MapReduce

Dr. Yashar Moshfeghi

Big Data 4

Tutorial, Week 2

*Part of the slides are taken from Dr. Nikos Ntarmos

Outline

- JobTracker UI
  - General information
  - Job information
  - Task information
- Hadoop programming
  - Hadoop data types
  - Basic Mapper/Reducer methods
  - Custom input/output formats
  - Word Count galore

JobTracker UI :: General information

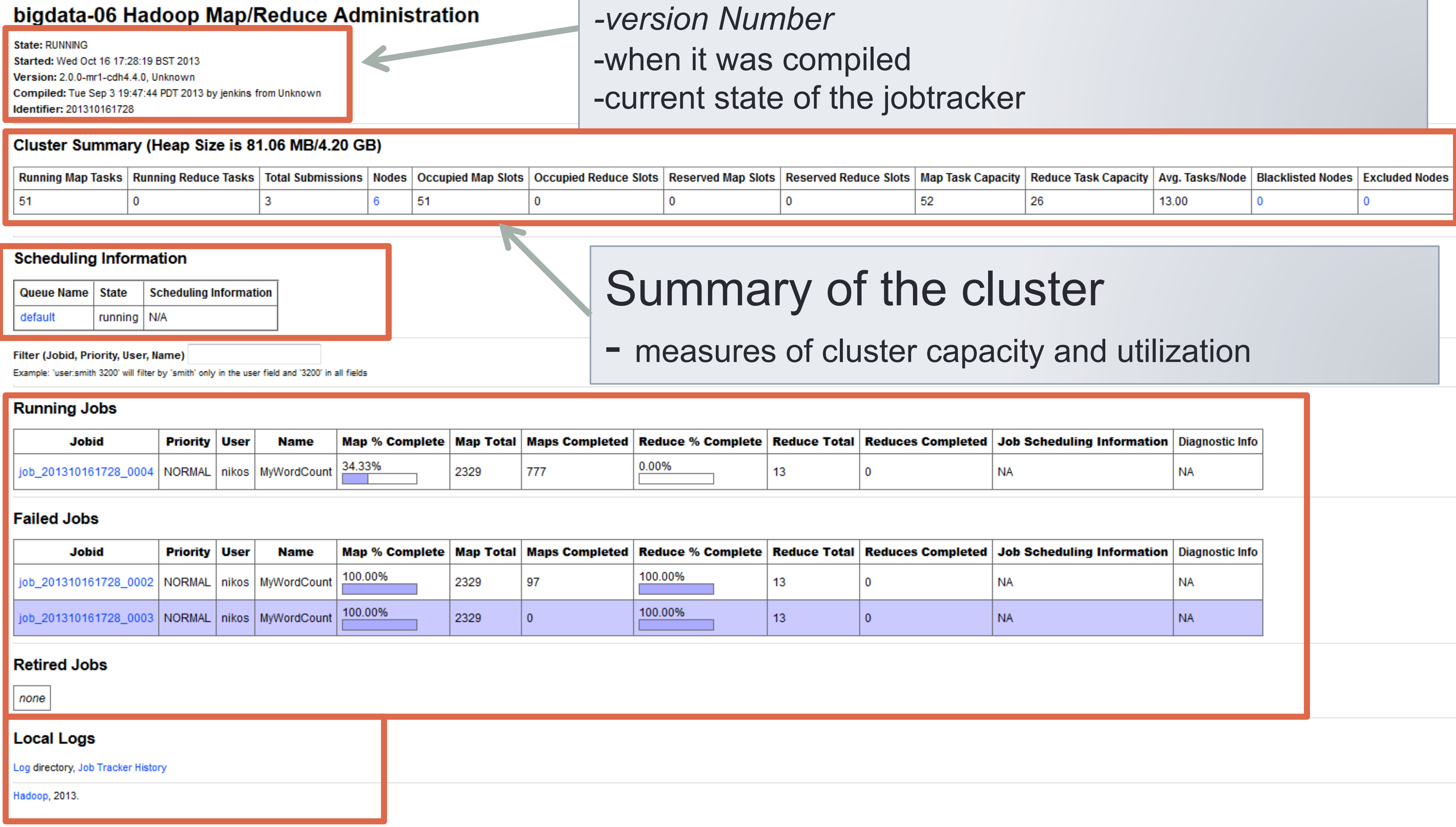
The Jobtracker P

Details of Hadoop Installation

-version Number

-when it was compiled

-current state of the jobtracker



bigdata-06 Hadoop Map/Reduce Administration

State: RUNNING
Started: Wed Oct 16 17:28:19 BST 2013
Version: 2.0.0-mr1-cdh4.4.0, Unknown
Compiled: Tue Sep 3 19:47:44 PDT 2013 by jenkins from Unknown
Identifier: 201310161728

Cluster Summary (Heap Size is 81.06 MB/4.20 GB)

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
51	0	3	6	51	0	0	0	52	26	13.00	0	0

Summary of the cluster

- measures of cluster capacity and utilization

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_201310161728_0004	NORMAL	nikos	MyWordCount	34.33%	2329	777	0.00%	13	0	NA	NA

Failed Jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information	Diagnostic Info
job_201310161728_0002	NORMAL	nikos	MyWordCount	100.00%	2329	97	100.00%	13	0	NA	NA
job_201310161728_0003	NORMAL	nikos	MyWordCount	100.00%	2329	0	100.00%	13	0	NA	NA

Retired Jobs

none

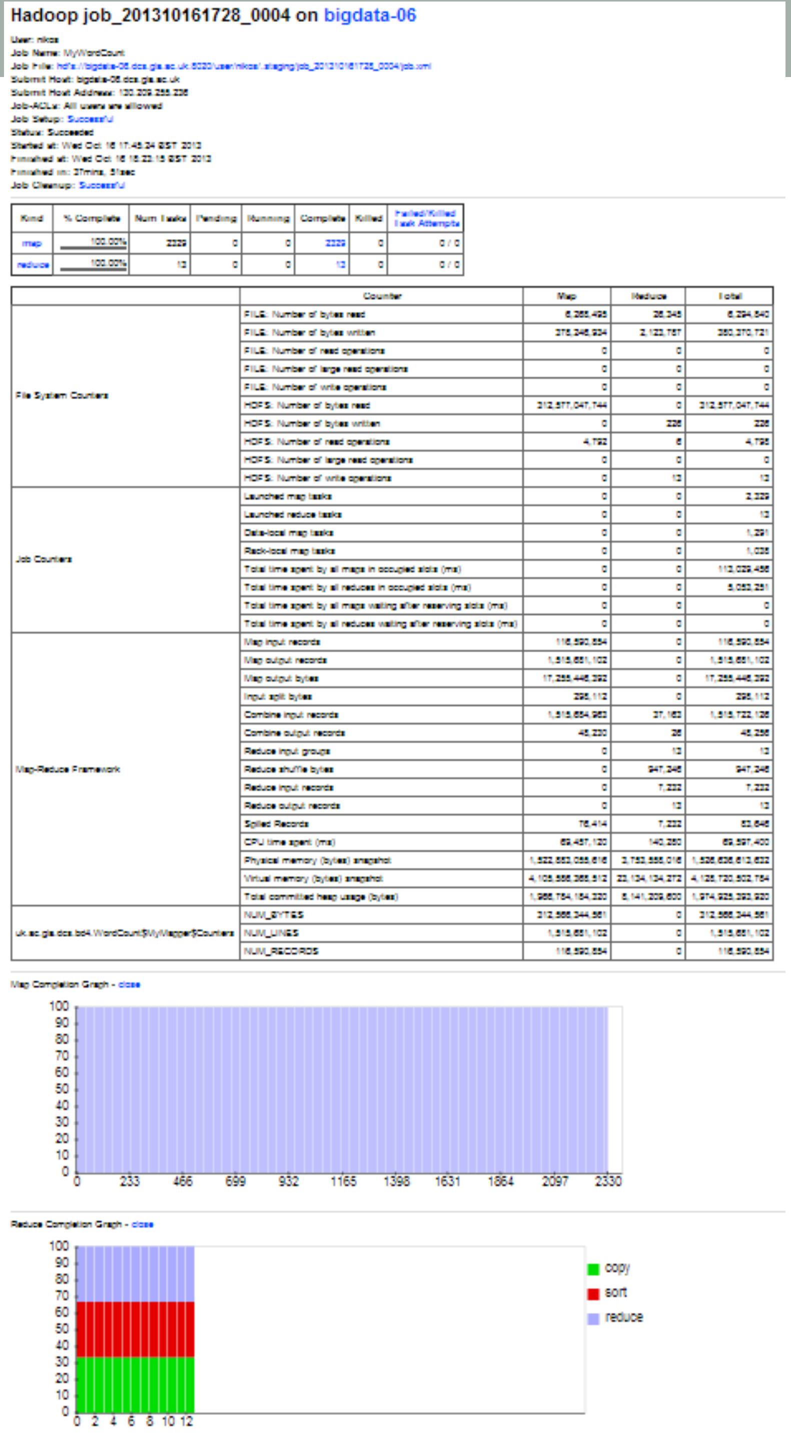
Local Logs

Log directory, Job Tracker History

Hadoop, 2013.

JobTracker UI :: Job information

The Job Page



Hadoop job_201310161728_0004 on bigdata-06

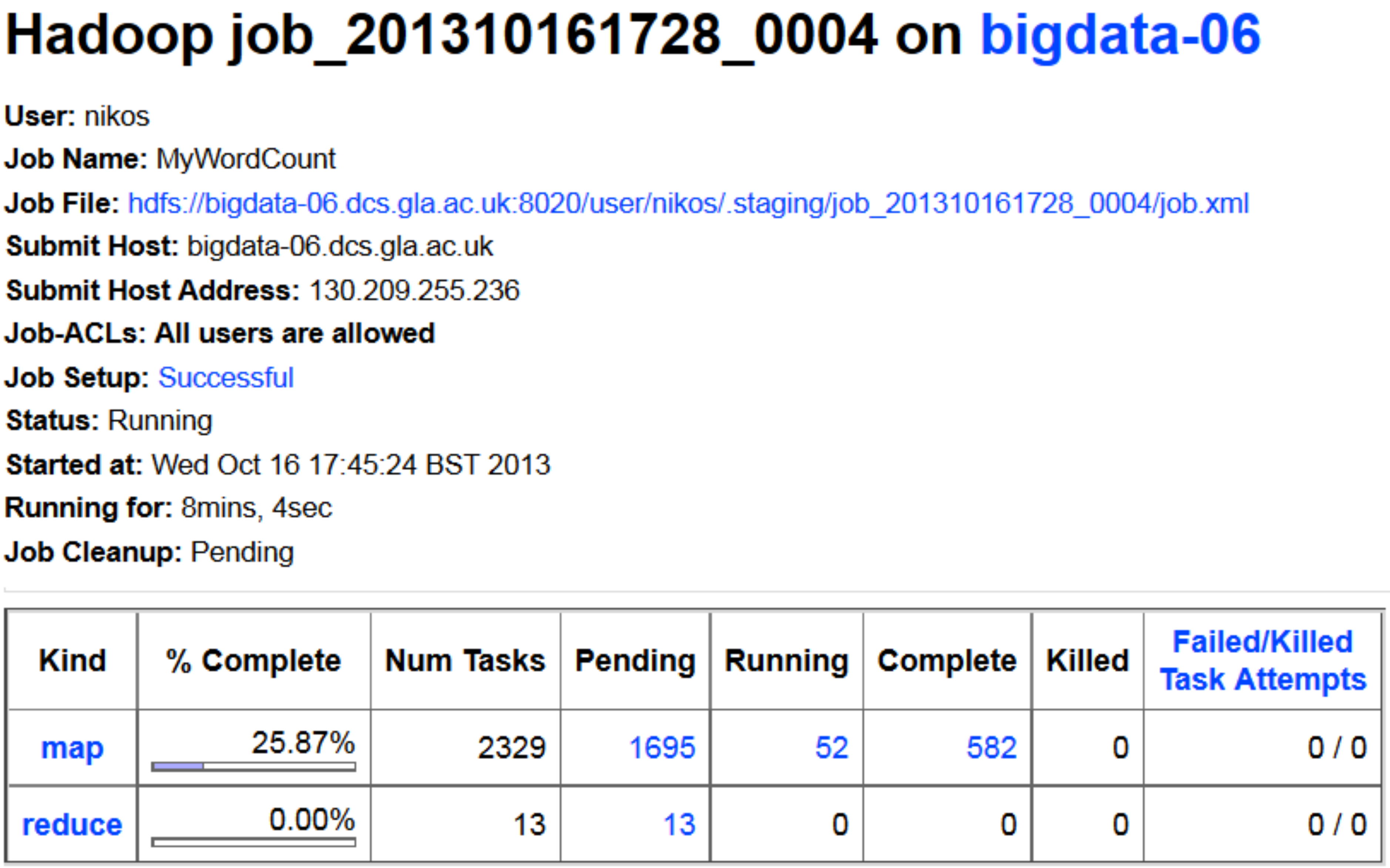
User: nikos
Job Name: MyWordCount
Job File: hdfs://bigdata-06.dcs.gla.ac.uk:8020/user/nikos/.staging/job_201310161728_0004/job.xml
Submit Host: bigdata-06.dcs.gla.ac.uk
Submit Host Address: 130.209.255.236
Job-ACLs: All users are allowed
Job Setup: Successful
Status: Succeeded
Started at: Wed Oct 16 17:45:24 BST 2013
Finished at: Wed Oct 16 18:23:15 BST 2013
Finished in: 37mins, 51sec
Job Cleanup: Successful

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	2329	0	0	2329	0	0 / 0
reduce	100.00%	13	0	0	13	0	0 / 0

	Counter	Map	Reduce	Total
File System Counters	FILE: Number of bytes read	6,268,495	26,345	6,294,840
	FILE: Number of bytes written	378,246,934	2,123,787	380,370,721
	FILE: Number of read operations	0	0	0
	FILE: Number of large read operations	0	0	0
	FILE: Number of write operations	0	0	0
	HDFS: Number of bytes read	312,577,047,744	0	312,577,047,744
	HDFS: Number of bytes written	0	226	226
	HDFS: Number of read operations	4,792	6	4,798
	HDFS: Number of large read operations	0	0	0
	HDFS: Number of write operations	0	13	13
Job Counters	Launched map tasks	0	0	2,329
	Launched reduce tasks	0	0	13
	Data-local map tasks	0	0	1,291
	Rack-local map tasks	0	0	1,038
	Total time spent by all maps in occupied slots (ms)	0	0	112,029,456
	Total time spent by all reduces in occupied slots (ms)	0	0	5,053,251
	Total time spent by all maps waiting after reserving slots (ms)	0	0	0
	Total time spent by all reduces waiting after reserving slots (ms)	0	0	0
Map-Reduce Framework	Map input records	116,590,554	0	116,590,554
	Map output records	1,515,651,102	0	1,515,651,102
	Map output bytes	17,255,446,392	0	17,255,446,392
	Input split bytes	295,112	0	295,112
	Combine input records	1,515,684,963	37,163	1,515,722,126
	Combine output records	45,230	26	45,256
	Reduce input groups	0	13	13
	Reduce shuffle bytes	0	947,246	947,246
	Reduce input records	0	7,232	7,232
	Reduce output records	0	13	13
	Spilled Records	76,414	7,232	83,646
	CPU time spent (ms)	69,457,120	140,280	69,597,400
	Physical memory (bytes) snapshot	1,522,883,055,616	3,753,558,016	1,526,636,613,632
	Virtual memory (bytes) snapshot	4,105,586,368,512	23,134,134,272	4,128,720,502,784
	Total committed heap usage (bytes)	1,968,784,184,320	6,141,209,600	1,974,925,393,920
uk.ac.gla.dcs.bd4.WordCount$MyMapper$Counters	NUM_BYTES	312,566,344,561	0	312,566,344,561
	NUM_LINES	1,515,651,102	0	1,515,651,102
	NUM_RECORDS	116,590,554	0	116,590,554

Map Completion Graph - close

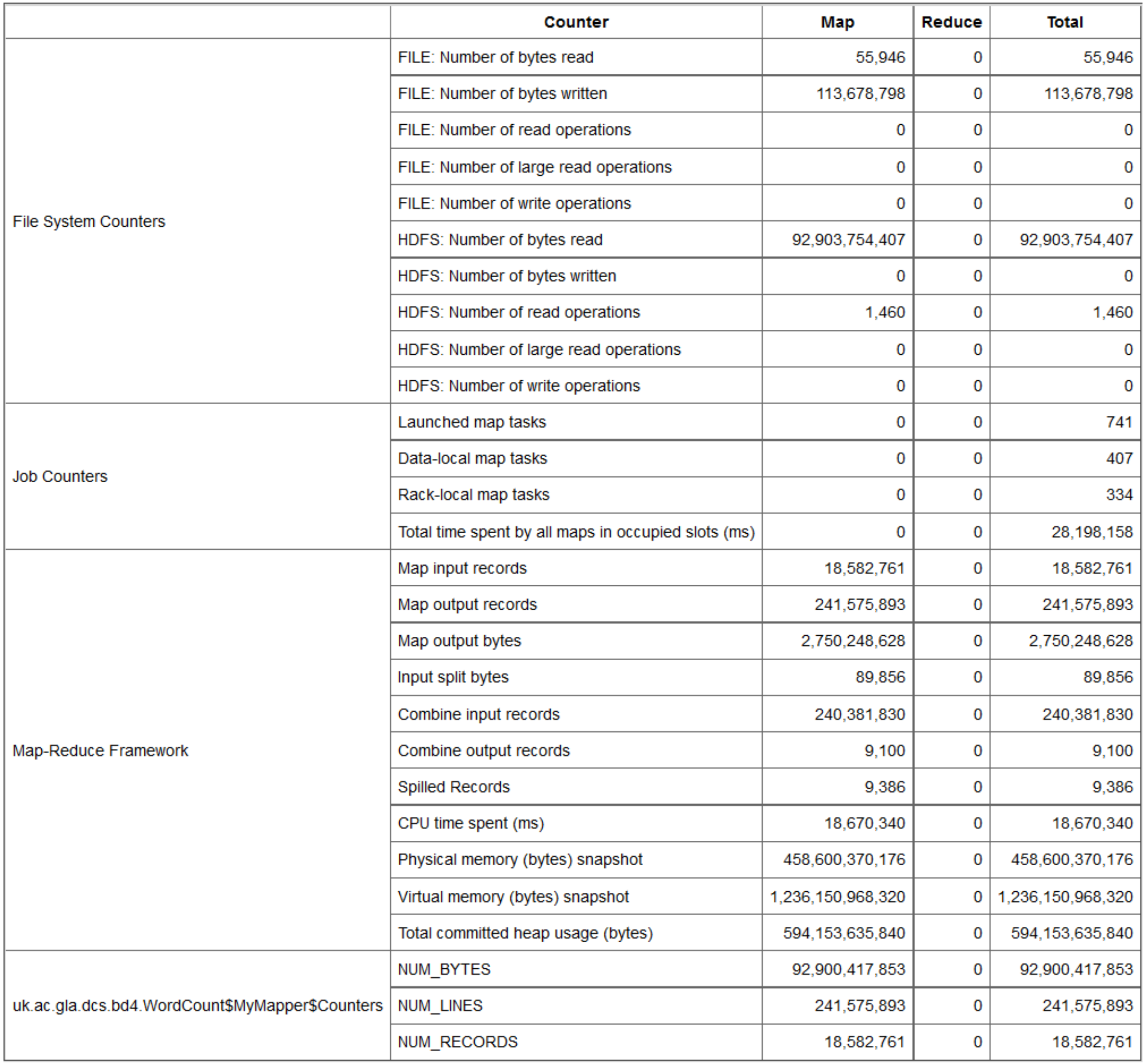
Reduce Completion Graph - close



Hadoop job_201310161728_0004 on bigdata-06

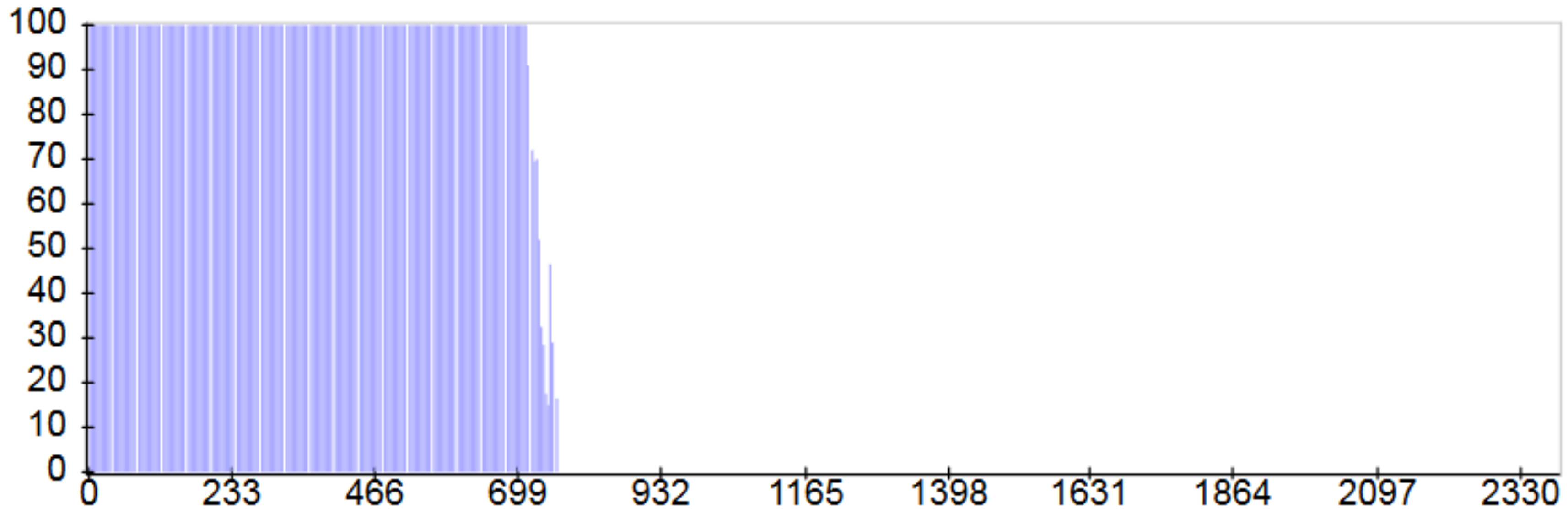
User: nikos
Job Name: MyWordCount
Job File: hdfs://bigdata-06.dcs.gla.ac.uk:8020/user/nikos/.staging/job_201310161728_0004/job.xml
Submit Host: bigdata-06.dcs.gla.ac.uk
Submit Host Address: 130.209.255.236
Job-ACLs: All users are allowed
Job Setup: Successful
Status: Running
Started at: Wed Oct 16 17:45:24 BST 2013
Running for: 8mins, 4sec
Job Cleanup: Pending

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	25.87%	2329	1695	52	582	0	0 / 0
reduce	0.00%	13	13	0	0	0	0 / 0



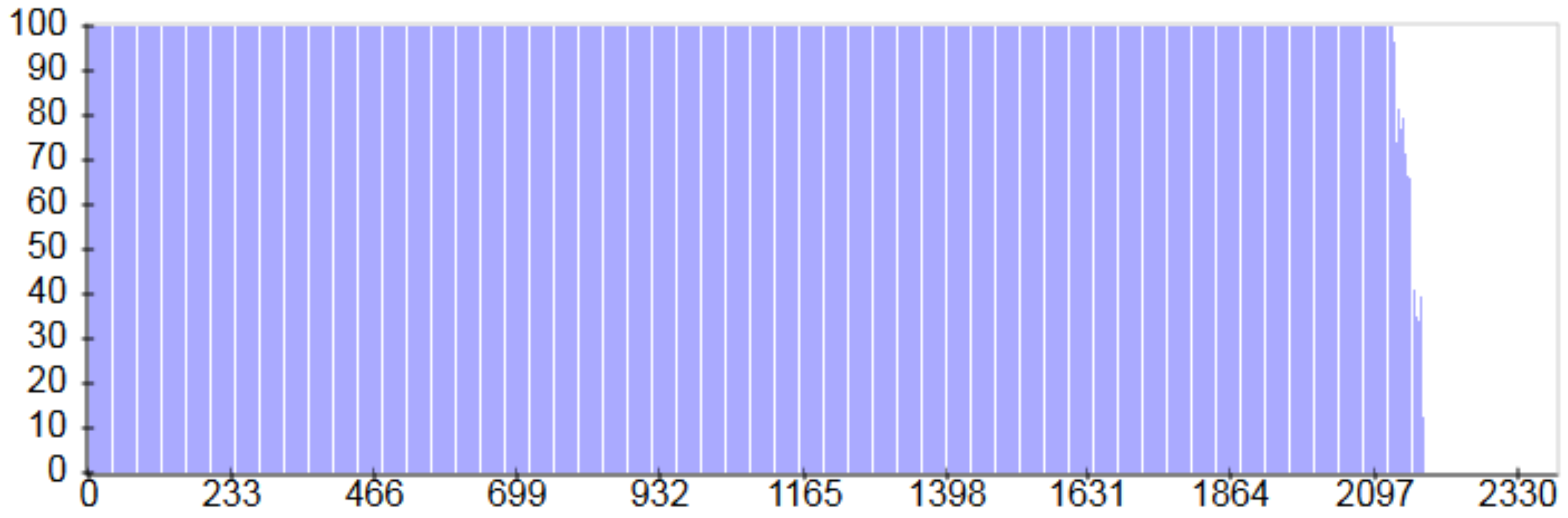
	Counter	Map	Reduce	Total
File System Counters	FILE: Number of bytes read	55,946	0	55,946
	FILE: Number of bytes written	113,678,798	0	113,678,798
	FILE: Number of read operations	0	0	0
	FILE: Number of large read operations	0	0	0
	FILE: Number of write operations	0	0	0
	HDFS: Number of bytes read	92,903,754,407	0	92,903,754,407
	HDFS: Number of bytes written	0	0	0
	HDFS: Number of read operations	1,460	0	1,460
	HDFS: Number of large read operations	0	0	0
	HDFS: Number of write operations	0	0	0
Job Counters	Launched map tasks	0	0	741
	Data-local map tasks	0	0	407
	Rack-local map tasks	0	0	334
	Total time spent by all maps in occupied slots (ms)	0	0	28,198,158
Map-Reduce Framework	Map input records	18,582,761	0	18,582,761
	Map output records	241,575,893	0	241,575,893
	Map output bytes	2,750,248,628	0	2,750,248,628
	Input split bytes	89,856	0	89,856
	Combine input records	240,381,830	0	240,381,830
	Combine output records	9,100	0	9,100
	Spilled Records	9,386	0	9,386
	CPU time spent (ms)	18,670,340	0	18,670,340
	Physical memory (bytes) snapshot	458,600,370,176	0	458,600,370,176
	Virtual memory (bytes) snapshot	1,236,150,968,320	0	1,236,150,968,320
	Total committed heap usage (bytes)	594,153,635,840	0	594,153,635,840
uk.ac.gla.dcs.bd4.WordCount$MyMapper$Counters	NUM_BYTES	92,900,417,853	0	92,900,417,853
	NUM_LINES	241,575,893	0	241,575,893
	NUM_RECORDS	18,582,761	0	18,582,761

Map Completion Graph - close

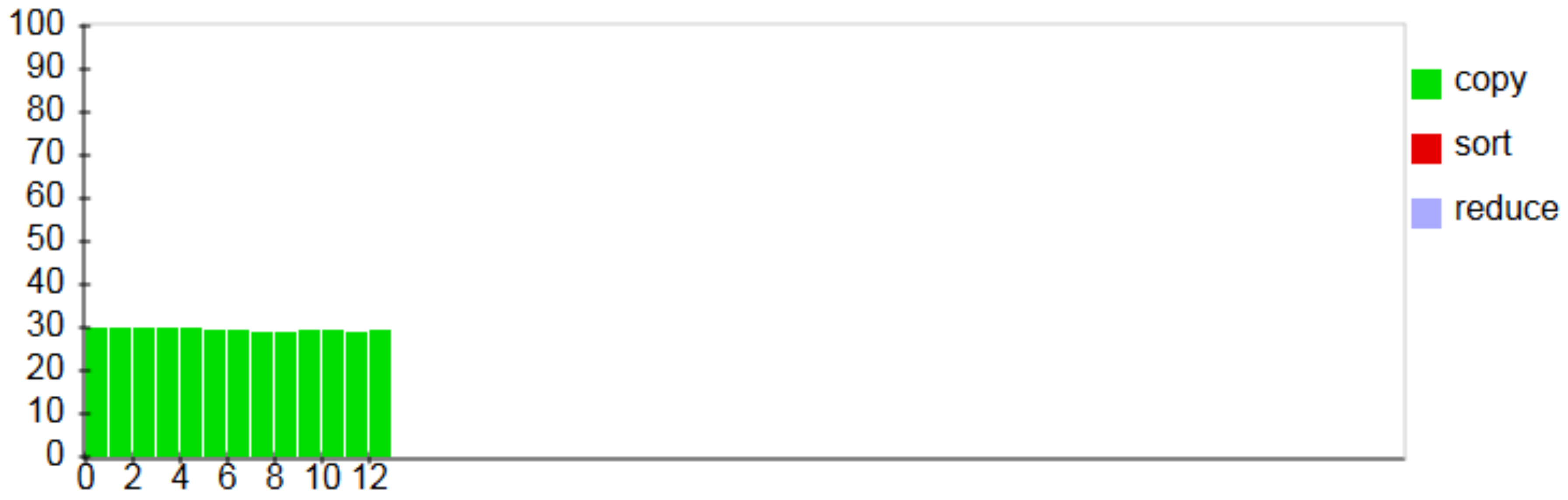


Reduce Completion Graph - close

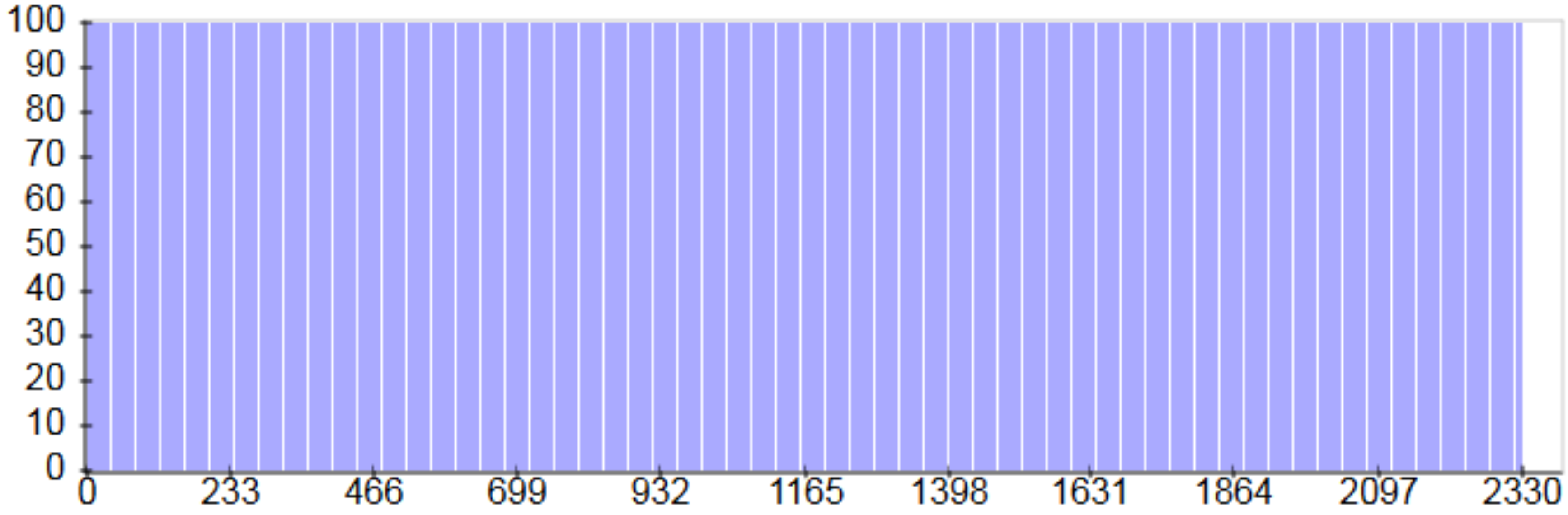
Map Completion Graph - close



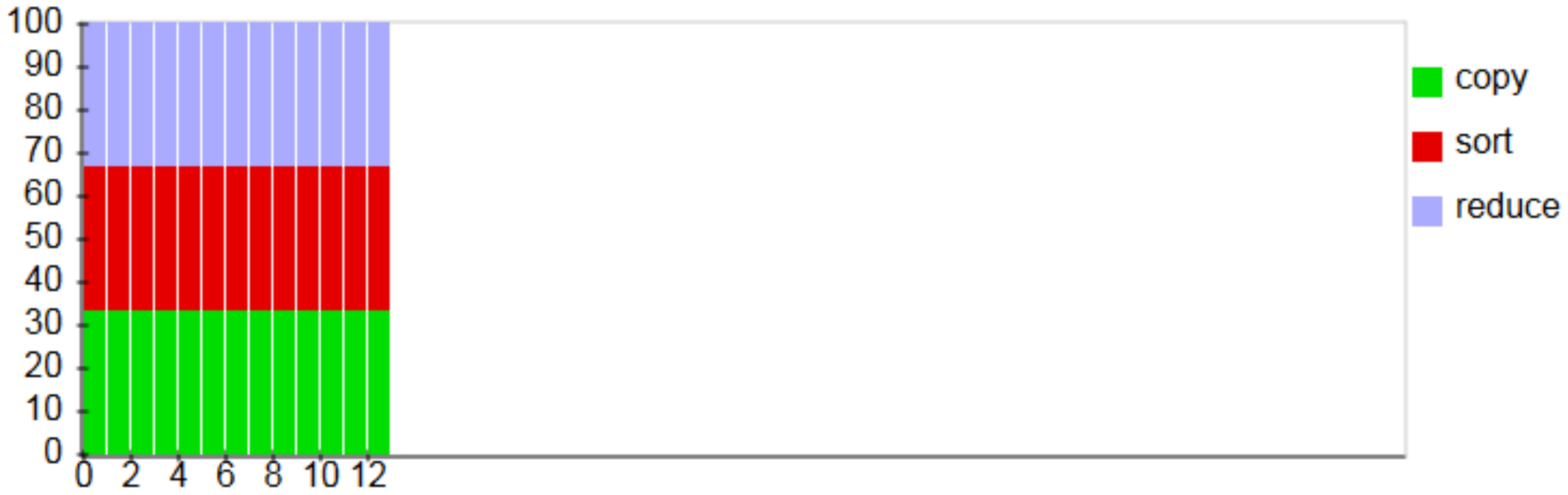
Reduce Completion Graph - close



Map Completion Graph - close

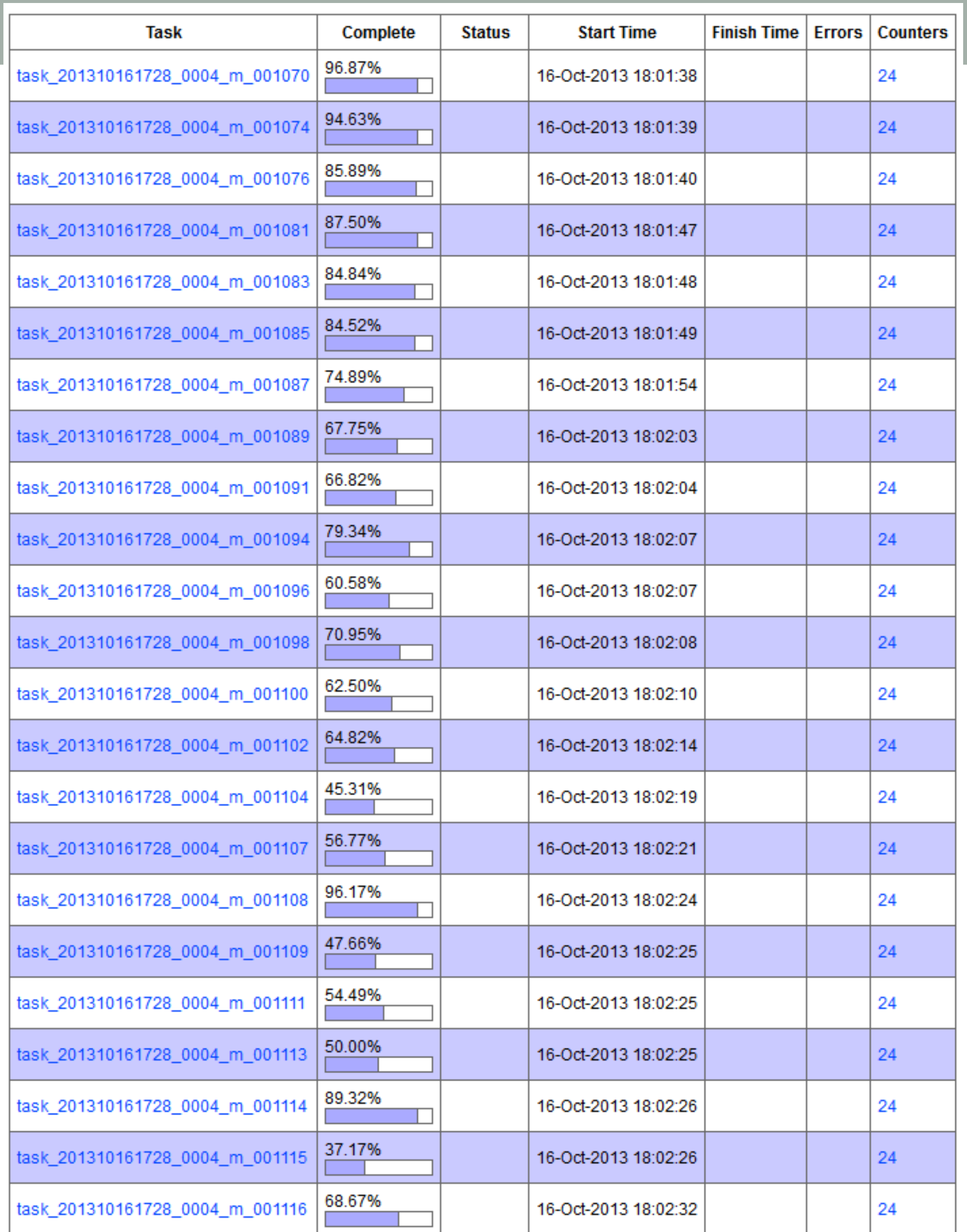


Reduce Completion Graph - close



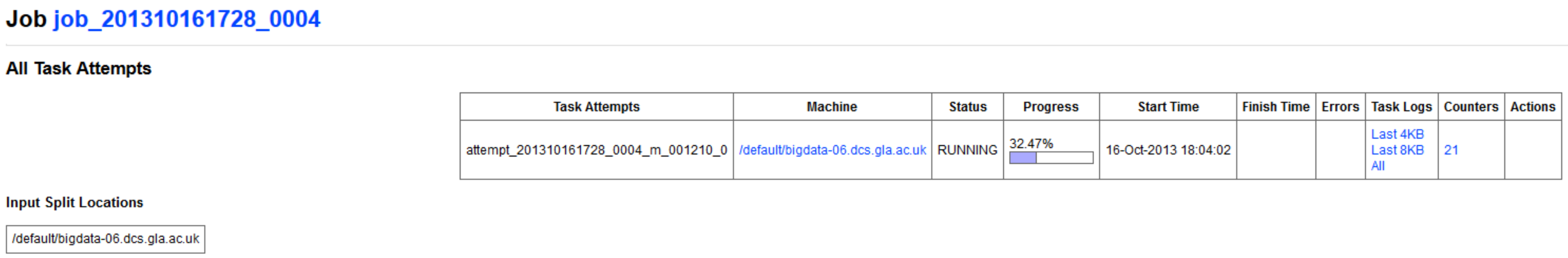
JobTracker UI :: Task information

The Task Page



Task	Complete	Status	Start Time	Finish Time	Errors	Counters
task_201310161728_0004_m_001070	96.87%		16-Oct-2013 18:01:38			24
task_201310161728_0004_m_001074	94.63%		16-Oct-2013 18:01:39			24
task_201310161728_0004_m_001076	85.89%		16-Oct-2013 18:01:40			24
task_201310161728_0004_m_001081	87.50%		16-Oct-2013 18:01:47			24
task_201310161728_0004_m_001083	84.84%		16-Oct-2013 18:01:48			24
task_201310161728_0004_m_001085	84.52%		16-Oct-2013 18:01:49			24
task_201310161728_0004_m_001087	74.89%		16-Oct-2013 18:01:54			24
task_201310161728_0004_m_001089	67.75%		16-Oct-2013 18:02:03			24
task_201310161728_0004_m_001091	66.82%		16-Oct-2013 18:02:04			24
task_201310161728_0004_m_001094	79.34%		16-Oct-2013 18:02:07			24
task_201310161728_0004_m_001096	60.58%		16-Oct-2013 18:02:07			24
task_201310161728_0004_m_001098	70.95%		16-Oct-2013 18:02:08			24
task_201310161728_0004_m_001100	62.50%		16-Oct-2013 18:02:10			24
task_201310161728_0004_m_001102	64.82%		16-Oct-2013 18:02:14			24
task_201310161728_0004_m_001104	45.31%		16-Oct-2013 18:02:19			24
task_201310161728_0004_m_001107	56.77%		16-Oct-2013 18:02:21			24
task_201310161728_0004_m_001108	96.17%		16-Oct-2013 18:02:24			24
task_201310161728_0004_m_001109	47.66%		16-Oct-2013 18:02:25			24
task_201310161728_0004_m_001111	54.49%		16-Oct-2013 18:02:25			24
task_201310161728_0004_m_001113	50.00%		16-Oct-2013 18:02:25			24
task_201310161728_0004_m_001114	89.32%		16-Oct-2013 18:02:26			24
task_201310161728_0004_m_001115	37.17%		16-Oct-2013 18:02:26			24
task_201310161728_0004_m_001116	68.67%		16-Oct-2013 18:02:32			24

The Task Detail Page



Job job_201310161728_0004

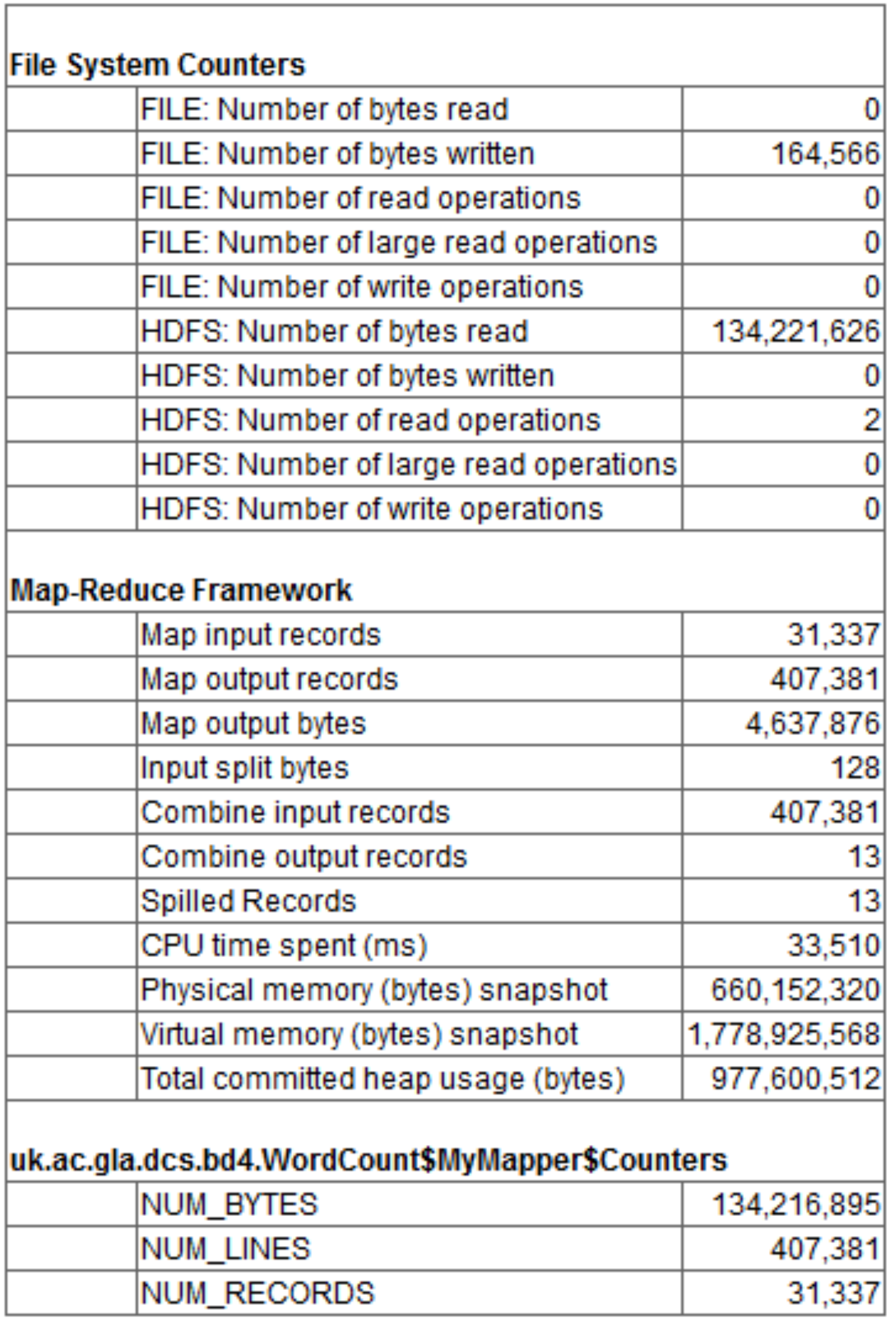
All Task Attempts

Task Attempts	Machine	Status	Progress	Start Time	Finish Time	Errors	Task Logs	Counters	Actions
attempt_201310161728_0004_m_001210_0	/default/bigdata-06.dcs.gla.ac.uk	RUNNING	32.47%	16-Oct-2013 18:04:02			Last 4KB Last 8KB All	21	

Input Split Locations

/default/bigdata-06.dcs.gla.ac.uk

Counters for task_201310161728_0004_m_001165



File System Counters		
	FILE: Number of bytes read	0
	FILE: Number of bytes written	164,566
	FILE: Number of read operations	0
	FILE: Number of large read operations	0
	FILE: Number of write operations	0
	HDFS: Number of bytes read	134,221,626
	HDFS: Number of bytes written	0
	HDFS: Number of read operations	2
	HDFS: Number of large read operations	0
	HDFS: Number of write operations	0
Map-Reduce Framework		
	Map input records	31,337
	Map output records	407,381
	Map output bytes	4,637,876
	Input split bytes	128
	Combine input records	407,381
	Combine output records	13
	Spilled Records	13
	CPU time spent (ms)	33,510
	Physical memory (bytes) snapshot	660,152,320
	Virtual memory (bytes) snapshot	1,778,925,568
	Total committed heap usage (bytes)	977,600,512
uk.ac.gla.dcs.bd4.WordCount$MyMapper$Counters		
	NUM_BYTES	134,216,895
	NUM_LINES	407,381
	NUM_RECORDS	31,337

Hadoop programming :: Hadoop data types

Hadoop data types

```
public interface Writable {
        void readFields(DataInput in);
        void write(DataOutput out);
}
```

```
public class MyWritable implements Writable {
    private int value;
    private long timestamp;

    public void write(DataOutput out) throws IOException {
            out.writeInt(value);
            out.writeLong(timestamp);
    }

    public void readFields(DataInput in) throws IOException {
            value = in.readInt();
            timestamp = in.readLong();
    }

    public static MyWritable read(DataInput in) throws IOException {
            MyWritable w = new MyWritable();
            w.readFields(in);
            return w;
    }
}
```

Hadoop data types

```
public interface WritableComparable<T> extends Writable, Comparable<T> {
        // Writable -> void readFields(DataInput in), void write(DataOutput out);
        // Comparable<T> -> int compareTo(T o);
}
```

```
public class MyWritableComparable implements WritableComparable {
        private int value;
        private long timestamp;

        public void write(DataOutput out) throws IOException {
                out.writeInt(value);
                out.writeLong(timestamp);
        }

        public void readFields(DataInput in) throws IOException {
                value = in.readInt();
                timestamp = in.readLong();
        }

        public int compareTo(MyWritableComparable o) {
                return (this.value < o.value ? -1 : (this.value == o.value ? 0 : 1));
        }

        public int hashCode() {
                final int prime = 31;
                return prime * (prime + value) + (int) (timestamp ^ (timestamp >>> 32));
        }
}
```

Hadoop data types

- ObjectWritable
- GenericWritable
- NullWritable

Generic or Abstract

- BooleanWritable
- ByteWritable
- ShortWritable, IntWritable, LongWritable
- FloatWritable, DoubleWritable
- VIntWritable, VLongWritable
- Text

Specific Java Data Types

Collection

- BytesWritable
- ArrayPrimitiveWritable, ArrayWritable, TwoDArrayWritable

- MapWritable, SortedMapWritable
- EnumSetWritable

Seldom Used

Hadoop programming :: Basic Mapper/Reducer methods

Hadoop data types

Context Object

Runs once when when this mapper instance is instantiated

Called in a loop

Runs once before destructor

```
public class Mapper<KEYIN,VALUEIN,KEYOUT,VALUEOUT> {
        static class Context { ... }
        protected void setup(Context context) { ... }
        protected void map(KEYIN key, VALUEIN value, Context context) { ... }
        protected void cleanup(Context context) { ... }
        void run(Context context) {
                setup(context);

                while (context.nextKeyValue())
                        map(context.getCurrentKey(), context.getCurrentValue(), context);

                cleanup(context);
        }
}

public class Reducer<KEYIN,VALUEIN,KEYOUT,VALUEOUT> {
        static class Context { ... }
        protected void setup(Context context) { ... }
        protected void reduce(KEYIN key, Iterable<VALUEIN> values, Context context) {
                ... }
        protected void cleanup(Context context) { ... }
        void run(Context context) {
                setup(context);
                while (context.nextKey())
                        reduce(context.getCurrentKey(), context.getValues(), context);
                cleanup(context);
        }
}
```

Hadoop programming :: Custom input/output formats

Custom InputFormat

```
public abstract class InputFormat<KEY,VALUE> {
        abstract List<InputSplit> getSplits(JobContext context);
        abstract RecordReader<KEY,VALUE> createRecordReader(InputSplit split,
                TaskAttemptContext context);
}

public abstract class RecordReader<KEY,VALUE> implements Closeable {
        abstract void initialize(InputSplit split, TaskAttemptContext context);
        abstract void close();
        abstract boolean nextKeyValue();
        abstract KEY getCurrentKey();
        abstract VALUE getCurrentValue();
        abstract float getProgress();
}

public abstract class InputSplit {
        abstract long getLength();
        abstract String[] getLocations();
}
```

Hadoop programming :: Word Count galore

Word Count v0 :: Built-in mappers/reducers

Word Count v0

```
public class WordCount extends Configured implements Tool {
        public int run(String[] args) throws Exception {
                Job job = new Job();
                job.setJobName("WordCount-v0");
                job.setJarByClass(WordCount.class);

                job.setMapperClass(TokenCounterMapper.class);
                job.setReducerClass(IntSumReducer.class);

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

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

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

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

Word Count v1 :: User-defined mappers/reducers

Word Count v1

```
public class WordCount extends Configured implements Tool {

   static class Map extends org.apache.hadoop.mapreduce.Mapper<LongWritable, Text, Text, IntWritable> {
               private final static IntWritable one = new IntWritable(1);
               private Text word = new Text();

               public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
                              String line = value.toString();
                              StringTokenizer tokenizer = new StringTokenizer(line);
                              while (tokenizer.hasMoreTokens()) {
                                             word.set(tokenizer.nextToken());
                                             context.write(word, one);
                              }
               }
   }

  public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {
               public void reduce(Text key, Iterable<IntWritable> values, Context context) throws
               IOException, InterruptedException {
                              int sum = 0;
                              for (IntWritable value: values)
                                             sum += value.get();
                              context.write(key, new IntWritable(sum));
               }
   }
```

Word Count v1 (cont.)

```
    public int run(String[] args) throws Exception {
        Job job = new Job(); job.setJobName("WordCount-v1");
        job.setJarByClass(WordCount.class);
        job.setMapperClass(Map.class);

        job.setCombinerClass(Reduce.class);
        job.setReducerClass(Reduce.class);
        job.setInputFormatClass(TextInputFormat.class);
        FileInputFormat.setInputPaths(job, new Path(args[0]));

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        job.setOutputFormatClass(TextOutputFormat.class);
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

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

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

The default is FileInputFormat

The default is FileOutputFormat

Word Count v2 ::
Distributed cache + configuration + counters + status messages + progress report

Word Count v2

- Count occurrences of words in the input stream, but also:
  - Allow user to define patterns/words to be skipped
  - Count total number of words processed
  - Report progress and update status messages as we go

Word Count v2

WordCount.java

```
public class WordCount extends Configured implements Tool {
    public int run(String[] args) throws Exception {
            Job job = new Job();
            job.setJobName("WordCount-v3");
            job.setJarByClass(WordCount.class);
            job.setMapperClass(Map.class);
            job.setCombinerClass(Reduce.class);
            job.setReducerClass(Reduce.class);
            job.setInputFormatClass(TextInputFormat.class);
            job.setOutputKeyClass(Text.class);
            job.setOutputValueClass(IntWritable.class);
            job.setOutputFormatClass(TextOutputFormat.class);

            List<String> other_args = new ArrayList<String>();
            for (int i = 0; i < args.length; ++i) {
                    if ("-skip".equals(args[i])) {
                            DistributedCache.addCacheFile(new Path(args[++i]).toUri(),
                                    job.getConfiguration());
                            job.getConfiguration().setBoolean("wordcount.skip.patterns", true);
                    } else
                            other_args.add(args[i]);
            }
            FileInputFormat.setInputPaths(job, new Path(other_args.get(0)));
            FileOutputFormat.setOutputPath(job, new Path(other_args.get(1)));
```

Important: Configurations/ Arguments on the job client are only propagate to mapper and reducer if they get added here!!

Word Count v2 (cont.)

WordCount.java (cont.)

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

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

Same Reducer as Before

Word Count v2

Parse the Skip File

Map.java

```
public class Map extends Mapper<LongWritable, Text, Text, IntWritable>{
        static enum Counters { INPUT_WORDS }
        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();
        private boolean caseSensitive = true;
        private Set<String> patternsToSkip = new HashSet<String>();
        private long numRecords = 0;
        private String inputFile;

        private void parseSkipFile(Path patternsFile) {
                try {
                        BufferedReader fis = new BufferedReader(new
                                FileReader(patternsFile.toString()));
                        String pattern = null;
                        while ((pattern = fis.readLine()) != null)
                                patternsToSkip.add(pattern);
                        fis.close();
                } catch (IOException ioe) {
                        System.err.println("Caught exception while parsing the cached file '" +
                                patternsFile + "' : " + StringUtils.stringifyException(ioe));
                }
        }
```

Word Count v2 (cont.)

Map.java (cont.)

```
public void setup(Context context) {
        Configuration conf = context.getConfiguration();
        inputFile = conf.get("map.input.file");

        if (conf.getBoolean("wordcount.skip.patterns", false)) {
                Path[] patternsFiles = new Path[0];
                try {
      ------->          patternsFiles = DistributedCache.getLocalCacheFiles(conf);
                } catch (IOException ioe) {
                        System.err.println("Caught exception while getting cached files: " +
                                        StringUtils.stringifyException(ioe));
                }
                for (Path patternsFile : patternsFiles)
                        parseSkipFile(patternsFile);
        }
}

public void cleanup(Context context) {
        patternsToSkip.clear();
}
```

Word Count v2 (cont.)

Map.java (cont.)

```
public void map(LongWritable key, Text value, Context context) throws
                        IOException, InterruptedException {
            String line = caseSensitive ? value.toString() :
                        value.toString().toLowerCase();

            for (String pattern : patternsToSkip)
                        line = line.replaceAll(pattern, "");

            StringTokenizer tokenizer = new StringTokenizer(line);
            while (tokenizer.hasMoreTokens()) {
                        word.set(tokenizer.nextToken());
                        context.write(word, one);
    ——→                 context.getCounter(Counters.INPUT_WORDS).increment(1);
            }

            if ((++numRecords % 100) == 0)
    ——→                 context.setStatus("Finished processing " + numRecords + " records " +
                                    "from the input file: " + inputFile);
            }
}
```

Word Count v2 (cont.)

Reduce.java

```
public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable>
{
        public void reduce (Text key, Iterable<IntWritable> values, Context context)
                throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable value: values)
                sum += value.get();
        context.write(key, new IntWritable(sum));
        }
}
```

Same Reducer as Before

Word Count v3 ::
Custom InputFormat + partitioner + more counters

Word Count v3

- Count occurrences of first word in each “record” in the input stream, but this time:
  - Input records are spread across multiple lines …
  - ... with a special sequence (\t\t\t) separating them …
  - Also make a custom partitioner so there are 27 reducers (for [a-z] + <everything else>)...
  - Also count total number of bytes, lines, and records processed

Word Count v3

MyPartitioner.java

```
public class MyPartitioner extends Partitioner<Text, IntWritable> {
        int getPartition(Text key, IntWritable value, int numPartitions) {
                int c = Character.toLowerCase(key.toString().charAt(0));
                if (c < ’a’ || c > ’z’)
                        return numPartitions - 1;
                return (int)Math.floor((float)(numPartitions - 2) * (c-’a’)/(’z’-’a’));
        }
}
```

MyInputFormat.java

```
public class MyInputFormat extends FileInputFormat<LongWritable, Text> {
        public RecordReader<LongWritable, Text> createRecordReader(InputSplit split,
                        TaskAttemptContext context) {
                return new MyRecordReader();
        }
}
```

Word Count v3 (cont.)

MyRecordReader.java

```
public class MyRecordReader extends RecordReader<LongWritable, Text> {
            private static final byte[] recordSeparator = "\t\t\t".getBytes();
            private FSDataInputStream fsin;
            private long start, end;
            private boolean stillInChunk = true;
            private DataOutputBuffer buffer = new DataOutputBuffer();
            private LongWritable key = new LongWritable();
            private Text value = new Text();

            public void initialize(InputSplit inputSplit, TaskAttemptContext context) throws IOException {
                        FileSplit split = (FileSplit) inputSplit;
                        Configuration conf = context.getConfiguration();
                        Path path = split.getPath();
                        FileSystem fs = path.getFileSystem(conf);

                        fsin = fs.open(path);
                        start = split.getStart();
                        end = split.getStart() + split.getLength();
                        fsin.seek(start);

                        if (start != 0)
                                    readRecord(false);
            }
```

Word Count v3 (cont.)

MyRecordReader.java (cont.)

```
private boolean readRecord(boolean withinBlock) throws IOException {
        int i = 0, b;
        while (true) {
                if ((b = fsin.read()) == -1)
                        return false;
                if (withinBlock)
                        buffer.write(b);
                if (b == recordSeparator[i]) {
                        if (++i == recordSeparator.length)
                                return fsin.getPos() < end;
                } else
                        i = 0;
        }
}
```

Word Count v3 (cont.)

MyRecordReader.java (cont.)

```
        public boolean nextKeyValue() throws IOException {
                if (!stillInChunk)
                        return false;
                boolean status = readRecord(true);
                value = new Text();
                value.set(buffer.getData(), 0, buffer.getLength());
                key.set(fsin.getPos());
                buffer.reset();
                if (!status)
                        stillInChunk = false;
                return true;
        }

        public LongWritable getCurrentKey() { return key; }

        public Text getCurrentValue() { return value; }

        public float getProgress() throws IOException {
                return (float) (fsin.getPos() - start) / (end - start);
        }

        public void close() throws IOException { fsin.close(); }
}
```

Word Count v3 (cont.)

MyMapper.java

```
public class MyMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
    static enum Counters { NUM_RECORDS, NUM_LINES, NUM_BYTES }
    private Text _key = new Text();
    private IntWritable _value = new IntWritable();

    protected void map(LongWritable key, Text value, Context context) throws
                                IOException, InterruptedException {
        StringTokenizer tokenizer = new StringTokenizer(value.toString(), "\n");
        while (tokenizer.hasMoreTokens()) {
            String line = tokenizer.nextToken();
            int sep = line.indexOf(' ');
            _key.set((sep == -1) ? line : line.substring(0, line.indexOf(' ')));
            _value.set(1);
            context.write(_key, _value);
            context.getCounter(Counters.NUM_LINES).increment(1);
        }

        context.getCounter(Counters.NUM_BYTES).increment(value.getLength());
        context.getCounter(Counters.NUM_RECORDS).increment(1);
    }
}
```

Word Count v3 (cont.)

MyReducer.java

```
public class MyReducer extends Reducer<Text, IntWritable, Text,
IntWritable> {
        private IntWritable _value = new IntWritable();
        protected void reduce(Text key, Iterable<IntWritable> values, Context
                        context) throws IOException, InterruptedException {
                int sum = 0;
                for (Iterator<IntWritable> it = values.iterator(); it.hasNext();)
                        sum += it.next().get();
                _value.set(sum);
      ──────→   context.write(key, _value);
        }
}
```

Word Count v3 (cont.)

WordCount.java

```
public class WordCount extends Configured implements Tool {
        public int run(String[] args) throws Exception {
                Job job = new Job();
                job.setJobName("MyWordCount(" + args[0] + ")");
                job.setJarByClass(WordCount.class);
    →           job.setInputFormatClass(MyInputFormat.class);
                job.setOutputFormatClass(TextOutputFormat.class);
                job.setMapperClass(MyMapper.class);
    →           job.setPartitionerClass(MyPartitioner.class);
                job.setMapOutputKeyClass(Text.class);
                job.setMapOutputValueClass(IntWritable.class);
                job.setReducerClass(MyReducer.class);
                job.setCombinerClass(MyReducer.class);
                FileInputFormat.setInputPaths(job, new Path(args[0]));
                FileOutputFormat.setOutputPath(job, new Path(job.getJobName() +
                        "_output"));
                job.submit();
                return job.waitForCompletion(true) ? 0 : 1;
        }

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