



# Chap11 & Lab6: Big Data Programming

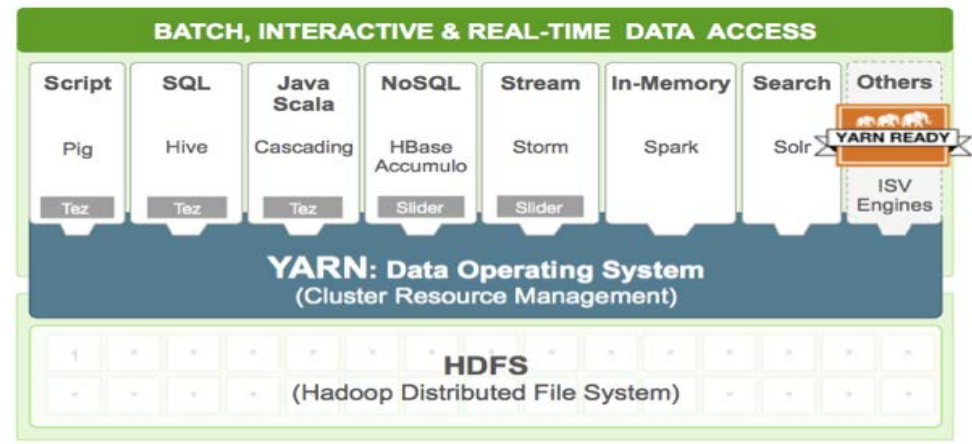
National Tsing Hua University  
2020, Fall Semester

# Outline

- Basic Hadoop Programming
  - Hadoop Classes
  - Mapper/Reducer
  - WordCount Example
- Hands-on Lab
  - HDFS/ MapReduce
  - Hive / Spark
- Advanced Hadoop Programming
  - Custom key & value types
  - Combiner/Partitioner
  - GroupingComparator/SortComparator
  - Secondarysort example

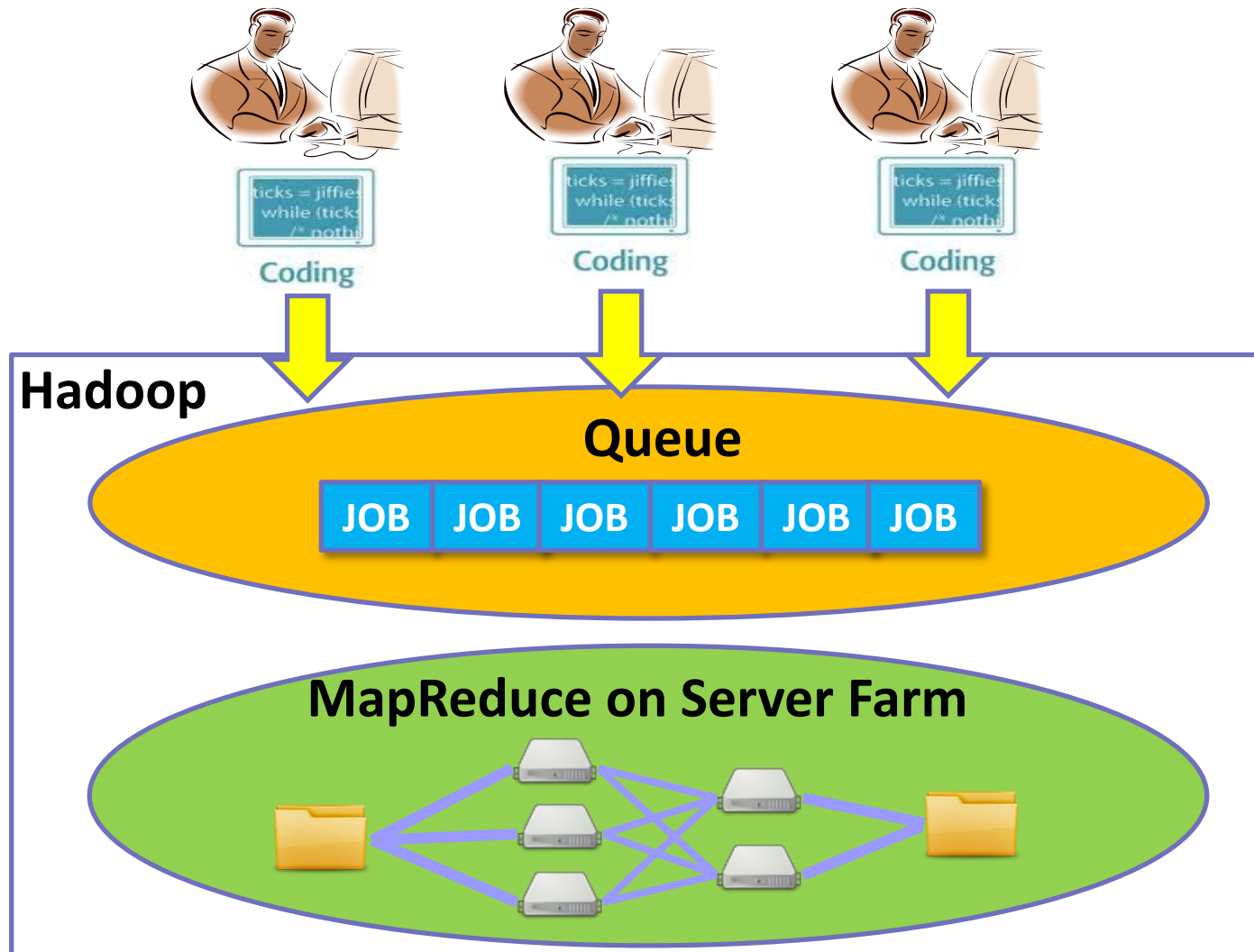
# Hadoop Implementation

- No major changes after Hadoop release 2.x
  - **New version with YARN** (resource manager)
  - Latest version is 3.3.0 (Nov. 2020)



- **Java Language**
  - Based on **inheritance and interface**
- **Official Tutorial**
  - <https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html>

# Hadoop Runtime



# Import hadoop package

Import classes in “**org.apache.hadoop.mapreduce**” package

- `import java.io.IOException; import java.util.StringTokenizer;`
- `import org.apache.hadoop.conf.Configuration;`
- `import org.apache.hadoop.fs.Path;`
- `import org.apache.hadoop.io.IntWritable;`
- `import org.apache.hadoop.io.Text;`
- `import org.apache.hadoop.mapreduce.Job;`
- `import org.apache.hadoop.mapreduce.Mapper;`
- `import org.apache.hadoop.mapreduce.Reducer;`
- `import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;`
- `import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;`
- [Other necessary classes called by your code .....]

Notice:

- “**mapreduce**” package is not interchangeable with “**mapred**” package
- Prevent using “**Deprecated**” methods

# Main Hadoop Classes

## ■ Configuration

- Hadoop **cluster** configuration

## ■ Job

- the primary interface for a user to describe a map-reduce **job** to the Hadoop framework for execution

## ■ Mapper

- maps input  $\langle K, V \rangle$  pairs to intermediate  $\langle K, V \rangle$  pairs

## ■ Reducer

- reduces intermediate values to a smaller set of values

## ■ Partitioner

- partitions the key of intermediate  $\langle K, V \rangle$  pairs to reducer

## ■ Combiner

- combine map-outputs  $\langle K, V \rangle$  pairs before being sent to reducers

## ■ RecordReader/RecordWriter

- Read input file & write output file

# Job Class

## ■ configure a job

- Specify the class for mapper, reducer, combiner, etc.

## ■ submit the job

- **Submit the job to the cluster and return immediately**
- **Or submit the job to the cluster and wait for it to finish**

## ■ control its execution

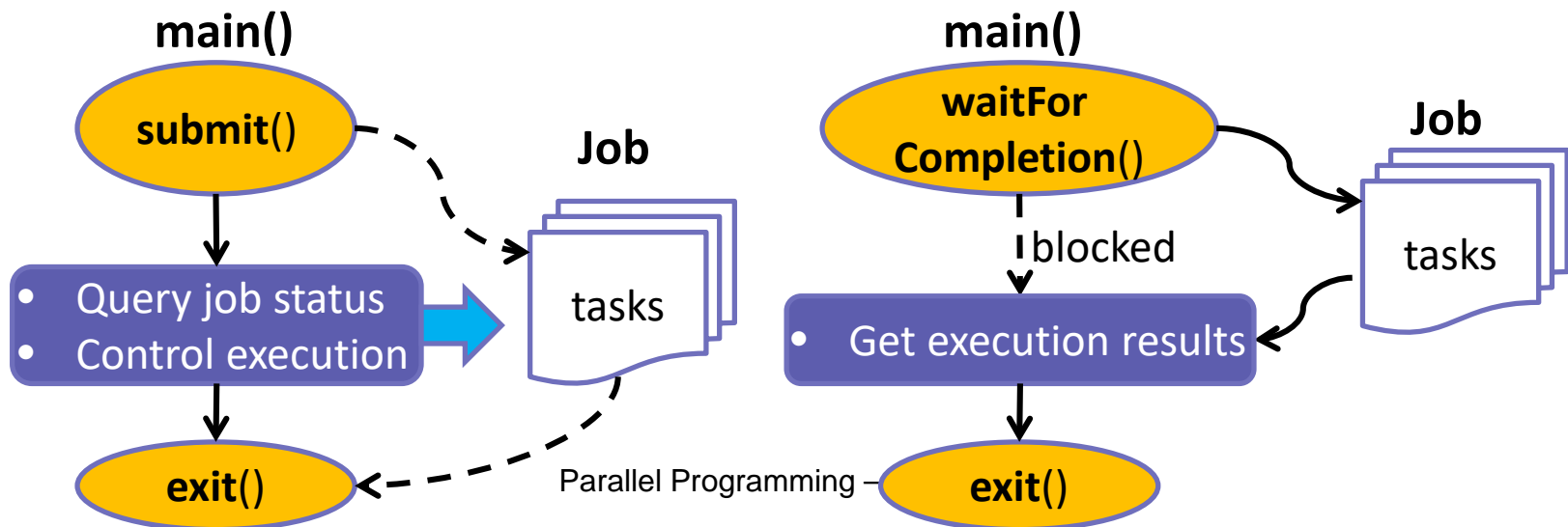
- Set the number of max attempts to run a reduce or map task.
- Set scheduling priority.
- Kill the running job, or specific task.
- Turn speculative execution on or off for this job.

## ■ query its state.

- Get the *progress* of the job's map-tasks or reduce-tasks (between 0 and 1).
- Returns the current state of the Job.
- Get start time of the job.
- Check if the job completed successfully.

# Job Creation & Submission

Method	Description
<b>getInstance</b> ( <b>Configuration</b> conf, <b>String</b> jobName)	Creates a new job with a given jobName.
<b>setJarByClass</b> ( <b>Class</b> <?> cls)	Set the Jar by finding where a given class came from.
<b>submit</b> ()	Submit the job to the cluster and return immediately. (non-blocking call)
<b>waitForCompletion</b> (boolean verbose)	Submit the job to the cluster and wait for it to finish. (blocking call)





# Query & Control Job Execution

Method	Description
<b>getStartTime()</b>	Get start time of the job.
<b>getFinishTime()</b>	Get finish time of the job.
<b>getStatus()</b>	Returns a JobStatus object contain all the current job state info
<b>mapProgress()</b> <b>reduceProgress()</b>	Get the <i>progress</i> of the job. , as a float between 0.0 and 1.0.
<b>getCounters()</b>	Gets the counters object for this job
<b>isComplete()</b>	Check if the job is finished or not.

Method	Description
<b>setPriority</b> (JobPriority prio)	High/Low/Normal/Very_High/Very_Low
<b>setNumReduceTasks</b> (int n)	Set the requisite number of reduce tasks for this job. (notice: no method for map tasks)
<b>setSpeculativeExecution</b> (boolean flag)	Turn speculative execution on or off for this job.
<b>killJob()</b>	Kill the running job.
<b>killTask</b> (TaskAttemptID taskId)	Kill indicated task attempt.

# Example

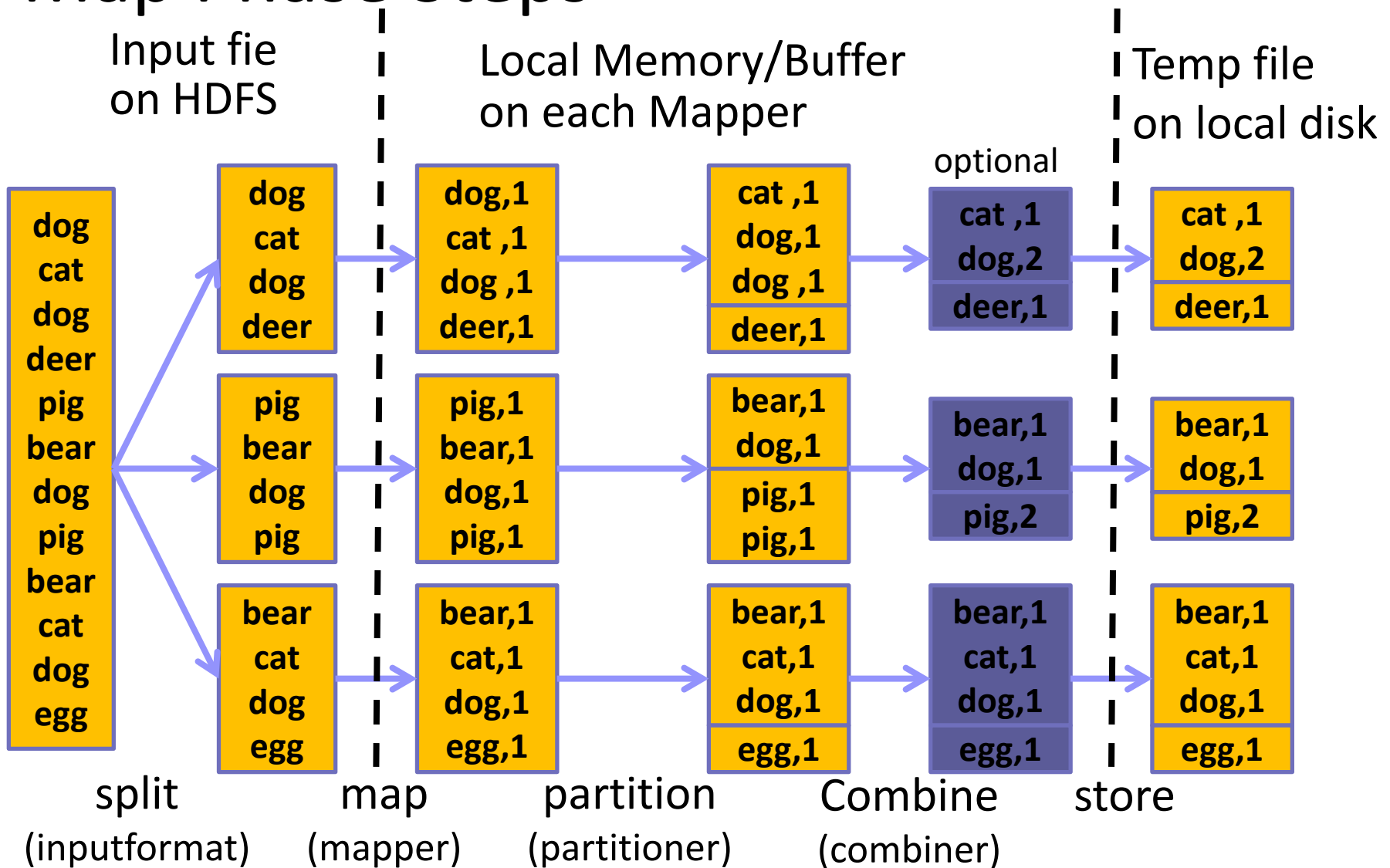
```
public class WordCount {  
    public static void main(String[] args) throws Exception {  
        Configuration conf = new Configuration();  
        Job job = Job.getInstance(conf, "wordcount");  
        job.setJarByClass(WordCount.class);  
        job.waitForCompletion(true); // Submit the job and wait for it to finish  
    }  
}
```

```
public class WordCount {  
    public static void main(String[] args) throws Exception {  
        Configuration conf = new Configuration();  
        Job job = Job.getInstance(conf, "wordcount");  
        job.setJarByClass(WordCount.class);  
        job.submit(); // Submit the job and return immediately  
        while(job.isComplete()==false) {  
            System.out.println(mapProgress());  
        }  
    }  
}
```

# Job Configuration on Compute Functions

Method	Description
<b>setMapperClass</b> (Class<? extends <b>Mapper</b> >)	Set the Mapper class for the job
<b>setReducerClass</b> (Class<? extends <b>Reducer</b> >)	Set the Reducer class for the job.
<b>setPartitionerClass</b> (Class<? extends <b>Partitioner</b> >)	partition Mapper-outputs to be sent to the reducers
<b>setCombinerClass</b> (Class<? extends <b>Reducer</b> >)	combine map-outputs before being sent to the reducer <b>It is an optional function during execution</b>
<b>setGroupingComparatorClass</b> (Class<? extends <b>RawComparator</b> >)	Define the comparator that controls which keys are grouped together for a single call to reducer
<b>setSortComparatorClass</b> (Class<? extends <b>RawComparator</b> >)	Define the comparator that controls how the keys are sorted before they are passed to the reducer

# Map Phase Steps

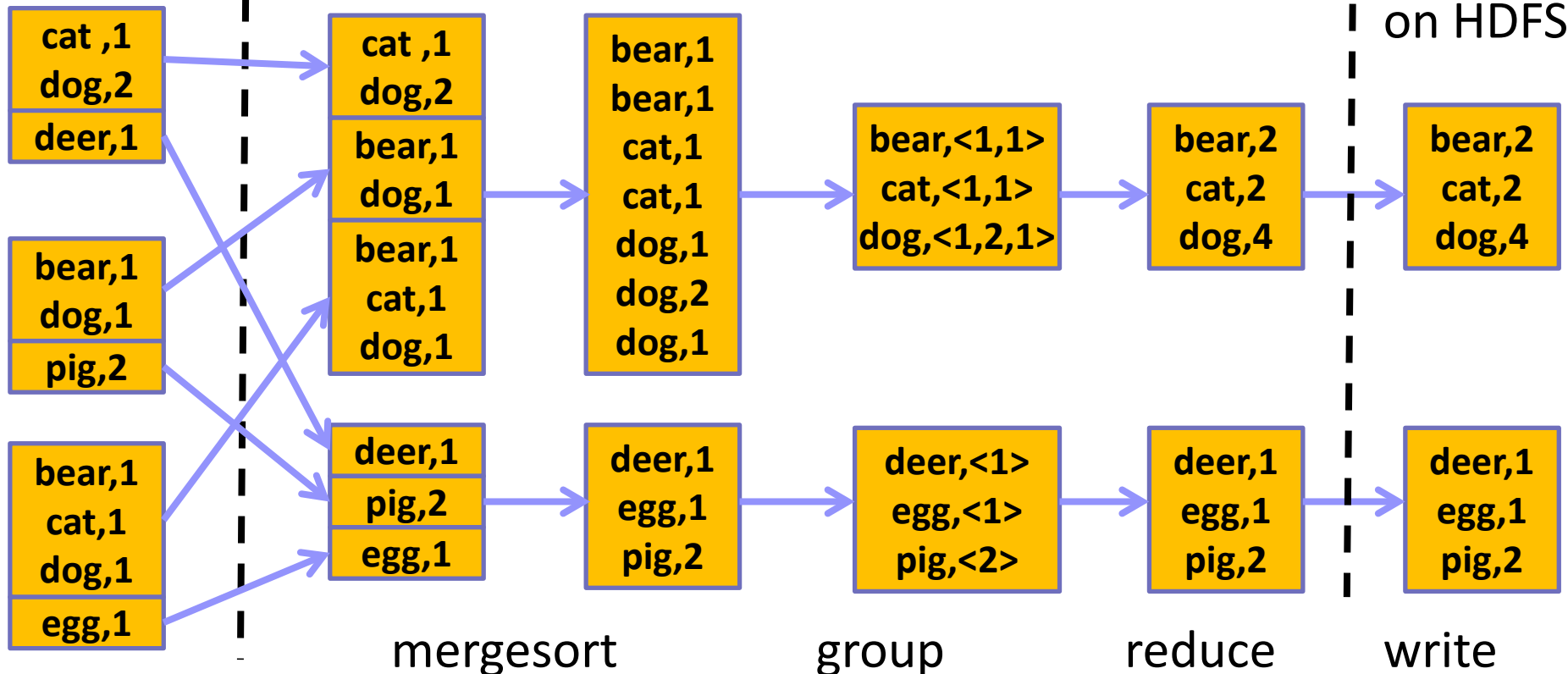


# Reduce Phase Steps

Temp file  
Mapper's  
local disk

Local Memory/Buffer  
on each Reducer

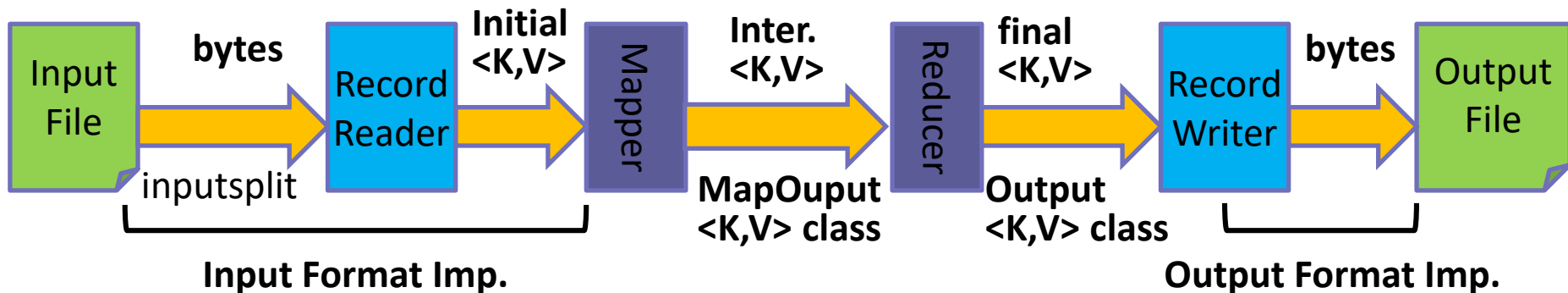
Output  
files  
on HDFS



fetch&shuffle (sortComparator) (groupingComparator) (reducer) (outputformat)

# Job Configuration on Data Type

Method	Description
<code>setInputFormatClass()</code>	Set the InputFormat <b>implementation</b> for the job
<code>setMapOutputKeyClass()</code>	Set the key class for the map output data <b>Same type as final output if not specify</b>
<code>setMapOutputValueClass()</code>	Set the value class for the map output data <b>Same type as final output if not specify</b>
<code>setOutputKeyClass()</code>	Set the key class for the job output data
<code>setOutputValueClass()</code>	Set the value class for job outputs
<code>setOutputFormatClass()</code>	Set the OutputFormat <b>implementation</b> for the job



# Input/Output Format Class

- The MapReduce operates **exclusively** on  $\langle K, V \rangle$  pairs
  - It views the job input as a set of  $\langle \text{key}, \text{value} \rangle$  pairs and produces a set of  $\langle \text{key}, \text{value} \rangle$  pairs as the output of the job
- InputFormat: parse input file into a set of  $\langle \text{key}, \text{value} \rangle$ 
  - **TextInputFormat**: Keys are the **position** in the file, and values are the **line of text**.
  - **KeyValueTextInputFormat**: Each line is divided into key and value parts by a **separator byte**. If no such a byte exists, the key will be the entire line and value will be empty.
- OutputFormat: write a set of  $\langle \text{key}, \text{value} \rangle$  to output file
  - **TextOutputFormat**: writes plain text: key, value, and `"\r\n"`.

# Key-Value Pair Class

- Both **key** and **value** must implement *Writable* interface
  - A serializable object which implements a simple, efficient, serialization protocol, based on **DataInput** and **DataOutput**
- **Key** also implements the interface of *WritableComparable*
  - Because key must be **sorted** by the framework
- Default supported types includes:
  - **BooleanWritable, BytesWritable, DoubleWritable, FloatWritable, IntWritable, LongWritable, Text, NullWritable**



# WordCount: Main()

```
public static void main(String[] args) throws Exception {  
    Configuration conf = new Configuration();  
    Job job = Job.getInstance(conf, "world count");  
    job.setJarByClass(WordCount.class);  
  
    job.setMapperClass(Tokenizer.class); // Tokenizer is the mapper function  
    job.setCombiner(IntSum.class);  
    job.setReducerClass(IntSum.class); // IntSum is the reducer function  
  
    //FileInputFormat is the base class for all file-based InputFormats  
    FileInputFormat.addInputPaths(job, new Path(args[0]));  
    FileOutputFormat.addOutputPath(job, new Path(args[1]));  
  
    job.setInputFormat(TextInputFormat.class); // inputs are texts  
    job.setOutputFormat(TextOutputFormat.class); // outputs are texts  
  
    job.setOutputKeyClass(Text.class); // intermediate and final key is text  
    job.setOutputValueClass(IntWritable.class); // intermediate and final value is int  
  
    job.waitForCompletion(true); // Submit the job and wait for it to finish  
}
```

# Mapper

- Mapper maps input key/value pairs to a set of intermediate key/value pairs
  - The transformed intermediate records do **NOT** need to be the same type as the input records.
  - A given input pair may map to **zero** or **many** output pairs.
- Each key/value pair is applied with a map function:
  - **map(WritableComparable, Writable, Context)**
  - **<WritableComparable, Writable>** are the input key-value pairs generated by the **InputFormat class**
  - **context.write(K, V)** collects output key-value pairs

# Mapper

Input <K,V> type  
from InputFormat

Default <K,V> type  
for final output

```
public static class Tokenizer extends Mapper<Object, Text, Text, IntWritable> {  
    private final static IntWritable one = new IntWritable(1);  
    private Text word = new Text();  
    public void map(Object key, Text value, Context context)  
        throws IOException {  
        String line = value.toString();  
        StringTokenizer iter = new StringTokenizer(line);  
        while (iter.hasMoreTokens()) { // each line has multiple words  
            word.set(iter.nextToken());  
            context.write(word, one);  
        }  
    }  
}
```

<K,V> must be private  
var to the class

Set the var value  
Don't declare a new var here

```
main():  
    job.setOutputKeyClass(Text.class);  
    job.setOutputValueClass(IntWritable.class);  
    job.setMapperClass(Tokenizer.class);  
    job.setInputFormat(TextInputFormat.class);
```

# Reducer

- Reducer reduces a set of intermediate values which share a key to a smaller set of values.
  - The transformed intermediate records do **NOT** need to be the same type as the input records.
  - A given input pair may map to **zero** or **many** output pairs.
- Each **group** of (K,V) pair applied with a reduce func:
  - `reduce(WritableComparable, Iterator<Writable>, Context)`
  - **WritableComparable** is the input key-value pairs generated by the **mapper class**
  - **Iterator**<Writable> is the **list of values grouped by the same key**
  - **context.write(K, V)** collects output key-value pairs

# Reducer

Output <K,V> type

```
public static class IntSum extends
    Reducer<Text, IntWritable, Text, IntWritable>
{
    private IntWritable result = new IntWritable();
    public void reduce(Text key, Iterator<IntWritable> values,
        Context context) throws IOException, InterruptedException
    {
        int sum = 0;
        while (values.hasNext()) sum += values.next().get();
        result.set(sum);
        context.write(key, result);
    }
}
```

Set the var value  
Don't declare a new var here

```
main():
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    job.setReducerClass(IntSum.class);
    job.setOutputFormat(TextInputFormat.class);
```

# How many Map/Reduce Tasks?

- The number of map tasks is controlled by the implementation of **inputsplit** in **inputFormat**
  - Default is to split by the **block size of files in HDFS**
  - But it can also be overwritten to split differently
- The number of reduce tasks is controlled by the job configuration: **job.setNumReduceTasks(int n)**
  - Default is 1, but the right number of reduces seems to be 0.95 or 1.75 multiplied by #reduce\_slots
  - More reducer ➔ **higher framework overhead, better load balancing and lowers failure cost.**

# Outline

- Basic Hadoop Programming
  - Hadoop Classes
  - Mapper/Reducer
  - WordCount Example
- Hands-on Lab
  - HDFS/ MapReduce
  - Hive / Spark
- Advanced Hadoop Programming
  - Custom key & value types
  - Combiner/Partitioner
  - GroupingComparator/SortComparator
  - Secondarysort example

# Hands-on Lab

- SSH login to our Hadoop cluster
  - Access info will be posted on ilms

```
 _ _ | ( _ _ | _ )
 _ | ( _ _ | _ ) Amazon Linux AMI
 _ | \ _ _ | _ |

https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
19 package(s) needed for security, out of 29 available
Run "sudo yum update" to apply all updates.

EEEEEEEEEEEEEEEEEEEE MMMMMMMM MMMMMMMM RRRRRRRRRRRRRRRR
E ::::::::::::::::::::E M ::::::::::M M ::::::::::M R :::::::::::R
EE ::::::::::::::::::::E M ::::::::::M M ::::::::::M R ::::RRRRRR:::R
E ::::E EEEEE M ::::::::::M M ::::::::::M RR ::::R R ::::R
E ::::E M ::::::::::M M ::::::::::M R :::R R :::R
E ::::EEEEEEEEEE M ::::M M ::::M M ::::M R :::RRRRRR:::R
E ::::::::::::::::::::E M ::::M M ::::M M ::::M R ::::::::::RR
E ::::EEEEEEEEEE M ::::M M ::::M M ::::M R :::RRRRRR:::R
E ::::E M ::::M M ::::M M ::::M R :::R R :::R
E ::::E EEEEE M ::::M MMM M ::::M R :::R R :::R
EE ::::EEEEEEEEEE:::E M ::::M M ::::M R :::R R :::R
E ::::::::::::::::::::E M ::::M M ::::M RR :::R R :::R
EEEEEEEEEEEEEEEEEEEE MMMMMMMM MMMMMMMM RRRRRRR RRRRRR

[hadoop@ip-172-31-90-14 ~]$
```



# Basic HDFS Commands

Command	Description
-ls <args>	List directory
-mkdir <paths>	Create a directory
-put <localsrc> <HDFS_dest_Path>	Upload files
-get <hdfs_src> <localdst>	Download file
-cat <path[filename]>	See content of files
-cp <source> <dest>	Copy files in HDFS
-rm <arg>	Remove files or directories
-tail <path[filename]>	Display last few lines of a file
-getmerge [hdfs_src_dir] [hdfs_dst_file]	Merge files (from reducers)

■ **\$hadoop fs [command]**

■ **Ref:** <https://hadoop.apache.org/docs/r2.7.1/hadoop-project-dist/hadoop-common/FileSystemShell.html>

# Steps to Prepare Input Files

- Copy files to your home directory & switch to the dir

```
$ cp -R /home/hadoop/lab .  
$ cd lab
```

- Create your own folder on HDFS:

```
$ hadoop fs -mkdir /user/hadoop/[username]
```

- Copy input test files from local file system to your HDFS

```
$ hadoop fs -put input /user/hadoop/[username]/
```

- List the input files on HDFS

```
$ hadoop fs -ls /user/hadoop/[username]/input/
```

# Steps to Prepare Input Files

```
-rw-r--r-- 1 jchou hadoop 144860 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-first-51.txt
-rw-r--r-- 1 jchou hadoop 182399 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-hamlet-25.txt
-rw-r--r-- 1 jchou hadoop 117902 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-julius-26.txt
-rw-r--r-- 1 jchou hadoop 157094 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-king-45.txt
-rw-r--r-- 1 jchou hadoop 154933 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-life-54.txt
-rw-r--r-- 1 jchou hadoop 148351 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-life-55.txt
-rw-r--r-- 1 jchou hadoop 122448 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-life-56.txt
-rw-r--r-- 1 jchou hadoop 14364 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-lovers-62.txt
-rw-r--r-- 1 jchou hadoop 129916 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-loves-8.txt
-rw-r--r-- 1 jchou hadoop 105202 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-macbeth-46.txt
-rw-r--r-- 1 jchou hadoop 130363 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-measure-13.txt
-rw-r--r-- 1 jchou hadoop 122508 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-merchant-5.txt
-rw-r--r-- 1 jchou hadoop 131401 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-merry-15.txt
-rw-r--r-- 1 jchou hadoop 96439 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-midsummer-16.txt
-rw-r--r-- 1 jchou hadoop 123284 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-much-3.txt
-rw-r--r-- 1 jchou hadoop 156338 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-othello-47.txt
-rw-r--r-- 1 jchou hadoop 111421 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-pericles-21.txt
-rw-r--r-- 1 jchou hadoop 84687 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-rape-61.txt
-rw-r--r-- 1 jchou hadoop 144138 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-romeo-48.txt
-rw-r--r-- 1 jchou hadoop 157146 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-second-52.txt
-rw-r--r-- 1 jchou hadoop 95659 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-sonnets-59.txt
-rw-r--r-- 1 jchou hadoop 124128 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-taming-2.txt
-rw-r--r-- 1 jchou hadoop 99303 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-tempest-4.txt
-rw-r--r-- 1 jchou hadoop 148008 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-third-53.txt
-rw-r--r-- 1 jchou hadoop 113037 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-timon-49.txt
-rw-r--r-- 1 jchou hadoop 123897 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-titus-50.txt
-rw-r--r-- 1 jchou hadoop 134743 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-tragedy-57.txt
-rw-r--r-- 1 jchou hadoop 180293 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-tragedy-58.txt
-rw-r--r-- 1 jchou hadoop 158763 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-troilus-22.txt
-rw-r--r-- 1 jchou hadoop 116626 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-twelfth-20.txt
-rw-r--r-- 1 jchou hadoop 101862 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-two-18.txt
-rw-r--r-- 1 jchou hadoop 54386 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-venus-60.txt
-rw-r--r-- 1 jchou hadoop 145677 2019-12-06 17:24 /user/hadoop/123456/input/shakespeare-winters-19.txt
```

# Steps to Compile & Run Jobs

## ■ Compile the Hadoop java files

```
$ javac -classpath `hadoop classpath` WordCount.java -d bin
```

## ■ Create a jar file for the executable

```
$ jar -cvf WordCount.jar -C bin .
```

It is normal to see the error:  
“No such file or directory”  
for your first time to run

## ■ Remove the output folder on HDFS

```
$ hadoop fs -rm -R /user/hadoop/[username]/output
```

## ■ Run the Hadoop job

```
$ hadoop jar WordCount.jar org.myorg.WordCount  
/user/hadoop/[username]/input  
/user/hadoop/[username]/output
```

# Steps to Compile & Run Jobs

```
19/12/06 17:26:46 INFO mapreduce.Job: map 70% reduce 7%
19/12/06 17:26:49 INFO mapreduce.Job: map 73% reduce 7%
19/12/06 17:26:50 INFO mapreduce.Job: map 73% reduce 8%
19/12/06 17:26:51 INFO mapreduce.Job: map 75% reduce 8%
19/12/06 17:26:52 INFO mapreduce.Job: map 77% reduce 8%
19/12/06 17:26:53 INFO mapreduce.Job: map 80% reduce 8%
19/12/06 17:26:56 INFO mapreduce.Job: map 82% reduce 9%
19/12/06 17:26:58 INFO mapreduce.Job: map 85% reduce 9%
19/12/06 17:26:59 INFO mapreduce.Job: map 88% reduce 9%
19/12/06 17:27:00 INFO mapreduce.Job: map 90% reduce 9%
19/12/06 17:27:02 INFO mapreduce.Job: map 90% reduce 10%
19/12/06 17:27:03 INFO mapreduce.Job: map 93% reduce 10%
19/12/06 17:27:05 INFO mapreduce.Job: map 95% reduce 10%
19/12/06 17:27:06 INFO mapreduce.Job: map 100% reduce 10%
19/12/06 17:27:08 INFO mapreduce.Job: map 100% reduce 33%
19/12/06 17:27:10 INFO mapreduce.Job: map 100% reduce 67%
19/12/06 17:27:11 INFO mapreduce.Job: map 100% reduce 100%
19/12/06 17:27:12 INFO mapreduce.Job: Job job_1575649265022_0005 completed successfully
19/12/06 17:27:12 INFO mapreduce.Job: Counters: 51
File System Counters
  FILE: Number of bytes read=1009048
  FILE: Number of bytes written=10539504
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=4978889
```

# Steps to Check Output Results

- Show the content of the output files on HDFS

```
$ hadoop fs -ls /user/hadoop/[username]/output  
$ hadoop fs -cat /user/hadoop/[username]/output/part-r-00002
```

- Merge & get the output files to local file system

```
$ hadoop fs -getmerge /user/hadoop/[username]/output output.txt
```

- Show the content of the output files to TA

```
$ tail output.txt
```



# Hadoop Web UI

## ■ Cluster Hadoop Status

➤ [http://\[Hadoop Master IP\]:9870](http://[Hadoop Master IP]:9870)

## ■ MapReduce Job Tracker

➤ [http://\[Hadoop Master IP\]:8088](http://[Hadoop Master IP]:8088)

## ■ Job History Server

➤ [http://\[Hadoop Master IP\]:19888](http://[Hadoop Master IP]:19888)

You must follow the instructions below to access the web portal on your EMR master  
(suggest to use option2)

[https://docs.aws.amazon.com/zh\\_tw/emr/latest/ManagementGuide/emr-web-interfaces.html](https://docs.aws.amazon.com/zh_tw/emr/latest/ManagementGuide/emr-web-interfaces.html)

# Cluster Hadoop Status

## ■ Browser HDFS on web console (port:9870)

**Hadoop** Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities

### Overview 'apollo31:9000' (active)

<b>Started:</b>	Fri Apr 27 11:45:04 +0800 2018
<b>Version:</b>	2.9.0, r756ebc8394e473ac25feac05fa493f6d612e6c50
<b>Compiled:</b>	Tue Nov 14 07:15:00 +0800 2017 by arsuresh from branch-2.9.0
<b>Cluster ID:</b>	CID-8c667c4c-a58b-41ef-bba5-4dce9296e20e
<b>Block Pool ID:</b>	BP-1793204651-10.88.88.31-1512724516979

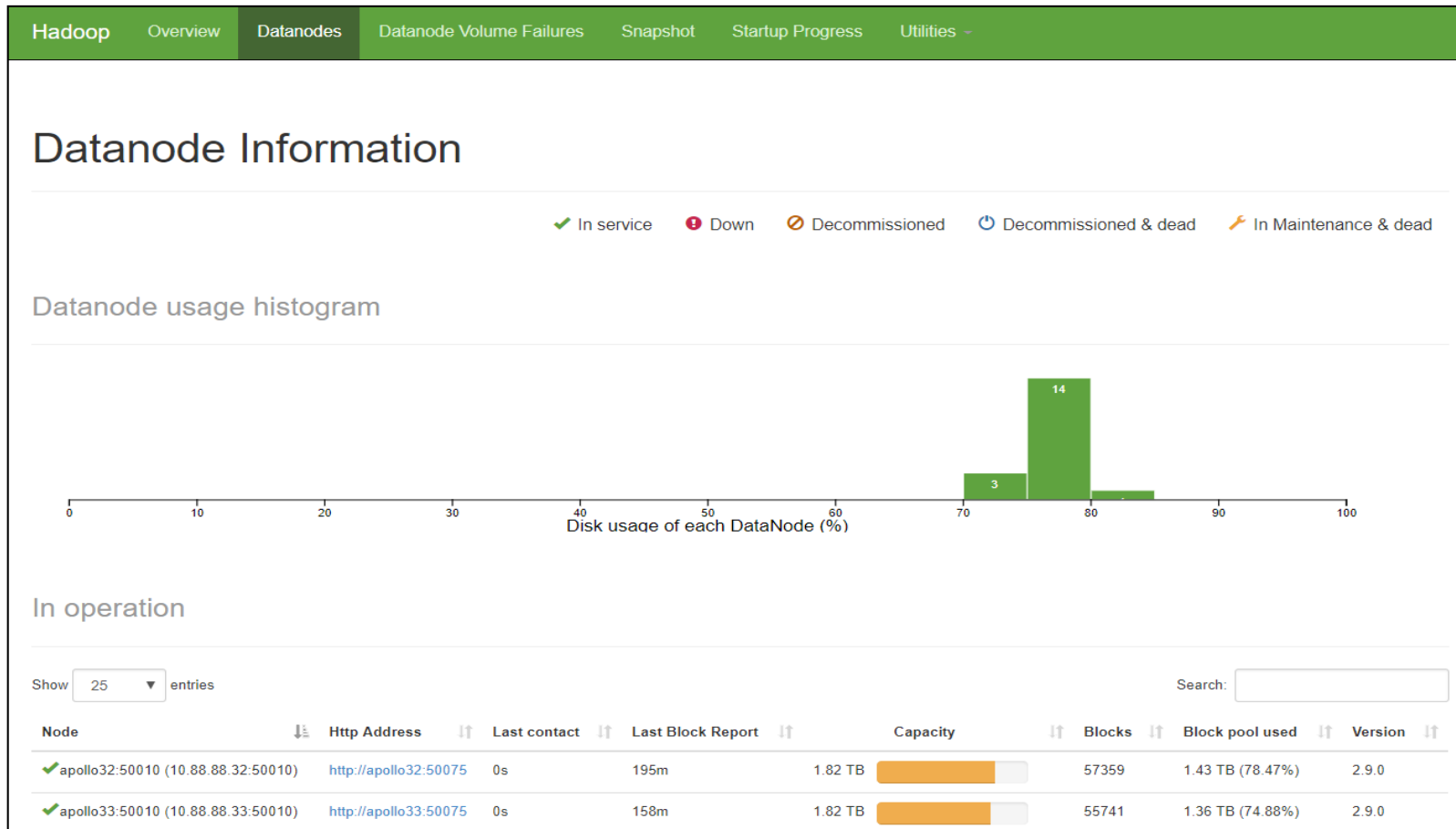
### Summary

Security is off.  
Safemode is off.  
533,892 files and directories, 525,110 blocks = 1,059,002 total filesystem object(s).  
Heap Memory used 278.74 MB of 705 MB Heap Memory. Max Heap Memory is 889 MB.  
Non Heap Memory used 63.8 MB of 65.06 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.



# Cluster Hadoop Status

## ■ Browser HDFS on web console (port:9870)



# Cluster Hadoop Status

## ■ Browser HDFS on web console (port:9870)

The screenshot displays the Hadoop web console interface. The top navigation bar includes links for Hadoop, Overview, Datanodes, Datanode Volume Failures, Snapshot, Startup Progress, and Utilities. The Utilities menu is expanded, showing options for 'Browse the file system' and 'Logs'. The main section is titled 'Browse Directory' and shows the root path '/'. Below the path bar, there is a search input field and a 'Go!' button. The directory listing shows 4 entries with columns for Permission, Owner, Group, Size, Last Modified, Replication, Block Size, and Name. The entries are mnt, system, tmp, and user.

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	hadoop	supergroup	0 B	Jan 19 11:15	0	0 B	mnt
drwxr-xr-x	hadoop	supergroup	0 B	Jan 22 22:32	0	0 B	system
drwxrwxrwt	hadoop	supergroup	0 B	Apr 28 13:34	0	0 B	tmp
drwxr-xr-x	hadoop	supergroup	0 B	Apr 13 21:37	0	0 B	user

# Job Tracker

- Browser all the job execution status (port:8088)



## All Applications

▼ Cluster

About

Nodes

Node Labels

Applications

NEW

NEW SAVING

SUBMITTED

ACCEPTED

RUNNING

FINISHED

FAILED

KILLED

Scheduler

► Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Mem...
3	0	0	3	0	0 B	24 GB	0 B

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy N...
2	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[MEMORY]	<memory:32, vCores:1>	<memory:12288, vCores:8>

Show 20 ▼ entries

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers
<a href="#">application_1575713584863_0003</a>	pp19s50	wordcount	MAPREDUCE	default	0	Sat Dec 7 18:30:33 +0800 2019	Sat Dec 7 18:31:37 +0800 2019	FINISHED	SUCCEEDED	N/A
<a href="#">application_1575713584863_0002</a>	hadoop	Spark shell	SPARK	default	0	Sat Dec 7 18:21:31 +0800 2019	Sat Dec 7 18:21:54 +0800 2019	FINISHED	SUCCEEDED	N/A
<a href="#">application_1575713584863_0001</a>	hadoop	HIVE-3fbbadbc-46c7-425a-	TEZ	default	0	Sat Dec 7 18:21:07	Sat Dec 7 18:26:18 +0800	FINISHED	SUCCEEDED	N/A

# Job History Server

## ■ Browser job history (port:19888)

← → ↻ 🏠 ⓘ Not secure | 19888/jobhistory/



# JobHistory

▸ Application

▼ Tools

- [Configuration](#)
- [Local logs](#)
- [Server stacks](#)
- [Server metrics](#)

### Retired Jobs

Show 20 ▼ entries

Submit Time ↕	Start Time ↕	Finish Time ▼	Job ID ↕
2019.03.25 11:05:00 CET	2019.03.25 11:05:07 CET	2019.03.25 11:08:51 CET	<a href="#">job_1553465137181_4755</a>
2019.03.25 11:04:50	2019.03.25 11:04:57	2019.03.25 11:08:04	<a href="#">job_1553465137181_4754</a>

# Job History Server

## ■ Check the log file in cluster mode

### ➤ Access Job History Server

1

Job ID	Name
<b>job_1513147375415_3573</b>	WordCount

2

Task Type
Map
<b>Reduce</b>

3

Name
<b>task_1513147375415_3573_r_000000</b>
task_1513147375415_3573_r_000001

4

Logs	St
<b>logs</b>	Tue
5:8042	17:
	+08

5

Log Type: prelaunch.err  
Log Upload Time: Tue Dec 26 17:02:45 +0800 2017  
Log Length: 0

Log Type: prelaunch.out  
Log Upload Time: Tue Dec 26 17:02:45 +0800 2017  
Log Length: 70  
Setting up env variables  
Setting up job resources  
Launching container

Log Type: stderr  
Log Upload Time: Tue Dec 26 17:02:45 +0800 2017  
Log Length: 0

Log Type: stdout  
Log Upload Time: Tue Dec 26 17:02:45 +0800 2017  
Log Length: 333  
Hadoop!!  
Hadoop!!  
Hadoop!!

# Cheat Sheet Hive for SQL Users

- Complete reference: <http://tw.gitbook.net/hive/index.html>
- <http://hortonworks.com/wp-content/uploads/2016/05/Hortonworks.CheatSheet.SQLtoHive.pdf>

Function	MySQL	HiveQL
Retrieving information	<code>SELECT from_columns FROM table WHERE conditions;</code>	<code>SELECT from_columns FROM table WHERE conditions;</code>
All values	<code>SELECT * FROM table;</code>	<code>SELECT * FROM table;</code>
Some values	<code>SELECT * FROM table WHERE rec_name = "value";</code>	<code>SELECT * FROM table WHERE rec_name = "value";</code>
Multiple criteria	<code>SELECT * FROM table WHERE rec1="value1" AND rec2="value2";</code>	<code>SELECT * FROM TABLE WHERE rec1 = "value1" AND rec2 = "value2";</code>
Selecting specific columns	<code>SELECT column_name FROM table;</code>	<code>SELECT column_name FROM table;</code>
Retrieving unique output records	<code>SELECT DISTINCT column_name FROM table;</code>	<code>SELECT DISTINCT column_name FROM table;</code>
Sorting	<code>SELECT col1, col2 FROM table ORDER BY col2;</code>	<code>SELECT col1, col2 FROM table ORDER BY col2;</code>
Sorting backward	<code>SELECT col1, col2 FROM table ORDER BY col2 DESC;</code>	<code>SELECT col1, col2 FROM table ORDER BY col2 DESC;</code>
Counting rows	<code>SELECT COUNT(*) FROM table;</code>	<code>SELECT COUNT(*) FROM table;</code>
Grouping with counting	<code>SELECT owner, COUNT(*) FROM table GROUP BY owner;</code>	<code>SELECT owner, COUNT(*) FROM table GROUP BY owner;</code>
Maximum value	<code>SELECT MAX(col_name) AS label FROM table;</code>	<code>SELECT MAX(col_name) AS label FROM table;</code>
Selecting from multiple tables (Join same table using alias w/"AS")	<code>SELECT pet.name, comment FROM pet, event WHERE pet.name = event.name;</code>	<code>SELECT pet.name, comment FROM pet JOIN event ON (pet.name = event.name);</code>

# Hands-on Lab: Hive

- Run in Hive SHELL

```
$ hive
```

- Create a database named by your studentID

```
hive> CREATE DATABASE test_[username];
```

- Create a table named “exam” under your DB with the following schema: id(int), name(string), score(int)

```
hive> CREATE TABLE test_[username].exam (id int, name string, score int)  
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';
```

- Load data to the table from filepath '/home/hive/sample.txt'

```
hive> LOAD DATA LOCAL INPATH '/home/[username]/lab/hive/sample.txt'  
OVERWRITE INTO TABLE test_[username].exam;
```

- Show all the content in table

```
hive> SELECT * from test_[username].exam;
```

# Hands-on Lab: Hive

## ■ Compute the average score from the table

```
hive> SELECT avg(score) from test_[username].exam;
```

```
Query ID = jchou_20191206171640_e0a851e3-8854-4197-89b2-c723f3eabfce
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1575649265022_0004)
```

VERTICES	MODE	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED	KILLED
Map 1 .....	container	SUCCEEDED	1	1	0	0	0	0
Reducer 2 .....	container	SUCCEEDED	1	1	0	0	0	0

```
VERTICES: 02/02 [=====>>] 100% ELAPSED TIME: 5.05 s
OK
81.66666666666667
Time taken: 11.068 seconds, Fetched: 1 row(s)
```





- spark-shell

- Show your output result
- Spark can also be written in Python and run on PySpark
- Reference: <https://spark.apache.org/examples.html>

# Hands-on Lab: Spark

- Type **WordCount** scala code line-by-line in Spark Shell

```
val textFile = sc.textFile("/user/hadoop/[username]/input/hello.dat")
val counts = textFile.flatMap(line => line.split(" ")) .map(word =>
(word, 1)) .reduceByKey(_ + _)
counts.saveAsTextFile("/user/spark/[username]")
```

- Show your output result

```
$ hadoop fs -cat /user/spark/[username]/part-00001
```

```
[hadoop@ip-172-31-90-14 WordCount]$ hadoop fs -ls /user/spark/123456/
Found 3 items
-rw-r--r--   1 hadoop spark          0 2019-12-06 16:09 /user/spark/123456/_SUCCESS
-rw-r--r--   1 hadoop spark        20 2019-12-06 16:09 /user/spark/123456/part-00000
-rw-r--r--   1 hadoop spark         9 2019-12-06 16:09 /user/spark/123456/part-00001
[hadoop@ip-172-31-90-14 WordCount]$ hadoop fs -cat /user/spark/123456/part-00000
(hello,2)
(world,1)
[hadoop@ip-172-31-90-14 WordCount]$ hadoop fs -cat /user/spark/123456/part-00001
(back,1)
```



# Spark Examples

## ■ WordCount

HDFS location



```
val textFile = sc.textFile("<input_directory_path>")
val counts = textFile.flatMap(line => line.split(" ")) .map(word
=> (word, 1)) .reduceByKey(_ + _)
counts.saveAsTextFile("<input_directory_path>")
```

## ■ Pi Estimation

```
val count = sc.parallelize(1 to NUM_SAMPLES).filter { _ =>
    val x = math.random
    val y = math.random x*x + y*y < 1 }.count()
println(s"Pi is roughly ${4.0 * count / NUM_SAMPLES}")
```

# Spark Examples

## ■ Prediction with Logistic Regression

```
// Every record of this DataFrame contains the label and  
// features represented by a vector.  
val df = sqlContext.createDataFrame(data).toDF("label", "features")  
// Set parameters for the algorithm.  
// Here, we limit the number of iterations to 10.  
val lr = new LogisticRegression().setMaxIter(10)  
// Fit the model to the data.  
val model = lr.fit(df)  
// Inspect the model: get the feature weights.  
val weights = model.weights  
// Given a dataset, predict each point's label, and show the results.  
model.transform(df).show()
```



Custom key & value types

Combiner

Partitioner

GroupingComparator

SortComparator

**ADVANCED PROG.**

# Custom Value Types

## ■ *Value in 3-dimensional coordinate*

```
struct point3d { float x; float y; float z; }
```

## ■ Implement *Writable* interface

- write: data serialization
- readFields: data de-serialization

```
public class Point3D implements Writable {  
    private float x; private float y; private float z;  
    public Point3D(float x, float y, float z) { this.x = x; this.y = y; this.z = z; }  
    public void write(DataOutput out) throws IOException {  
        out.writeFloat(x); out.writeFloat(y); out.writeFloat(z);  
    }  
    public void readFields(DataInput in) throws IOException  
        { x = in.readFloat(); y = in.readFloat(); z = in.readFloat(); }  
}
```

# Custom Key Types

- *Key in 3-dimensional coordinate*

```
struct point3d { float x; float y; float z; }
```

- Implement all functions in the **writable** interface

- write(), readFields()

- Implement additional functions in the **writablecomparable** interface

- compareTo(): used for **sorting**

- ◆ Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

- hashCode(): used for **partitioning**

# Custom Key Types

```
public class Point3D implements WritableComparable <Point3D> {  
    private float x; private float y; private float z;  
    public 3DPoint (){x=0.0f; y=0.0f; z=0.0f;}  
    public void set(float x, float y, float z) { this.x = x; this.y = y; this.z = z; }  
    public float distanceFromOrigin() {  
        return (float)Math.sqrt(x*x + y*y + z*z);  
    }  
    public int compareTo(Point3D other) {  
        float myDistance = distanceFromOrigin();  
        float otherDistance = other.distanceFromOrigin();  
        return Float.compare(myDistance, otherDistance);  
    }  
    public int hashCode() {  
        return Float.floatToIntBits(x) ^ Float.floatToIntBits(y) ^  
            Float.floatToIntBits(z);  
    }  
    // overwrite other methods in Writable interface: write & readFields  
}
```



# Use Case Example

- Given a list of 3D-coordinates, sort them in order in each of the output file:
  - **key type:** Point3D
  - **Value type:** NullWritable
  - **Mapper:** map each line to {<x,y,z>, Null}
  - **Reducer:** write key to file

➔ **Data is sorted automatically by Key in the MapReduce process**

# Point3D Sorting Example

```
public class TestPoint3D {  
    public static class TokenizerMapper  
        extends Mapper<Object, Text, Point3D, NullWritable>  
    {  
        private Point3D point = new Point3D();  
        public void map(Object key, Text value, Context context  
            ) throws IOException, InterruptedException {  
            String line = value.toString();  
            String[] tokens = line.split(",");  
            float x = Float.parseFloat(tokens[0]);  
            float y = Float.parseFloat(tokens[1]);  
            float z = Float.parseFloat(tokens[2]);  
            point.set(x,y,z);  
            context.write(point, NullWritable.get());  
        }  
    }  
}
```

Input file:

0,0,0  
1,0,2  
4,4,4  
2,2,2



Output file:

0,0,0  
1,0,2  
2,2,2  
4,4,4

main():

```
job.setOutputKeyClass(Point3D.class);  
job.setOutputValueClass(NullWritable.class);  
job.setMapClass(Tokenizer.class);
```



Custom key & value types

**Combiner**

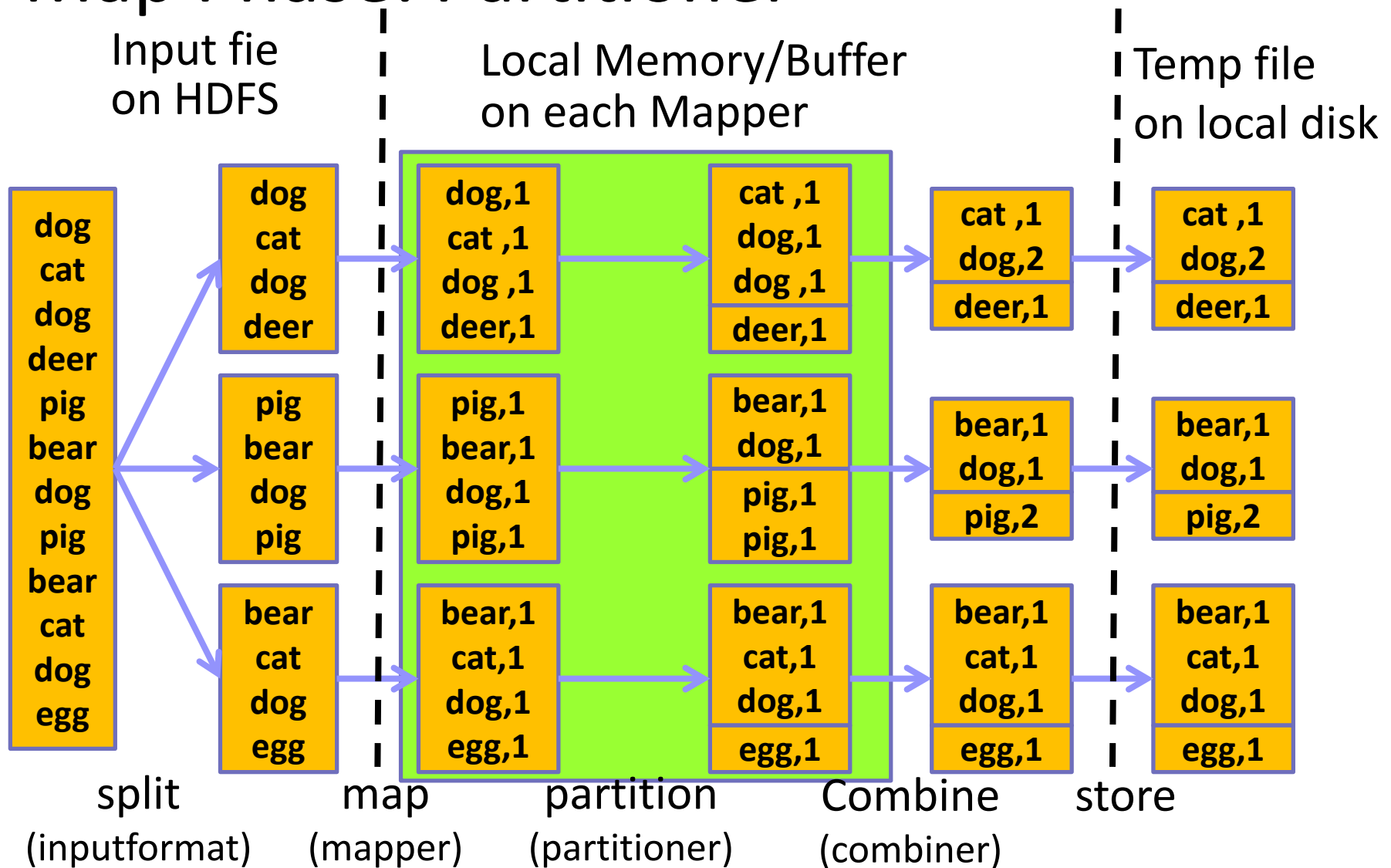
**Partitioner**

GroupingComparator

SortComparator

**ADVANCED PROG.**

# Map Phase: Partitioner



# Map Phase: Partitioner

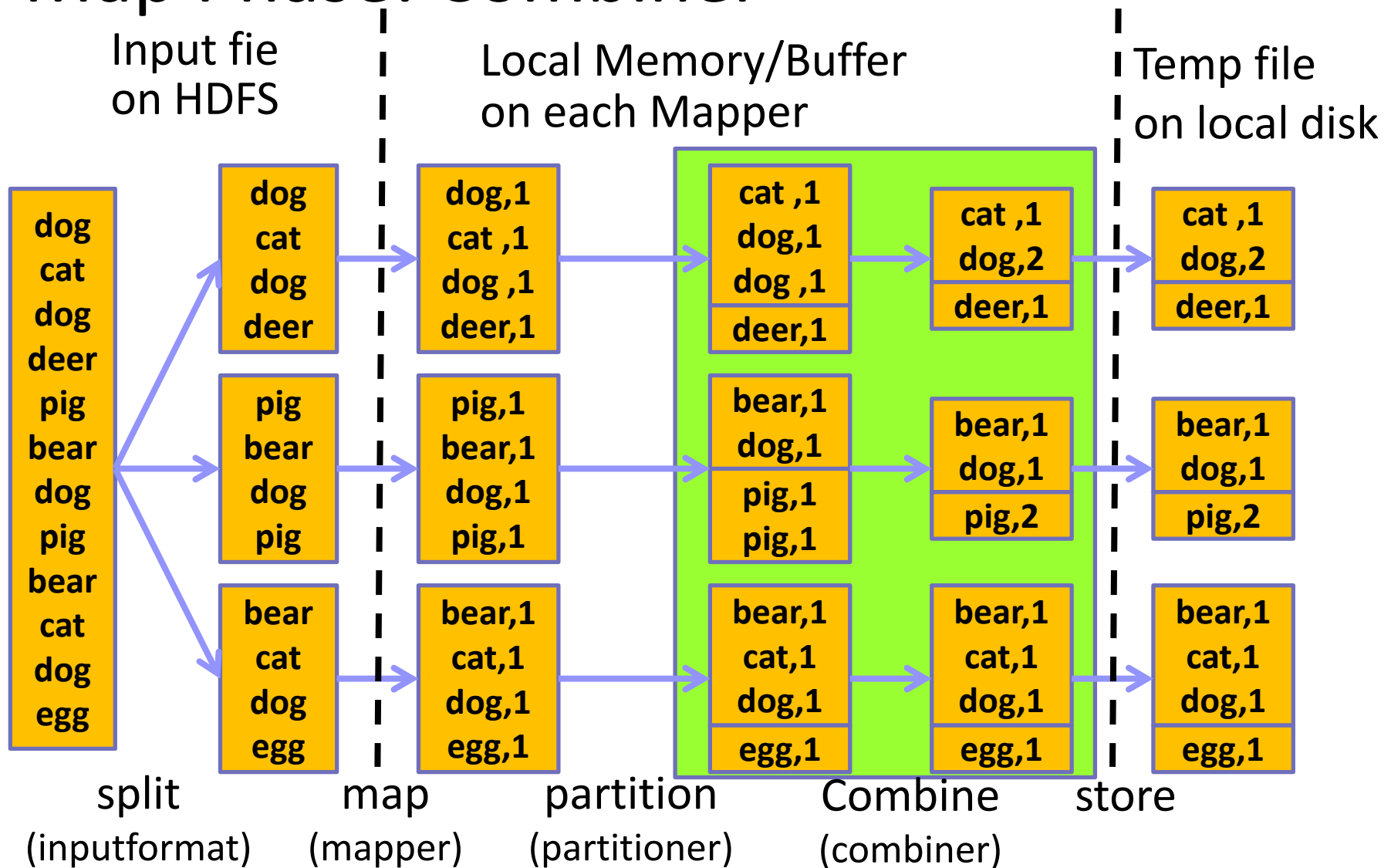
- Partitioner decides which intermediate (K,V) pair is sent to **which reducer**
- The total number of partitions is the same as the number of reduce tasks for the job.
- Default partitioner: “**HashPartitioner**”
- Write a custom Partitioner:

```
public class MyPartitioner implements Partitioner<Point3D, Writable> {  
    public int getPartition(Point3D key, Writable value, int numPart) {  
        return Math.abs(key.hashCode()) % numPart;  
    }  
}
```

Total number  
of partitions

```
main(){  
    job.setPartitionerClass(MyPartitioner.class);  
}
```

# Map Phase: Combiner



# Map Phase: Combiner

- An **OPTIONAL optimization** step in mapping phase
  - Combiner combines map-outputs before being sent to the reducers → reduce intermediate file size and transfer time
  - Combiner could be run **many** or **ZERO** time → program results can't depend on combiner
  - $\langle K, V \rangle$  data type must be the **same** for INPUT & OUTPUT
    - ◆ Reducer can emit a different output type to file
  - Reducer and combiner could be but **NOT ALWAYS** the same
    - ◆ E.g.: compute the avg of each key
    - ◆  $\text{MEAN}(\{1,2,3,4,5\}) \neq \text{MEAN}(\text{MEAN}(\{1,2\}), \text{MEAN}(\{3,4,5\}))$
  - Some problem can be difficult to apply combiner
    - ◆ E.g.: Find the median value of each key



Custom key & value types

Combiner

Partitioner

GroupingComparator

SortComparator

**ADVANCED PROG.**

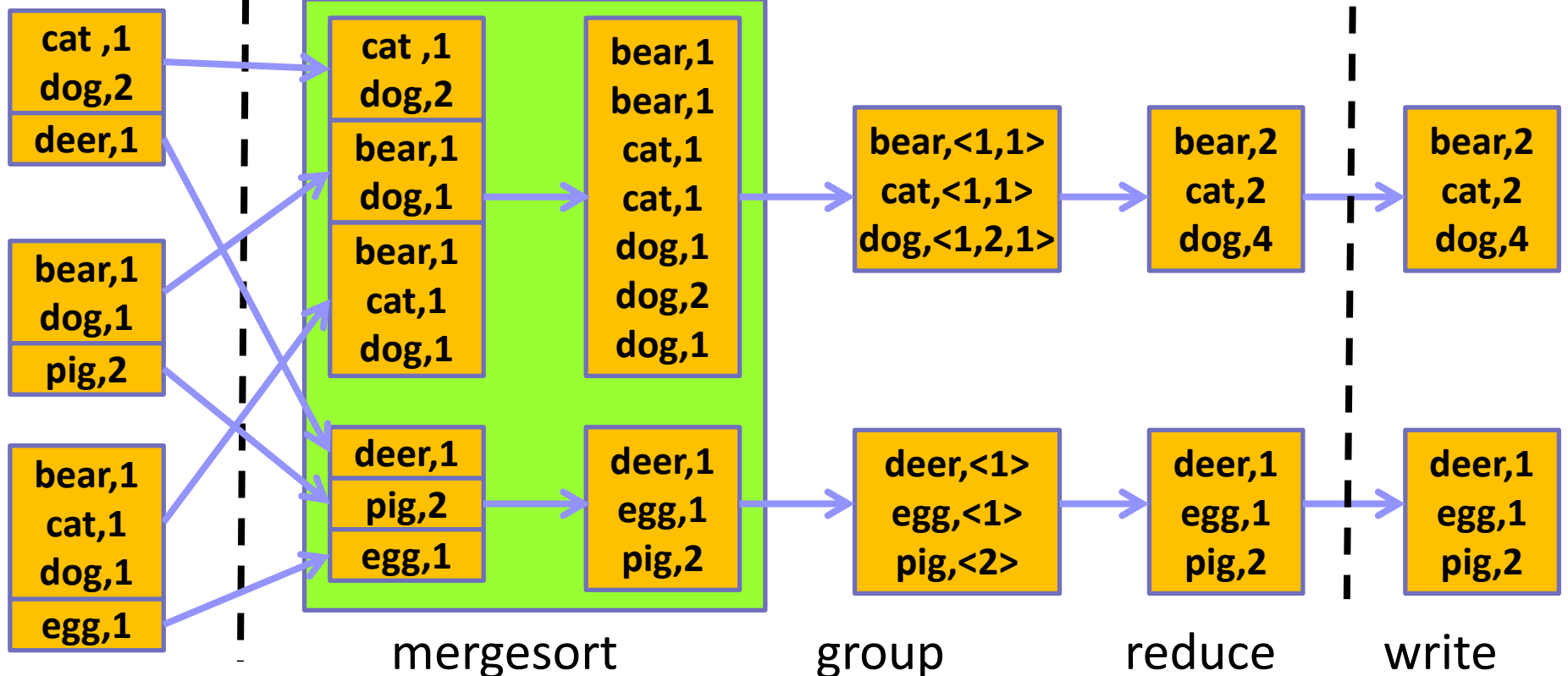


# Reduce Phase: sortComparator

Temp file  
Mapper's  
local disk

Local Memory/Buffer  
on each Reducer

Output file  
on HDFS



fetch&shuffle (sortComparator) (groupingComparator) (reducer) (outputformat)

# Reduce Phase: sortComparator

- $\langle K, V \rangle$  pairs are sorted by **key** using a comparator class called the **sortComparator**
  - The comparator can be set by **“job.setSortComparatorClass()”**
  - The comparator must implement the **“*rawComparator*” interface** or extend **“*writeComparator*” class**
    - ◆ Override the function: **compare**
- Implementation:
  - **Mergesort** is used by the framework to effectively merge the output from mappers, and sort the result in one stage

# Reduce Phase: sortComparator

- Let keys in the form of <string1>:<string2>
- Sort keys in the ascending order of <string1>

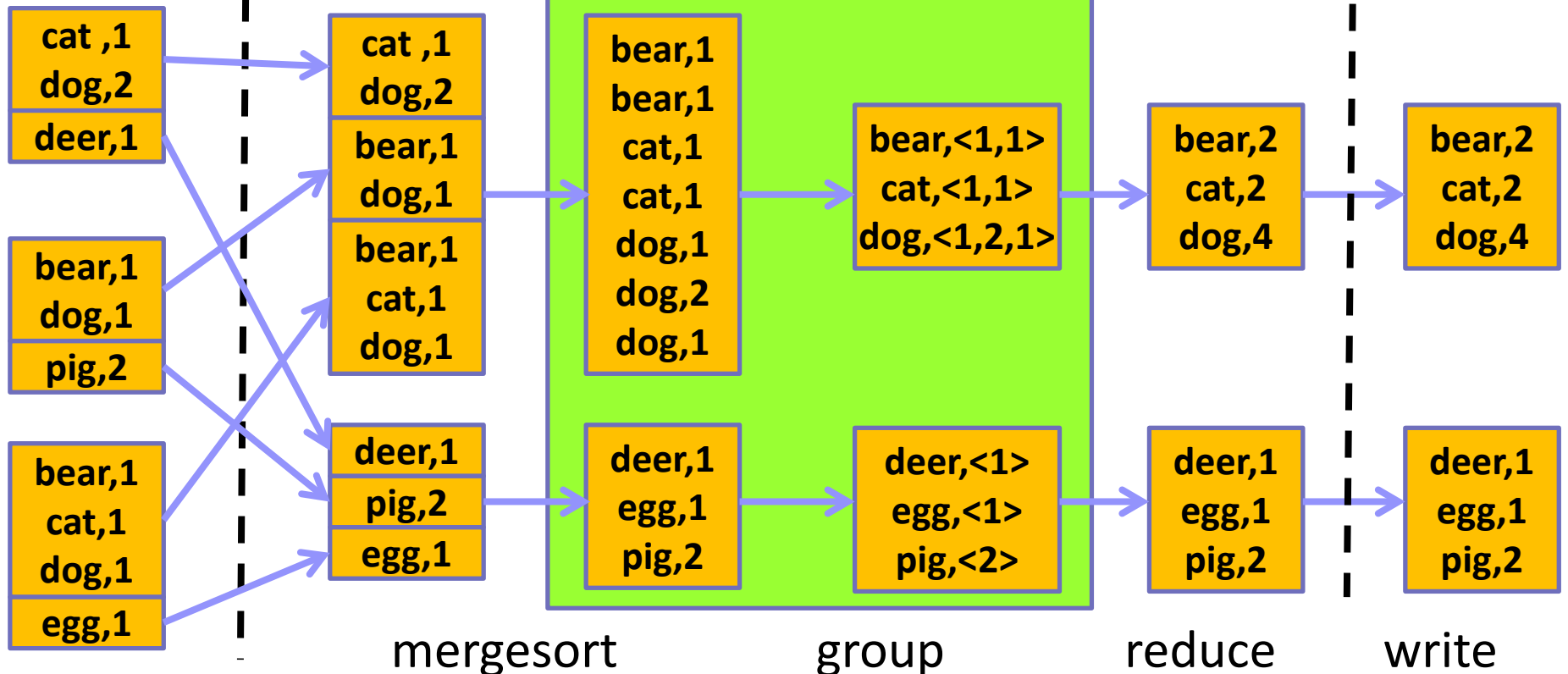
```
public static class MySortComparator extends WritableComparator {  
    protected MySortComparator() { super(Text.class, true); }  
    public int compare(WritableComparable w1,  
                      WritableComparable w2) {  
        Text t1 = (Text) w1;  
        Text t2 = (Text) w2;  
        String[] t1Items = t1.toString().split(":");  
        String[] t2Items = t2.toString().split(":");  
        return t1Items[0].compareTo(t2Items[0]);  
    }  
}  
  
main(){  
    job.setSortComparatorClass(MySortComparator.class);  
}
```

# Reduce Phase: groupingComparator

Temp file  
Mapper's  
local disk

Local Memory/Buffer  
on each Reducer

Output file  
on HDFS



fetch&shuffle (sortComparator) (groupingComparator) (reducer) (outputformat)

# Reduce Phase: groupingComparator

- $\langle K, V \rangle$  pairs are grouped together if their **keys** are compared as equal by using a comparator called **groupingComparator**
  - The comparator can be set by **`“job.setGroupingComparatorClass()”`**
  - The comparator must implement the **`“rawComparator”`** interface or extend **`“writeComparator”`** class
    - ◆ Override the function: **compare**
- If multiple keys in the same group, **`“sortComparator”`** is used to decide the key for the group
  - Input:  $\langle A1, V1 \rangle, \langle A2, V2 \rangle, \langle A3, V3 \rangle, \langle B1, V4 \rangle, \langle B2, V5 \rangle$
  - Grouping comparator to just compare the first letter
  - Output:  $(A1, \{V1, V2, V3\}); (B1, \{V4, V5\});$

# Reduce Phase: groupingComparator

- Only compare the first letter

```
public static class MyGroupComp extends WritableComparator {  
    protected MyGroupCom() { super(Text.class, true); }  
    public int compare(WritableComparable w1,  
                      WritableComparable w2) {  
        Text t1 = (Text) w1;          Text t2 = (Text) w2;  
        int t1char = t1.charAt(0);    int t2char = t2.charAt(0);  
        if (t1char < t2char) return -1;  
        else if (t1char > t2char) return 1;  
        else return 0;  
    }  
}
```

```
main(){  
    job.setGroupingComparatorClass(MyGroupComp.class);  
}
```



# SECONDARYSORT EXAMPLE

# SecondarySort

## ■ What is SecondarySort?

- **Sorting values** associated with a key in the reduce phase

## ■ Examples:

- Input: A dump of the temperature data with 4 columns  
year, month, day, daily\_temperature

- Output: The temperature for every  
year-month with the values sorted

```
2012-01: 5, 35, 45  
2001-11, 46, 47, 48
```

....

```
2012, 01, 01, 5  
2012, 01, 02, 45  
2012, 01, 03, 35  
...  
2001, 11, 01, 46  
2001, 11, 02, 47  
2001, 11, 03, 48
```



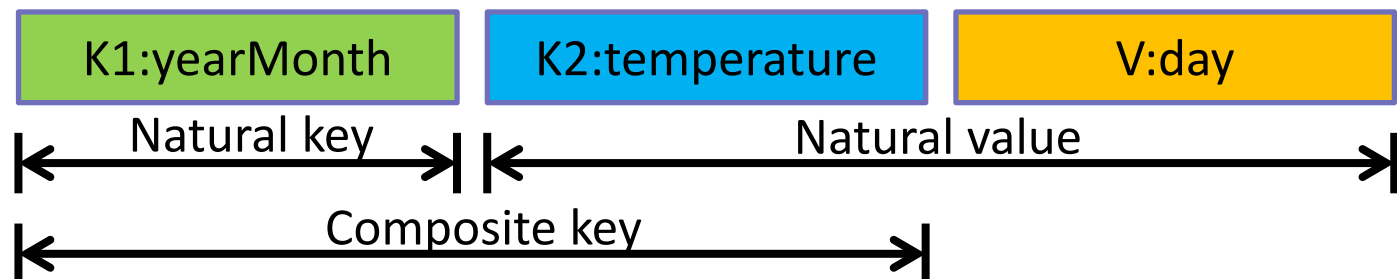
# SecondarySort

## ■ Solution1:

- having the reducer **buffer all of the values** for a given *key*
- then doing an **in-reducer sort** on the values
- ➔ might cause the reducer to run out of memory

## ■ Solution2:

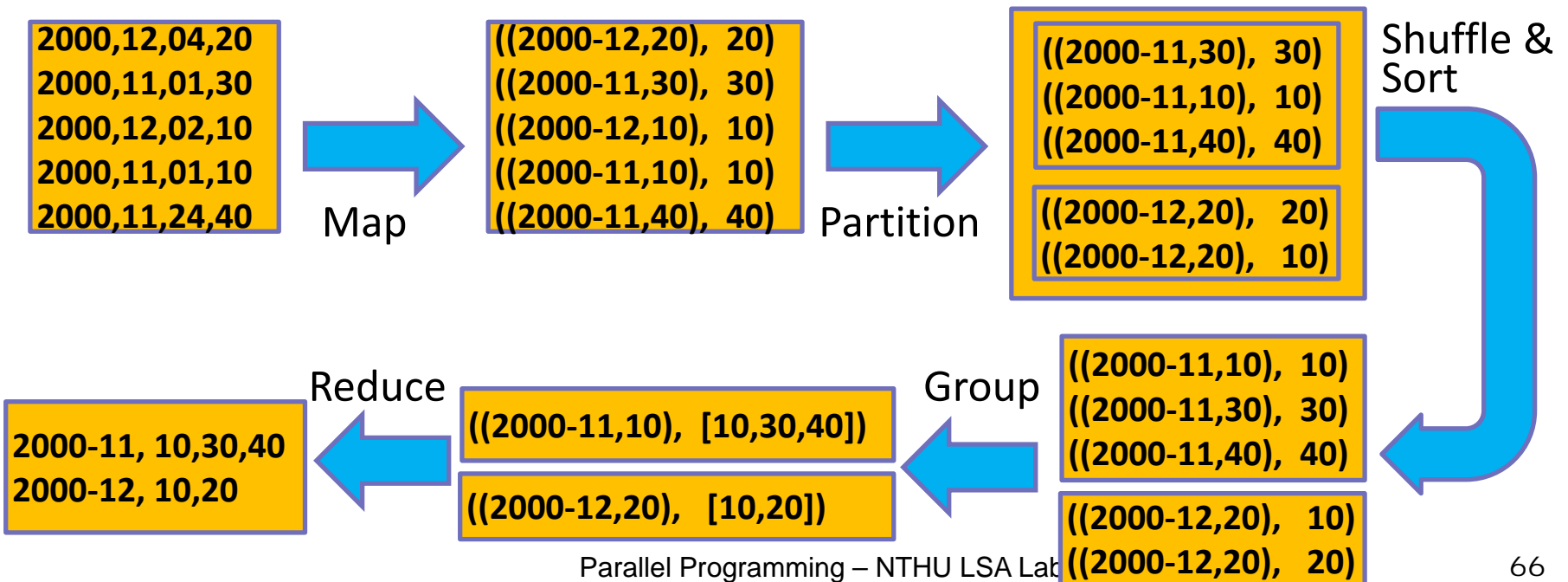
- Trick MapReduce to sort the reducer values
- *Value-to-Key Conversion design pattern*: “Creating a **composite key** by adding a part of, or the entire value to, the natural key to achieve your sorting objectives”



# SecondarySort

## ■ Implementation details:

- Map Output Key: {yearMonth}+{temperature}
- Map Output Value: temperature
- Partitioner: by yearMonth
- sortComparator: by yearMonth and then ascending temp.
- groupingComparator: by yearMonth



# Reference

- Distributed system lecture slides from Gregory Kesden
- Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. Proceedings of the 6th Symposium on Operating System Design and Implementation (OSDI 2004), pages 137-150
- Hadoop tutorial:
  - <https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html>