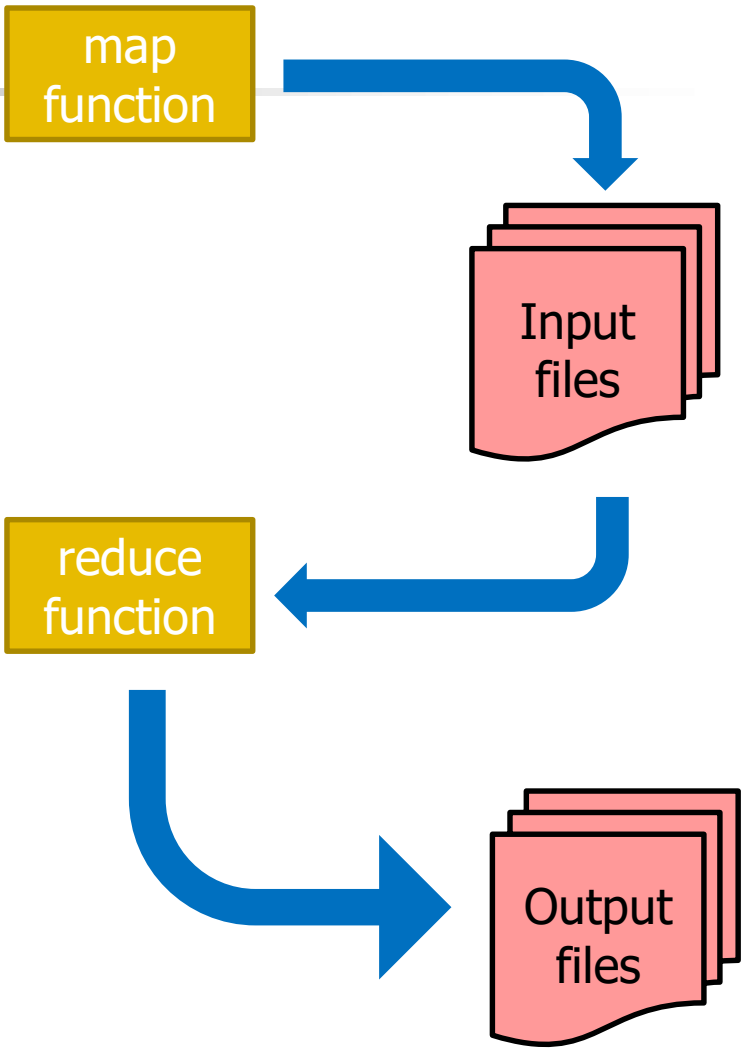
MapReduce

Applications

ISEL
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

MR applications

- MapReduce programming model is very simple, the user only need to implement the map and reduce functions that will be invoked by the underlying infrastructure in a transparent way!



Minimal MapReduce application

- In the context of the next examples we are going to assume that the input is contained in two files with the following contents:

File01.txt

Hello World, Bye World! The World, is blue. In the World again.

File02.txt

Hello Hadoop, Goodbye to hadoop.

MR applications

- The MapReduce programming model was design to hide infrastructure details such as the number of mappers and reducers, the type and location of input and output, etc.

- The next example represents the minimal MapReduce application that one can build

Minimal MapReduce application

Ex26

```
import ...;

public class MinimalMapReduce extends Configured implements Tool {
  public int run(String[] args) throws Exception {

    Job job = Job.getInstance( getConf() );
    job.setJarByClass( getClass() );
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    return job.waitForCompletion(true) ? 0 : 1;
  }

  public static void main(String[] args) throws Exception {
    int exitCode = ToolRunner.run(new MinimalMapReduce(), args);
    System.exit(exitCode);
  }
}
```

Minimal MapReduce application

usermr@hadoop: ~/examples/Projects/06-MapReduce/Ex26-MapReduce-01

```
usermr@hadoop:~/examples/Projects/06-MapReduce/Ex26-MapReduce-01$ cat ~/examples/input/gutenberg/small/file01.txt
Hello World, Bye World! The World, is blue.
In the World again.

usermr@hadoop:~/examples/Projects/06-MapReduce/Ex26-MapReduce-01$ cat ~/examples/input/gutenberg/small/file02.txt
Hello Hadoop, Goodbye to hadoop.

usermr@hadoop:~/examples/Projects/06-MapReduce/Ex26-MapReduce-01$ cat ~/examples/output/gutenberg/small/part-r-00000
0       Hello Hadoop, Goodbye to hadoop.
0       Hello World, Bye World! The World, is blue.
33
44      In the World again.
64
usermr@hadoop:~/examples/Projects/06-MapReduce/Ex26-MapReduce-01$
```

Minimal MapReduce application

- By default, the following configurations are assumed
  - Input format is of type `TextInputFormat` that generates:
    - Keys of type `LongWritable` (offset of the beginning of the line in the file)
    - Values of type `Text` (the line of text)
  - A mapper is supported in the `Mapper` class, which writes (to the output) the input key and value
  - The default partitioner is `HashPartitioner`
  - Each partition is processed by a map task, so the number of partitions is equal to the number of map tasks for the job
  - A reducer is supported in the `Reducer` class, which simply writes all its input to its output

Minimal MapReduce application

Ex27

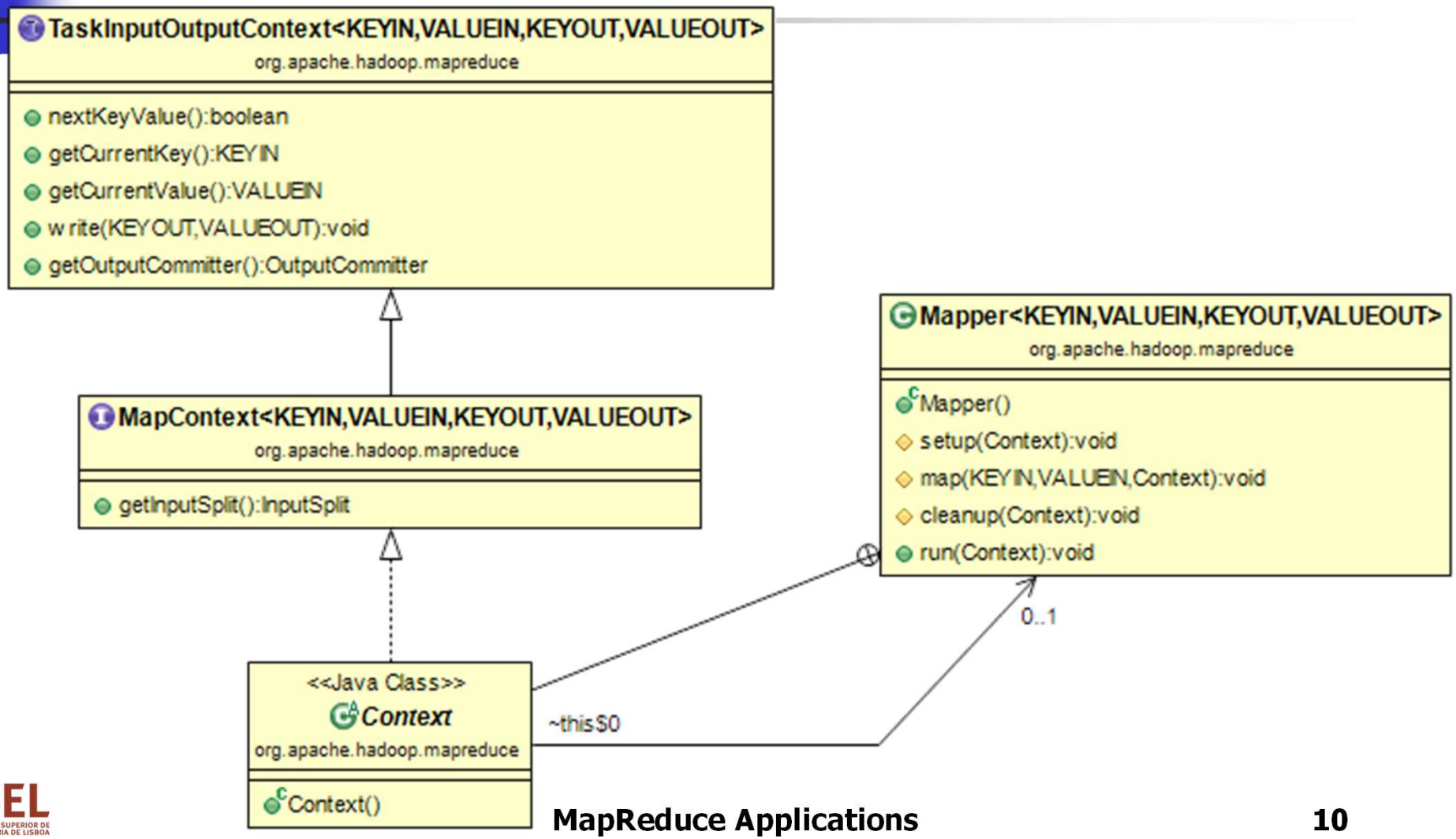
```
import ...;

public class MinimalMapReduceWithDefaults extends Configured implements Tool {
  public static void main(String[] args) throws Exception {
    int exitCode = ToolRunner.run(new MinimalMapReduceWithDefaults(), args);
    System.exit(exitCode);
  }

  @Override
  public int run(String[] args) throws Exception {
    if (args.length != 2) {
      System.err.printf("Usage: %s [options] <in> <out>\n", getClass() );
      ToolRunner.printGenericCommandUsage(System.err);
      return -1;
    }
    ...
```

Minimal MapReduce application

Ex27

```
    Job job = Job.getInstance( getConf() );
    job.setJarByClass( getClass() );
    FileInputFormat.addInputPath( job, new Path( args[ 0 ] ) );
    FileOutputFormat.setOutputPath( job, new Path( args[ 1 ] ) );
    job.setInputFormatClass( TextInputFormat.class );
    job.setMapperClass( Mapper.class);
    job.setMapOutputKeyClass( LongWritable.class );
    job.setMapOutputValueClass( Text.class );
    job.setPartitionerClass( HashPartitioner.class );
    job.setNumReduceTasks( 1 );
    job.setReducerClass( Reducer.class );
    job.setOutputKeyClass( LongWritable.class );
    job.setOutputValueClass( Text.class );
    job.setOutputFormatClass( TextOutputFormat.class );

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

Default mapper – UML



Default mapper

```
public class Mapper<KEYIN, VALUEIN, KEYOUT, VALUEOUT> {

  public abstract class Context
      implements MapContext<KEYIN,VALUEIN,KEYOUT,VALUEOUT> {}

  protected void setup(Context context)
      throws IOException, InterruptedException {}

  protected void map(KEYIN key, VALUEIN value, Context context)
      throws IOException, InterruptedException {
    context.write((KEYOUT) key, (VALUEOUT) value);
  }

  protected void cleanup(Context context)
      throws IOException, InterruptedException {}
  ...
```

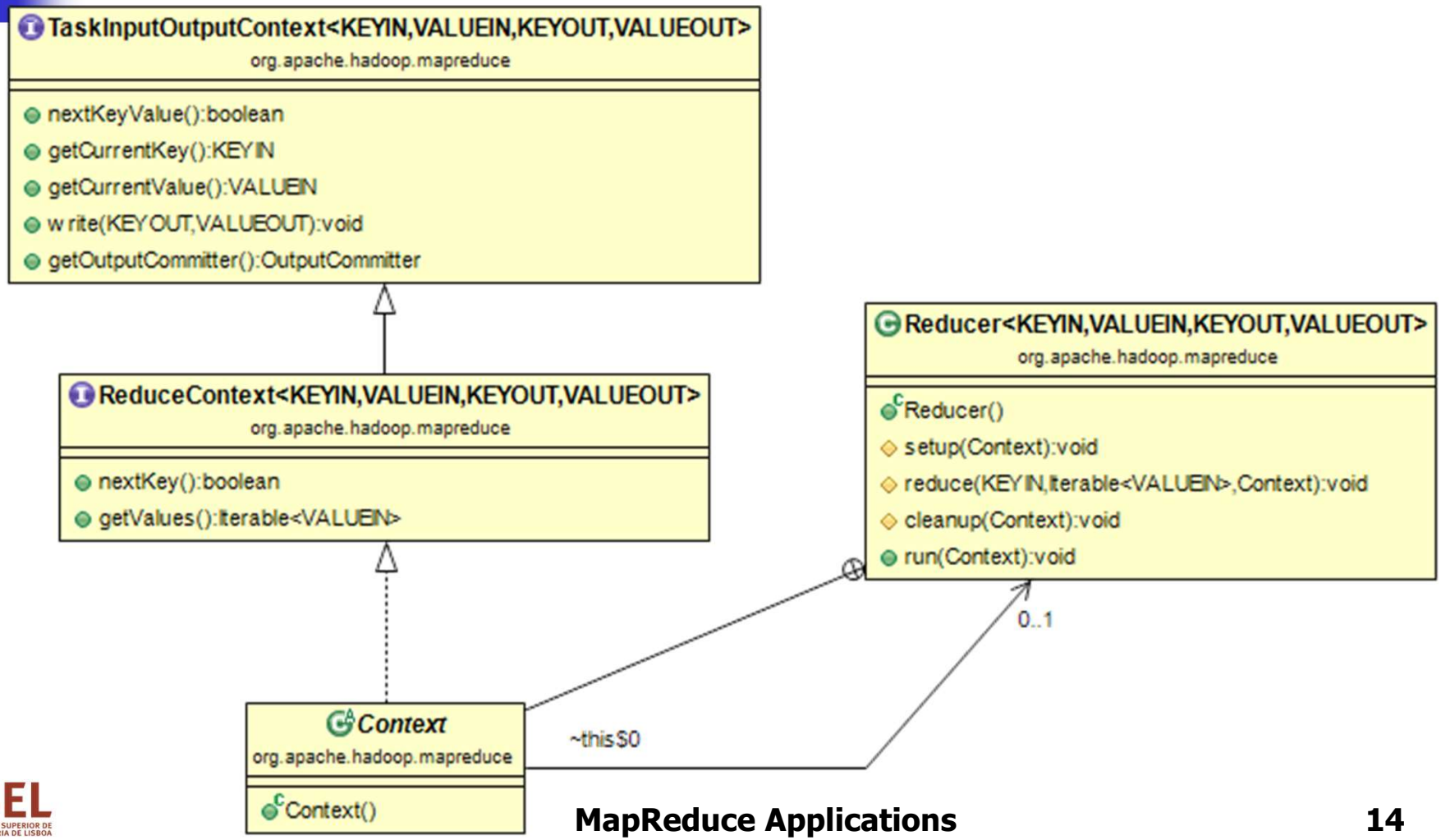
Default mapper

```
  ...
  public void run(Context context) throws IOException, InterruptedException {

    setup(context);

    try {
      while (context.nextKeyValue()) {
        map(context.getCurrentKey(), context.getCurrentValue(), context);
      }
    }
    finally {
      cleanup(context);
    }
  }
}
```

Default mapper

"After running `setup()`, the `nextKeyValue()` is called repeatedly on the `Context` (which delegates to the identically named method on the `RecordReade`r) to populate the key and value objects for the mapper. The key and value are retrieved from the `RecordReader` by way of the `Context` and are passed to the `map()` method for it to do its work. When the reader gets to the end of the stream, the `nextKeyValue()` method returns false, and the map task runs its `cleanup()` method and then completes." [1]

Default reducer – UML



Default reducer

```
public class Reducer<KEYIN,VALUEIN,KEYOUT,VALUEOUT> {

  public abstract class Context
      implements ReduceContext<KEYIN,VALUEIN,KEYOUT,VALUEOUT> {}

  protected void setup(Context context)
      throws IOException, InterruptedException {}

  protected void reduce(KEYIN key, Iterable<VALUEIN> values, Context context)
      throws IOException, InterruptedException {

    for(VALUEIN value: values) {
      context.write((KEYOUT) key, (VALUEOUT) value);
    }
  }
  ...
```

Default reducer

```
  ...
  protected void cleanup(Context context)
      throws IOException, InterruptedException {}

  public void run(Context context) throws IOException, InterruptedException {
    setup(context);
    try {
      while (context.nextKey()) {
        reduce(context.getCurrentKey(), context.getValues(), context);
        Iterator<VALUEIN> iter = context.getValues().iterator();
        if(iter instanceof ReduceContext.ValueIterator) {
          ((ReduceContext.ValueIterator<VALUEIN>)iter).resetBackupStore();
        }
      }
    } finally { cleanup(context); }
  }
}
```

`InputSplits` and `RecordReader`

- An input split is a chunk of the input that is processed by a single map, where each split is divided into records (key-value pairs)

- Splits and records are logical. For e.g., in a database context, a split might correspond to a range of rows from a table and a record to a row in that range

InputSplits and RecordReader

Partitioner<KEY,VALUE>
org.apache.hadoop.mapreduce

- Partitioner()
- *getPartition(KEY,VALUE,int):int*

RecordReader<KEYIN,VALUEIN>
org.apache.hadoop.mapreduce

- RecordReader()
- *initialize(InputSplit,TaskAttemptContext):void*
- *nextKeyValue():boolean*
- *getCurrentKey():KEYIN*
- *getCurrentValue():VALUEIN*
- *getProgress():float*
- *close():void*

Mapper<KEYIN,VALUEIN,KEYOUT,VALUEOUT>
org.apache.hadoop.mapreduce

- Mapper()
- setup(Context):void
- map(KEYIN,VALUEIN,Context):void
- cleanup(Context):void
- run(Context):void

InputSplit
org.apache.hadoop.mapreduce

- InputSplit()
- *getLength():long*
- *getLocations():String[]*
- getLocationInfo():SplitLocationInfo[]

InputFormat<K,V>
org.apache.hadoop.mapreduce

- InputFormat()
- *getSplits(JobContext):List<InputSplit>*
- *createRecordReader(InputSplit,TaskAttemptContext):RecordReader<K,V>*

InputSplit

```
public abstract class InputSplit {

  public abstract long getLength()
      throws IOException, InterruptedException;

  public abstract String[] getLocations()
      throws IOException, InterruptedException;

  public SplitLocationInfo[] getLocationInfo()
      throws IOException {
    return null;
  }
}
```

InputSplit

- An `InputSplit` has a length in bytes and a set of storage locations, which are just hostname strings
- A split doesn't contain the input data; it is just a reference to the data
- The storage locations are used by the MapReduce system to place map tasks as close to the split's data as possible, and the size is used to order the splits so that the largest get processed first, in an attempt to minimize the job runtime

InputFormat

```
public abstract class InputFormat<K, V> {

  public abstract List<InputSplit> getSplits(JobContext context)
      throws IOException, InterruptedException;

  public abstract RecordReader<K,V> createRecordReader(
        InputSplit split,
        TaskAttemptContext context)
      throws IOException, InterruptedException;
}
```

InputFormat

- A MapReduce application developer do not need to deal with `InputSplits` directly, as they are created by an `InputFormat`
- The `InputFormat` is responsible for creating the input splits and dividing them into records
- The client running the job calculates the splits for the job by calling `getSplits()`, then sends them to the application master, which uses their storage locations to schedule map tasks that will process them on the cluster

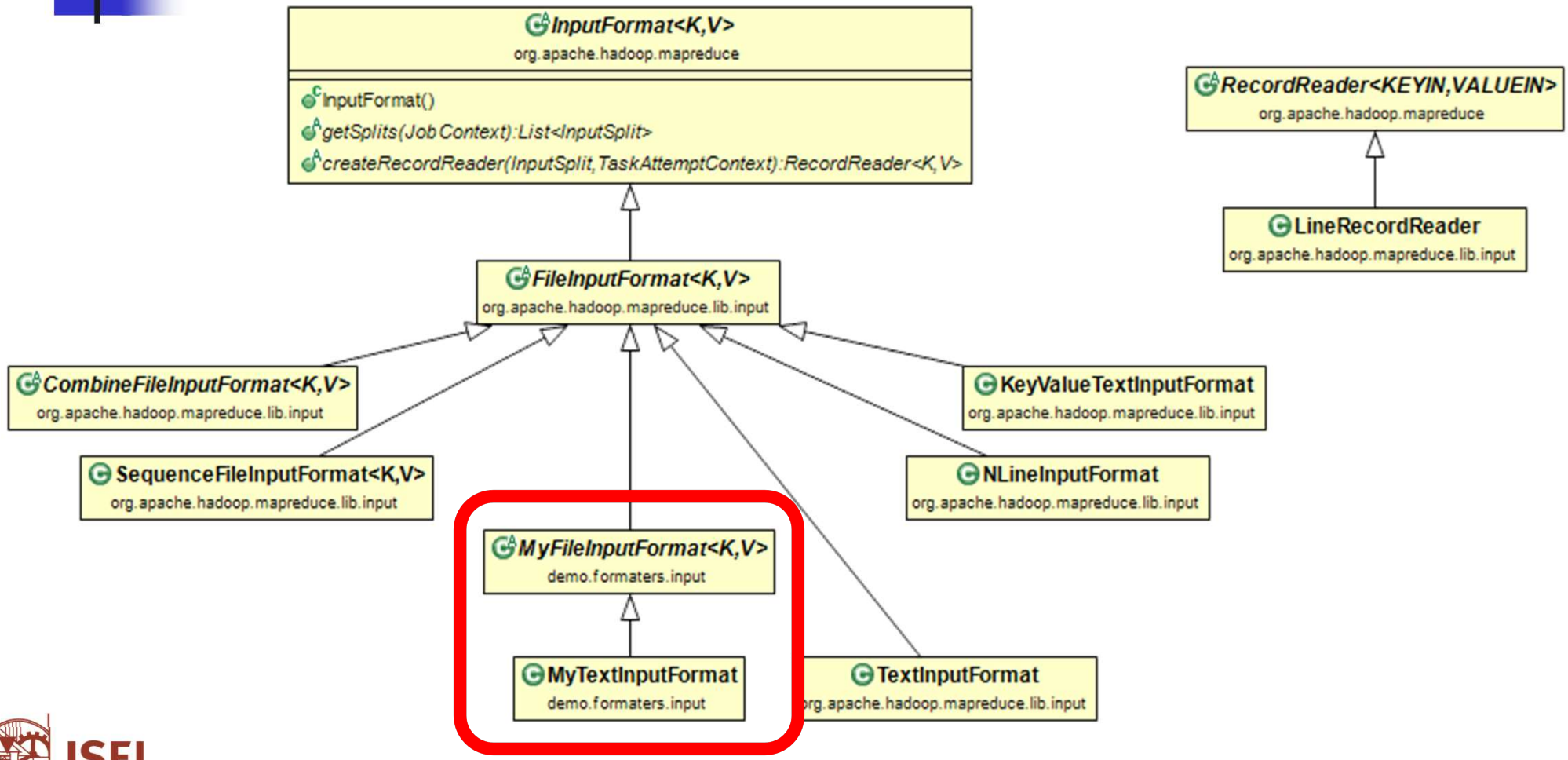
FileInputFormat

- `FileInputFormat` is the base class for all implementations of `InputFormat` that use files as their data source.

- It provides two things:
  - A place to define which files are included as the input to a job
  - An implementation for generating splits for the input files

- The job of dividing splits into records is performed by subclasses.

FileInputFormat – UML



My input formatters

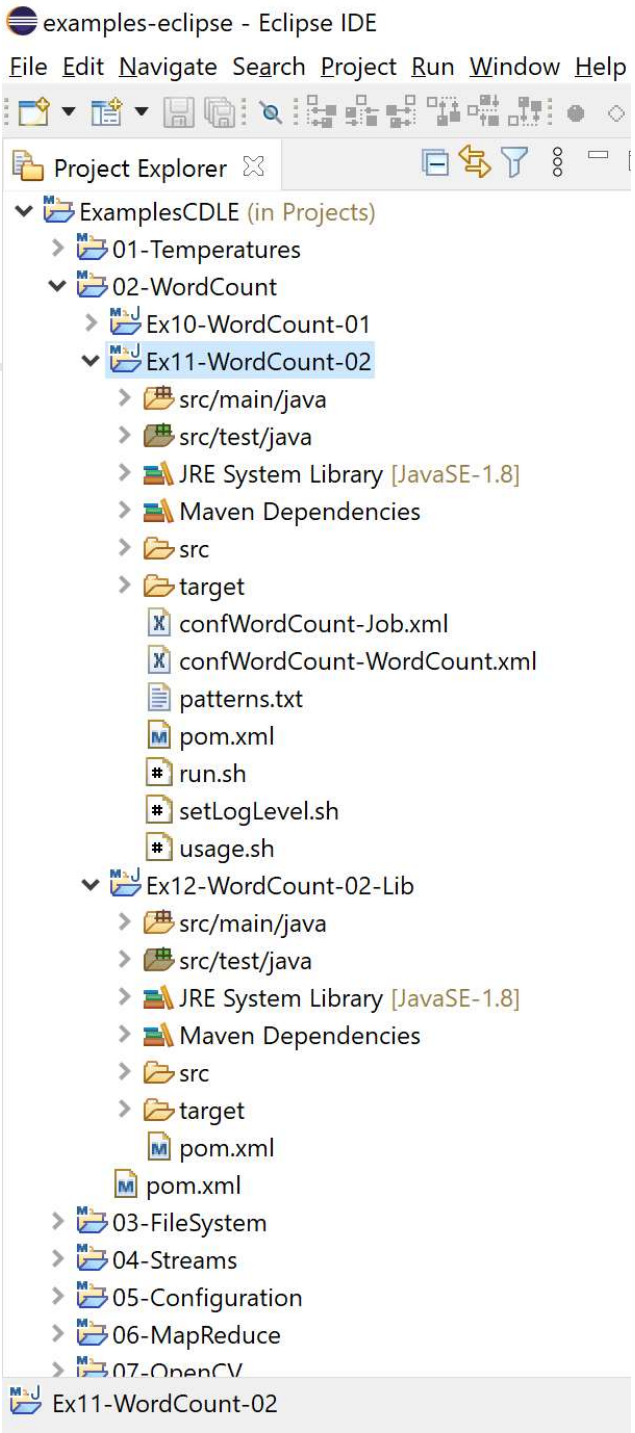
- The classes `MyFileInputFormat` and `MyTextInputFormat` were adapted from the originals `FileInputFormat` and `TextInputFormat` in order to add a set of debug messages

- This set of messages are written a file whose location by default is `/tmp/dbg.log`

My input formatters

- The aim of these files is to understand how the MapReduce infrastructure handles the input of data

- These classes are contained in the project "Ex11-…-02-Lib"



MyLogUtils

```
public class MyLogUtils {
  private static String lF = "/tmp/dbg.log";

  public static void setLogFile(String lF) { MyLogUtils.lF = lF;}

  public static void log(Log log, String msg) {
    try (PrintWriter out=new PrintWriter(new FileOutputStream(lF,true))){
      out.println(msg);
    }
    catch (Exception e) {
      e.printStackTrace( System.err );
    }

    System.out.println(msg); System.out.flush();
    log.debug(msg); log.info(msg);
  }
}
```

Ex12

MyFileInputFormat<K, V>

```
public abstract class MyFileInputFormat<K,V> extends FileInputFormat<K,V>{
  private static Log log = LogFactory.getLog( MyFileInputFormat.class );

  static { myLog( "MyFileInputFormat<K, V>#ctor" ); }

  private static void myLog(String msg) { MyLogUtils.log( log, msg); }

  public static void setInputDirRecursive(
      Job job,
      boolean inputDirRecursive) {
    FileInputFormat.setInputDirRecursive(job, inputDirRecursive);

    myLog( String.format("MyFileInputFormat#setInputDirRecursive(%s, %b)",
      job.getJobName(), inputDirRecursive ) );
  }
  ...
```

Ex12

MyTextInputFormat

```
public class MyTextInputFormat
      extends MyFileInputFormat<LongWritable, Text> {

  private static Log log = LogFactory.getLog( MyTextInputFormat.class);

  private static void myLog(String msg) { MyLogUtils.log( log, msg); }

  private TextInputFormat theFormater;

  public MyTextInputFormat() {
    super();

    this.theFormater = new TextInputFormat();

    myLog("MyTextInputFormat#ctor");
  }
  ...
```

Ex12

MyTextInputFormat

```
@Override
public RecordReader<LongWritable, Text> createRecordReader(
      InputSplit split, TaskAttemptContext context) {
  RecordReader<LongWritable, Text> result;
  result = theFormater.createRecordReader(split, context);

  myLog(
    String.format("MyTextInputFormat#createRecordReader(%s, %s) - >%s",
    split.toString(), context.getJobName(), result.toString() ) );
  return result;
}

...
```

Ex12

MyTextInputFormat

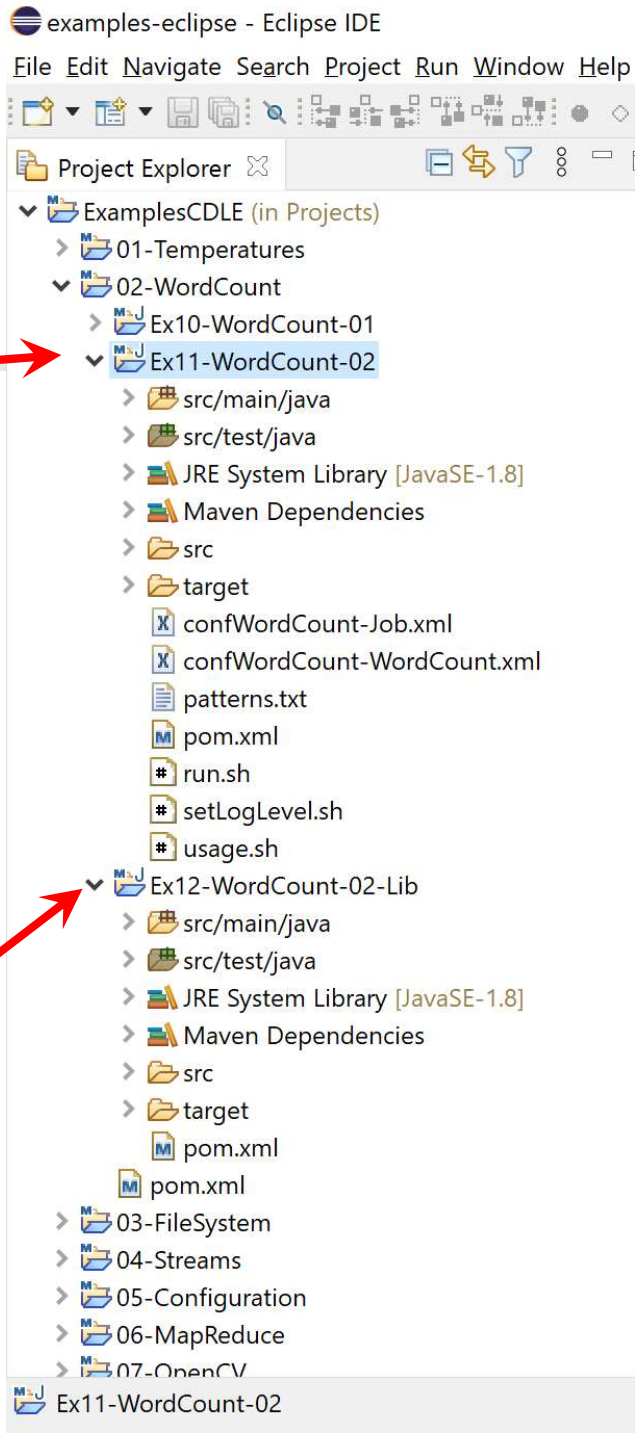
```
  @Override
  protected boolean isSplitable(JobContext ctx, Path f) {
    boolean result;
    final CompressionCodec codec;
    codec = new compressionCodecFactory(ctx.getConfiguration()).getCodec(f);
    if ( codec==null ) {
      result = true;
    }
    result = codec instanceof SplittableCompressionCodec;
    myLog(
      String.format( "MyTextInputFormat#isSplitable(%s, %s) -> %b",
      ctx.getJobName(), f.getName(), result ) );
    return result;
  }
}
```

Ex12

Everything in action

- Example "Ex11" represents a MapReduce application that is able to count words

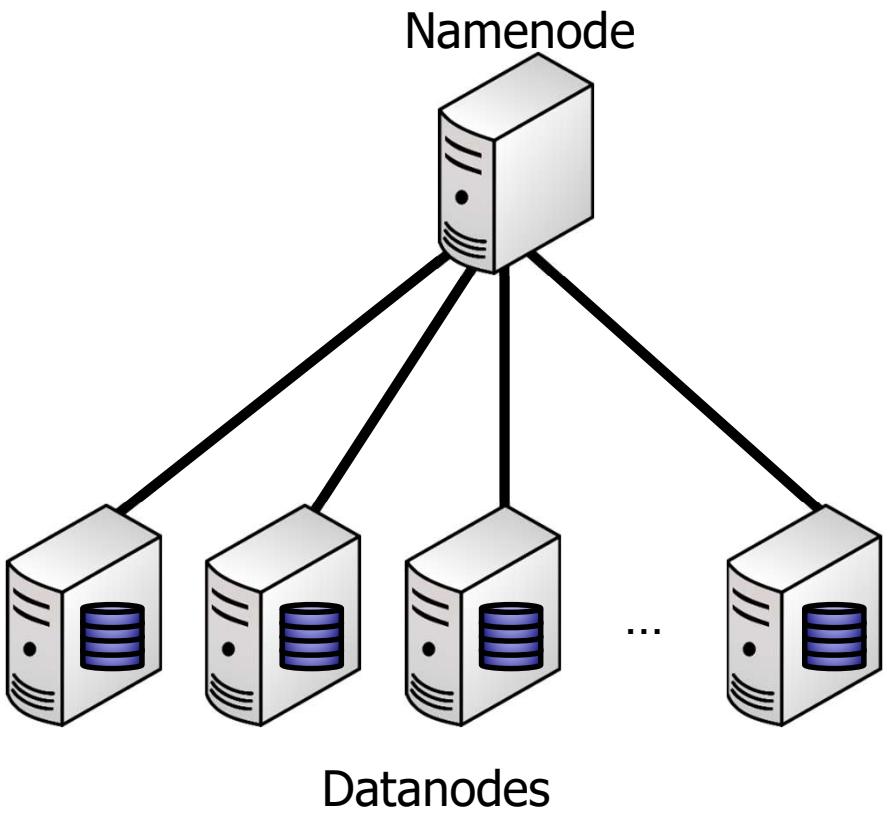
- However:
  - It relies on a set of auxiliary classes that are in another project (Ex12)
  - The application is configured using configuration files



Everything in action

- The configuration files and the auxiliary classes must be available in all the cluster nodes

- We are going to distribute these files using the distributed cache mechanism that is shipped with Hadoop



Everything in action

- Every MapReduce application that is a sub class of Configure and Tool accepts the following configurations:
- **-archives <comma separated list of archives>**
  - Archives to be unarchived on the compute machines. Applies only to job.
- **-conf <configuration file>**
  - Configuration file.
- **-D<property>=<value>**
  - Specify a property.
- **-files <comma separated list of files>**
  - Files to be copied to the map reduce cluster. Applies only to job.
- **-fs <file:///> or <hdfs://namenode:port>**
  - Default filesystem URL to use. Overrides 'fs.defaultFS' property from configurations.
- **-jt <local> or <resourcemanager:port>**
  - Specify a Resource Manager. Applies only to job.
- **-libjars <comma separated list of jars>**
  - jar files to include in the class path. Applies only to job.

Everything in action – Counters

- Another element that this application will use is the usage of counters
- Counters are a useful approach for gathering statistics about the job: for quality control or for application-level statistics
- Besides the built-in counters applications can define new counters

Everything in action – Counters

- Counters updated in the context of map or reduce task are aggregated when the job (application) ends

- Counters are defined by a Java enumerate that can be used to group related counters

Word count – Ver02


```
public class WordCountApplicationVer02
        extends Configured implements Tool {

  public static void main(String[] args) throws Exception {
    System.exit(ToolRunner.run(new WordCountApplicationVer02(),args));
  }

  @Override
  public int run(String[] args) throws Exception {
    if (args.length != 2) {
      System.err.printf("Usage: %s [opts] <in> <out>\n",getClass());
      ToolRunner.printGenericCommandUsage( System.err );
      return -1;
    }

    ...
```

Word count – Ver02

Ex11

```
// Create a job
Job job = Job.getInstance( getConf() );

// Set job name
job.setJobName( "Word Count - Version 02" );

// Set jar file
job.setJarByClass( WordCountApplicationVer02.class );

// Set map, reducer and combiner
job.setMapperClass( WordCountMapperVer02.class );
job.setCombinerClass( WordCountCombinerVer02.class );
job.setReducerClass( WordCountReducerVer02.class );

...
```

Word count – Ver02


```
    // Set output types of map and reduce functions
    job.setMapOutputKeyClass( Text.class );
    job.setMapOutputValueClass( IntWritable.class );

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

    // Set input path and class format
    MyFileInputFormat.addInputPath(job, new Path(args[0]) );
    job.setInputFormatClass( MyTextInputFormat.class );

    // Set output path
    FileOutputFormat.setOutputPath(job, new Path(args[1]) );

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

Running

- In this example we assume that:
  - The input and output is on the `HDFS` file system
  - The auxiliary classes (contained within a jar file) are available in the `HDFS` file system
  - The application can be configure using configuration files
  - Before the execution of the application the input and auxiliary classes are copied to the `HDFS` file system
  - Debug information is written to Hadoop logs using `log4java`

Word count – Ver02 Map


```
public class WordCountMapperVer02
        extends Mapper<Object, Text, Text, IntWritable> {

  private static final Log log =
        LogFactory.getLog( WordCountMapperVer02.class);

  private static final IntWritable one = new IntWritable(1);

  private Text word = new Text();

  private boolean caseSensitive;
  private Set<String> patternsToSkip = new HashSet<String>();

  ...
```

Initialize log

Word count – Ver02 Map

Ex11

```
@Override
public void setup(Context context)
      throws IOException, InterruptedException {
  Configuration conf = context.getConfiguration();
  caseSensitive=conf.getBoolean("wordcount.case.sensitive",true);
  boolean skipPatterns=conf.getBoolean("wordcount.skip.patterns", false);
  if ( skipPatterns==true ) {
    for (URI patternsURI : Job.getInstance( conf ).getCacheFiles()) {
      Path patternsPath = new Path( patternsURI.getPath() );
      String patternsFileName = patternsPath.getName().toString();
      if ( patternsFileName.endsWith( ".txt" ) ) {
        WordCountUtils.parseSkipFile( patternsToSkip, patternsFileName);
      }
    }
  }
}
...
```

Word count – Ver02 Map

Ex11

```
...
@Override
public void map(Object key, Text value, Context context)
      throws IOException, InterruptedException {
  String line;
  line =(caseSensitive)?value.toString(): value.toString().toLowerCase();
  for (String pattern : patternsToSkip ) {
    line = line.replaceAll(pattern, "" );
  }
  StringTokenizer itr = new StringTokenizer( line );
  while ( itr.hasMoreTokens() ) {
    this.word.set( itr.nextToken() );
    context.write( this.word, WordCountMapperVer02.one);
    context.getCounter(WordCountUtils.Statistics.TotalWords).increment(1);
  }
}
...
```

Update counter that represents the total number of words

Word count – Ver02 Map


```
  ...
  @Override
  public void cleanup(Context context)
      throws IOException, InterruptedException {

    if ( log.isDebugEnabled() ) {
      String msg;
      msg = "WordCountMapperVer02#cleanup(Context) called";

      System.out.println( msg );
      log.debug( msg );
    }

    super.cleanup(context);
  }
}
```

stdout log

syslog log

Word count – Ver02 Reduce


```
public class WordCountReducerVer02
        extends Reducer<Text, IntWritable, Text, IntWritable> {

  private static final Log log =
    LogFactory.getLog( WordCountReducerVer02.class );

  private IntWritable result = new IntWritable();

  @Override
  public void setup(Context context)
        throws IOException, InterruptedException {

    if ( log.isDebugEnabled() ) {...}
    super.setup( context );
  }
  ...
```

Word count – Ver02 Reduce


```
  @Override
  public void reduce(Text key, Iterable<IntWritable> values, Context context)
        throws IOException, InterruptedException {
    int sum = 0;  for (IntWritable val : values) { sum += val.get(); }
    this.result.set( sum );
    context.write(key, this.result );
    context.getCounter( WordCountUtils.Statistics.Distincts ).increment( 1 );
    if ( sum==1 ) {
      context.getCounter( WordCountUtils.Statistics.Singletons).increment(1);
    }
  }
  @Override
  public void cleanup(Context context)
        throws IOException, InterruptedException {
    super.cleanup(context);
  }
}
```

```
Creating input directory in HDFS file system...
hadoop fs  -mkdir -p hdfs:///user/usermr/examples/input/gutenberg/small

Resuming in 1 seconds

Copying input files to HDFS file system...
hadoop fs  -cp -f file:///home/usermr/examples/input/gutenberg/small/*.* hdfs:///user/usermr/examples/input/gutenberg/small

Resuming in 1 seconds

Removing previous output...
hadoop fs  -rm -f -r hdfs:///user/usermr/examples/output/gutenberg/small
Deleted hdfs:///user/usermr/examples/output/gutenberg/small

Resuming in 1 seconds

Exporting classpath...
export HADOOP_CLASSPATH=/home/usermr/examples/Projects/02-WordCount/Ex11-WordCount-02/../Ex12-WordCount-02-Lib/target/Ex12-WordCount-0
2-Lib-2020.2021.SemInv.jar

Press ENTER to run the example...


Running...
hadoop jar /home/usermr/examples/Projects/02-WordCount/Ex11-WordCount-02/target/Ex11-WordCount-02-2020.2021.SemInv.jar -libjars file:/
//home/usermr/examples/Projects/02-WordCount/Ex11-WordCount-02/../Ex12-WordCount-02-Lib/target/Ex12-WordCount-02-Lib-2020.2021.SemInv.
jar  -files patterns.txt,file:///home/usermr/examples/Projects/02-WordCount/Ex11-WordCount-02/../Ex12-WordCount-02-Lib/target/Ex12-Wor
dCount-02-Lib-2020.2021.SemInv.jar hdfs:///user/usermr/examples/input/gutenberg/small hdfs:///user/usermr/examples/output/gutenberg/sm
all
```

```
        WRONG_REDUCE=0
    cdle.wordcount.mr.WordCountUtils$Statistics
            Distincts=15
            Singletons=13
            TotalWords=17
    File Input Format Counters
            Bytes Read=95
    File Output Format Counters
            Bytes Written=114
```

```
                WRONG_REDUCE=0
                                                  tistics
                Distincts=15
                Singletons=13
                TotalWords=17
                                                  Counters
                Bytes Read=95
        File Output Format Counters
                Bytes Written=114

Press ENTER to view the results...

hadoop fs -cat /user/usermr/examples/output/gutenberg/small/part-r-00000

Bye     1
Goodbye 1
Hadoop, 1
Hello   2
In      1
The     1
World   1
World!  1
World,  2
again.  1
blue.   1
hadoop. 1
is      1
the     1
to      1
usermr@hadoop: .../Ex11-WordCount-02$
```

confWordCount.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>

nfiguration>

<property>
    <name>mapreduce.map.log.level</name>
    <value>DEBUG</value>
</property>

<property>
    <name>mapreduce.reduce.log.level</name>
    <value>DEBUG</value>
</property>

<property>
    <name>wordcount.case.sensitive</name>
    <!-- <value>false</value> -->
    <value>true</value>
</property>

<property>
    <name>wordcount.skip.patterns</name>
    <!-- <value>true</value> -->
    <value>false</value>
</property>

</configuration>
```

JobHistory

hadoop.lrcd.e.ipl.pt:19888/jobhistory

JobHistory

- Application
  - About
  - Jobs
- Tools

Retired Jobs

Show 20 entries

Submit Time	Start Time	Finish Time	Job ID	Name	User	Queue	State	Maps Total	Maps Completed
2021.11.12 15:21:01 WET	2021.11.12 15:21:09 WET	2021.11.12 15:21:26 WET	job_1636652927199_0018	Word Count - Version 02	usermr	default	SUCCEEDED	2	2
2021.11.12 09:07:56 WET	2021.11.12 09:08:03 WET	2021.11.12 09:08:26 WET	job_1636652927199_0017	Word Count - By CJSG	usermr	default	SUCCEEDED	3	3
2021.11.11 23:11:38 WET	2021.11.11 23:11:45 WET	2021.11.11 23:11:59 WET	job_1636652927199_0016	Word Count - By CJSG	usermr	default	SUCCEEDED	1	1
2021.11.11 23:07:37 WET	2021.11.11 23:07:44 WET	2021.11.11 23:07:57 WET	job_1636652927199_0015	Word Count - By CJSG	usermr	default	SUCCEEDED	2	2

hadoop.lrcd.e.ipl.pt:19888/jobhistory/job/job_1636652927199_0018

MapReduce Job job_1636652927199

hadoop.lrcd.e.ipl.pt:19888/jobhistory/job/job_1636652927199_0018

Job

- Overview
- Counters
- Configuration
- Map tasks
- Reduce tasks

Tools

User Name:	usermr
Queue:	default
State:	SUCCEEDED
Uberized:	false
Submitted:	Fri Nov 12 15:21:01 WET 2021
Started:	Fri Nov 12 15:21:09 WET 2021
Finished:	Fri Nov 12 15:21:26 WET 2021
Elapsed:	17sec
Diagnostics:	
Average Map Time	8sec
Average Shuffle Time	3sec
Average Merge Time	0sec
Average Reduce Time	0sec

ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Fri Nov 12 15:21:03 WET 2021	mrnode01.lrcd.e.ipl.pt:8042	logs

Task Type	Total	Complete
Map	2	2
Reduce	1	1

Attempt Type	Failed	Killed	Successful
Maps	0	0	2
Reduces	0	0	1

hadoop.lrcd.e.ipl.pt:19888/jobhistory/tasks/job_1636652927199_0018/m

Map Tasks for job_1636652927199_(

hadoop.lrcd.e.ipl.pt:19888/jobhistory/tasks/job_1636652927199_0018/m

Logged in as: dr.who

Map Tasks for job_1636652927199_0018

- Application
- Job
  - Overview
  - Counters
  - Configuration
  - Map tasks
  - Reduce tasks
- Tools

Show 20 entries Search:

Task					Successful Attempt		
Name	State	Start Time	Finish Time	Elapsed Time	Start Time	Finish Time	Elapsed Time
task_1636652927199_0018_m_000000	SUCCEEDED	Fri Nov 12 15:21:11 +0000 2021	Fri Nov 12 15:21:19 +0000 2021	8sec	Fri Nov 12 15:21:11 +0000 2021	Fri Nov 12 15:21:19 +0000 2021	8sec
task_1636652927199_0018_m_000001	SUCCEEDED	Fri Nov 12 15:21:11	Fri Nov 12 15:21:19	8sec	Fri Nov 12 15:21:11	Fri Nov 12 15:21:19	8sec

- Application
- Job
- Task
  - Task Overview
  - Counters
- Tools

Show 20 entries Search:

Attempt	State	Status	Node	Logs	Start Time	Finish Time	Elaps Tim
attempt_1636652927199_0018_m_000000_0	SUCCEEDED	map	/default-rack/mrnode03.lrcd.e.ipl.pt:8042	logs	Fri Nov 12 15:21:11 +0000 2021	Fri Nov 12 15:21:19 +0000 2021	8sec
Attempt	State	Status	Node	Logs	Start T	Finish T	Elaps

Showing 1 to 1 of 1 entries First Previous 1

```
For class org.apache.hadoop.mapreduce.Mapper log level DEBUG is active
For class org.apache.hadoop.mapreduce.Mapper log level ERROR is active
For class org.apache.hadoop.mapreduce.Mapper log level FATAL is active
For class org.apache.hadoop.mapreduce.Mapper log level INFO is active
For class org.apache.hadoop.mapreduce.Mapper log level WARN is active
For class cdle.wordcount.mr.WordCountMapperVer02 log level DEBUG is active
For class cdle.wordcount.mr.WordCountMapperVer02 log level ERROR is active
For class cdle.wordcount.mr.WordCountMapperVer02 log level FATAL is active
For class cdle.wordcount.mr.WordCountMapperVer02 log level INFO is active
For class cdle.wordcount.mr.WordCountMapperVer02 log level WARN is active
For class cdle.wordcount.mr.formaters.input.MyFileInputFormat log level DEBUG is active
For class cdle.wordcount.mr.formaters.input.MyFileInputFormat log level ERROR is active
For class cdle.wordcount.mr.formaters.input.MyFileInputFormat log level FATAL is active
For class cdle.wordcount.mr.formaters.input.MyFileInputFormat log level INFO is active
For class cdle.wordcount.mr.formaters.input.MyFileInputFormat log level WARN is active
MyFileInputFormat<K, V>#ctor
For class cdle.wordcount.mr.formaters.input.MyTextInputFormat log level DEBUG is active
For class cdle.wordcount.mr.formaters.input.MyTextInputFormat log level ERROR is active
For class cdle.wordcount.mr.formaters.input.MyTextInputFormat log level FATAL is active
```

Logs for container_16366

container_1636652927199_0018_01_000002

ResourceManager

RM Home

NodeManager

Tools

Local Logs:

directory.info : Total file length is 2661 bytes.
launch_container.sh : Total file length is 5940 bytes.
prelaunch.err : Total file length is 0 bytes.
prelaunch.out : Total file length is 100 bytes.
stderr : Total file length is 285 bytes.
stdout : Total file length is 2512 bytes.
syslog : Total file length is 294030 bytes.

```
log4j:WARN No appenders could be found for logger (org.apache.hadoop.fs.FileSystem).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
```

Logs for container_1636652927199_

mrnode03.lrcd.e.ipl.pt:8042/node/containerlogs/co

```
2021-11-12 15:21:17,201 DEBUG [main] org.apache.hadoop.ipc.Server: rpcKind=RPC_PROTOCOL_BUFFER, rpcRequestWrapperClass=class c
2021-11-12 15:21:17,211 DEBUG [main] org.apache.hadoop.ipc.Client: getting client out of cache: Client-c6ef4fdae6084bdabe8bc8e
2021-11-12 15:21:18,254 DEBUG [client DomainSocketWatcher] org.apache.hadoop.net.unix.DomainSocketWatcher: org.apache.hadoop.n
2021-11-12 15:21:18,267 DEBUG [main] org.apache.hadoop.util.PerformanceAdvisory: Both short-circuit local reads and UNIX domai
2021-11-12 15:21:18,280 DEBUG [main] org.apache.hadoop.hdfs.protocol.datatransfer.sasl.DataTransferSaslUtil: DataTransferProtc
2021-11-12 15:21:18,289 DEBUG [main] org.apache.hadoop.fs.FileSystem: Creating FS hdfs://hadoop.lrcd.e.ipl.pt:8020: duration C
2021-11-12 15:21:18,300 DEBUG [main] org.apache.hadoop.mapred.Task: using new api for output committer
2021-11-12 15:21:18,302 DEBUG [main] org.apache.hadoop.mapreduce.lib.output.PathOutputCommitterFactory: Looking for committer
2021-11-12 15:21:18,302 DEBUG [main] org.apache.hadoop.mapreduce.lib.output.PathOutputCommitterFactory: No scheme-specific fac
2021-11-12 15:21:18,302 DEBUG [main] org.apache.hadoop.mapreduce.lib.output.PathOutputCommitterFactory: No output committer fa
2021-11-12 15:21:18,330 DEBUG [main] org.apache.hadoop.mapreduce.lib.output.PathOutputCommitterFactory: Creating FileOutputCom
2021-11-12 15:21:18,331 DEBUG [main] org.apache.hadoop.mapreduce.lib.output.PathOutputCommitter: Instantiating committer FileC
2021-11-12 15:21:18,331 INFO [main] org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter: File Output Committer Algorith
2021-11-12 15:21:18,331 INFO [main] org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter: FileOutputCommitter skip clear
2021-11-12 15:21:18,412 INFO [main] org.apache.hadoop.mapred.Task:  Using ResourceCalculatorProcessTree : [ ]
2021-11-12 15:21:18,522 DEBUG [main] org.apache.hadoop.yarn.util.ProcfsBasedProcessTree: [ 7822 7849 ]
2021-11-12 15:21:18,553 DEBUG [main] cdle.wordcount.mr.formaters.input.MyFileInputFormat: MyFileInputFormat<K, V>#ctor
2021-11-12 15:21:18,555 DEBUG [main] cdle.wordcount.mr.formaters.input.MyTextInputFormat: MyTextInputFormat#ctor
2021-11-12 15:21:18,578 DEBUG [main] org.apache.hadoop.ipc.Client: The ping interval is 60000 ms.
2021-11-12 15:21:18,578 DEBUG [main] org.apache.hadoop.ipc.Client: Connecting to hadoop.lrcd.e.ipl.pt/10.62.73.181:8020
2021-11-12 15:21:18,578 DEBUG [main] org.apache.hadoop.ipc.Client: Setup connection to hadoop.lrcd.e.ipl.pt/10.62.73.181:8020
2021-11-12 15:21:18,591 DEBUG [main] org.apache.hadoop.security.UserGroupInformation: PrivilegedAction [as: usermr (auth:SIMPI
java.lang.Exception
        at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1875)
        at org.apache.hadoop.ipc.Client$Connection.setupIOstreams(Client.java:828)
        at org.apache.hadoop.ipc.Client$Connection.access$3800(Client.java:414)
        at org.apache.hadoop.ipc.Client.getConnection(Client.java:1647)
        at org.apache.hadoop.ipc.Client.call(Client.java:1463)
        at org.apache.hadoop.ipc.Client.call(Client.java:1416)
        at org.apache.hadoop.ipc.ProtobufRpcEngine2$Invoker.invoke(ProtobufRpcEngine2.java:242)
        at org.apache.hadoop.ipc.ProtobufRpcEngine2$Invoker.invoke(ProtobufRpcEngine2.java:129)
        at com.sun.proxy.$Proxy13.getBlockLocations(Unknown Source)
```

cdle Highlight All Match Case Match Diacritics Whole Words 1 of 10 matches

usermr@mrnode03: ~

```
usermr@mrnode03:~$ cat /tmp/dbg.log
MyFileInputFormat<K, V>#ctor
MyTextInputFormat#ctor
MyFileInputFormat<K, V>#ctor
MyTextInputFormat#ctor
MyTextInputFormat#createRecordReader(hdfs://hadoop.lrcd.e.ipl.pt:8020/user/usermr/examples/input/gutenberg/small/file01.txt:0+63, Word Count - Version 02) -> org.apache.hadoop.mapreduce.lib.input.LineRecordReader@26fb628
MyTextInputFormat#createRecordReader(hdfs://hadoop.lrcd.e.ipl.pt:8020/user/usermr/examples/input/gutenberg/small/file02.txt:0+32, Word Count - Version 02) -> org.apache.hadoop.mapreduce.lib.input.LineRecordReader@26fb628
usermr@mrnode03:~$
```

Setting Hadoop container memory

Hadoop memory management

Memory management

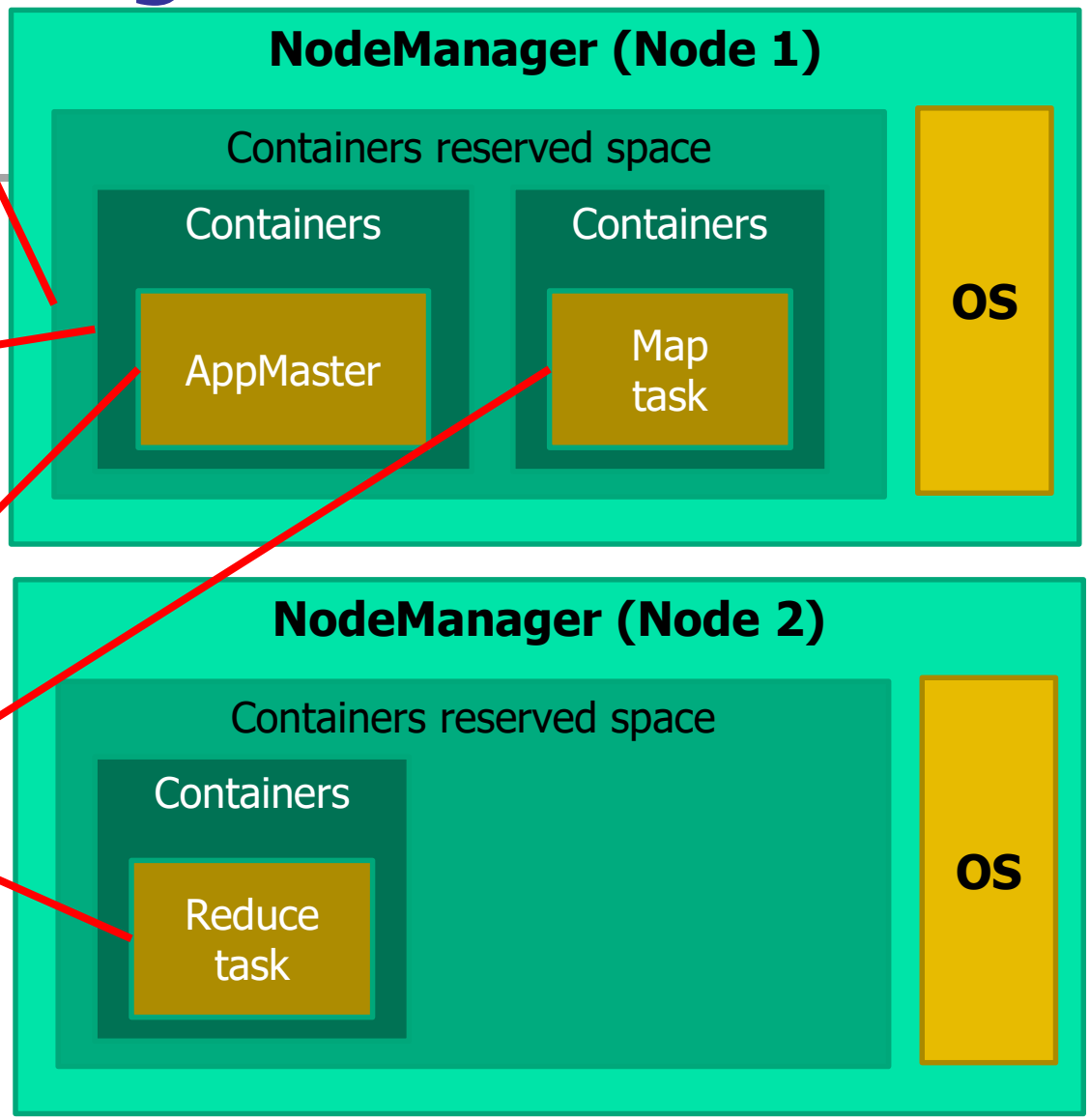
yarn.nodemanager.resource.memory-mb

yarn.scheduler.maximum-allocation-mb
yarn.scheduler.minimum-allocation-mb

yarn.app.mapreduce.am.resource.memory-mb

mapreduce.map.resource.memory-mb

mapreduce.reduce.resource.memory-mb



Adapted from [3]

Memory management
/etc/hadoop/yarn-site.xml

```
<configuration>
        ...
        <property>
                <name>yarn.nodemanager.resource.memory-mb</name>
                <value>...</value>
        </property>

        <property>
                <name>yarn.scheduler.maximum-allocation-mb</name>
                <value>...</value>
        </property>

        <property>
                <name>yarn.scheduler.minimum-allocation-mb</name>
                <value>...</value>
        </property>
</configuration
```

Memory management

/etc/hadoop/mapred-site.xml

```
<configuration>
        ...
        <property>
                <name>yarn.app.mapreduce.am.resource.memory-mb</name>
                <value>...</value>
        </property>

        <property>
                <name>mapreduce.map.resource.memory-mb</name>
                <value>...</value>
        </property>

        <property>
                <name>mapreduce.reduce.resource.memory-mb</name>
                <value>...</value>
        </property>
</configuration
```

Memory management

Property	Value – 2 Gbyte	Value – 8 Gbyte
yarn.nodemanager.resource.memory-mb	1356	8192
yarn.scheduler.maximum-allocation-mb	1536	8192
yarn.scheduler.minimum-allocation-mb	128	1024
yarn.app.mapreduce.am.resource.memory-mb	512	1536
mapreduce.map.resource.memory-mb	256	1024
mapreduce.reduce.resource.memory-mb	256	1024

The default values assume a node with 8 Gbyte of memory

Setting Hadoop local file system permissions

Hadoop file system permissions

Memory management

/etc/hadoop/hdfs-site.xml

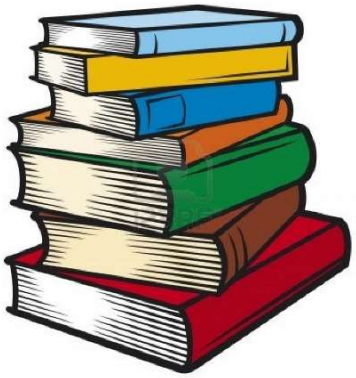
```
<configuration>
        ...
        <property>
                <name>fs.permissions.umask-mode</name>
                <value>002</value>
        </property>
</configuration
```

Job management

Commands to manage submitted Jobs

- Get list of running Hadoop (MapReduce) jobs
  - `mapred job -list`
- Get list of running Yarn applications
  - `yarn application -list`
- Terminate (kill) a Hadoop (MapReduce) job
  - `mapred job -kill` **<jobId>**
- Terminate (kill) a Yarn application
  - `yarn application -kill` **<ApplicationId>**

References



[1] T. White, "Hadoop - The Definitive Guide" 4th Edition", ISBN-13: 9781491901632, ISBN-10: 1491901632

[2] Apache Hadoop, http://hadoop.apache.org/

[3] How to Install and Set Up a 3-Node Hadoop Cluster https://www.linode.com/docs/guides/how-to-install-and-set-up-hadoop-cluster/