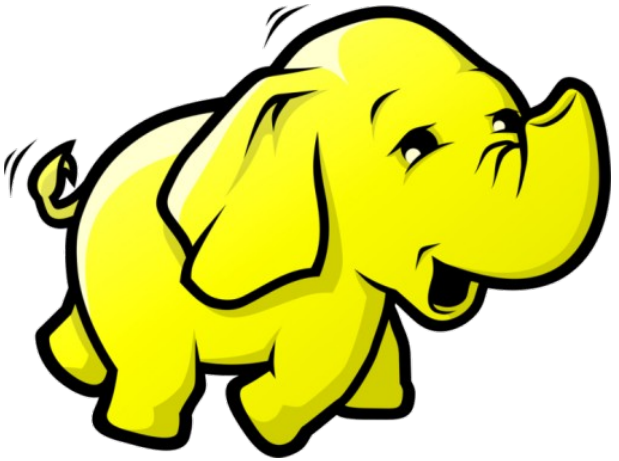
Hadoop and MapReduce



"The name my kid gave a stuffed yellow elephant.
Short, relatively easy to spell and pronounce,
meaningless, and not used elsewhere:
those are my naming criteria.
Kids are good at generating such.
Googol is a kid's term."

– Doug Cutting[1]

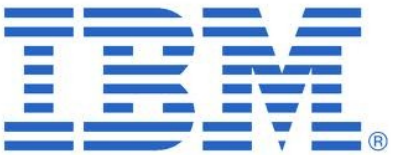
1 https://issues.apache.org/jira/browse/NUTCH-193

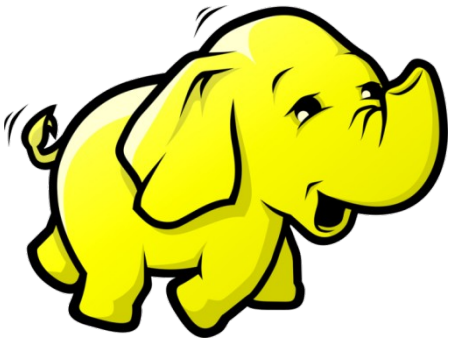
History

Dec 2004	MapReduce paper (http://labs.google.com/papers/mapreduce.html) published by Google. Doug Cutting and Mike Cafarella took the idea and they implemented it in Nutch.
Dec 2005	Hadoop came out of Nutch.
Jan 2006	Doug Cutting joins Yahoo!
Feb 2006	Apache Hadoop project started and it is used by Yahoo! Grid team.
Apr 2008	Hadoop wins the terabyte sort benchmark: 209 seconds on 900 nodes.
Apr 2009	Hadoop wins the minute sort by sorting 500 GB in 59 seconds on 1,400 nodes and 100 TB sort in 173 minutes on 3,400 nodes.

– Eric Baldeschwieler, OSCON, July 25 2007

Today





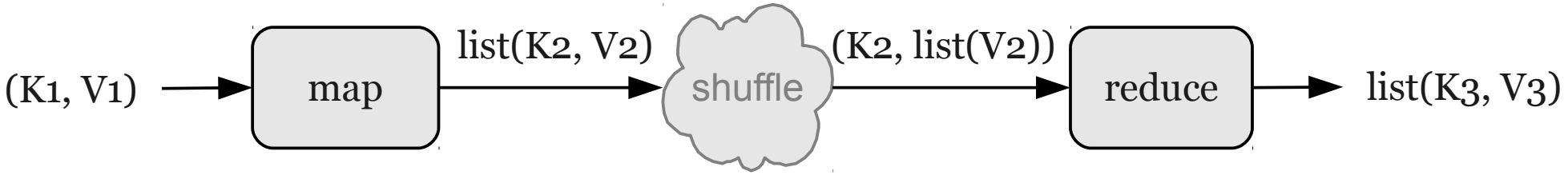
... and many others: http://wiki.apache.org/hadoop/PoweredBy

"Recent papers on parallel programming by researchers at Google and Microsoft present the results of using up to 1800 processors to perform computations accessing up to 10 terabytes of data. How can university researchers demonstrate the credibility of their work without having comparable computing facilities available?"

– Randal E. Bryant,
Dean of School of Computer Science
Carnegie Mellon University

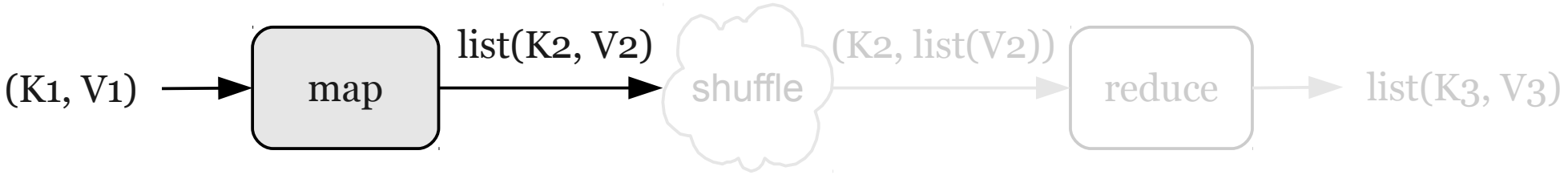
What's so special?

Easy to learn.



What's so special?

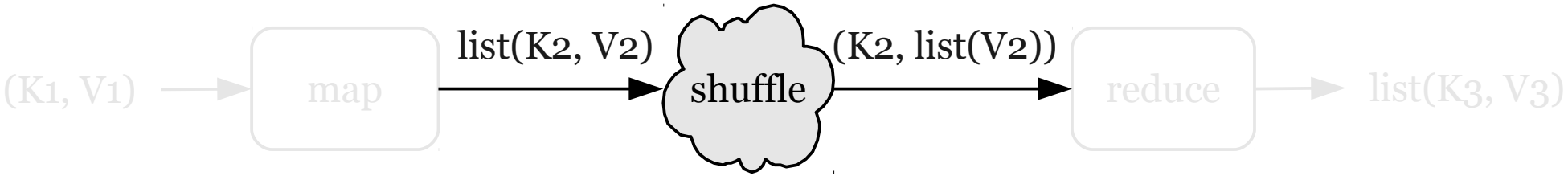
Easy to learn.



(K1, V1)	list(K2, V2)
(0, Lorem ipsum dolor sit ...)	(lorem, 1), (ipsum, 1), (dolor, 1), (sit, 1) ...
(57, Donec condimentum turpis ...)	(donec, 1), (condimentum, 1), (turpis, 1) ...
(131, Donec erat elit ...)	(donec, 1), (erat, 1), (elit, 1) ...
(193, Nulla facilisi. Quisque sit ...)	(nulla, 1), (facilisi, 1), (quisque, 1), (sit, 1) ...

What's so special?

Easy to learn.

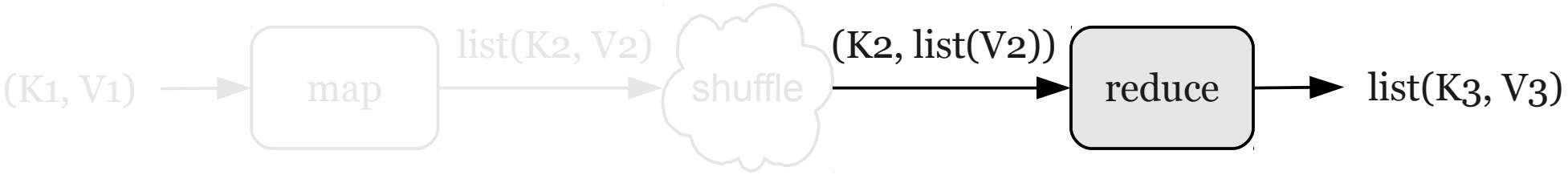


The MapReduce framework takes care of the shuffle phase.
Shuffle phase is where the "*magic*" happens:

- all the values with the same are delivered to the same reducer: hash(key) / #reducers
- keys are sorted: all reducers receive keys in sorted order

What's so special?

Easy to learn.



(K2, list(V2))	list(K2, V2)
(condimentum, (1))	(condimentum, 1)
(dolor, (1))	(dolor, 1)
(donec, (1, 1))	(donec, 1)
(elit, (1))	(elit, 1)
(erat, (1, 1, 1, 1, 1, 1, 1))	(erat, 7)
(facilisi, (1))	(facilisi, 1)
(lorem, (1, 1))	(lorem, 2)
(ipsum, (1, 1, 1))	(ipsum, 3)

What's so special?

Easy to use.

... and relatively easy to install and manage.

What's so special?

Hadoop Ecosystem

Hadoop Common	Components for distributed filesystems and I/O (serialization, Java RPC, persistent data structures).
Avro	Serialization framework, cross-language RPC and persistent data storage format.
MapReduce	Distributed and parallel processing model and execution environment.
HDFS	Distributed filesystem.
Pig	Data flow language (i.e. Pig Latin) which is compiled into MapReduce jobs.
Hive	Data warehouse with a SQL-like query language which is compiled into MapReduce jobs.
HBase	Distributed, column-oriented database.
ZooKeeper	Distributed, highly available coordination service.
Mahout	Scalable machine learning and data mining library.
Cascading	Data flow processing system.
Lucene and Sorl	Information retrieval systems whose indexes can be build using MapReduce.
Nutch	Web crawler which can use MapReduce to parallelize crawling and build indexes.
Sqoop	Tool to move data between relational databases and HDFS.
Tika	Tool for extracting metadata and structured text content from documents in verious formats.
UIMA	Natural language processing (NLP).
Oozie	Workflow system for MapReduce jobs.
Flume	Collecting and aggregating log data.
Hue	Browser-based desktop interface for interacting with Hadoop.

What's so special?

Fault tolerant: failed tasks are automatically re-executed, without interrupting a running job.

Scale linearly (for embarrassingly parallel problems).

Well tested in production with large clusters.

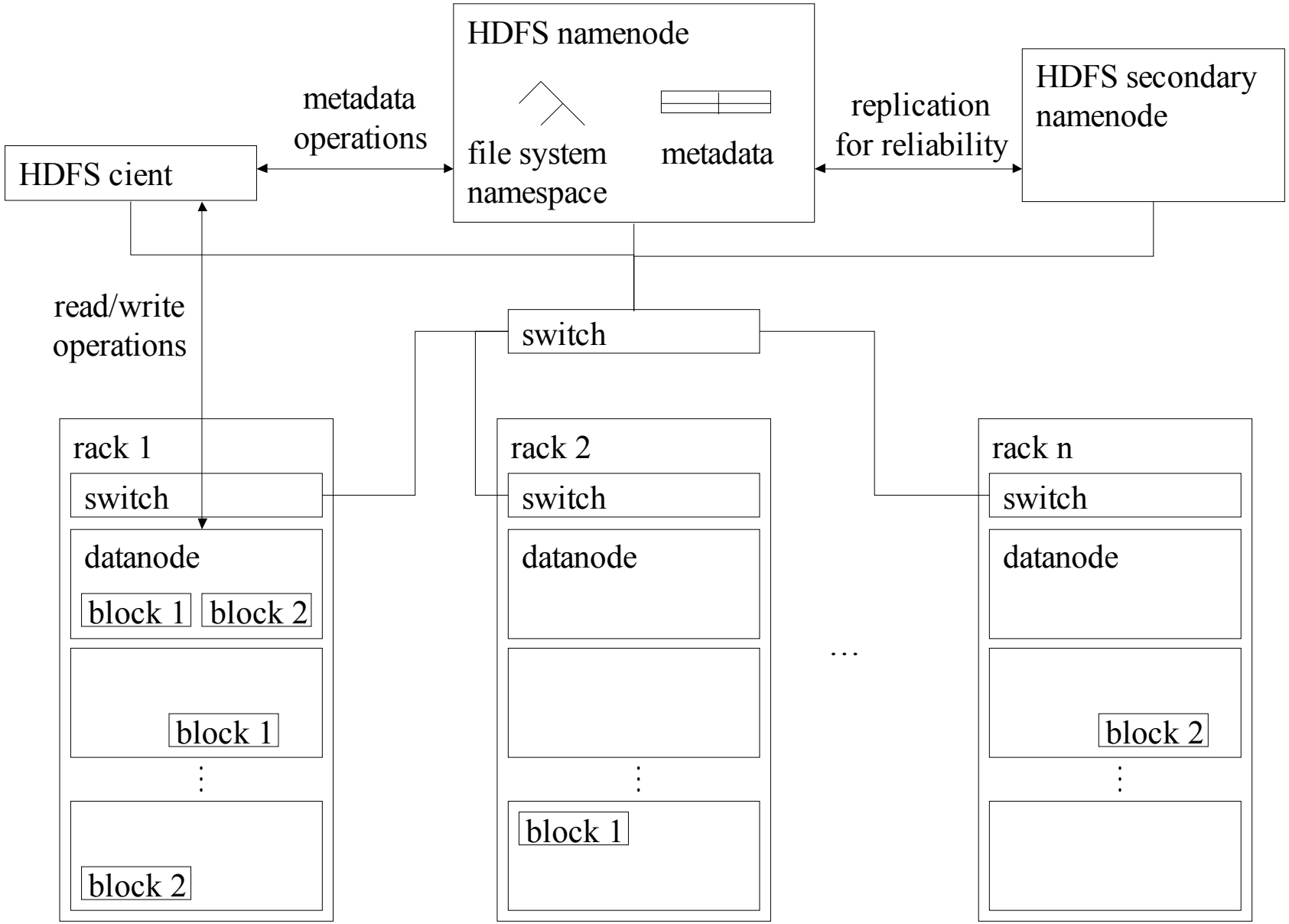
"The beauty of MapReduce if that any programmer can understand it, and its power comes from being able to harness thousands of computers behind that simple interface."

– David A. Patterson
"The Data Center Is The Computer"
Communications of the ACM
January 2008

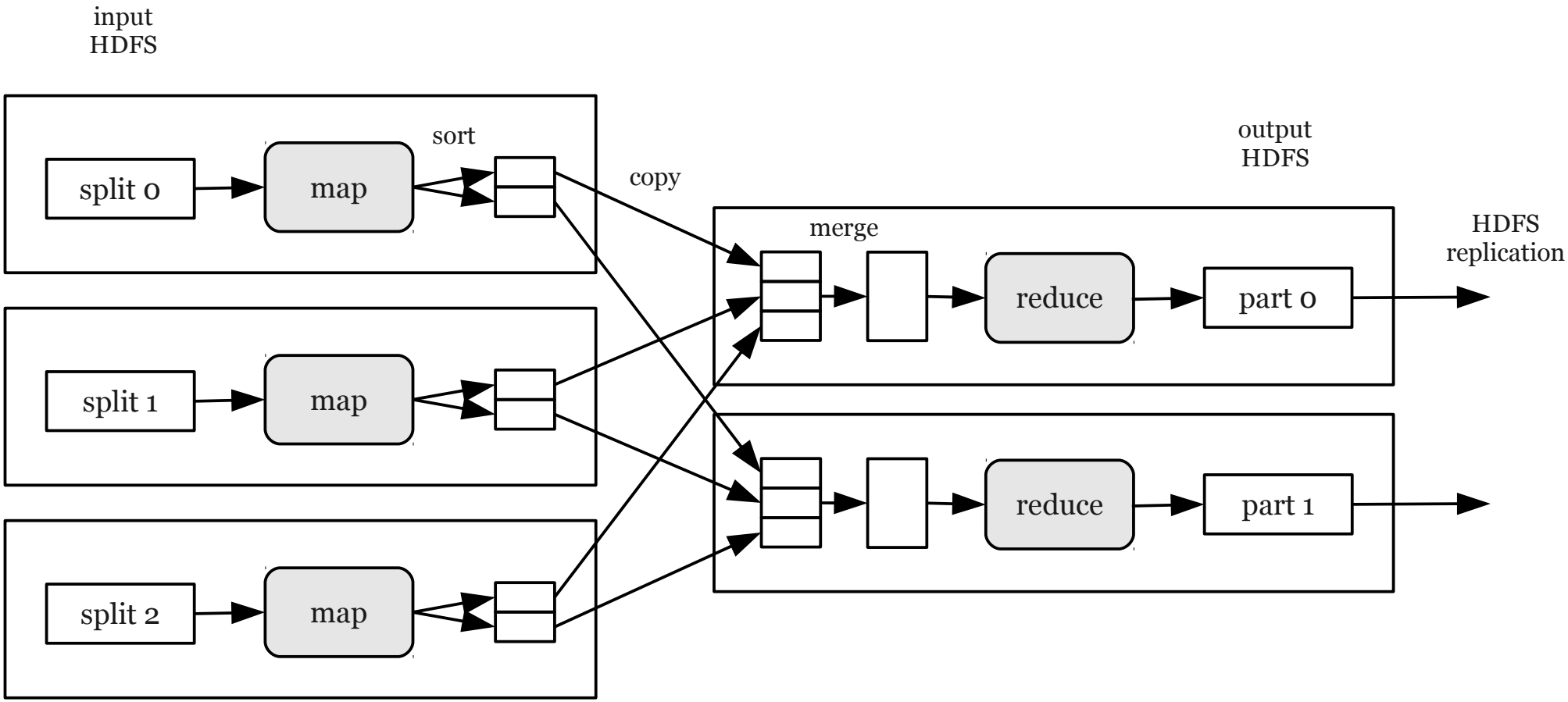
"It is tempting, if the only tool you have is a hammer,
to treat everything as if it were a nail."

– "The Psychology of Science: A Reconnaissance"
Abraham H. Maslow (1966)

HDFS Architecture



MapReduce Data Flow



What's not!

Hadoop is not a “classical” grid solution.

HDFS is not a POSIX file system.

HDFS is not designed for low latency access to a large number of small files.

MapReduce is not designed for interactive/real-time applications.

HBase is not a relational database and does not have transactions or SQL support.

HDFS and HBase are not focused on security, encryption or multitenancy. Although, Kerberos support has been recently added to HDFS.

Word Count

The “Hello World” in MapReduce

Word Count Example

Mapper

```
package com.talis.mapreduce.wordcount.newapi;

import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class WordCountMapper extends Mapper<LongWritable, Text, Text, IntWritable>
{
     private final static IntWritable one = new IntWritable(1);
     private Text word = new Text();

     @Override
     protected void map(LongWritable key, Text value, Context context)
          throws IOException ,InterruptedException
     {
          StringTokenizer iter = new StringTokenizer(value.toString(), " \t\n\r\f.,;:!?", false);
          while (iter.hasMoreTokens())
          {
               word.set(iter.nextToken().toLowerCase());
               Context.write(word, one);
          }
     }

}
```

Word Count Example

Reducer

```
package com.talis.mapreduce.wordcount.newapi;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class WordCountReducer extends Reducer<Text,IntWritable,Text,IntWritable>
{
     private IntWritable result = new IntWritable();

     @Override
     protected void reduce(Text key, Iterable<IntWritable> values, Context context)
          throws IOException , InterruptedException
     {
          int sum = 0;
          for (IntWritable value : values)
          {
               sum += value.get();
          }
          result.set(sum);
          context.write(key, result);
     }

}
```

Word Count Example

Driver

```
...

Job job = new Job(getConf(), getClass().getSimpleName());
job.setJarByClass(getClass());

job.setMapperClass(WordCountMapper.class);
job.setReducerClass(WordCountReducer.class);

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

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

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

Combiner and Partitioner

map: (K1, V1) → list(K2, V2)
combine: (K2, list(V2)) → list(K2, V2)
reduce: (K2, list(V2)) → list(K3, V3)

partition: (K2, V2) → integer

```
...

Job job = new Job(getConf(), getClass().getSimpleName());
job.setJarByClass(getClass());

job.setMapperClass(WordCountMapper.class);
job.setCombinerClass(WordCountReducer.class);
job.setReducerClass(WordCountReducer.class);

job.setPartitionerClass(HashPartitioner.class);

...
```

Old APIs

```
import org.apache.hadoop.mapred.*

extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable>

      map (LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter)
      throws IOException

extends MapReduceBase implements Reducer<Text,IntWritable,Text,IntWritable>

      reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter reporter)
      throws IOException
```

New APIs

```
import org.apache.hadoop.mapreduce.*

extends Mapper<LongWritable, Text, Text, IntWritable>

      map(LongWritable key, Text value, Context context)
      throws IOException ,InterruptedException

extends Reducer<Text,IntWritable,Text,IntWritable>

      reduce(Text key, Iterable<IntWritable> values, Context context)
      throws IOException ,InterruptedException
```

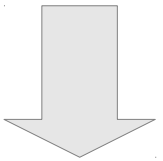
A Few Tips

- If you have many small files (i.e. less than an HDFS block: 64MB-128MB), use `CombineFileSplit` or group them into a `SequenceFile` or `MapFile`.
- Implement the `Tool` interface.
- If your task does complex computations, always (i.e. at least every 10 minutes) report progress using `progress()` or `setStatus(...)` methods from `Context`.
- If possible, use a combiner to reduce bandwidth utilization between map and reduce phases.
- If you use your custom `Writable` provide a `RawComparator` for it.
- ...

Dictionary Encoding

Dictionary Encoding

```
<http://example.org/alice/foaf.rdf#me> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person>
<http://example.org/alice/foaf.rdf> .
<http://example.org/alice/foaf.rdf#me> <http://xmlns.com/foaf/0.1/name> "Alice" <http://example.org/alice/foaf.rdf> .
<http://example.org/alice/foaf.rdf#me> <http://xmlns.com/foaf/0.1/knows> _:bnode1 <http://example.org/alice/foaf.rdf> .
_:bnode1 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person>
<http://example.org/alice/foaf.rdf> .
_:bnode1 <http://xmlns.com/foaf/0.1/name> "Bob" <http://example.org/alice/foaf.rdf> .
_:bnode1 <http://xmlns.com/foaf/0.1/homepage> <http://example.org/bob/> <http://example.org/alice/foaf.rdf> .
_:bnode1 <http://www.w3.org/2000/01/rdf-schema#seeAlso> <http://example.org/bob/foaf.rdf> <http://example.org/alice/foaf.rdf> .
<http://example.org/bob/foaf.rdf#me> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person>
<http://example.org/bob/foaf.rdf> .
<http://example.org/bob/foaf.rdf#me> <http://xmlns.com/foaf/0.1/name> "Bob" <http://example.org/bob/foaf.rdf> .
<http://example.org/bob/foaf.rdf#me> <http://xmlns.com/foaf/0.1/homepage> <http://example.org/bob/>
<http://example.org/bob/foaf.rdf> .
```



```
 0 "Alice"                                              14 7 9 3
 1 "Bob"                                                2 11 14 3
 2 <http://example.org/alice/foaf.rdf#me>               14 13 1 3
 3 <http://example.org/alice/foaf.rdf>                  2 13 0 3
 4 <http://example.org/bob/>                            5 7 9 6
 5 <http://example.org/bob/foaf.rdf#me>                 5 13 1 6
 6 <http://example.org/bob/foaf.rdf>                    5 10 4 6
 7 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>    14 8 6 3
 8 <http://www.w3.org/2000/01/rdf-schema#seeAlso>       14 10 4 3
 9 <http://xmlns.com/foaf/0.1/Person>                   2 7 9 3
10 <http://xmlns.com/foaf/0.1/homepage>
11 <http://xmlns.com/foaf/0.1/knows>
12 <http://xmlns.com/foaf/0.1/mbox>
13 <http://xmlns.com/foaf/0.1/name>
14 _:bnode1
```

Dictionary Encoding

1st MapReduce job

- map function:
  - input:
    - key: …, value: (g,s,p,o)
  - output:
    - foreach r in { g,s,p,o } key: r, value: hash(quad) + position
- reduce function:
  - output:
    - key: id(r), value: hash(quad) + position

Dictionary Encoding

2nd MapReduce job

- map function:
  - input:
    - key: id(r), value: hash(quad) + position
  - output:
    - key: hash(quad), value: id(r) + position
- reduce function:
  - output:
    - key: …, value: (id(r1), id(r2), id(r3), id(r4))

Dictionary Encoding

- Jacopo Urbani
  "*RDFS/OWL reasoning using the MapReduce framework*"
  July 2009, Master thesis, Vrije Universiteit
  http://www.few.vu.nl/~jui200/thesis.pdf
    - Section 4.4 "Dictionary encoding using only MapReduce"
      (pagg. 30-34)

- Jacopo Urbani, Jason Maassen, Henri Bal
  "*Massive Semantic Web data compression with MapReduce*"
  In Proceedings of the MapReduce workshop at High Performance Distributed Computing (HPDC) 2010
  http://www.few.vu.nl/~jui200/papers/HPDC10-mapreduce.pdf

“They (Google) did MapReduce;
we have this thing called Dryad that's better,
but they'll do one that's better.”

– Bill Gates, Microsoft
The New York Times, July 3, 2006

PageRank

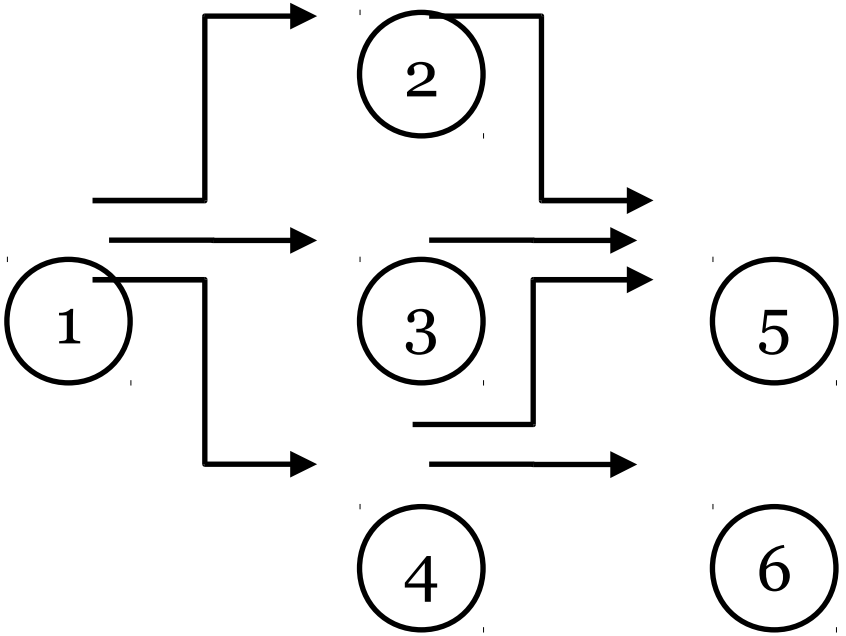
PageRank

$$r_{k+1}(p_i) = d\left(\sum_{\substack{p_j \in B_{p_i} \\ |p_j| \neq 0}} \frac{r_k(p_j)}{|p_j|} + \sum_{\substack{p_j \\ |p_j| = 0}} \frac{r_k(p_j)}{N} \right) + \frac{(1-d)}{N}$$

$$r_0(p_i) = \frac{1}{N}$$

Adjacency List

better for sparse matrices



1	2, 3, 4
2	5
3	5
4	5, 6
5	
6	

Adjacency List

1 r1 2 3 4

2 r2 5

3 r3 5

4 r4 5 6

5 r5

6 r6

PageRank with MapReduce

job 1 – total number of pages

for max n iterations or until convergence

job 2 – contribution from dangling pages

job 3 – page ranks from backward links

every y iterations

- job 4 – check for convergence

Total number of jobs <= $1 + 2n + n/y$

MapReduce

job 1 – total number of pages

map

input

- key = p value = ...

output

- key = 1 value = 1

combine and reduce

input

- key = 1 values = (vj)*

output

- key = ... value = N

$$N = \sum_j v_j$$

MapReduce

job 2 – contribution from dangling pages

map

input

- key = *p* value = *rp* *dangling page*

output

- key = 1 value = *rp /N* *N* total number of pages

combine and reduce

input

- key = 1 values = (*rj*)*

$$r_d = \frac{d}{N} \sum_j r_j$$

output

- key = … value = *rd* only one value

MapReduce

job 3 – page ranks from backward links

map

input

- key = *p* value = (*rp, p1, p2, … , pn*)

output

- key = *pi* value = *rp /n* *i* = (*1, 2, … , n*)
- key = *p* value = (*p1, p2, … , pn*)

reduce

input

- key = *p* values = (*rj /nj*)*, (*p1, p2, … , pn*)

output

- key = *p* value = (*rp, p1, p2, … , pn*)

$$r_p = d\sum_j \frac{r_j}{n_j} + r_d + \frac{(1-d)}{N}$$

MapReduce

job 4 – check for convergence

map

input

- key = *p* value = (*rk+1p, rkp, p1, p2, … , pn*)

output

- key = 1 value = abs (*rk+1p* - *rkp*)

combine and reduce

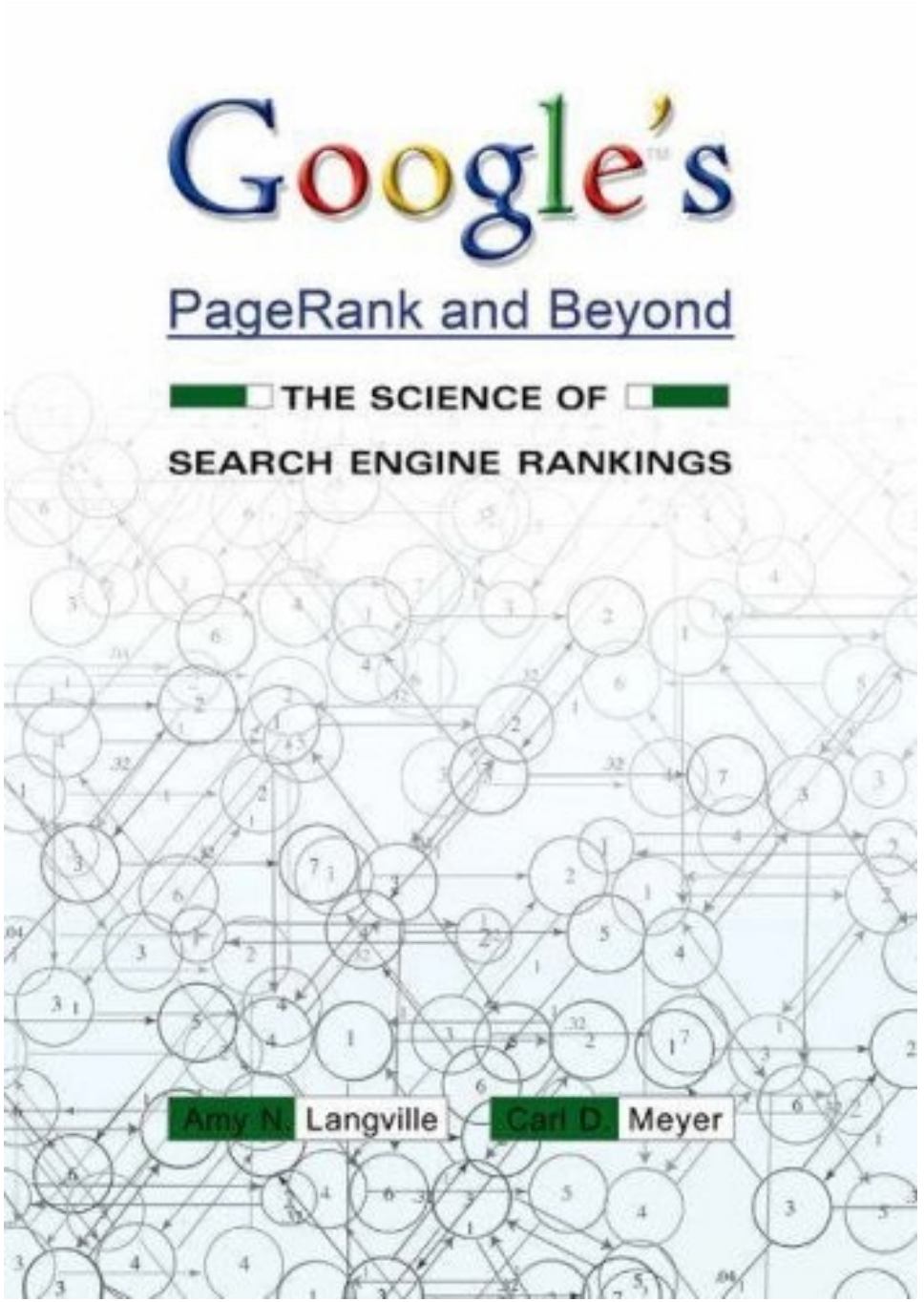
input

- key = 1 values = (*vj*)*

$$\varepsilon = \sum_j v_j$$

output

- key = … value = ε ε *tolerance*


Google's
PageRank and Beyond
THE SCIENCE OF
SEARCH ENGINE RANKINGS
Amy N. Langville
Carl D. Meyer

“Pregel computes over large graphs
much faster than alternatives,
and the application programming interface is easy to use.
Implementing PageRank, for example,
takes only about 15 lines of code.”

– Grzegorz Czajkowski
Official Google Research Blog
June 15, 2009

Matrix Multiplication

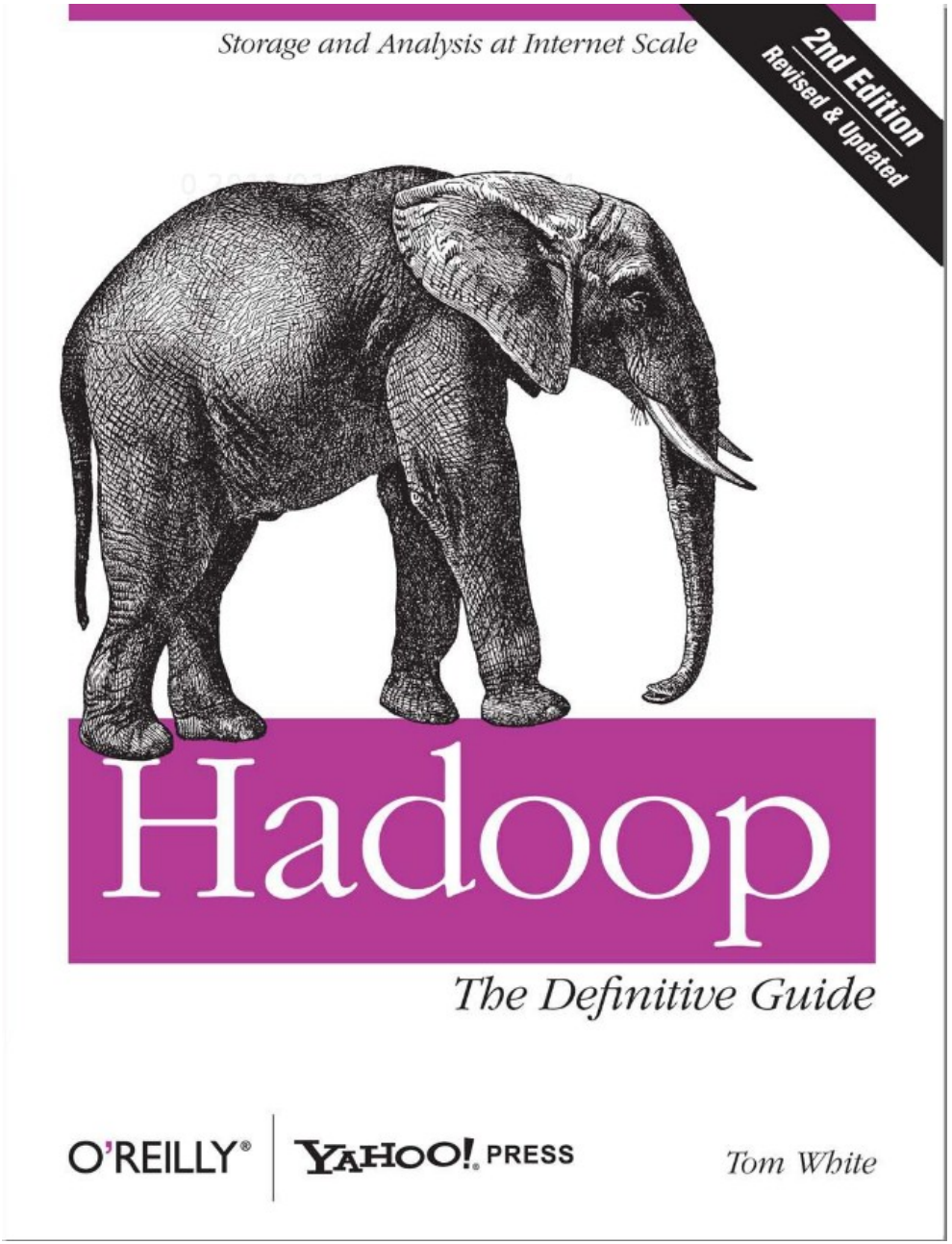
Matrix Multiplication

- John Norstad
  "*A MapReduce Algorithm for Matrix Multiplication*"
  Northwestern University
  http://homepage.mac.com/j.norstad/matrix-multiply/

Exercise left to the reader :-)

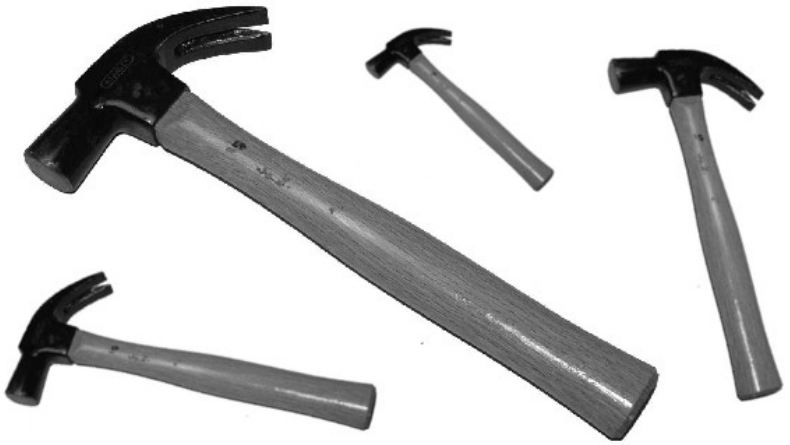
MapReduce @ Talis

- Ingestion Pipeline:
  - Data deduplication/validation
  - Lucene and Solr indexes
  - RDF indexes (i.e. B+Trees) (not parallel yet)
- Geo location and Linked Data:
  - Linking nearby “things”
- Recommendation algorithms
- …


Storage and Analysis at Internet Scale
2nd Edition
Revised & Updated
Hadoop
The Definitive Guide
O'REILLY®
YAHOO! PRESS
Tom White

More Hammers...

- Message Passing Interface (MPI)
- Compute Unified Device Architecture (CUDA) by NVIDIA
  - OpenCL (Open Computing Language)
- More RAM + compression techniques + clever data structures + efficient algorithms
- Stream processing
  - S4: Distributed Stream Computing Platform: http://s4.io/
- ...



shared innovation™

We are hiring!

http://www.talis.com/careers/