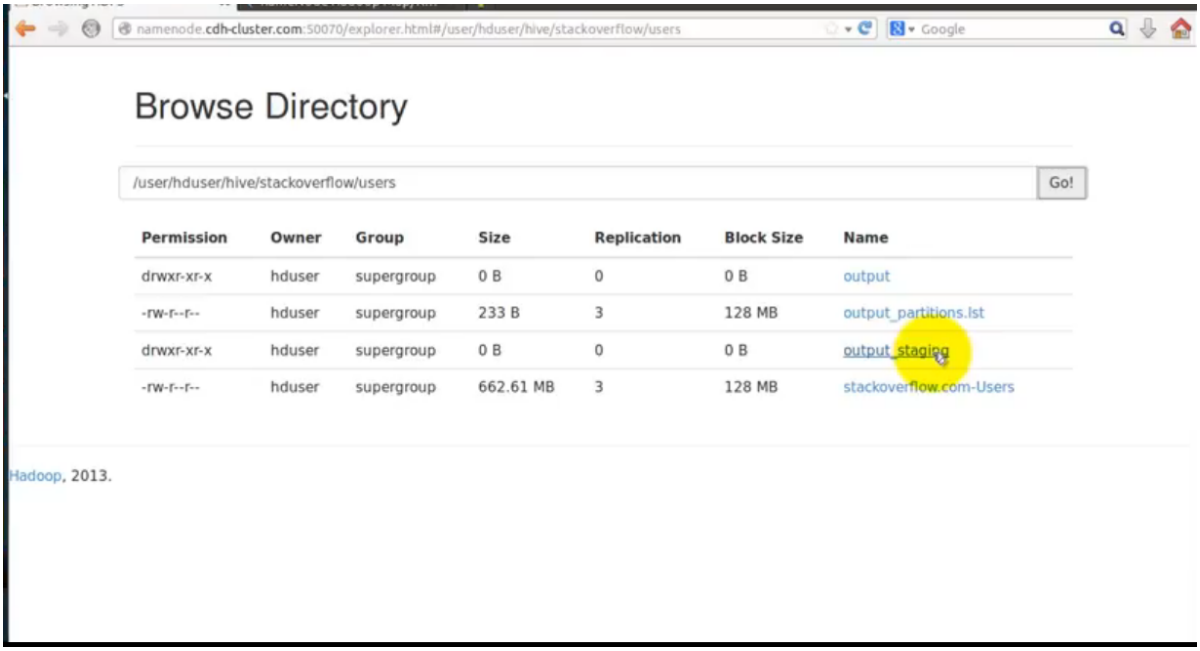

File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
PyDev
TotalOrderSorting.java
FormatConverterMapper.java
LastAccessDateTotalSortMapper.java
LastAccessDateTotalSortReducer.java
package com.hadoop.training.examples.totalsort;
import java.io.IOException;
public class LastAccessDateTotalSortReducer extends Reducer<LongWritable, Text, Text, Text> {
@Override
public void reduce(LongWritable key, Iterable<Text> values, Context context) throws IOException, InterruptedException {
long ts = key.get();
Date d = new Date(ts);
String date = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSS").format(d);
for (Text t : values) {
context.write(new Text(date), t);
}
}
}
Writable
Smart Insert
1 : 1


Quick Access
Java
Debug
PyDev
TotalOrderSorting.j
FormatConverterMa
LastAccessDateTotal
LastAccessDateTotal
NaturalKeyPartition
CSVInputFormat.java
InputSampler.Sampler<LongWritable, Text> sampler =
new InputSampler.RandomSampler<LongWritable, Text>(0.1, 10000);
Job totalSortJob = new Job(conf, "TotalOrderSorting-SortUserByLastAccessDate");
totalSortJob.setJarByClass(TotalOrderSorting.class);
totalSortJob.setMapperClass(LastAccessDateTotalSortMapper.class);
totalSortJob.setReducerClass(LastAccessDateTotalSortReducer.class);
totalSortJob.setNumReduceTasks(numReducers);
totalSortJob.setInputFormatClass(SequenceFileInputFormat.class);
SequenceFileInputFormat.setInputPaths(totalSortJob, outputStage);
totalSortJob.setOutputFormatClass(TextOutputFormat.class);
// Use hadoop's TotalOrderPartitioner class and set the partition file
totalSortJob.setPartitionerClass(TotalOrderPartitioner.class);
TotalOrderPartitioner.setPartitionFile(totalSortJob.getConfiguration(), partitionFile);
//totalSortJob.setSortComparatorClass(DescSortKeyComparator.class);
totalSortJob.setMapOutputKeyClass(LongWritable.class);
totalSortJob.setMapOutputValueClass(Text.class);
totalSortJob.setOutputKeyClass(Text.class);
totalSortJob.setOutputValueClass(Text.class);
FileOutputFormat.setOutputPath(totalSortJob, outputPath);
// Set the separator to an empty string
//orderJob.getConfiguration().set("mapred.textoutputformat.separator", "");
InputSampler.writePartitionFile(totalSortJob, sampler);
// Submit the job
code = totalSortJob.waitForCompletion(true) ? 0 : 1;
Writable
Smart Insert
70 : 64

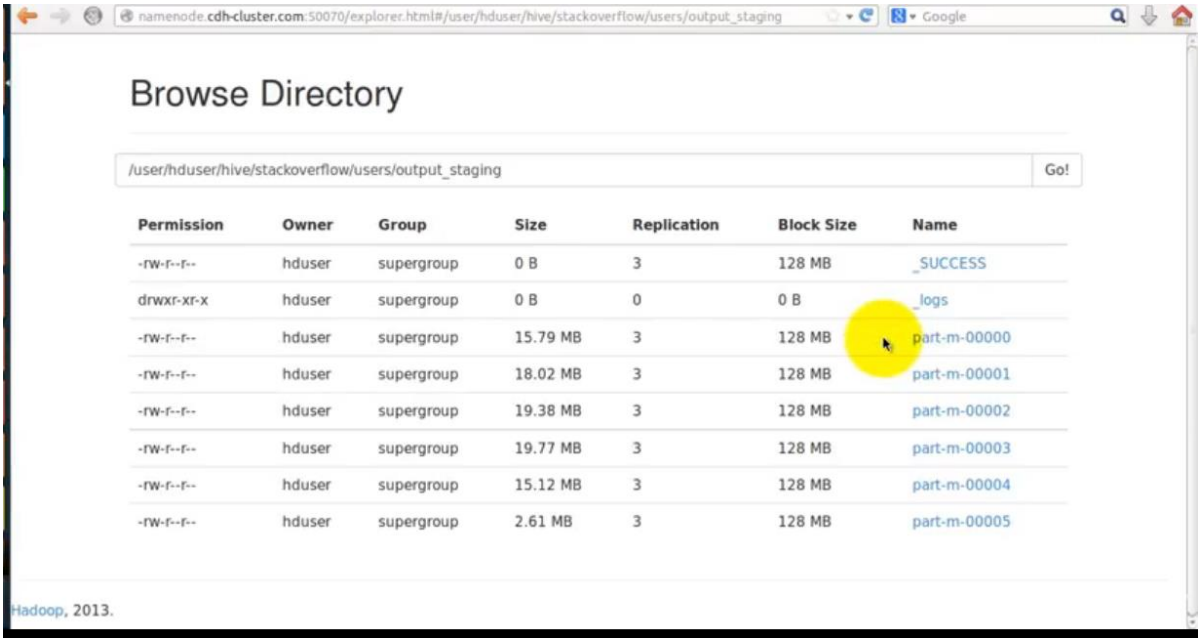


Two jobs running , first job is map only job and second job is traditional job

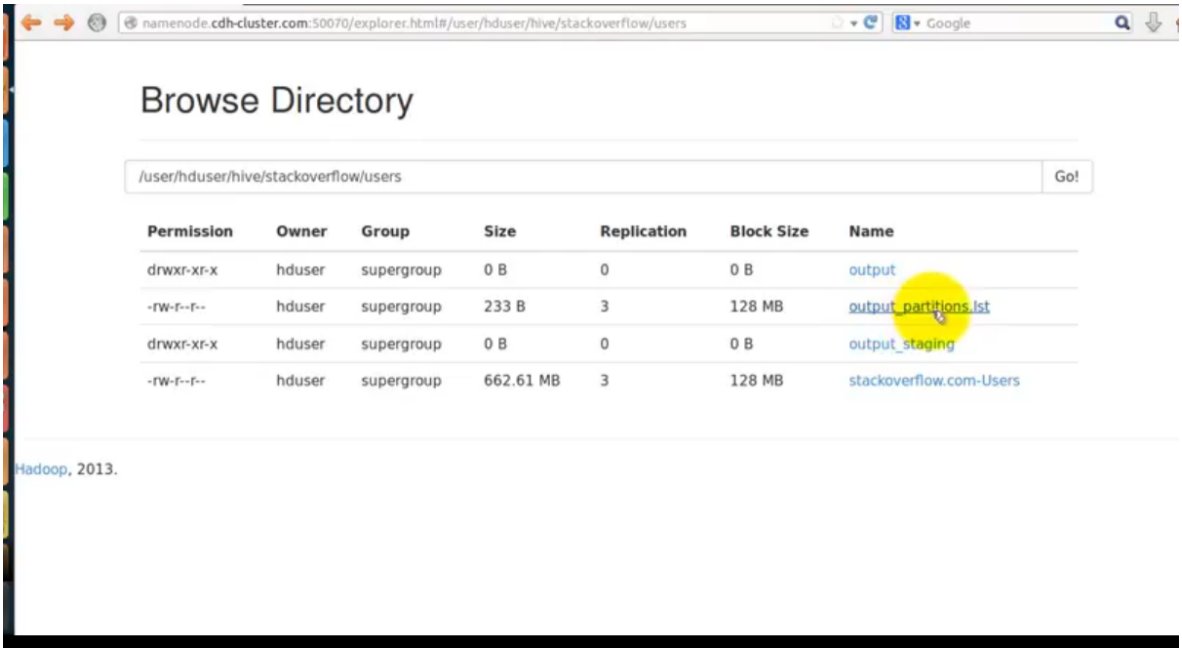
Output



Output-staging contains output of the first job (map only job)

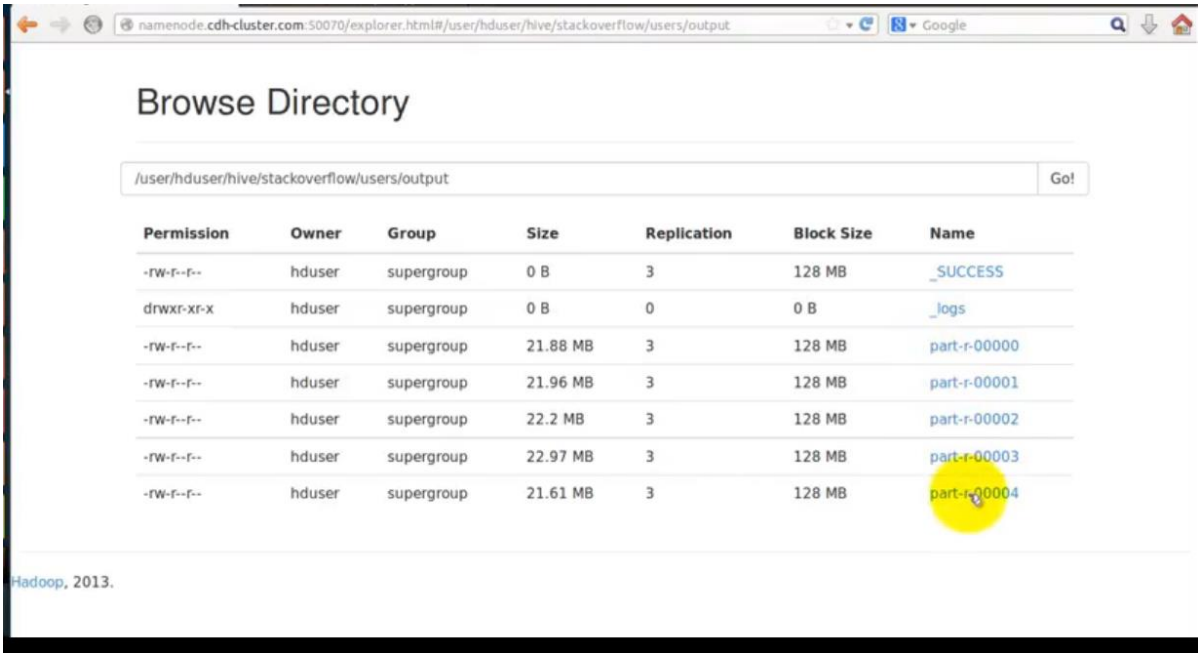


6 map tasks created 6 map files and these are sequence files

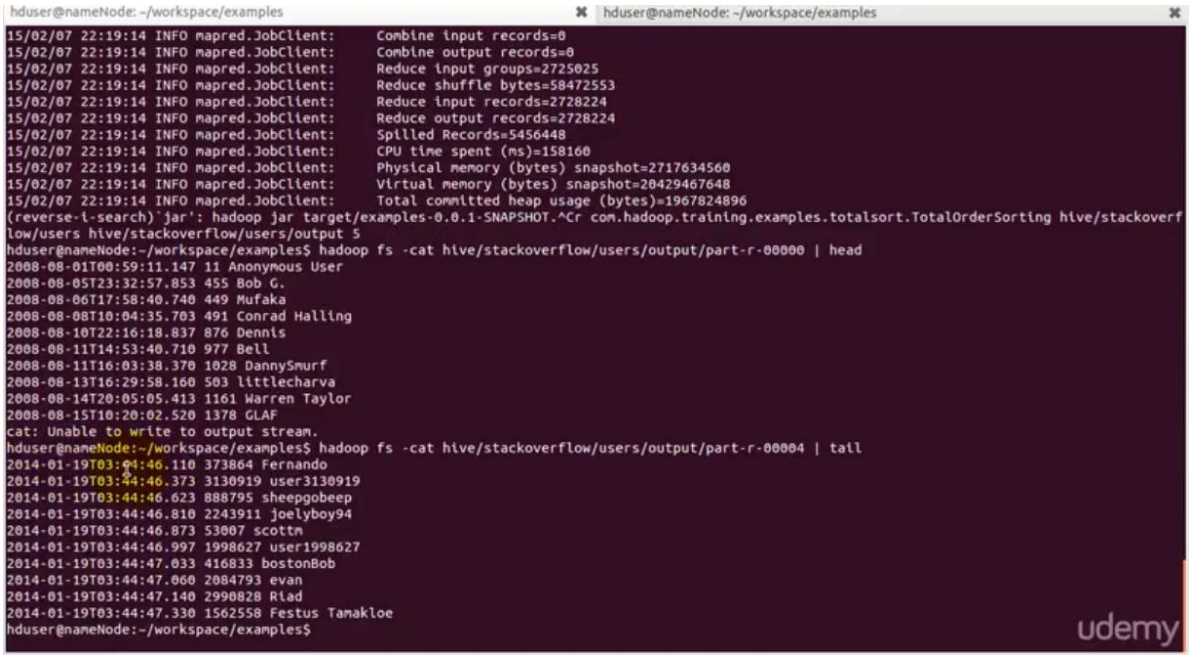


Output-partitions.txt is partition file, it contains 4 random selected keys

And output folder is final output directory

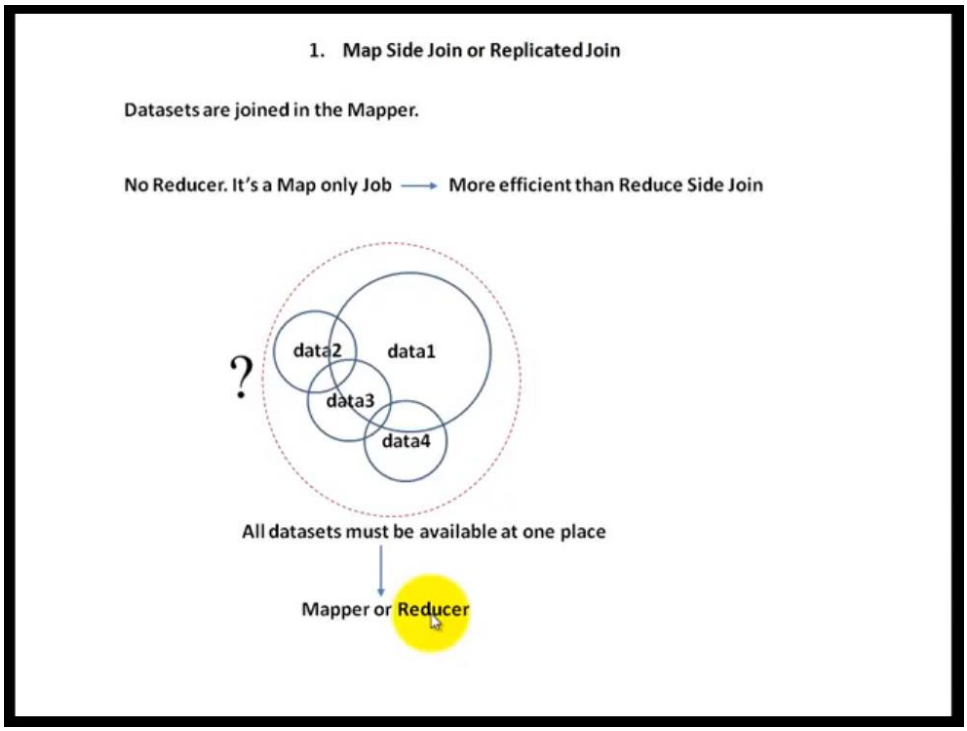


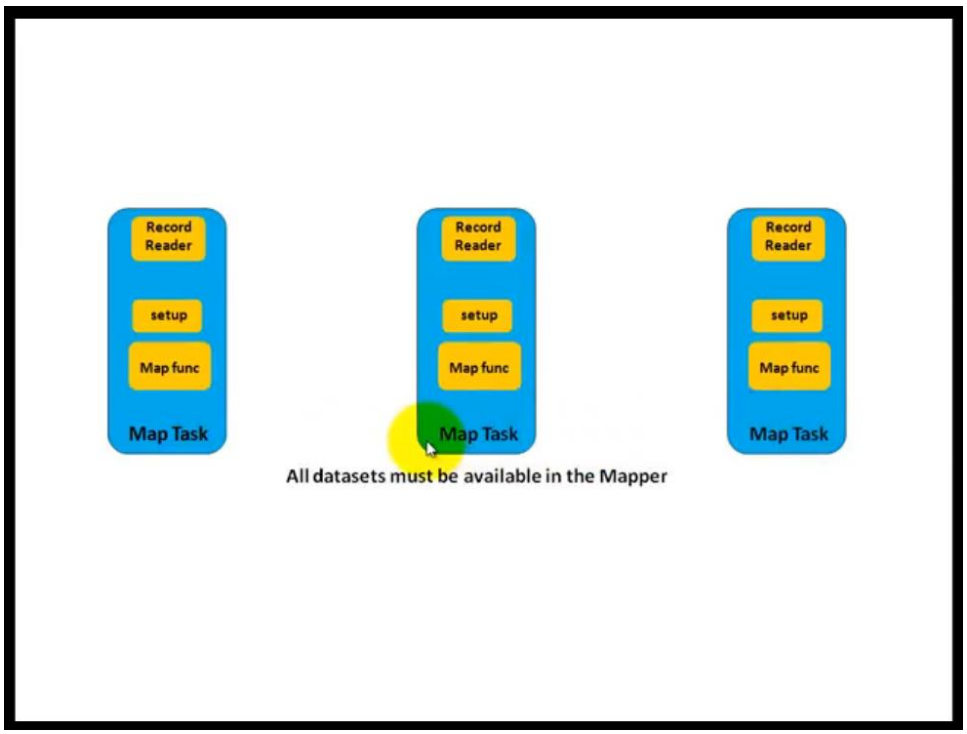
5 output files are created

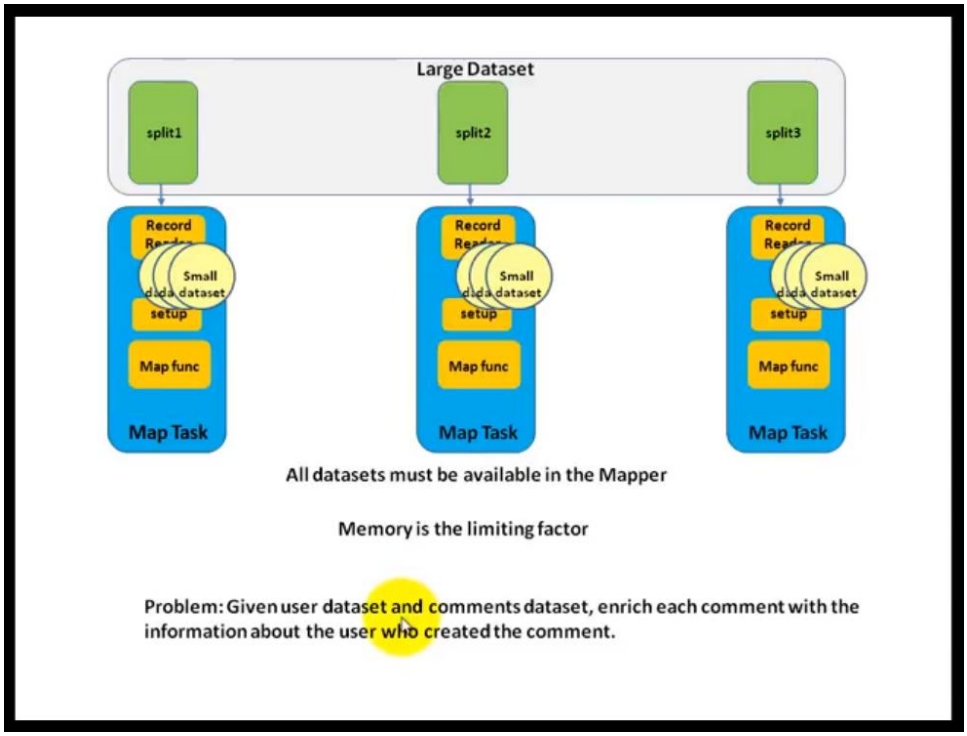

hduser@nameNode: ~/workspace/examples
hduser@nameNode: ~/workspace/examples
15/02/07 22:19:14 INFO mapred.JobClient: Combine input records=0
15/02/07 22:19:14 INFO mapred.JobClient: Combine output records=0
15/02/07 22:19:14 INFO mapred.JobClient: Reduce input groups=2725025
15/02/07 22:19:14 INFO mapred.JobClient: Reduce shuffle bytes=58472553
15/02/07 22:19:14 INFO mapred.JobClient: Reduce input records=2728224
15/02/07 22:19:14 INFO mapred.JobClient: Reduce output records=2728224
15/02/07 22:19:14 INFO mapred.JobClient: Spilled Records=5456448
15/02/07 22:19:14 INFO mapred.JobClient: CPU time spent (ms)=158160
15/02/07 22:19:14 INFO mapred.JobClient: Physical memory (bytes) snapshot=2717634560
15/02/07 22:19:14 INFO mapred.JobClient: Virtual memory (bytes) snapshot=20429467648
15/02/07 22:19:14 INFO mapred.JobClient: Total committed heap usage (bytes)=1967824896
(reverse-i-search)`jar': hadoop jar target/examples-0.0.1-SNAPSHOT.^Cr com.hadoop.training.examples.totalsort.TotalOrderSorting hive/stackoverf
low/users hive/stackoverflow/users/output 5
hduser@nameNode:~/workspace/examples$ hadoop fs -cat hive/stackoverflow/users/output/part-r-00000 | head
2008-08-01T00:59:11.147 11 Anonymous User
2008-08-05T23:32:57.853 455 Bob G.
2008-08-06T17:58:40.740 449 Mufaka
2008-08-08T10:04:35.703 491 Conrad Halling
2008-08-10T22:16:18.837 876 Dennis
2008-08-11T14:53:40.710 977 Bell
2008-08-11T16:03:38.370 1028 DannySmurf
2008-08-13T16:29:58.160 503 littlecharva
2008-08-14T20:05:05.413 1161 Warren Taylor
2008-08-15T10:20:02.520 1378 GLAF
cat: Unable to write to output stream.
hduser@nameNode:~/workspace/examples$ hadoop fs -cat hive/stackoverflow/users/output/part-r-00004 | tail
2014-01-19T03:44:46.110 373864 Fernando
2014-01-19T03:44:46.373 3130919 user3130919
2014-01-19T03:44:46.623 888795 sheepgobeep
2014-01-19T03:44:46.810 2243911 joelyboy94
2014-01-19T03:44:46.873 53007 scottm
2014-01-19T03:44:46.997 1998627 user1998627
2014-01-19T03:44:47.033 416833 bostonBob
2014-01-19T03:44:47.060 2084793 evan
2014-01-19T03:44:47.140 2990828 Riad
2014-01-19T03:44:47.330 1562558 Festus Tamakloe
hduser@nameNode:~/workspace/examples$
udemy

Map Side join Pattern (Replicated Join)

- Map side join
- No reducer class
- Join between one large dataset and many smaller dataset



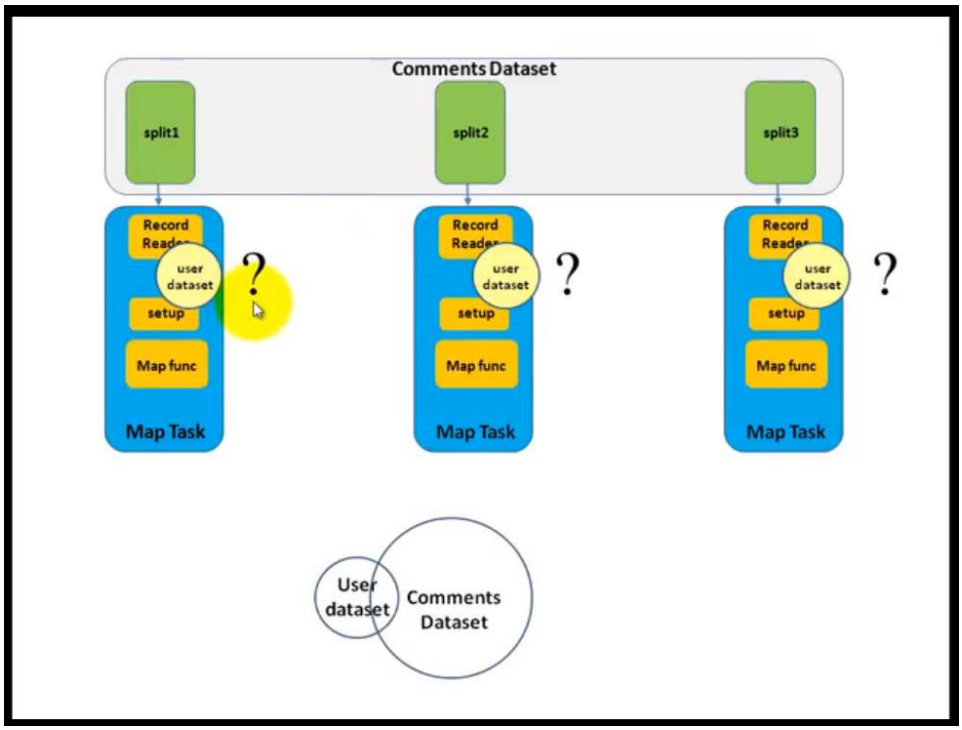




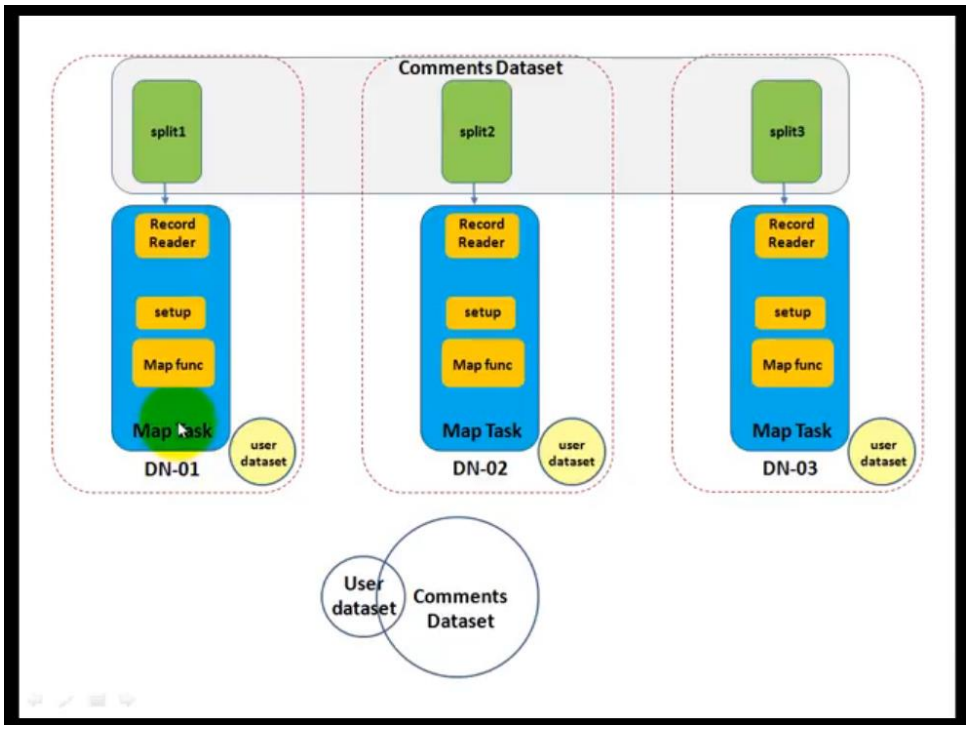
Large dataset read from split files and all smaller dataset must be copied to the memory of the map tasks

Memory is the limiting factor and that's the why all dataset must be small except for one dataset

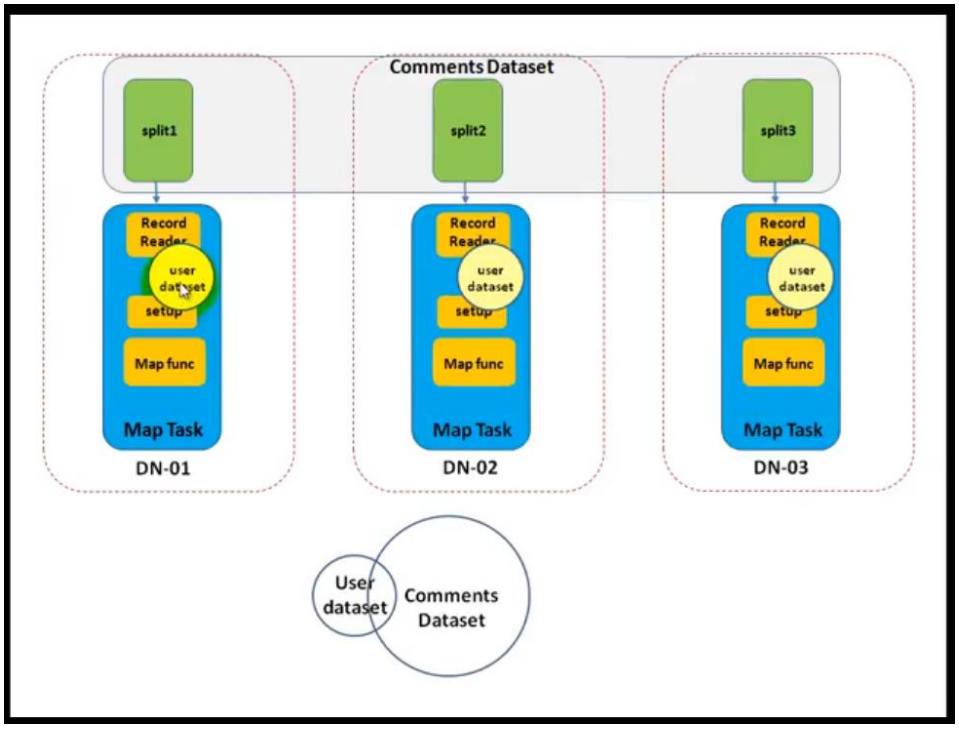
For stackoverflow problem we have smaller user dataset and large comment dataset hence we need to copy user dataset to the memory of the user dataset



First user dataset must be copied to the datanode where the map tasks are running



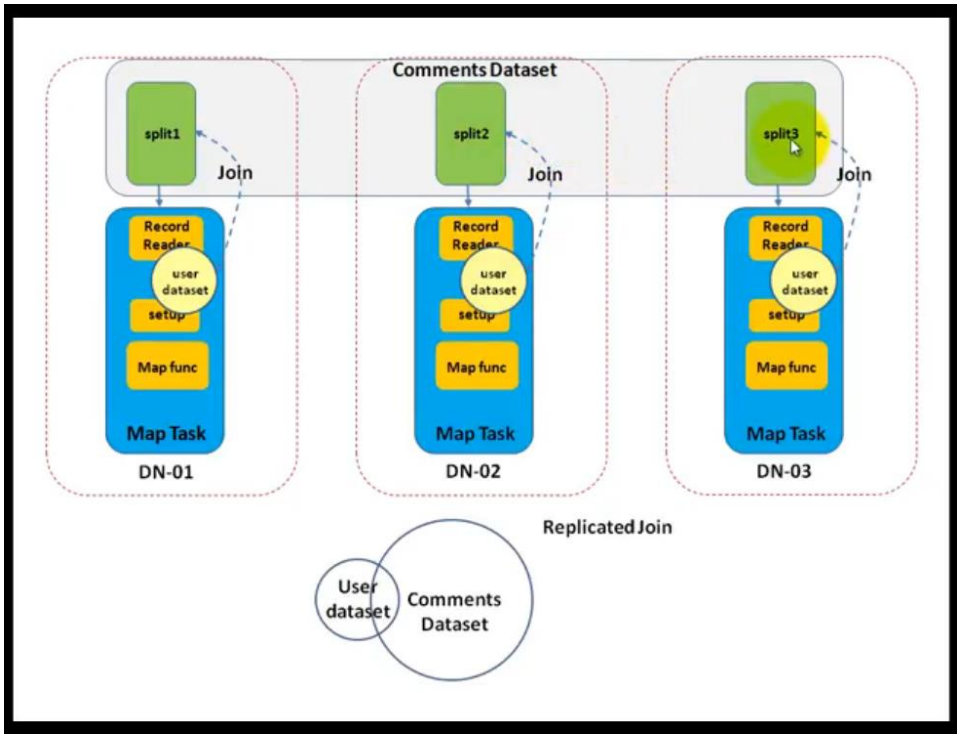
From there it has to be copied into memory of the map task



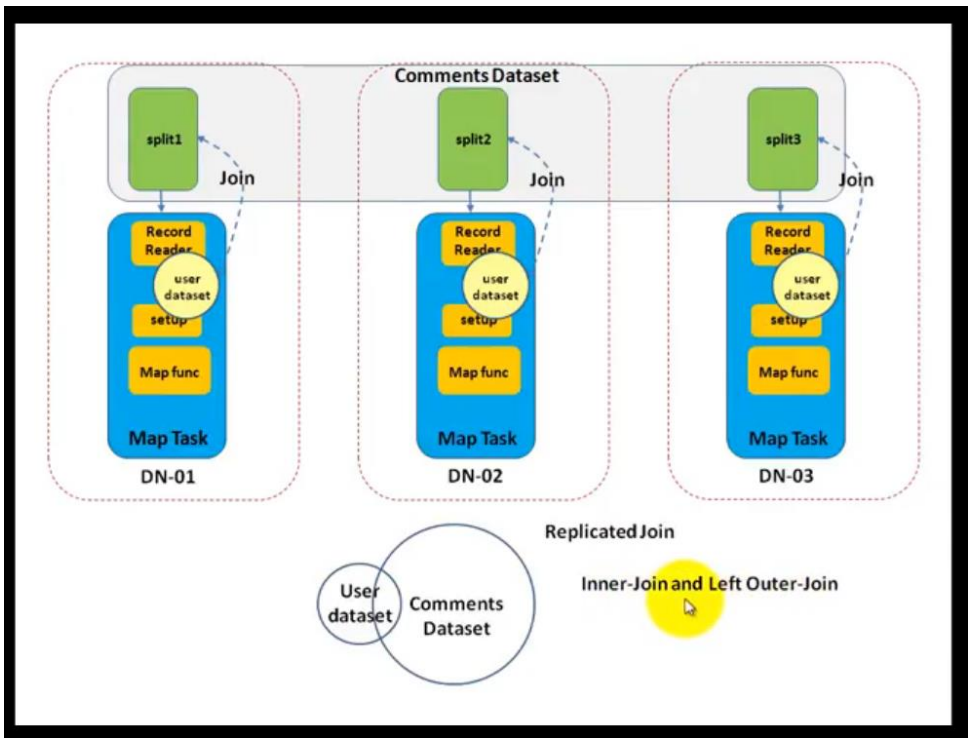
As a result each map tasks will have a copy of the user dataset

Since smaller dataset is replicated on each map tasks this join also called a Replicated join

Input split1 ,2 and 3 all will join with user dataset in each map tasks



Here only full user dataset available in each map tasks and part of the comment dataset available in the map task because of this reason only inner-join and left outer-join is possible with map side join



ReplicatedJoin.java | ReplicatedJoinMapper.java

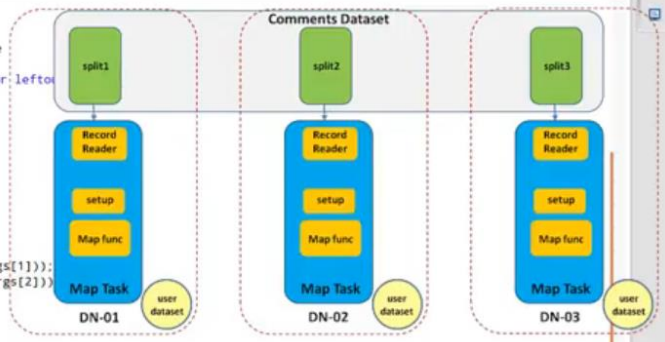
```
package com.hadoop.training.examples.replicatedjoin;

import org.apache.hadoop.conf.Configuration;

public class ReplicatedJoin {

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        String[] otherArgs = new GenericOptionsParser(conf, args)
        .getRemainingArgs();
        if (otherArgs.length != 4) {
            System.err
            .println("Usage: ReplicatedJoin <user data> <comment data> <out> [inner|leftouter]");
            System.exit(1);
        }

        String joinType = otherArgs[3];
        if (!(joinType.equalsIgnoreCase("inner") || joinType
                .equalsIgnoreCase("leftouter"))) {
            System.err.println("Join type not set to inner or leftouter");
            System.exit(2);
        }

        // Configure the join type
        Job job = new Job(conf, "Replicated Join");
        job.getConfiguration().set("join.type", joinType);
        job.setJarByClass(ReplicatedJoin.class);

        job.setMapperClass(ReplicatedJoinMapper.class);
        job.setNumReduceTasks(0);

        TextInputFormat.setInputPaths(job, new Path(otherArgs[1]));
```

ReplicatedJoin.java | ReplicatedJoinMapper.java

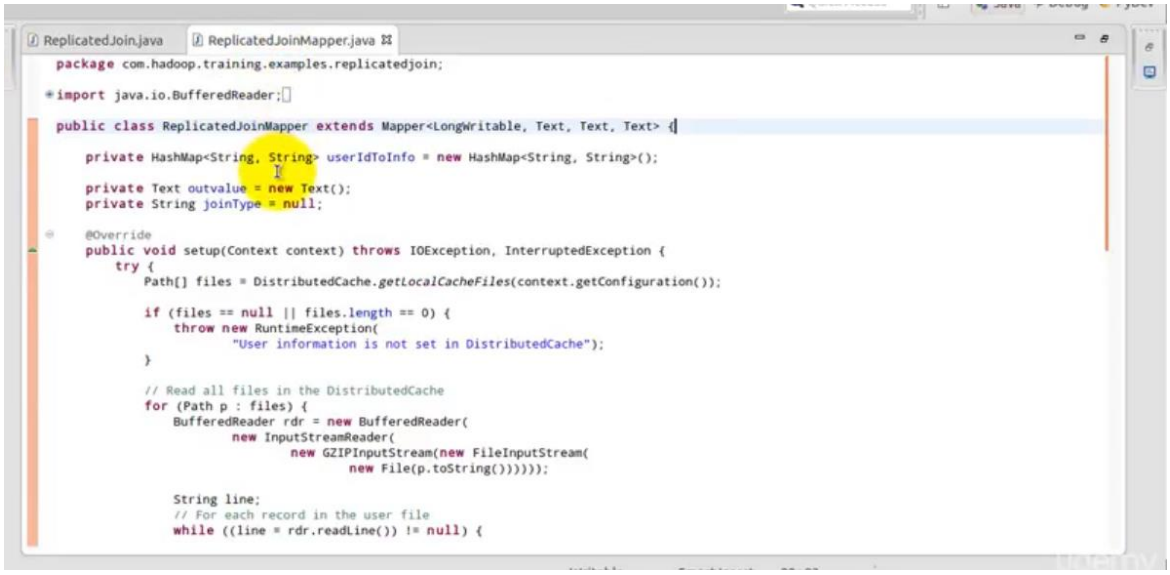
```
public class ReplicatedJoin {

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        String[] otherArgs = new GenericOptionsParser(conf, args)
        .getRemainingArgs();
        if (otherArgs.length != 4) {
            System.err
            .println("Usage: ReplicatedJoin <user data> <comment data> <out> [inner|leftouter]");
            System.exit(1);
        }

        String joinType = otherArgs[3];
        if (!(joinType.equalsIgnoreCase("inner") || joinType
                .equalsIgnoreCase("leftouter"))) {
            System.err.println("Join type not set to inner or leftouter");
            System.exit(2);
        }

        // Configure the join type
        Job job = new Job(conf, "Replicated Join");
        job.getConfiguration().set("join.type", joinType);
        job.setJarByClass(ReplicatedJoin.class);

        job.setMapperClass(ReplicatedJoinMapper.class);
        job.setNumReduceTasks(0);

        TextInputFormat.setInputPaths(job, new Path(otherArgs[1]));
        TextOutputFormat.setOutputPath(job, new Path(otherArgs[2]));

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

ReplicatedJoin.java | ReplicatedJoinMapper.java

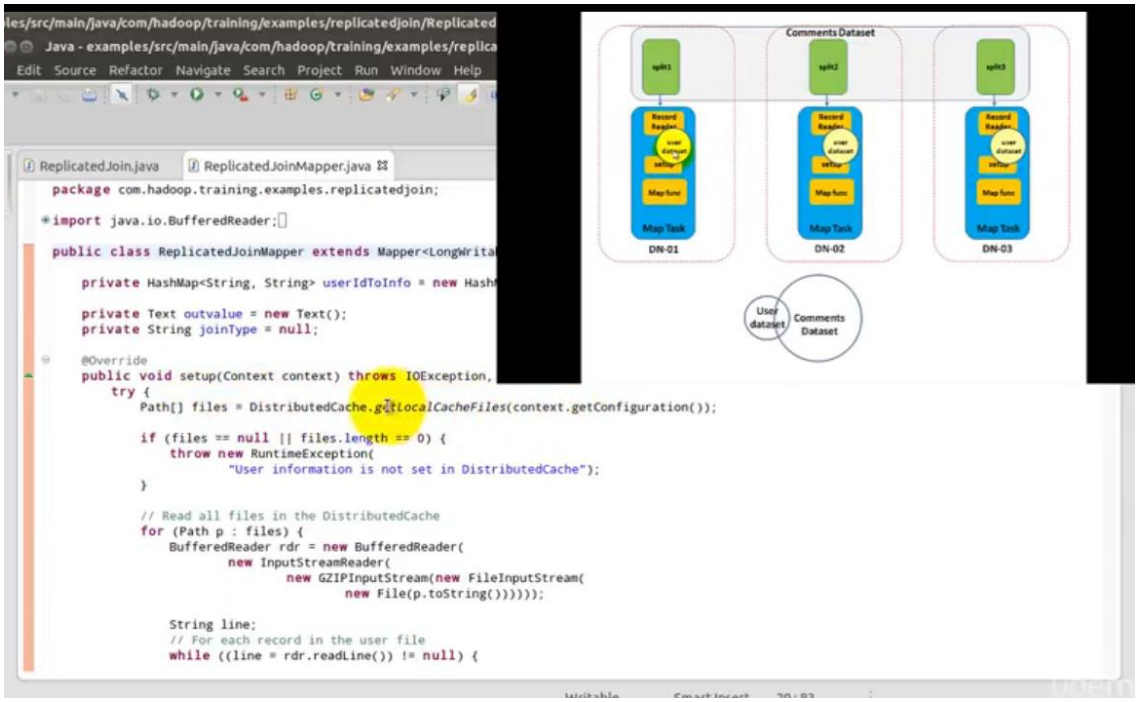
```
        }

        String joinType = otherArgs[3];
        if (!(joinType.equalsIgnoreCase("inner") || joinType
                .equalsIgnoreCase("leftouter"))) {
            System.err.println("Join type not set to inner or lefto
            System.exit(2);
        }

        // Configure the join type
        Job job = new Job(conf, "Replicated Join");
        job.getConfiguration().set("join.type", joinType);
        job.setJarByClass(ReplicatedJoin.class);

        job.setMapperClass(ReplicatedJoinMapper.class);
        job.setNumReduceTasks(0);

        TextInputFormat.setInputPaths(job, new Path(otherArgs[1]));
        TextOutputFormat.setOutputPath(job, new Path(otherArgs[2]));

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

        // Configure the DistributedCache

        DistributedCache.addCacheFile(new Path(otherArgs[0]).toUri(),
                job.getConfiguration());

        System.exit(job.waitForCompletion(true) ? 0 : 3);
    }
}
```



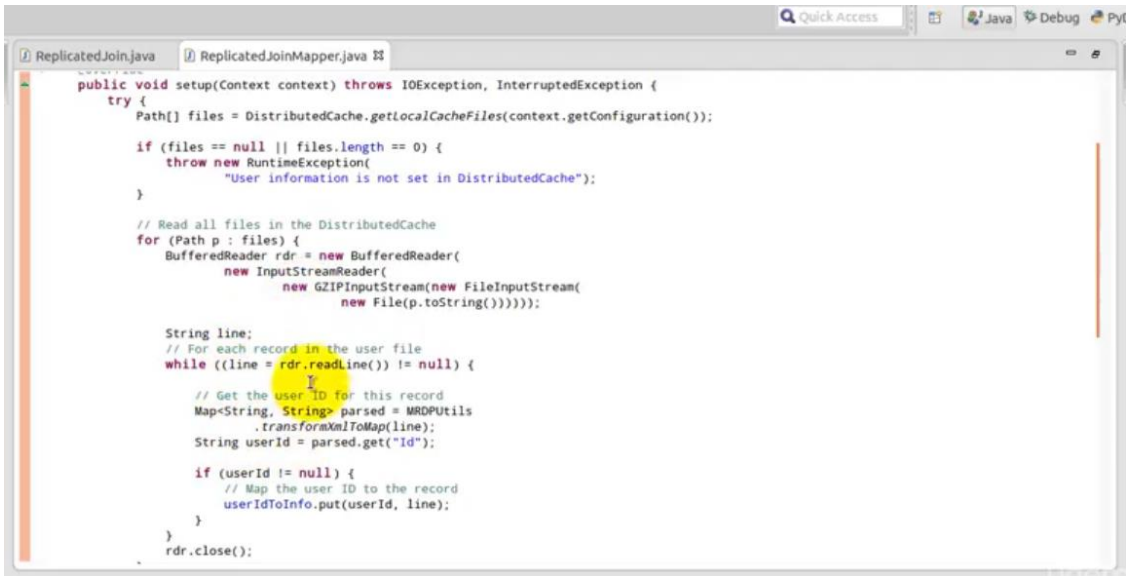
Using distributed cache we are copying user dataset to the data node where the map tasks are running



In the setup function we are reading data from distributed cache



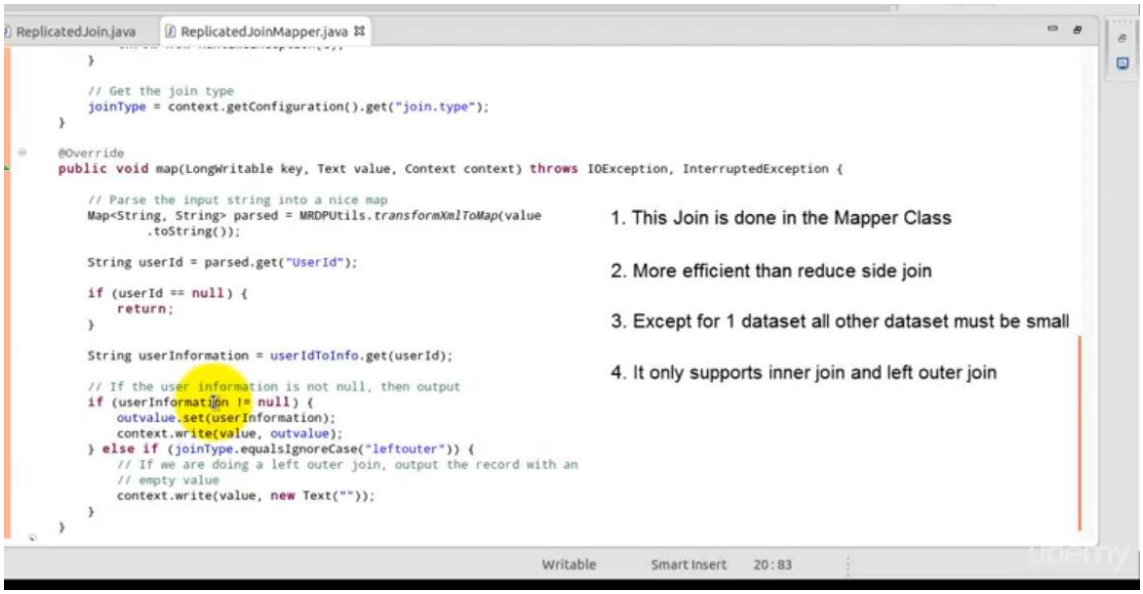
Distributed cache is used to copy small dataset from local file system to HDFS and vice versa



We are storing user data in map in this way all the user dataset will be copied into memory of mapper tasks

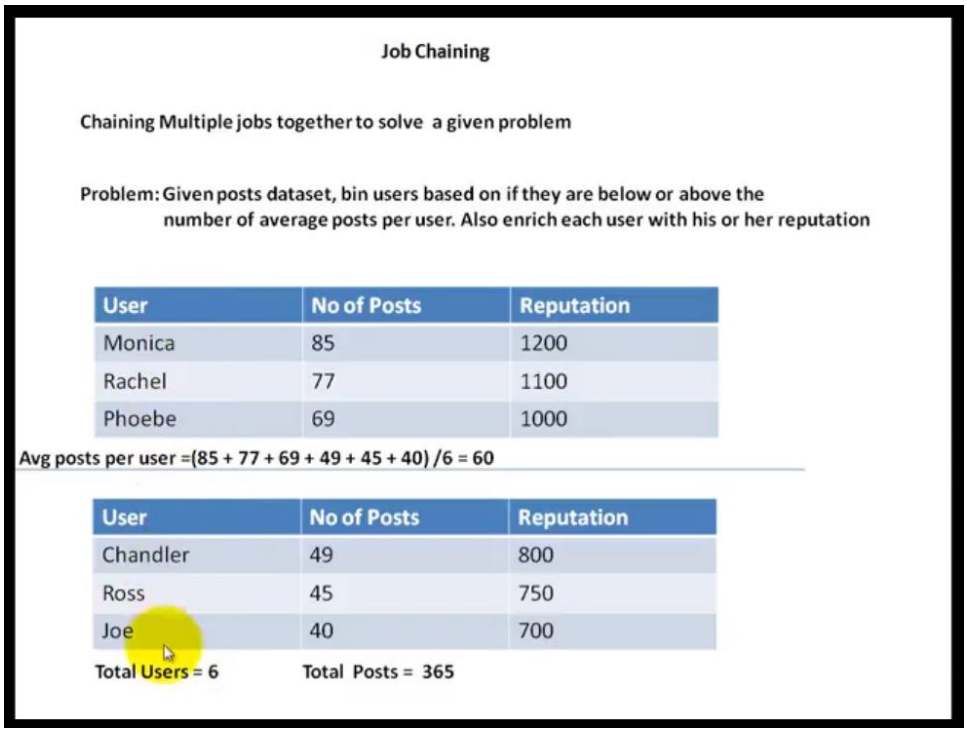
In map function user dataset is joined with the comment dataset,

Comment dataset is read through input split


ReplicatedJoin.java
ReplicatedJoinMapper.java
}
// Get the join type
joinType = context.getConfiguration().get("join.type");
}
@Override
public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
// Parse the input string into a nice map
Map<String, String> parsed = MRDPUtils.transformXmlToMap(value
.toString());
String userId = parsed.get("UserId");
if (userId == null) {
return;
}
String userInformation = userIdToInfo.get(userId);
// If the user information is not null, then output
if (userInformation != null) {
outvalue.set(userInformation);
context.write(value, outvalue);
} else if (joinType.equalsIgnoreCase("leftouter")) {
// If we are doing a left outer join, output the record with an
// empty value
context.write(value, new Text(""));
}
}
1. This Join is done in the Mapper Class
2. More efficient than reduce side join
3. Except for 1 dataset all other dataset must be small
4. It only supports inner join and left outer join
Writable
Smart Insert
20 : 83

Job Chaining Pattern

First Problem



Job Chaining

Chaining Multiple jobs together to solve a given problem

Problem: Given posts dataset, bin users based on if they are below or above the number of average posts per user. Also enrich each user with his or her reputation

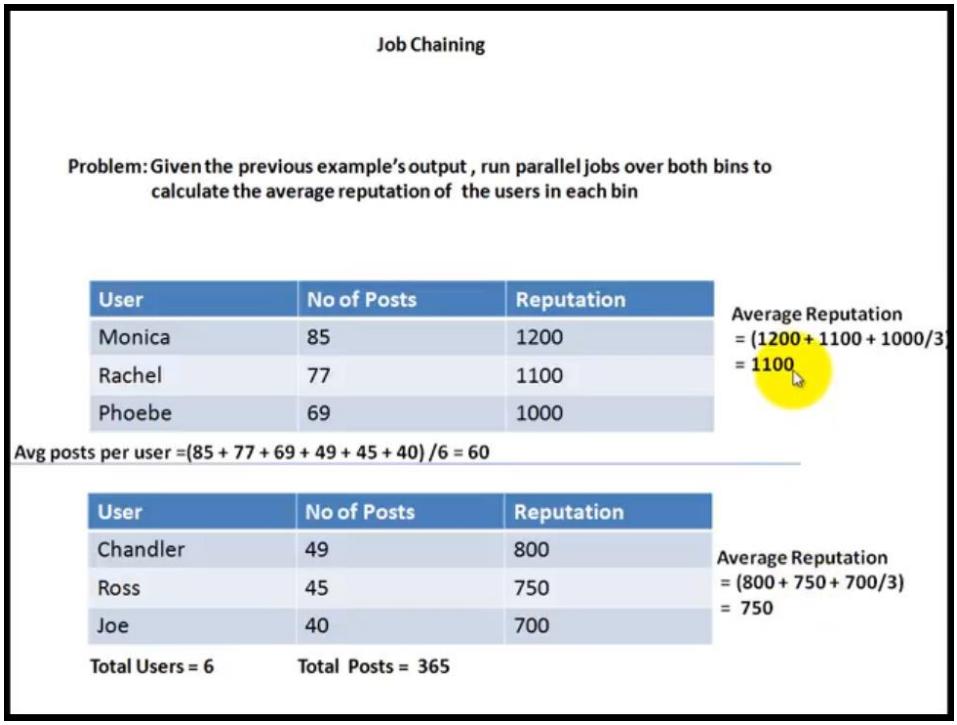
User	No of Posts	Reputation
Monica	85	1200
Rachel	77	1100
Phoebe	69	1000

Avg posts per user =(85 + 77 + 69 + 49 + 45 + 40) /6 = 60

User	No of Posts	Reputation
Chandler	49	800
Ross	45	750
Joe	40	700

Total Users = 6 Total Posts = 365

Second problem



Job Chaining

Problem: Given the previous example's output , run parallel jobs over both bins to calculate the average reputation of the users in each bin

User	No of Posts	Reputation
Monica	85	1200
Rachel	77	1100
Phoebe	69	1000

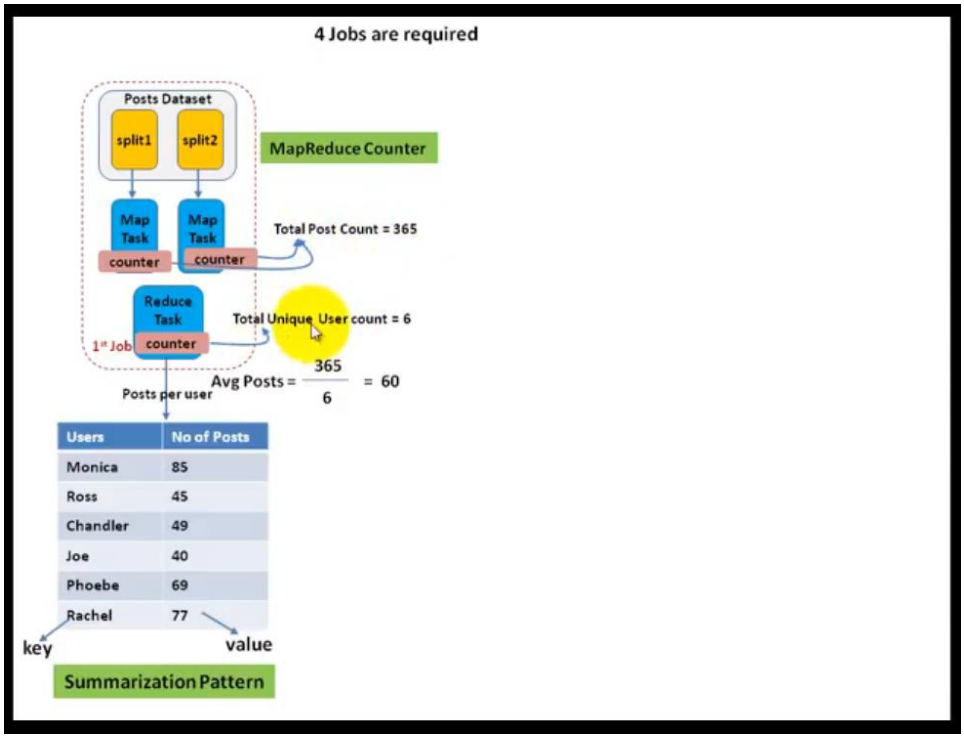
Average Reputation = (1200 + 1100 + 1000/3) = 1100

Avg posts per user =(85 + 77 + 69 + 49 + 45 + 40) /6 = 60

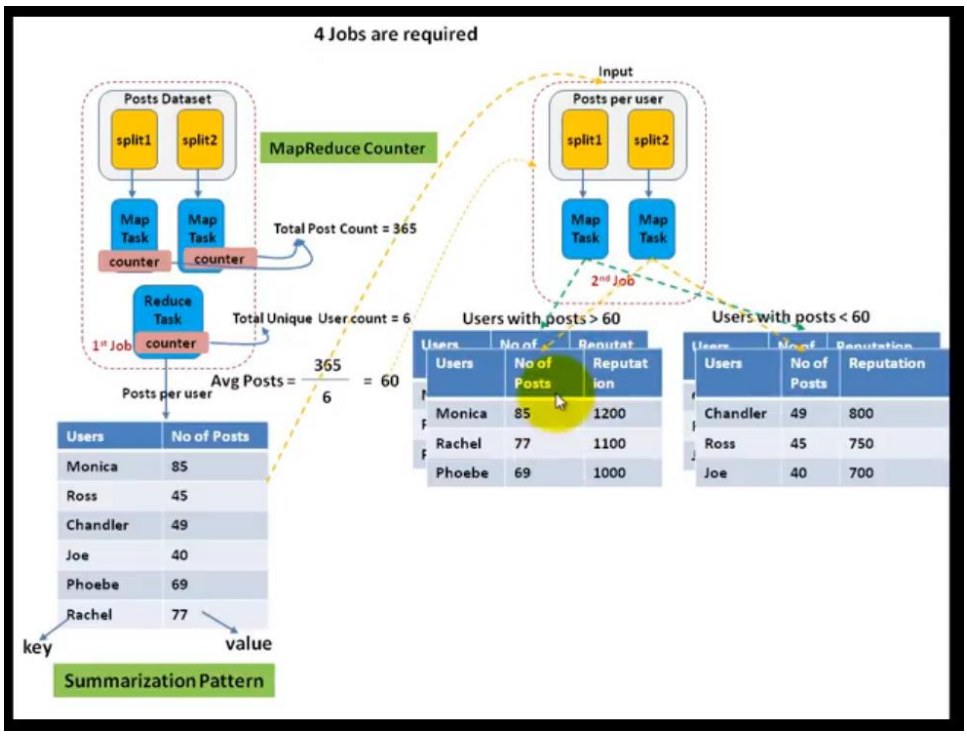
User	No of Posts	Reputation
Chandler	49	800
Ross	45	750
Joe	40	700

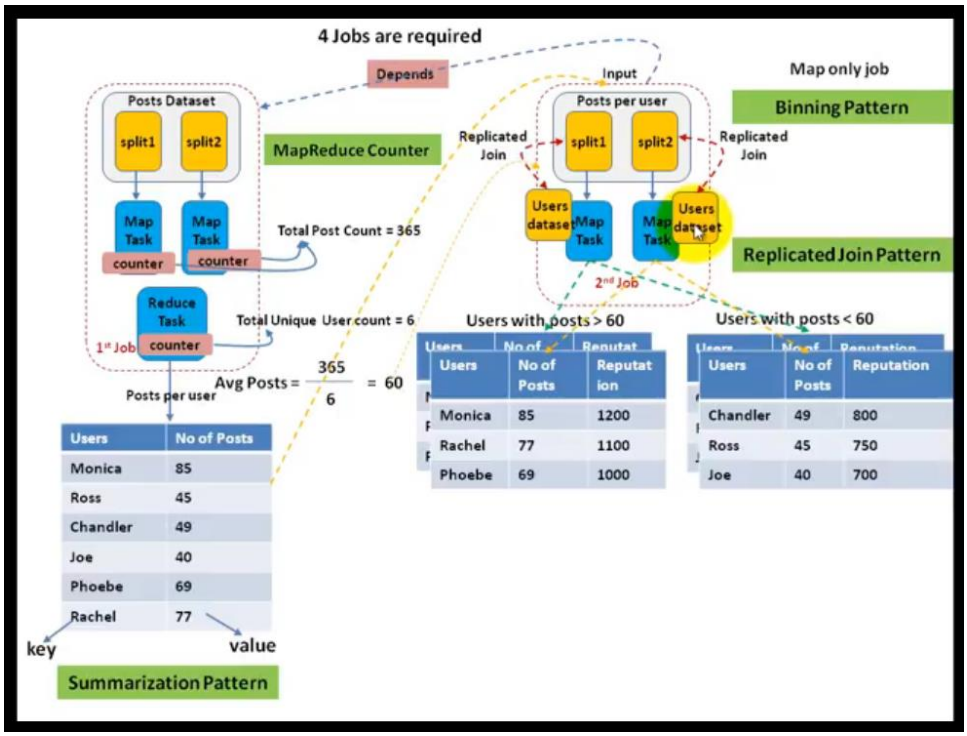
Average Reputation = (800 + 750 + 700/3) = 750

Total Users = 6 Total Posts = 365

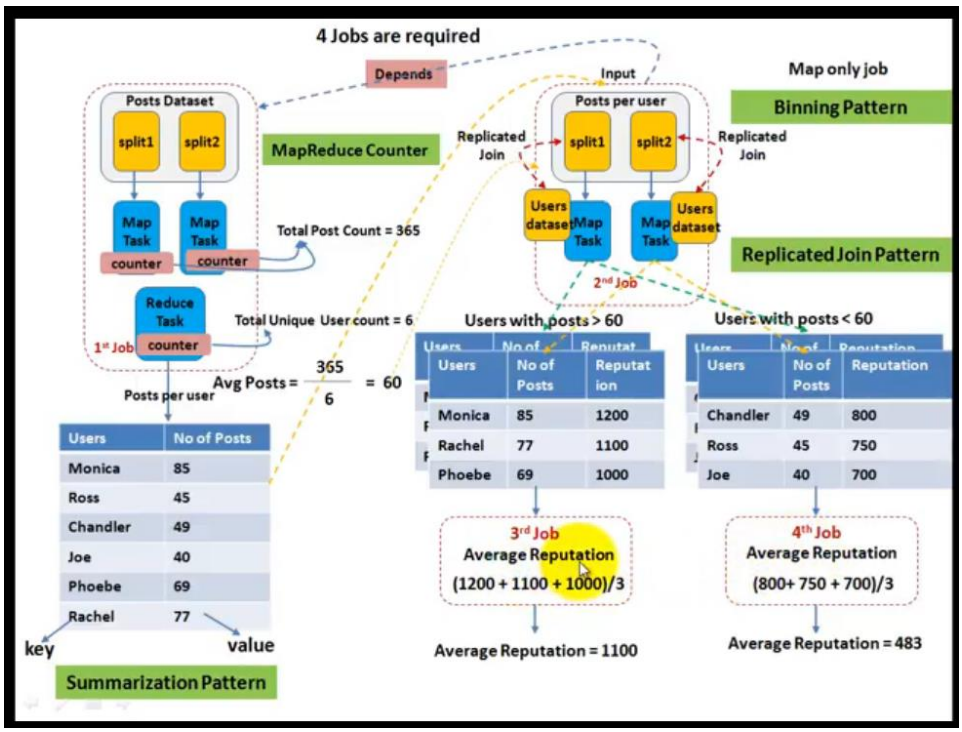


Counter in mapper will give total post count and counter in reducer side give unique number of user count





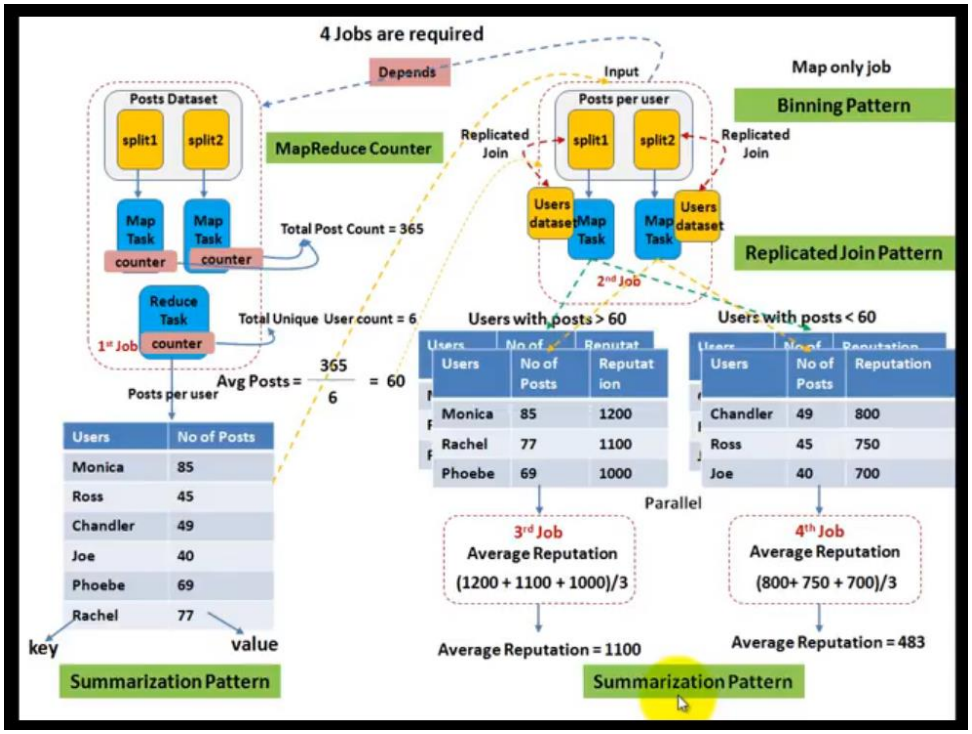
In second job we are using binning pattern and Replicated join pattern

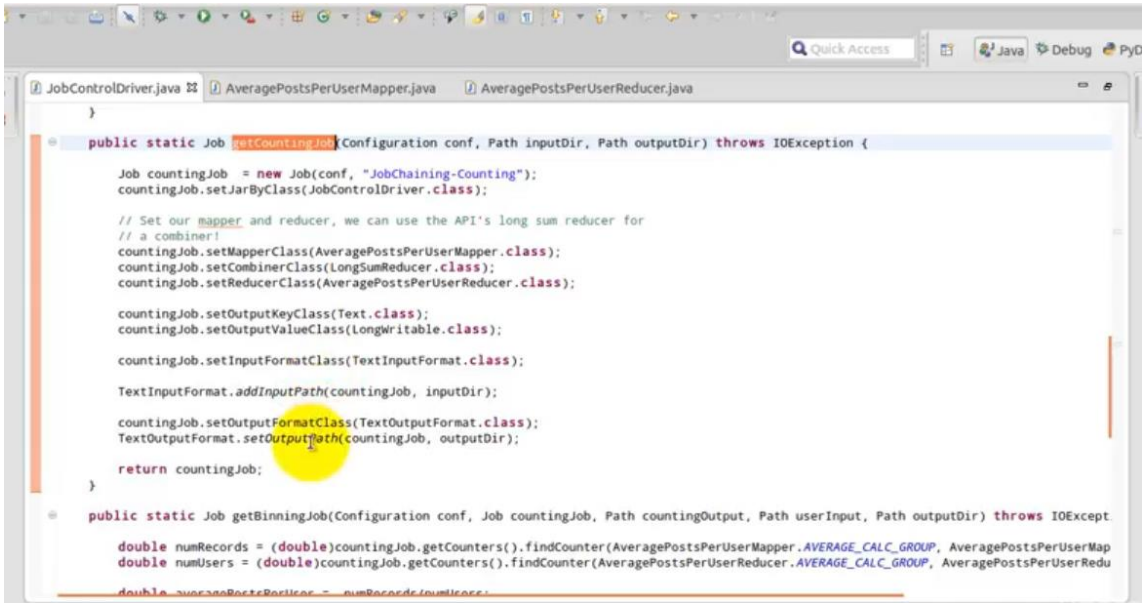


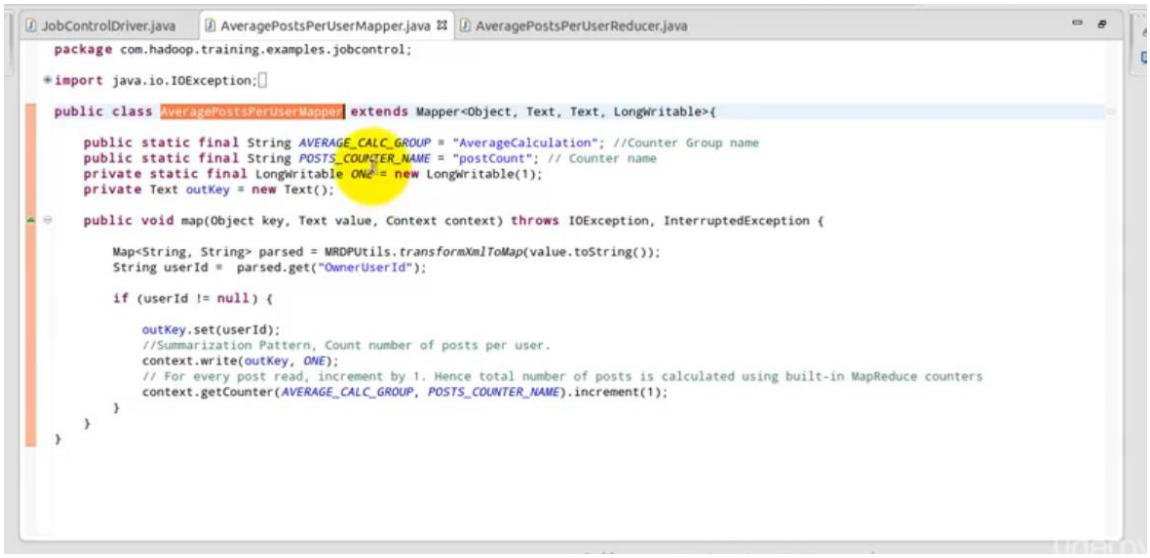
3rd and 4th job compute average reputation of each bin

So 1st bin in input to 3rd job and 2 bin is input to 4th job

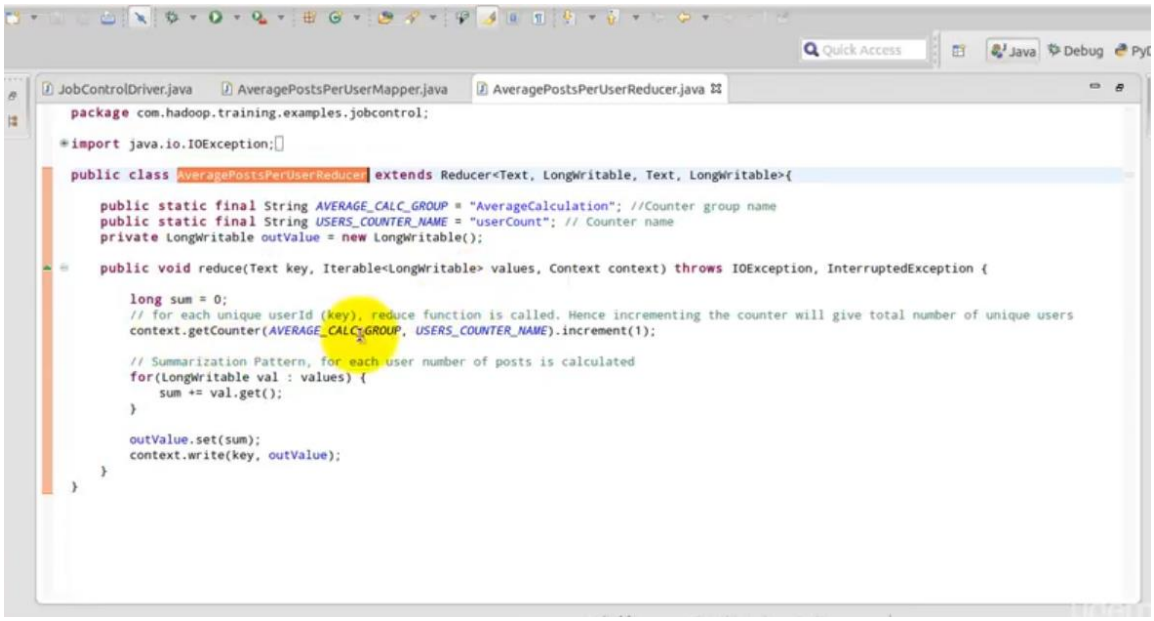
3rd and 4th job are independent so they can run parallel


4 Jobs are required
Depends
Input
Map only job
Binning Pattern
Posts Dataset
split1
split2
MapReduce Counter
Replicated Join
Posts per user
Replicated Join
Map Task
counter
Total Post Count = 365
Users dataset
Replicated Join Pattern
2nd Job
Reduce Task
1st Job
Total Unique User count = 6
Users with posts > 60
Users with posts < 60
Avg Posts = 365/6 = 60
Posts per user
Users
No of Posts
Reputation
Monica 85 1200
Rachel 77 1100
Phoebe 69 1000
Chandler 49 800
Ross 45 750
Joe 40 700
Monica 85
Ross 45
Chandler 49
Joe 40
Phoebe 69
Rachel 77
key
value
Parallel
3rd Job
Average Reputation
(1200 + 1100 + 1000)/3
4th Job
Average Reputation
(800+ 750 + 700)/3
Average Reputation = 1100
Average Reputation = 483
Summarization Pattern
Summarization Pattern


Quick Access
Java
Debug
JobControlDriver.java
AveragePostsPerUserMapper.java
AveragePostsPerUserReducer.java
public static Job getCountingJob(Configuration conf, Path inputDir, Path outputDir) throws IOException {
Job countingJob = new Job(conf, "JobChaining-Counting");
countingJob.setJarByClass(JobControlDriver.class);
// Set our mapper and reducer, we can use the API's long sum reducer for
// a combiner!
countingJob.setMapperClass(AveragePostsPerUserMapper.class);
countingJob.setCombinerClass(LongSumReducer.class);
countingJob.setReducerClass(AveragePostsPerUserReducer.class);
countingJob.setOutputKeyClass(Text.class);
countingJob.setOutputValueClass(LongWritable.class);
countingJob.setInputFormatClass(TextInputFormat.class);
TextInputFormat.addInputPath(countingJob, inputDir);
countingJob.setOutputFormatClass(TextOutputFormat.class);
TextOutputFormat.setOutputPath(countingJob, outputDir);
return countingJob;
public static Job getBinningJob(Configuration conf, Job countingJob, Path countingOutput, Path userInput, Path outputDir) throws IOExcept
double numRecords = (double)countingJob.getCounters().findCounter(AveragePostsPerUserMapper.AVERAGE_CALC_GROUP, AveragePostsPerUserMap
double numUsers = (double)countingJob.getCounters().findCounter(AveragePostsPerUserReducer.AVERAGE_CALC_GROUP, AveragePostsPerUserRedu



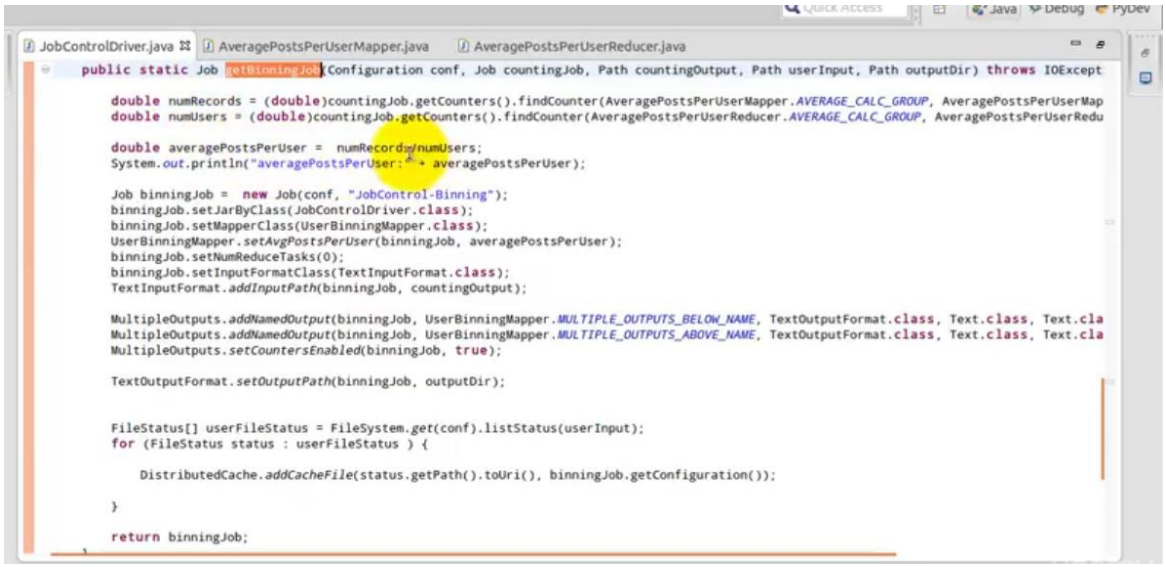
```java
package com.hadoop.training.examples.jobcontrol;

import java.io.IOException;

public class AveragePostsPerUserMapper extends Mapper<Object, Text, Text, LongWritable>{

    public static final String AVERAGE_CALC_GROUP = "AverageCalculation"; //Counter Group name
    public static final String POSTS_COUNTER_NAME = "postCount"; // Counter name
    private static final LongWritable ONE = new LongWritable(1);
    private Text outKey = new Text();

    public void map(Object key, Text value, Context context) throws IOException, InterruptedException {

        Map<String, String> parsed = MRDPUtils.transformXmlToMap(value.toString());
        String userId =  parsed.get("OwnerUserId");

        if (userId != null) {

            outKey.set(userId);
            //Summarization Pattern, Count number of posts per user.
            context.write(outKey, ONE);
            // For every post read, increment by 1. Hence total number of posts is calculated using built-in MapReduce counters
            context.getCounter(AVERAGE_CALC_GROUP, POSTS_COUNTER_NAME).increment(1);
        }
    }
}
```

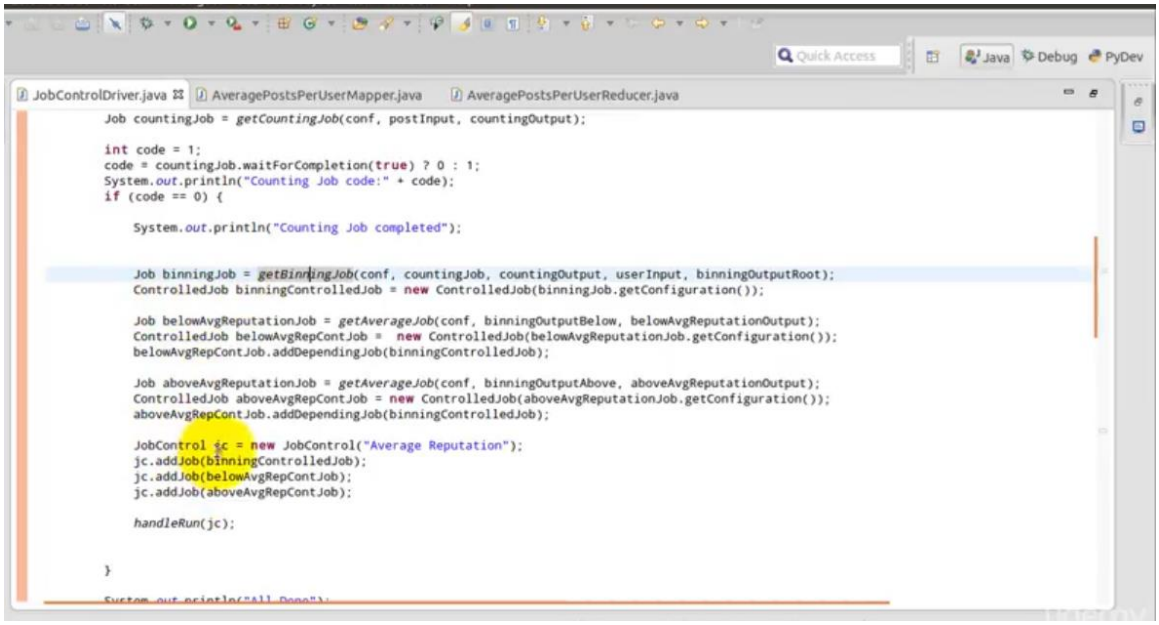


```java
package com.hadoop.training.examples.jobcontrol;

import java.io.IOException;

public class AveragePostsPerUserReducer extends Reducer<Text, LongWritable, Text, LongWritable>{

    public static final String AVERAGE_CALC_GROUP = "AverageCalculation"; //Counter group name
    public static final String USERS_COUNTER_NAME = "userCount"; // Counter name
    private LongWritable outValue = new LongWritable();

    public void reduce(Text key, Iterable<LongWritable> values, Context context) throws IOException, InterruptedException {

        long sum = 0;
        // for each unique userId (key), reduce function is called. Hence incrementing the counter will give total number of unique users
        context.getCounter(AVERAGE_CALC_GROUP, USERS_COUNTER_NAME).increment(1);

        // Summarization Pattern, for each user number of posts is calculated
        for(LongWritable val : values) {
            sum += val.get();
        }

        outValue.set(sum);
        context.write(key, outValue);
    }
}
```

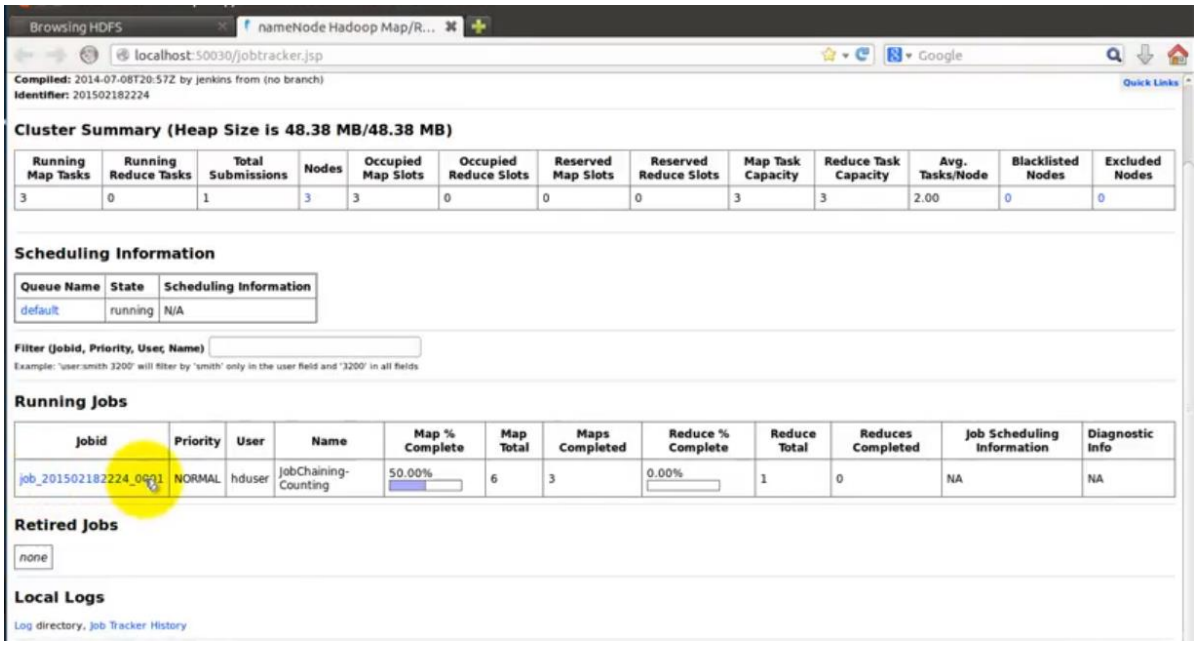
```java
            .println("Usage: JobControlDriver <posts> <users> <out>");
            System.exit(2);
        }

        Path postInput = new Path(otherArgs[0]);
        Path userInput = new Path(otherArgs[1]);
        Path countingOutput = new Path(otherArgs[2] + "_count");
        Path binningOutputRoot = new Path(otherArgs[2] + "_bins");
        Path binningOutputBelow = new Path(binningOutputRoot + "/" + UserBinningMapper.MULTIPLE_OUTPUTS_BELOW_NAME);
        Path binningOutputAbove = new Path(binningOutputRoot + "/" + UserBinningMapper.MULTIPLE_OUTPUTS_ABOVE_NAME);
        Path aboveAvgReputationOutput = new Path(otherArgs[2] + "/" + UserBinningMapper.MULTIPLE_OUTPUTS_ABOVE_NAME);
        Path belowAvgReputationOutput = new Path(otherArgs[2] + "/" + UserBinningMapper.MULTIPLE_OUTPUTS_BELOW_NAME);

        Job countingJob = getCountingJob(conf, postInput, countingOutput);

        int code = 1;
        code = countingJob.waitForCompletion(true) ? 0 : 1;
        System.out.println("Counting Job code:" + code);
        if (code == 0) {

            System.out.println("Counting Job completed");


            Job binningJob = getBinningJob(conf, countingJob, countingOutput, userInput, binningOutputRoot);
            ControlledJob binningControlledJob = new ControlledJob(binningJob.getConfiguration());

            Job belowAvgReputationJob = getAverageJob(conf, binningOutputBelow, belowAvgReputationOutput);
            ControlledJob belowAvgRepContJob =  new ControlledJob(belowAvgReputationJob.getConfiguration());
            belowAvgRepContJob.addDependingJob(binningControlledJob);

            Job aboveAvgReputationJob = getAverageJob(conf, binningOutputAbove, aboveAvgReputationOutput);
```

Second job

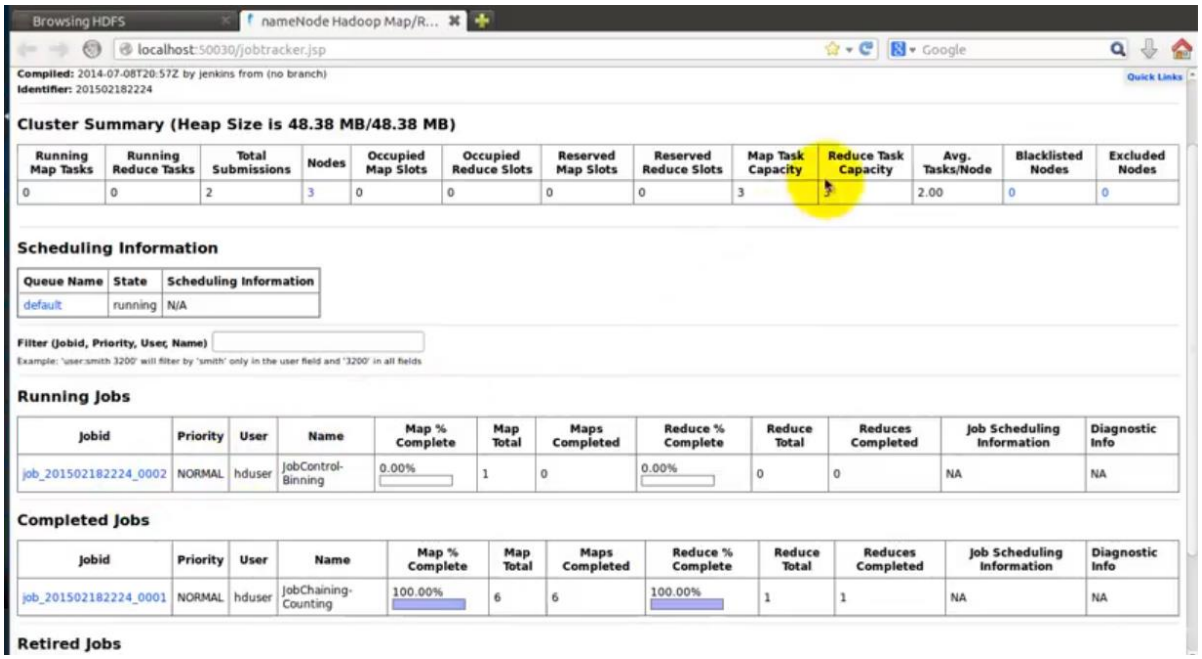




Since 3rd and 4th job dependent on 2nd job second job is added as depending job

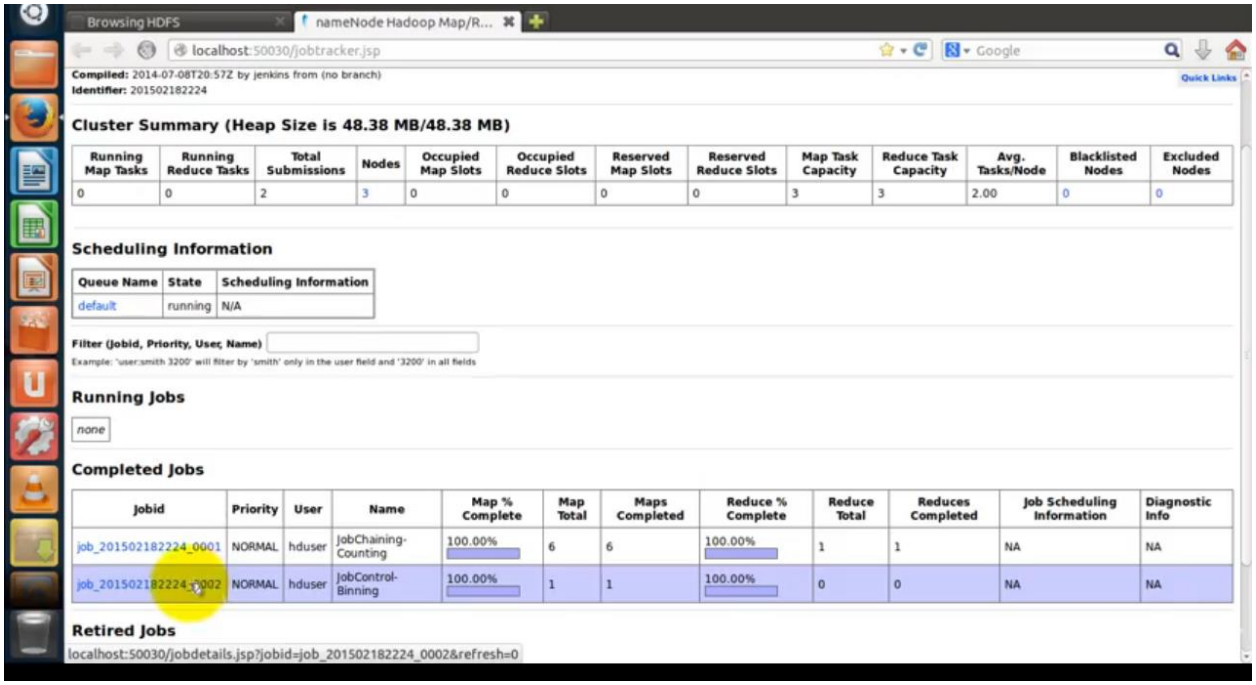


First job is running

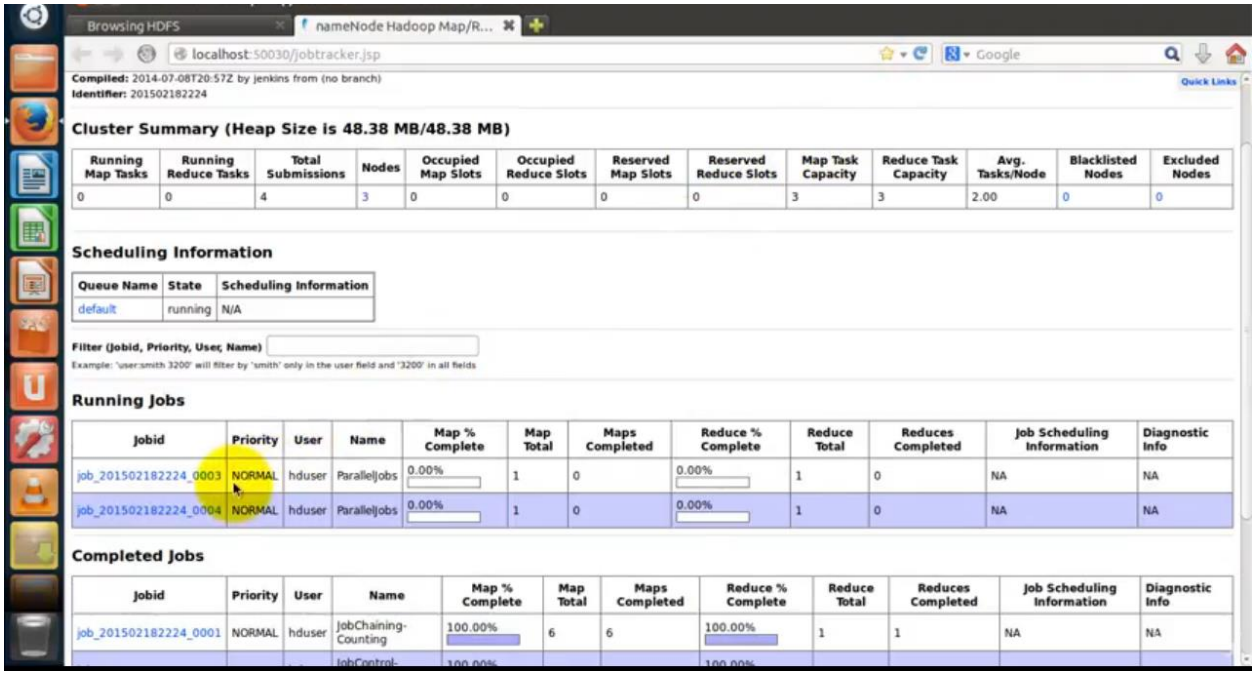


After completion of first job second job is running

Now second job is completed successfully

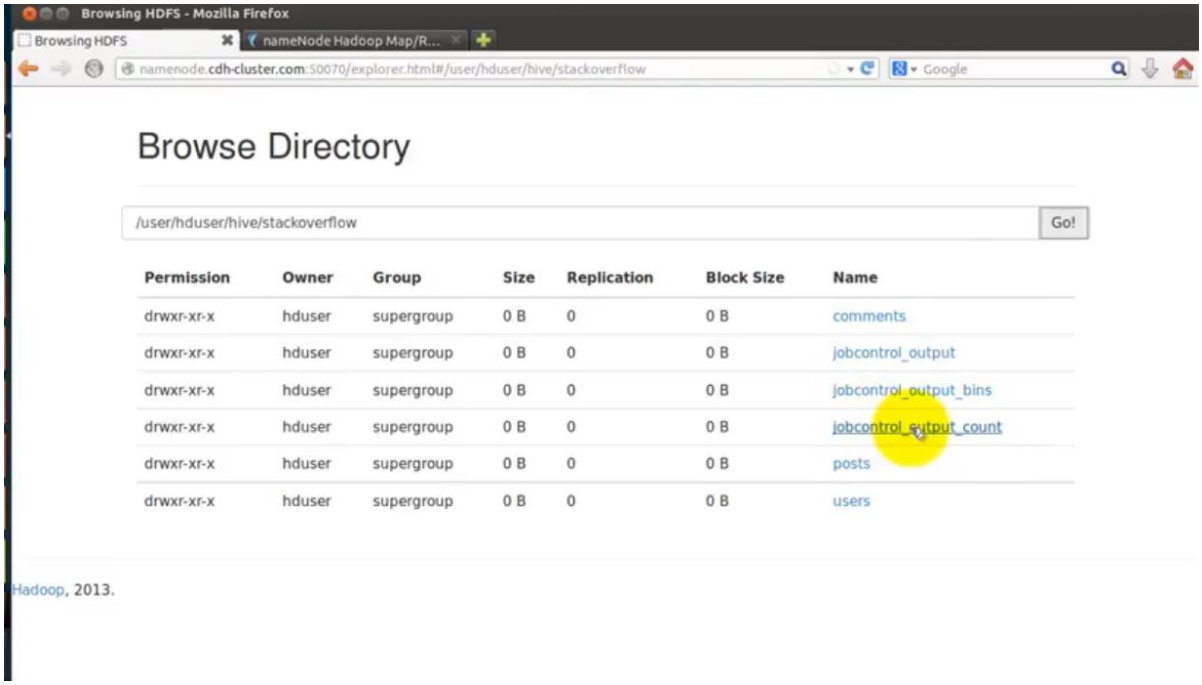


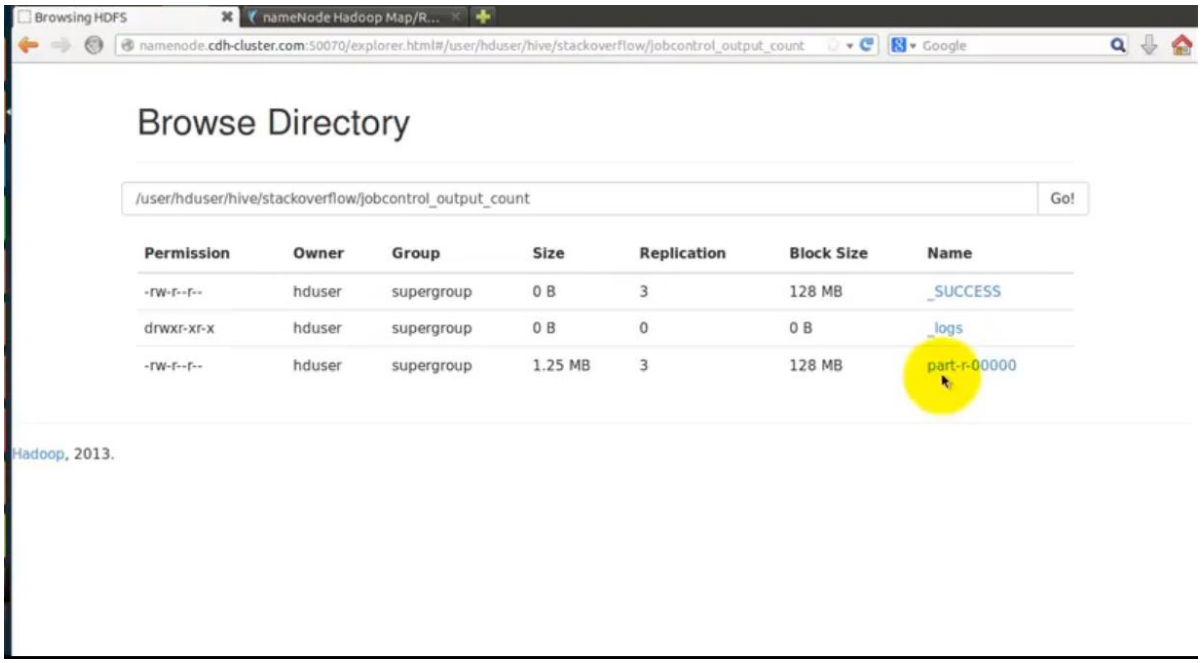
After completion of second job third and fourth job running simultaneously



Check output

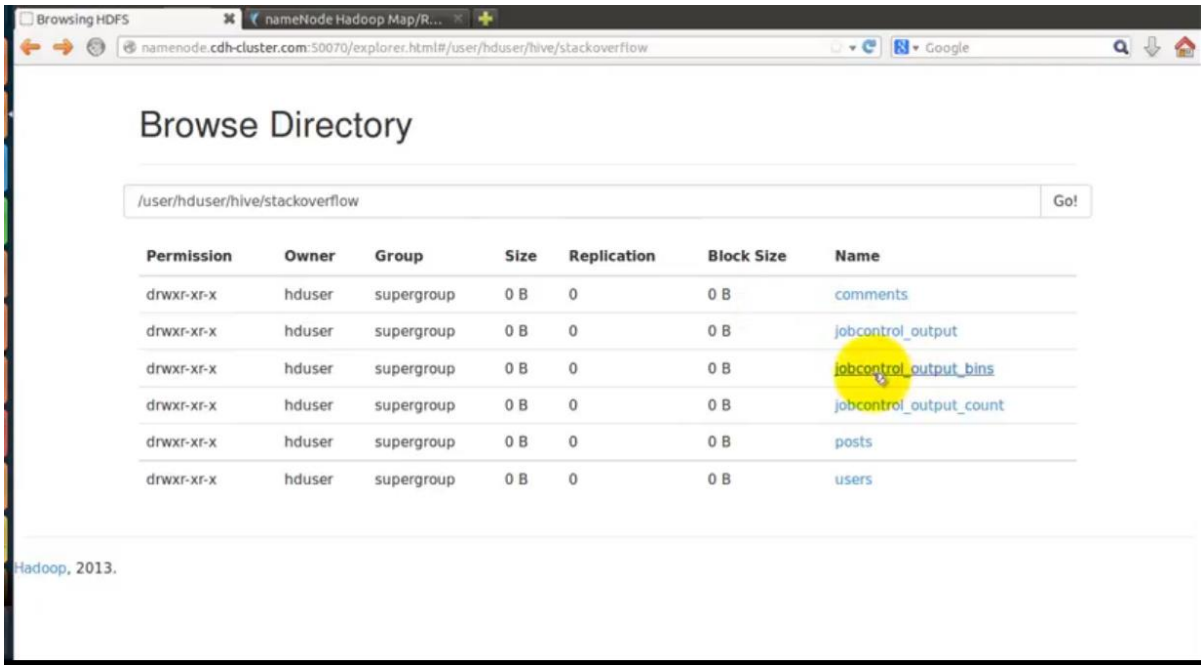
First job output

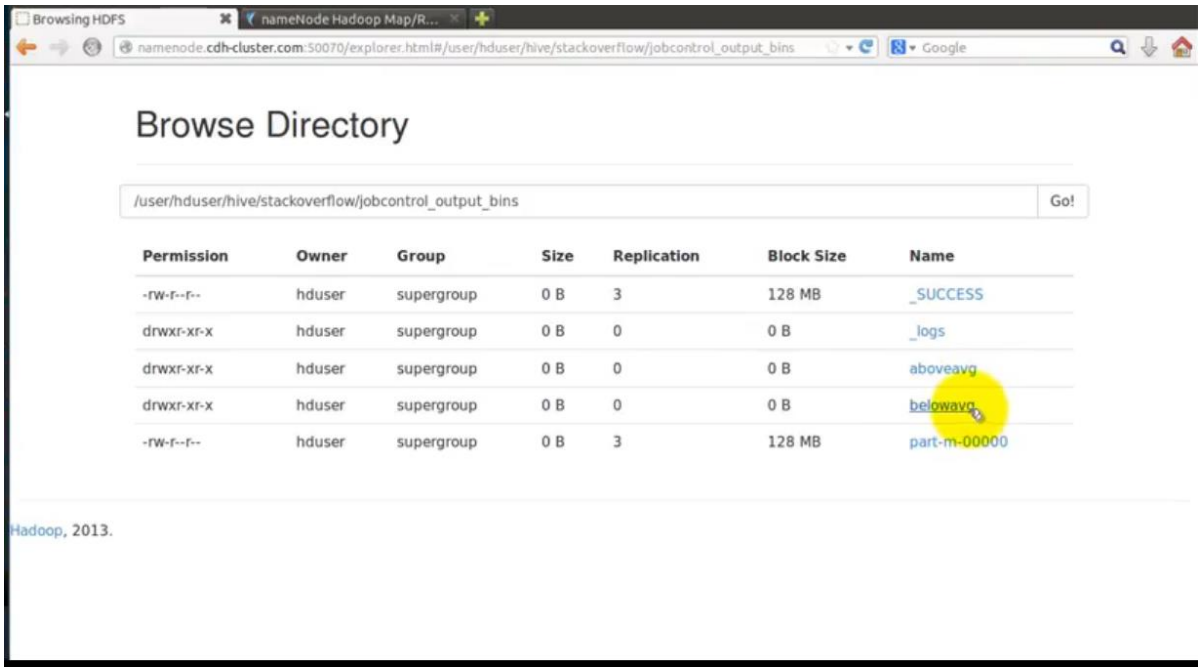




It contains total number of posts per user

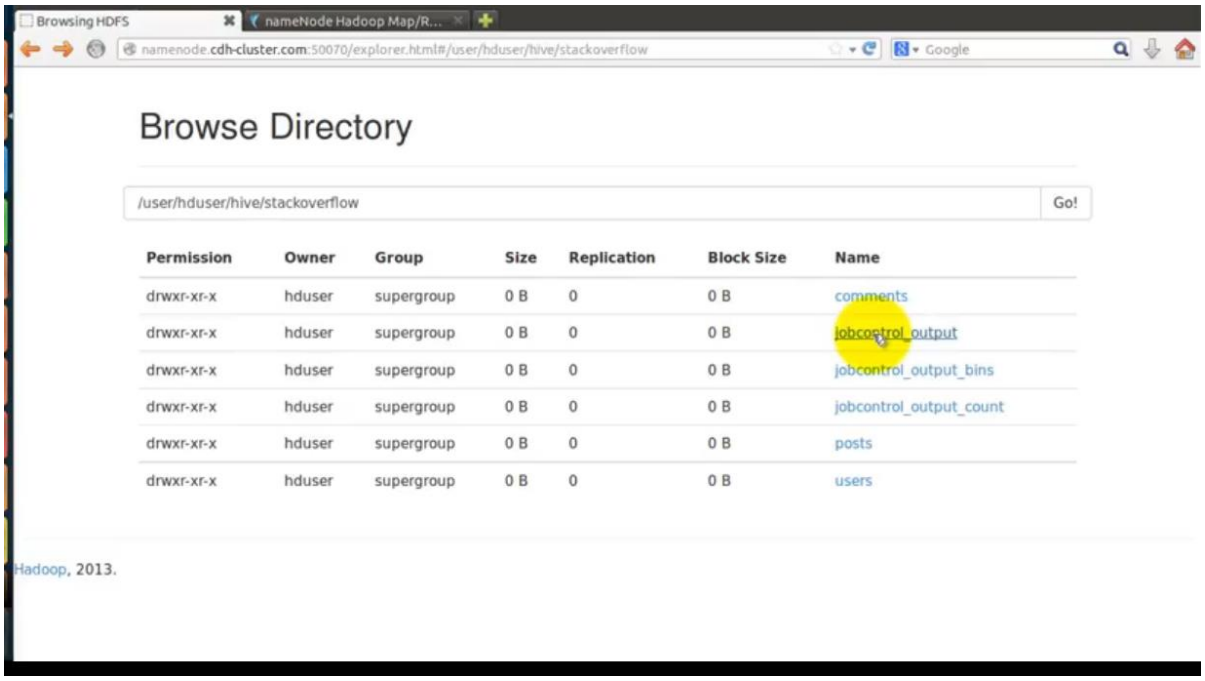
Second job output folder



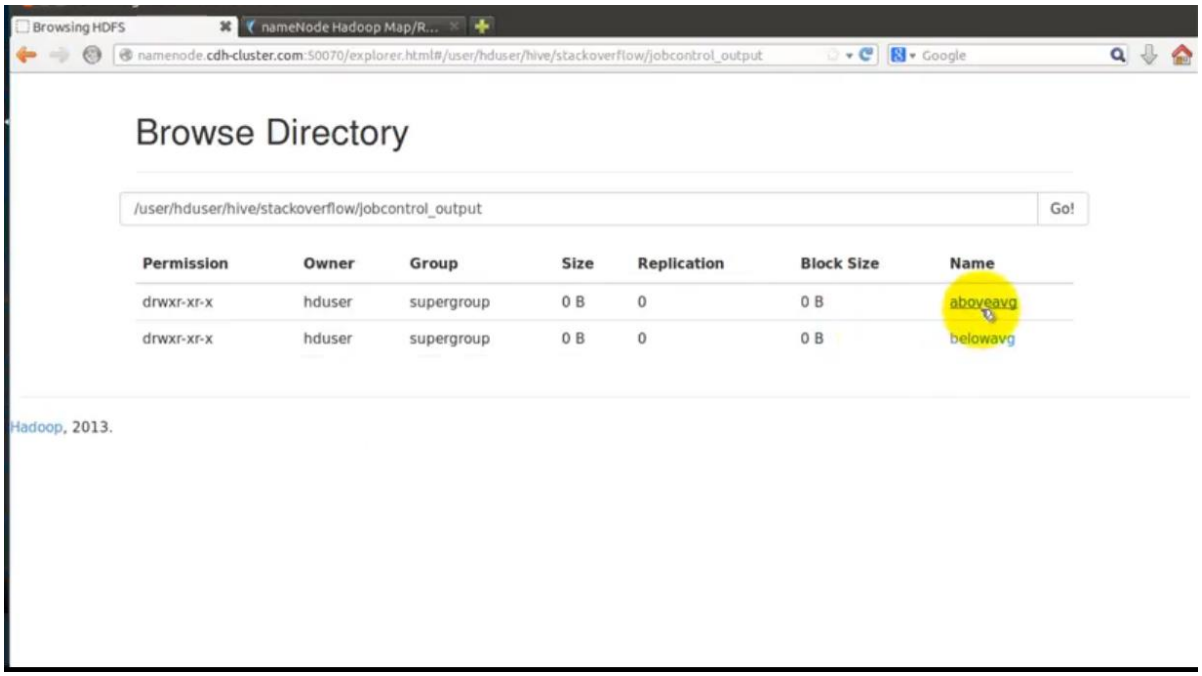


Two bins are created from the second job

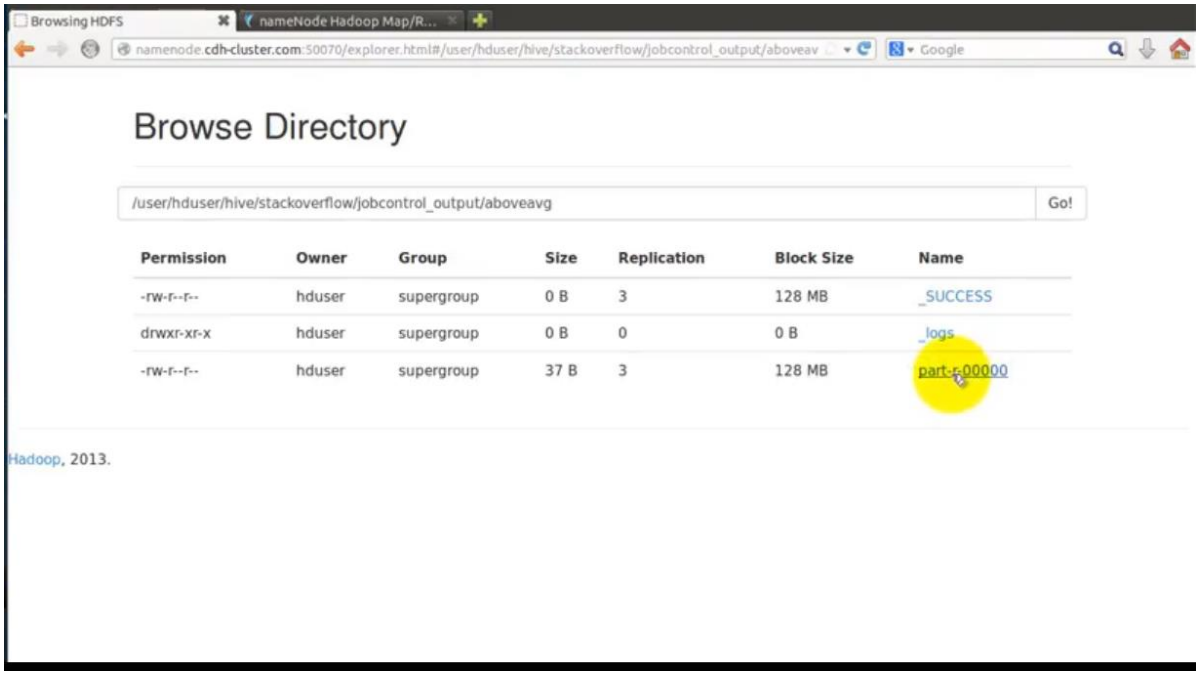
Final output



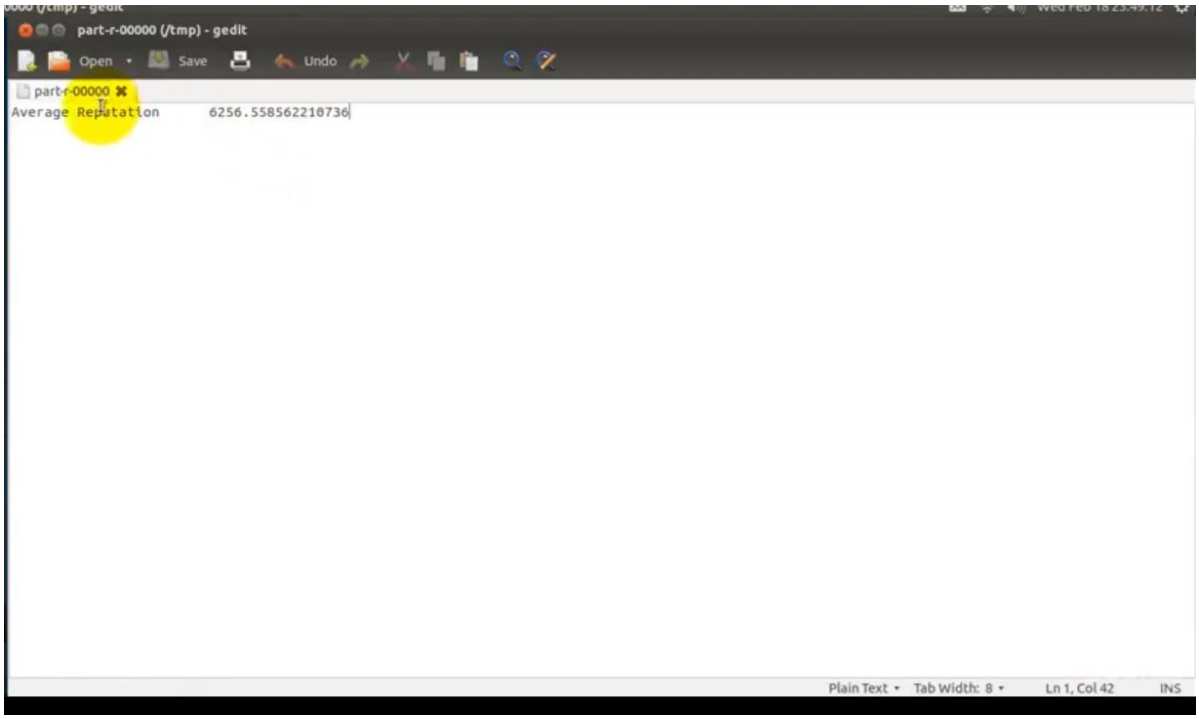
It contains average reputation of all the users in each bin



Hadoop, 2013.



Hadoop, 2013.



Average reputation of all the users in above average bin