Hadoop and MapReduce

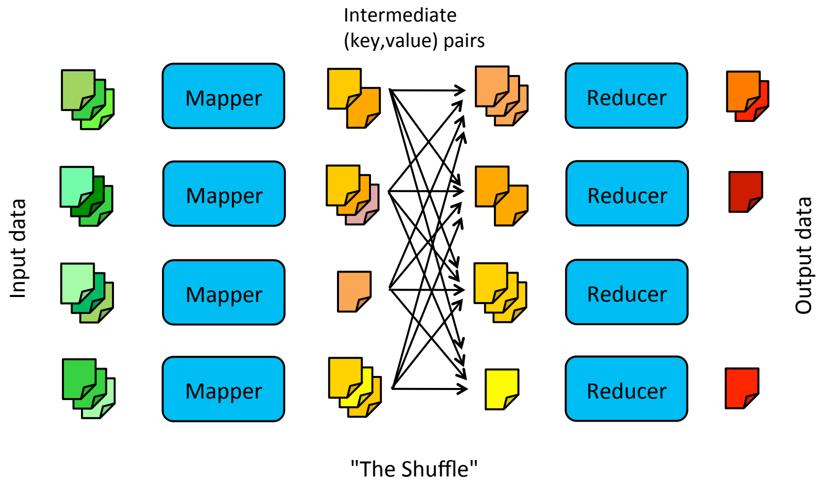
Dominic Duggan
Stevens Institute of Technology
Based on materials by
A. Haeberlen, Z. Ives, Jimmy Lin

1

HADOOP

2

Recap: MapReduce dataflow



3

What do we need to write?

- A mapper
  - Accepts (key,value) pairs from the input
  - Produces intermediate (key,value) pairs
  - Intermediate pairs shuffled
- A reducer
  - Accepts intermediate (key,value) pairs
  - Produces final (key,value) pairs for the output
- A driver
  - Specifies which inputs to use, where to put the outputs
  - Chooses the mapper and the reducer to use

4

Input Formats

Format	Key Type	Value Type
TextInputFormat (Default)	File Offset	Text Line
KeyValue InputFormat	Text (up to \t)	RemainingText
SequenceFile InputFormat	User-Defined	User-Defined

5

Output Formats

Format	Description
TextOutputFormat (default)	Key \t Value \n
SequenceFileOutputFormat	Binary Serialized keys and values
NullOutputFormat	Discards Output

6

The Mapper

Input format
(file offset, line)

Intermediate format
can be freely chosen

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

public class FooMapper extends Mapper<LongWritable, Text, Text, Text> {
    public void map(LongWritable key, Text value, Context context) {
      context.write(new Text("foo"), value);
    }
}
```


The Reducer

Intermediate format
(same as mapper output)

Output format

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

public class FooReducer extends Reducer<Text, Text, IntWritable, Text> {
    public void reduce(Text key, Iterable<Text> values, Context context)
           throws java.io.IOException, InterruptedException
    {
      for (Text value: values)
        context.write(new IntWritable(4711), value);
    }
}
```

Note: We may get
multiple values for
the same key!

The Driver

```
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class FooDriver {
  public static void main(String[] args) throws Exception {
    Job job = new Job();
    job.setJarByClass(FooDriver.class);

    FileInputFormat.addInputPath(job, new Path("in"));
    FileOutputFormat.setOutputPath(job, new Path("out"));

    job.setMapperClass(FooMapper.class);
    job.setReducerClass(FooReducer.class);

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

    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Mapper&Reducer are in the same Jar as FooDriver

Input and Output paths

Format of the (key,value) pairs output by the reducer

9

HDFS

10

HDFS

- Master-Worker Architecture
- Single NameNode
- Many (Thousands) DataNodes

11

HDFS Master (NameNode)

- Manages filesystem namespace
- File metadata ("inodes")
- Mapping "inode" to list of blocks and locations
- Authorization and Authentication
- Checkpoint & journal namespace changes

12

Namenode

Name node

foo.txt: 3,9,6
bar.data: 2,4
blah.txt: 17,18,19,20
xyz.img: 8,5,1,11

fsimage

Created abc.txt
Appended block 21 to blah.txt
Deleted foo.txt
Appended block 22 to blah.txt
Appended block 23 to xyz.img
...

edits

- State stored in two files: fsimage and edits
  – fsimage: Snapshot of file system metadata
  – edits: Changes since last snapshot

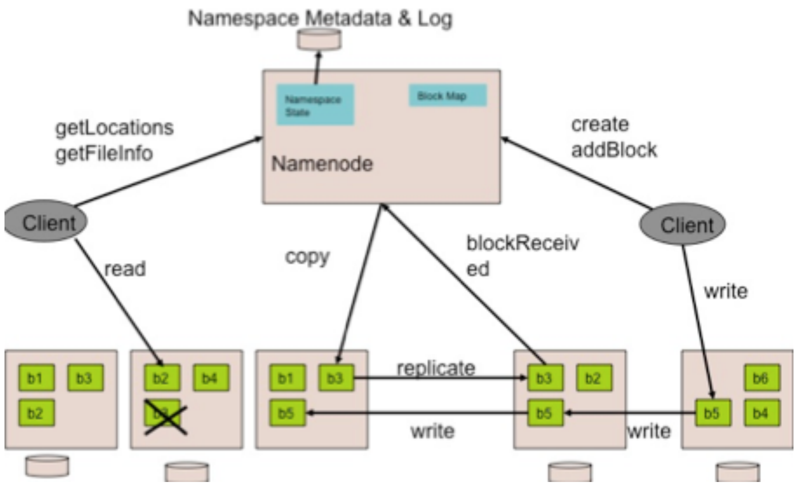
13

Secondary Namenode

- What if the state of the namenode is lost?
- Solution #1: Metadata backups
- Solution #2: Secondary Namenode
  – Has a copy of the metadata
  – Periodically merge edit log with fsimage
  – Lags lead to data loss

14

HDFS Architecture



15

Accessing data in HDFS

```
$ ls -la /tmp/hadoop-ahae/dfs/data/current/
total 209588
drwxrwxr-x 2 ahae ahae     4096 2013-10-08 15:46 .
drwxrwxr-x 5 ahae ahae     4096 2013-10-08 15:39 ..
-rw-rw-r-- 1 ahae ahae 11568995 2013-10-08 15:44 blk_-3562426239750716067
-rw-rw-r-- 1 ahae ahae    90391 2013-10-08 15:44 blk_-3562426239750716067_1020.meta
-rw-rw-r-- 1 ahae ahae        4 2013-10-08 15:40 blk_5467088600876920840
-rw-rw-r-- 1 ahae ahae       11 2013-10-08 15:40 blk_5467088600876920840_1019.meta
-rw-rw-r-- 1 ahae ahae 67108864 2013-10-08 15:44 blk_7080460240917416109
-rw-rw-r-- 1 ahae ahae   524295 2013-10-08 15:44 blk_7080460240917416109_1020.meta
-rw-rw-r-- 1 ahae ahae 67108864 2013-10-08 15:44 blk_-8388309644856805769
-rw-rw-r-- 1 ahae ahae   524295 2013-10-08 15:44 blk_-8388309644856805769_1020.meta
-rw-rw-r-- 1 ahae ahae 67108864 2013-10-08 15:44 blk_-9220415087134372383
-rw-rw-r-- 1 ahae ahae   524295 2013-10-08 15:44 blk_-9220415087134372383_1020.meta
-rw-rw-r-- 1 ahae ahae      158 2013-10-08 15:40 VERSION
$
```

- HDFS implements a separate namespace
  - Only blocks and block metadata are visible

16

Accessing data in HDFS

```
$ /usr/local/hadoop/bin/hadoop fs -ls /user/ahae
Found 4 items
-rw-r--r--   1 ahae supergroup       1366 2013-10-08 15:46 /user/ahae/README.txt
-rw-r--r--   1 ahae supergroup          0 2013-10-083 15:35 /user/ahae/input
-rw-r--r--   1 ahae supergroup          0 2013-10-08 15:39 /user/ahae/input2
-rw-r--r--   1 ahae supergroup  212895587 2013-10-08 15:44 /user/ahae/input3
$
```

- Examples:
  - hadoop fs -put [file] [hdfsPath] Stores a file in HDFS
  - hadoop fs -ls [hdfsPath] List a directory
  - hadoop fs -get [hdfsPath] [file] Retrieves a file from HDFS
  - hadoop fs -rm [hdfsPath] Deletes a file in HDFS
  - hadoop fs -mkdir [hdfsPath] Makes a directory in HDFS

17

HDFS Java API

```
// Get default file system instance
fs = Filesystem.get(new Configuration());

// Or Get file system instance from URI
fs = Filesystem.get(URI.create(uri),
                    new Configuration());

// Create, open, list, ...
OutputStream out = fs.create(path, ...);
InputStream in = fs.open(path, ...);
boolean isDone = fs.delete(path, recursive);
FileStatus[] fstat = fs.listStatus(path);
```

18

INSTALLING AND RUNNING

19

Prerequisites for Hadoop

- Java 1.6+
- Recommended: create a hadoop user
  - Standard privileges only
- SSH
  - Need ssh key for access to hosts in cluster

```
$ ssh-keygen -t rsa -P ””
$ cat $HOME/.ssh/id_rsa.pub >> \
      $HOME/.ssh/authorized_keys
```

20

Download Hadoop

- Download from Apache:
  - http://hadoop.apache.org/releases.html#Download

- Install (e.g. in /usr/local)

```
$ cd /usr/local
$ gunzip $HOME/Downloads/hadoop-2.X.X.tar.gz
$ sudo tar xvf $HOME/Downloads/hadoop-2.X.X.tar
$ sudo ln -s hadoop-2.X.X hadoop
$ sudo chown -R hadoop:staff hadoop-2.X.X hadoop
```

21

Setting Your User Profile

- Edit `$HOME/.bash_profile`
- Add these lines:

```
export HADOOP_PREFIX="/usr/local/hadoop"
export HADOOP_HOME="${HADOOP_PREFIX}"
export HADOOP_COMMON_HOME="${HADOOP_PREFIX}"
export HADOOP_CONF_DIR="${HADOOP_PREFIX}/etc/hadoop"
export HADOOP_HDFS_HOME="${HADOOP_PREFIX}"
export HADOOP_MAPRED_HOME="${HADOOP_PREFIX}"
export HADOOP_YARN_HOME="${HADOOP_PREFIX}”
export "PATH=${PATH}:${HADOOP_PREFIX}/bin:
                  ${HADOOP_PREFIX}/sbin”
```

- Load into current shell

```
. $HOME/.bash_profile
```

22

Configuring Hadoop: `hadoop-env.sh`

- Located in `$HADOOP/etc/hadoop/hadoop-env.sh`
- Edit to set JAVA_HOME
  - E.g. on MacOS:

```
export JAVA_HOME=`/usr/libexec/java_home -v 1.7+`
```


Manual compilation

- Step #1: Put hadoop-core-...jar into classpath:

```
export
CLASSPATH=$CLASSPATH:/path/to/hadoop/hadoop-core-...jar
```

- Step #2: Compile mapper, reducer, driver:

```
javac FooMapper.java FooReducer.java FooDriver.java
```

- Step #3: Package into a JAR file:

```
jar cvf Foo.jar *.class
```

- Alternative: "Export..."/"Java JAR file" in Eclipse

Standalone Operation

- One Java process

```
$ mkdir input

$ cp ... input

$ hadoop jar jarfile-name class-name \
   input output program-arguments

$ cat output/*
```


Pseudo-Distributed Operation

- One site, one Java process per node

- $HADOOP_HOME/etc/hadoop/core-site.xml

```
<configuration>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://localhost:9000</value>
    </property>
</configuration>
```

Pseudo-Distributed Operation

- One site, one Java process per node

- $HADOOP_HOME/etc/hadoop/hdfs-site.xml

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
</configuration>
```

27

Pseudo-Distributed Operation

- Format the file system

```
$ hdfs namenode -format
```

- Start daemons (NameNode and DataNode)

```
$ start-dfs.sh
```

  – Log written to $HADOOP_HOME/logs
  – REST interface for NameNode: http://localhost:50070

- Make HDFS directories

```
$ hdfs dfs -mkdir /user
$ hdfs dfs -mkdir /user/joe
```

28

Pseudo-Distributed Operation

- Copy input files into the Hadoop file system
```
$ hdfs dfs –put ... input
```

- Run examples
```
$ hadoop jar jarfile class-name \
     input output arguments
```

- Examine output
```
$ hdfs dfs -get output output
$ cat output/*
```

- Stop the daemons
```
$ stop-dfs.sh
```

29

Pseudo-Distributed with Yarn

- etc/hadoop/mapred-site.xml:
```
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
</configuration>
```

- etc/hadoop/yarn-site.xml:
```
<configuration>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
</configuration>
```

30

Pseudo-Distributed with Yarn

- Start ResourceManager and NodeManager daemons:

```
$ start-yarn.sh
```

  – REST interface for ResourceManager: http://localhost:8088

- Run a MR job

- Stop the daemons:

```
$ stop-yarn.sh
```

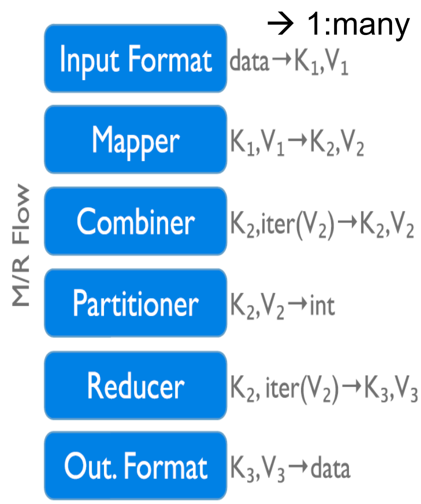
31

HADOOP COMPONENTS

32

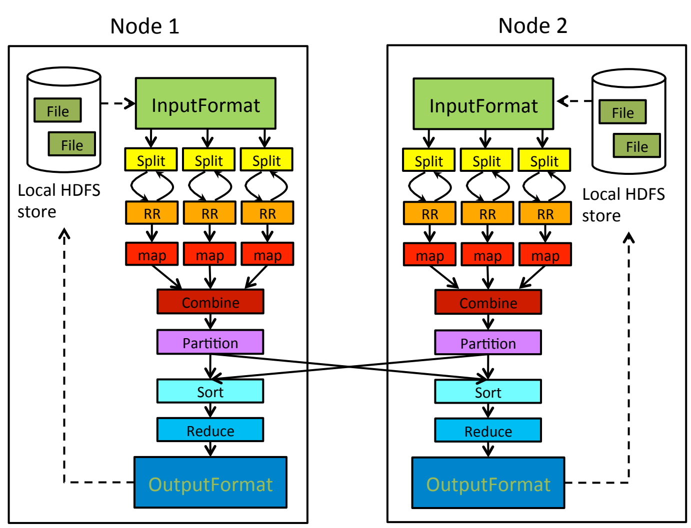
Data Flow in Hadoop

- InputFormat
- Map function
- Partitioner
- Sorting & Merging
- Combiner
- Shuffling
- Merging
- Reduce function
- OutputFormat



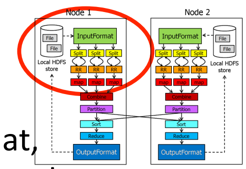
33

Detailed dataflow in Hadoop



34

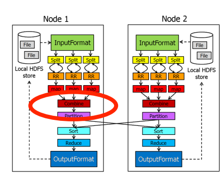
Input Format



- Input files and format
  - Defaults provided, e.g., TextInputFormat, DBInputFormat, KeyValueTextInputFormat...
- Defines InputSplits
  - Break file into separate tasks
  - Example: one task for each 64MB block
- Factory for RecordReaders
  - RecordReaders read the file into (key,value) pairs
  - TextInputFormat: byte offset in file as key
  - KeyValueInputFormat: key is everything up to first tab

35

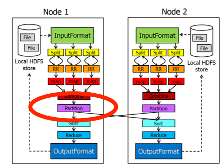
Combiners



- Optional component after mappers
  - Input: All data emitted by mappers on a given node
  - Output passed to the partitioner
- Why is this useful?
  - Word count emits (xyz, 1) pairs for each word xyz
  - Pass (xyz, k) to the reducer

36

Partitioner



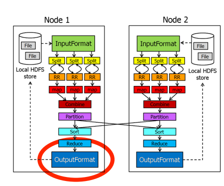
- Which intermediate key-value pairs should go to which reducer

- Defines a partition on the set of KV pairs
  – Number of partitions = number of reducers

- Default partitioner (HashPartitioner): partition based on a hash of the key

37

Output Format



- Counterpart to InputFormat

- Where output is stored
  – Factory for RecordWriter

- Several implementations provided
  – TextOutputFormat (default)
  – DBOutputFormat
  – MultipleTextOutputFormat
  – …

38

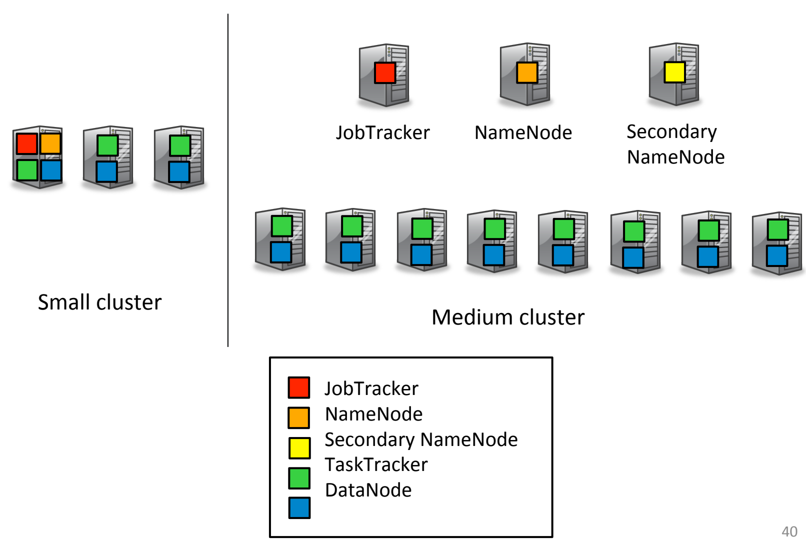
Hadoop daemons

- TaskTracker
  - Runs maps and reduces. One per node.
- JobTracker
  - Accepts jobs; assigns tasks to TaskTrackers
- DataNode
  - Stores HDFS blocks
- NameNode
  - Stores HDFS metadata
- SecondaryNameNode
  - Merges edits file with snapshot; "backup" for NameNode

A single node can run more than one of these!

39

An example configuration

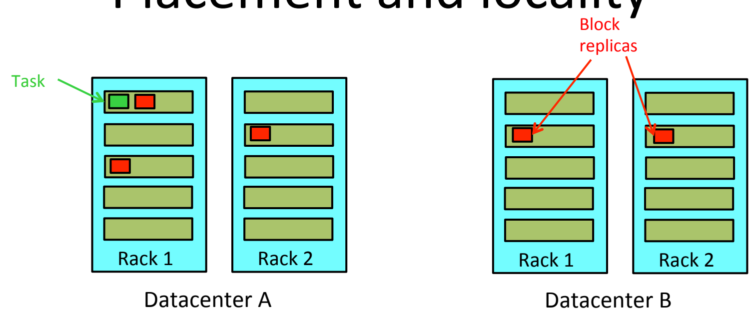


40

Fault tolerance

- What if a node fails during a job?
  - JobTracker re-executes the failed node's tasks
- What specifically should be re-executed?
  - Depends on the phase the job was in
  - Mapping phase: Re-execute all maps assigned to failed node
  - Reduce phase: Re-execute all reduces assigned to node
    - Need to re-execute map tasks on the failed node as well!

Placement and locality



- Which of the replicated blocks should be read?
  - If possible, pick the closest one (reduces network load)
  - Distance metric takes into account: Nodes, racks, datacenters
- Where should the replicas be put?
  - Tradeoff between fault tolerance and locality/performance

GRAPH ALGORITHMS

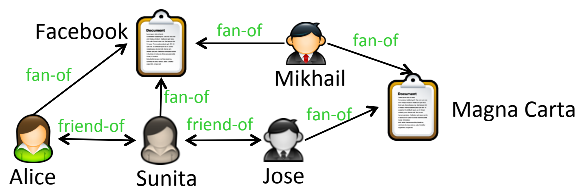
43

Beyond average/sum/count

- Networks of relationships and shared features
  - Members of a social network
  - Customers
  - The Web (documents with links)
  - Documents: topics, words, authors, etc.

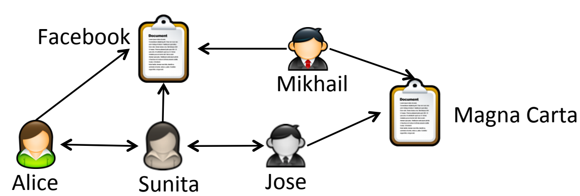
44

Thinking about related objects



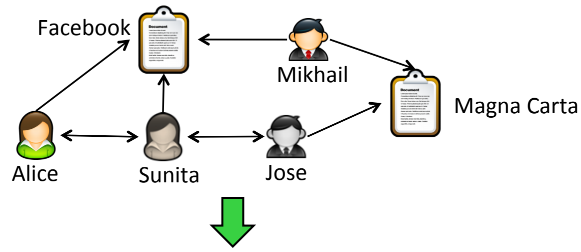
- Represent related objects as labeled, directed graph
- Entities == nodes
  - Nodes: IDs
- Relationships == edges
  - Edges: values

Encoding the data in a graph



- G = (V, E) where V is vertices, E is edges of the form (v_1, v_2) where $v_1, v_2 \in V$
- Assume we only care about connected vertices
  - Then we can capture a graph simply as the edges
  - … or as an adjacency list: v_i goes to $[v_j, v_{j+1}, \ldots]$

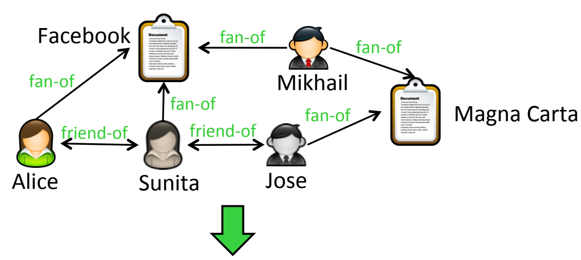
Graph encodings: Set of edges



(Alice, Facebook)
(Alice, Sunita)
(Jose, Magna Carta)
(Jose, Sunita)
(Mikhail, Facebook)
(Mikhail, Magna Carta)
(Sunita, Facebook)
(Sunita, Alice)
(Sunita, Jose)

47

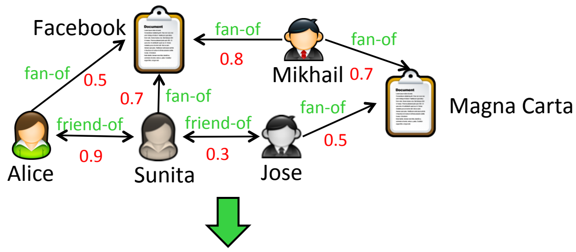
Graph encodings: Adding edge types



(Alice, fan-of, Facebook)
(Alice, friend-of, Sunita)
(Jose, fan-of, Magna Carta)
(Jose, friend-of, Sunita)
(Mikhail, fan-of, Facebook)
(Mikhail, fan-of, Magna Carta)
(Sunita, fan-of, Facebook)
(Sunita, friend-of, Alice)
(Sunita, friend-of, Jose)

48

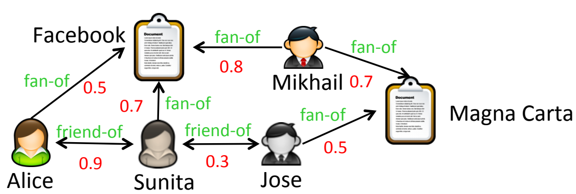
Graph encodings: Adding weights



(Alice, fan-of, 0.5, Facebook)
(Alice, friend-of, 0.9, Sunita)
(Jose, fan-of, 0.5, Magna Carta)
(Jose, friend-of, 0.3, Sunita)
(Mikhail, fan-of, 0.8, Facebook)
(Mikhail, fan-of, 0.7, Magna Carta)
(Sunita, fan-of, 0.7, Facebook)
(Sunita, friend-of, 0.9, Alice)
(Sunita, friend-of, 0.3, Jose)

49

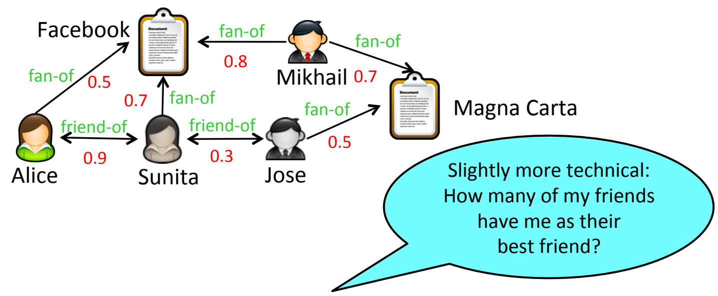
A computation model for graphs



- Perform computations
  - Simple example: Which users are their friends' best friend?
- Method
  - annotating the vertices with additional information
  - propagating the information along the edges

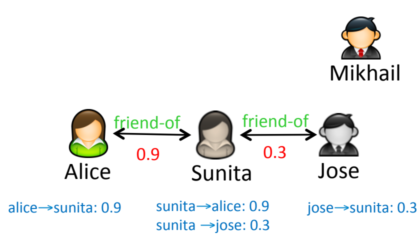
50

A computation model for graphs



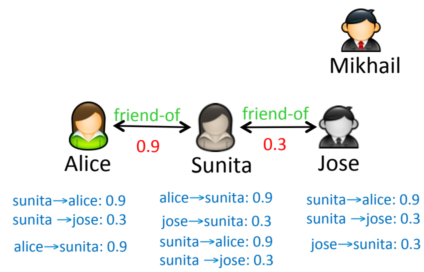
- Example: Am I my friends' best friend?
  - Step #1: Discard irrelevant vertices and edges

A computation model for graphs



- Example: Am I my friends' best friend?
  - Step #1: Discard irrelevant vertices and edges
  - Step #2: Annotate each vertex with list of friends

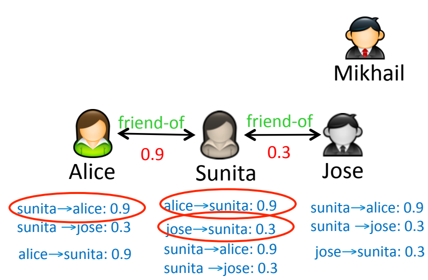
A computation model for graphs



- Example: Am I my friends' best friend?
  - Step #1: Discard irrelevant vertices and edges
  - Step #2: Annotate each vertex with list of friends
  - Step #3: Push annotations along each edge

53

A computation model for graphs



- Example: Am I my friends' best friend?
  - Step #1: Discard irrelevant vertices and edges
  - Step #2: Annotate each vertex with list of friends
  - Step #3: Push annotations along each edge
  - Step #4: Determine result at each vertex

54

Can we do this in MapReduce?

```
map(key: node, value: list of <otherNode, relType, strength>)
{

}
reduce(key: ________, values: list of _________)
{

}
```

- Using adjacency list representation?

55

Can we do this in MapReduce?

```
map(key: node, value: <otherNode, relType, strength>)
{

}
reduce(key: ________, values: list of _________)
{

}
```

- Using single-edge data representation?

56

Generalizing…

- Beyond direct friend relationships
  - Example: How many of my friends' friends (distance-2 neighbors) have me as their best friend's best friend?
- How about distance k>2?
- Requires multiple iterations of MapReduce!

57

Iterative MapReduce

```
copy files from input dir → staging dir 1
(optional: do some preprocessing)

while (!terminating condition) {
  map from staging dir 1
  reduce into staging dir 2
  move files from staging dir 2 → staging dir 1
}

(optional: postprocessing)
move files from staging dir 2 → output dir
```

- Reduce output must be compatible with the map input
  - What can happen if we filter out some information in the mapper or in the reducer?

58

Graph algorithms and MapReduce

- Multiple map/reduce stages processing one "wave" at a time
  - Iterative MapReduce
  - Chains of map/reduce

59

PATH-BASED ALGORITHMS

60

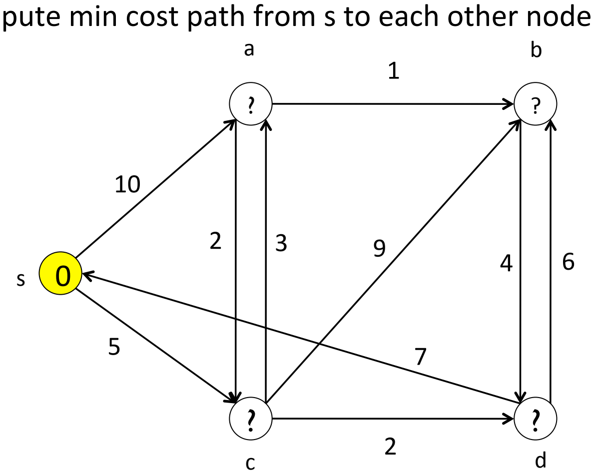
Path-based algorithms

- Compute information about the paths (sets of paths) between nodes
  - Edges may be annotated with cost, distance, or similarity
- Examples:
  - Shortest path from one node to another
  - Minimum spanning tree (minimal-cost tree connecting all vertices in a graph)
  - Steiner tree (minimal-cost tree connecting certain nodes)
  - Topological sort (node in a DAG comes before all nodes it points to)

61

Single-Source Shortest Path (SSSP)

- Given directed graph where each edge (u,v) has cost dist(u,v)
- Given a start node s
- Compute min cost path from s to each other node



62

SSSP: Intuition

- The shortest path follows the principle of optimality: the last step (u,v) makes use of the shortest path to u

```
bestDistance(v) {
  if (v == source) {
    return distance 0
  } else {
    du = min{ bestDistance(u) + dist(u,v) | u adjacent to v}
    return du
  }
}
```


SSSP: Intuition

- The shortest path follows the principle of optimality: the last step (u,v) makes use of the shortest path to u

```
bestDistanceAndPath(v) {
  if (v == source) {
    return (distance 0, list(v))
  } else {
    du = min{ d + dist(u,v) | u adjacent to v and
               bestDistanceAndPath(u) == (d, L) }
    u = node with bestDistanceAndPath(u) == (d, _)
               and d + dist(u,v) == du
    (d,L) = bestDistanceAndPath(u)
    return (distance du, list L with v added)
  }
}
```

SSSP: Solution

- Traditional approach: Dijkstra's algorithm

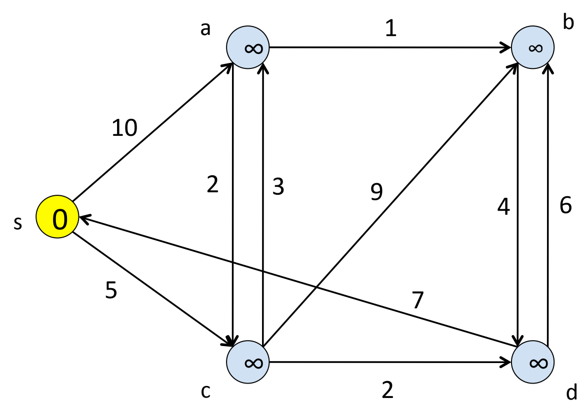
```
V: vertices, E: edges, S: start node

foreach v in V
  dist_S_to[v] = infinity
  predecessor[v] = nil
spSet = {}
Q := V
while (Q not empty) do
  u := Q.removeNodeClosestTo(S)
  spSet := spSet + {u}
  foreach v in V where (u,v) in E
    if (dist_S_To[v] > dist_S_To[u] + dist(u,v)) {
      dist_S_To[v] = dist_S_To[u] + dist(u,v)
      predecessor[v] = u
    }
```

Initialize length and last step of path to default values

Update length and path based on edges radiating from u

SSSP: Dijkstra in Action

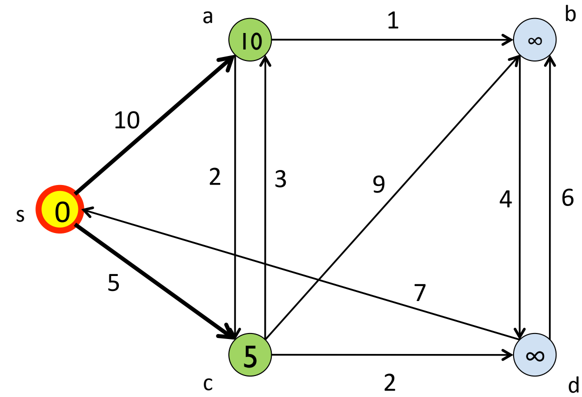
Example from CLR 2nd ed. p. 528



Q = {s,a,b,c,d} spSet = {}
dist_S_To: {(a,∞), (b,∞), (c,∞), (d,∞)}
predecessor: {(a,nil), (b,nil), (c,nil), (d,nil)}

SSSP: Dijkstra in Action

Example from CLR 2nd ed. p. 528

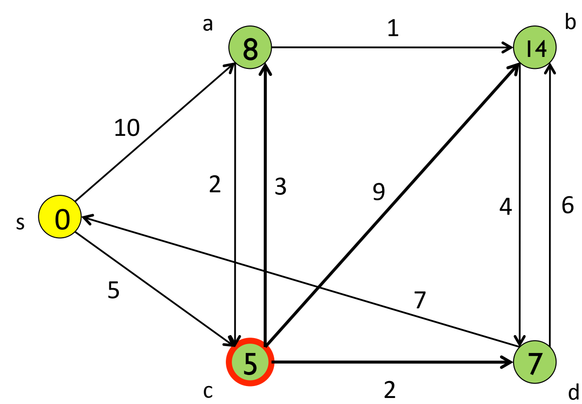


Q = {a,b,c,d}   spSet = {s}
dist_S_To: {(a,10), (b,∞), (c,5), (d,∞)}
predecessor: {(a,s), (b,nil), (c,s), (d,nil)}

67

SSSP: Dijkstra in Action

Example from CLR 2nd ed. p. 528

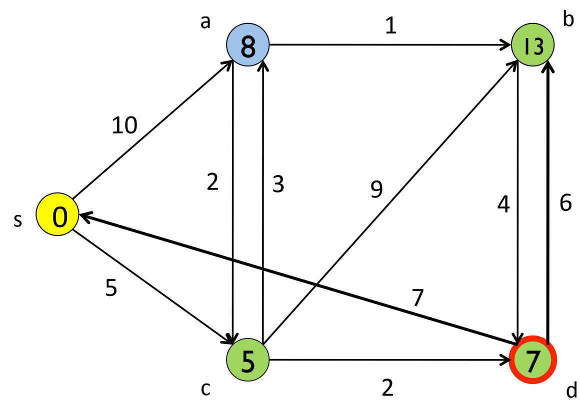


Q = {a,b,d}   spSet = {c,s}
dist_S_To: {(a,8), (b,14), (c,5), (d,7)}
predecessor: {(a,c), (b,c), (c,s), (d,c)}

68

Example from CLR 2nd ed. p. 528

SSSP: Dijkstra in Action

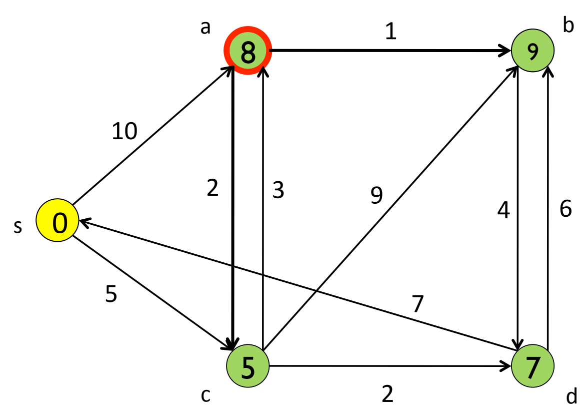


Q = {a,b} spSet = {c,d,s}
dist_S_To: {(a,8), (b,13), (c,5), (d,7)}
predecessor: {(a,c), (b,d), (c,s), (d,c)}

69

Example from CLR 2nd ed. p. 528

SSSP: Dijkstra in Action

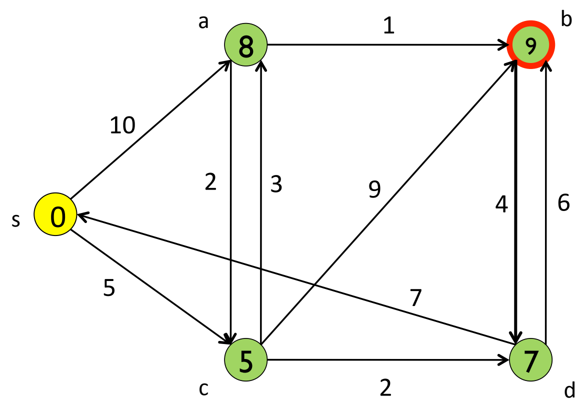


Q = {b} spSet = {a,c,d,s}
dist_S_To: {(a,8), (b,9), (c,5), (d,7)}
predecessor: {(a,c), (b,a), (c,s), (d,c)}

70

SSSP: Dijkstra in Action

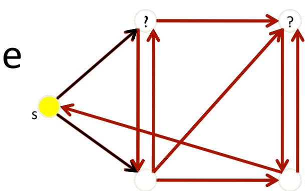
Example from CLR 2nd ed. p. 528



Q = {} spSet = {a,b,c,d,s}
dist_S_To: {(a,8), (b,9), (c,5), (d,7)}
predecessor: {(a,c), (b,a), (c,s), (d,c)}

71

SSSP: How to parallelize?



- Dijkstra single route at a time
  - No real parallelism
- Alternatively:
  - "radiate" from the origin
  - one "edge hop distance" at a time

72

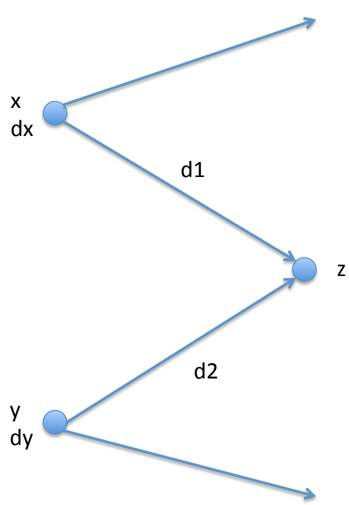
SSSP: Revisiting the inductive definition

```
bestDistance(v) {
  if (v == source) {
    return distance 0
  } else {
    du = min{ bestDistance(u) + dist(u,v) | u adjacent to v}
    return du
  }
}
```

- Dijkstra's algorithm
  - Select min u (prunes certain points)
- Instead look at all potential u's
  - Compute iteratively
  - "frontier set" of u nodes

73

SSSP: MapReduce formulation



Map Input:
x → …{<z, d1>}
y → …{<z,d2>}

Map Output:
z → {<x, dx+d1>}
z → {<y, dy+d2>}

Reduce: pick min of
{dx+d1, dy+d2}

Reduce Output:
z → dz, x or y, …

74

SSSP: MapReduce formulation

The shortest path we have found so far from the source to nodeID has length ∞...

this is the next hop on that path...

... and here is the adjacency list for nodeID

- init:
  - For each node, node ID → <∞, -, {<succ-node-ID,edge-cost>}>
- map:
  - take node ID → <distance, next, {<succ-node-ID,edge-cost>}>
  - For each succ-node-ID:

    emit succ-node ID → {<node ID, distance + edge-cost>}

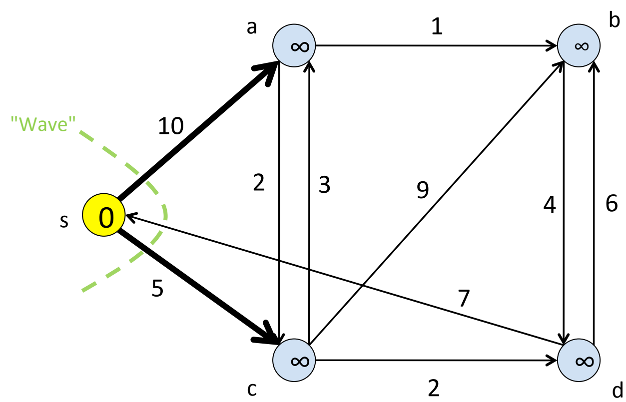
    This is a new path from the source to succ-node-ID that we just discovered (not necessarily shortest)
  - emit node ID → <distance, next, {<succ-node-ID,edge-cost>}>

    Why is this necessary?
- reduce:
  - distance := min cost from a predecessor; next := that predec.
  - emit node ID → <distance, next, {<succ-node-ID,edge-cost>}>
- Repeat until no changes
- Postprocessing: Remove adjacency lists

75

Iteration 0: Base case

mapper:	(a,<s,10>) (c,<s,5>) edges
reducer:	(a,<10, ...>) (c,<5, ...>)

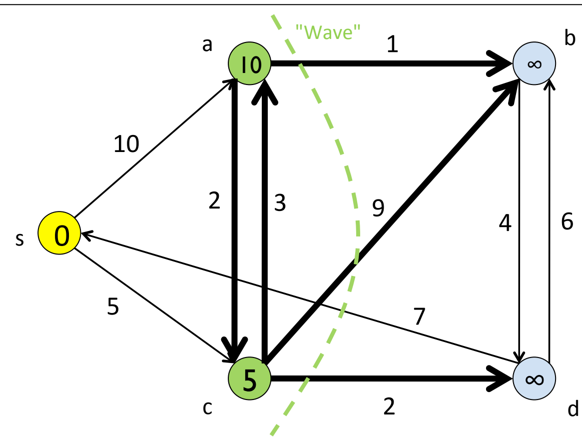


76

Iteration 1

mapper: (a,<s,10>) (c,<s,5>) (a,<c,8>) (c,<a,9>) (b,<a,11>)
(b,<c,14>) (d,<c,7>) edges

reducer: (a,<8, ...>) (c,<5, ...>) (b,<11, ...>) (d,<7, ...>)

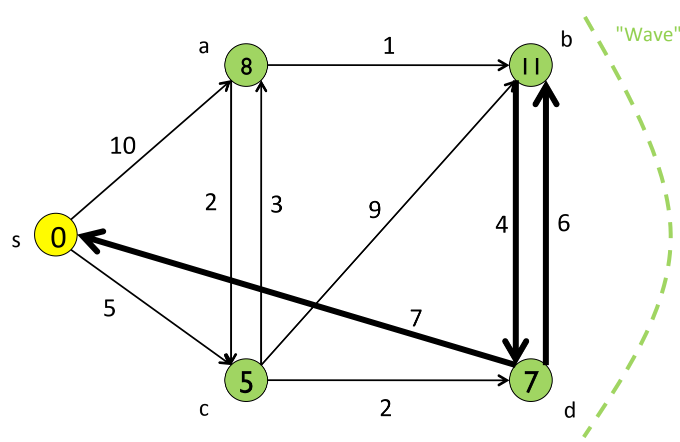


77

Iteration 2

mapper: (a,<s,10>) (c,<s,5>) (a,<c,8>) (c,<a,9>) (b,<a,11>) (b,<c,14>) (d,<c,7>)
(b,<d,13>) (d,<b,15>) edges

reducer: (a,<8>) (c,<5>) (b,<11>) (d,<7>)

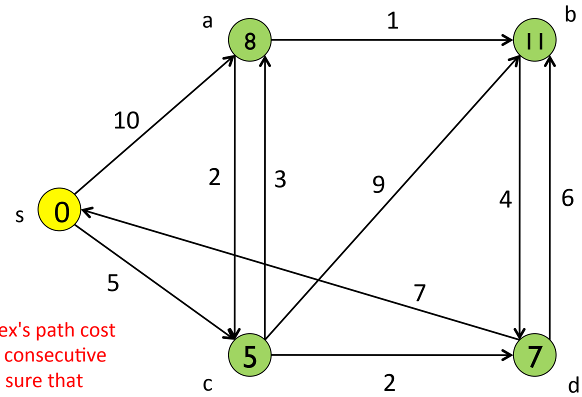


78

Iteration 3

No change!
Convergence!

mapper:	(a,<s,10>) (c,<s,5>) (a,<c,8>) (c,<a,9>) (b,<a,11>) (b,<c,14>) (d,<c,7>) (b,<d,13>) (d,<b,15>) edges
reducer:	(a,<8>) (c,<5>) (b,<11>) (d,<7>)



Question: If a vertex's path cost is the same in two consecutive rounds, can we be sure that this vertex has converged?

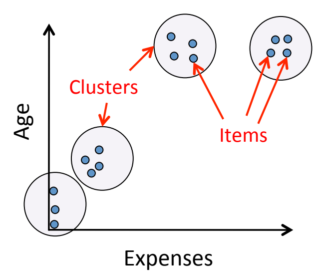
CLUSTERING

Learning (clustering / classification)

- Group related entities
  - Clustering: based on similarity
  - Classification: based on putting them into a semantically meaningful class
- Both are instances of machine learning

81

The k-clustering Problem



- Given: A set of items in a n-dimensional feature space
  - Example: data points from survey, people in a social network
- Goal: Group the items into k “clusters”

82

Approach: k-Means

- Let $m_1, m_2, \ldots, m_k$ be representative points for each of our k clusters
  - Specifically: the centroid of the cluster
- Initialize $m_1, m_2, \ldots, m_k$ to random values in the data
- For t = 1, 2, …:
  - Map each observation to the closest mean

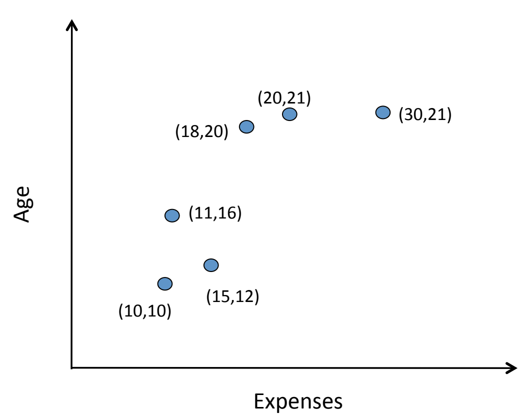
$$S_i^{(t)} = \left\{ x_j : \left\| x_j - m_i^{(t)} \right\| \le \left\| x_j - m_{i*}^{(t)} \right\|, i* = 1, \ldots, k \right\}$$

  - Assign the m_i to be a new centroid for each set

$$m_i^{(t+1)} = \frac{1}{\left| S_i^{(t)} \right|} \sum_{x_j \in S_i^{(t)}} x_j$$

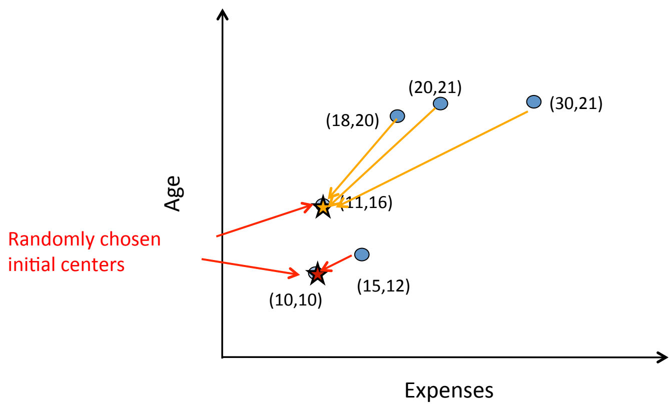
83

A simple example (1/4)



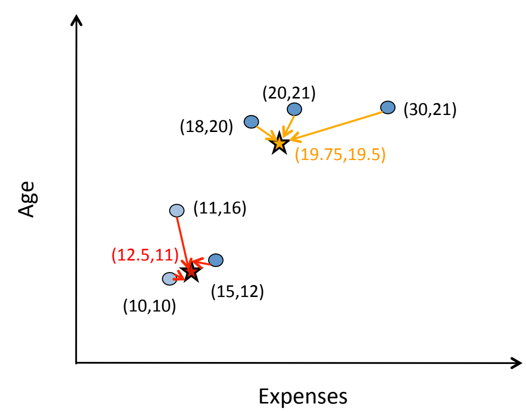
84

A simple example (2/4)



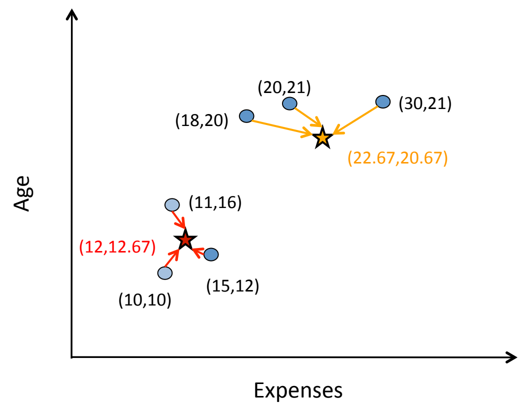
85

A simple example (3/4)



86

A simple example (4/4)



Stable!

k-Means in MapReduce

- Map #1:
  - Input:
    node ID → <position, centroid ID,
    {centroid IDs and positions}>
  - Compute nearest centroid; emit
    centroid ID → <node ID, position>
- Reduce #1:
  - Recompute centroid position from positions of nodes in it
  - Emit centroidID → <node IDs, positions> and
    for all other centroid IDs, emit
    otherCentroidID → centroid(centroidID,X,Y)
    - Each centroid will need to know where all the other centroids are

k-Means in MapReduce

- Map #2:
  - Pass through values to Reducer #2
- Reduce #2:
  - For each node in the current centroid, emit
    node ID → <position, centroid ID,
    {centroid IDs and positions}>
    - Input for the next map iteration
  - Also, emit <X, <centroid ID, position>>
    - This will be the 'result'
- Repeat until no change

CLASSIFICATION

Classification

- Suppose we want to learn what is spam (or interesting, or …)
  - Predefine a set of classes with semantic meaning
  - Train an algorithm to look at data and assign a class
    - Based on giving it some examples of data in each class
    - … and the sets of features they have

A simple example

- Look at the keywords in the email's title:

```
Message(1, "Won contract")
Message(2, "Won award")
Message(3, "Won the lottery")
Message(4, "Unsubscribe")
Message(5, "Millions of customers")
Message(6, "Millions of dollars")
```

- What is **probability** message "Won Millions" is ?

$$p(\text{spam}|\text{containsWon},\text{containsMillions}) = \frac{p(\text{spam})\, p(\text{containsWon},\text{containsMillions}\,|\text{spam})}{p(\text{containsWon},\text{containsMillions})}$$

Bayes' Theorem

Classification using Naïve Bayes

- Basic assumption: Probabilities of events are independent
- Under this assumption,

$$\frac{p(\text{spam})\ p(\text{containsWon}, \text{containsMillions} \mid \text{spam})}{p(\text{containsWon}, \text{containsMillions})}$$

$$= \frac{p(\text{spam})\ p(\text{containsWon} \mid \text{spam})\ p(\text{containsMillions} \mid \text{spam})}{p(\text{containsWon})\ p(\text{containsMillions})}$$

= 0.5 * 0.67 * 0.33 / (0.5 * 0.33) = 0.67

- Train a learner (compute the above probabilities) using MapReduce?

93

Train the learner

- p(spam)
  - Count how many spam emails there are Easy
  - Count total number of emails Easy
- p(containsXYZ | spam)
  - Count how many spam emails contain XYZ 1
  - Count how many spam emails there are Easy
- p(containsXYZ)
  - Count how many emails contain XYZ overall 2
  - Count total number of emails Easy

94

Training a Naïve Bayes Learner

- map 1:
  - takes messageId → <class, {words}>
  - emits <word, class> → 1
- reduce 1:
  - emits <word, class> → count

Count how many emails in the class contain the word (modified WordCount)

- map 2:
  - takes messageId → <class, {words}>
  - emits word → 1
- reduce 2:
  - emits word → totalCount

Count how many emails contain the word overall (WordCount)

95

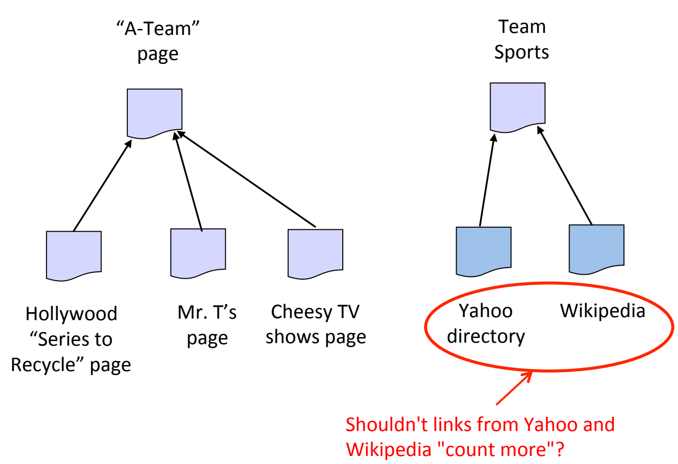
PAGERANK

96

Link analysis

- Search engine challenge
  - Problem: how to prioritize pages?
- Idea: Hyperlinks encode human judgment
  - Intra-domain links: internal navigation
  - Inter-domain links: confer authority?
- Idea: Boost the rank of pages with lots of inbound links?

Problem: Popularity ≠ relevance!

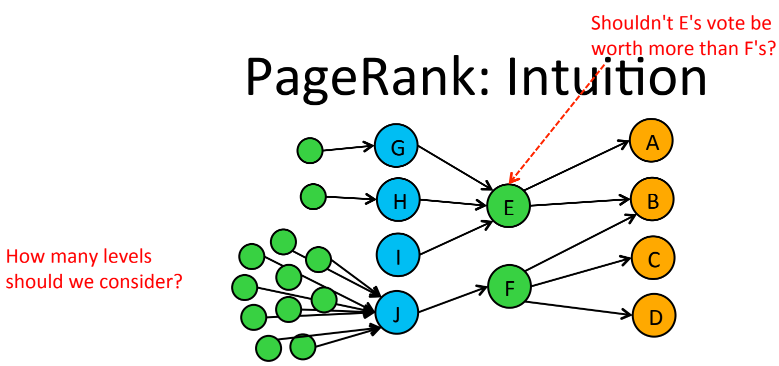


Other applications

- How do we measure the "impact" of a researcher?
  (#papers? #citations?)
- Who are the most "influential" individuals in a social network?
  (#friends?)
- Which programmers are writing the "best" code?
  (#uses?)
- ...

99

PageRank: Intuition



- Imagine a contest for The Web's Best Page
  - Initially, each page has one vote
  - A page votes for all pages it links to
  - A page's vote is split vote equally between endorsed pages
  - Voting proceeds in rounds
  - In each round, each page has the number of votes it received in the previous round

100

PageRank

- Each page i is given a rank x_i
- Goal: Assign the x_i such that the rank of each page is governed by the ranks of the pages linking to it:

$$x_i = \sum_{j \in B_i} \frac{1}{N_j} x_j$$

Rank of page i

Rank of page j

Every page
j that links to i

Number of
links out
from page j

How do we compute
the rank values?

101

Random Surfer Model

- Random surfer starts on a random page and, in each step:
  - with probability d, clicks on a random link
  - with probability 1-d, jumps to a random page
    - Ignoring links on page
- PageRank of a page: the fraction of steps the surfer spends on that page
  - Transition matrix can be interpreted as a Markov Chain

102

Iterative PageRank (simplified)

Initialize all ranks to be equal, e.g.:

$$x_i^{(0)} = \frac{1}{n}$$

Iterate until convergence

$$x_i^{(k+1)} = \sum_{j \in B_i} \frac{1}{N_j} x_j^{(k)}$$

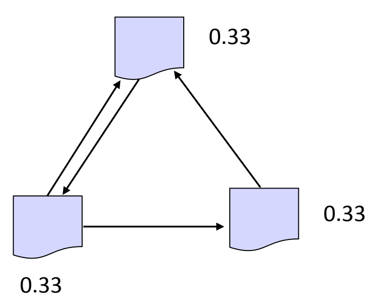
No need to decide how many levels to consider!

103

Example: Step 0

Initialize all ranks to be equal

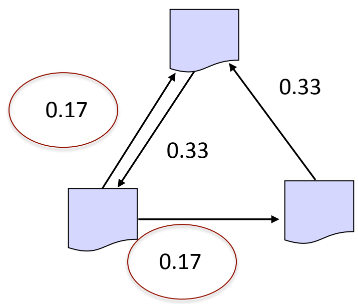
$$x_i^{(0)} = \frac{1}{n}$$



104

Example: Step 1

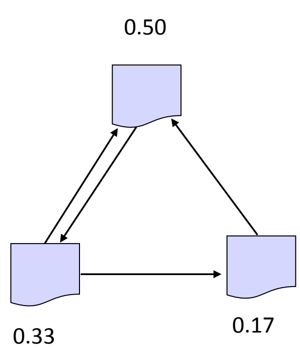
Propagate weights
across out-edges



Example: Step 2

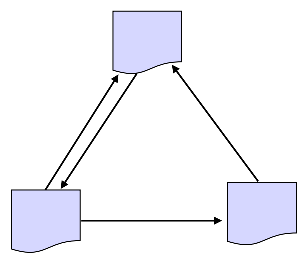
Compute weights
based on in-edges

$$x_i^{(1)} = \sum_{j \in B_i} \frac{1}{N_j} x_j^{(0)}$$



Example: Convergence

$$x_i^{(k+1)} = \sum_{j \in B_i} \frac{1}{N_j} x_j^{(k)}$$



Naïve PageRank Algorithm Restated

- Let
  - N(p) = number outgoing links from page p
  - B(p) = number of back-links to page p

$$PageRank(p) = \sum_{b \in B(p)} \frac{1}{N(b)} PageRank(b)$$

- Each page b *distributes its importance* to all of the pages it points to (so we scale by 1/N(b))
- Page p's importance is increased by the *importance of its back set*

Linear Algebra formulation

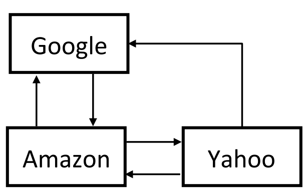
- Create an m x m matrix M to capture links:
  - M(i, j) = 1 / n_j if page i is pointed to by page j and page j has n_j outgoing links
    - = 0 otherwise
  - Initialize all PageRanks to 1, multiply by M repeatedly until all values converge:

$$\begin{bmatrix} PageRank(p_1') \\ PageRank(p_2') \\ \dots \\ PageRank(p_m') \end{bmatrix} = M \begin{bmatrix} PageRank(p_1) \\ PageRank(p_2) \\ \dots \\ PageRank(p_m) \end{bmatrix}$$

  - Computes principal eigenvector via power iteration

109

Example



$$\begin{vmatrix} g' \\ y' \\ a' \end{vmatrix} = \begin{vmatrix} 0 & 0.5 & 0.5 \\ 0 & 0 & 0.5 \\ 1 & 0.5 & 0 \end{vmatrix} * \begin{vmatrix} g \\ y \\ a \end{vmatrix}$$

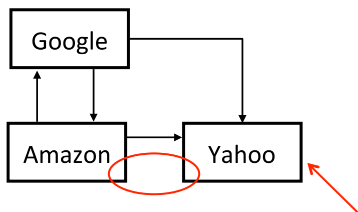
Running for multiple iterations:

$$\begin{vmatrix} g \\ y \\ a \end{vmatrix} = \begin{vmatrix} 1 \\ 1 \\ 1 \end{vmatrix}, \begin{vmatrix} 1 \\ 0.5 \\ 1.5 \end{vmatrix}, \begin{vmatrix} 1 \\ 0.75 \\ 1.25 \end{vmatrix}, \dots \quad \begin{vmatrix} 1 \\ 0.67 \\ 1.33 \end{vmatrix}$$

Total rank sums to number of pages

110

PageRank Sinks



'dead end' - PageRank is lost after each round

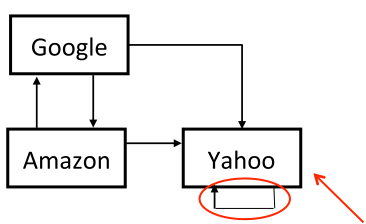
$$\begin{vmatrix} g' \\ y' \\ a' \end{vmatrix} = \begin{vmatrix} 0 & 0 & 0.5 \\ 0.5 & 0 & 0.5 \\ 0.5 & 0 & 0 \end{vmatrix} * \begin{vmatrix} g \\ y \\ a \end{vmatrix}$$

Running for multiple iterations:

$$\begin{vmatrix} g \\ y \\ a \end{vmatrix} = \begin{vmatrix} 1 \\ 1 \\ 1 \end{vmatrix}, \begin{vmatrix} 0.5 \\ 1 \\ 0.5 \end{vmatrix}, \begin{vmatrix} 0.25 \\ 0.5 \\ 0.25 \end{vmatrix}, \dots, \begin{vmatrix} 0 \\ 0 \\ 0 \end{vmatrix}$$

111

PageRank Hogs



PageRank cannot flow out and accumulates

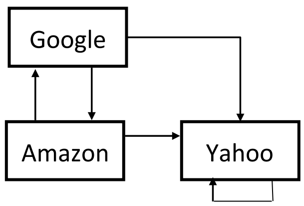
$$\begin{vmatrix} g' \\ y' \\ a' \end{vmatrix} = \begin{vmatrix} 0 & 0 & 0.5 \\ 0.5 & 1 & 0.5 \\ 0.5 & 0 & 0 \end{vmatrix} * \begin{vmatrix} g \\ y \\ a \end{vmatrix}$$

Running for multiple iterations:

$$\begin{vmatrix} g \\ y \\ a \end{vmatrix} = \begin{vmatrix} 1 \\ 1 \\ 1 \end{vmatrix}, \begin{vmatrix} 0.5 \\ 2 \\ 0.5 \end{vmatrix}, \begin{vmatrix} 0.25 \\ 2.5 \\ 0.25 \end{vmatrix}, \dots, \begin{vmatrix} 0 \\ 3 \\ 0 \end{vmatrix}$$

112

Stopping the Hog



$$\begin{vmatrix} g' \\ y' \\ a' \end{vmatrix} = 0.85 \begin{vmatrix} 0 & 0 & 0.5 \\ 0.5 & 1 & 0.5 \\ 0.5 & 0 & 0 \end{vmatrix} * \begin{vmatrix} g \\ y \\ a \end{vmatrix} + \begin{vmatrix} 0.15 \\ 0.15 \\ 0.15 \end{vmatrix}$$

Running for multiple iterations:

$$\begin{vmatrix} g \\ y \\ a \end{vmatrix} = \begin{vmatrix} 0.57 \\ 1.85 \\ 0.57 \end{vmatrix}, \begin{vmatrix} 0.39 \\ 2.21 \\ 0.39 \end{vmatrix}, \begin{vmatrix} 0.32 \\ 2.36 \\ 0.32 \end{vmatrix}, \ldots, \begin{vmatrix} 0.26 \\ 2.48 \\ 0.26 \end{vmatrix}$$

… *though does this seem right?*

113

Improved PageRank

- Remove out-degree 0 nodes
- Add decay factor d to deal with sinks

$$PageRank(p) = (1-d) + d \sum_{b \in B_p} \frac{1}{N(b)} PageRank(b)$$

- Typical value: d=0.85
- "Random surfer" Intuition:
  - Surfer occasionally stops following link sequence and jumps to new random page, with probability 1 - d

114

PageRank on MapReduce

- Inputs
  - page → <currentWeightOfPage, {adjacency list}>
- Map
  - Page p "propagates" $1/N_p$ of its d * weight(p) to the destinations of its out-edges (think like a vertex!)
- Reduce
  - p-th page sums the incoming weights and adds (1-d), to get its weight'(p)
- Iterate until convergence

115

PageRank on MapReduce

- Iterate until convergence
  - Common practice: run some fixed number of times, e.g., 25x
  - Alternatively: Test after each iteration with a second MapReduce job, to determine the maximum change between old and new weights

116

Conclusions

- Common kinds of algorithms used on the Web
  - Path analysis
  - Clustering and classification
  - Link analysis
- Straightforward, often iterative, MapReduce formulation

117