Large-scale social media analysis with Hadoop

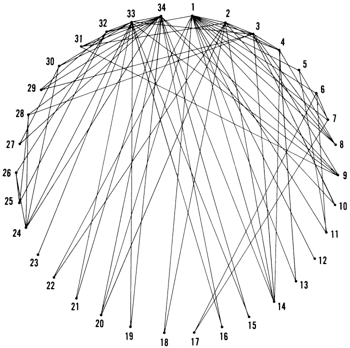
Jake Hofman

Yahoo! Research

May 23, 2010


digg
You Tube
Broadcast Yourself
upcoming.org
last.fm
the social music revolution
flickr
BETA
myspace.com
Technorati
twitter
facebook
MyBlogLog
slideshare
jaiku

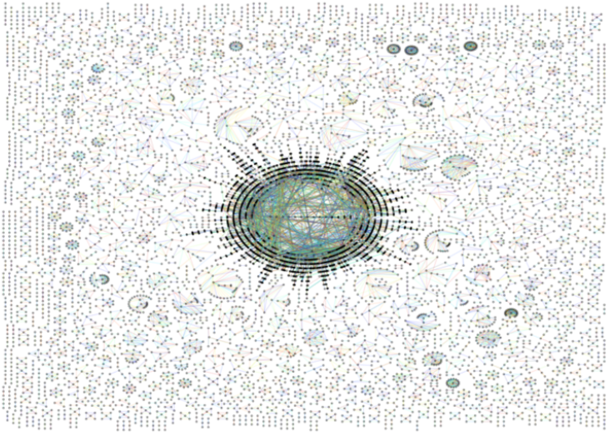
1970s $\sim 10^1$ nodes



The karate club was observed for a period of three years, from 1970 to 1972. In addition to direct observation, the history of the club prior to the period of the study was reconstructed through informants and club records in the university archives. During the period of observation, the

- Few direct observations; highly detailed info on nodes and edges
- E.g. karate club (Zachary, 1977)

1990s $\sim 10^4$ nodes



- Larger, indirect samples; relatively few details on nodes and edges
- E.g. APS co-authorship network (`http://bit.ly/aps08jmh`)

Present $\sim 10^7$ nodes $+$


```xml
<?xml version="1.0" encoding="UTF-8"?>
<user>
  <id>18805477</id>
  <name>jake hofman</name>
  <screen_name>jakehofman</screen_name>
  <location>new york, ny</location>
  <description>research scientist interested in machine learning for large data sets, including network, text and image data</description>
  <profile_image_url>http://a3.twimg.com/profile_images/71044209/jmh_dot_normal.jpg</profile_image_url>
  <url>http://www.jakehofman.com</url>
  <protected>false</protected>
  <followers_count>292</followers_count>
  <friends_count>208</friends_count>
  <created_at>Fri Jan 09 16:20:08 +0000 2009</created_at>
  <favourites_count>823</favourites_count>
  <statuses_count>628</statuses_count>
  ...
  <status>
    <created_at>Thu Oct 01 23:46:06 +0000 2009</created_at>
    <id>4538350570</id>
    <text>RT @atveit: RT @atbrox Mapreduce &amp; Hadoop Algorithms in Academic Papers - http://bit.ly/2rPgG #hadoopworld</text>
    <source>&lt;a href=&quot;http://www.atebits.com/&quot; rel=&quot;nofollow&quot;&gt;Tweetie&lt;/a&gt;</source>
    <truncated>false</truncated>
    <in_reply_to_status_id></in_reply_to_status_id>
    <in_reply_to_user_id></in_reply_to_user_id>
    <favorited>false</favorited>
    <in_reply_to_screen_name></in_reply_to_screen_name>
    <geo/>
  </status>
</user>
```


- Very large, dynamic samples; many details in node and edge metadata
- E.g. Mail, Messenger, Facebook, Twitter, etc.

If you had all of twitter's data, what question would you ask of it?

djw2451

If you had all of twitter's data, what question would you ask of it?

djw2451

What *could* you ask of it?

Look familiar?

```
# ls -talh neat_dataset.tar.gz
-rw-r--r--  100T May 23 13:00 neat_dataset.tar.gz
```

Look familiar?[1]

```
# ls -talh twitter_rv.tar
-rw-r--r--  24G May 23 13:00 twitter_rv.tar
```

[1] http://an.kaist.ac.kr/traces/WWW2010.html

Agenda

Large-scale social media analysis with Hadoop

Agenda

Large-scale social media analysis with Hadoop

GB/TB/PB-scale, 10,000+ nodes

Agenda

Large-scale social media analysis with Hadoop

network & text data

Agenda

Large-scale social media analysis with Hadoop

network analysis & machine learning

Agenda

Large-scale social media analysis with Hadoop

open source Apache project for distributed storage/computation

Warning

You may be bored if you already know how to ...

Warning

You may be bored if you already know how to ...

- Install and use Hadoop (on a single machine and EC2)

Warning

You may be bored if you already know how to ...

- Install and use Hadoop (on a single machine and EC2)
- Run jobs in local and distributed modes

Warning

You may be bored if you already know how to ...

- Install and use Hadoop (on a single machine and EC2)
- Run jobs in local and distributed modes
- Implement distributed solutions for:

Warning

You may be bored if you already know how to ...

- Install and use Hadoop (on a single machine and EC2)
- Run jobs in local and distributed modes
- Implement distributed solutions for:
    - Parsing and manipulating large text collections

Warning

You may be bored if you already know how to ...

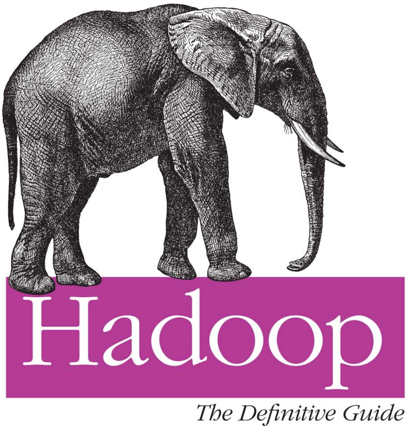
- Install and use Hadoop (on a single machine and EC2)
- Run jobs in local and distributed modes
- Implement distributed solutions for:
    - Parsing and manipulating large text collections
    - Clustering coefficient, BFS, etc., for networks w/ billions of edges

Warning

You may be bored if you already know how to ...

- Install and use Hadoop (on a single machine and EC2)
- Run jobs in local and distributed modes
- Implement distributed solutions for:
  - Parsing and manipulating large text collections
  - Clustering coefficient, BFS, etc., for networks w/ billions of edges
  - Classification, clustering

Selected resources



http://www.hadoopbook.com/

Selected resources

http://www.cloudera.com/

Selected resources

Data-Intensive Text Processing with MapReduce

Jimmy Lin and Chris Dyer
University of Maryland, College Park

http://www.umiacs.umd.edu/~jimmylin/book.html

Selected resources

... and many more at

`http://delicious.com/jhofman/hadoop`

Selected resources

... and **many** more at

`http://delicious.com/pskomoroch/hadoop`

Outline

1. Background (5 Ws)
2. Introduction to MapReduce (How, Part I)
3. Applications (How, Part II)

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

Cutting and 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.

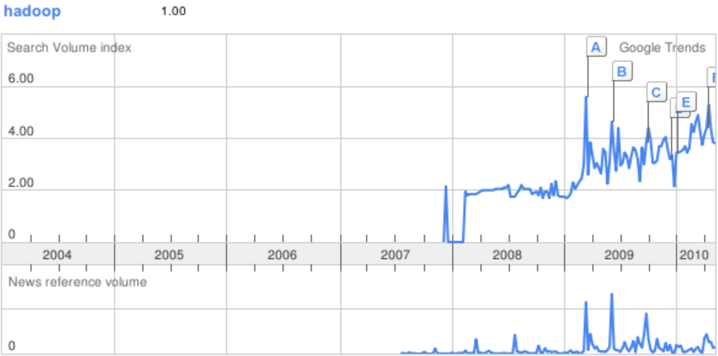
given day, etc. Most such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across hundreds or thousands of machines in order to finish in a reasonable amount of time. The issues of how to parallelize the computation, distribute 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?



hadoop
1.00
Search Volume index
Google Trends
6.00
4.00
2.00
0
A
B
C
E
2004
2005
2006
2007
2008
2009
2010
News reference volume
0

Where?

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?

$\approx$ 4hrs

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`

Outline

1. Background (5 Ws)
2. Introduction to MapReduce (How, Part I)
3. Applications (How, Part II)

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)"
```

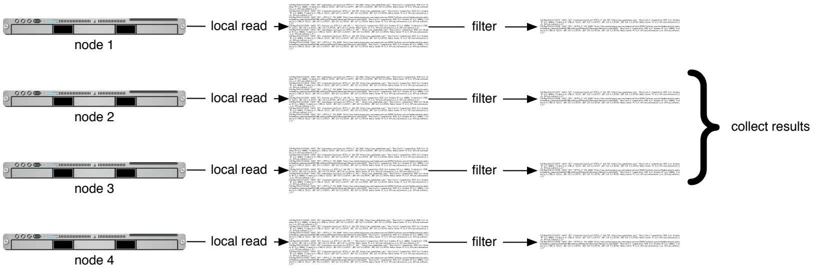
e.g. how many search queries match "icwsm"?

MapReduce: 30k ft

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

Typical scenario

"Embarassingly parallel"
(or nearly so)

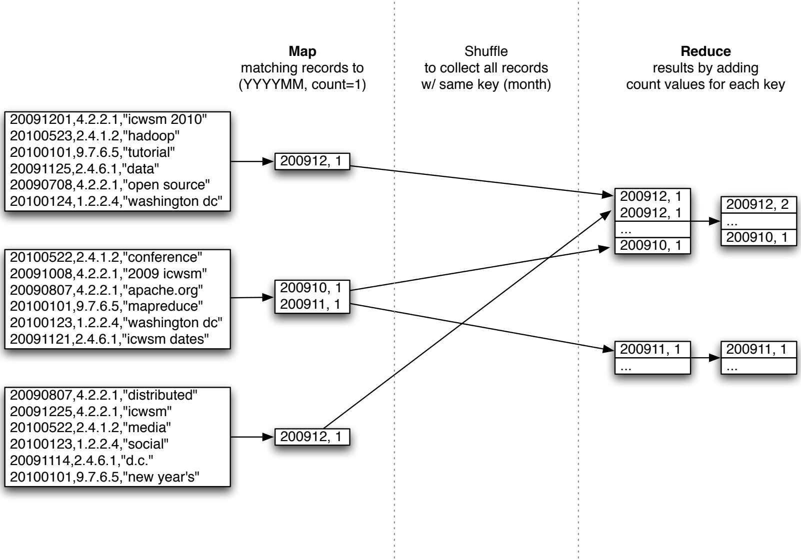


Typical scenario++

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

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
```

- Mapper reads input data from stdin
- Mapper writes output to stdout
- Reducer receives input, sorted 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()
```

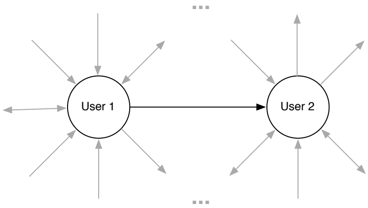
Outline

1. Background (5 Ws)
2. Introduction to MapReduce (How, Part I)
3. **Applications (How, Part II)**

Network data

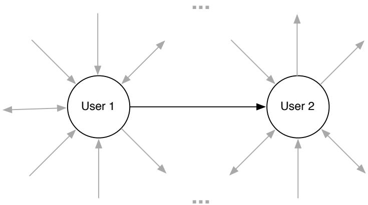
Scale

- Example numbers:
  - $\sim 10^7$ nodes
  - $\sim 10^2$ edges/node
  - no node/edge data
  - static
  - $\sim$8GB



Scale

- Example numbers:
    - $\sim 10^7$ nodes
    - $\sim 10^2$ edges/node
    - no node/edge data
    - static
    - $\sim$8GB



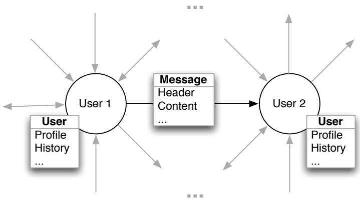
Simple, static networks push memory limit for commodity machines

Scale


```xml
<?xml version="1.0" encoding="UTF-8"?>
<user>
  <id>18805477</id>
  <name>jake hofman</name>
  <screen_name>jakehofman</screen_name>
  <location>new york, ny</location>
  <description>research scientist interested in machine learning for large data sets, including network, text and image data</description>
  <profile_image_url>http://a3.twimg.com/profile_images/71044209/jmh_dot_normal.jpg</profile_image_url>
  <url>http://www.jakehofman.com</url>
  <protected>false</protected>
  <followers_count>292</followers_count>
  <friends_count>208</friends_count>
  <created_at>Fri Jan 09 16:20:08 +0000 2009</created_at>
  <favourites_count>823</favourites_count>
  <statuses_count>628</statuses_count>
  ...
  <status>
    <created_at>Thu Oct 01 23:46:06 +0000 2009</created_at>
    <id>4538350570</id>
    <text>RT @atveit: RT @atbrox Mapreduce &amp; Hadoop Algorithms in Academic Papers - http://bit.ly/2rPgG #hadoopworld</text>
    <source>&lt;a href=&quot;http://www.atebits.com/&quot; rel=&quot;nofollow&quot;&gt;Tweetie&lt;/a&gt;</source>
    <truncated>false</truncated>
    <in_reply_to_status_id></in_reply_to_status_id>
    <in_reply_to_user_id></in_reply_to_user_id>
    <favorited>false</favorited>
    <in_reply_to_screen_name></in_reply_to_screen_name>
    <geo/>
  </status>
</user>
```

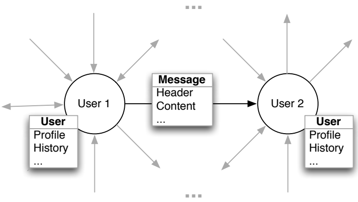
Scale

- Example numbers:
  - $\sim 10^7$ nodes
  - $\sim 10^2$ edges/node
  - node/edge metadata
  - dynamic
  - $\sim$100GB/day



Scale

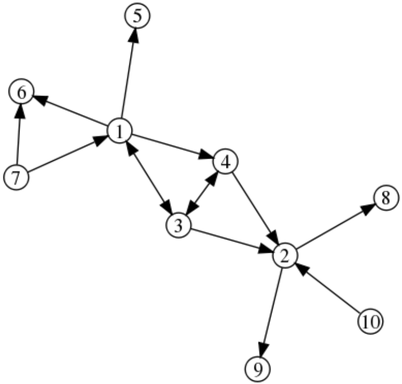
- Example numbers:
  - $\sim 10^7$ nodes
  - $\sim 10^2$ edges/node
  - node/edge metadata
  - dynamic
  - $\sim$100GB/day



Dynamic, data-rich social networks exceed memory limits; require considerable storage

Assumptions

Look only at topology, ignoring node and edge metadata



Assumptions

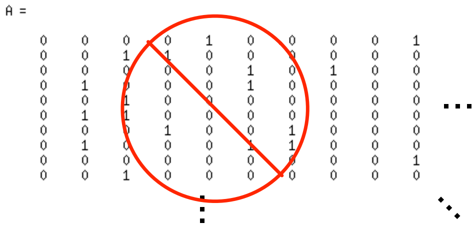
Full network exceeds memory of single machine

A =

$$
\begin{array}{cccccccccc}
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & \vdots & & & & & & \ddots
\end{array}
$$

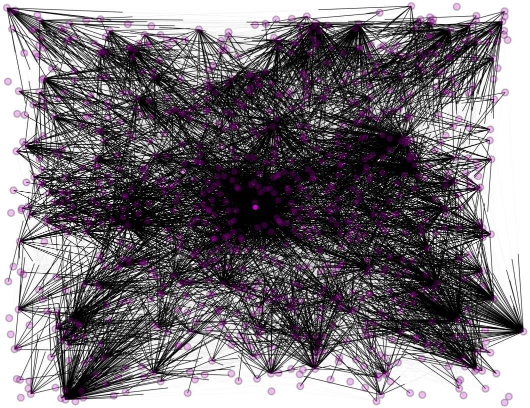
Assumptions

Full network exceeds memory of single machine



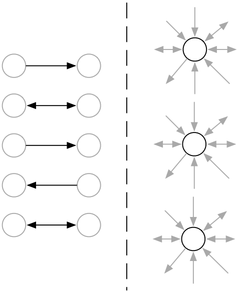
Assumptions

First-hop neighborhood of any individual node fits in memory



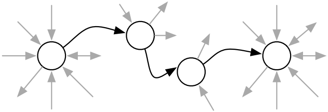
Distributed network analysis

MapReduce convenient for parallelizing individual node/edge-level calculations



Distributed network analysis

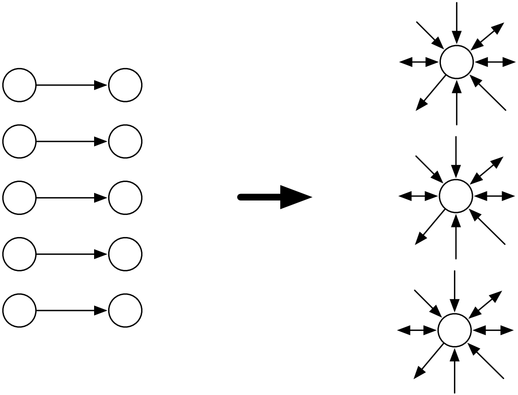
Higher-order calculations more difficult , but can be adapted to MapReduce framework



Distributed network analysis

- Network creation/manipulation
  - Logs $\rightarrow$ edges
  - Edge list $\leftrightarrow$ adjacency list
  - Directed $\leftrightarrow$ undirected
  - Edge thresholds
- First-order descriptive statistics
  - Number of nodes
  - Number of edges
  - Node degrees
- Higher-order node-level descriptive statistics
  - Clustering coefficient
  - Implicit degree
  - ...
- Global calculations
  - Pairwise connectivity
  - Connected components
  - Minimum spanning tree
  - Breadth-first search
  - Pagerank
  - Community detection

Edge list → adjacency list



Edge list → adjacency list

source	target
1	4
1	5
1	6
7	6
7	1
3	1
1	3
4	2
3	2
2	8
10	2
2	9
3	4
4	3

→

node	in/out neighbors	
1	3 7	3 5 4 6
10		2
2	10 3 4	9 8
3	1 4	1 2 4
4	1 3	3 2
5	1	
6	1 7	
7		1 6
8	2	
9	2	

Edge list → adjacency list

Map: for each (source, target), output (source, →, target) & (target, ←, source)

source	target
1	4
1	5
1	6
7	6
7	1
3	1
1	3

node	direction	neighbor
1	>	4
4	<	1
1	>	5
5	<	1
1	>	6
6	<	1
7	>	6
6	<	7
7	>	1
1	<	7
3	>	1
1	<	3
1	>	3
3	<	1

Edge list → adjacency list

Shuffle: collect each node's records

source	**target**
1	4

1	5

1	6

7	6

7	1

3	1

1	3
---------	

node	**direction**	**neighbor**
1	<	3
1	<	7
1	>	3
1	>	4
1	>	5
1	>	6

10	>	2

2	<	10
2	<	3
2	<	4
2	>	8
2	>	9

3	<	1
3	<	4
3	>	1
3	>	2
3	>	4

Edge list → adjacency list

Reduce: for each node, concatenate all in- and out-neighbors

node	direction	neighbor
1	<	3
1	<	7
1	>	3
1	>	4
1	>	5
1	>	6
-----	-----	-----
10	>	2
-----	-----	-----
2	<	10
2	<	3
2	<	4
2	>	8
2	>	9
-----	-----	-----
3	<	1
3	<	4
3	>	1
3	>	2
3	>	4
	...	

node	in/out neighbors	
1	3 7	3 5 4 6
10		2
2	10 3 4	9 8
3	1 4	1 2 4
4	1 3	3 2
5	1	
6	1 7	
7		1 6
8	2	
9	2	

Edge list → adjacency list

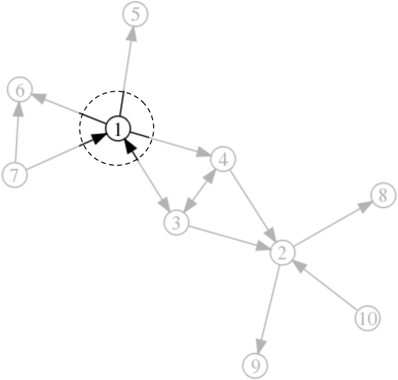
source	target
1	4
1	5
1	6
7	6
7	1
3	1
1	3
...	

node	direction	neighbor
1	<	3
1	<	7
1	>	3
1	>	4
1	>	5
1	>	6
10	>	2
2	<	10
2	<	3
2	<	4
2	>	8
2	>	9
3	<	1
3	<	4
3	>	1
3	>	2
3	>	4
	...	

node	in/out neighbors	
1	3 7	3 5 4 6
10		2
2	10 3 4	9 8
3	1 4	1 2 4
4	1 3	3 2
5	1	
6	1 7	
7		1 6
8	2	
9	2	

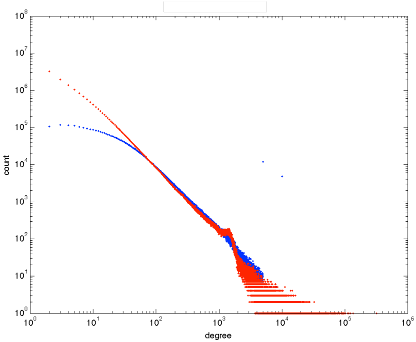
Edge list → adjacency list

Adjacency lists provide access to a node's *local structure* — e.g. we can *pass messages* from a node to its neighbors.



Degree distribution

node	in/out-degree	
1	2	4
10	0	1
2	3	2
3	2	3
4	2	2
5	1	0
6	2	0
7	0	2
8	1	0
9	1	0



Degree distribution

Map: for each node, output in- and out-degree with count (of 1)

node	in/out-degree	
1	2	4
10	0	1
2	3	2
3	2	3
4	2	2
5	1	0
6	2	0
7	0	2
8	1	0
9	1	0

bin	count
in_0	1
in_0	1
---------	
in_1	1
in_1	1
in_1	1
---------	
in_2	1
in_2	1
in_2	1
in_2	1
---------	
in_3	1
---------	
out_0	1
out_0	1
out_0	1
out_0	1
---------	
out_1	1
---------	
out_2	1
out_2	1
out_2	1
---------	
out_3	1
---------	
out_4	1

in/out	degree	count
in	0	2
in	1	3
in	2	4
in	3	1
out	0	4
out	1	1
out	2	3
out	3	1
out	4	1

Degree distribution

Shuffle: collect counts for each in/out-degree

node	**in/out-degree**	
1	2	4
10	0	1
2	3	2
3	2	3
4	2	2
5	1	0
6	2	0
7	0	2
8	1	0
9	1	0

→

```
bin   count
in_0 1
in_0 1
---------
in_1 1
in_1 1
in_1 1
---------
in_2 1
in_2 1
in_2 1
in_2 1
---------
in_3 1
---------
out_0     1
out_0     1
out_0     1
out_0     1
---------
out_1     1
---------
out_2     1
out_2     1
out_2     1
---------
out_3     1
---------
out_4     1
```

→

in/out	**degree**	**count**
in	0	2
in	1	3
in	2	4
in	3	1
out	0	4
out	1	1
out	2	3
out	3	1
out	4	1

Degree distribution

Reduce: add counts

node	in/out-degree	
1	2	4
10	0	1
2	3	2
3	2	3
4	2	2
5	1	0
6	2	0
7	0	2
8	1	0
9	1	0

→

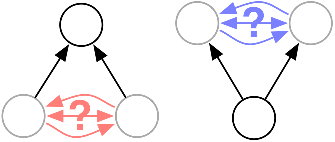
bin	count
in_0	1
in_0	1
---------	
in_1	1
in_1	1
in_1	1
---------	
in_2	1
in_2	1
in_2	1
in_2	1
---------	
in_3	1
---------	
out_0	1
out_0	1
out_0	1
out_0	1
---------	
out_1	1
---------	
out_2	1
out_2	1
out_2	1
---------	
out_3	1
---------	
out_4	1

→

in/out	degree	count
in	0	2
in	1	3
in	2	4
in	3	1
out	0	4
out	1	1
out	2	3
out	3	1
out	4	1

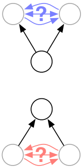
Clustering coefficient

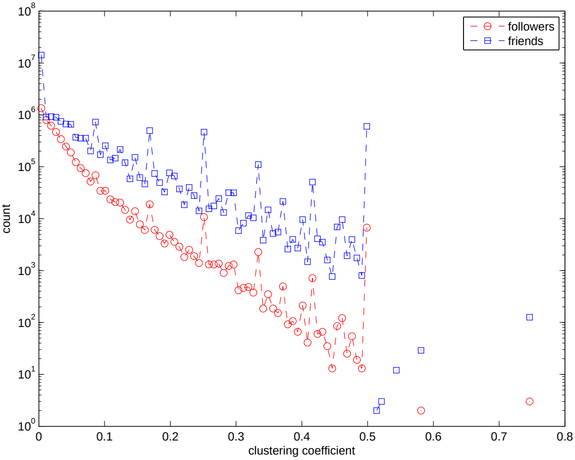
Fraction of edges amongst a node's in/out-neighbors



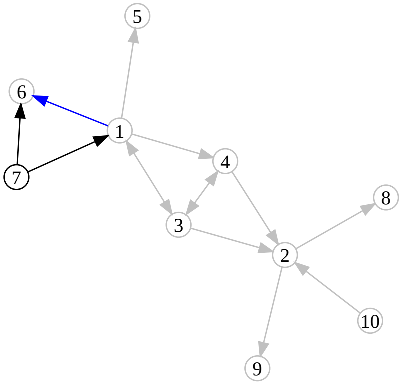
— e.g. how many of a node's friends are following each other?

Clustering coefficient

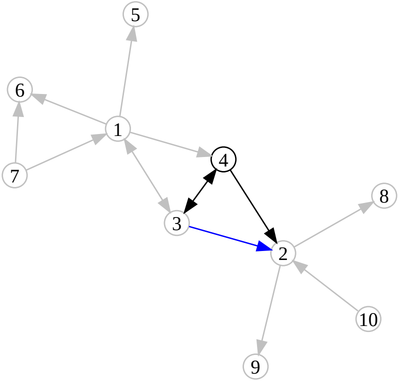




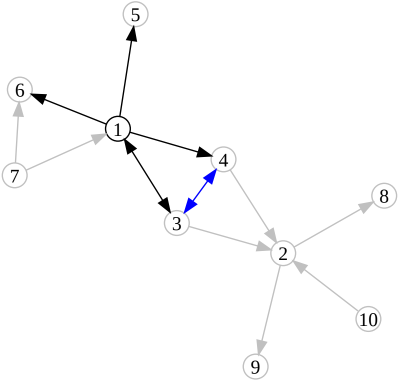
Clustering coefficient



Clustering coefficient



Clustering coefficient



Clustering coefficient

Map: pass all of a node's out-neighbors to each of its in-neighbors

node	in/out-neighbors	
1	3 7	3 5 4 6
10		2
2	10 3 4	9 8
3	1 4	1 2 4
4	1 3	3 2
5	1	
6	1 7	
7		1 6
8	2	
9	2	

node	out neighbor	two-hop neighbors
3	1	3 5 4 6
7	1	3 5 4 6
10	2	9 8
3	2	9 8
4	2	9 8
1	3	1 2 4
4	3	1 2 4
1	4	3 2
3	4	3 2
1	5	
1	6	
7	6	
2	8	
2	9	

Clustering coefficient

Shuffle: collect each node's two-hop neighborhoods

node	in/out-neighbors	
1	3 7	3 5 4 6
10		2
2	10 3 4	9 8
3	1 4	1 2 4
4	1 3	3 2
5	1	
6	1 7	
7		1 6
8	2	
9	2	

node	out neighbor	two-hop neighbors
1	3	1 2 4
1	4	3 2
1	5	
1	6	
10	2	9 8
2	8	
2	9	
3	1	3 5 4 6
3	2	9 8
3	4	3 2
4	2	9 8
4	3	1 2 4
7	1	3 5 4 6
7	6	

Clustering coefficient

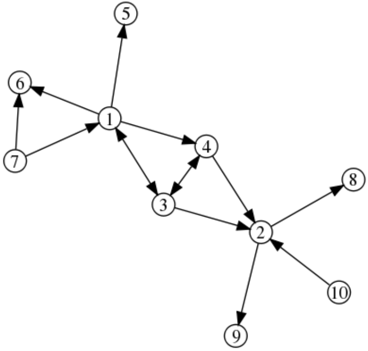
Reduce: count a half-triangle for each node reachable by both a one- and two-hop path

node	out neighbor	two-hop neighbors
1	3	1 2 4
1	4	3 2
1	5	
1	6	
10	2	9 8
2	8	
2	9	
3	1	3 5 4 6
3	2	9 8
3	4	3 2
4	2	9 8
4	3	1 2 4
7	1	3 5 4 6
7	6	

→

node	triangles
1	1.0
10	0.0
2	0.0
3	1.0
4	0.5
7	0.5

Clustering coefficient

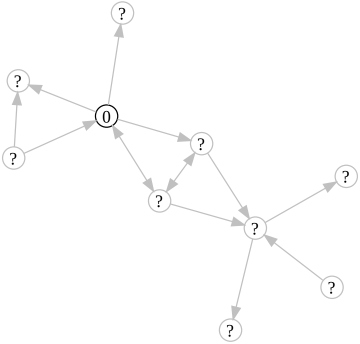


node	triangles
1	1.0
10	0.0
2	0.0
3	1.0
4	0.5
7	0.5

Note: this approach generates large amount of intermediate data relative to final output.

Breadth first search

Iterative approach: each MapReduce round expands the frontier

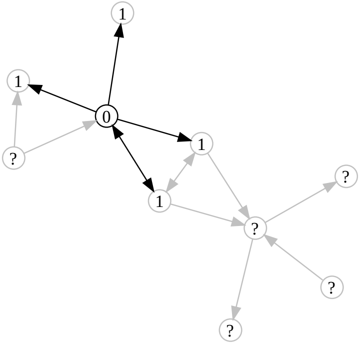


Map: If node's distance d to source is finite, output neighbor's distance as d+1

Reduce: Set node's distance to minimum received from all in-neighbors

Breadth first search

Iterative approach: each MapReduce round expands the frontier

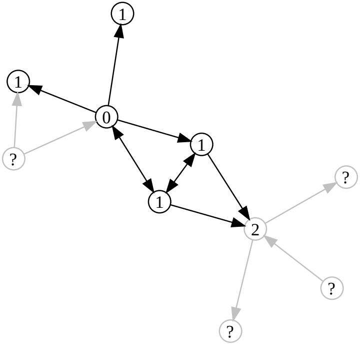


Map: If node's distance d to source is finite, output neighbor's distance as d+1

Reduce: Set node's distance to minimum received from all in-neighbors

Breadth first search

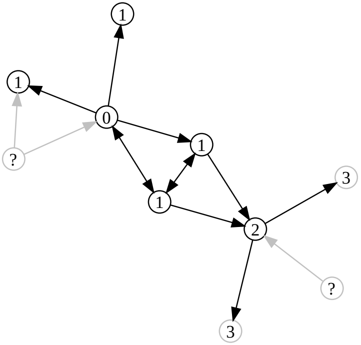
Iterative approach: each MapReduce round expands the frontier



Map: If node's distance d to source is finite, output neighbor's distance as d+1
Reduce: Set node's distance to minimum received from all in-neighbors

Breadth first search

Iterative approach: each MapReduce round expands the frontier

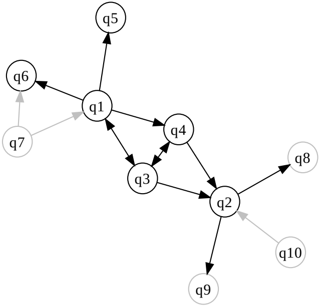


Map: If node's distance d to source is finite, output neighbor's distance as d+1
Reduce: Set node's distance to minimum received from all in-neighbors

Break complicated tasks into multiple, simpler MapReduce rounds.

Pagerank

Iterative approach: each MapReduce round broadcasts and collects edge messages for power method [3]



Map: Output current pagerank over degree to each out-neighbor
Reduce: Sum incoming probabilities to update estimate

http://bit.ly/nielsenpagerank

[3] Extra rounds for random jump, dangling nodes

Machine learning

Machine learning

- Often use MapReduce for feature extraction, then fit/optimize locally
- Useful for "embarassingly parallel" parts of learning, e.g.
    - parameters sweeps for cross-validation
    - independent restarts for local optimization
    - making predictions on independent examples
- Remember: MapReduce isn't always the answer

Classification

Example: given words in an article, assign article to one of K classes

"Floyd Landis showed up at the Tour of California"

Classification

Example: given words in an article, assign article to one of K classes

"Floyd Landis showed up at the Tour of California"

$$\Downarrow$$

Classification

Example: given words in an article, assign article to one of K classes

"Floyd Landis showed up at the Tour of California"

$$\Downarrow$$

...

Classification: naive Bayes

- Model presence/absence of each word as independent coin flip

$$
\begin{aligned}
p\,(\text{word}|\text{class}) &= \text{Bernoulli}(\theta_{wc}) \\
p\,(\text{words}|\text{class}) &= p\,(\text{word}_1|\text{class})\,p\,(\text{word}_2|\text{class})\ldots
\end{aligned}
$$

Classification: naive Bayes

- Model presence/absence of each word as independent coin flip

$$
\begin{aligned}
p\,(\text{word}|\text{class}) &= \text{Bernoulli}(\theta_{wc}) \\
p\,(\text{words}|\text{class}) &= p\,(\text{word}_1|\text{class})\,p\,(\text{word}_2|\text{class})\ldots
\end{aligned}
$$

- Maximum likelihood estimates of probabilities from word and class counts

$$
\begin{aligned}
\hat{\theta}_{wc} &= \frac{N_{wc}}{N_c} \\
\hat{\theta}_c &= \frac{N_c}{N}
\end{aligned}
$$

Classification: naive Bayes

- Model presence/absence of each word as independent coin flip

$$\begin{aligned} p\left(\mathrm{word}|\mathrm{class}\right) &= \mathrm{Bernoulli}(\theta_{wc}) \\ p\left(\mathrm{words}|\mathrm{class}\right) &= p\left(\mathrm{word}_1|\mathrm{class}\right) p\left(\mathrm{word}_2|\mathrm{class}\right) \ldots \end{aligned}$$

- Maximum likelihood estimates of probabilities from word and class counts

$$\begin{aligned} \hat{\theta}_{wc} &= \frac{N_{wc}}{N_c} \\ \hat{\theta}_c &= \frac{N_c}{N} \end{aligned}$$

- Use bayes' rule to calculate distribution over classes given words

$$p\left(\mathrm{class}|\mathrm{words}, \Theta\right) = \frac{p\left(\mathrm{words}|\mathrm{class}, \Theta\right) p\left(\mathrm{class}, \Theta\right)}{p\left(\mathrm{words}, \Theta\right)}$$

Classification: naive Bayes

Naive $\leftrightarrow$ independent features

Classification: naive Bayes

Naive $\leftrightarrow$ independent features

$$\Downarrow$$

Class-conditional word count

Classification: naive Bayes

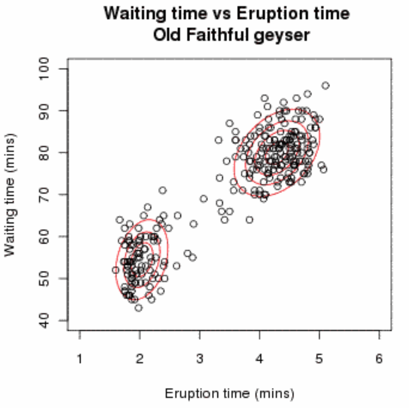
class	words
world	Economics Is on Agenda for U.S. Meetings in China
world	U.K. Backs Germany's Effort to Support Euro
sports	A Pitchers' Duel Ends in Favor of the Yankees
sports	After Doping Allegations, a Race for Details

→

class	word	count
sports	an	355
sports	be	317
sports	first	318
sports	game	379
sports	has	374
sports	have	284
sports	one	296
sports	said	325
sports	season	295
sports	team	279
sports	their	334
sports	this	293
sports	when	290
sports	who	363
world	after	347
world	but	299
world	government	300
world	had	352
world	have	342
world	he	355
world	its	308
world	mr	293
world	united	313
world	were	319

Clustering:

Find clusters of "similar" points



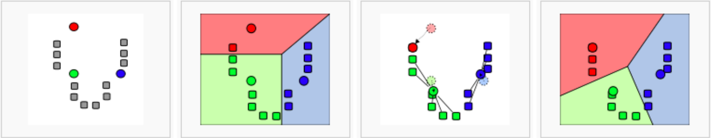
http://bit.ly/oldfaithful

Clustering: K-means

Map: Assign each point to cluster with closest mean[4], output (cluster, features)

Reduce: Update clusters by calculating new class-conditional means

Demonstration of the standard algorithm



1) *k* initial "means" (in this case *k*=3) are randomly selected from the data set (shown in color).	2) *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.	3) The centroid of each of the *k* clusters becomes the new means.	4) Steps 2 and 3 are repeated until convergence has been reached.

http://en.wikipedia.org/wiki/K-means_clustering

[4]Each mapper loads all cluster centers on init

Mahout

Apache Lucene Mahout

Mahout's goal is to build scalable machine learning libraries. With scalable we mean:

- Scalable to reasonably large data sets. Our core algorithms for clustering, classfication and batch based collaborative filtering are implemented on top of Apache Hadoop using the map/reduce paradigm. However we do not restrict contributions to Hadoop based implementations: Contributions that run on a single node or on a non-Hadoop cluster are welcome as well. The core libraries are highly optimized to allow for good performance also for non-distributed algorithms.
- Scalable to support your business case. Mahout is distributed under a commercially friendly Apache Software license.
- Scalable community. The goal of Mahout is to build a vibrant, responsive, diverse community to facilitate discussions not only on the project itself but also on potential use cases. Come to the mailing lists to find out more.

Currently Mahout supports mainly four use cases: Recommendation mining takes users' behavior and from that tries to find items users might like. Clustering takes e.g. text documents and groups them into groups of topically related documents. Classification learns from exisiting categorized documents what documents of a specific category look like and is able to assign unlabelled documents to the (hopefully) correct category. Frequent itemset mining takes a set of item groups (terms in a query session, shopping cart content) and identifies, which individual items usually appear together.

`http://mahout.apache.org/`

Thanks

- Sharad Goel[y]
- Winter Mason[y]
- Sid Suri[y]
- Sergei Vassilvitskii[y]
- Duncan Watts[y]
- Eytan Bakshy[m,y]

[y] Yahoo! Research (`http://research.yahoo.com`)
[m] University of Michigan

Thanks.

Questions?[5]

[5] http://jakehofman.com/icwsm2010