cloudera

Practical MapReduce

Top ten tips

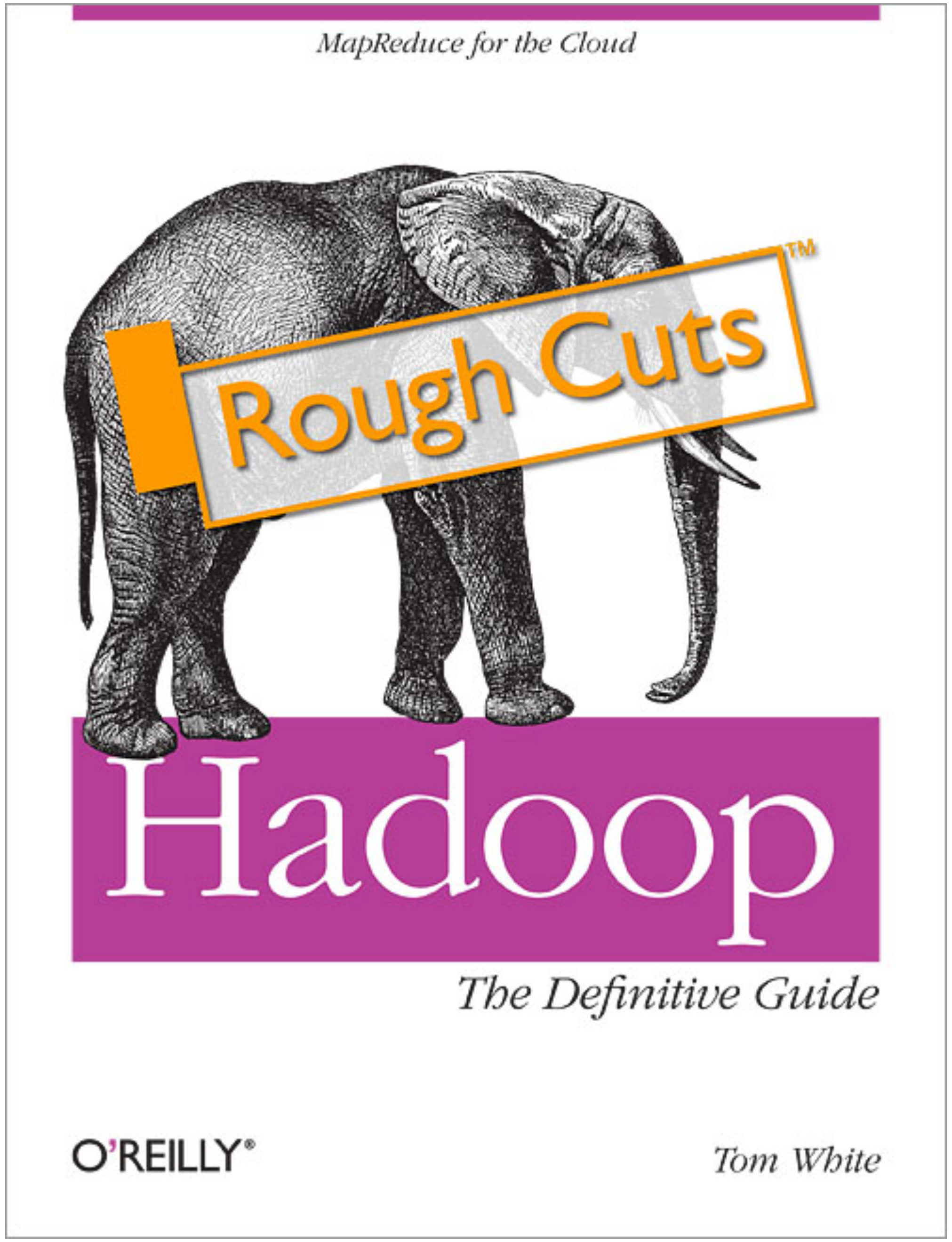
Tom White, Cloudera

Hadoop User Group UK, London

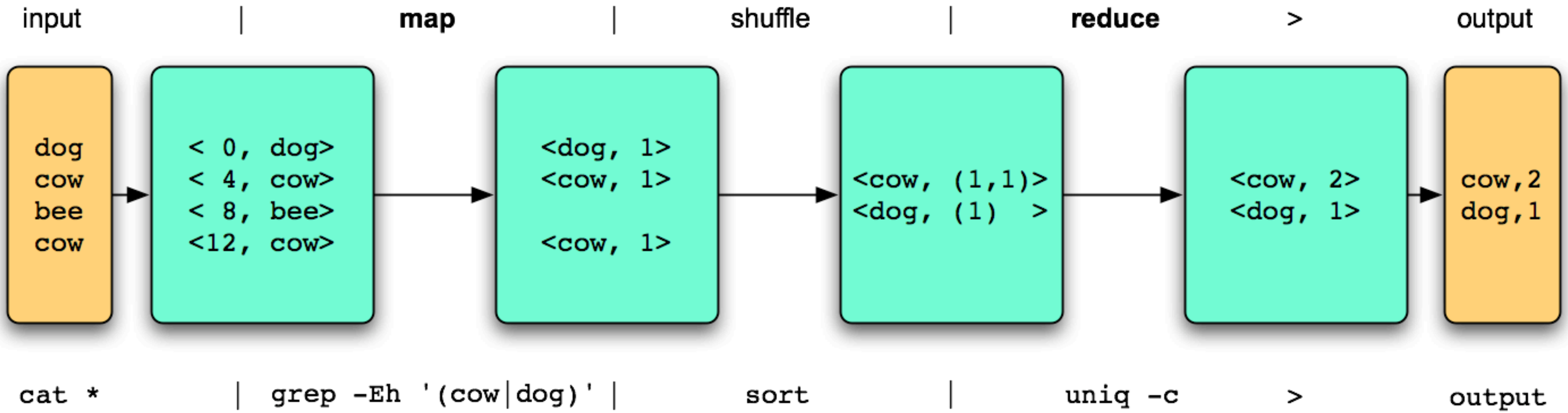
14 April 2009

About me

- Apache Hadoop Committer, PMC Member, Apache Member
- Employed by Cloudera
- Writing a book on Hadoop for O'Reilly
  - http://hadoopbook.com



What is MapReduce?



input | **map** | shuffle | **reduce** > output

```
dog
cow
bee
cow
```

```
< 0, dog>
< 4, cow>
< 8, bee>
<12, cow>
```

```
<dog, 1>
<cow, 1>

<cow, 1>
```

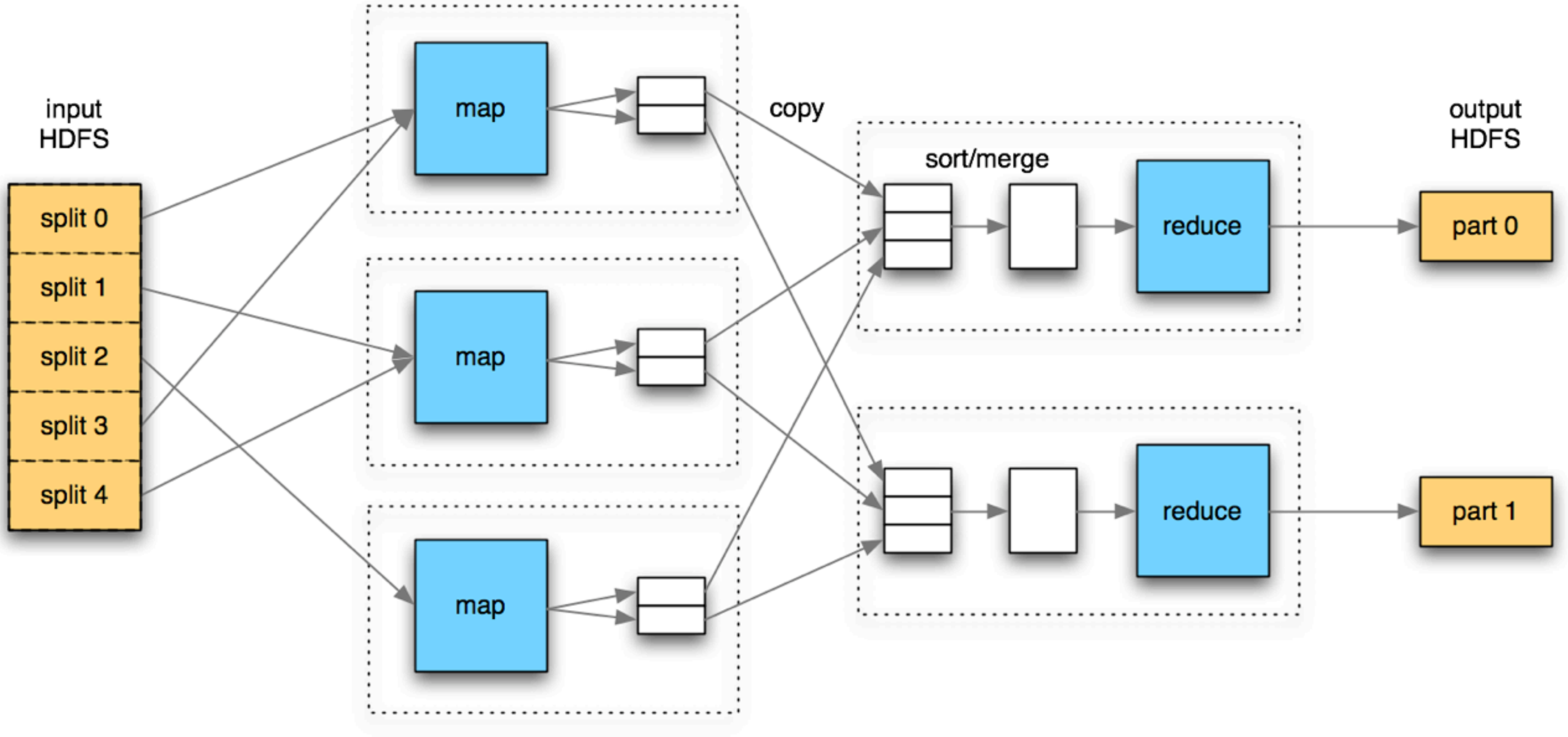
```
<cow, (1,1)>
<dog, (1)  >
```

```
<cow, 2>
<dog, 1>
```

```
cow,2
dog,1
```

```
cat *    | grep -Eh '(cow|dog)' |    sort    |    uniq -c    >    output
```

Another way of looking at it



Tips

1. Use the right MapReduce language

Structured	Pig, Hive
Dynamic	Streaming, Dumbo
System	Java

2. Consider your input data “chunk” size

- Hadoop dislikes lots of small files
  - Namenode eats memory
  - MapReduce produces too many trivial maps
- CombineFileInputFormat
  - Packs multiple files into one split
  - Considers locality
- Large unsplittable files aren’t great either
    - But see LzoTextInputFormat

3. Use SequenceFile and MapFile containers

- Splittable
- Compressed
  - Use block compression
- Compact

- But
  - Java-centric

4. Implement Tool

- Supports lots of options:
  - -D
  - -files
  - -archives
  - -libjars
- More testable
  - Inject arbitrary configurations

```
public class MyJob extends Configured implements Tool {

  public int run(String[] args) throws Exception {
      JobConf job = new JobConf(getConf(), MyJob.class);
      // run job ...
  }

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

5. Chain your jobs

- Don’t force the processing into one job
- Job overhead is coming down
- Scheduling is getting better
  - FairScheduler
- Take a leaf out of Hive and Pig’s book
  - EXPLAIN
- ChainMapper, ChainReducer
  - M*RM*
- JobControl

6. Multiple partitions good

- Move away from the idea of a single output file
  - Change default number of reducers
- Why do you need only one?
  - Next stage is probably another MapReduce anyway
  - Or can be digested in parallel
    - See MapFileOutputFormat.getReaders()
- Use a final MapReduce or a local script to combine partitions if needed
- hadoop fs -getmerge hdfs-output-dir local-file

7. Tell Hadoop your progress

- Periodically call
  - reporter.progress()
  - reporter.setStatus(“6 of 1 gazillion records processed”)
- Won’t kill your job, and get a nice message into the bargain
- If you can’t do this, then consider (but beware hanging processes)
  - -D mapred.task.timeout=0
- Watch the UI
  - “a watched hadoop job never completes”
    - http://twitter.com/sethladd/statuses/1499223543
  - Get to see where the slow bits are

8. Debug with status and counters

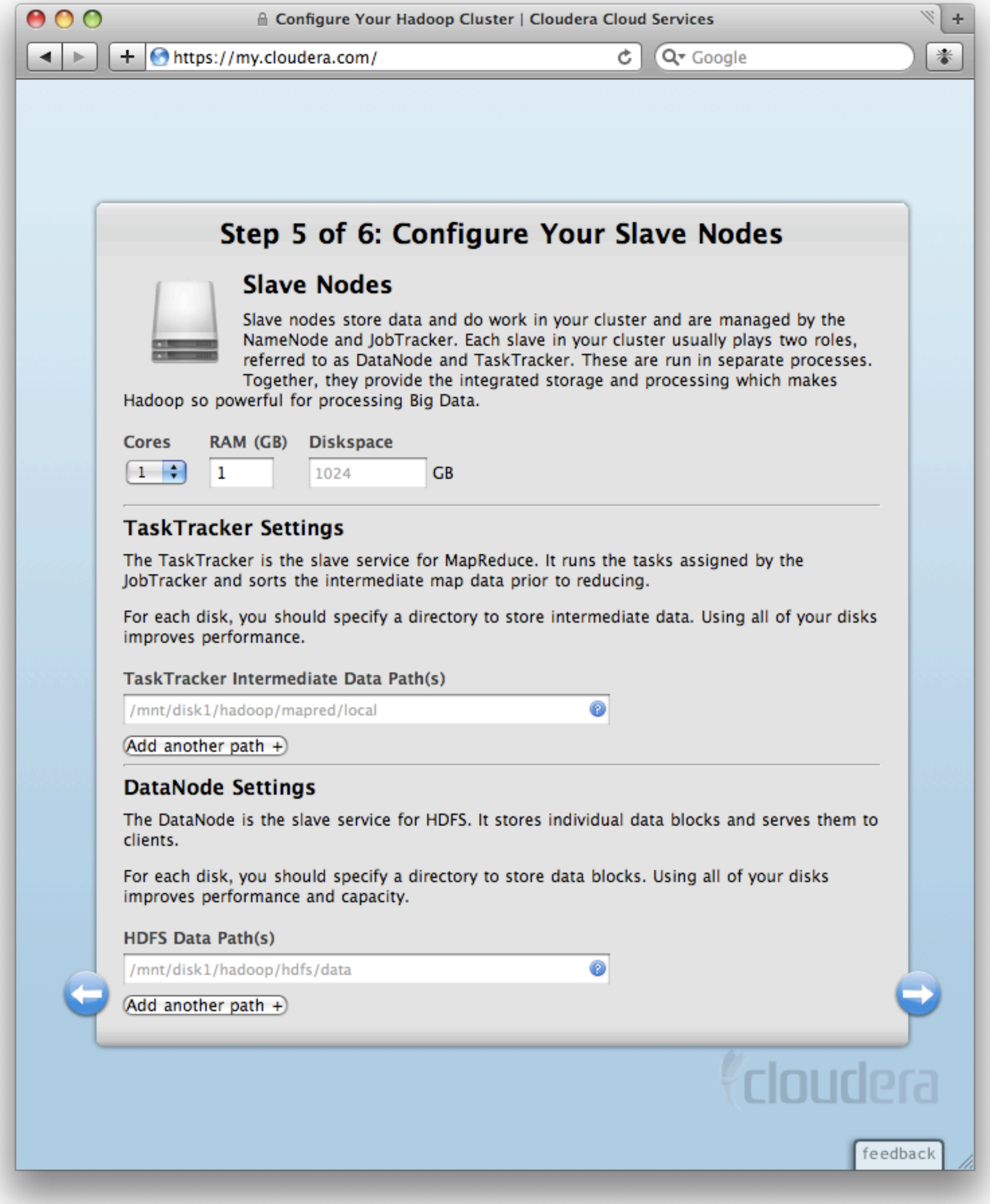
- Hadoop’s answer to System.err.println is
  - reporter.setStatus(“this record looks fishy”);
  - reporter.incrCounter(MyCounter.WEIRDNESS, 1);
  - System.err.println(“Some extra debug info...”);
- Or in Streaming
  - sys.stderr.write(“reporter:status:this record looks fishy\n”)
  - sys.stderr.write(“reporter:counter:MyCounter,WEIRDNESS,1\n”)
  - sys.stderr.write(“Some extra debug info...\n”);
- Then you can look in the web UI as the job is running for any problems, and click through to the logs.

9. Tune at the job level before the task level

- Usual suspects
  - # mappers, reducers
  - Combiner
  - Intermediate compression
  - Custom Writables
  - Shuffle tweaks
    - “If you’re storing stuff in a hash map, you’ve lost”
    - Lean on the framework, but give it the memory to do its job

10. Let someone else do the cluster admin

- Cloudera's Distribution for Hadoop
  - RPMs
  - Debs
  - Init scripts
  - Configuration web site
- Hadoop on Amazon EC2
  - Scripts from Apache, Cloudera
  - Elastic MapReduce



Configure Your Hadoop Cluster | Cloudera Cloud Services

https://my.cloudera.com/

Step 5 of 6: Configure Your Slave Nodes

Slave Nodes

Slave nodes store data and do work in your cluster and are managed by the NameNode and JobTracker. Each slave in your cluster usually plays two roles, referred to as DataNode and TaskTracker. These are run in separate processes. Together, they provide the integrated storage and processing which makes Hadoop so powerful for processing Big Data.

Cores: 1 RAM (GB): 1 Diskspace: 1024 GB

TaskTracker Settings

The TaskTracker is the slave service for MapReduce. It runs the tasks assigned by the JobTracker and sorts the intermediate map data prior to reducing.

For each disk, you should specify a directory to store intermediate data. Using all of your disks improves performance.

TaskTracker Intermediate Data Path(s): /mnt/disk1/hadoop/mapred/local

DataNode Settings

The DataNode is the slave service for HDFS. It stores individual data blocks and serves them to clients.

For each disk, you should specify a directory to store data blocks. Using all of your disks improves performance and capacity.

HDFS Data Path(s): /mnt/disk1/hadoop/hdfs/data

Questions?

- tom@cloudera.com

- Apache Hadoop
  - http://hadoop.apache.org
- Cloudera’s Distribution for Hadoop
  - http://www.cloudera.com/hadoop
- “Hadoop: The Definitive Guide”
  - http://hadoopbook.com

cloudera