Massive Distributed Processing using Map-Reduce

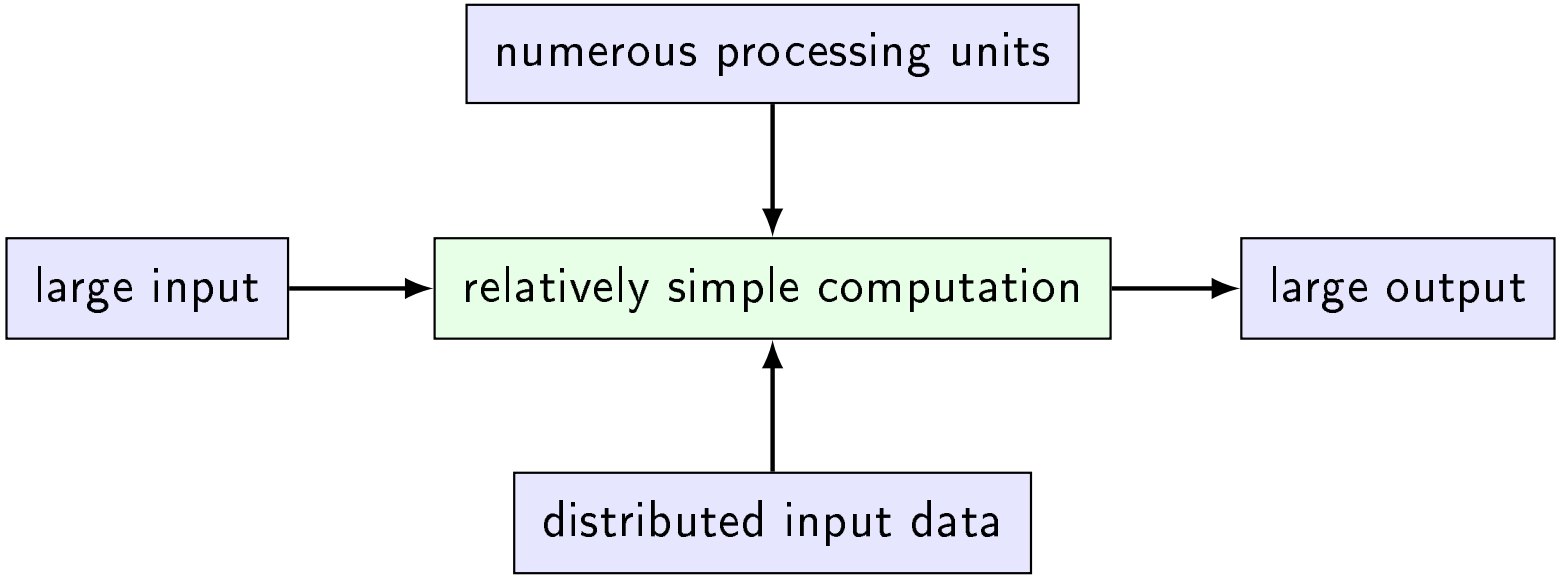
(Przetwarzanie rozproszone w technice *map-reduce*)

Dawid Weiss

Institute of Computing Science
Poznań University of Technology

01/2007

Massive distributed processing problems



- Computations are most often very simple.
- Data instances huge.
- Input can be fragmented into continuous 'splits'.

Examples of MDP problems

- Search/ scan problems (grep).
- Counting problems (URL access).
- Indexing problems (reverse link, inverted indices).
- Sorting problems.

The overhead of custom solutions

- Parallelization is never easy.
- Job scheduling.
- Failure detection and recovery.
- Job progress/ status tracking.

The overhead of custom solutions

- Parallelization is never easy.
- Job scheduling.
- Failure detection and recovery.
- Job progress/ status tracking.

Simplicity of the original computation is lost.

Map Reduce

Map Reduce (Jeffrey Dean, Sanjay Ghemawat; Google Inc.)

A technique of automatic parallelization of computations by enforcing a **restricted programming model**, derived from functional languages.

- Inspiration: **map** and **reduce** operations in Lisp.
- Hide the messy details, keep the programmer happy.
- Achieve scalability, robustness and fault-tolerance by adding processing units.

The programming model

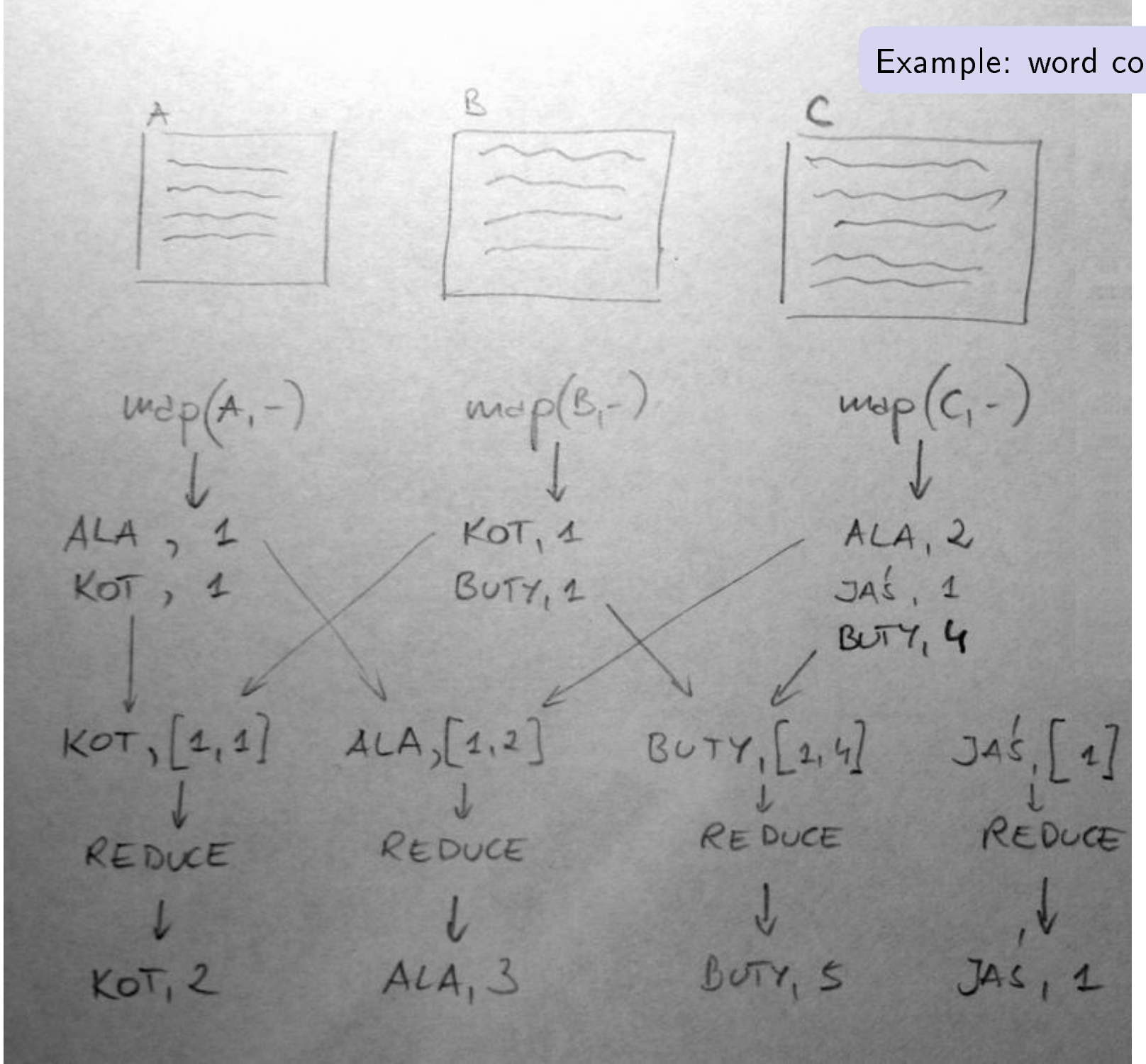
1. The input is parcelled into keys and associated values.
2. The **map** function takes (in_key, in_value) pairs and produces (out_key, im_value) pairs:

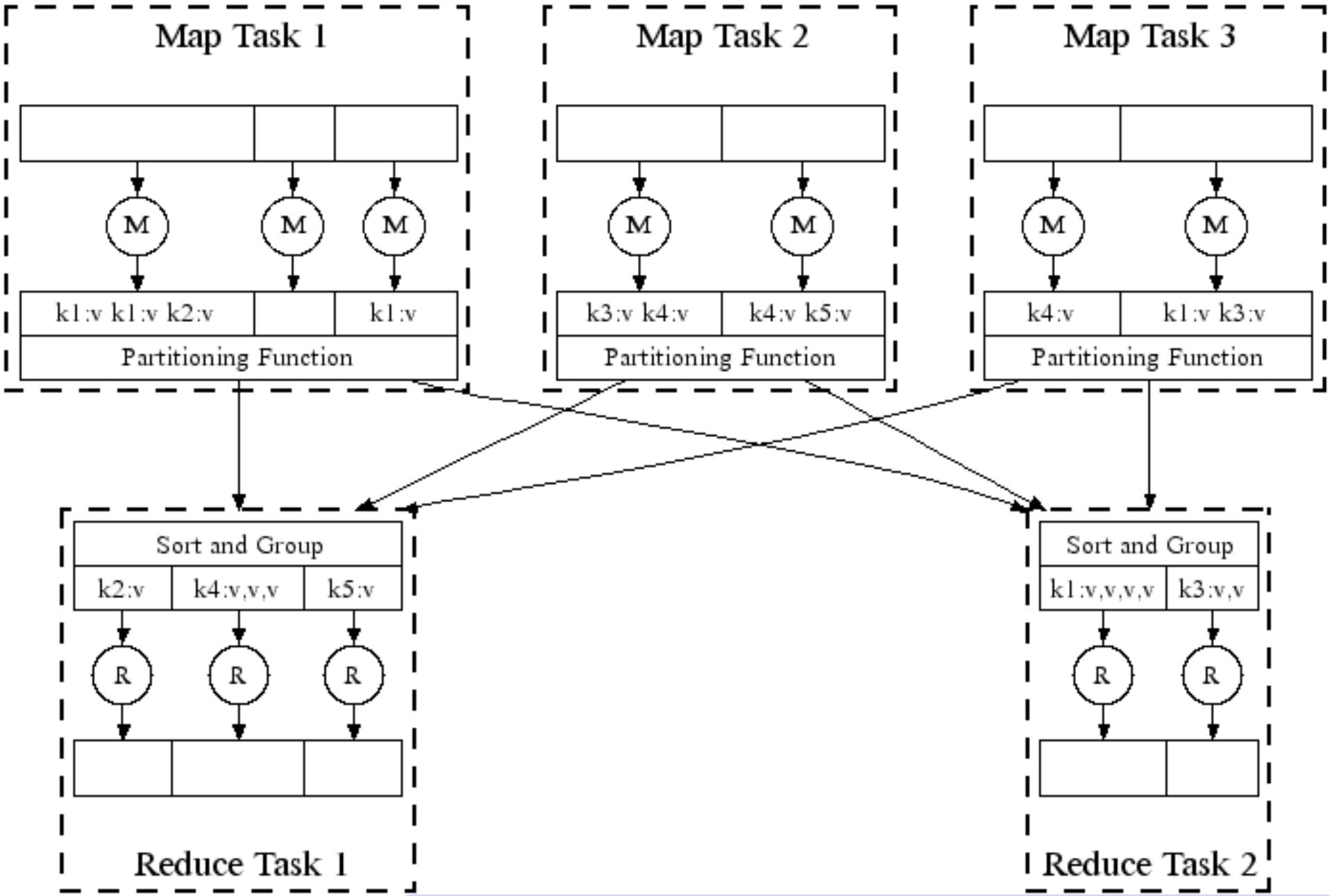
```
map(in_key, in_value)
            → [(out_key, im_value)]
```

3. All values for identical keys are grouped.
4. The **reduce** function reduces a list of values for a single key to a fewer list of results (typically one or zero):

```
reduce(out_key_x, [im_value_1, im_value_2, ...])
            → [out_value]
```

Example: word counting





Source: http://labs.google.com/papers/mapreduce.html

Example: word counting [Dean and Ghemawat, 2004]

Map function

```
map(String key, String value):
   // key: document name
   // value: document contents
   for each word w in value:
      EmitIntermediate(w, "1");
```

Reduce function

```
reduce(String key, Iterator values):
   // key: a word
   // values: a list of counts
   int result = 0;
   for each v in values:
      result += ParseInt(v);
      Emit(AsString(result));
```

More examples

- Distributed grep.

```
map:    (--, line) -> (line)
reduce: identity
```

- Reverse Web link graph.

```
map:    (source-url, html-content) -> (target-url, source-url)
reduce: (target-url, [source-urls]) -> (target-url, concat(source-urls))
```

- Inverted index of documents.

```
map:    (doc-id, content) -> (word, doc-id)
reduce: (word, [doc-ids]) -> (word, concat(doc-ids))
```

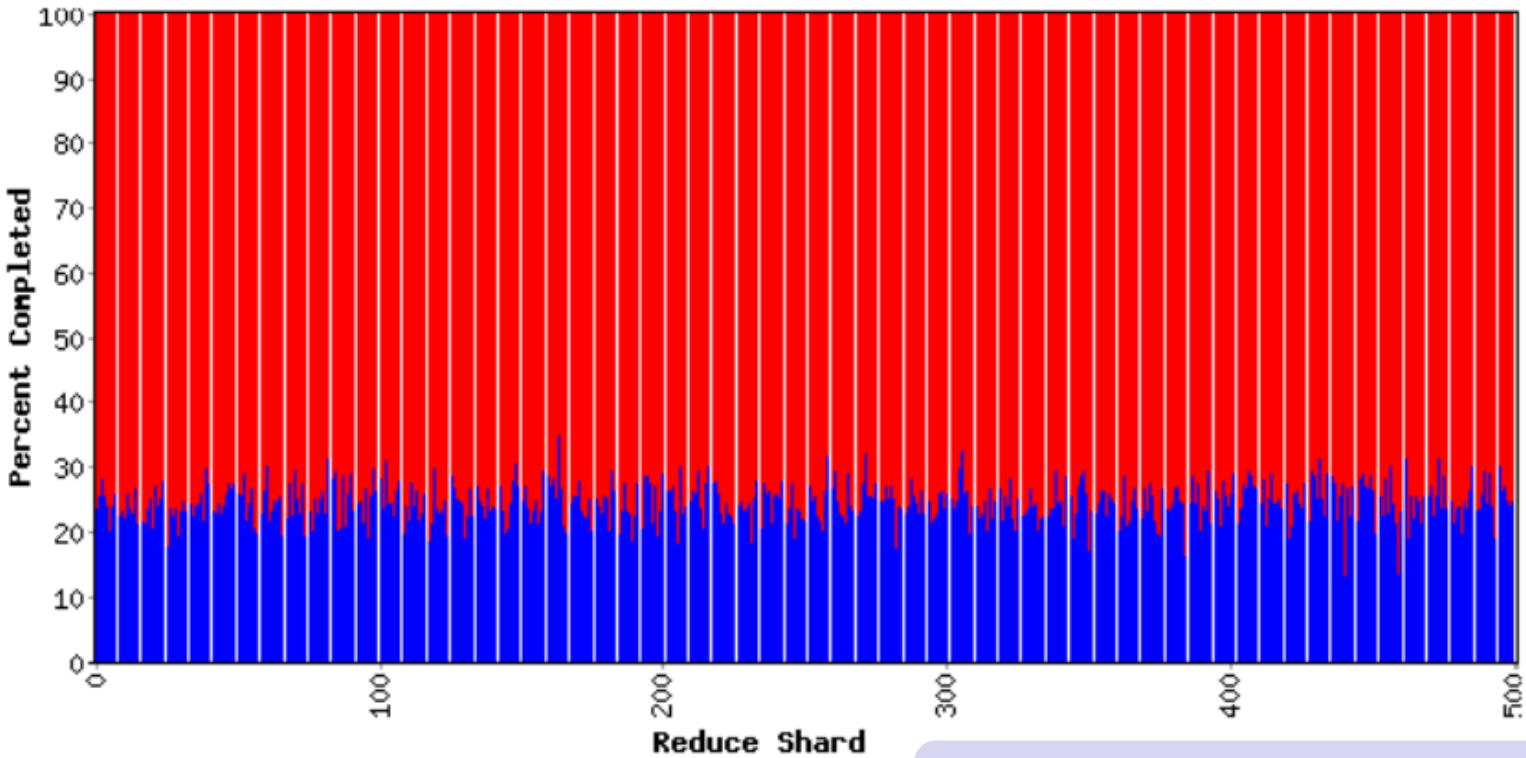
- More complex tasks achieved by combining Map-Reduce jobs (the indexing process at Google – more than 20 MR tasks!).

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 31 min 34 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	500	0	523499.2	523499.5	523499.5
Reduce	500	0	500	523499.5	133837.8	136929.6



Counters

Variable	Minute
Mapped (MB/s)	0.0
Shuffle (MB/s)	0.1
Output (MB/s)	1238.8
doc-index-hits	0
docs-indexed	0
dups-in-index-merge	0
mr-merge-calls	51738599
mr-merge-	51738599

Example reduce phase (indexing at Google).

Further improvements

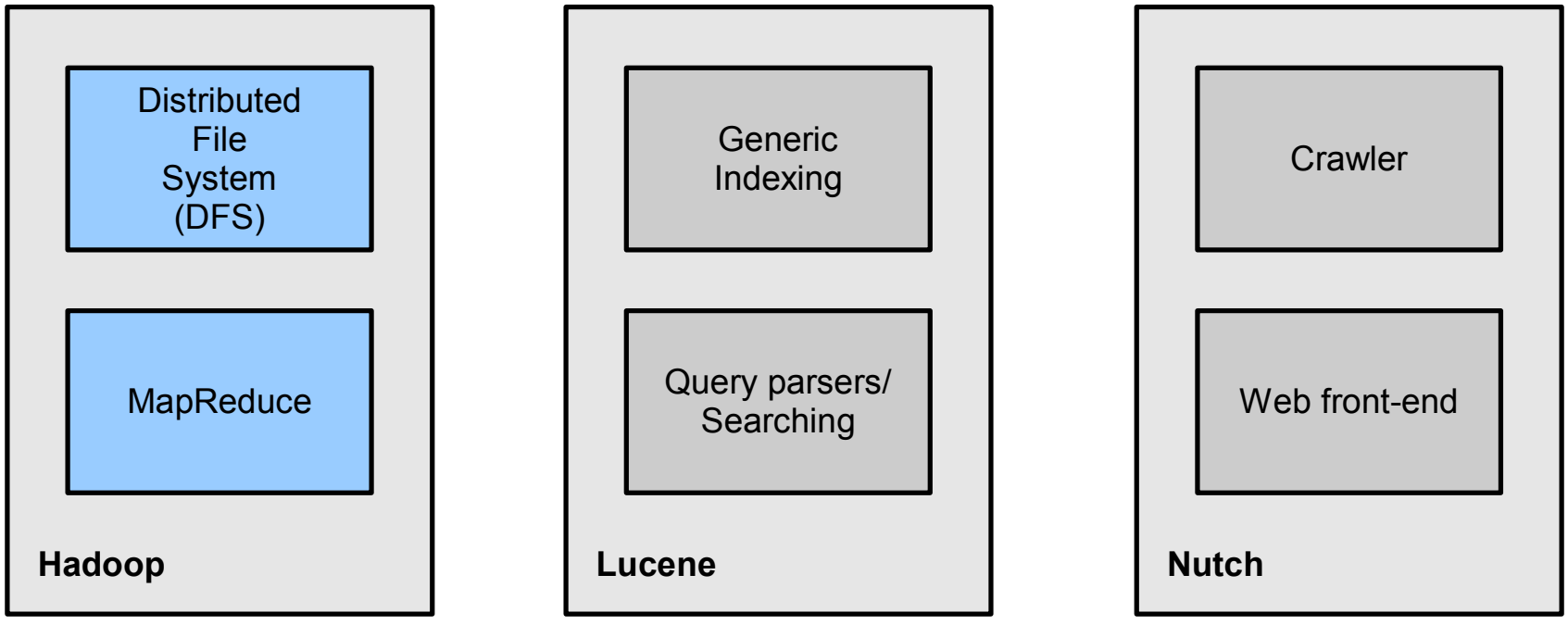
- Combiners (avoid too much intermediate traffic).
- Speculative execution (anticipate invalid/ broken nodes).
- Load balancing (split your input into possibly many map tasks).
- Data access optimizations (keep processing close to the input).

The Hadoop project

- Mark Carafella, Doug Cutting and others.
- Originally part of Apache Lucene codebase.
- Impressively dynamic growth as a Lucene sub-project.
- Apache-license.

http://lucene.apache.org/hadoop/

The open source MapReduce environment



HDFS assumptions

HDFS is inspired by GFS (Google File System).

Design goals:

- expect hardware failures (processes, disk, nodes),
- streaming data access, large files (TB of data),
- simple coherence model (one writer, many readers),
- optimization of computation in MapReduce (locality),
- single master (name node), multiple slaves (data nodes).

Hadoop requirements

Installation/ operation requirements:

- Java `1.5.x` or higher, preferably from Sun.
- Linux and Windows (under CygWin).

MapReduce jobs:

- Preferably implemented in Java.
- Hadoop Streaming (arbitrary shell commands).
- C/C++ APIs to HDFS.

Example: word counting

```
1  /**
2   * Counts the words in each line.
3   * For each line of input, break the line into words and
4   * emit them as (<b>word</b>, <b>1</b>).
5   */
6  public static class MapClass
7    extends MapReduceBase implements Mapper {
8
9    private final static IntWritable one = new IntWritable(1);
10   private Text word = new Text();
11
12   public void map(WritableComparable key, Writable value,
13       OutputCollector output, Reporter reporter) throws IOException {
14     final String line = ((Text) value).toString();
15     final StringTokenizer itr = new StringTokenizer(line);
16     while (itr.hasMoreTokens()) {
17       word.set(itr.nextToken());
18       output.collect(word, one);
19     }
20   }
21 }
```

Example: word counting

```
1  /**
2   * A reducer class that just emits the sum of the input values.
3   */
4  public static class Reduce
5    extends MapReduceBase implements Reducer {
6
7    public void reduce(WritableComparable key, Iterator values,
8       OutputCollector output, Reporter reporter)
9       throws IOException
10   {
11     int sum = 0;
12     while (values.hasNext()) {
13       sum += ((IntWritable) values.next()).get();
14     }
15     output.collect(key, new IntWritable(sum));
16   }
17 }
```

Example: word counting

```
1  public static void main(String[] args) throws IOException {
2    final JobConf conf = new JobConf(WordCount.class);
3    conf.setJobName("wordcount");
4
5    // The keys are words (strings).
6    conf.setOutputKeyClass(Text.class);
7    // The values are counts (ints).
8    conf.setOutputValueClass(IntWritable.class);
9
10   conf.setMapperClass(MapClass.class);
11   conf.setCombinerClass(Reduce.class);
12   conf.setReducerClass(Reduce.class);
13
14   // [...]
15
16   conf.setInputPath(new Path(input));
17   conf.setOutputPath(new Path(output));
18
19   // Uncomment to run locally in a single process
20   // conf.set("mapred.job.tracker", "local");
21
22   JobClient.runJob(conf);
23 }
```

The trickery of Hadooping...

- Windows installation often broken (scripts, paths).
- Documentation scarce and not up-to-date.
- Real setup of a distributed cluster requires some initial work (account setup, moving distributions around).

Requirements

- Linux-based systems.
- Shell access, password-less SSH access within the cluster's nodes.
- Certain open ports within the cluster (for DFS, trackers, Web interface).

Requirements

- Linux-based systems.
- Shell access, password-less SSH access within the cluster's nodes.
- Certain open ports within the cluster (for DFS, trackers, Web interface).

Conclusion

At the moment setting up a Hadoop installation at lab-45 is problematic.

Test installation at lab-142/ lab-143

Installation profile:

- Out-of-the-box installation of Hadoop 0.10.0.
- Cluster of 7, then 28 machines.
- Code/ configuration distribution provided by the NFS.
- One master (name node, job tracker), multiple data nodes (DFS/ MR).

Simple experiments performed:

- DFS performance.
- Word counting example.
- Sorting example.

Setup:

- nodes: 7,
- replication factor: 3,
- block size: 64 MB,
- local file size: 1.6 GB (entire Rzeczpospolita corpus, concatenated).

Setup:

- nodes: 7,
- replication factor: 3,
- block size: 64 MB,
- local file size: 1.6 GB (entire Rzeczpospolita corpus, concatenated).

Results:

- copy to DFS: 5'25s., 5-8 MB/s (network and local-machine bound),
- random-write from within Map-Reduce job, 28 DFS nodes: 2,73 GB – 1'20s.

Access to DFS: command-line and Web frontend.

...ut.poznan.pl:50000'

Started:	Mon Jan 08 18:49:38 CET 2007
Version:	0.10.1-dev, r
Compiled:	Mon Jan 8 15:33:55 CET 2007 by dweiss

Browse the filesystem

Cluster Summary

Capacity	:	70.02 GB
Remaining	:	22.55 GB
Used	:	67.79 %
Live Nodes	:	7
Dead Nodes	:	0

Live Datanodes : 7

Node	Last Contact	Admin State	Size (GB)	Used (%)	Blocks
lab-142-10	2	In Service	10.00	68.17	7
lab-142-11	2	In Service	10.00	66.09	3
lab-142-12	2	In Service	10.00	67.97	6
lab-142-13	1	In Service	10.00	67.61	6
lab-142-14	1	In Service	10.00	69.23	8
lab-142-15	1	In Service	10.00	67.14	5
lab-142-9	0	In Service	10.00	68.32	7

Access to DFS: command-line and Web frontend.

```
dweiss@idss-dw:/usr/home/dweiss/hadoop/hadoop-0.10.0> bin/hadoop fs -ls /experiments/input/
Found 1 items
/experiments/input/all.txt      <r 3>   1659267263
dweiss@idss-dw:/usr/home/dweiss/hadoop/hadoop-0.10.0>
```

```
dweiss@lab-142-8:~/research/hadoop/hadoop-0.10.0> bin/hadoop fsck -blocks
....................................................................................................
....................................................................................................
....................................................................................................
....................................................................................................
....................................................................................................
.....................................................Status: HEALTHY
 Total size:    4642921778 B
 Total blocks:  586 (avg. block size 7923074 B)
 Total dirs:    10
 Total files:   562
 Over-replicated blocks:        0 (0.0 %)
 Under-replicated blocks:       0 (0.0 %)
 Target replication factor:     3
 Real replication factor:       3.0
```

Setup:

- nodes: initially 7, repeated for 28,
- replication factor: 3,
- block size: 64 MB,
- input: DFS file – 1.6 GB („Rzepus”),
- maps: 67, reduces: 7.

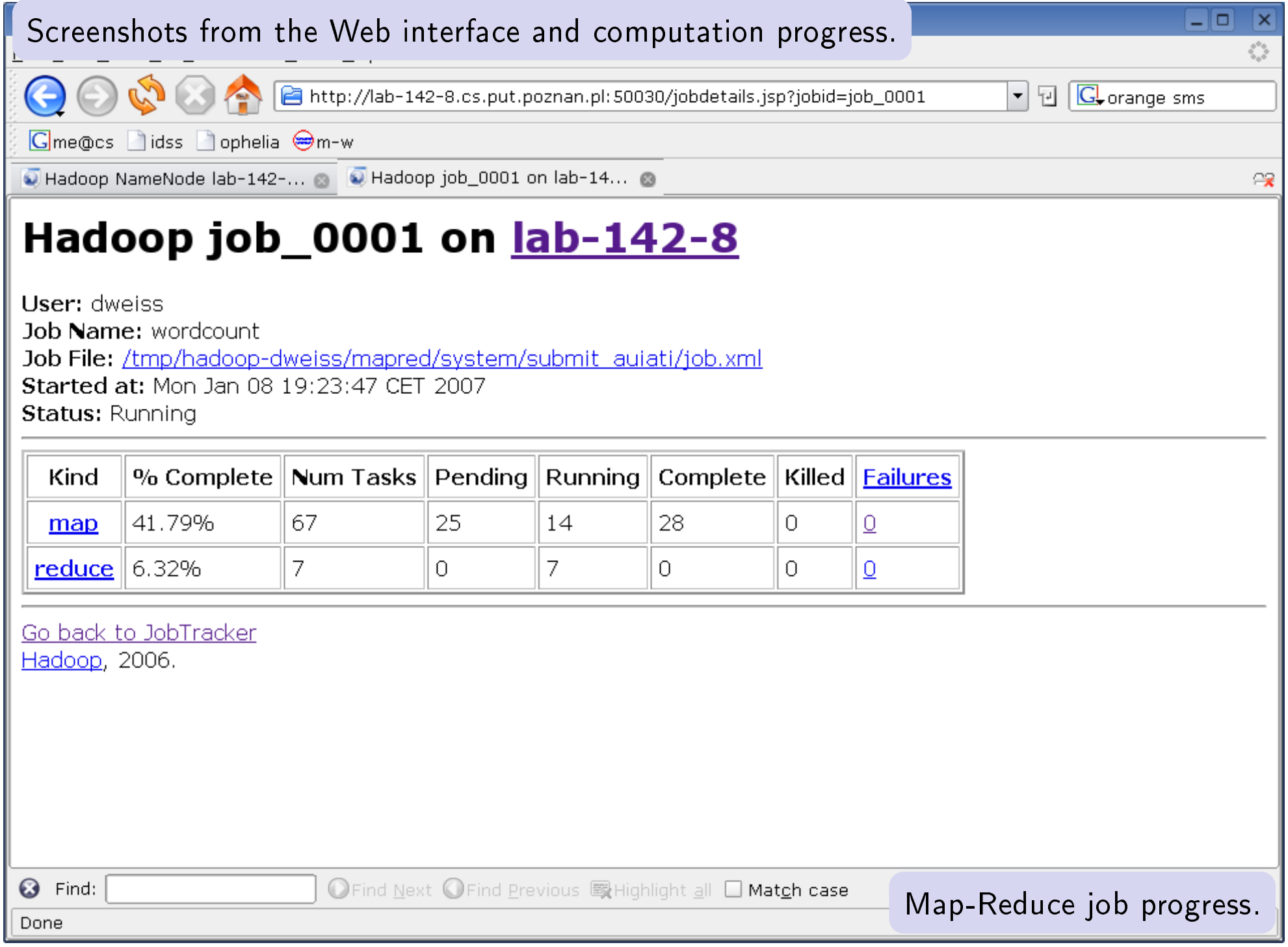
Setup:

- nodes: initially 7, repeated for 28,
- replication factor: 3,
- block size: 64 MB,
- input: DFS file – 1.6 GB („Rzepus"),
- maps: 67, reduces: 7.

Results:

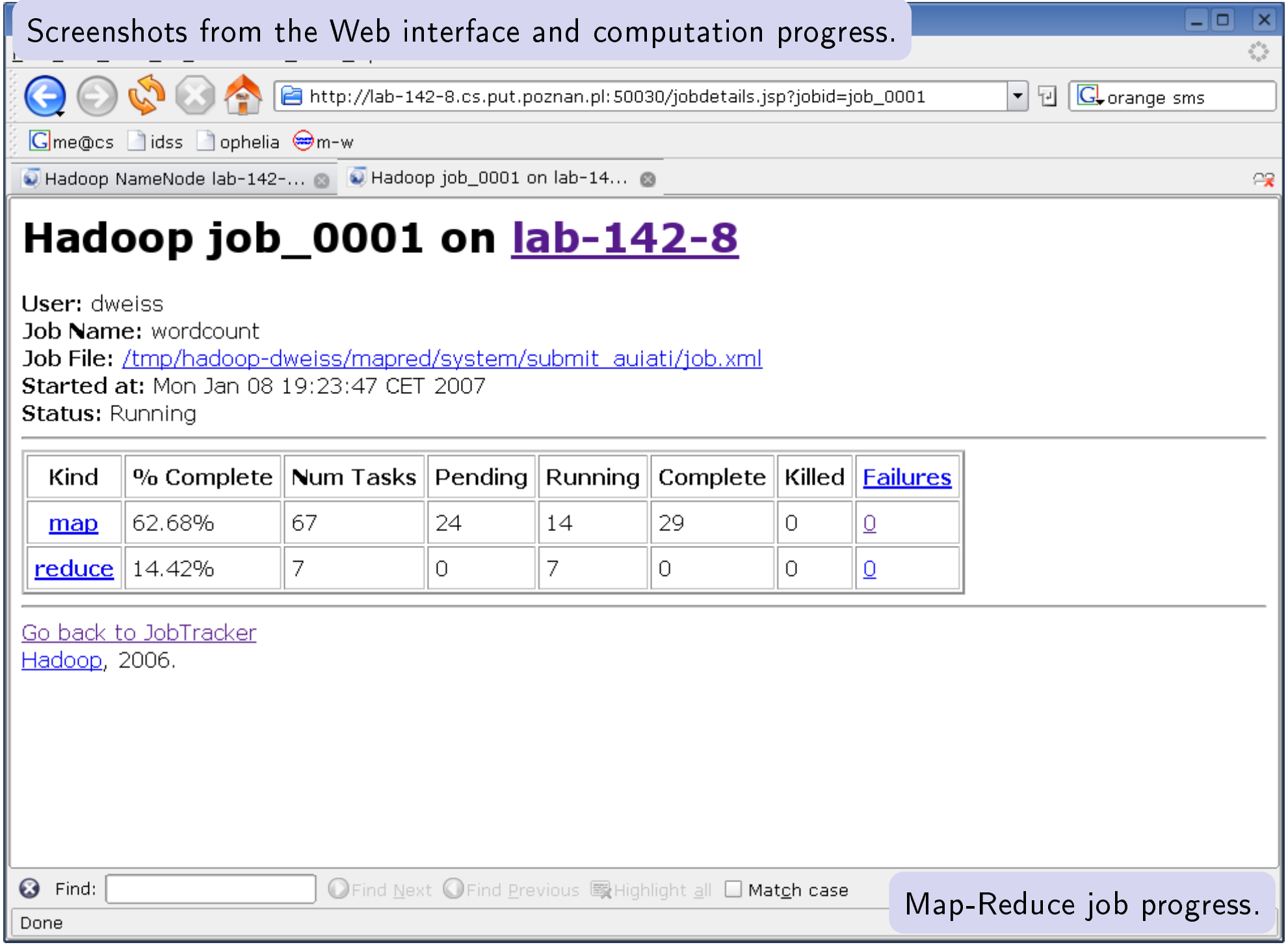
- 7 nodes – 5'31s. (note: full-cycle of read, word count, write),
- 28 nodes – 2'21s,
- 28 nodes (281 maps, 29 reduces) – 2'31s.

Screenshots from the Web interface and computation progress.



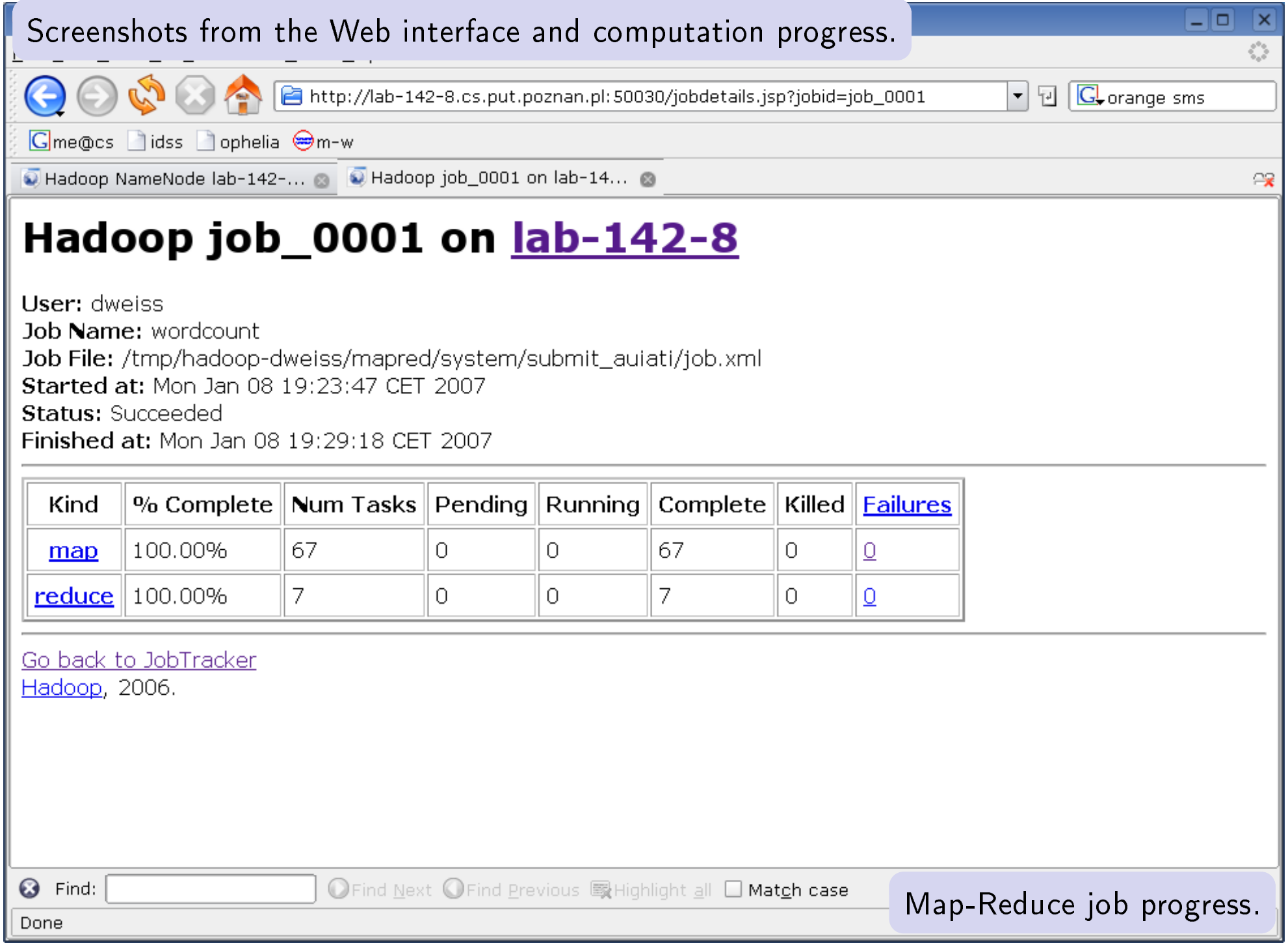
Map-Reduce job progress.

Screenshots from the Web interface and computation progress.



Map-Reduce job progress.

Screenshots from the Web interface and computation progress.



Map-Reduce job progress.

Screenshots from the Web interface and computation progress.

lab-142-8 Hadoop Map/Reduce Administration

Started: Mon Jan 08 19:22:37 CET 2007
Version: 0.10.1-dev, r
Compiled: Mon Jan 8 15:33:55 CET 2007 by dweiss

Cluster Summary

Maps	Reduces	Tasks/Node	Nodes
0	0	2	7

Running Jobs

Running Jobs
none

Completed Jobs

Completed Jobs								
Jobid	User	Name	Map % complete	Map total	Maps completed	Reduce % complete	Reduce total	Reduces completed
job_0001	dweiss	wordcount	100.00%	67	67	100.00%	7	7

MR cluster administration interface.

Screenshots from the Web interface and computation progress.

Contents of directory /experiments/output/out

Go to parent directory

Name	Type	Size	Replication	BlockSize
part-00000	file	13.56 MB	3	13.56 MB
part-00001	file	13.55 MB	3	13.55 MB
part-00002	file	13.57 MB	3	13.57 MB
part-00003	file	13.53 MB	3	13.53 MB
part-00004	file	13.53 MB	3	13.53 MB
part-00005	file	13.54 MB	3	13.54 MB
part-00006	file	13.55 MB	3	13.55 MB

Go back to DFS home

Local logs

Log directory

Hadoop, 2006.

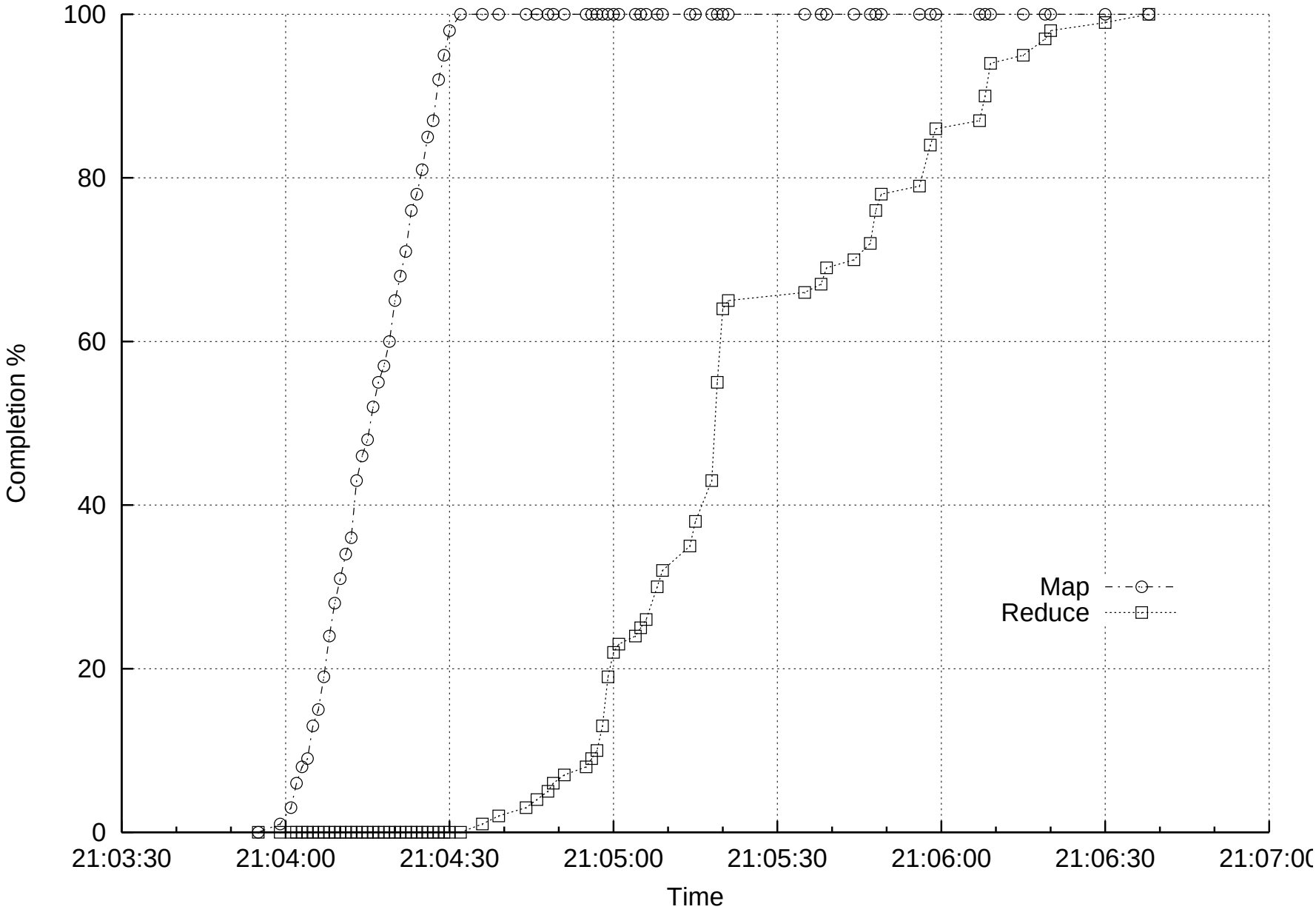
The output.

Screenshots from the Web interface and computation progress.

```
informatyczni<p>        1
informatyk<p>   1
informatyka-administratora      1
informatyka-specjalist 1
informatyka</TD></TR>   6
informatyka</TITLE>     5
informatyka</b></FONT><p>       2
informatyka?),  1
informatykami), 1
informatykami?  1
informatyki     493
informatyki,    118
informatyki</TD></TR>   15
informatyk</TITLE>      2
informatyków,   59
informatyków</TD></TR>  6
informatyzacja  55
informatyzacji</B>      1
informatyzowania        2
```

The output.

MR progress in the word counting task.



Failure handling (robustness).

Hadoop job_0003 on lab-142-8

User: dweiss
Job Name: wordcount
Job File: /tmp/hadoop-dweiss/mapred/system/submit_9k59xr/job.xml
Started at: Mon Jan 08 20:37:02 CET 2007
Status: Succeeded
Finished at: Mon Jan 08 20:39:23 CET 2007

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failures
map	100.00%	67	0	0	67	0	1
reduce	100.00%	7	0	0	7	0	0

Go back to JobTracker
Hadoop, 2006.

Failure handling (robustness).

Hadoop job_0003 failures on lab-142-8

Attempt	Task	Machine	Error
task_0003_m_000042_0	tip_0003_m_000042	lab-143-14.cs.put.poznan.pl	java.lang.IndexOutOfBoundsException at java.io.ByteArrayOutputStream.write(ByteArra at java.io.DataOutputStream.write(DataOutputStr at org.apache.hadoop.mapred.MRSortResultIterato at org.apache.hadoop.mapred.ReduceTask$ValuesIt at org.apache.hadoop.mapred.ReduceTask$ValuesIt at org.apache.hadoop.examples.WordCount$Reduce. at org.apache.hadoop.mapred.MapTask$MapOutputBu at org.apache.hadoop.mapred.MapTask$MapOutputBu at org.apache.hadoop.mapred.MapTask$MapOutputBu at org.apache.hadoop.mapred.MapTask.run(MapTask at org.apache.hadoop.mapred.TaskTracker$Child.m

Go back to JobTracker

Hadoop, 2006.

Failure handling (robustness).

s on lab-142-8

Machine	Error
43-14.cs.put.poznan.pl	java.lang.IndexOutOfBoundsException at java.io.ByteArrayOutputStream.write(ByteArra at java.io.DataOutputStream.write(DataOutputStr at org.apache.hadoop.mapred.MRSortResultIterato at org.apache.hadoop.mapred.ReduceTask$ValuesIt at org.apache.hadoop.mapred.ReduceTask$ValuesIt at org.apache.hadoop.examples.WordCount$Reduce. at org.apache.hadoop.mapred.MapTask$MapOutputBu at org.apache.hadoop.mapred.MapTask$MapOutputBu at org.apache.hadoop.mapred.MapTask$MapOutputBu at org.apache.hadoop.mapred.MapTask.run(MapTask at org.apache.hadoop.mapred.TaskTracker$Child.m

Broken node?

Setup:

- nodes: 28,
- replication factor: 3,
- block size: 64 MB,
- input: DFS files – total 2,73 GB (random byte sequences),
- maps: 280, reduces: 7.

Setup:

- nodes: 28,
- replication factor: 3,
- block size: 64 MB,
- input: DFS files – total 2,73 GB (random byte sequences),
- maps: 280, reduces: 7.

Results:

- read-sort-write time: 4'18s.

Conclusions

- Map-Reduce is an interesting programming paradigm.
- Automatic paralellism, scalability, fault-tolerance.

- Hadoop provides a cost-effective option for experiments with Map-Reduce.
- Lack of documentation, but source code available.

References

- Dean, J. and Ghemawat, S. (2004). MapReduce: Simplified Data Processing on Large Clusters. In *Proceedings of the 6th Symposium on Operating System Design and Implementation, OSDI '2004*, pages 137–150
- lucene (2007). Apache lucene. On-line: `http://lucene.apache.org/`
- nutch (2007). Apache nutch. On-line: `http://lucene.apache.org/nutch/`
- hadoop (2007). Apache hadoop. On-line: `http://lucene.apache.org/hadoop/`

Other relevant links

- http://wiki.apache.org/lucene-hadoop/HowToConfigure
- http://wiki.apache.org/nutch/NutchHadoopTutorial
- http://wiki.apache.org/lucene-hadoop/GettingStartedWithHadoop

Thank you.