Counting at Scale

APAM E4990
Modeling Social Data

Jake Hofman

Columbia University

February 10, 2017

Previously

Claim:

Solving the counting problem at scale enables you to investigate many interesting questions in the social sciences

Learning to count

Last week:

Counting at small/medium scales on a single machine

Learning to count

Last week:

Counting at small/medium scales on a single machine

This week:

Counting at large scales in parallel

What?

What?

"... to create building blocks for programmers who just happen to have **lots of data to store**, **lots of data to analyze**, *or* **lots of machines to coordinate**, *and who don't have the time, the skill, or the inclination to become distributed systems experts to build the infrastructure to handle it."*

-Tom White
Hadoop: The Definitive Guide

What?

Hadoop contains many subprojects:

- **Hadoop Common**: The common utilities that support the other Hadoop subprojects.
- **Chukwa**: A data collection system for managing large distributed systems.
- **HBase**: A scalable, distributed database that supports structured data storage for large tables.
- **HDFS**: A distributed file system that provides high throughput access to application data.
- **Hive**: A data warehouse infrastructure that provides data summarization and ad hoc querying.
- **MapReduce**: A software framework for distributed processing of large data sets on compute clusters.
- **Pig**: A high-level data-flow language and execution framework for parallel computation.
- **ZooKeeper**: A high-performance coordination service for distributed applications.

We'll focus on distributed computation with MapReduce.

Who/when?

An overly brief history

Who/when?

pre-2004

Doug Cutting and Mike Cafarella develop open source projects for web-scale indexing, crawling, and search

Who/when?

2004

Dean and Ghemawat publish MapReduce programming model, used internally at Google

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

Abstract

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a *map* function that processes a key/value pair to generate a set of intermediate key/value pairs, and a *reduce* function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown in the paper.

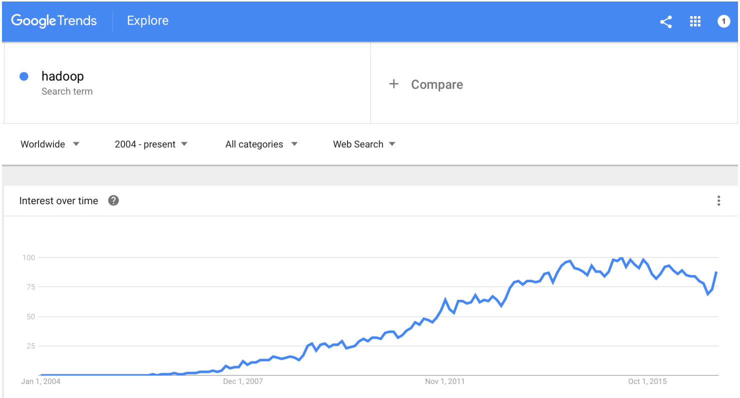
given day, etc. Most such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across hundreds or thousands of machines in order to finish in a reasonable amount of time. The issues of how to parallelize the computation, distribute the data, and handle failures conspire to obscure the original simple computation with large amounts of complex code to deal with these issues.

Who/when?

2006

Hadoop becomes official Apache project, Cutting joins Yahoo!, Yahoo adopts Hadoop

Who/when?



Google Trends
Explore
hadoop
Search term
Compare
Worldwide
2004 - present
All categories
Web Search
Interest over time
100
75
50
25
Jan 1, 2004
Dec 1, 2007
Nov 1, 2011
Oct 1, 2015

Where?

amazon.com
facebook
Google
IBM
JPMORGAN CHASE & CO.
LinkedIn
last.fm
The New York Times
VISA
twitter
YAHOO!

http://wiki.apache.org/hadoop/PoweredBy

Why?

Why *yet another* solution?

(I already use too many languages/environments)

Why?

Why a *distributed* solution?

(My desktop has TBs of storage and GBs of memory)

Why?

Roughly how long to read 1TB from a commodity hard disk?

Why?

Roughly how long to read 1TB from a commodity hard disk?

$$\frac{1}{2}\frac{\text{Gb}}{\text{sec}} \times \frac{1}{8}\frac{\text{B}}{\text{b}} \times 3600\frac{\text{sec}}{\text{hr}} \approx 225\frac{\text{GB}}{\text{hr}}$$

Why?

Roughly how long to read 1TB from a commodity hard disk?[1]

$\approx$ 4hrs

[1] SSDs provide a $\sim$10x speedup

Why?

MAY 11, 2009

Hadoop Sorts a Petabyte in 16.25 Hours and a Terabyte in 62 Seconds

We used **Apache Hadoop** to compete in **Jim Gray's Sort** benchmark. Jim's Gray's sort benchmark consists of a set of many related benchmarks, each with their own rules. All of the sort benchmarks measure the time to sort different numbers of 100 byte records. The first 10 bytes of each record is the key and the rest is the value. The **minute sort** must finish end to end in less than a minute. The **Gray sort** must sort more than 100 terabytes and must run for at least an hour. The best times we observed were:

Bytes	Nodes	Maps	Reduces	Replication	Time
500,000,000,000	1406	8000	2600	1	59 seconds
1,000,000,000,000	1460	8000	2700	1	62 seconds
100,000,000,000,000	3452	190,000	10,000	2	173 minutes
1,000,000,000,000,000	3658	80,000	20,000	2	975 minutes

`http://bit.ly/petabytesort`

Typical scenario

Store, parse, and analyze high-volume server logs,

```
[16/May/2010:07:28:49 -0400] "GET /autonomous_css/style.css HTTP/1.1" 200 2806 "http://www.jakehofman.com/" "Mozilla/4.0 (compatible; MSIE 8.0; Wi
ndows NT 6.1; WOW64; Trident/4.0; GTB6.4; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; OfficeLiveConnec
tor.1.4; OfficeLivePatch.1.3)"
[16/May/2010:07:28:49 -0400] "GET /cleanlooks/test3.gif HTTP/1.1" 404 339 "http://www.jakehofman.com/" "Mozilla/4.0 (compatible; MSIE 8.0; Windows
 NT 6.1; WOW64; Trident/4.0; GTB6.4; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; OfficeLiveConnector.1
.4; OfficeLivePatch.1.3)"
[16/May/2010:07:28:49 -0400] "GET / HTTP/1.1" 200 16458 "http://www.technologyreview.com/communications/25326/?a=f&utm_source=feedburner&utm_mediu
m=feed&utm_campaign=Feed%3A+YDNLinkBlog+%28Yahoo%21+Developer+Network+Linkblog%29" "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trid
ent/4.0; GTB6.4; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; OfficeLiveConnector.1.4; OfficeLivePatch.
1.3)"
[16/May/2010:07:28:50 -0400] "GET /favicon.ico HTTP/1.1" 404 330 "-" "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; GTB6.
4; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; OfficeLiveConnector.1.4; OfficeLivePatch.1.3)"
[16/May/2010:07:28:57 -0400] "GET /autonomous_css/style.css HTTP/1.1" 304 - "http://www.jakehofman.com/" "Mozilla/4.0 (compatible; MSIE 8.0; Windo
ws NT 6.1; WOW64; Trident/4.0; GTB6.4; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; OfficeLiveConnector
.1.4; OfficeLivePatch.1.3)"
[16/May/2010:07:28:57 -0400] "GET /cleanlooks/test3.gif HTTP/1.1" 404 339 "http://www.jakehofman.com/" "Mozilla/4.0 (compatible; MSIE 8.0; Windows
 NT 6.1; WOW64; Trident/4.0; GTB6.4; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; OfficeLiveConnector.1
.4; OfficeLivePatch.1.3)"
[16/May/2010:07:28:56 -0400] "GET / HTTP/1.1" 200 16458 "http://www.technologyreview.com/communications/25326/?a=f&utm_source=feedburner&utm_mediu
m=feed&utm_campaign=Feed%3A+YDNLinkBlog+%28Yahoo%21+Developer+Network+Linkblog%29" "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trid
ent/4.0; GTB6.4; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; OfficeLiveConnector.1.4; OfficeLivePatch.
1.3)"
```

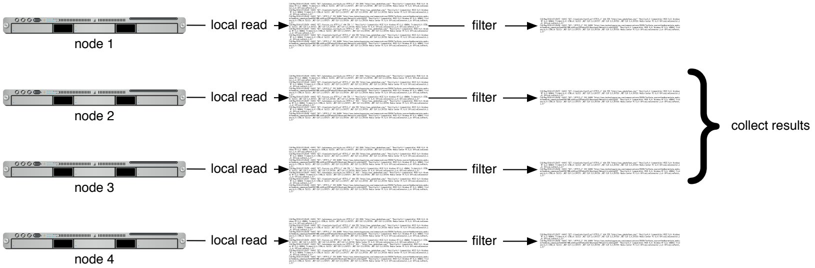
e.g. how many search queries match “icwsm”?

MapReduce: 30k ft

Break large problem into smaller parts, solve in parallel, combine results

Typical scenario

"Embarassingly parallel"
(or nearly so)

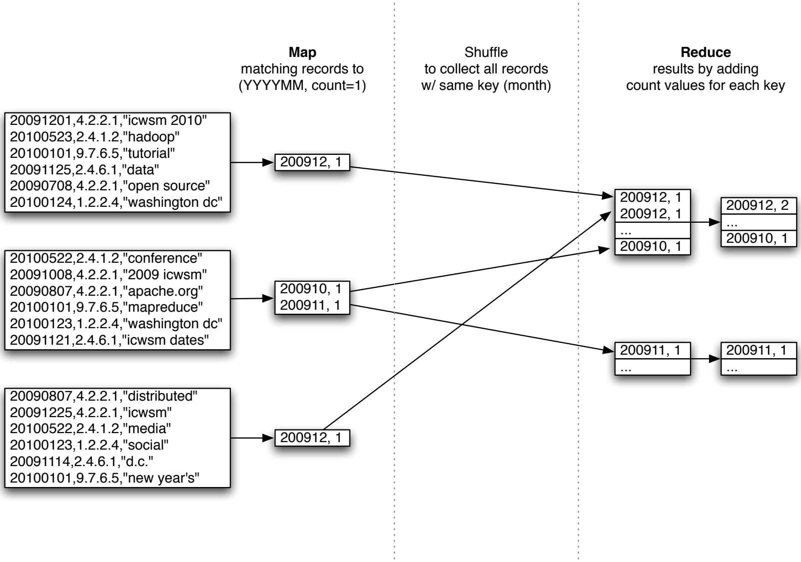


Typical scenario++

How many search queries match “icwsm”, grouped by month?

```
[16/May/2010:07:28:49 -0400] "GET /autonomous_css/style.css HTTP/1.1" 200 2806 "http://www.jakehofman.com/" "Mozilla/4.0 (compatible; MSIE 8.0; Wi
ndows NT 6.1; WOW64; Trident/4.0; GTB6.4; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; OfficeLiveConnec
tor.1.4; OfficeLivePatch.1.3)"
[16/May/2010:07:28:49 -0400] "GET /cleanlooks/test3.gif HTTP/1.1" 404 339 "http://www.jakehofman.com/" "Mozilla/4.0 (compatible; MSIE 8.0; Windows
 NT 6.1; WOW64; Trident/4.0; GTB6.4; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; OfficeLiveConnector.1
.4; OfficeLivePatch.1.3)"
[16/May/2010:07:28:49 -0400] "GET / HTTP/1.1" 200 16458 "http://www.technologyreview.com/communications/25326/?a=f&utm_source=feedburner&utm_mediu
m=feed&utm_campaign=Feed%3A+YDNLinkBlog+%28Yahoo%21+Developer+Network+Linkblog%29" "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trid
ent/4.0; GTB6.4; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; OfficeLiveConnector.1.4; OfficeLivePatch.
1.3)"
[16/May/2010:07:28:50 -0400] "GET /favicon.ico HTTP/1.1" 404 330 "-" "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; GTB6.
4; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; OfficeLiveConnector.1.4; OfficeLivePatch.1.3)"
[16/May/2010:07:28:57 -0400] "GET /autonomous_css/style.css HTTP/1.1" 304 - "http://www.jakehofman.com/" "Mozilla/4.0 (compatible; MSIE 8.0; Windo
ws NT 6.1; WOW64; Trident/4.0; GTB6.4; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; OfficeLiveConnector
.1.4; OfficeLivePatch.1.3)"
[16/May/2010:07:28:57 -0400] "GET /cleanlooks/test3.gif HTTP/1.1" 404 339 "http://www.jakehofman.com/" "Mozilla/4.0 (compatible; MSIE 8.0; Windows
 NT 6.1; WOW64; Trident/4.0; GTB6.4; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; OfficeLiveConnector.1
.4; OfficeLivePatch.1.3)"
[16/May/2010:07:28:56 -0400] "GET / HTTP/1.1" 200 16458 "http://www.technologyreview.com/communications/25326/?a=f&utm_source=feedburner&utm_mediu
m=feed&utm_campaign=Feed%3A+YDNLinkBlog+%28Yahoo%21+Developer+Network+Linkblog%29" "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trid
ent/4.0; GTB6.4; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; OfficeLiveConnector.1.4; OfficeLivePatch.
1.3)"
```

MapReduce: example



MapReduce: paradigm

Programmer specifies map and reduce functions

MapReduce: paradigm

Map: tranforms input record to intermediate (key, value) pair

```
def mapper(record):
    # input: a single record

    # parse / transform / filter record
    ...

    # output: intermediate key(s) and value(s)
    output( (key, value) )
```

MapReduce: paradigm

Shuffle: collects all intermediate records by key

Record assigned to reducers by `hash(key) % num_reducers`

Reducers perform a merge sort to collect records with same key

MapReduce: paradigm

Reduce: transforms all records for given key to final output

```
def reducer(key, records):
    # input: intermediate key and all values

    # initialize variables, e.g. counters

    for record in records:
        # parse record
        ...

        # update variables, e.g. count

    # output: final key(s) and value(s)
    output( (key, value) )
```

MapReduce: paradigm

Distributed read, shuffle, and write are transparent to programmer

MapReduce: principles

- Move code to data (local computation)
- Allow programs to scale transparently w.r.t size of input
- Abstract away fault tolerance, synchronization, etc.

MapReduce: strengths

- Batch, offline jobs
- Write-once, read-many across full data set
- Usually, though not always, simple computations
- I/O bound by disk/network bandwidth

!MapReduce

What it's not:

- High-performance parallel computing, e.g. MPI
- Low-latency random access relational database
- Always the right solution

Word count

the quick brown fox
jumps over the lazy dog
who jumped over that
lazy dog -- the fox ?

→

dog	2
--	1
the	3
brown	1
fox	2
jumped	1
lazy	2
jumps	1
over	2
quick	1
that	1
who	1
?	1

Word count

Map: for each line, output each word and count (of 1)

```
the quick brown fox
--------------------------------
jumps over the lazy dog
--------------------------------
who jumped over that
--------------------------------
lazy dog -- the fox ?
```

```
the	1
quick	1
brown	1
fox	1
---------
jumps	1
over	1
the	1
lazy	1
dog	1
---------
who	1
jumped	1
over	1
---------
that	1
lazy	1
dog	1
--	1
the	1
fox	1
?	1
```

Word count

Shuffle: collect all records for each word

```
the quick brown fox
--------------------------------
jumps over the lazy dog
--------------------------------
who jumped over that
--------------------------------
lazy dog -- the fox ?
```

→

```
--      1
---------
?       1
---------
brown   1
---------
dog     1
dog     1
---------
fox     1
fox     1
---------
jumped  1
---------
jumps   1
---------
lazy    1
lazy    1
---------
over    1
over    1
---------
quick   1
---------
that    1
---------
the     1
the     1
the     1
---------
who     1
```

Word count

Reduce: add counts for each word

```
--      1
---------
?       1
---------
brown   1
---------
dog     1
dog     1
---------
fox     1
fox     1
---------
jumped  1
---------
jumps   1
---------
lazy    1
lazy    1
---------
over    1
over    1
---------
quick   1
---------
that    1
---------
the     1
the     1
the     1
---------
who     1
```

→

```
--      1
?       1
brown   1
dog     2
fox     2
jumped  1
jumps   1
lazy    2
over    2
quick   1
that    1
the     3
who     1
```

Word count

```
the quick brown fox
--------------------------------
jumps over the lazy dog
--------------------------------
who jumped over that
--------------------------------
lazy dog -- the fox ?
```

```
dog	1
dog	1
---------
--	1
---------
the	1
the	1
the	1
---------
brown	1
---------
fox	1
fox	1
---------
jumped	1
---------
lazy	1
lazy	1
---------
jumps	1
---------
over	1
over	1
---------
quick	1
---------
that	1
---------
?	1
---------
who	1
```

```
dog	2
--	1
the	3
brown	1
fox	2
jumped	1
lazy	2
jumps	1
over	2
quick	1
that	1
who	1
?	1
```

WordCount.java

```java
package org.myorg;

import java.io.IOException;
import java.util.*;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;

public class WordCount {

 public static class Map extends MapReduceBase implements 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, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
   String line = value.toString();
   StringTokenizer tokenizer = new StringTokenizer(line);
   while (tokenizer.hasMoreTokens()) {
    word.set(tokenizer.nextToken());
    output.collect(word, one);
   }
  }
 }

 public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {
  public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
   int sum = 0;
   while (values.hasNext()) {
    sum += values.next().get();
   }
   output.collect(key, new IntWritable(sum));
  }
 }

 public static void main(String[] args) throws Exception {
  JobConf conf = new JobConf(WordCount.class);
  conf.setJobName("wordcount");

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

  conf.setMapperClass(Map.class);
  conf.setCombinerClass(Reduce.class);
  conf.setReducerClass(Reduce.class);

  conf.setInputFormat(TextInputFormat.class);
  conf.setOutputFormat(TextOutputFormat.class);

  FileInputFormat.setInputPaths(conf, new Path(args[0]));
  FileOutputFormat.setOutputPath(conf, new Path(args[1]));

  JobClient.runJob(conf);
 }
}
```

Hadoop streaming

Hadoop streaming is a utility that comes with the Hadoop distribution. The utility allows you to create and run map/reduce jobs with any executable or script as the mapper and/or the reducer. For example:

```
$HADOOP_HOME/bin/hadoop  jar $HADOOP_HOME/hadoop-streaming.jar \
    -input myInputDirs \
    -output myOutputDir \
    -mapper /bin/cat \
    -reducer /bin/wc
```

Hadoop streaming

MapReduce for *nix geeks[2]:

```
# cat data | map | sort | reduce
```

Where the sort is a hack to approximate a distributed group-by:

- Mapper reads input data from stdin
- Mapper writes output to stdout
- Reducer receives input, grouped by key, on stdin
- Reducer writes output to stdout

[2] http://bit.ly/michaelnoll

wordcount.sh

Locally:

```
# cat data | tr " " "\n" | sort | uniq -c
```

wordcount.sh

Locally:

```
# cat data | tr " " "\n" | sort | uniq -c
```

$\Downarrow$

Distributed:

```
$HADOOP_HOME/bin/hadoop jar $HADOOP_HOME/hadoop-streaming.jar \
    -input README.txt \
    -output wordcount \
    -mapper 'tr " " "\n"' \
    -reducer 'uniq -c'
```

Transparent scaling

Use **the same code** on **MBs locally** or **TBs across thousands of machines**.

wordcount.py

```
from hstream import HStream
import sys
import re
from collections import defaultdict

class WordCount(HStream):

    def mapper(self, record):
        for word in " ".join(record).split():
            self.write_output((word, 1))

    def reducer(self, key, records):
        total = 0

        for record in records:
            word, count = record

            total += int(count)

        self.write_output( (word, total) )

if __name__ == '__main__':

    WordCount()
```

Higher level abstractions

Higher level languages like Pig or Hive provide robust implementations for many common MapReduce operations

(e.g., filter, sort, join, group by, etc.)

They also allow for user-defined map and reduce functions

Higher level abstractions

Pig looks like SQL, but is more procedural

```
input_lines = LOAD '/tmp/my-copy-of-all-pages-on-internet' AS (line:chararray);

-- Extract words from each line and put them into a pig bag
-- datatype, then flatten the bag to get one word on each row
words = FOREACH input_lines GENERATE FLATTEN(TOKENIZE(line)) AS word;

-- filter out any words that are just white spaces
filtered_words = FILTER words BY word MATCHES '\\w+';

-- create a group for each word
word_groups = GROUP filtered_words BY word;

-- count the entries in each group
word_count = FOREACH word_groups GENERATE COUNT(filtered_words) AS count, group AS word;

-- order the records by count
ordered_word_count = ORDER word_count BY count DESC;
STORE ordered_word_count INTO '/tmp/number-of-words-on-internet';
```

pig.apache.org

Higher level abstractions

Hive is closer to SQL, with nested declarative statements

```
DROP TABLE IF EXISTS docs;
CREATE TABLE docs (line STRING);
LOAD DATA INPATH 'input_file' OVERWRITE INTO TABLE docs;
CREATE TABLE word_counts AS
SELECT word, count(1) AS count FROM
 (SELECT explode(split(line, '\s')) AS word FROM docs) temp
GROUP BY word
ORDER BY word;
```

hive.apache.org