



Problems with typical distributed system

1. Huge dependency on network and huge bandwidth demands
2. Scaling up and down is not a smooth process
3. Partial failures are difficult to handle
4. A lot of processing power is spent on transporting data
5. Data synchronization is required during exchange

This is where Hadoop comes in

© Copyright Zeppelin Software Solutions Pvt. Ltd. All rights reserved.

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 2013 IntelliPaat.com All rights reserved

What is Hadoop

IntelliPaat

- The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using a simple programming models
- In more simplistic terms, Hadoop is a framework that facilitates functioning of several machines together to achieve the goal of analyzing large sets of data
- Google created its own distributed computing framework and published papers about the same. Hadoop was developed on the basis of papers released by Google
- Core Hadoop consists of 2 core components:
 - The Hadoop Distributed File System (HDFS)
 - MapReduce
- A set of machines running HDFS and Mapreduce is known as Hadoop Cluster
 - Individual machines are known as nodes
 - A cluster can have as many as 1 node to several thousand nodes

© Copyright 2013 Intellipaat.com All rights reserved

WordCountJob :

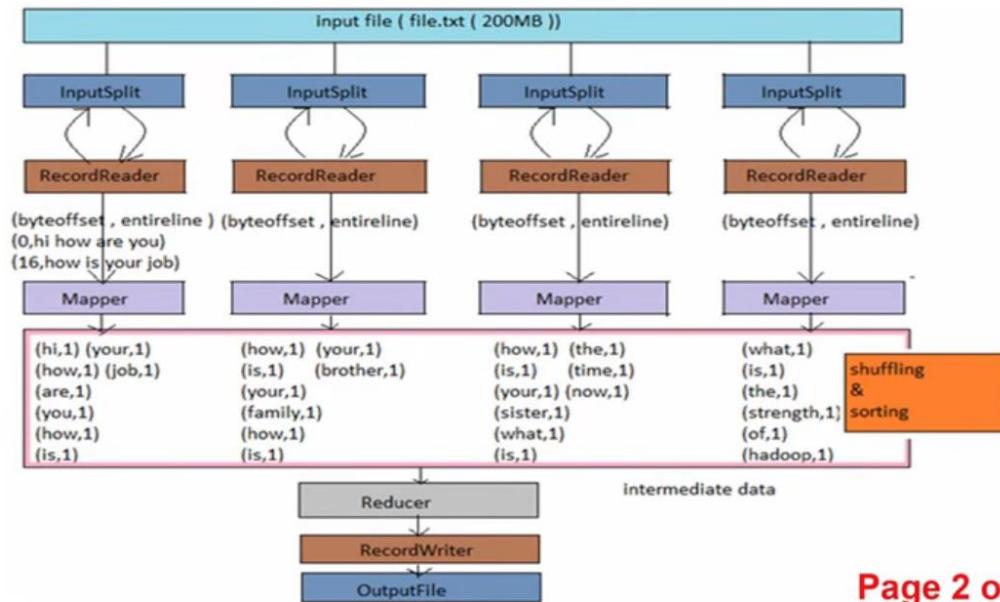
MapReduce Flow Chart

file.txt (200MB)	
hi how are you	64MB
how is your job	64MB
how is your family	64MB
how is your brother	64MB
how is your sister	64MB
what is the time now	8MB
what is the strength of hadoop	8MB

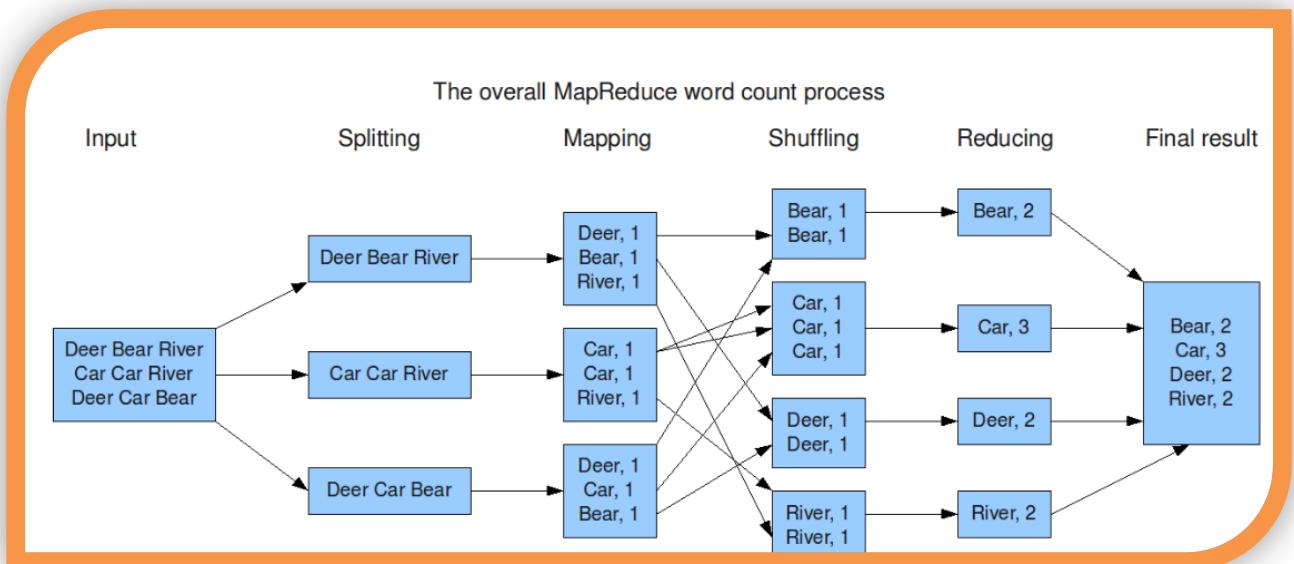
Input File Formats :

1. TextInputFormat
2. KeyValueTextInputFormat
3. SequenceFileInputFormat

MapReduce Flow Chart :



Page 2 of 5

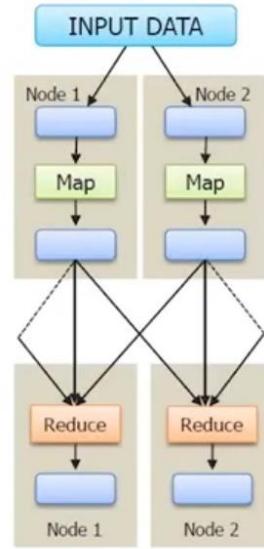


MapReduce Job Submission Flow

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



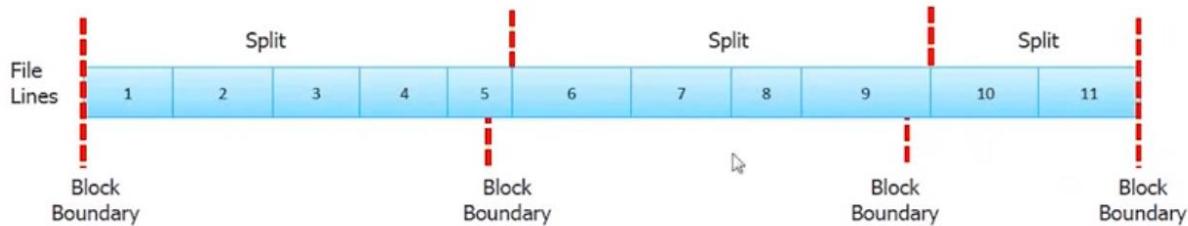
<http://www.edureka.co/comprehensive-mapreduce>



Input Split and HDFS Blocks

Relation Between Input Splits and HDFS Blocks

edureka!

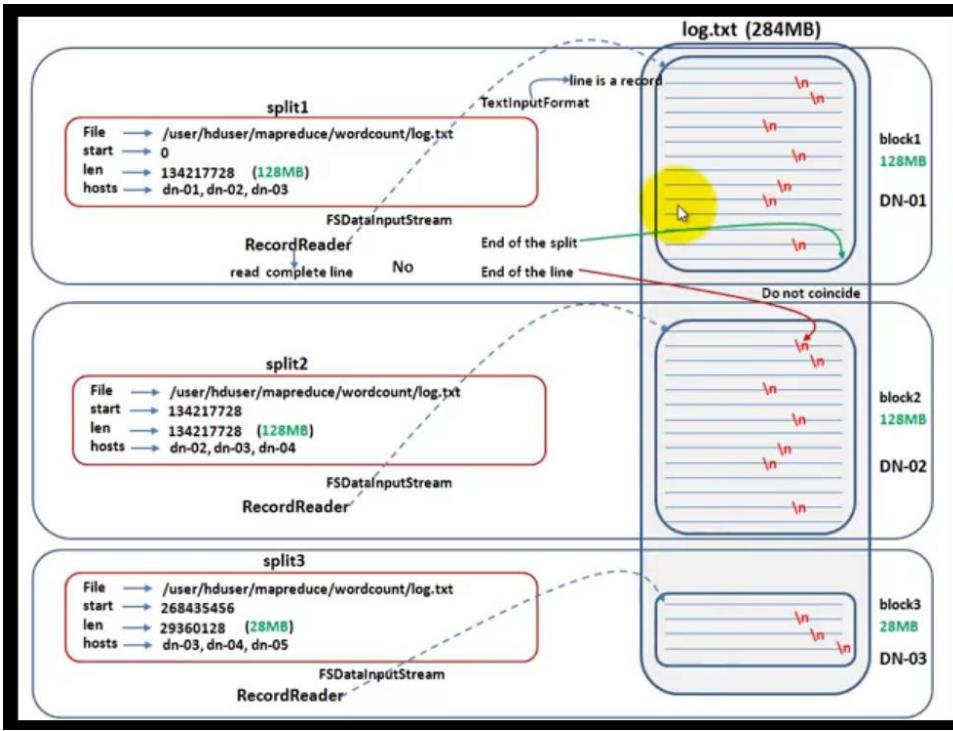


- Logical records do not fit neatly into the HDFS blocks.
- Logical records are lines that cross the boundary of the blocks.
- First split contains line 5 although it spans across blocks.

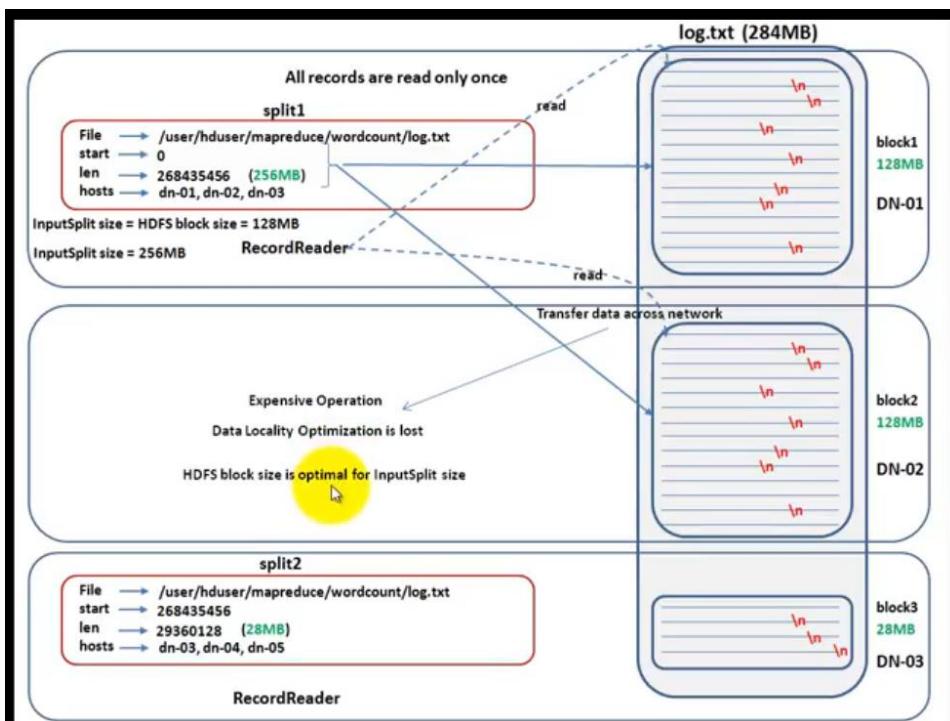
In case of record 5 and 9 (line 5 and 9) records are split in two blocks so record reader 1 will read the entire 5th line using FSDataInputStream via RPC and process the whole record. FSDataInputStream has all the information regarding file like file size, block location etc.

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

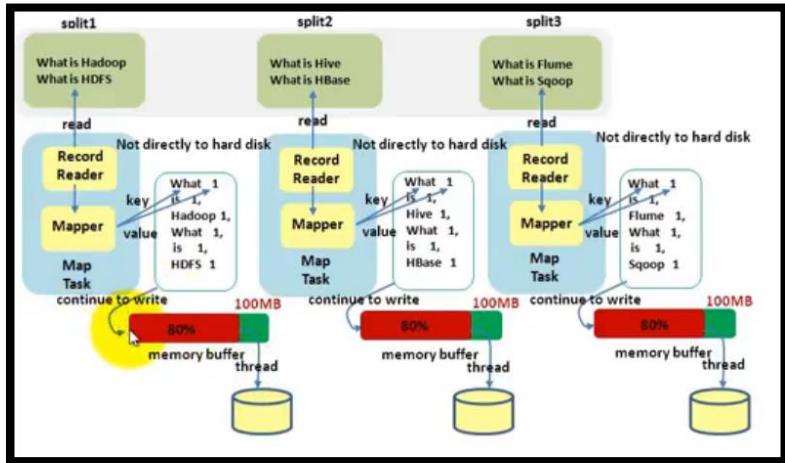


If split size change to 256 MB and block size is 128 MB

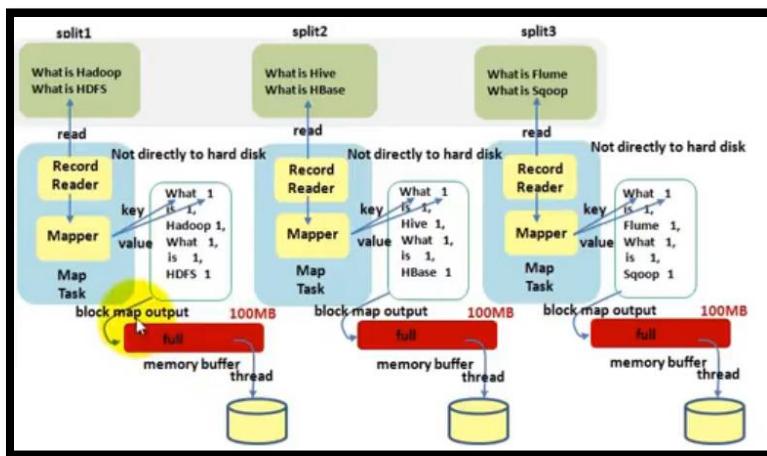


Mapper Reducer Shuffling Sorting

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

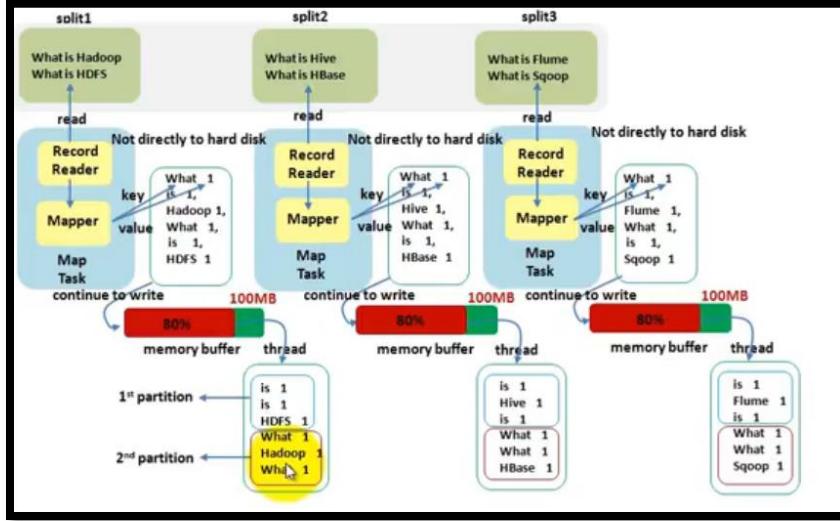


Map block if buffer is full till the time it is spilled to disk

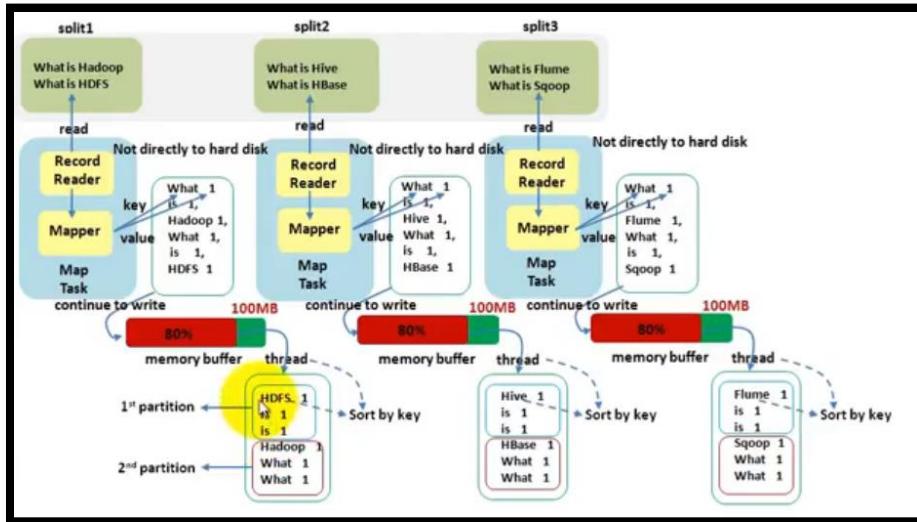


If we used partition in our mapreduce program, first background thread creates partition before writing onto the disk

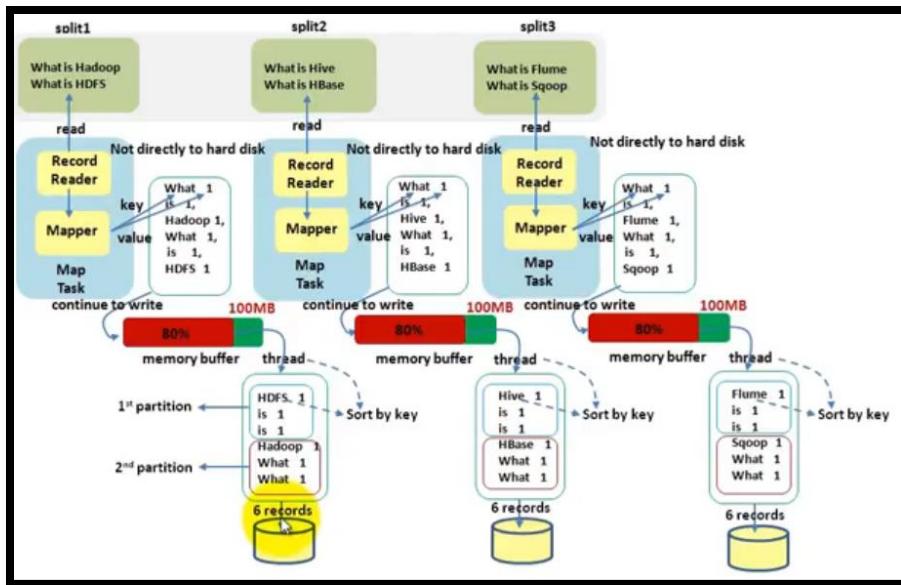
Here 2 partition created as we set 2 reduce task in program



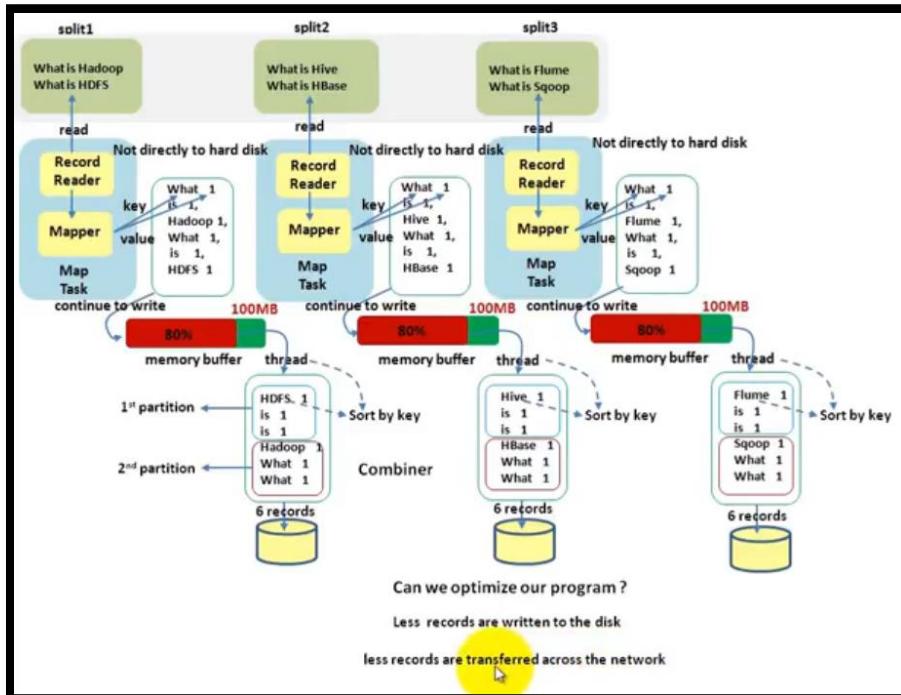
Background thread sort all the records by key



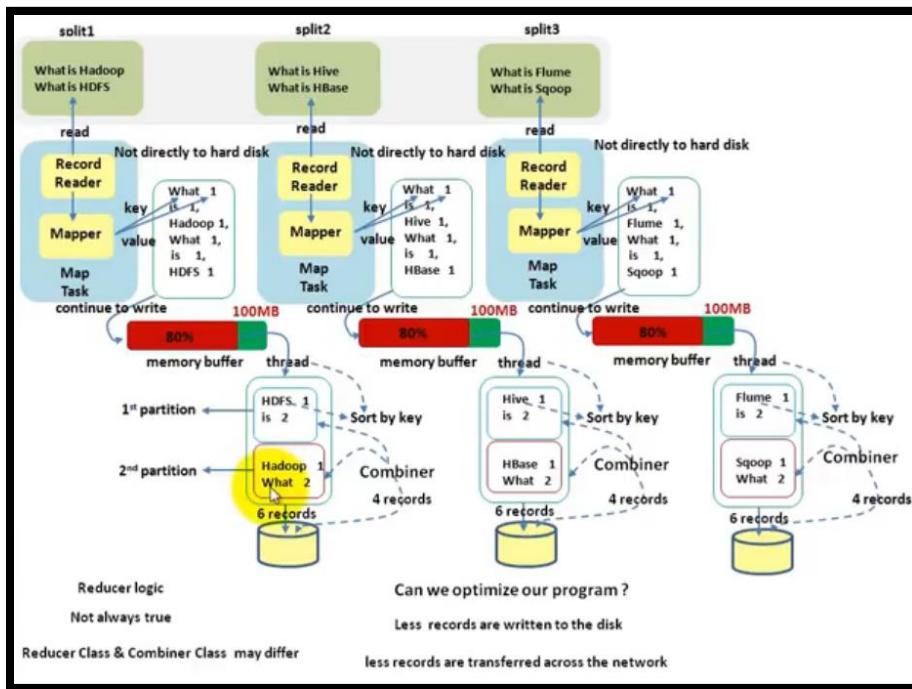
All these operation (partition + sorting) performed inside the memory buffer and later it will write into local filesystem disk



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

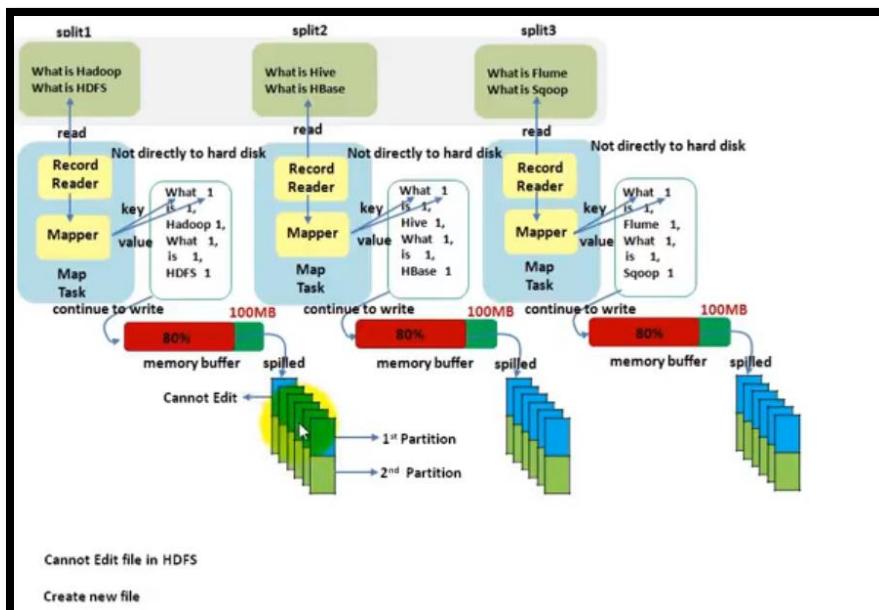


Instead of 6 now only 4 records are written to disk

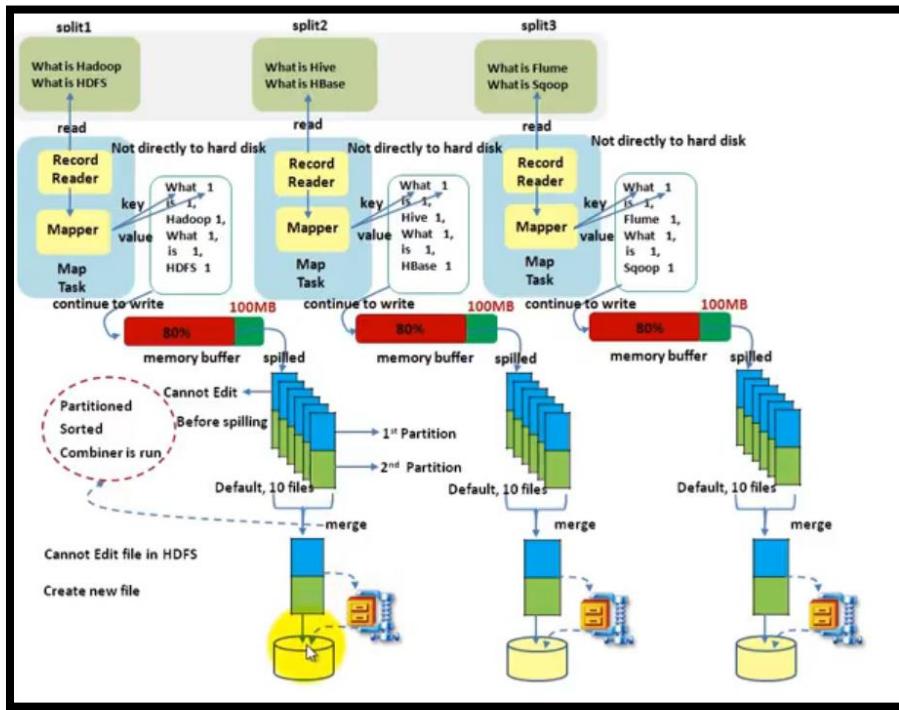


As we know background thread write the data onto the disk once buffer reach to threshold value so once it written data in one spilled files we cannot append the content on same file when threshold limit again reaches

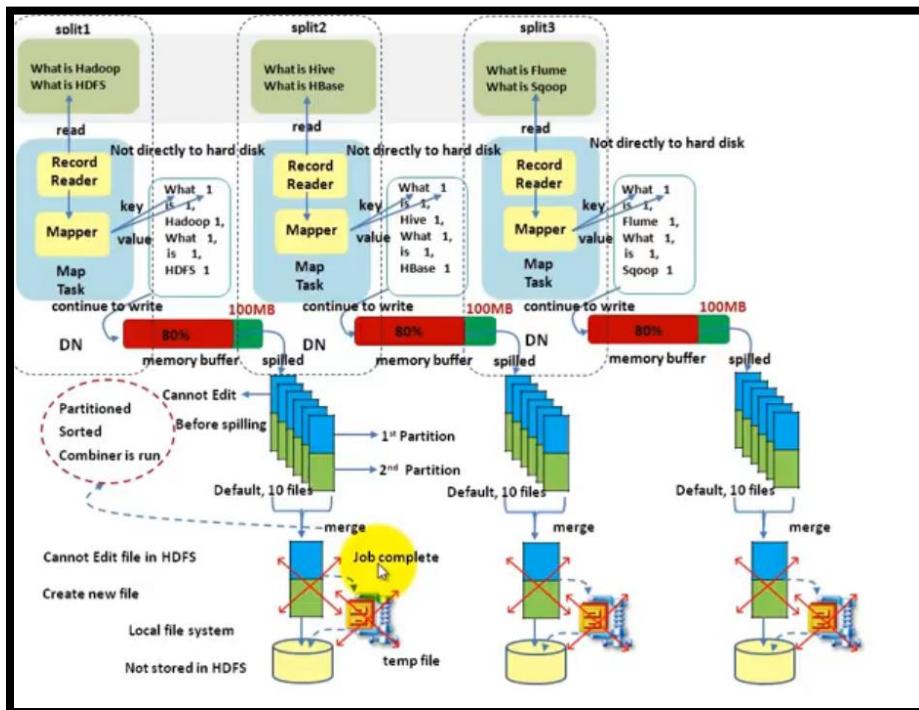
So for every threshold one new file is being created



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

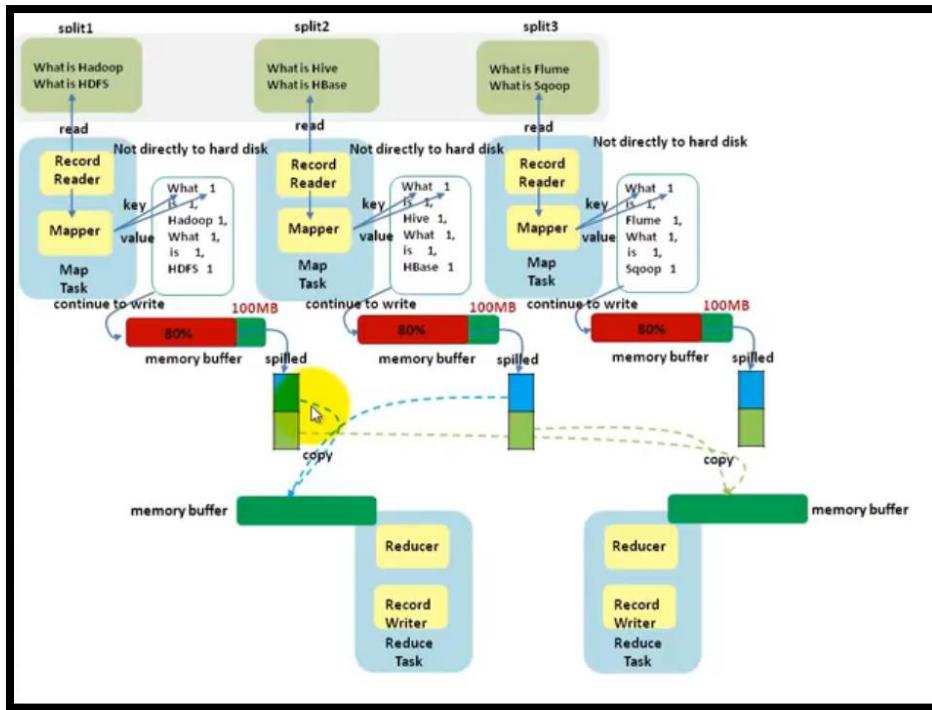


These files are stored on local file system **not on HDFS** of data node where the map task is running, these files are temp files and all are deleted once job finished

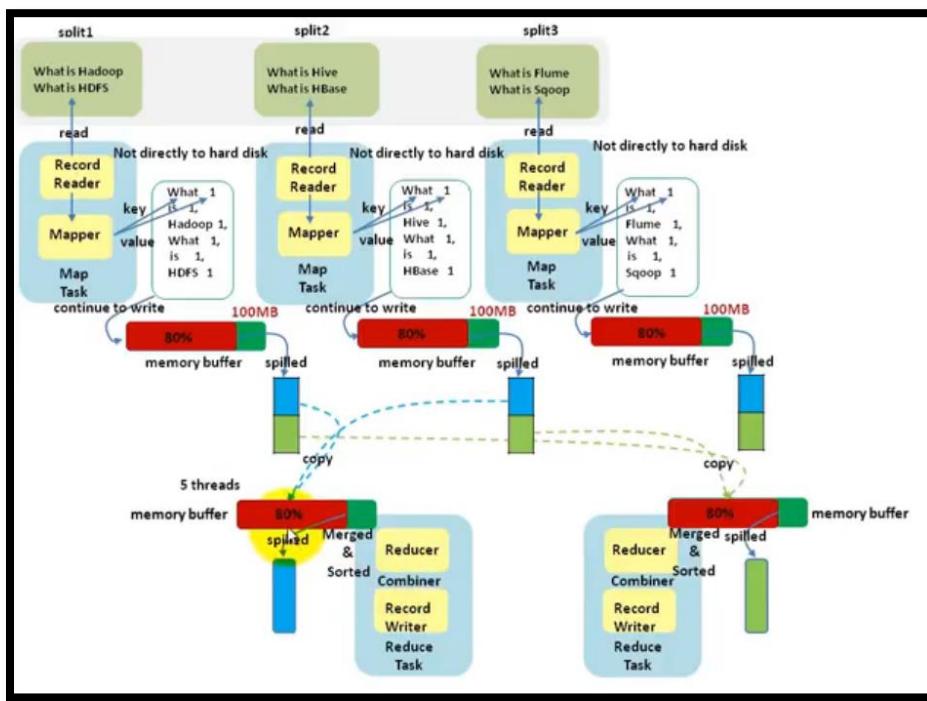


Reduce task starts copying the file from the map output, reduce task will not wait for all the map task to complete , as soon as one map task is complete reduce task start copying the file, that is why some time you see in log that reducer starts before all mapper finish, in that case reducer start copying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

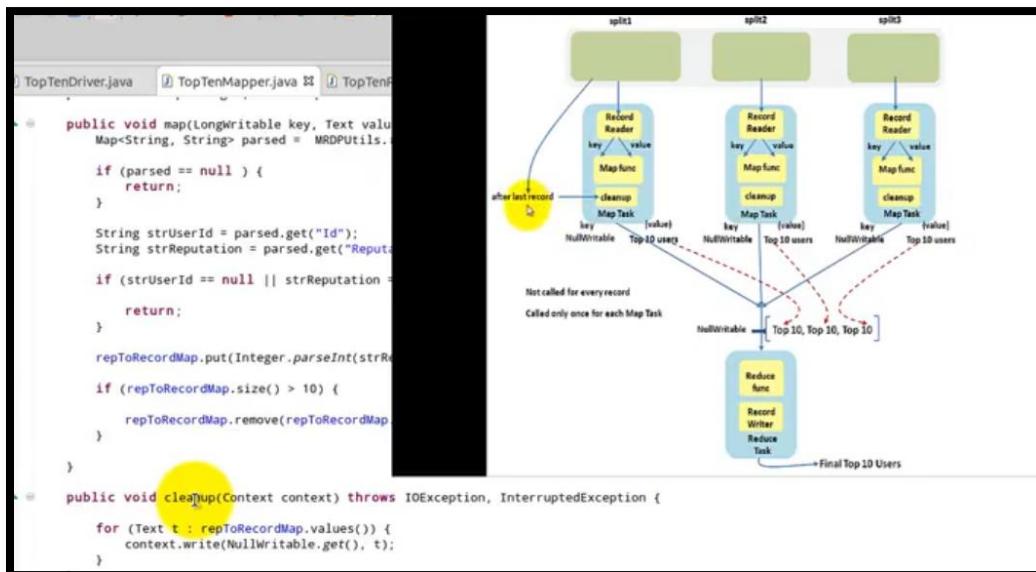
By default reduce task use 5 threads to copying data from map output in parallel



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

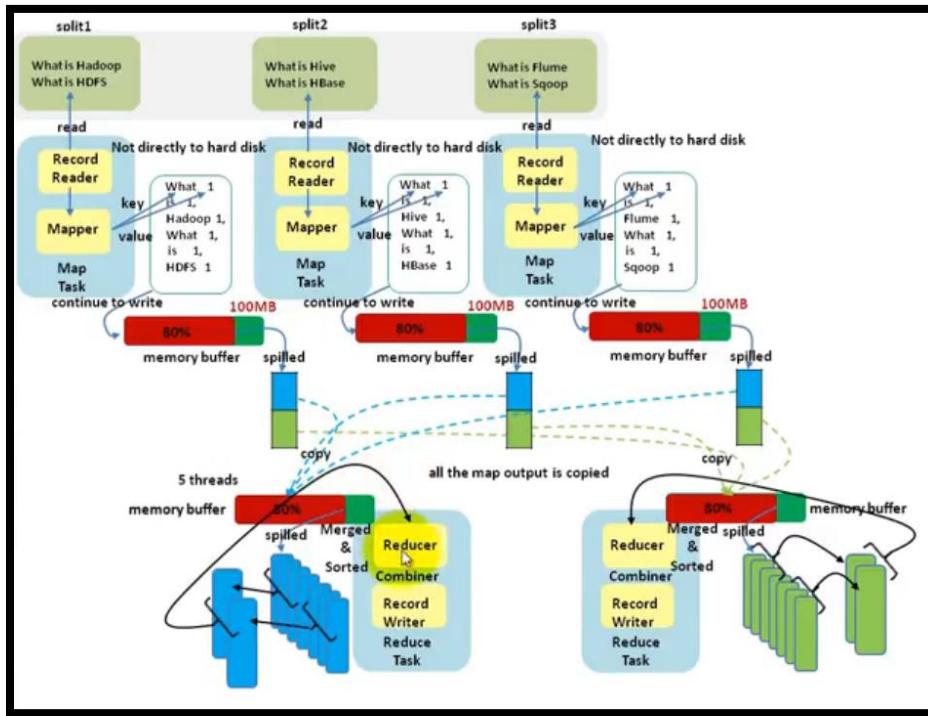


Cleanup function call at the end of map task when all records are processed

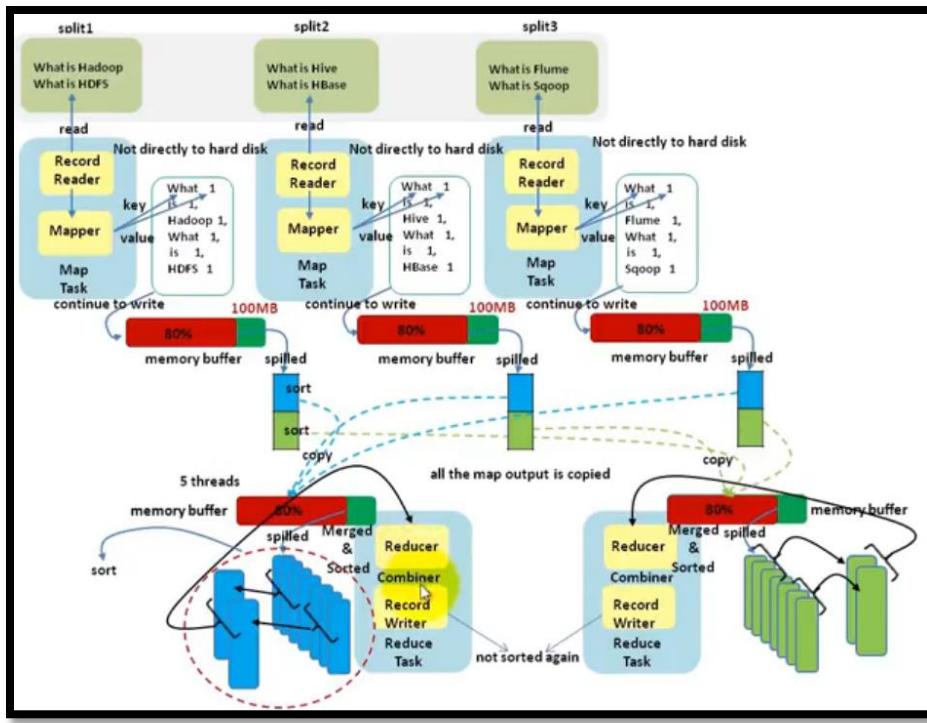


Same as map task here also we could have several spills files which will be merged into large sorted files from background thread

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

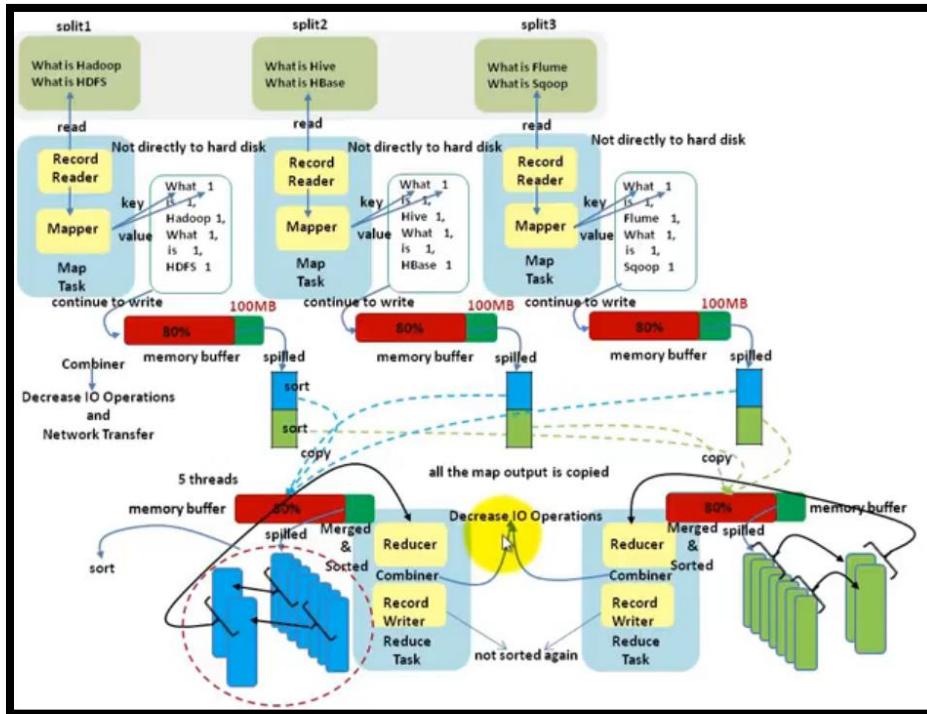


So sorting is done in the map phase as well as in reduce phase, in reduce phase it is done before the reduce function is called, reduce output is not sorted again



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

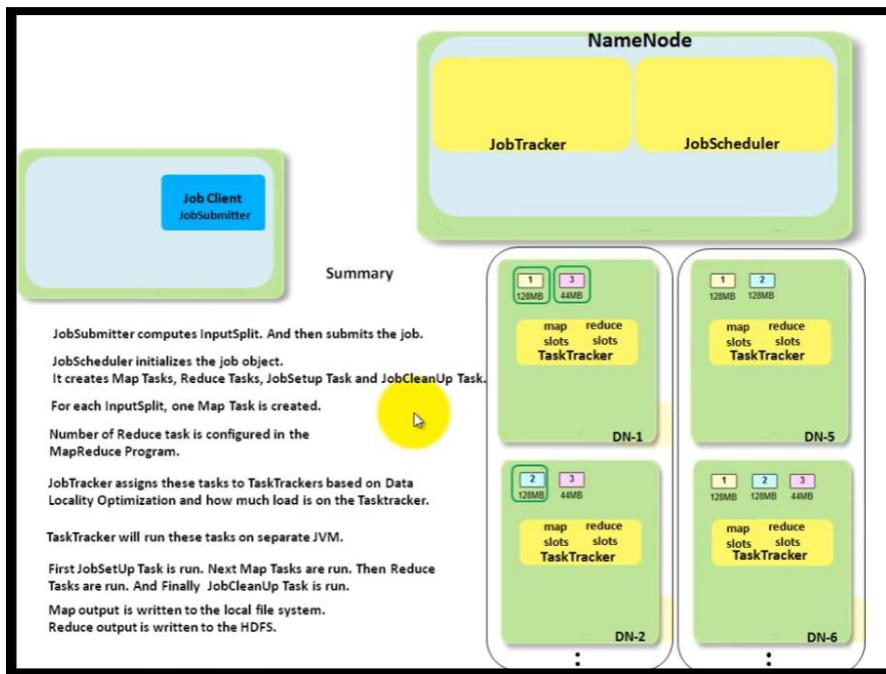


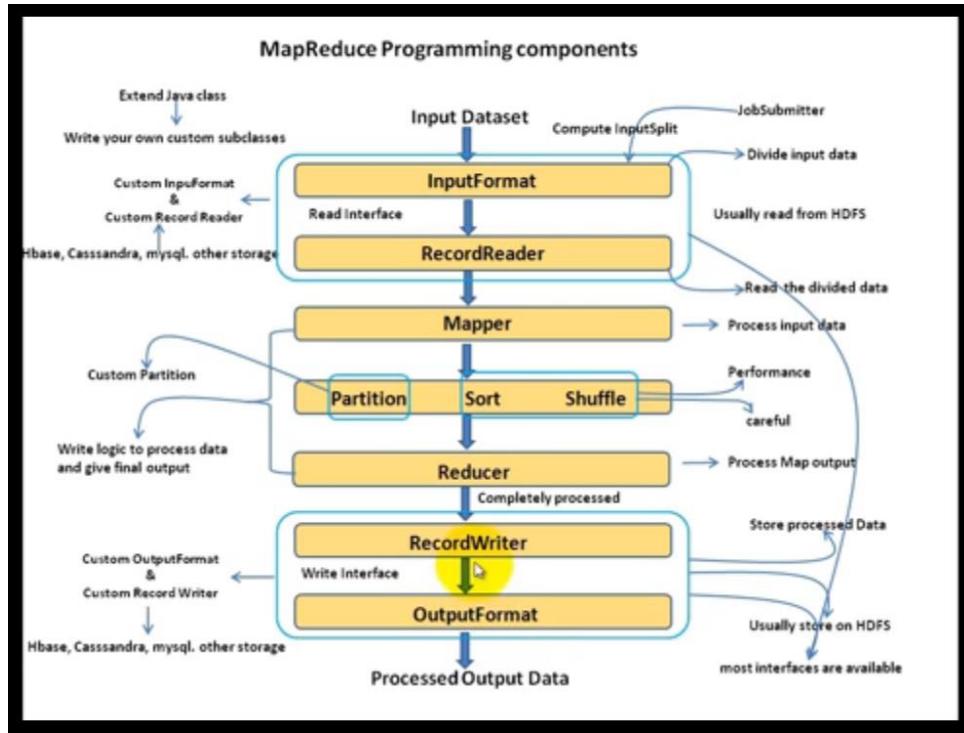
<https://www.youtube.com/watch?v=F4Zc4S-8n0w>

pramod narayana

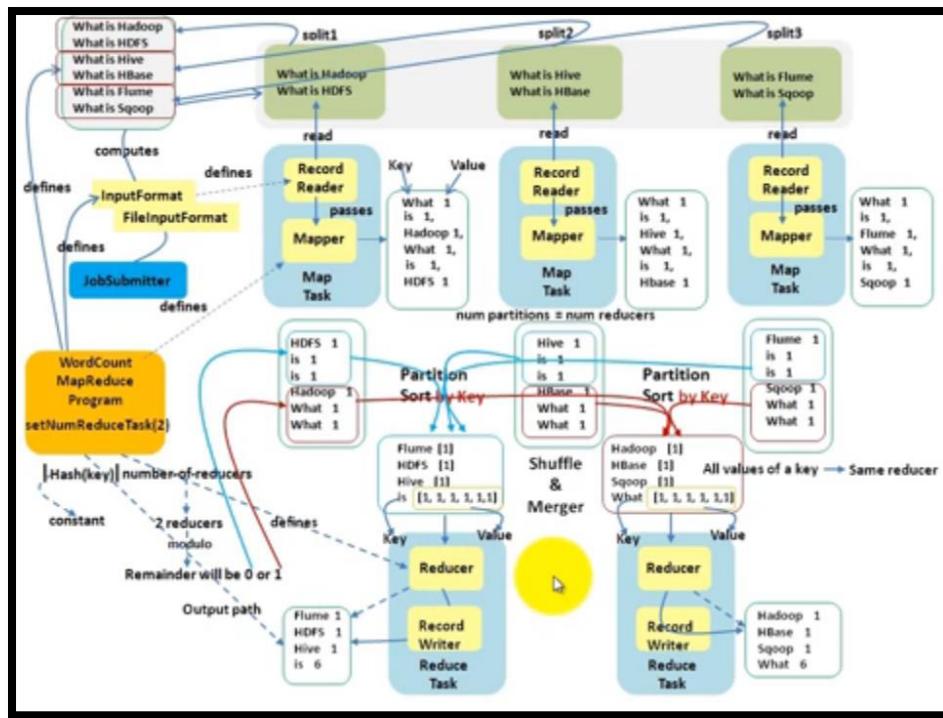
<https://www.youtube.com/channel/UCDApOTqXw80sSxKdyQpcnUA>

Map Reduce version 1 flow





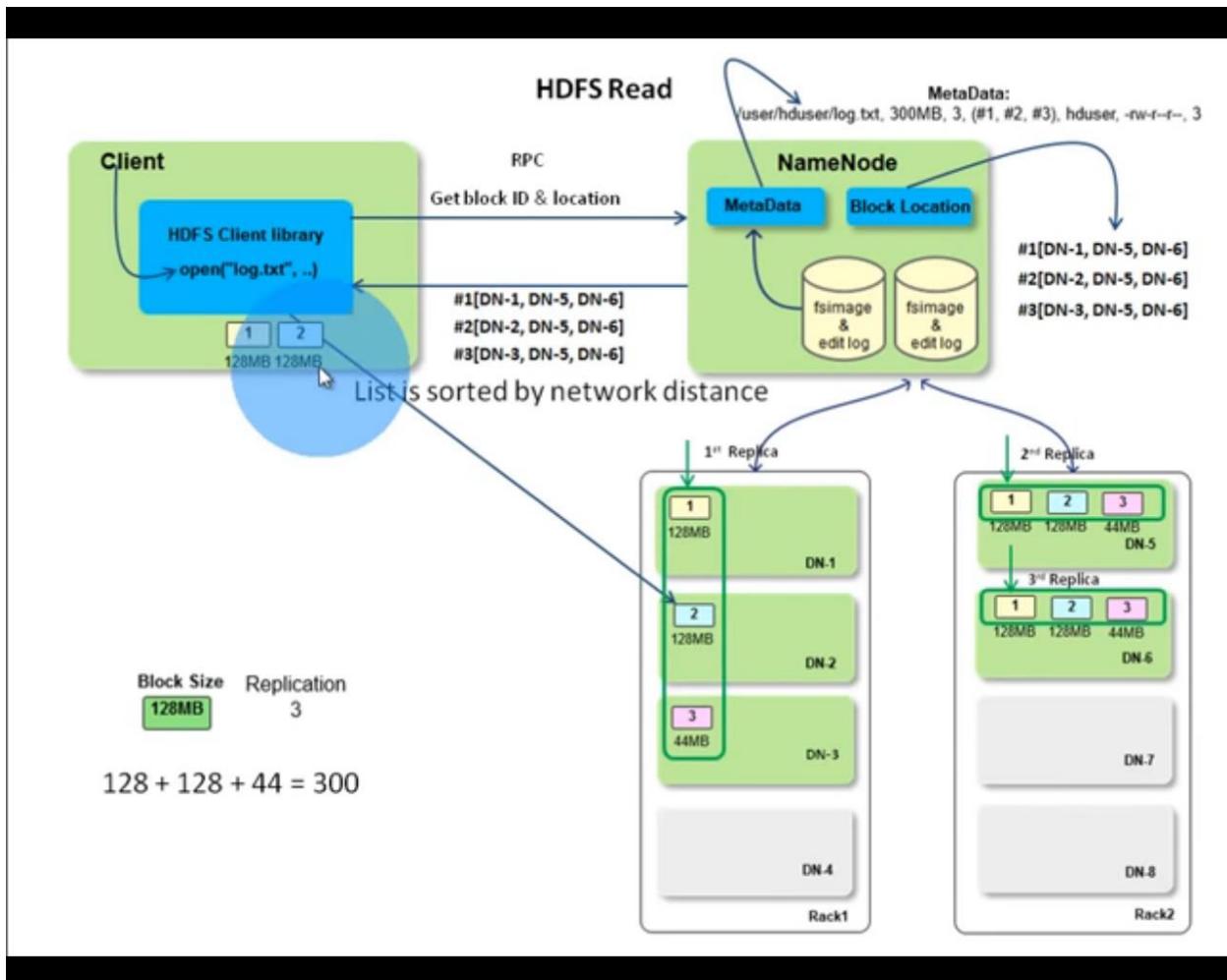
Word count flow



HDFS READ

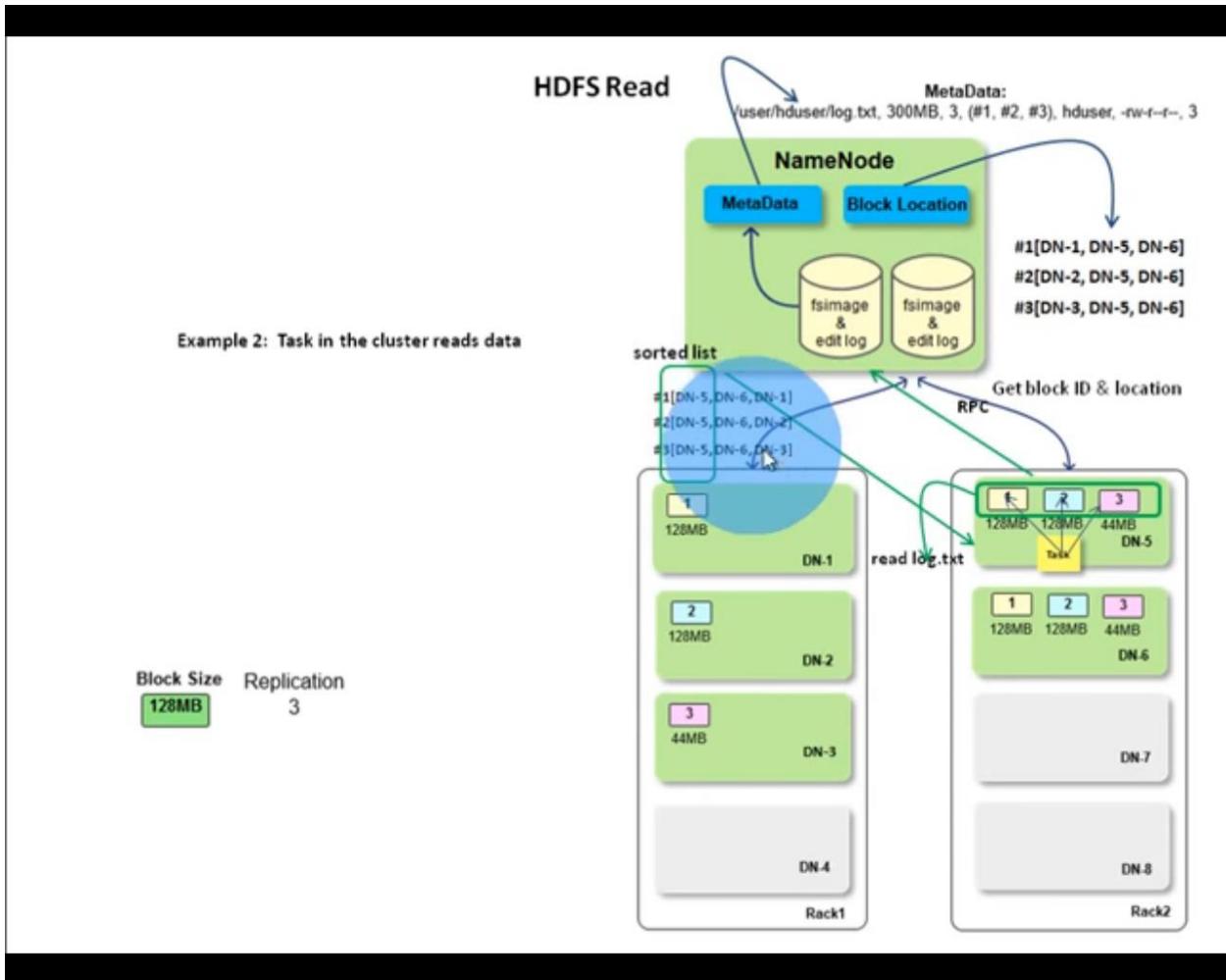
Client outside cluster

Block location returned is sorted distance from client

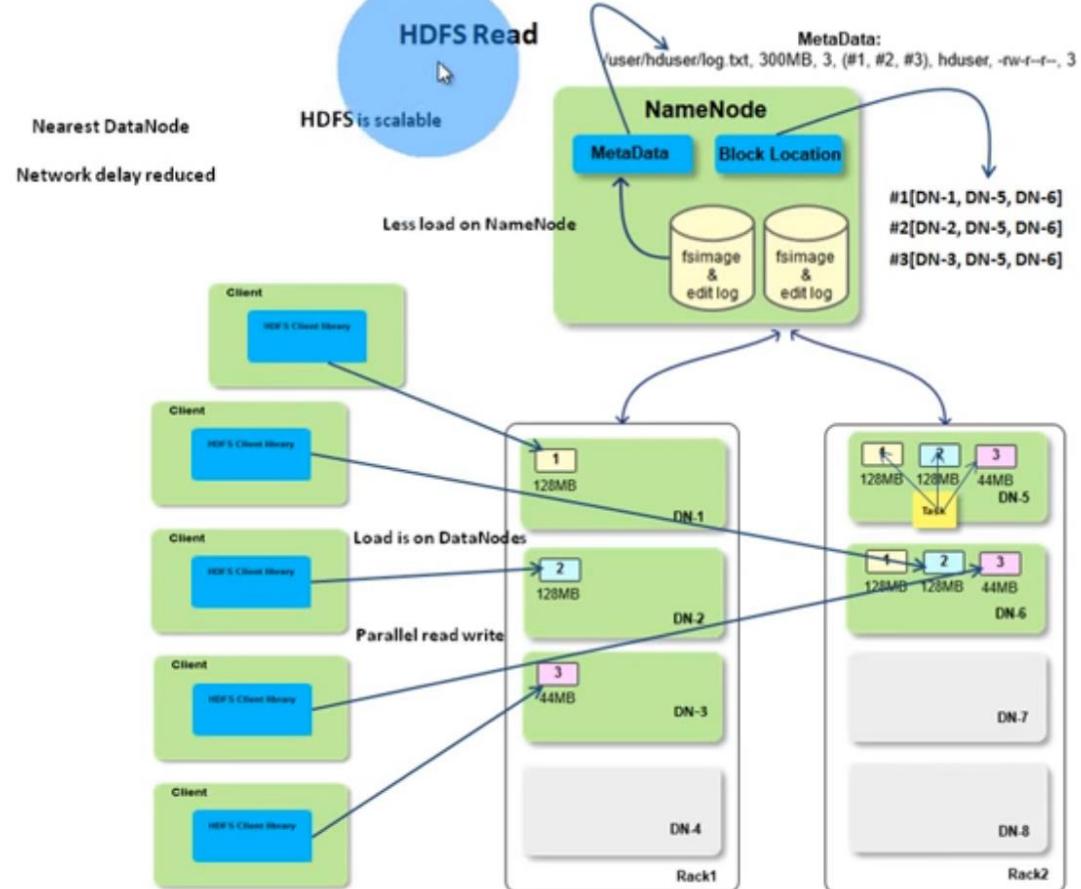


Client inside cluster

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

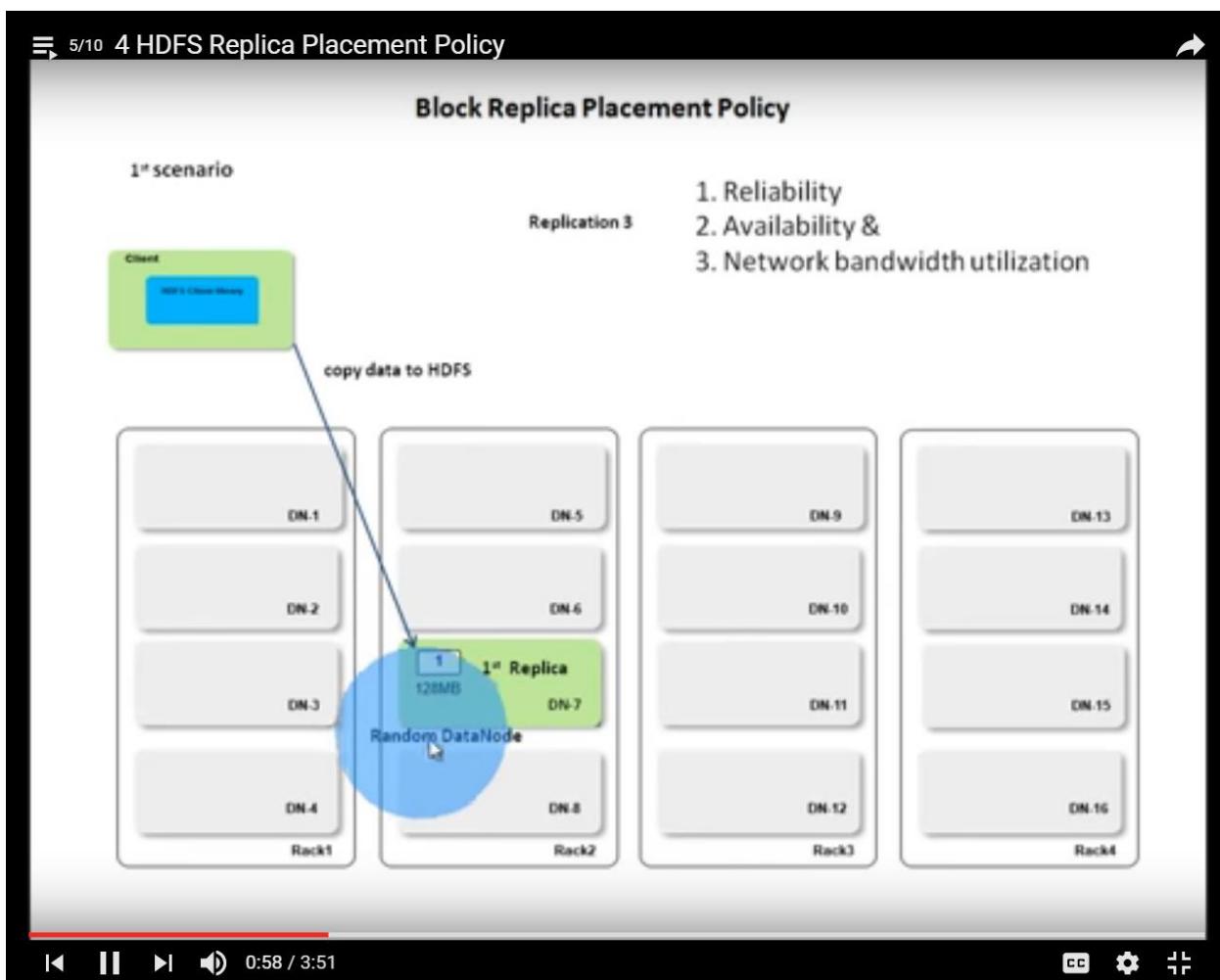


Client directly communicate with data node as per block location given by namenode and that is why name node is not overloaded



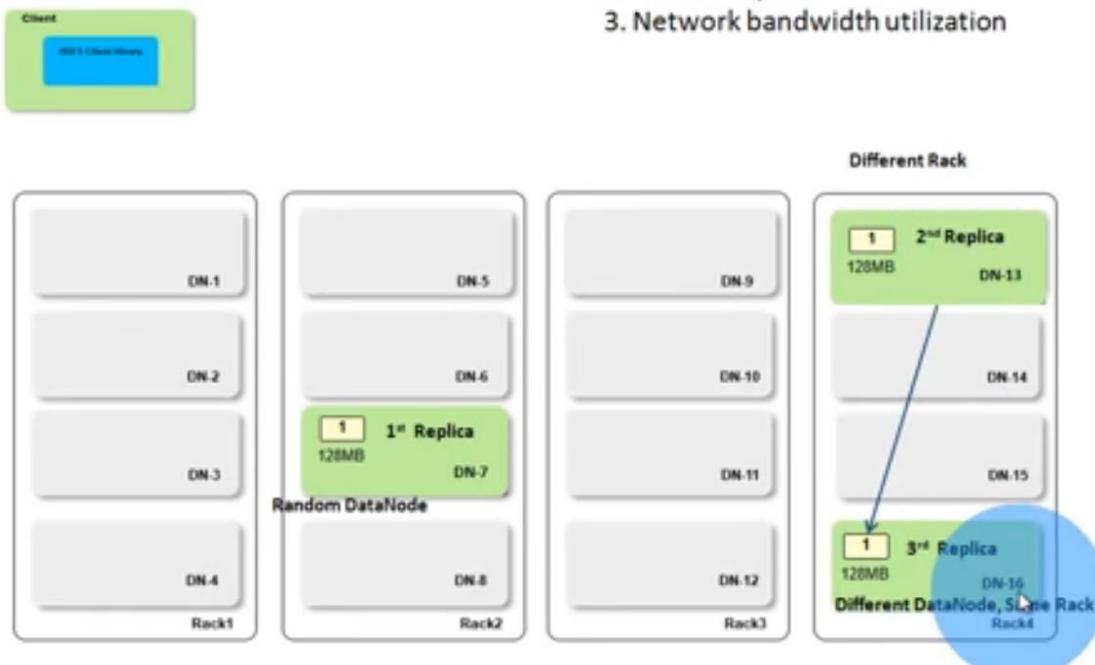
Data Replication policy

Client is outside cluster



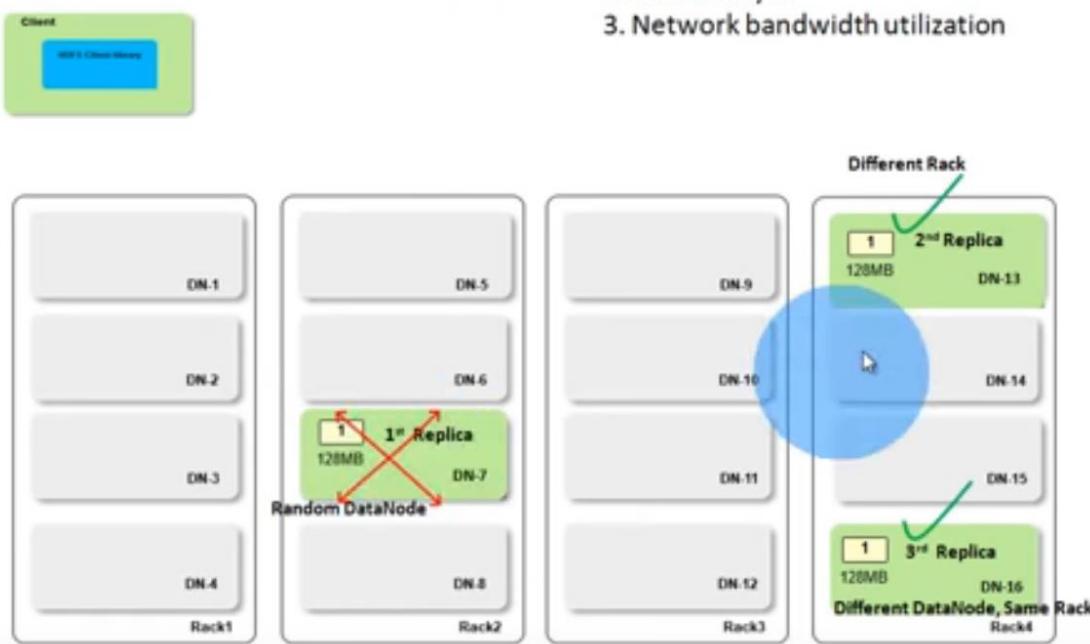
Block Replica Placement Policy

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



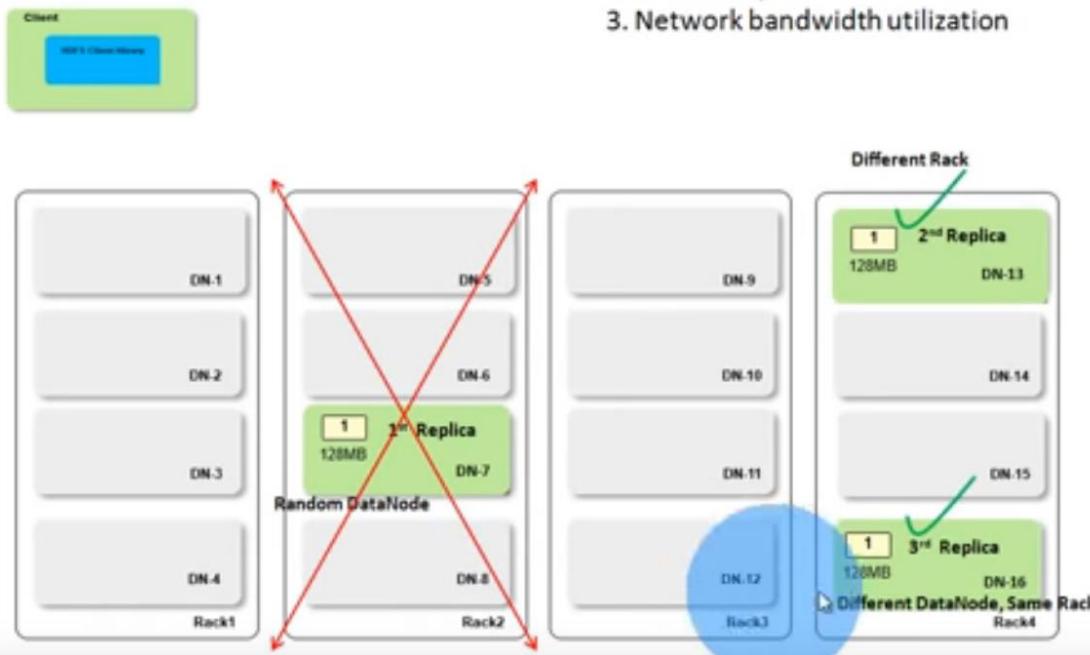
Block Replica Placement Policy

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

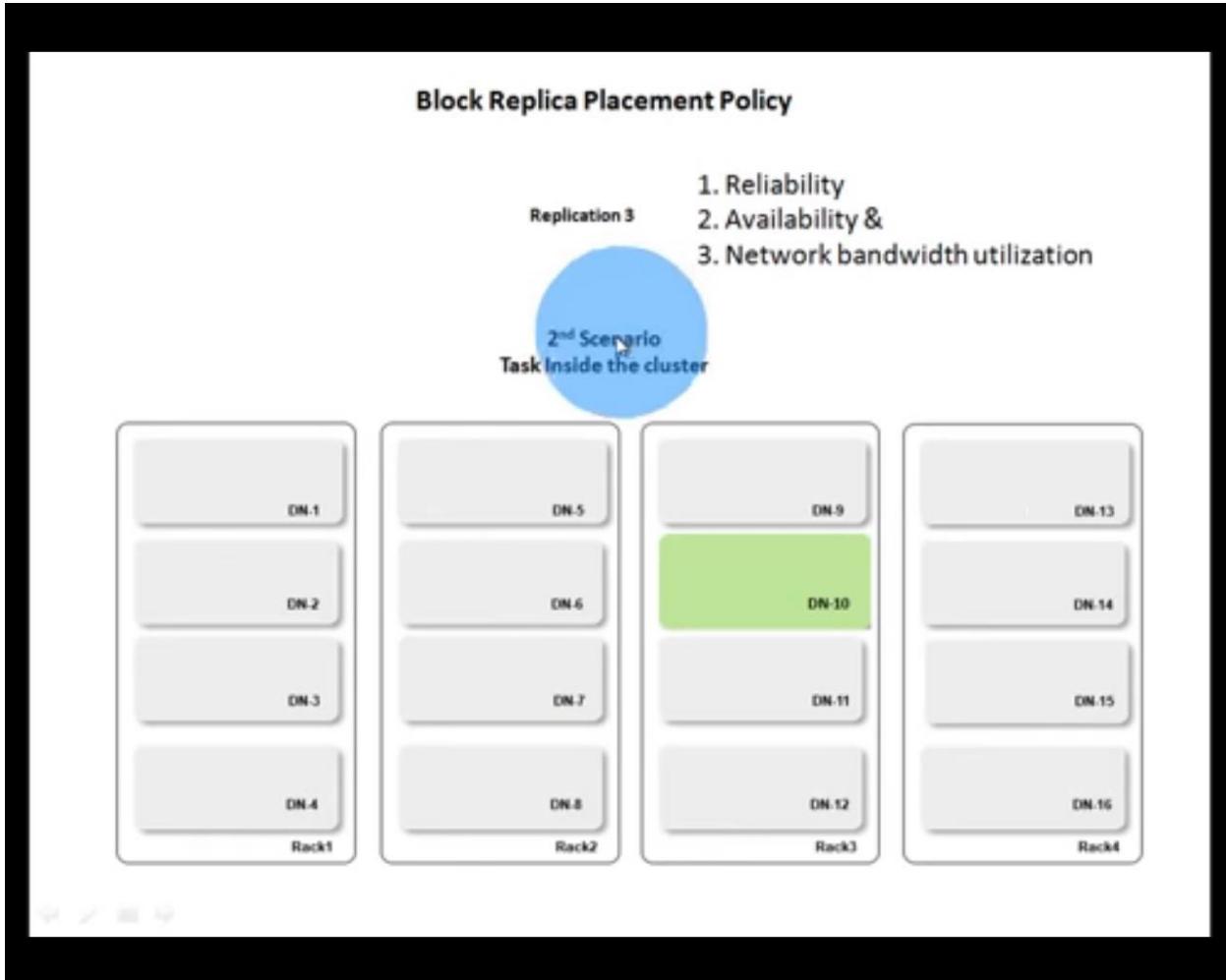


Block Replica Placement Policy

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



Client inside cluster



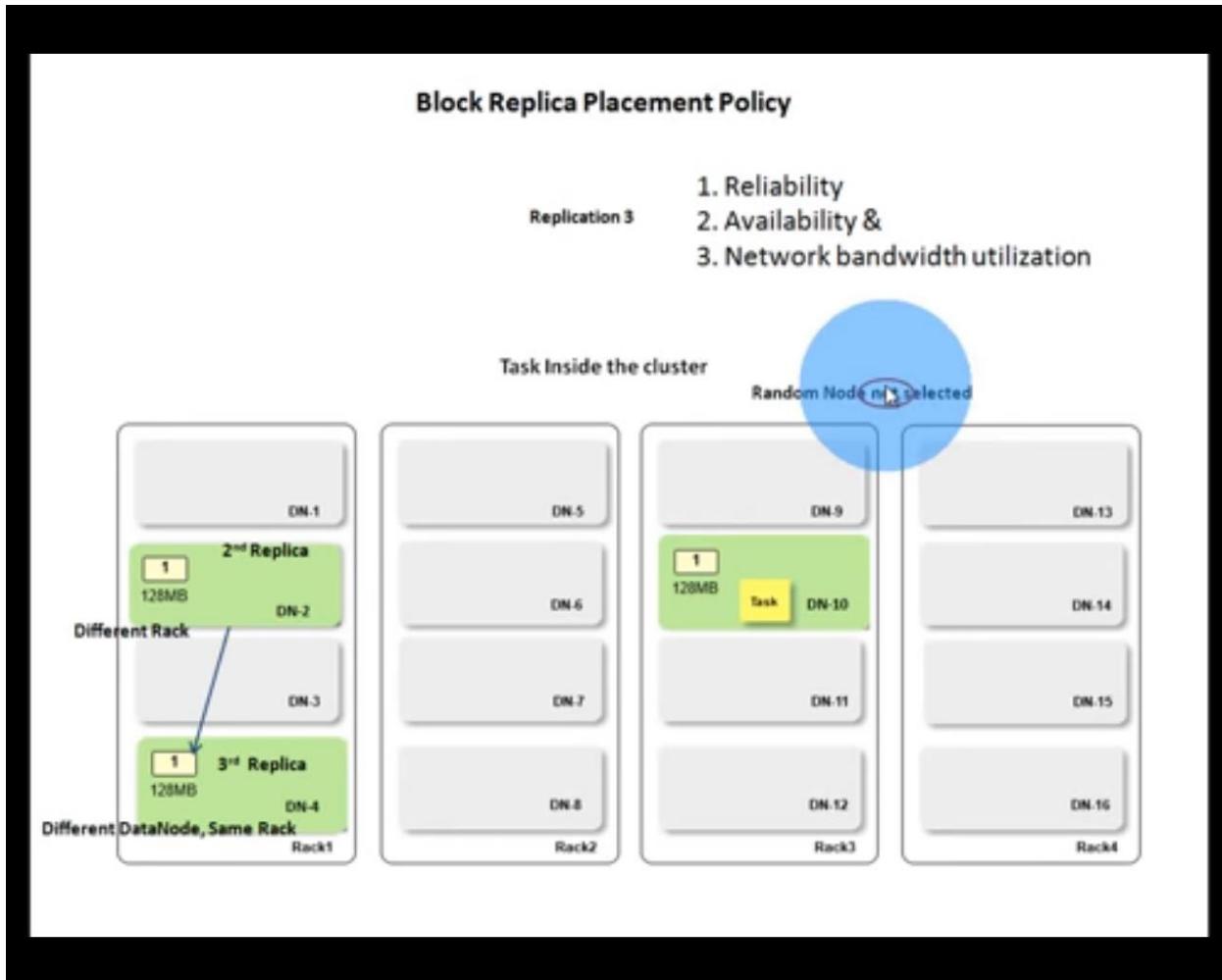
Block Replica Placement Policy

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

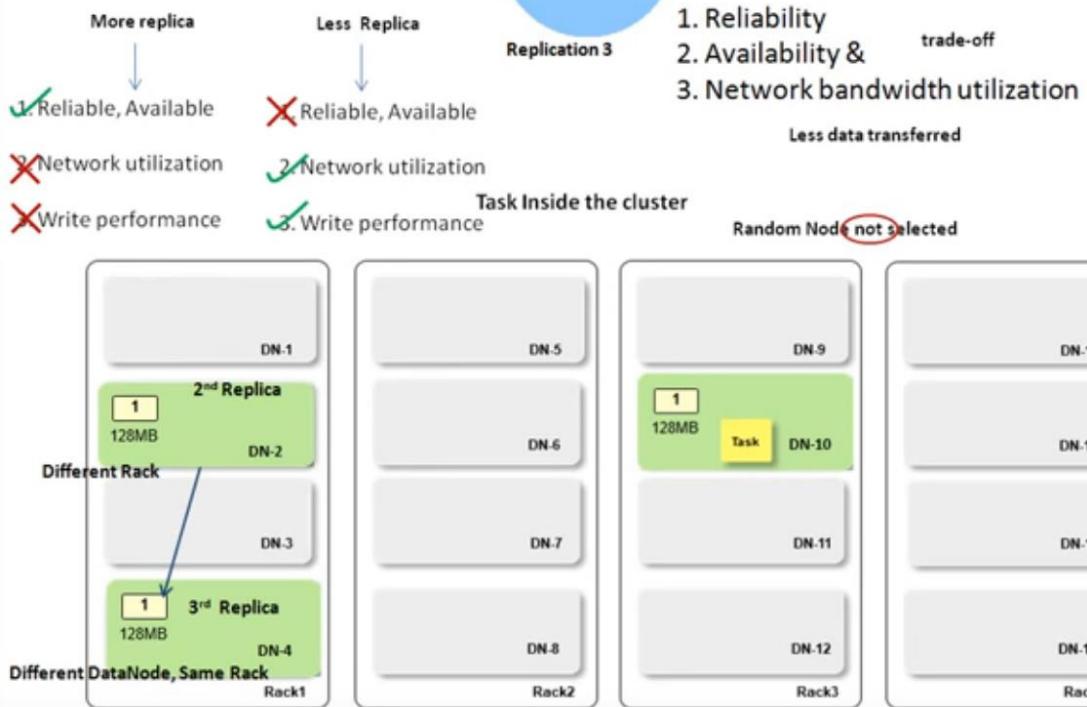
Task Inside the cluster



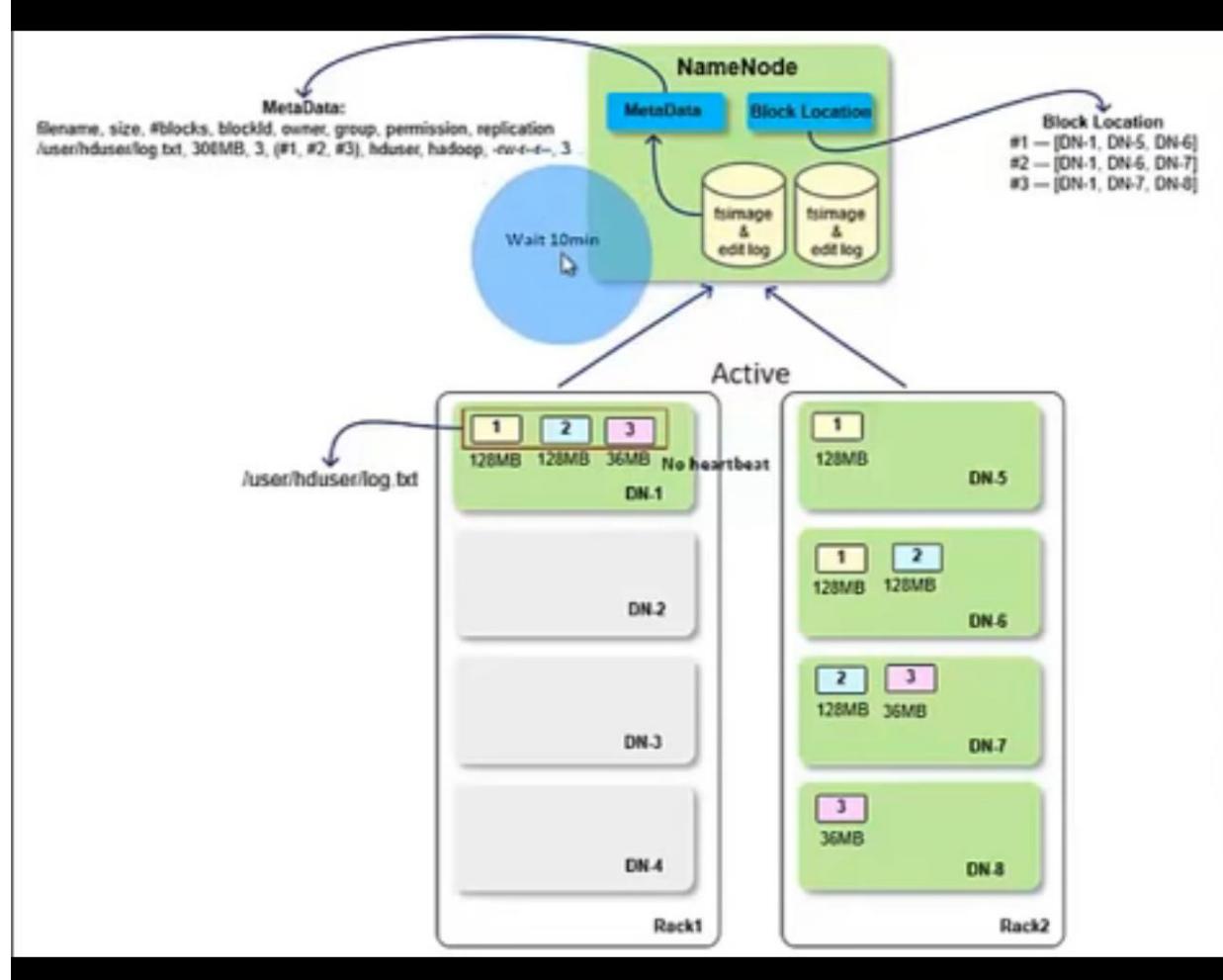
Written in same datanode where task is running



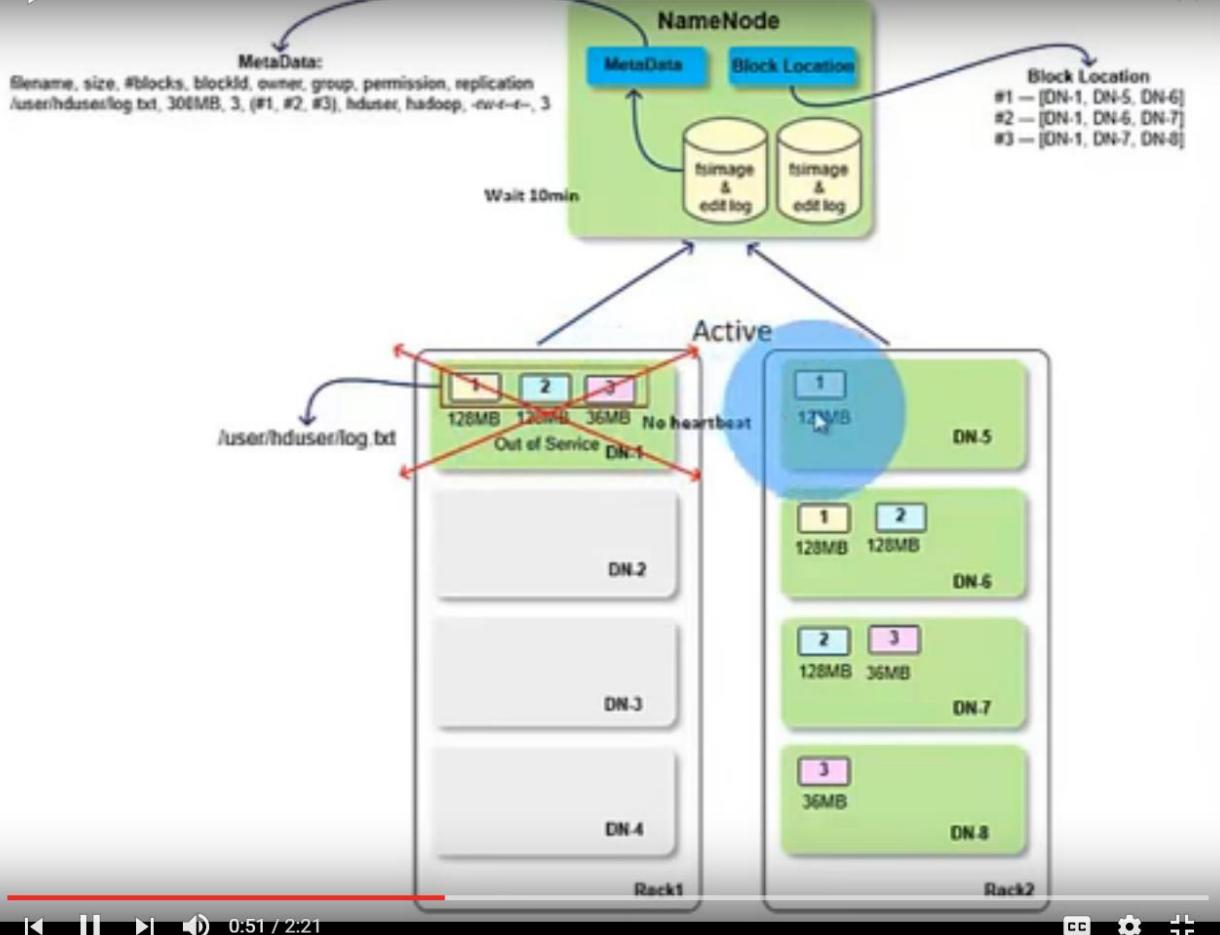
Block Replica Placement Policy



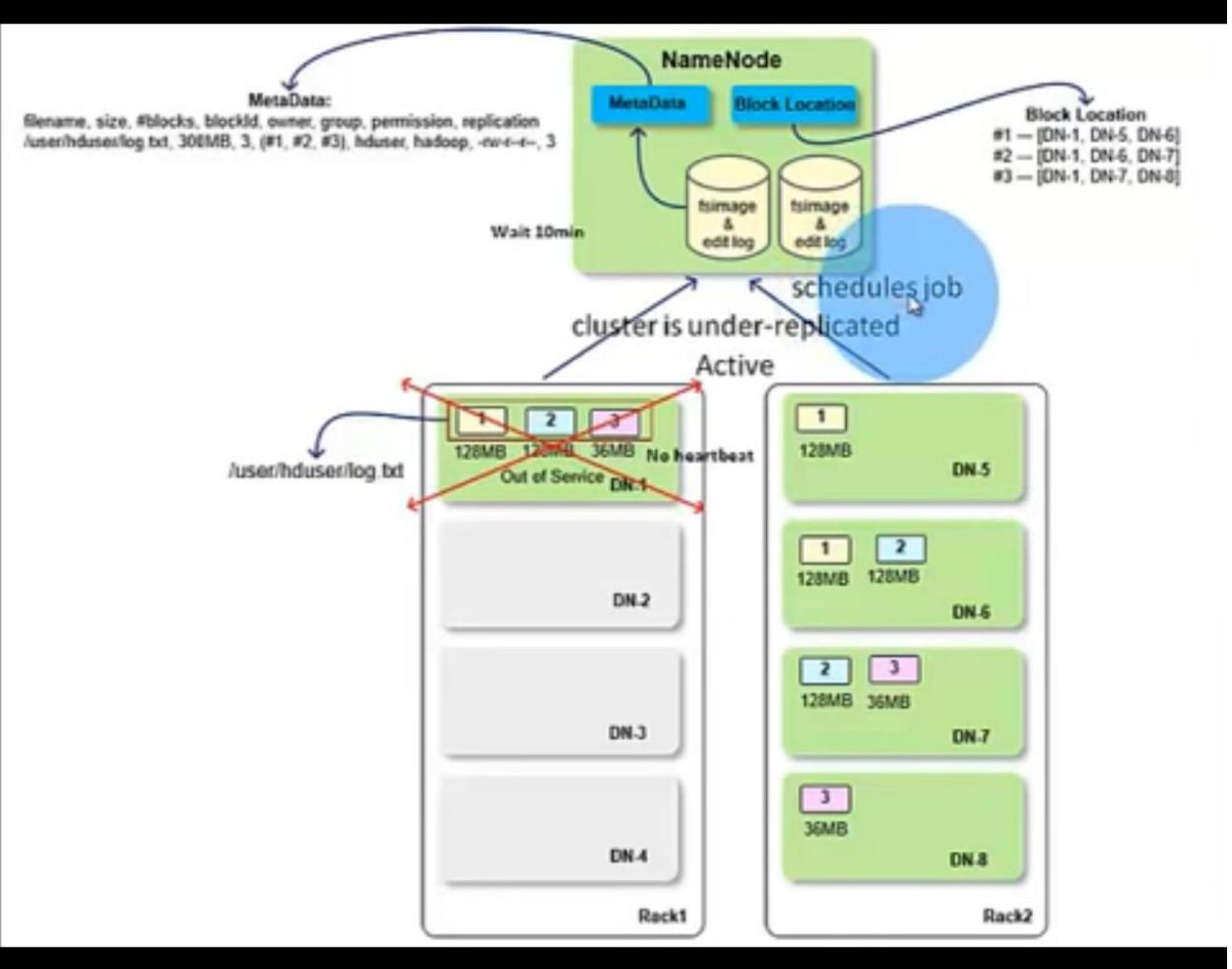
DataNode Out of Service

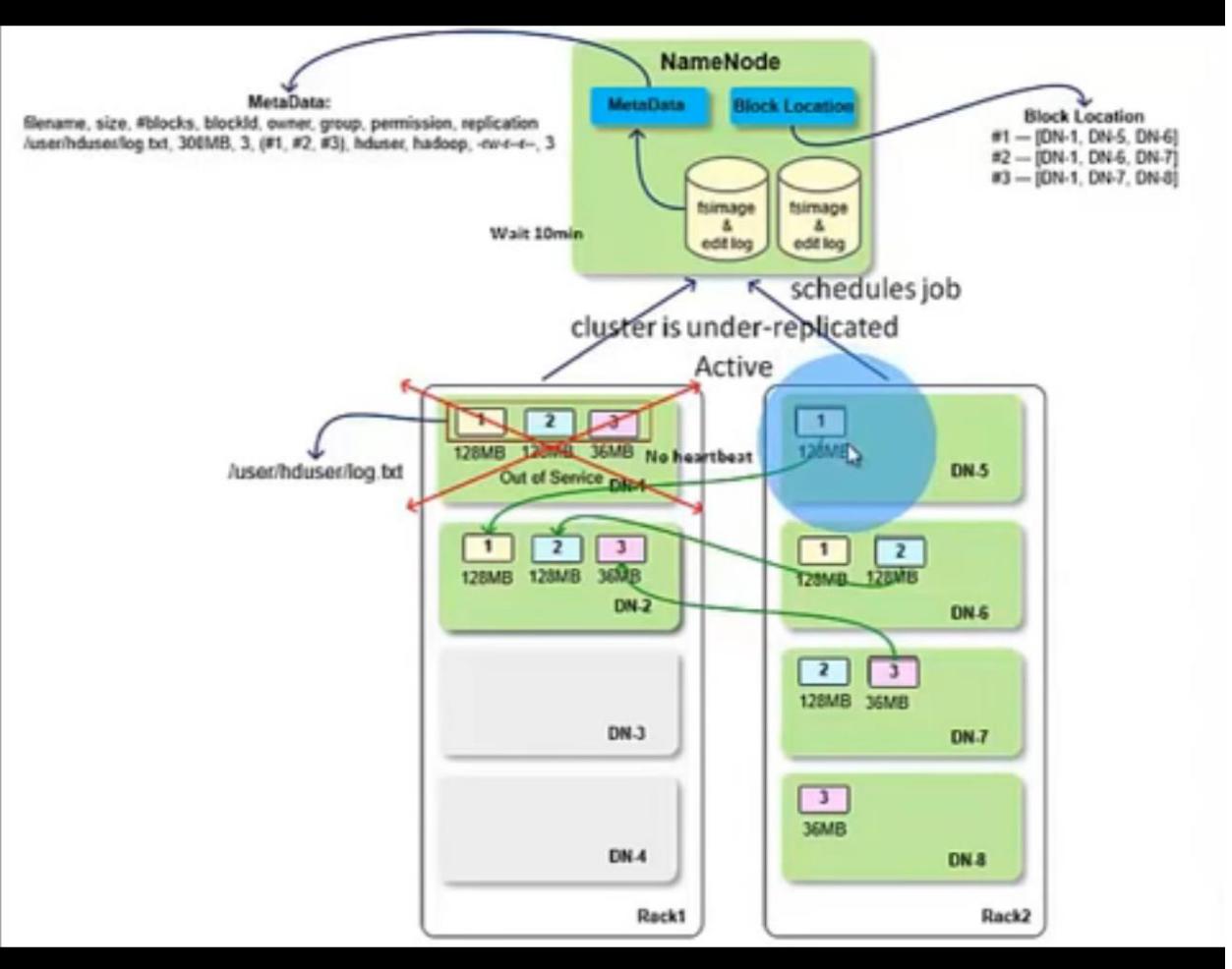


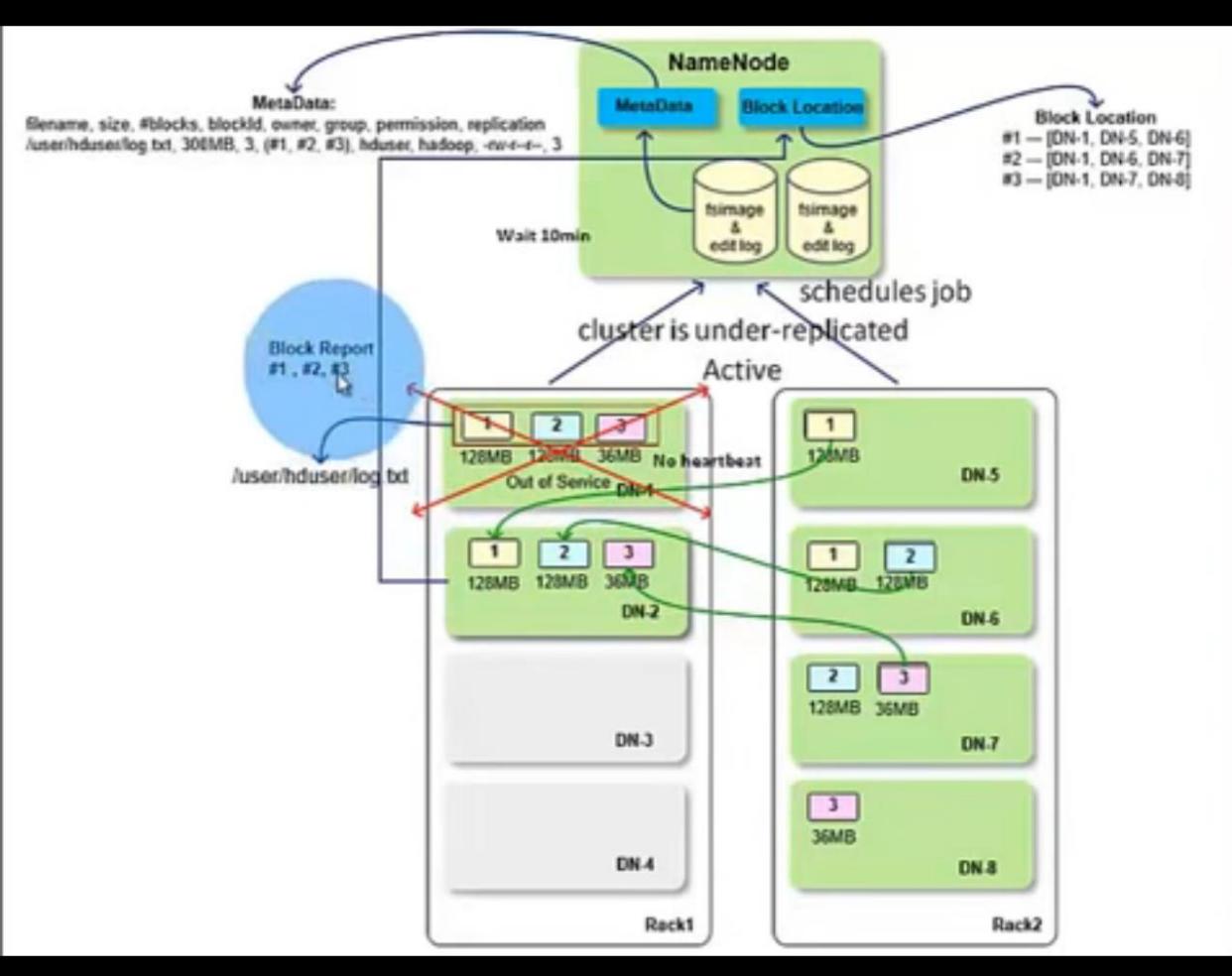
6/10 5 HDFS DataNode Out Of Service



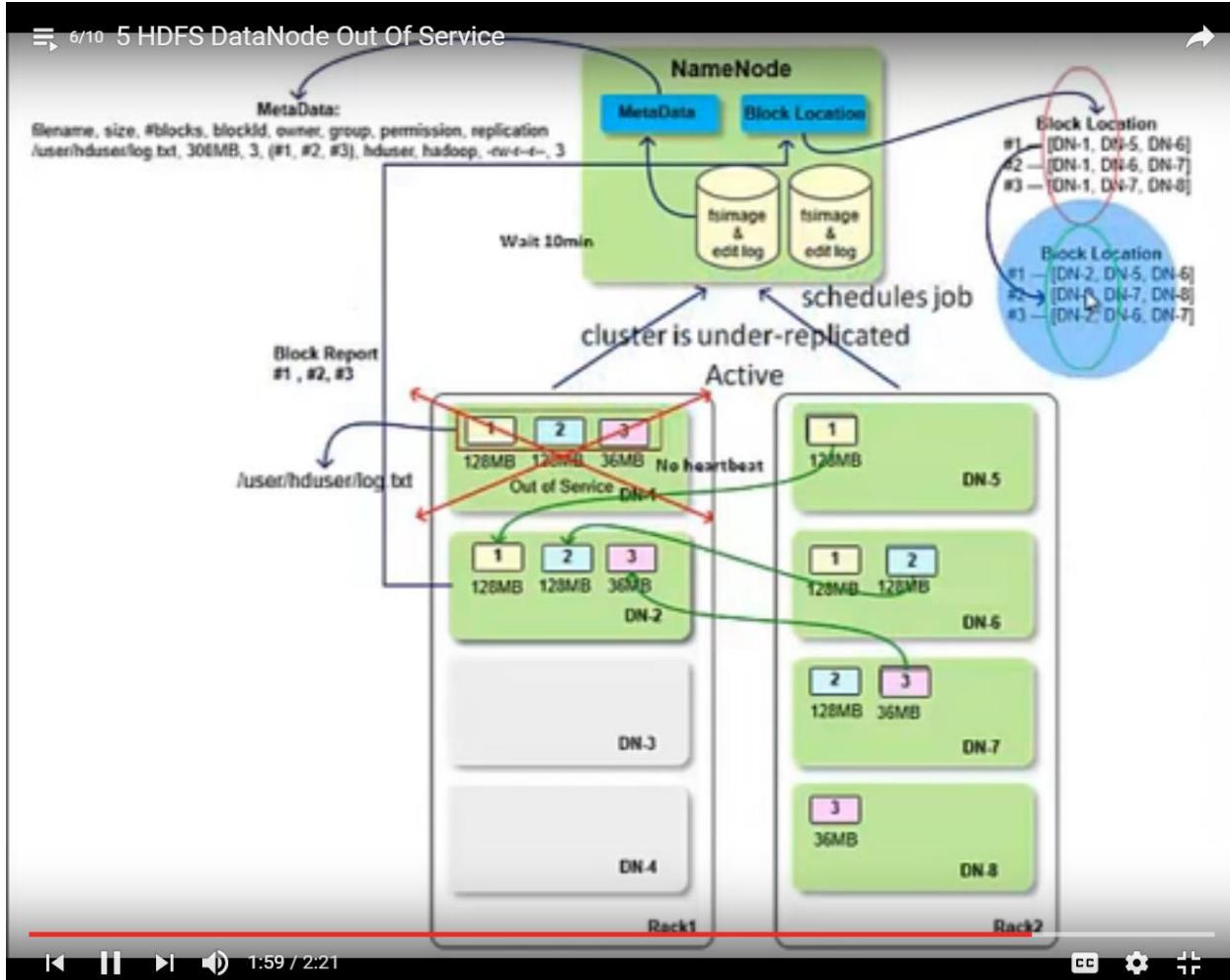
0:51 / 2:21



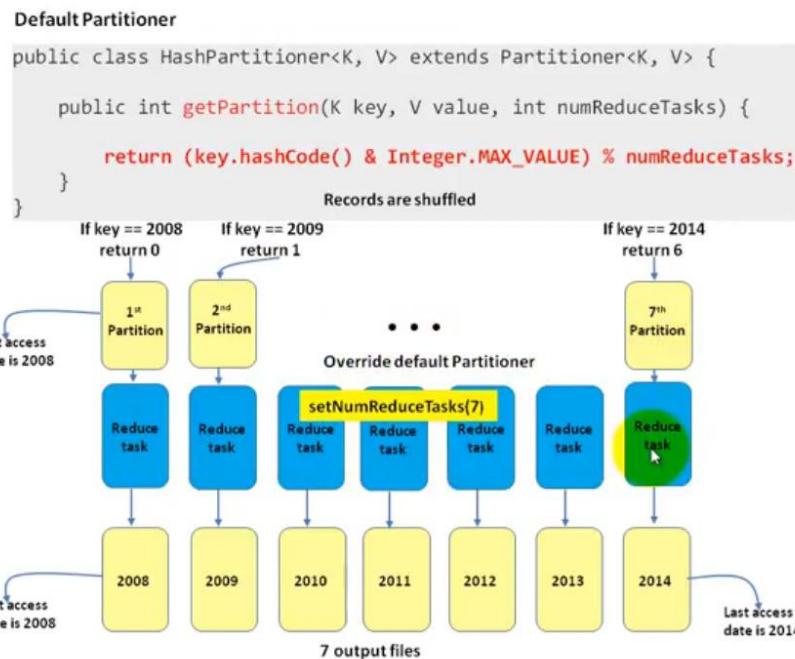




6/10 5 HDFS DataNode Out Of Service



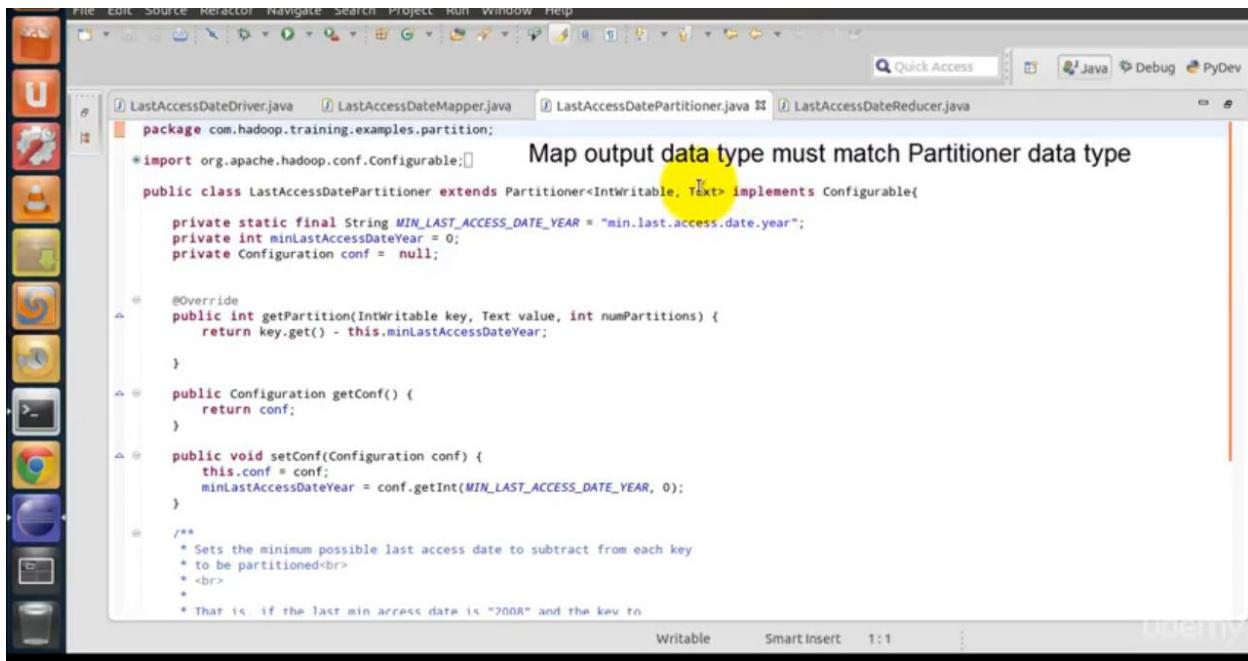
Custom Partition



```
package com.hadoop.training.examples.partition;
import org.apache.hadoop.conf.Configuration;

public class LastAccessDateDriver {
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        String[] otherArgs = new GenericOptionsParser(conf, args)
            .getRemainingArgs();
        if (otherArgs.length != 2) {
            System.err.println("Usage: 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);
    }
}
```



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

import org.apache.hadoop.conf.Configurable;

public class LastAccessDatePartitioner extends Partitioner<IntWritable, Text> implements Configurable{

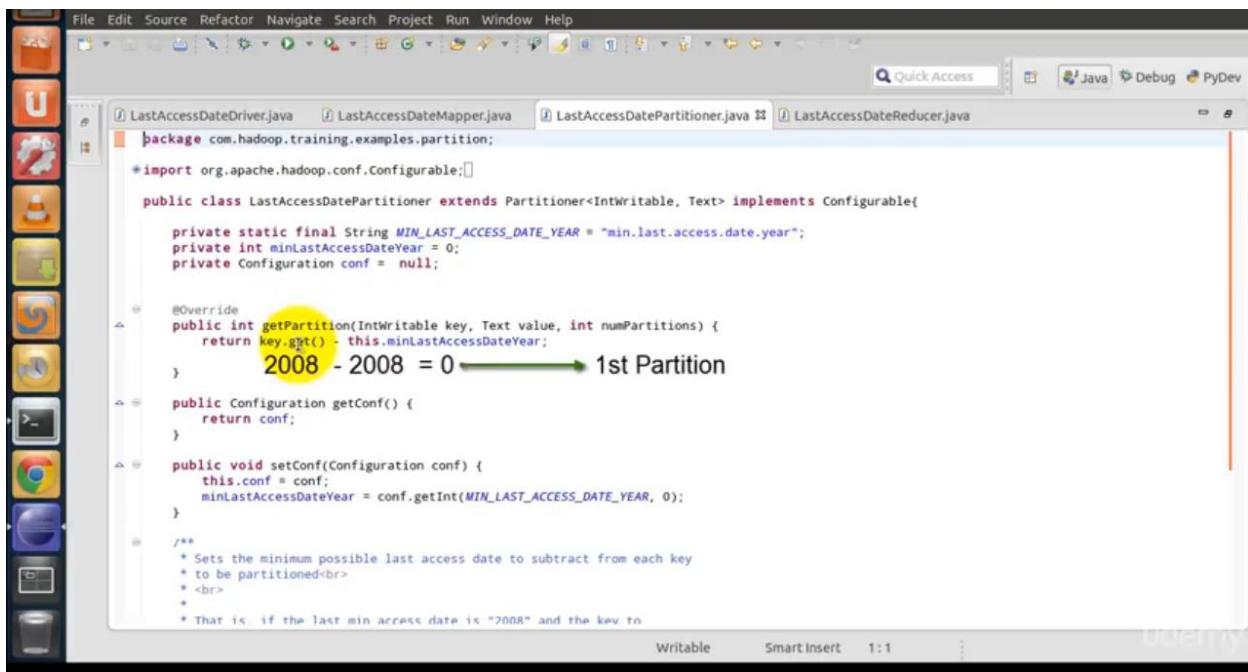
    private static final String MIN_LAST_ACCESS_DATE_YEAR = "min.last.access.date.year";
    private int minLastAccessDateYear = 0;
    private Configuration conf = null;

    @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<br>
     * <br>
     * That is, if the last min access date is "2008" and the key to
    
```



```
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;
    } 2008 - 2008 = 0 → 1st 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<br>
     * <br>
     * That is, if the last min access date is "2008" and the key to
    
```

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

import org.apache.hadoop.conf.Configurable;
import org.apache.hadoop.conf.Configuration;

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<br>
     * <br>
     * That is, if the last min access date is "2008" and the key to
    
```

```
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 = MRDPUtility.transformXmlToMap(value.toString());
        String strdate = parsed.get("LastAccessDate");
        if(strdate == null) {
            return;
        }

        try {
            Date lastAccessDate = frmt.parse(strdate);
            Calendar calendar = Calendar.getInstance();
            calendar.setTime(lastAccessDate);
            calendar.set(calendar.get(Calendar.YEAR));
            context.write(outkey, value);
        } catch (ParseException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

Screenshot of the Eclipse IDE interface showing a Java file named `LastAccessDateReducer.java`. The code implements a `Reducer` class that writes key-value pairs to the context. A yellow circle highlights the `values` parameter in the `reduce` method.

```
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);
        }
    }
}
```

Screenshot of the Eclipse IDE interface showing the same Java code for `LastAccessDateReducer.java`. A yellow circle highlights the `values` parameter in the `reduce` method. A callout diagram provides a visual representation of the output format:

Default OutputFormat is
TextOutputFormat

Tab space

<key> <value>
<key> <value>
<key> <value>

The callout diagram shows three lines of output, each consisting of a key and a value separated by a tab character. The text "Default OutputFormat is TextOutputFormat" is at the top, followed by an arrow pointing to the tab space between the key and value in the first line of the callout.

The screenshot shows the Eclipse IDE interface with the Java perspective selected. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons for file operations like Open, Save, Cut, Copy, Paste, Find, and Run. The Quick Access search bar is at the top right. The status bar at the bottom shows Writable, Smart Insert, and the time 22:10.

The code editor displays the `LastAccessDateDriver.java` file. The code is a Java class for a Hadoop job. It starts by printing usage information and exiting with code 2 if no arguments are provided. It then creates a new Job object, sets the jar by class, and specifies the output key and value classes as IntWritable and Text respectively. It sets the mapper and reducer classes, and configures the job to handle access dates from 2008 to 2014. It uses a custom partitioner and sets the minimum last access date to 2008. The input and output formats are set to TextOutputFormat, and the separator is set to a tab character (`\t`). Finally, it exits with code 0 or 1 based on whether the job completed successfully.

```
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);

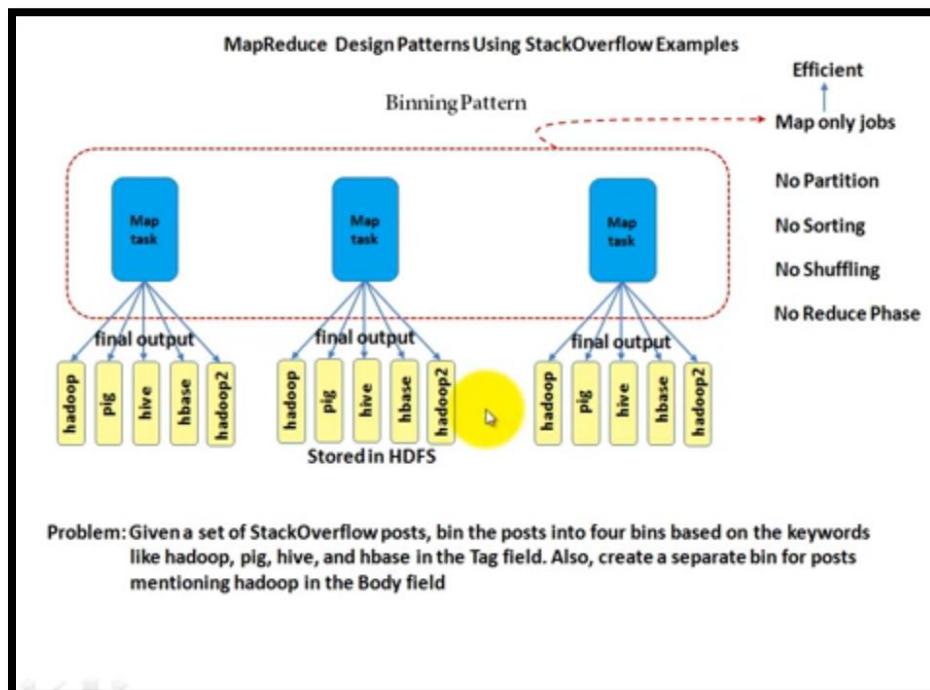
FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));

job.setOutputFormatClass(TextOutputFormat.class);
job.getConfiguration().set("mapred.textoutputformat.separator", "\t");

System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

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



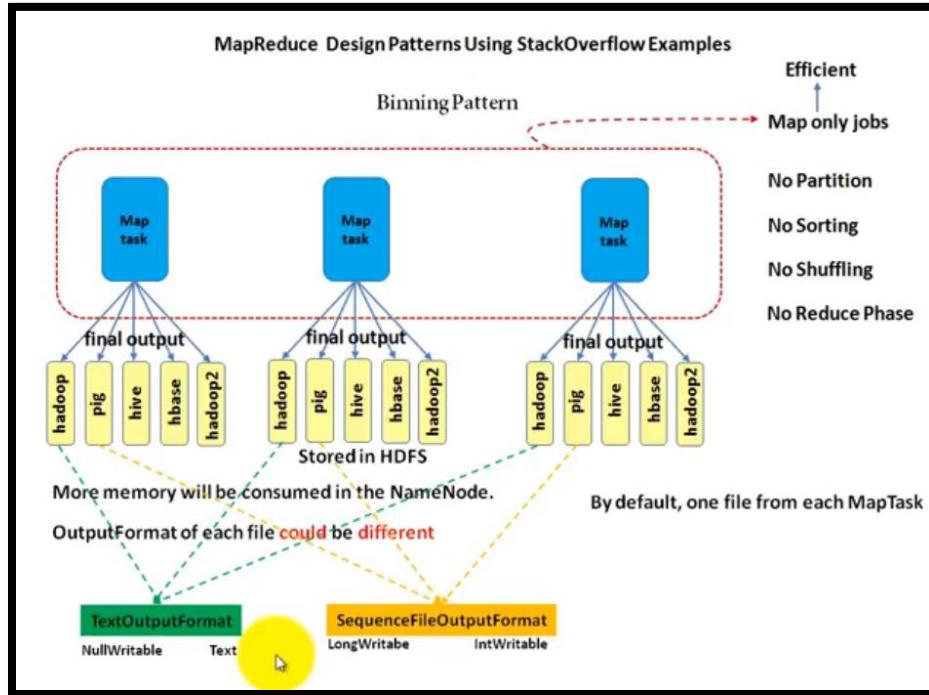
```

hduser@nameNode:~/hadoop/data/stackoverflow
hduser@nameNode:~/hadoop/data/stackoverflow
<?xml version="1.0" encoding="utf-8"?>
<posts>
    <row Id="4" PostTypeId="1" AcceptedAnswerId="7" CreationDate="2008-07-31T21:42:52.667" Score="251" ViewCount="15207" Body="&lt;p&gt;I want to use a track-bar to change a form's opacity.&lt;/p&gt;&lt;x&gt;&lt;p&gt;This is my code:&lt;/p&gt;&lt;x&gt;&lt;code&gt;&lt;input type="range" min="0" max="100" value="50" /&gt;&lt;/code&gt;&lt;pre&gt;&lt;code&gt;&lt;input type="range" min="0" max="100" value="50" /&gt;&lt;/code&gt;&lt;/pre&gt;&lt;/p&gt;&lt;p&gt;When I try to build it, I get this error:&lt;/p&gt;&lt;x&gt;&lt;p&gt;I tried making &lt;strong&gt;trans&lt;/strong&gt; to &lt;strong&gt;double&lt;/strong&gt;, but then the control doesn't work. This code has worked fine for me in VB.NET in the past. &lt;/p&gt;&lt;x&gt;" OwnerUserId="8" LastEditorUserId="2648239" LastEditorDisplayName="Rich B" LastEditDate="2014-01-03T02:42:54.963" LastActivityDate="2014-01-03T02:42:54.963" Title="When setting a form's opacity should I use a decimal or double?" Tags="&lt;code&gt;&lt;input type="range" min="0" max="100" value="50" /&gt;&lt;/code&gt;" AnswerCount="13" CommentCount="25" FavoriteCount="23" CommunityOwnedDate="2012-10-31T16:42:47.213" />
    <row Id="6" PostTypeId="1" AcceptedAnswerId="31" CreationDate="2008-07-31T22:08:08.620" Score="121" ViewCount="8524" Body="&lt;p&gt;I have an absolutely positioned &lt;code&gt;div&lt;/code&gt; containing several children, one of which is a relatively positioned &lt;code&gt;div&lt;/code&gt; child. When I use a percentage-based width on the child &lt;code&gt;div&lt;/code&gt;, it collapses to &lt;code&gt;0px&lt;/code&gt;. If I use pixel width, it works. If the parent is relatively positioned, the percentage width on the child works. &lt;/p&gt;&lt;x&gt;&lt;p&gt;Is there something I'm missing here? &lt;/p&gt;&lt;x&gt;Is there an easy fix for this besides the pixel-based width on the child? &lt;/p&gt;&lt;x&gt;Is there an area of the CSS specification that covers this?&lt;/p&gt;&lt;x&gt;" OwnerUserId="9" LastEditorUserId="243557" LastEditorDisplayName="Rich B" LastEditDate="2013-10-04T01:14:10.373" LastActivityDate="2013-11-20T04:16:38.813" Title="Why doesn't the percentage width child in absolutely positioned parent work" Tags="&lt;html&gt;&lt;css&gt;&lt;div&gt;&lt;internet-explorer-7&gt;" AnswerCount="5" CommentCount="12" FavoriteCount="7" />
    <row Id="7" PostTypeId="2" ParentId="4" CreationDate="2008-07-31T22:17:57.883" Score="193" Body="&lt;p&gt;An explicit cast to double isn't necessary.&lt;/p&gt;&lt;x&gt;&lt;p&gt;double trans = (double)trackBar1.Value / 5000.0;&lt;/p&gt;&lt;x&gt;&lt;code&gt;double trans = trackBar1.Value / 5000.0;double trans = trackBar1.Value / 5000d;&lt;/code&gt; (or &lt;code&gt;double trans = 5000d*1l;&lt;/code&gt;) is sufficient.&lt;/p&gt;&lt;x&gt;&lt;p&gt;&lt;code&gt;double trans = trackBar1.Value / 5000d;&lt;/code&gt;&lt;/p&gt;&lt;x&gt;" OwnerUserId="9" LastEditorUserId="967315" LastEditDate="2012-10-14T11:50:16.703" LastActivityDate="2012-10-14T11:50:16.703" CommentCount="0" />
    <row Id="8" PostTypeId="1" AcceptedAnswerId="162" CreationDate="2008-07-31T23:33:19.290" Score="35" ViewCount="2834" Body="&lt;p&gt;Are there any conversion tools for porting from &lt;strong&gt;Visual J#&lt;/strong&gt; code to &lt;strong&gt;C#&lt;/strong&gt;?&lt;/p&gt;&lt;x&gt;" OwnerUserId="9" LastEditorUserId="464552" LastEditorDisplayName="Rich B" LastEditDate="2012-11-12T17:24:03.030" LastActivityDate="2013-03-29T06:32:49.430" Title="Tool for Converting Visual J# code to C#?" Tags="&lt;code&gt;&lt;code-generation&gt;&lt;j#&gt;&lt;visual-j#&gt;" AnswerCount="3" CommentCount="1" FavoriteCount="1" ClosedDate="2013-06-03T04:00:25.587" />
    <row Id="9" PostTypeId="1" AcceptedAnswerId="1484" CreationDate="2008-07-31T23:40:59.743" Score="695" ViewCount="161688" Body="&lt;p&gt;Given a User Id, &lt;code&gt;User&lt;/code&gt; representing a person's birthday, how do I calculate their age? &lt;/p&gt;&lt;x&gt;" OwnerUserId="1" LastEditorUserId="212" LastEditorDisplayName="Rich B" LastEditDate="2013-11-05T13:48:37.393" LastActivityDate="2013-12-21T04:53:09.307" Title="How do I calculate someone's age in C#?" Tags="&lt;code&gt;&lt;datetime&gt;&lt;/code&gt; value, how do I display relative time, like &lt;code&gt;ulngt;2 hours ago&lt;/ulngt;&lt;code&gt;3 days ago&lt;/code&gt; etc? Et cetera?&lt;/p&gt;&lt;x&gt;-- More -->

```

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

The screenshot shows the Eclipse IDE interface with the BinningDriver.java file open in the editor. The code defines a main() method that sets up a Job with a Mapper class and multiple outputs. A yellow box highlights the line where the number of reduce tasks is set to 0.

```
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access Java Debug PyDev
BinningDriver.java BinningMapper.java
package com.hadoop.training.examples.binning;

import org.apache.hadoop.conf.Configuration;
public class BinningDriver {
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        String[] otherArgs = new GenericOptionsParser(conf, args)
            .getRemainingArgs();
        if (otherArgs.length != 2) {
            System.err.println("Usage: Binning <posts> <outdir>");
            System.exit(1);
        }
        Job job = new Job(conf, "Binning");
        job.setJarByClass(BinningDriver.class);
        job.setMapperClass(BinningMapper.class);
        job.setNumReduceTasks(0); MultipleOutputs.addNamedOutput(job, "seqbins", SequenceFileOutputFormat.Text, IntWritable);
        TextInputFormat.setInputPaths(job, new Path(otherArgs[0]));
        FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
        // Configure the MultipleOutputs by adding an output called "bins"
        // With the proper output format and mapper key/value pairs
        MultipleOutputs.addNamedOutput(job, "textbins", TextOutputFormat.class,
            NullWritable.class, Text.class);
        LazyOutputFormat.setOutputFormatClass(job, TextOutputFormat.class);
        job.setConfiguration(hadoopConf);
    }
}
```

Number of reduce task must be set to 0

The screenshot shows the Eclipse IDE interface with the BinningMapper.java file open in the editor. A yellow box highlights the setup() method, which is annotated with @Override and @SuppressWarnings("unchecked").

```
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access Java Debug PyDev
BinningDriver.java BinningMapper.java
public class BinningMapper extends Mapper<Object, Text, Text, NullWritable> {
    private MultipleOutputs<NullWritable, Text> multipleOutputs = null;
    @SuppressWarnings({"unchecked", "rawtypes"})
    @Override
    protected void setup(Context context) {
        // Create a new MultipleOutputs using the context object
        multipleOutputs = new MultipleOutputs(context);
    }
    @Override
    protected void map(Object key, Text value, Context context)
        throws IOException, InterruptedException {
        // Parse the input string into a nice map
        Map<String, String> parsed = MRDPUtility.transformXmlToMap(value
            .toString());
        String rawtags = parsed.get("Tags");
        if (rawtags == null) {
            return;
        }
        // Tags are delimited by ><, i.e. <tag1><tag2><tag3>
        String[] tagTokens = StringEscapeUtils.unescapeHtml(rawtags).split(
            "><");
        // For each tag
        for (String tag : tagTokens) {
    }
}
```

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

The diagram illustrates the execution flow of Map and Reduce tasks across three splits (split1, split2, split3). Each split has its own Record Reader and Map Task. The Map Task consists of four phases: setup, Map func, cleanup, and Map Task. The Reduce Task also consists of four phases: setup, Reduce func, cleanup, and Reduce Task. Initialization arrows point from the Map Task and Reduce Task boxes to their respective components. Annotations provide details: 'Initialization' points to the Map Task; 'Called only once for each MapTask' points to the Map Task box; 'Before the Map function is called for the first key and value' points to the Map func phase; 'Initialization' points to the Reduce Task; 'Called only once for each Reduce Task' points to the Reduce Task box; and 'Before the Reduce function is called for the first key and values' points to the Reduce func phase.

```

public class BinningMapper extends MapReduceMapper {
    private MultipleOutputs<NullWritable> multipleOutputs;
    @SuppressWarnings({"unchecked", "rawtypes"})
    @Override
    protected void setup(Context context) throws IOException, InterruptedException {
        // Create a new MultipleOutputs
        multipleOutputs = new MultipleOutputs();
    }
    @Override
    protected void map(Object key, Text value, Context context)
            throws IOException, InterruptedException {
        // Parse the input string into
        Map<String, String> parsed = MRInputFormat.parseText(value);
        String rawtags = parsed.get("Tags");
        if (rawtags == null) {
            return;
        }
        // Tags are delimited by ><, i.e. <tag1><tag2><tag3>
        String[] tagTokens = StringEscapeUtils.unescapeHtml(rawtags).split("><");
        // For each tag
        for (String tag : tagTokens) {
            ...
        }
    }
}

```

This screenshot shows the BinningDriver.java code in an IDE. The code is identical to the one shown in the first screenshot, with the addition of a closing brace at the end of the main block. The IDE interface includes tabs for BinningDriver.java and BinningMapper.java, and various toolbars and status bars at the bottom.

```

    }
}

// Tags are delimited by ><, i.e. <tag1><tag2><tag3>
String[] tagTokens = StringEscapeUtils.unescapeHtml(rawtags).split("><");

// For each tag
for (String tag : tagTokens) {
    // Remove any > or < from the token
    String groomed = tag.replaceAll(">|<", "").toLowerCase();

    // If this tag is one of the following, write to the named bin
    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");
    }

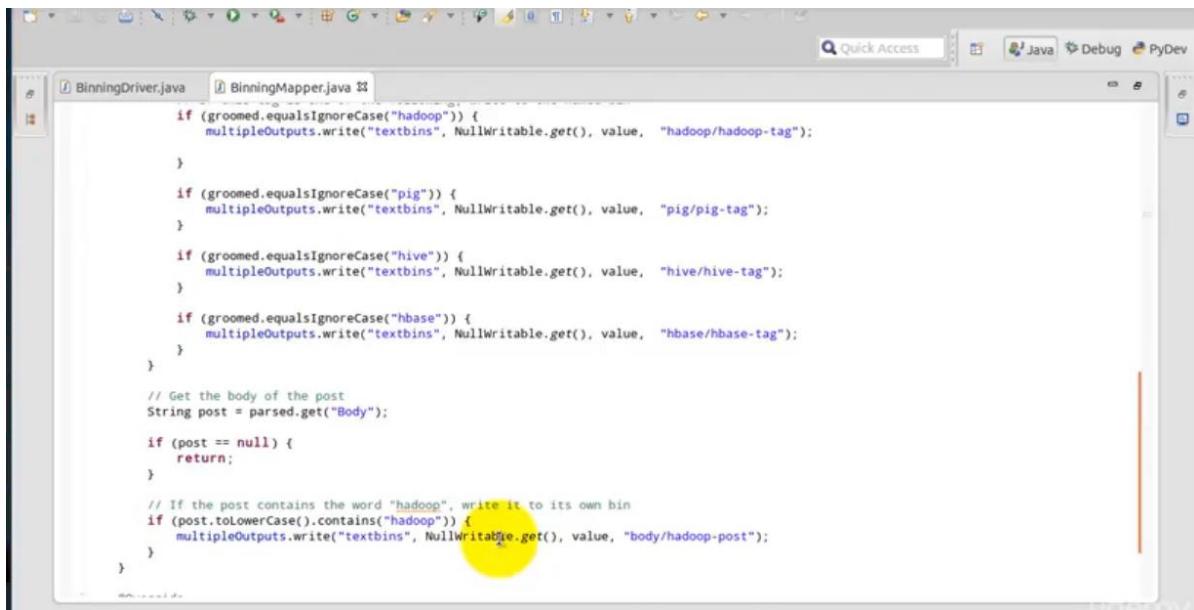
    if (groomed.equalsIgnoreCase("hive")) {
        multipleOutputs.write("textbins", NullWritable.get(), value, "hive/hive-tag");
    }

    if (groomed.equalsIgnoreCase("hbase")) {
        multipleOutputs.write("textbins", NullWritable.get(), value, "hbase/hbase-tag");
    }
}

// Get the body of the post
String nntc = narsad.getFirst("NnDv");

```

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



The screenshot shows the Eclipse IDE interface with two tabs open: "BinningDriver.java" and "BinningMapper.java". The "BinningMapper.java" tab is active, displaying the following Java code:

```
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

Browse Directory

/user/hduser/hive/stackoverflow/posts

Permission	Owner	Group	Size	Replication	Block Size	Name
-rW-r--r--	hduser	supergroup	672.91 MB	3	128 MB	posts

Hadoop, 2013.

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

Hadoop job_201502050923_0001 on nameNode

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	0.00%	6	3	3	0	0	0 / 0
reduce	0.00%	0	0	0	0	0	0 / 0

Job Counters

	Counter	Map	Reduce	Total
Launched map tasks		0	0	3
Data-local map tasks		0	0	3
Total time spent by all maps in occupied slots (ms)		0	0	12,271
Map-Reduce Framework	Spilled Records	0	0	0

Map Completion Graph - close

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

Output file

The screenshot shows a web browser window titled "Browsing HDFS" with the URL "namenode.cdh-cluster.com:50070/explorer.html#/user/hduser/hive/stackoverflow/posts/output". The main content is a table titled "Browse Directory" listing files in the "/user/hduser/hive/stackoverflow/posts/output" directory. The table columns are: Permission, Owner, Group, Size, Replication, Block Size, and Name. The files listed are: _SUCCESS (0 B, 3 replication, 128 MB block size), _logs (0 B, 0 replication, 0 B block size), body (0 B, 0 replication, 0 B block size), hadoop (0 B, 0 replication, 0 B block size), hbase (0 B, 0 replication, 0 B block size), hive (0 B, 0 replication, 0 B block size), and log (0 B, 0 replication, 0 B block size). A yellow circle highlights the "log" file entry.

Permission	Owner	Group	Size	Replication	Block Size	Name
-rW-f--f--	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	log

Hadoop, 2013.

The screenshot shows a web browser window titled "Browsing HDFS" with the URL "namenode.cdh-cluster.com:50070/explorer.html#/user/hduser/hive/stackoverflow/posts/output/hadoop". The main content is a table titled "Browse Directory" listing files in the "/user/hduser/hive/stackoverflow/posts/output/hadoop" directory. The table columns are: Permission, Owner, Group, Size, Replication, Block Size, and Name. The files listed are: hadoop-tag-m-00000 (3.35 KB, 3 replication, 128 MB block size), hadoop-tag-m-00001 (24.75 KB, 3 replication, 128 MB block size), hadoop-tag-m-00002 (62.7 KB, 3 replication, 128 MB block size), hadoop-tag-m-00003 (106.98 KB, 3 replication, 128 MB block size), hadoop-tag-m-00004 (179.25 KB, 3 replication, 128 MB block size), and hadoop-tag-m-00005 (54.63 KB, 3 replication, 128 MB block size). A yellow circle highlights the "hadoop-tag-m-00004" file entry.

Permission	Owner	Group	Size	Replication	Block Size	Name
-rW-f--f--	hduser	supergroup	3.35 KB	3	128 MB	hadoop-tag-m-00000
-rW-f--f--	hduser	supergroup	24.75 KB	3	128 MB	hadoop-tag-m-00001
-rW-f--f--	hduser	supergroup	62.7 KB	3	128 MB	hadoop-tag-m-00002
-rW-f--f--	hduser	supergroup	106.98 KB	3	128 MB	hadoop-tag-m-00003
-rW-f--f--	hduser	supergroup	179.25 KB	3	128 MB	hadoop-tag-m-00004
-rW-f--f--	hduser	supergroup	54.63 KB	3	128 MB	hadoop-tag-m-00005

Hadoop, 2013.

C 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-f--f--	hduser	supergroup	1.35 KB	3	128 MB	hbase-tag-m-00000
-rW-f--f--	hduser	supergroup	4.87 KB	3	128 MB	hbase-tag-m-00001
-rW-f--f--	hduser	supergroup	12.32 KB	3	128 MB	hbase-tag-m-00002
-rW-f--f--	hduser	supergroup	17.26 KB	3	128 MB	hbase-tag-m-00003
-rW-f--f--	hduser	supergroup	15.66 KB	3	128 MB	hbase-tag-m-00004
-rW-f--f--	hduser	supergroup	5.8 KB	3	128 MB	hbase-tag-m-00005

Hadoop, 2013.

Open Save Undo Redo Cut Copy Paste Find Replace

hbase-tag-m-00000

```
<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 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><div style="text-align: right;">OwnerUserId="2588" OwnerDisplayName="mrhahn" LastActivityDate="2013-02-05T03:54:59.697" Title="How does Hive compare to HBase?"</div><div style="margin-top: 10px;">Tags="hadoop"&gt;&lt;hbase&gt;&lt;hive&gt;" AnswerCount="5" FavoriteCount="17" />
<row Id="48041" PostTypeId="1" AcceptedAnswerId="48112" CreationDate="2008-09-07T02:14:07.543" Score="18" ViewCount="17507" Body="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? OwnerUserId="4962" OwnerDisplayName="zecharlahs" LastActivityDate="2013-06-23T11:02:02.460" Title="Hbase / Hadoop Query Help" Tags="hadoop"&gt;&lt;hbase&gt;" AnswerCount="6" FavoriteCount="5" />
```

Plain Text Tab Width: 8 Ln 1, Col 361 INS

Reduce Side Join Pattern

Join Pattern

Join in MapReduce is similar to join in SQL

In SQL, 2 or more tables are joined using foreign key

In MapReduce, 2 or more datasets are joined using a common key

4 types of join

1. Reduce Side Join
2. Map Side Join or Replicated Join
3. Composite Join
4. **Cartesian Product**



1. Reduce Side Join

Join is done in the Reducer

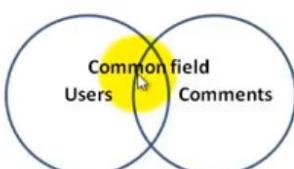
Reduce Side Join supports

Inner Join,
Outer Join,
Left Outer Join,
Right Outer Join,
Anti-Join



SQL Joins

Problem: Given user dataset and comments dataset, enrich each comment with the information about the user who created the comment.

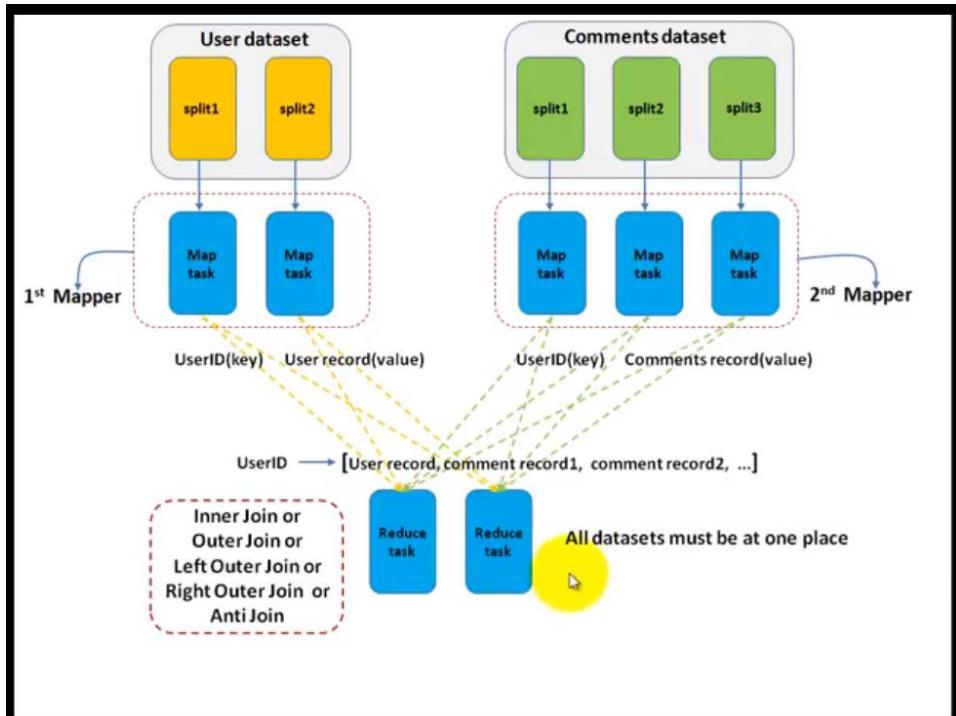


```

hduser@nameNode:~/hadoop/data/stackoverflow
<?xml version="1.0" encoding="utf-8"?>
<users>
  <row Id="-1" Reputation="1" CreationDate="2008-07-31T00:00:00.000" DisplayName="Community" LastAccessDate="2008-08-26T00:16:53.810" WebsiteUrl="http://stackoverflow.com" Location="on the server farm" AboutMe="I'm not really a person.&lt;/p&gt;&lt;x&gt;&lt;p&gt;I'm a background process that helps keep the site clean!&lt;/p&gt;&lt;x&gt;&lt;p&gt;I do things like&lt;/p&gt;&lt;x&gt;&lt;p&gt;Randomly poke some random people with unanswered questions every hour so they get some attention&lt;/p&gt;&lt;x&gt;&lt;p&gt;Own community questions and answers so nobody gets unnecessary reputation from them&lt;/p&gt;&lt;x&gt;&lt;p&gt;Own downvotes on spam/evil posts that get permanently deleted&lt;/p&gt;&lt;x&gt;&lt;p&gt;Own suggested edits from anonymous users&lt;/p&gt;&lt;x&gt;&lt;p&gt;Views="649" UserAccountId="1" />
  <row Id="1" Reputation="29330" CreationDate="2008-07-31T14:22:31.287" DisplayName="Jeff Atwood" LastAccessDate="2014-01-18T05:39:39.280" WebsiteUrl="http://www.codinghorror.com/blog/" Location="El Cerrito, CA" AboutMe="About Me:&lt;p&gt;&lt;a href=&quot;http://www.codinghorror.com/blog/archives/001169.html&quot; rel=&quot;nofollow&quot;&gt;Stack Overflow Valued Associate #000001&lt;/a&gt;&lt;/p&gt;&lt;x&gt;&lt;p&gt;Wondering how our software development process works? &lt;a href=&quot;http://www.youtube.com/watch?v=08xQLGWTsg&quot; rel=&quot;nofollow&quot;&gt;Take a look!&lt;/a&gt;&lt;/p&gt;&lt;x&gt;&lt;p&gt;Find me &lt;a href=&quot;http://

```

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



File Edit Source Refactor Navigate Search Project Run Window Help

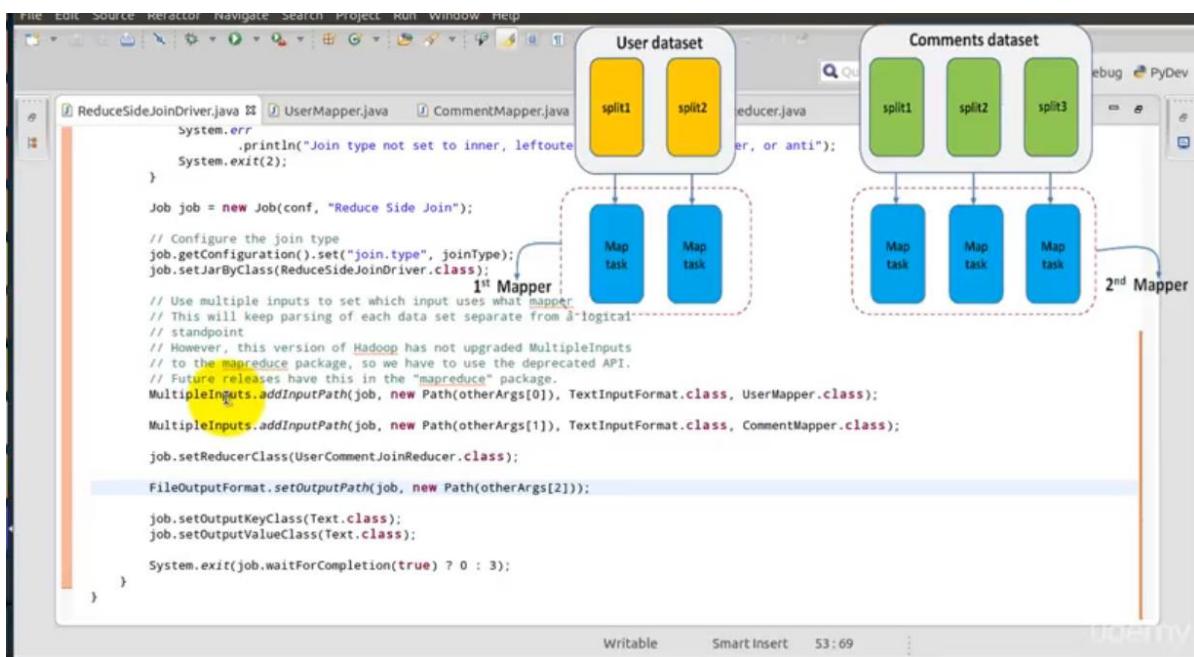
Quick Access Java Debug PyDev

```

    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);
            job.setJarByClass(ReduceSideJoinDriver.class);
        }
    }

```

Writable SmartInsert 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, leftouter, rightouter, 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(UserComment.JoinReducer.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

Comments dataset

split1 split2

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

File Edit Source Refactor Navigate Search Project Run Window Help

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 = MRUTILS.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

Comments dataset

split1 split2

split1 split2 split3

Map task Map task

Map task Map task Map task

UserID(key) Userrecord(value)

UserID(key) Commentsrecord(value)

UserID —> [User record, commentrecord1, comment record2, ...]

Reduce task Reduce task

which value is from which dataset?

```

    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 = WMRDUtils.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("X" + value.toString());
            context.write(outkey, outvalue);
        }
    }

```

which value is from which dataset?

```

    package com.hadoop.training.examples.join;

    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)));
                }
            }
        }
    }

```

The screenshot shows the Eclipse IDE interface with the following details:

- Toolbar:** Quick Access, Java, Debug, PyDev.
- Open Editors:** ReduceSideJoinDriver.java (active), UserMapper.java, CommentMapper.java, UserCommentJoinReducer.java.
- Code View:** The code for the `ReduceSideJoinDriver` class is displayed. It includes logic for binning values based on a tag character ('A' or 'B'), executing join logic for different join types (inner, leftouter), and handling list operations like `listA.add(new Text(t.toString().substring(1)))`.

```
// 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()) {

```

```
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);
                }
            }
        }
    }
}
```

```
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("right")) {
    // 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) {
            context.write(new Text(""), B);
        }
    }
}
```

```
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) {
            context.write(new Text(""), B);
        }
    }
}
```

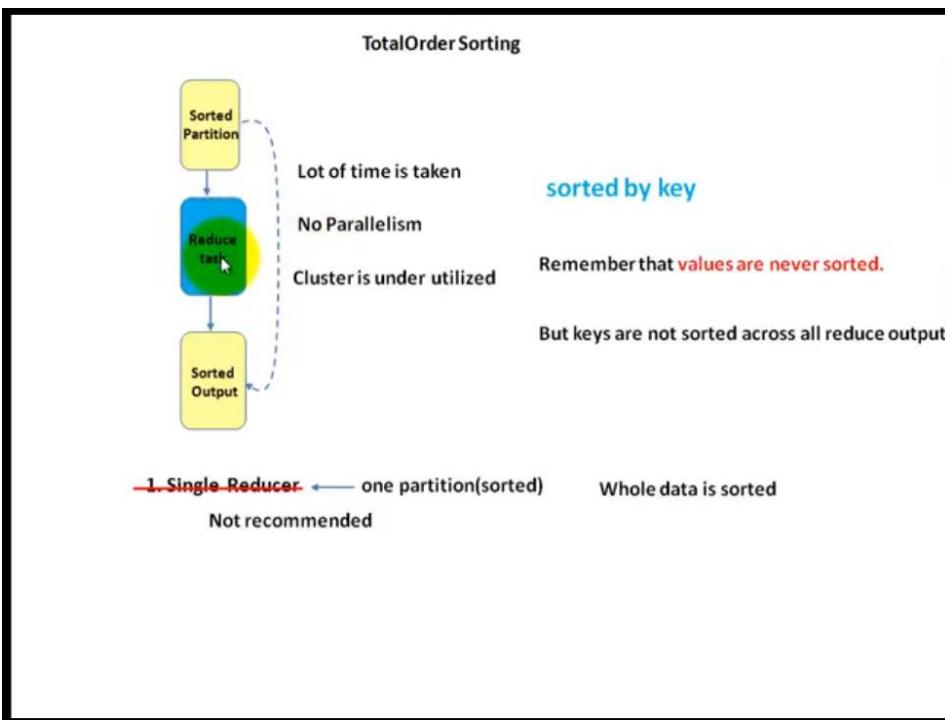
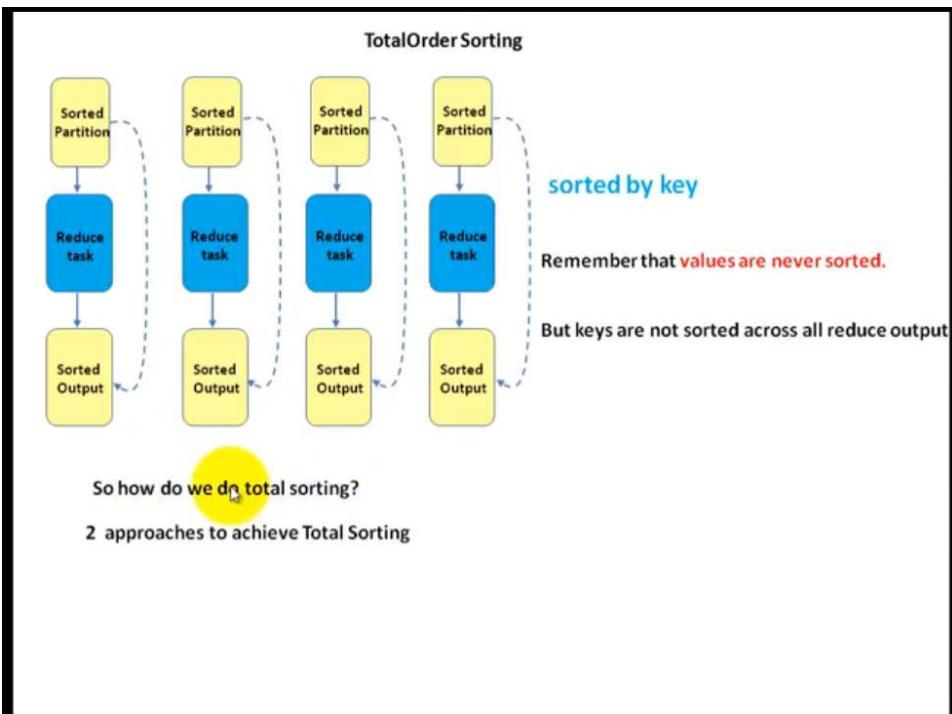
1. Reduce side join is done in the reducer

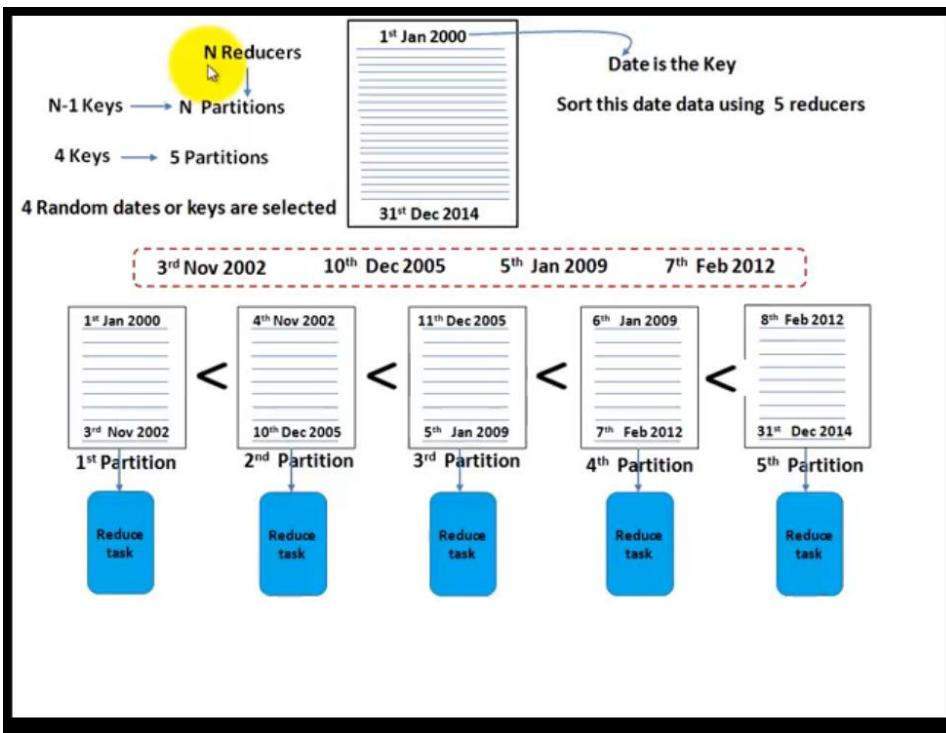
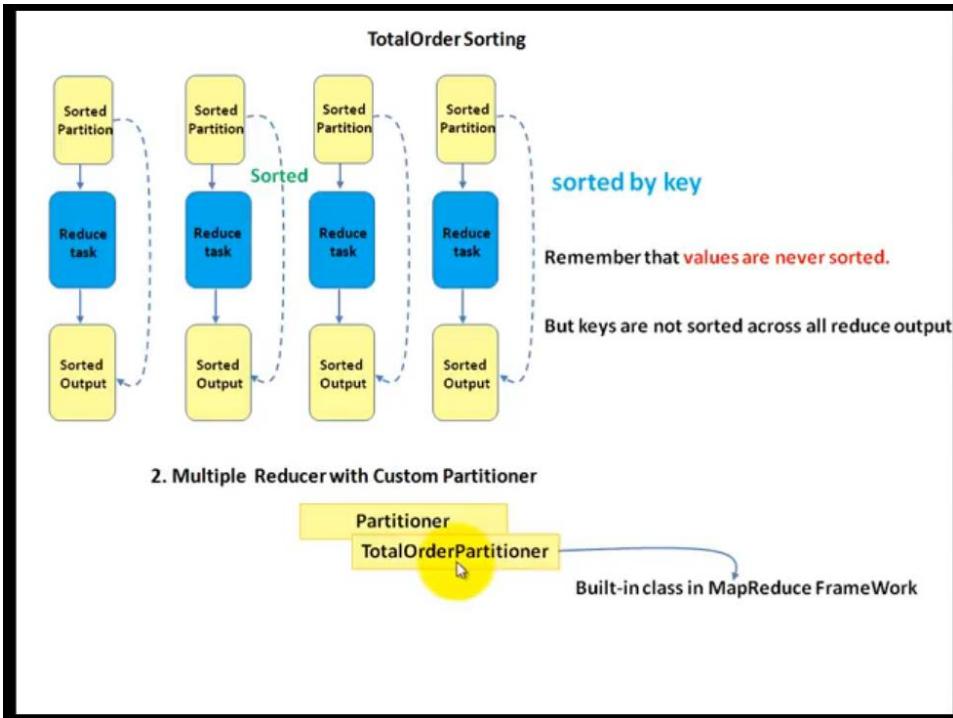
2. Supports InnerJoin, OuterJoin, Left OuterJoin, Right OuterJoin, AntiJoin

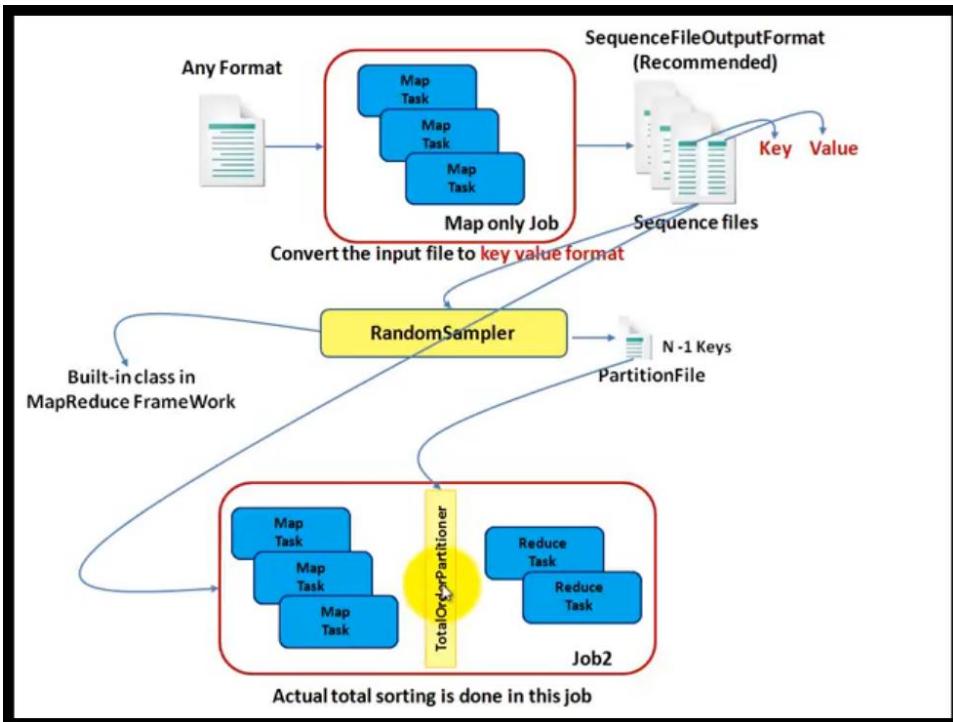
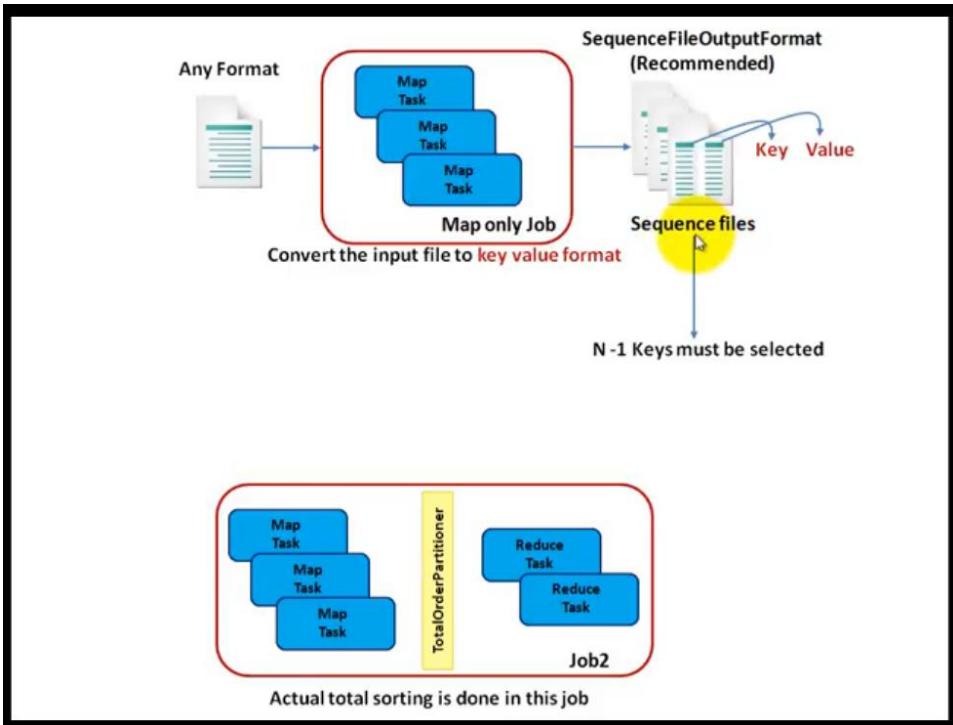
3. High Network Bandwidth & High Memory

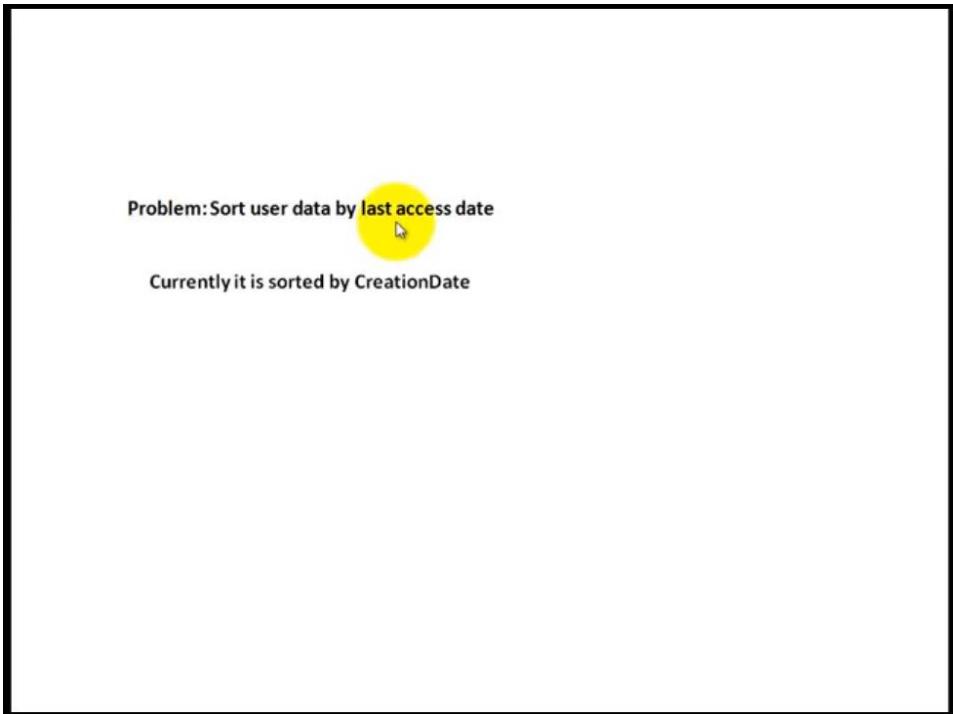
4. All datasets are large

Total Order Sorting Pattern









The image shows an IDE interface with several tabs at the top: TotalOrderSorting.java, FormatConverterMapper.java, LastAccessDateTotalSortMapper.java, and LastAccessDateTotalSortReducer.java. The TotalOrderSorting.java tab is active.

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

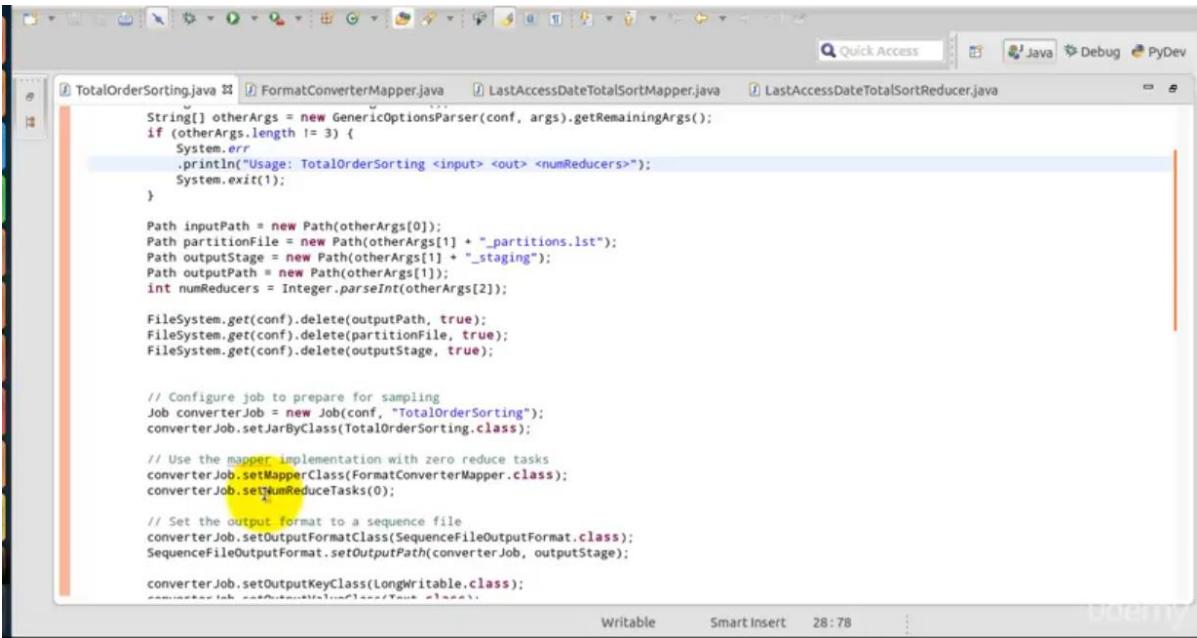
import java.io.IOException;
public class TotalOrderSorting {
    public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {
        Configuration conf = new Configuration();
        String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
        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);
    }
}
```



```
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]);

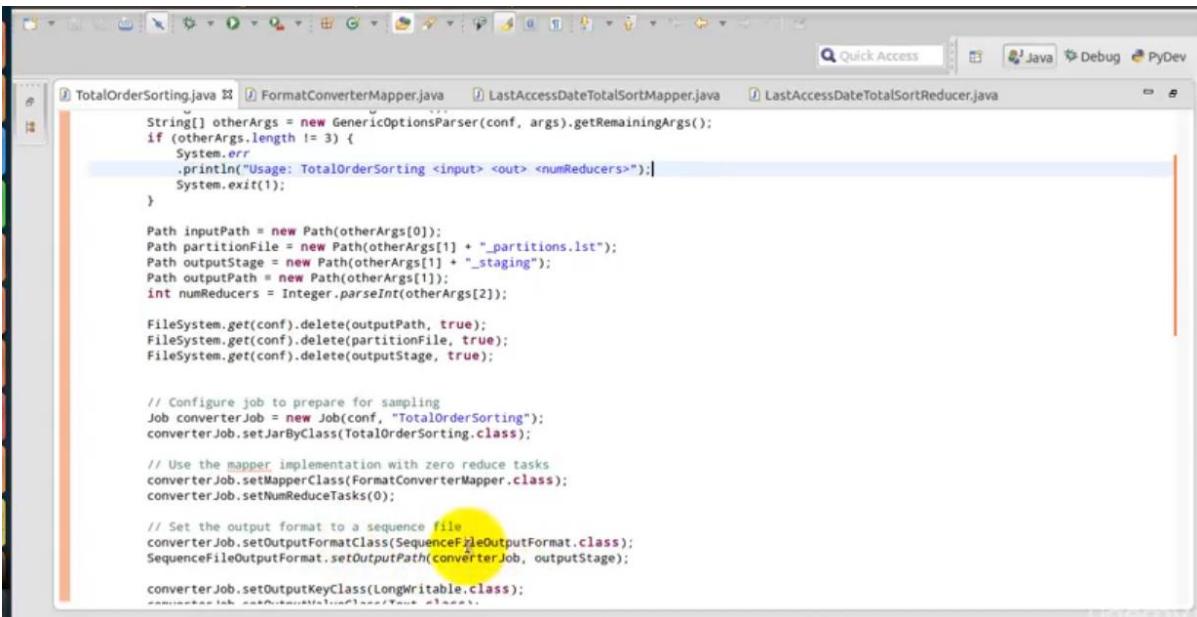
FileSystem.get(conf).delete(outputPath, true);
FileSystem.get(conf).delete(partitionFile, true);
FileSystem.get(conf).delete(outputStage, true);

// 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);
```



```
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);
```



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

import java.io.IOException;

public class FormatConverterMapper extends Mapper<LongWritable, Text, LongWritable, Text> {

    private LongWritable outKey = new LongWritable();
    private final static SimpleDateFormat dateFmt = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSS");

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

        Map<String, String> parsed = MRDPUtility.transformXmlToMap(value.toString());
        String accessDate = parsed.get("LastAccessDate");
        String userId = parsed.get("id");
        String displayName = parsed.get("DisplayName");
        String strValue = userId + " " + displayName;
        if (accessDate != null) {

            try {
                outKey.set(dateFmt.parse(accessDate).getTime());
            } catch (ParseException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }

        //outKey.set(accessDate);
        context.write(outKey, new Text(strValue));
    }
}
```



```
// Submit the job and get completion code.
int code = converterJob.waitForCompletion(true) ? 0 : 1;

if (code == 0) {
    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, exemplars);
}
```

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



```
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 LastAccessDateTotalSortMapper extends Mapper<LongWritable, Text, LongWritable, Text>{
    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        context.write(key, value);
    }
}
```

Writable Smart Insert 1:48



```
// Submit the job and get completion code.
int code = converterJob.waitForCompletion(true) ? 0 : 1;

if (code == 0) {
    InputSampler.Sampler<LongWritable, Text> sampler =
        new InputSampler.RandomSampler<LongWritable, Text>(0.1, 1000);

    Job totalSortJob = new Job(conf, "TotalOrderSorting-SortUserByLastAccessDate");
    totalSortJob.setJarByClass(TotalOrderSorting.class);
    totalSortJob.setMapperClass(LastAccessDateTotalSortMapper.class); totalSortJob.setMapperClass(Mapper.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", "");
}

Writable Smart Insert 65:80
```

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

The screenshot shows the Eclipse IDE interface with the Java perspective selected. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons for file operations like Open, Save, and Cut. The top status bar displays 'Quick Access' and tabs for Java, Debug, and PyDev. The main editor area shows the code for `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);
        }
    }
}
```

The code is syntax-highlighted with blue for keywords and red for annotations. A yellow circle highlights the `String date = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSS").format(d);` line.

The screenshot shows the Eclipse IDE interface with the Java perspective selected. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons for file operations like Open, Save, and Cut. The top status bar displays 'Quick Access' and tabs for Java, Debug, and PyDev. The main editor area shows the code for `TotalOrderSortingj.java`:

```
if (code == 0) {
    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;
}
```

The code is syntax-highlighted with blue for keywords and red for annotations. A yellow circle highlights the `InputSampler.writePartitionFile(totalSortJob, sampler);` line.

Browsing HDFS nameNode Hadoop Map/R... +

namenode.cdh-cluster.com:50030/jobtracker.jsp

Quick Links

Queue Name	State	Scheduling Information
default	running	N/A

Filter (jobid, Priority, User, Name)

Example: 'usersmith 3200' will filter by 'smith' only in the user field and '3200' in all fields

Running Jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information	Diagnostic Info
job_201502070904_0002	NORMAL	hduser	TotalOrderSorting-SortUserByLastAccessDate	100.00%	6	6	73.33%	5	3	NA	NA

Completed Jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information	Diagnostic Info
job_201502070904_0001	NORMAL	hduser	TotalOrderSorting	100.00%	6	6	100.00%	0	0	NA	NA

Retired Jobs

none

Local Logs

Log directory: Job Tracker History

namenode.cdh-cluster.com:50030/jobdetails.jsp?jobid=job_201502070904_0002&refresh=30

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

Output

Browse Directory

/user/hduser/hive/stackoverflow/users Go!

Permission	Owner	Group	Size	Replication	Block Size	Name
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	output
-rw-r--r--	hduser	supergroup	233 B	3	128 MB	output_partitions.lst
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	output_staging
-rw-r--r--	hduser	supergroup	662.61 MB	3	128 MB	stackoverflow.com-Users

Hadoop, 2013.

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

Browse Directory

/user/hduser/hive/stackoverflow/users/output_staging 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
-rw-r--r--	hduser	supergroup	15.79 MB	3	128 MB	part-m-00000
-rw-r--r--	hduser	supergroup	18.02 MB	3	128 MB	part-m-00001
-rw-r--r--	hduser	supergroup	19.38 MB	3	128 MB	part-m-00002
-rw-r--r--	hduser	supergroup	19.77 MB	3	128 MB	part-m-00003
-rw-r--r--	hduser	supergroup	15.12 MB	3	128 MB	part-m-00004
-rw-r--r--	hduser	supergroup	2.61 MB	3	128 MB	part-m-00005

Hadoop, 2013.

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

namenode.cdh-cluster.com:50070/explorer.html#/user/hduser/hive/stackoverflow/users

Google

Browse Directory

/user/hduser/hive/stackoverflow/users

Permission	Owner	Group	Size	Replication	Block Size	Name
drwxr-xf-x	hduser	supergroup	0 B	0	0 B	output
-rw-r--r--	hduser	supergroup	233 B	3	128 MB	output_partitions.lst
drwxr-xf-x	hduser	supergroup	0 B	0	0 B	output_staging
-rw-r--r--	hduser	supergroup	662.61 MB	3	128 MB	stackoverflow.com-Users

Hadoop, 2013.

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

And output folder is final output directory

namenode.cdh-cluster.com:50070/explorer.html#/user/hduser/hive/stackoverflow/users/output

Google

Browse Directory

/user/hduser/hive/stackoverflow/users/output

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
-rw-r--r--	hduser	supergroup	21.88 MB	3	128 MB	part-r-00000
-rw-r--r--	hduser	supergroup	21.96 MB	3	128 MB	part-r-00001
-rw-r--r--	hduser	supergroup	22.2 MB	3	128 MB	part-r-00002
-rw-r--r--	hduser	supergroup	22.97 MB	3	128 MB	part-r-00003
-rw-r--r--	hduser	supergroup	21.61 MB	3	128 MB	part-r-00004

Hadoop, 2013.

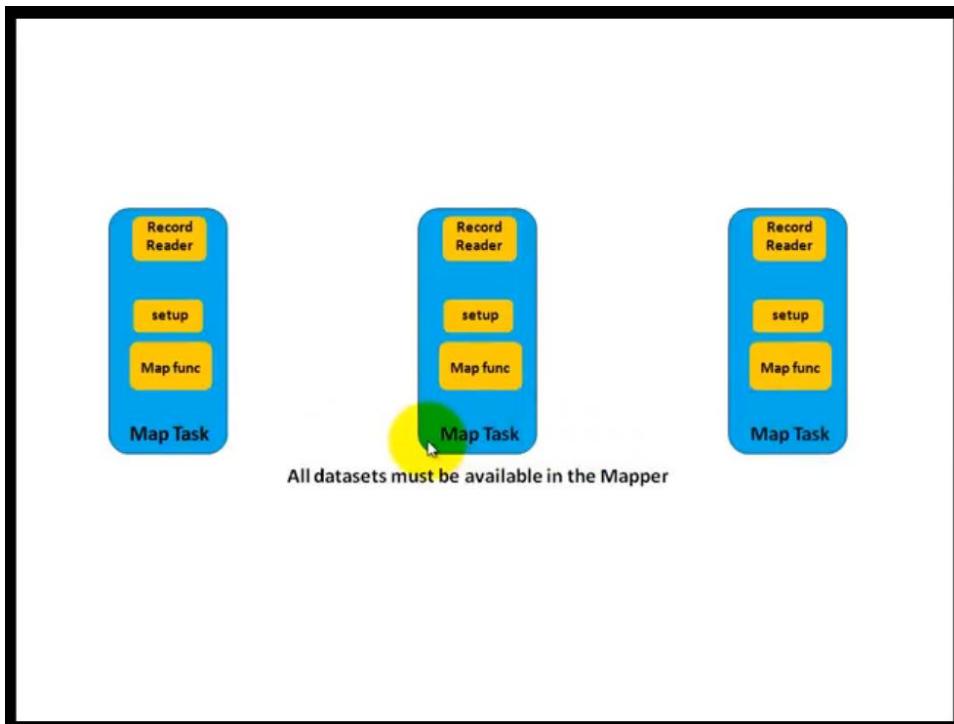
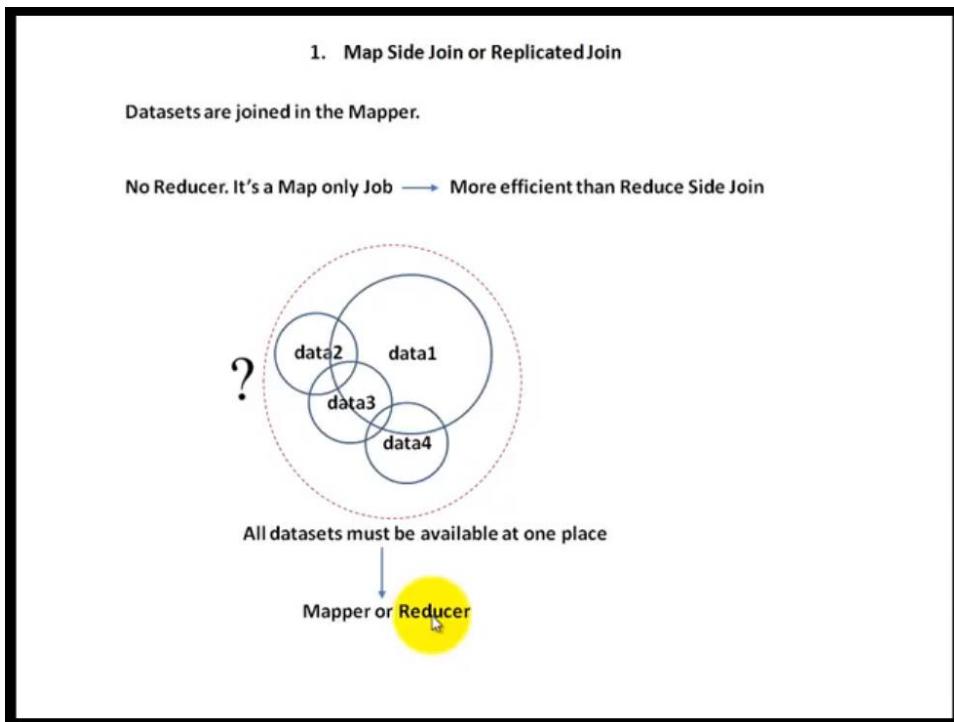
5 output files are created

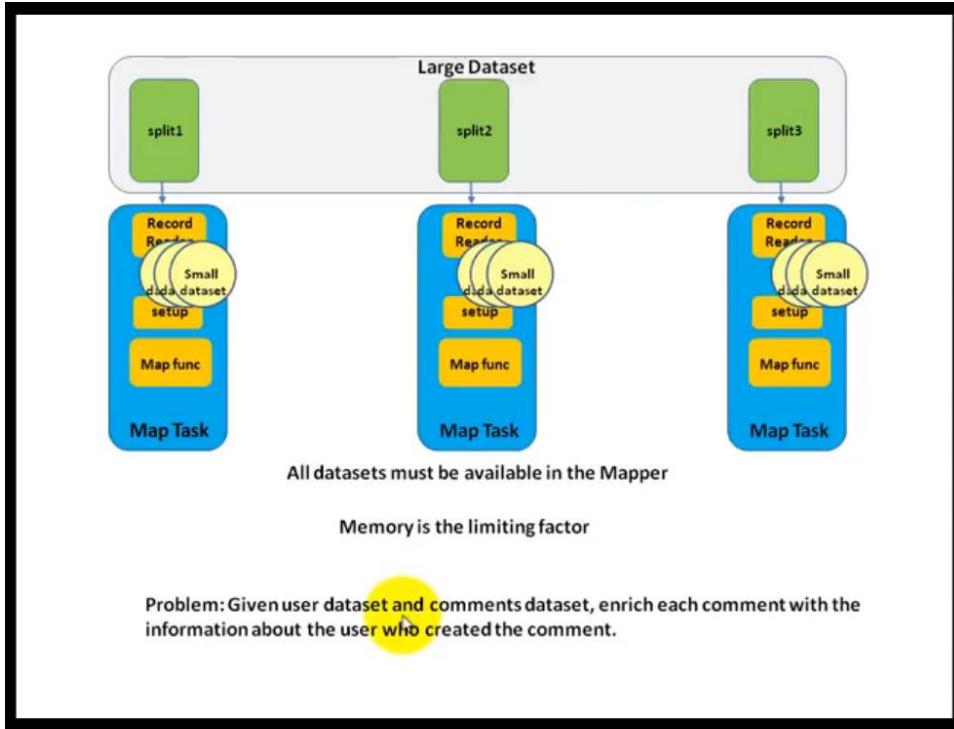
```
hduser@nameNode:~/workspace/examples          | hduser@nameNode:~/workspace/examples
15/02/07 22:19:14 INFO mapred.JobClient:      Combine input records=0
15/02/07 22:19:14 INFO mapred.JobClient:      Combine output records=0
15/02/07 22:19:14 INFO mapred.JobClient:      Reduce input groups=2725025
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/stackoverflow
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
2008-08-11T14:53:46.710 977 Bell
2008-08-11T16:03:38.370 1028 DannySmurf
2008-08-13T16:29:58.160 503 littlecharava
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 sheepgobleep
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

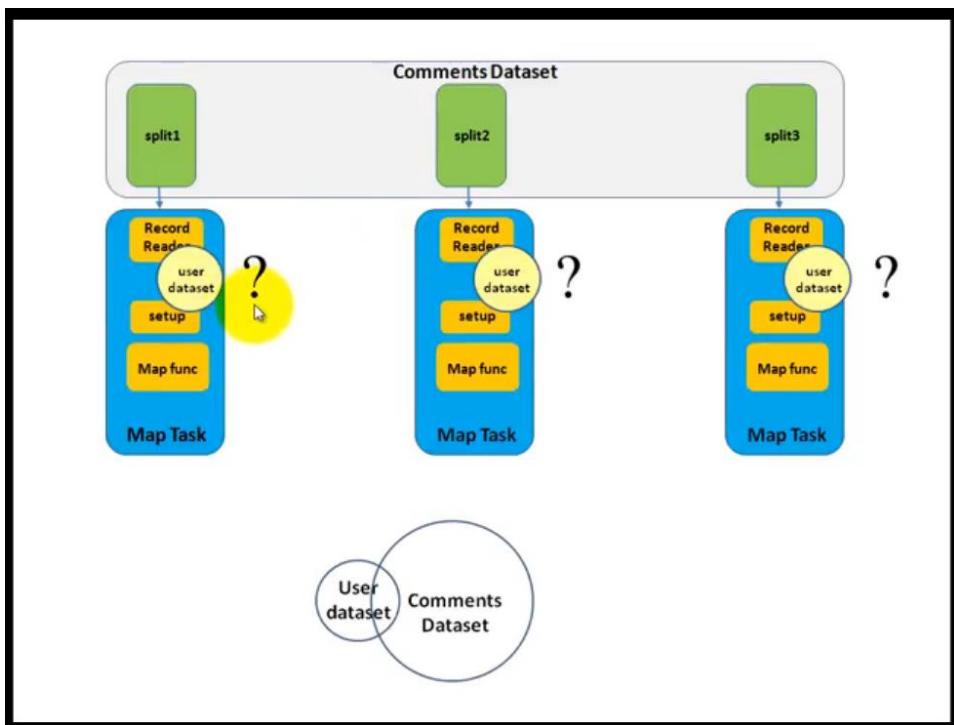




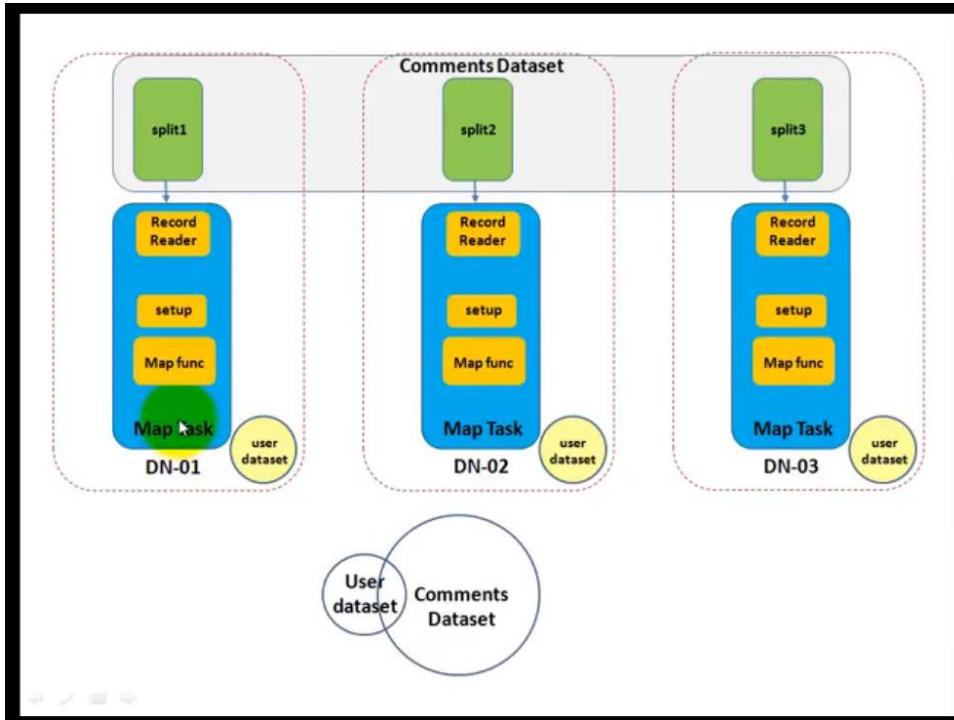
Large dataset read from split files and all smaller dataset must be copied to the memory of the map tasks

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

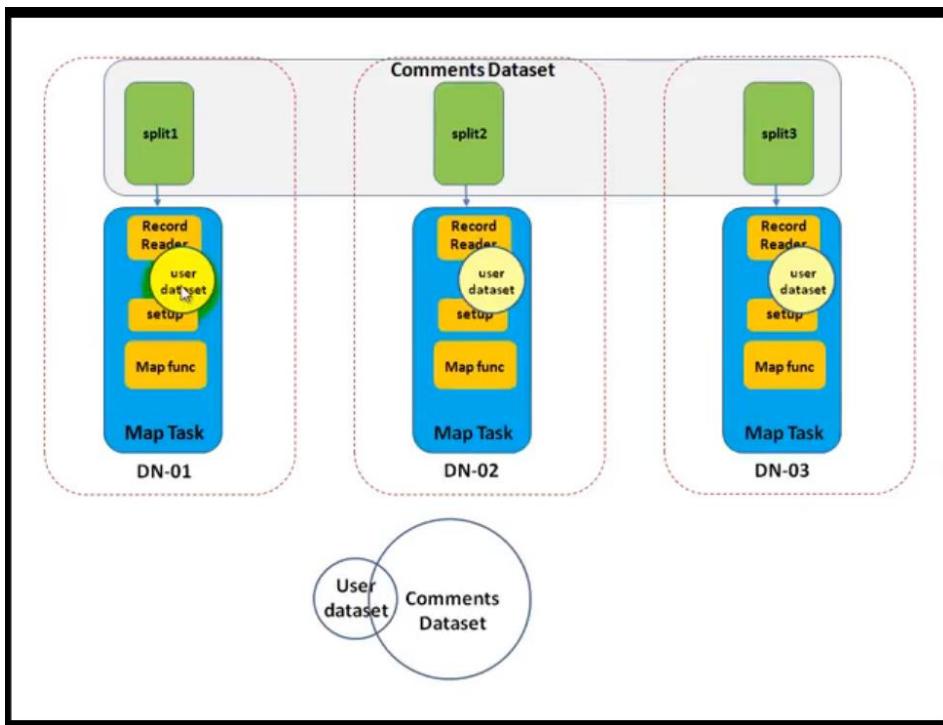
For stackoverflow problem we have smaller user dataset and large comment dataset hence we need to copy user dataset to the memory of the user dataset



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



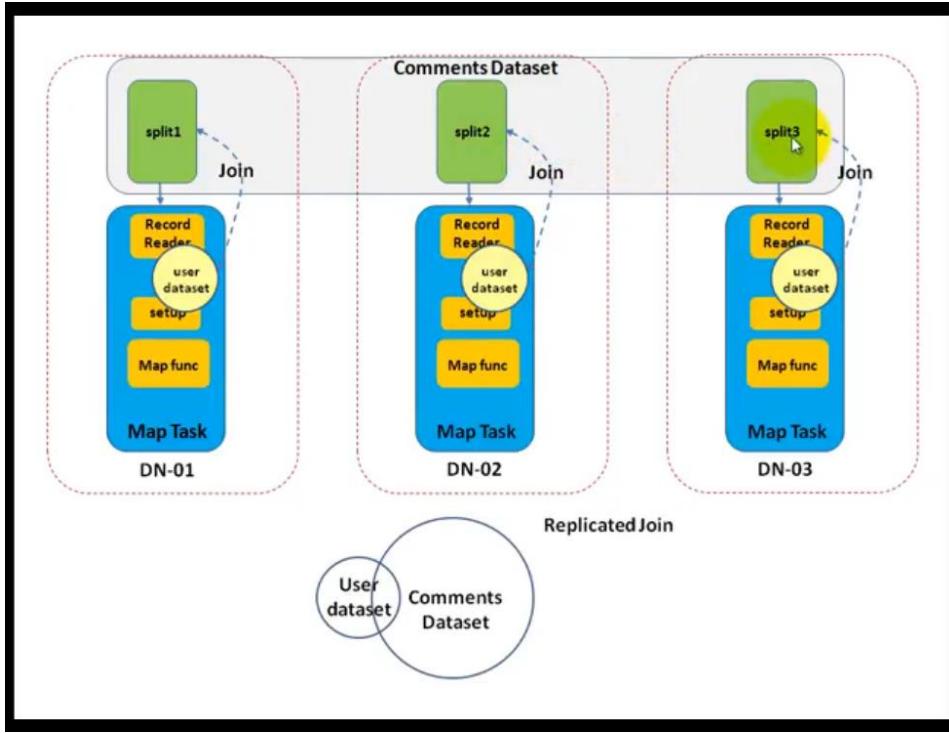
From there it has to be copied into memory of the map task



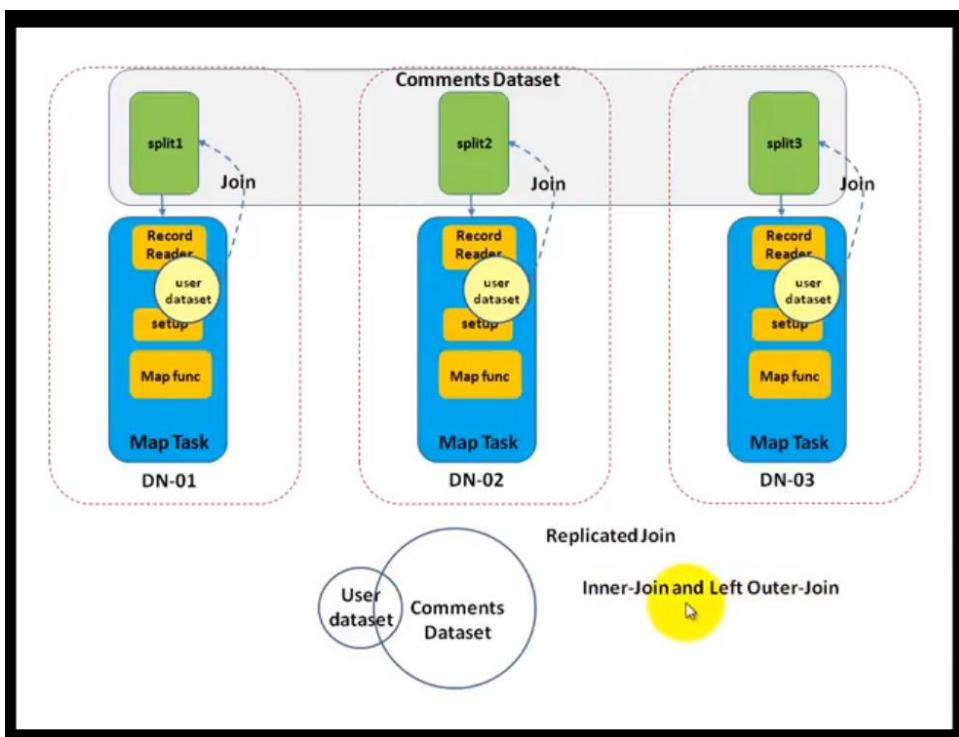
As a result each map tasks will have a copy of the user dataset

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

```

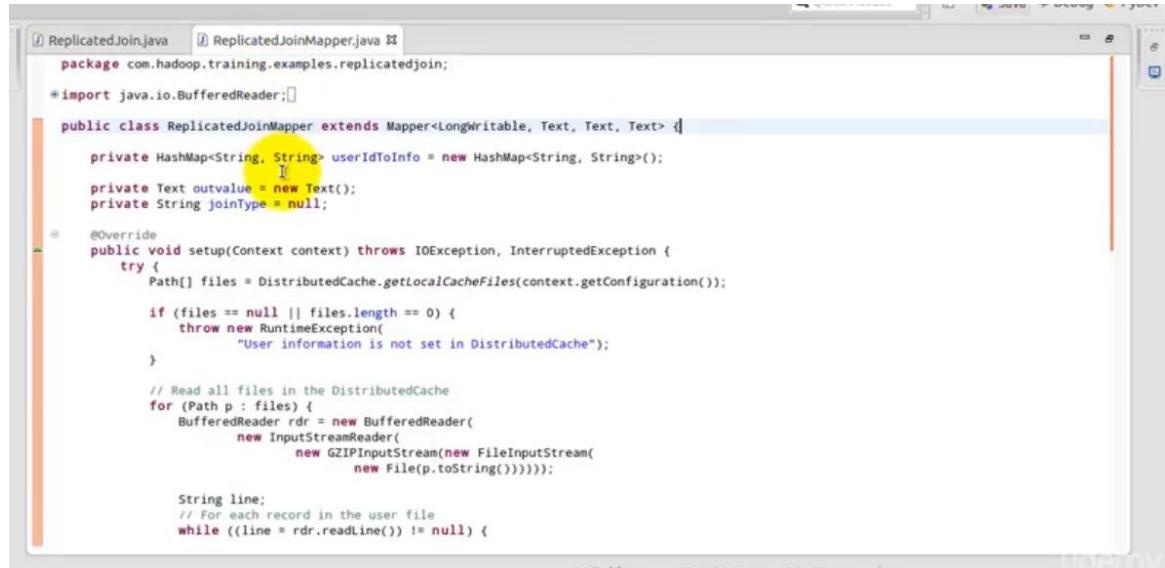
```

  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 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);
  job.setOutputValueClass(Text.class);
  // Configure the DistributedCache
  DistributedCache.addCacheFile(new Path(otherArgs[0]).toUri(),
    job.getConfiguration());
  System.exit(job.waitForCompletion(true) ? 0 : 1);
}

```

The diagram illustrates the execution of three Map Tasks (DN-01, DN-02, DN-03) across three splits (split1, split2, split3) of the Comments Dataset. Each Map Task is composed of a Record Reader, setup, and Map func steps. A user dataset is shown being processed by each Map Task.

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



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

import java.io.BufferedReader;

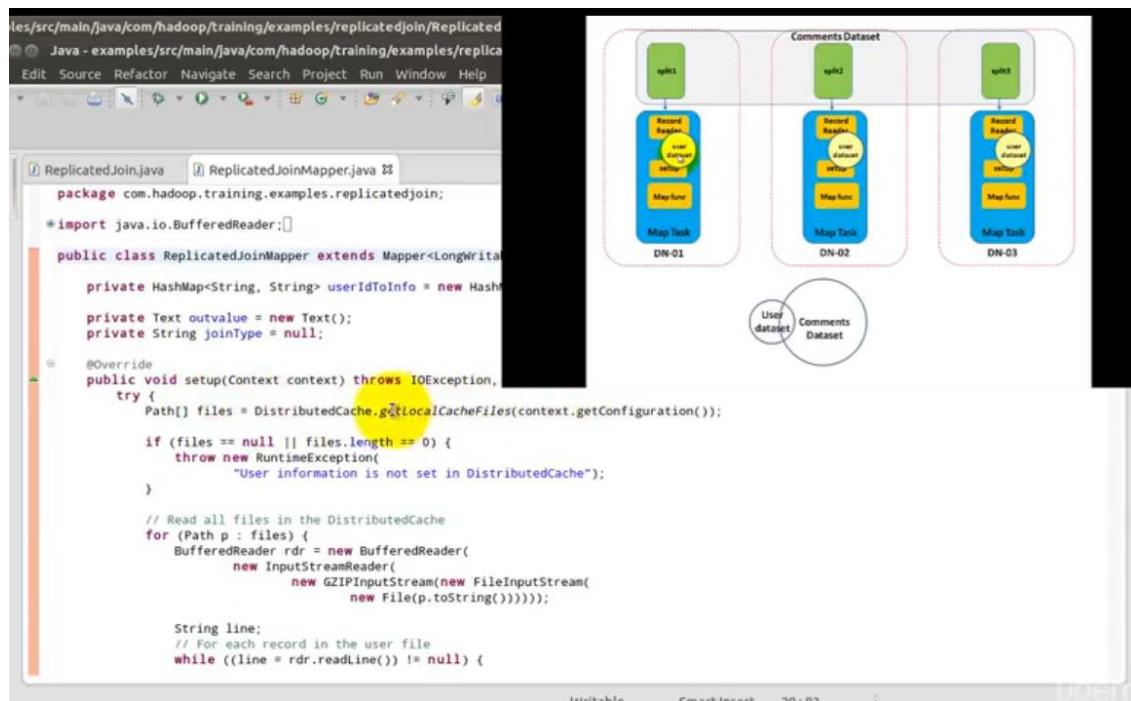
public class ReplicatedJoinMapper extends Mapper<LongWritable, Text, Text, Text> {

    private HashMap<String, String> userIdToInfo = new HashMap<String, String>();
    private Text outvalue = new Text();
    private String joinType = null;

    @Override
    public void setup(Context context) throws IOException, InterruptedException {
        try {
            Path[] files = DistributedCache.getLocalCacheFiles(context.getConfiguration());
            if (files == null || files.length == 0) {
                throw new RuntimeException(
                    "User information is not set in DistributedCache");
            }
            // Read all files in the DistributedCache
            for (Path p : files) {
                BufferedReader rdr = new BufferedReader(
                    new InputStreamReader(
                        new GZIPInputStream(new FileInputStream(
                            new File(p.toString())))));
                String line;
                // For each record in the user file
                while ((line = rdr.readLine()) != null) {

```

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

```
public void setup(Context context) throws IOException, InterruptedException {
    try {
        Path[] files = DistributedCache.getLocalCacheFiles(context.getConfiguration());
        if (files == null || files.length == 0) {
            throw new RuntimeException(
                "User information is not set in DistributedCache");
        }

        // Read all files in the DistributedCache
        for (Path p : files) {
            BufferedReader rdr = new BufferedReader(
                new InputStreamReader(new GZIPInputStream(new FileInputStream(
                    new File(p.toString())))));
            String line;
            // For each record in the user file
            while ((line = rdr.readLine()) != null) {
                // Get the user ID for this record
                Map<String, String> parsed = MRDPUtility
                    .transformXmlToMap(line);
                String userId = parsed.get("Id");

                if (userId != null) {
                    // Map the user ID to the record
                    userIdToInfo.put(userId, line);
                }
            }
            rdr.close();
        }
    }
}
```

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

```
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 = MRDPUtility
        .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(key, outvalue);
    } else if (joinType.equalsIgnoreCase("leftouter")) {
        // If we are doing a left outer join, output the record with an
        // empty value
        context.write(key, new Text(""));
    }
}
```

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

Comment dataset is read through input split

The screenshot shows a Java code editor with two tabs: "ReplicatedJoin.java" and "ReplicatedJoinMapper.java". The "ReplicatedJoinMapper.java" tab is active, displaying the following code:

```
    }
    // 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 = MRDPUtility.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(""));
    }
}
```

Annotations are placed next to the code:

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

At the bottom of the editor, there are status bars for "Writable", "SmartInsert", and the time "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

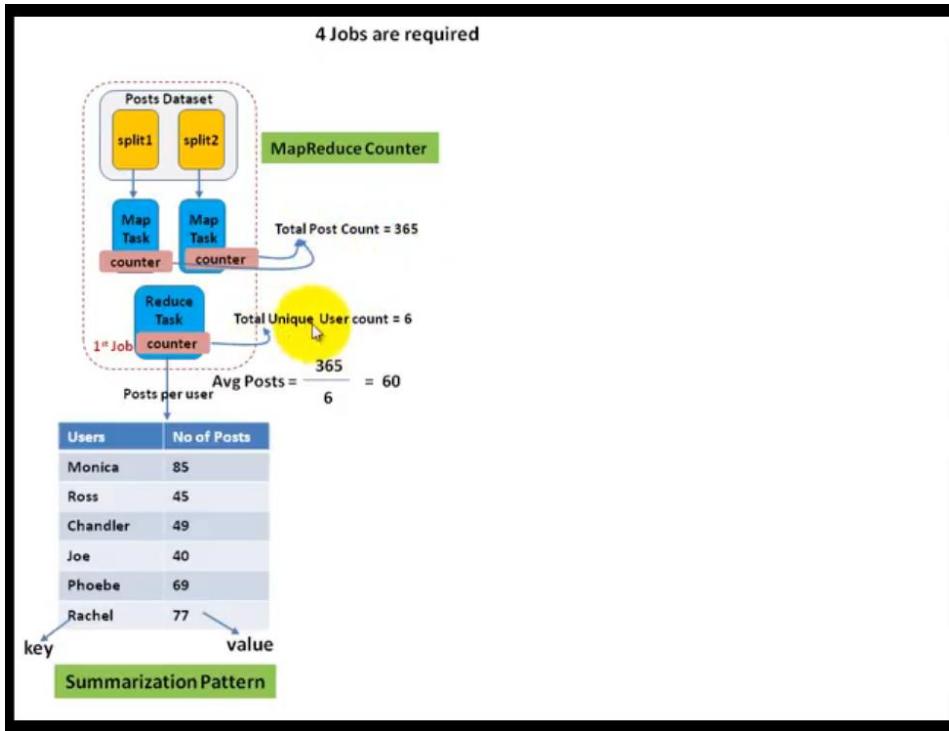
Avg posts per user = $(85 + 77 + 69 + 49 + 45 + 40) / 6 = 60$

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

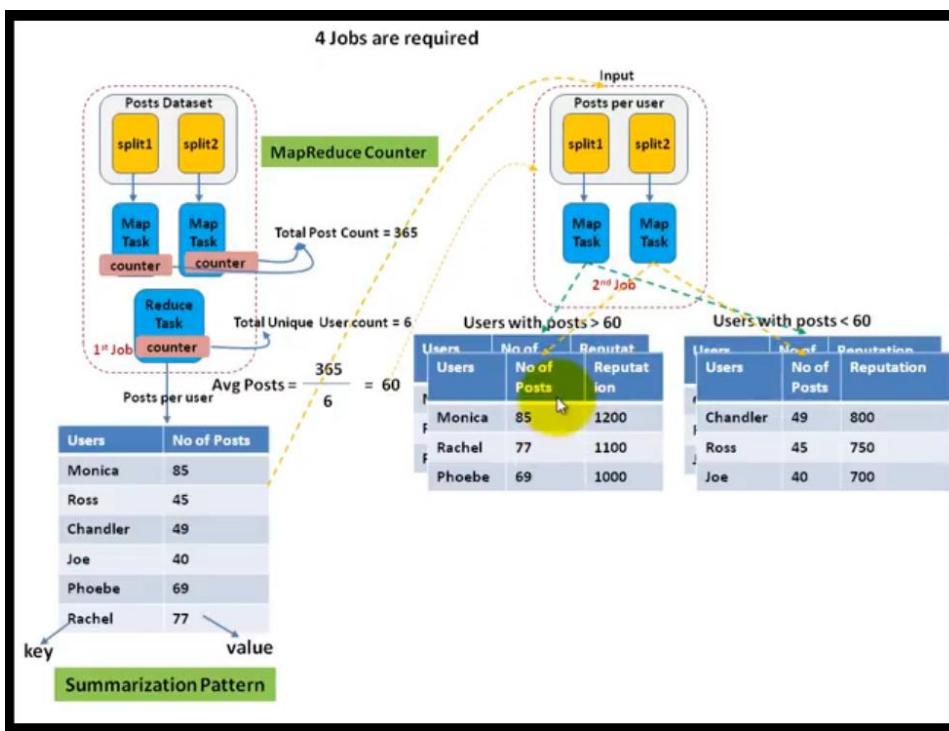
Total Users = 6 Total Posts = 365

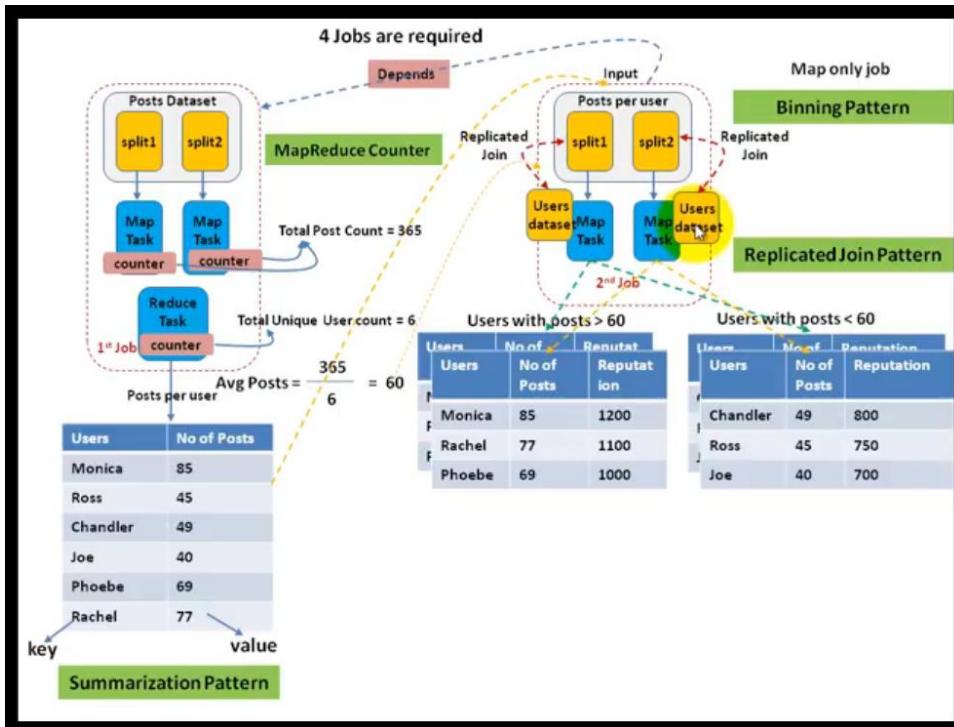
Average Reputation
= $(1200 + 1100 + 1000 / 3)$
= 1100

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

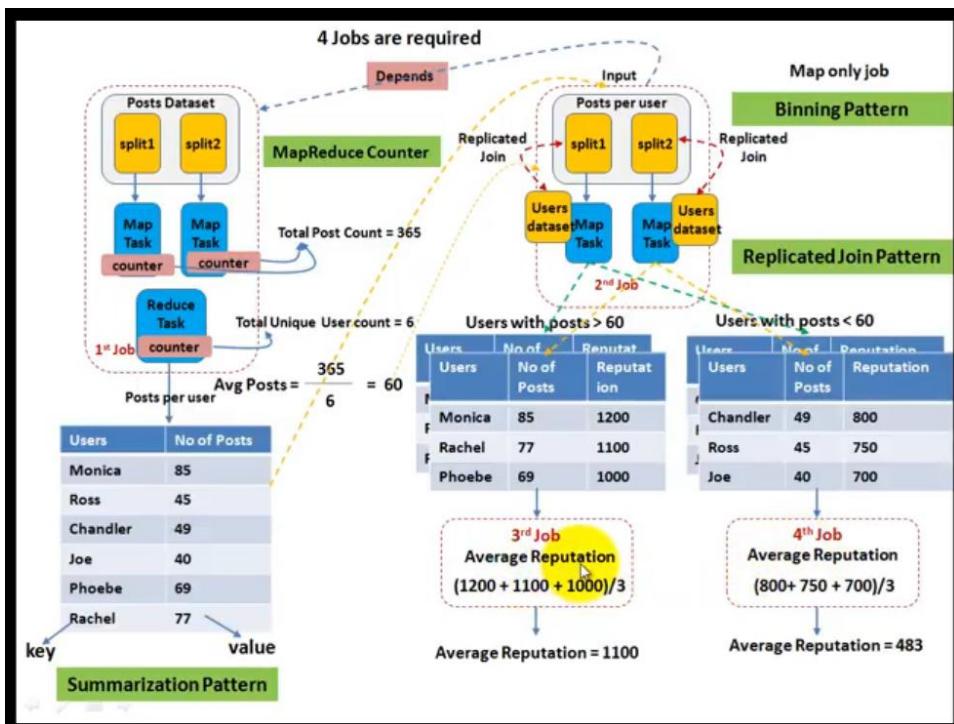


Counter in mapper will give total post count and counter in reducer side give unique number of user count





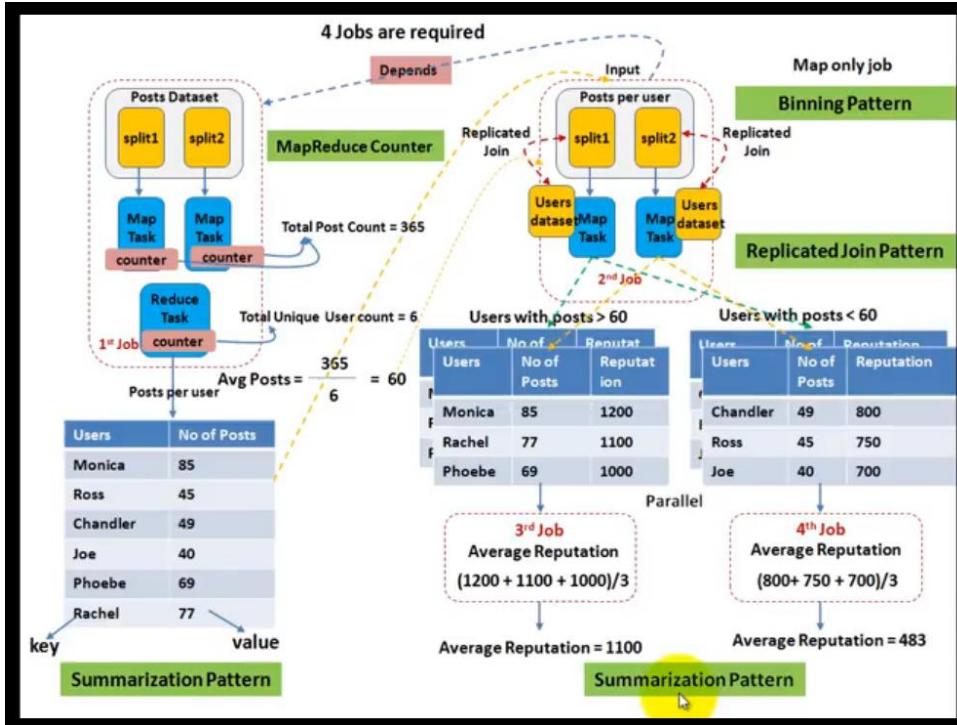
In second job we are using binning pattern and Replicated join pattern



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



```

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 IOException {
    double numRecords = (double)countingJob.getCounter(AveragePostsPerUserMapper.AVERAGE_CALC_GROUP, AveragePostsPerUserMapper.AVERAGE_CALC_COUNTER).get();
    double numUsers = (double)countingJob.getCounter(AveragePostsPerUserReducer.AVERAGE_CALC_GROUP, AveragePostsPerUserReducer.AVERAGE_CALC_COUNTER).get();

    double averagePostsPerUser = numRecords / numUsers;
}

```

```
JobControlDriver.java AveragePostsPerUserMapper.java AveragePostsPerUserReducer.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("OwnerId");
        if (userId != null) {
            outKey.set(userId);
            context.write(outKey, ONE);
            context.getCounter(AVERAGE_CALC_GROUP, POSTS_COUNTER_NAME).increment(1);
        }
    }
}
```

```
JobControlDriver.java AveragePostsPerUserMapper.java AveragePostsPerUserReducer.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);
    }
}
```

```
JobControlDriver.java AveragePostsPerUserMapper.java AveragePostsPerUserReducer.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);

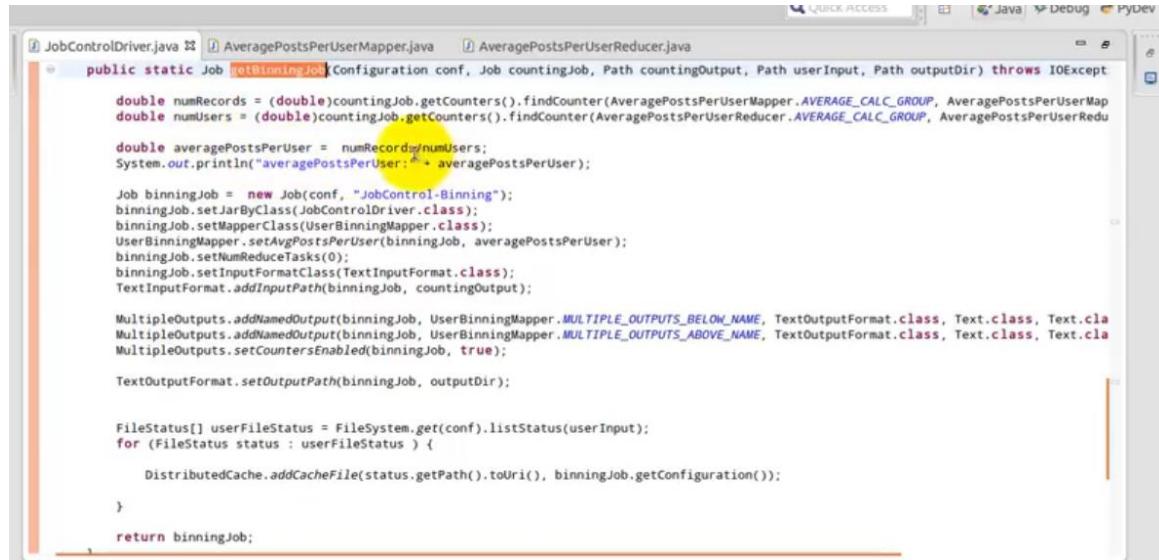
int code = 1;
code = countingJob.waitForCompletion(true) ? 0 : 1;
System.out.println("Counting Job code:" + code);
if (code == 0) {
    System.out.println("Counting Job completed");

    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



The screenshot shows the Eclipse IDE interface with three tabs open: JobControlDriver.java, AveragePostsPerUserMapper.java, and AveragePostsPerUserReducer.java. The current tab is JobControlDriver.java. The code is as follows:

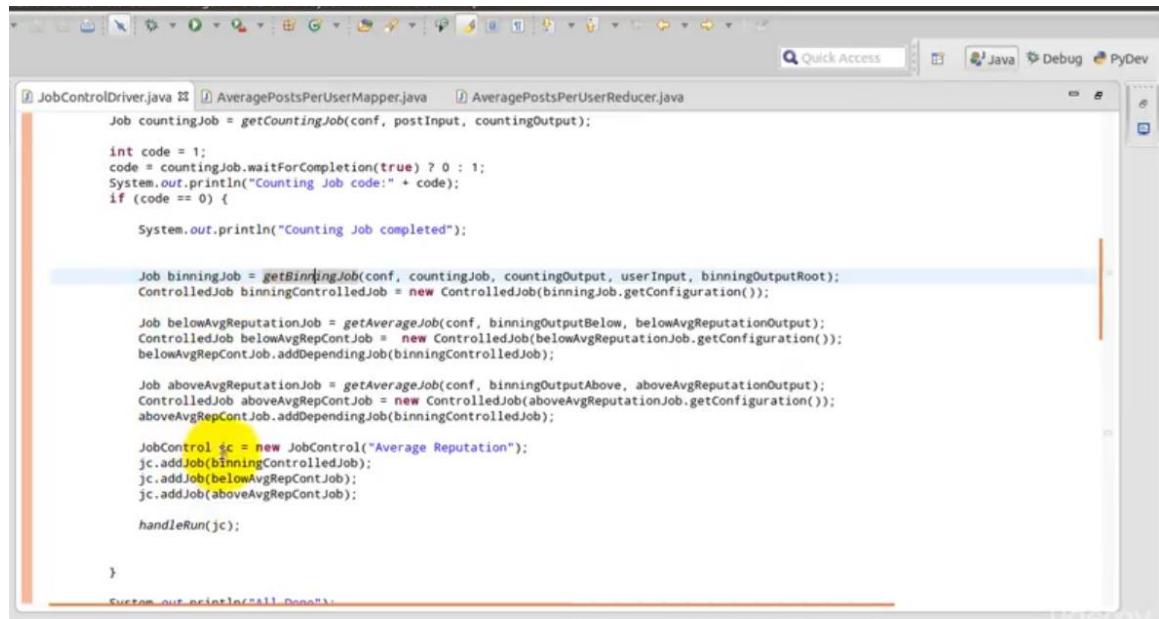
```
public static Job getBinningJob(Configuration conf, Job countingJob, Path countingOutput, Path userInput, Path outputDir) throws IOException {
    double numRecords = (double)countingJob.getCounters().findCounter(AveragePostsPerUserMapper.AVERAGE_CALC_GROUP, AveragePostsPerUserMapper.AVERAGE_CALC_GROUP).get();
    double numUsers = (double)countingJob.getCounters().findCounter(AveragePostsPerUserReducer.AVERAGE_CALC_GROUP, AveragePostsPerUserReducer.AVERAGE_CALC_GROUP).get();
    double averagePostsPerUser = numRecords/numUsers;
    System.out.println("averagePostsPerUser: " + averagePostsPerUser);

    Job binningJob = new Job(conf, "JobControl-Binning");
    binningJob.setJarByClass(JobControlDriver.class);
    binningJob.setMapperClass(UserBinningMapper.class);
    UserBinningMapper.setAvgPostsPerUser(binningJob, averagePostsPerUser);
    binningJob.setNumReduceTasks(0);
    binningJob.setInputFormatClass(TextInputFormat.class);
    TextInputFormat.addInputPath(binningJob, countingOutput);

    MultipleOutputs.addNamedOutput(binningJob, UserBinningMapper.MULTIPLE_OUTPUTS_BELOW_NAME, TextOutputFormat.class, Text.class, Text.class);
    MultipleOutputs.addNamedOutput(binningJob, UserBinningMapper.MULTIPLE_OUTPUTS_ABOVE_NAME, TextOutputFormat.class, Text.class, Text.class);
    MultipleOutputs.setCountersEnabled(binningJob, true);

    TextOutputFormat.setOutputPath(binningJob, outputDir);

    FileStatus[] userFileStatus = FileSystem.get(conf).listStatus(userInput);
    for (FileStatus status : userFileStatus) {
        DistributedCache.addCacheFile(status.getPath().toUri(), binningJob.getConfiguration());
    }
    return binningJob;
}
```



The screenshot shows the Eclipse IDE interface with three tabs open: JobControlDriver.java, AveragePostsPerUserMapper.java, and AveragePostsPerUserReducer.java. The current tab is JobControlDriver.java. The code is as follows:

```
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);
    ControlledJob aboveAvgRepContJob = new ControlledJob(aboveAvgReputationJob.getConfiguration());
    aboveAvgRepContJob.addDependingJob(binningControlledJob);

    JobControl jc = new JobControl("Average Reputation");
    jc.addJob(binningControlledJob);
    jc.addJob(belowAvgRepContJob);
    jc.addJob(aboveAvgRepContJob);

    handleRun(jc);
}

System.out.println("all Done");
}
```

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

Browsing HDFS nameNode Hadoop Map/R... +

localhost:50030/jobtracker.jsp

Compiled: 2014-07-08T20:57Z by Jenkins from (no branch)
Identifier: 201502182224

Cluster Summary (Heap Size is 48.38 MB/48.38 MB)

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes	Excluded Nodes
3	0	1	3	3	0	0	0	3	3	2.00	0	0

Scheduling Information

Queue Name	State	Scheduling Information
default	running	N/A

Filter (Jobid, Priority, User, Name)
Example: 'user:smith 3200' will filter by 'smith' only in the user field and '3200' in all fields

Running Jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information	Diagnostic Info
job_201502182224_0001	NORMAL	hduser	JobChaining-Counting	50.00%	6	3	0.00%	1	0	NA	NA

Retired Jobs
 none

Local Logs
[Log directory](#), [Job Tracker History](#)

First job is running

Browsing HDFS nameNode Hadoop Map/R... +

localhost:50030/jobtracker.jsp

Compiled: 2014-07-08T20:57Z by Jenkins from (no branch)
Identifier: 201502182224

Cluster Summary (Heap Size is 48.38 MB/48.38 MB)

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes	Excluded Nodes
0	0	2	3	0	0	0	0	3	3	2.00	0	0

Scheduling Information

Queue Name	State	Scheduling Information
default	running	N/A

Filter (Jobid, Priority, User, Name)
Example: 'user:smith 3200' will filter by 'smith' only in the user field and '3200' in all fields

Running Jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information	Diagnostic Info
job_201502182224_0002	NORMAL	hduser	JobControl-Binning	0.00%	1	0	0.00%	0	0	NA	NA

Completed Jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information	Diagnostic Info
job_201502182224_0001	NORMAL	hduser	JobChaining-Counting	100.00%	6	6	100.00%	1	1	NA	NA

Retired Jobs

After completion of first job second job is running

Now second job is completed successfully

Browsing HDFS × nameNode Hadoop Map/R... +

localhost:50030/jobtracker.jsp

Compiled: 2014-07-08T20:57Z by jenkins from (no branch)
Identifier: 20150218224

Cluster Summary (Heap Size is 48.38 MB/48.38 MB)

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes	Excluded Nodes
0	0	2	3	0	0	0	0	3	3	2.00	0	0

Scheduling Information

Queue Name	State	Scheduling Information
default	running	N/A

Filter (Jobid, Priority, User, Name) Example: 'user:smith 3200' will filter by 'smith' only in the user field and '3200' in all fields

Running Jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information	Diagnostic Info
job_201502182224_0001	NORMAL	hduser	JobChaining-Counting	100.00%	6	6	100.00%	1	1	NA	NA
job_201502182224_0002	NORMAL	hduser	JobControl-Binning	100.00%	1	1	100.00%	0	0	NA	NA

Completed jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information	Diagnostic Info
job_201502182224_0001	NORMAL	hduser	JobChaining-Counting	100.00%	6	6	100.00%	1	1	NA	NA
job_201502182224_0002	NORMAL	hduser	JobControl-Binning	100.00%	1	1	100.00%	0	0	NA	NA

Retired Jobs

localhost:50030/jobdetails.jsp?jobid=job_201502182224_0002&refresh=0

After completion of second job third and fourth job running simultaneously

Browsing HDFS × nameNode Hadoop Map/R... +

localhost:50030/jobtracker.jsp

Compiled: 2014-07-08T20:57Z by jenkins from (no branch)
Identifier: 20150218224

Cluster Summary (Heap Size is 48.38 MB/48.38 MB)

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes	Excluded Nodes
0	0	4	3	0	0	0	0	3	3	2.00	0	0

Scheduling Information

Queue Name	State	Scheduling Information
default	running	N/A

Filter (Jobid, Priority, User, Name) Example: 'user:smith 3200' will filter by 'smith' only in the user field and '3200' in all fields

Running Jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information	Diagnostic Info
job_201502182224_0003	NORMAL	hduser	ParallelJobs	0.00%	1	0	0.00%	1	0	NA	NA
job_201502182224_0004	NORMAL	hduser	ParallelJobs	0.00%	1	0	0.00%	1	0	NA	NA

Completed jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information	Diagnostic Info
job_201502182224_0001	NORMAL	hduser	JobChaining-Counting	100.00%	6	6	100.00%	1	1	NA	NA
job_201502182224_0002	NORMAL	hduser	JobControl-Binning	100.00%	1	1	100.00%	0	0	NA	NA

Check output

First job output

Browsing HDFS - Mozilla Firefox

Browsing HDFS nameNode Hadoop Map/R... namenode.cdh-cluster.com:50070/explorer.html#/user/hduser/hive/stackoverflow

Browse Directory

/user/hduser/hive/stackoverflow Go!

Permission	Owner	Group	Size	Replication	Block Size	Name
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	comments
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	jobcontrol_output
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	jobcontrol_output_bins
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	jobcontrol_output_count
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	posts
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	users

Hadoop, 2013.

Browsing HDFS - Mozilla Firefox

Browsing HDFS nameNode Hadoop Map/R... namenode.cdh-cluster.com:50070/explorer.html#/user/hduser/hive/stackoverflow/jobcontrol_output_count

Browse Directory

/user/hduser/hive/stackoverflow/jobcontrol_output_count 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
-rw-r--r--	hduser	supergroup	1.25 MB	3	128 MB	part-r-00000

Hadoop, 2013.

It contains total number of posts per user

Second job output folder

Browsing HDFS nameNode Hadoop Map/R... +

namenode.cdh-cluster.com:50070/explorer.html#/user/hduser/hive/stackoverflow

Browse Directory

/user/hduser/hive/stackoverflow Go!

Permission	Owner	Group	Size	Replication	Block Size	Name
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	comments
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	jobcontrol_output
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	jobcontrol_output_bins
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	jobcontrol_output_count
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	posts
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	users

Hadoop, 2013.

Browsing HDFS nameNode Hadoop Map/R... namenode.cdh-cluster.com:50070/explorer.html#/user/hduser/hive/stackoverflow/jobcontrol_output_bins Go!

Browse Directory

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	aboveavg
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	belowavg
-rW-r--r--	hduser	supergroup	0 B	3	128 MB	part-m-00000

Hadoop, 2013.

Two bins are created from the second job

Final output

Browsing HDFS nameNode Hadoop Map/R... namenode.cdh-cluster.com:50070/explorer.html#/user/hduser/hive/stackoverflow Go!

Browse Directory

Permission	Owner	Group	Size	Replication	Block Size	Name
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	comments
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	jobcontrol_output
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	jobcontrol_output_bins
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	jobcontrol_output_count
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	posts
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	users

Hadoop, 2013.

It contains average reputation of all the users in each bin

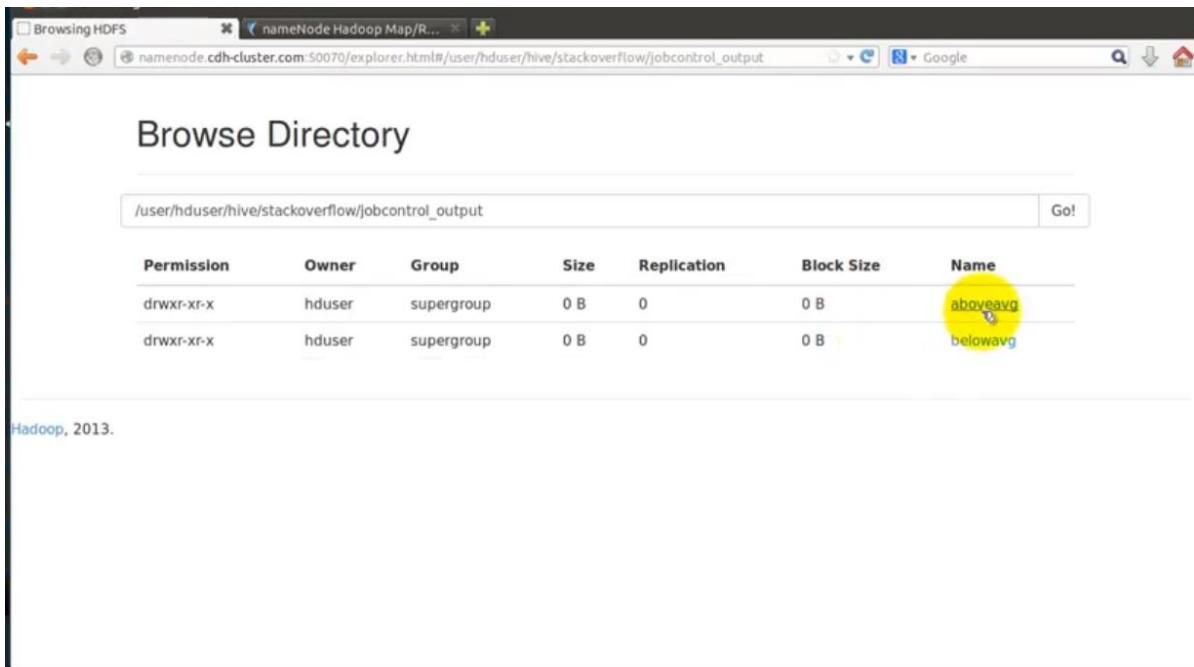
Browsing HDFS nameNode Hadoop Map/R... namenode.cdh-cluster.com:50070/explorer.html#/user/hduser/hive/stackoverflow/jobcontrol_output Google

Browse Directory

/user/hduser/hive/stackoverflow/jobcontrol_output Go!

Permission	Owner	Group	Size	Replication	Block Size	Name
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	aboveavg
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	belowavg

Hadoop, 2013.



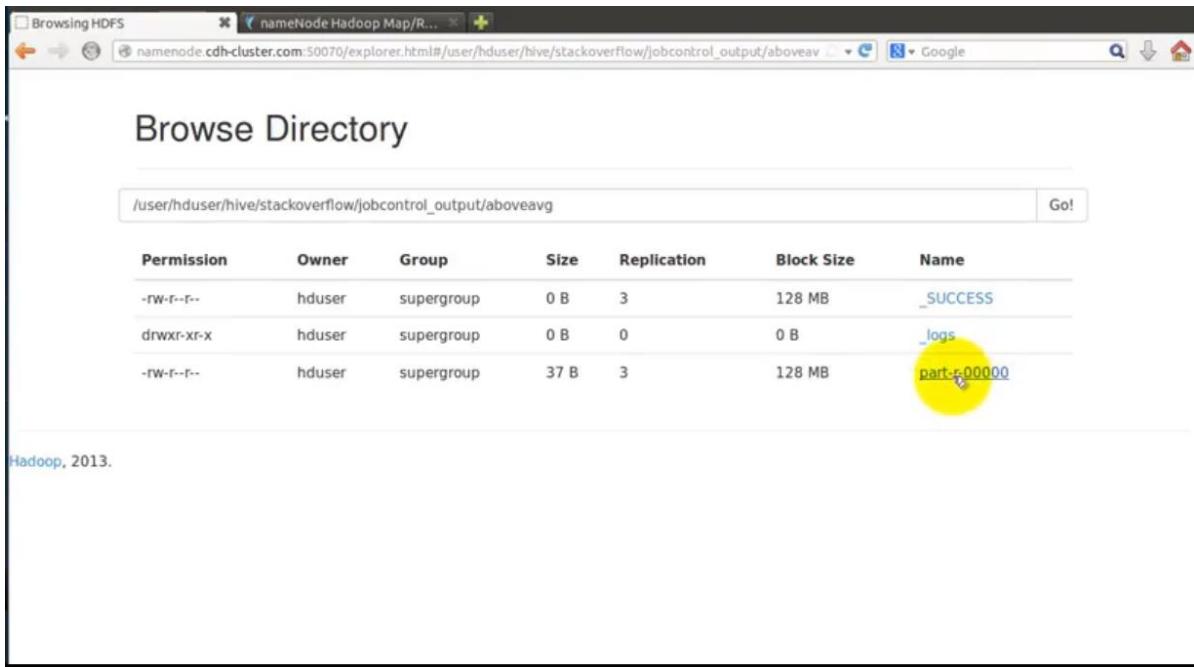
Browsing HDFS nameNode Hadoop Map/R... namenode.cdh-cluster.com:50070/explorer.html#/user/hduser/hive/stackoverflow/jobcontrol_output/aboveavg Google

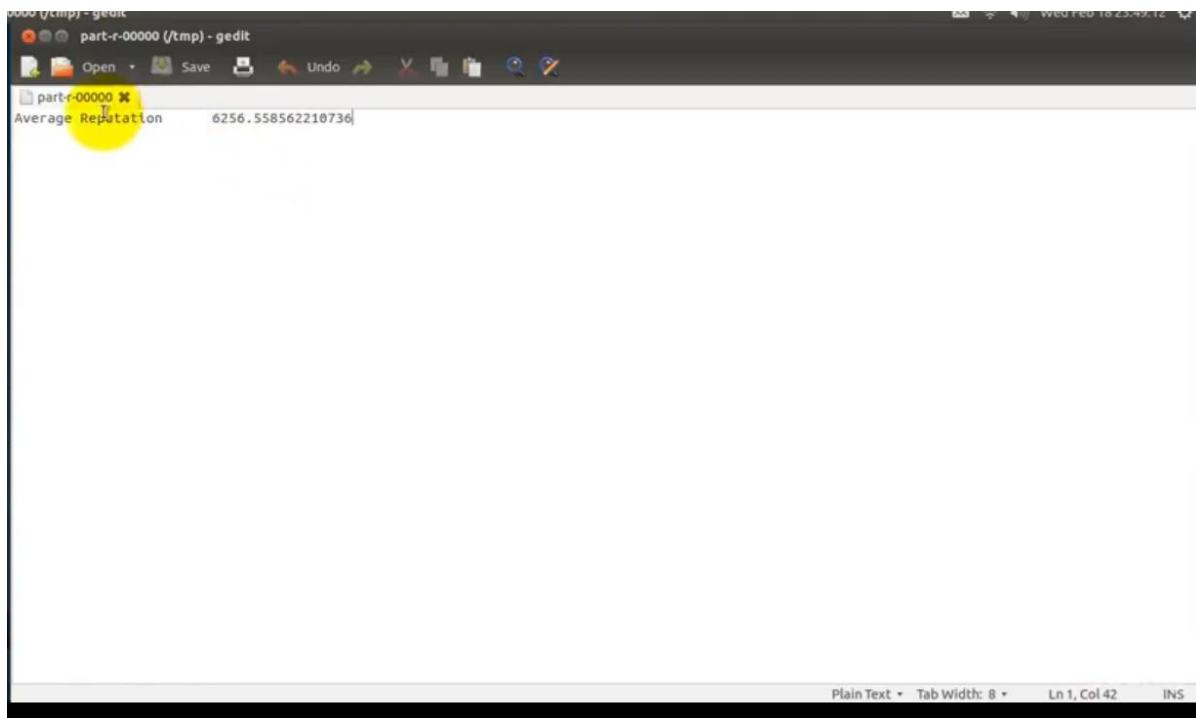
Browse Directory

/user/hduser/hive/stackoverflow/jobcontrol_output/aboveavg 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
-rw-r--r--	hduser	supergroup	37 B	3	128 MB	part-r-00000

Hadoop, 2013.





0000 /tmp - gedit
part-r-00000 /tmp - gedit
Open Save Undo Redo Cut Copy Paste Find Replace
part-r-00000 x Average Reputation 6256.558562210736
Plain Text Tab Width: 8 Ln 1, Col 42 INS

Average reputation of all the users in above average bin