Journey
Dynamics

MapReduce with Apache Hadoop

Analysing Big Data

April 2010
Gavin Heavyside
gavin.heavyside@journeydynamics.com

FROST & SULLIVAN

2009 BEST PRACTICES AWARD

European Telematics & Navigation
Product Innovation Award

About Journey Dynamics

- Founded in 2006 to develop software technology to address the issues of congestion, fuel efficiency, driving safety and eco-driving
- Based in the Surrey Technology Centre, Guildford, UK
- Analyse large amounts (TB) of GPS data from cars, vans & trucks
- TrafficSpeedsEQ® - Accurate traffic speed forecasts by hour of day and day of week for every link in the road network
- MyDrive® - Unique & sophisticated system that learns how drivers behave

  Drivers can improve fuel economy

  Insurance companies can understand driver risk

  Navigation devices can improved route choice & ETA

  Fleet managers can monitor their fleet to improve safety & eco-driving

Big Data

- Data volumes increasing
- NYSE: 1TB new trade data/day
- Google: Processes 20PB/day (Sep 2007) http://tcrn.ch/agYjEL
- LHC: 15PB data/year
- Facebook: several TB photos uploaded/day

“Medium” Data

- Most of us aren’t at Google or Facebook scale
- But: data at the GB/TB scale is becoming more common
- Outgrow conventional databases
- Disks are cheap, but slow

- 1TB drive - £50
- 2.5 hours to read 1TB at 100MB/s

Two Challenges

- Managing lots of data
- Doing something useful with it

Managing Lots of Data

- Access and analyse any or all of your data
- SAN technologies (FC, iSCSI, NFS)
- Querying (MySQL, PostgreSQL, Oracle)

➡ Cost, network bandwidth, concurrent access, resilience

➡ When you have 1000s of nodes, MTBF < 1 day

Analysing Lots of Data

- Parallel processing
- HPC
- Grid Computing
- MPI
- Sharding

➡ Too big for memory, specialised HW, complex, scalability

➡ Hardware reliability in large clusters

Apache Hadoop

- Reliable, scalable distributed computing platform

- HDFS - high throughput fault-tolerant distributed file system
- MapReduce - fault-tolerant distributed processing

- Runs on commodity hardware
- Cost-effective

- Open source (Apache License)

Hadoop History

- 2003-2004 Google publishes MapReduce & GFS papers
- 2004 Doug Cutting add DFS & MapReduce to Nutch
- 2006 Cutting joins Yahoo!, Hadoop moves out of Nutch
- Jan 2008 - top level Apache project
- April 2010: 95 companies on PoweredBy Hadoop wiki
- Yahoo!, Twitter, Facebook, Microsoft, New York Times, LinkedIn, Last.fm, IBM, Baidu, Adobe

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

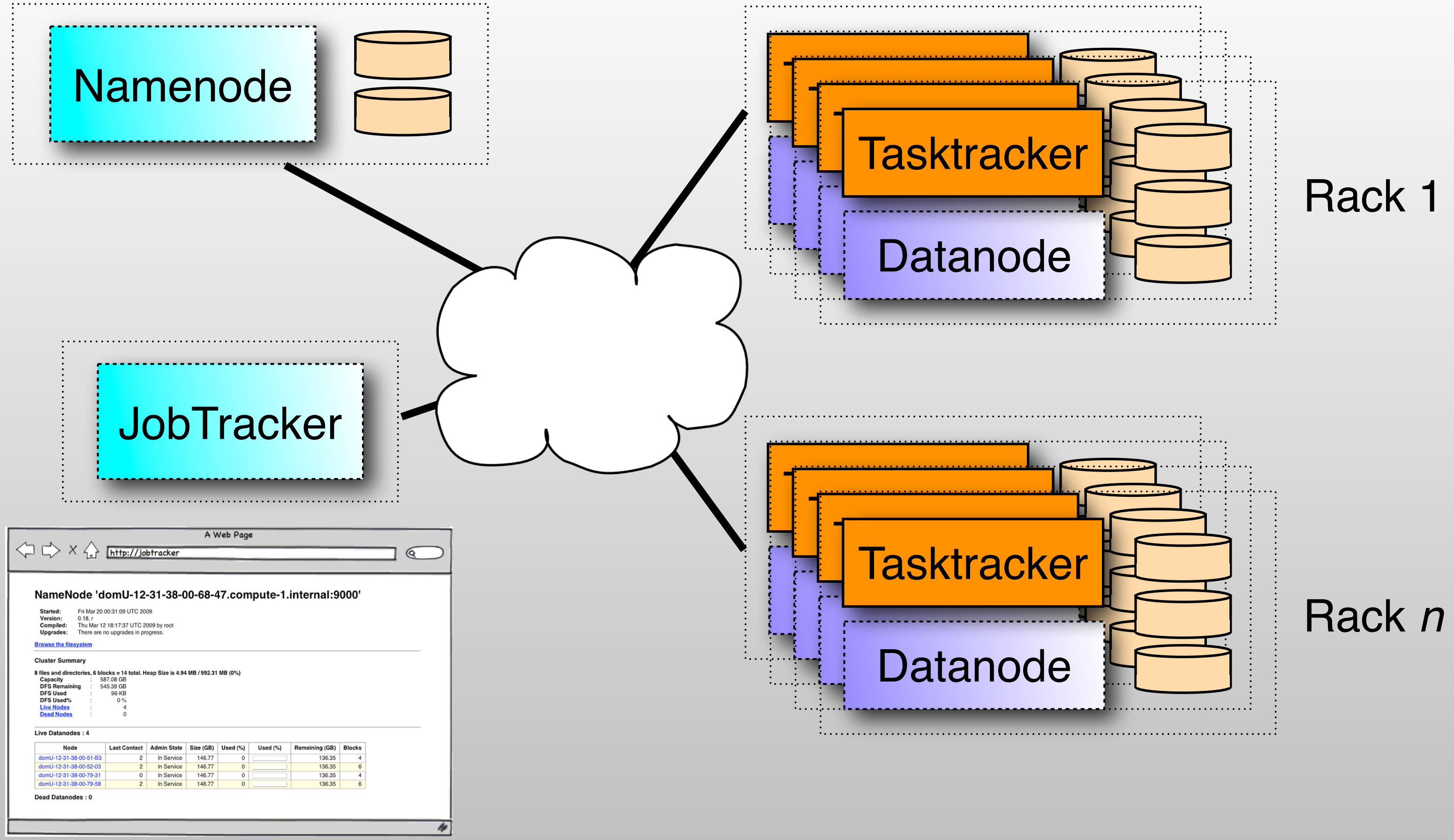
Hadoop Ecosystem

- **HDFS**
- **MapReduce**

- HBase
- ZooKeeper
- **Pig**
- **Hive**

- Chukwa
- Avro

Anatomy of a Hadoop Cluster



NameNode 'domU-12-31-38-00-68-47.compute-1.internal:9000'

Started: Fri Mar 20 00:31:09 UTC 2009
Version: 0.18, r
Compiled: Thu Mar 12 18:17:37 UTC 2009 by root
Upgrades: There are no upgrades in progress.

Browse the filesystem

Cluster Summary

8 files and directories, 6 blocks = 14 total. Heap Size is 4.94 MB / 992.31 MB (0%)

Capacity	:	587.08 GB
DFS Remaining	:	545.39 GB
DFS Used	:	96 KB
DFS Used%	:	0 %
Live Nodes	:	4
Dead Nodes	:	0

Live Datanodes : 4

Node	Last Contact	Admin State	Size (GB)	Used (%)	Used (%)	Remaining (GB)	Blocks
domU-12-31-38-00-51-B3	2	In Service	146.77	0		136.35	4
domU-12-31-38-00-52-03	2	In Service	146.77	0		136.35	6
domU-12-31-38-00-79-31	0	In Service	146.77	0		136.35	4
domU-12-31-38-00-79-58	2	In Service	146.77	0		136.35	6

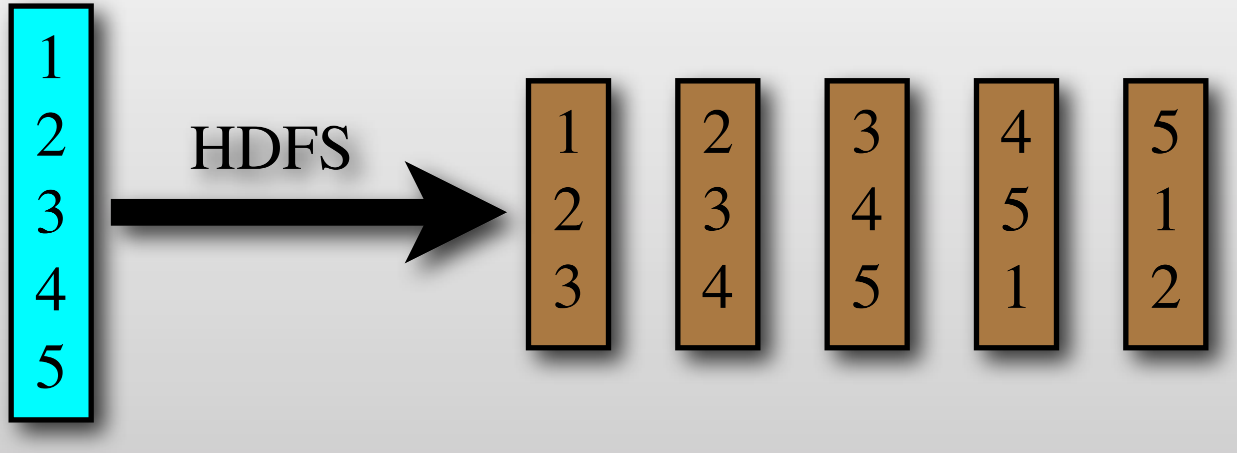
Dead Datanodes : 0

HDFS

- Reliable shared storage
- Modelled after GFS
- Very large files
- Streaming data access
- Commodity Hardware
- Replication
- Tolerate regular hardware failure

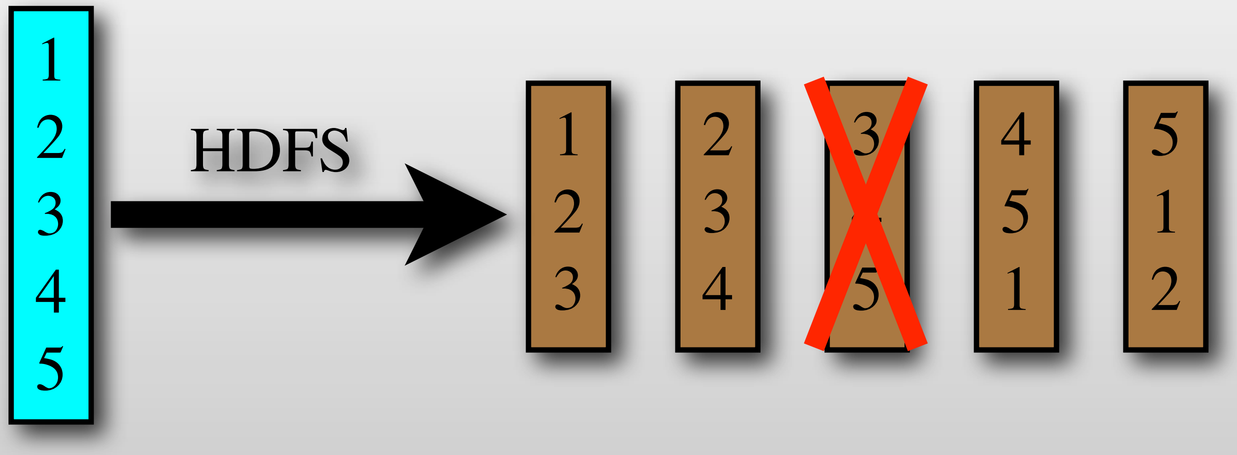
HDFS

- Block size 64MB
- Default replication factor = 3



HDFS

- Block size 64MB
- Default replication factor = 3



MapReduce

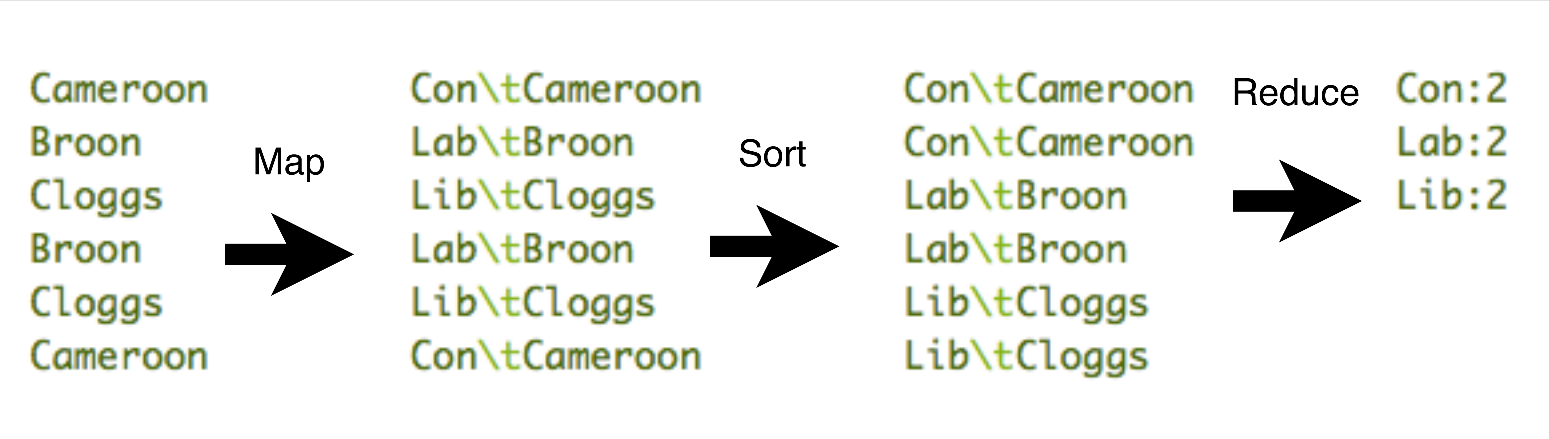
- Based on 2004 Google paper
- Concepts from Functional Programming
- Used for lots of things within Google (and now everywhere)
- Parallel Map => Shuffle & Sort => Parallel Reduce
- Easy to understand and write MapReduce programs
- Move the computation to the data
- Rack-aware
- Linear Scalability
- Works with HDFS, S3, KFS, file:// and more

MapReduce

- “Single Threaded” MapReduce:

```
cat input/* | map | sort | reduce > output
```

- Map program parses the input and emits [key,value] pairs
- Sort by key
- Reduce computes output from values with same key



```
Cameroon
Broon
Cloggs
Broon
Cloggs
Cameroon
```

Map

```
Con\tCameroon
Lab\tBroon
Lib\tCloggs
Lab\tBroon
Lib\tCloggs
Con\tCameroon
```

Sort

```
Con\tCameroon
Con\tCameroon
Lab\tBroon
Lab\tBroon
Lib\tCloggs
Lib\tCloggs
```

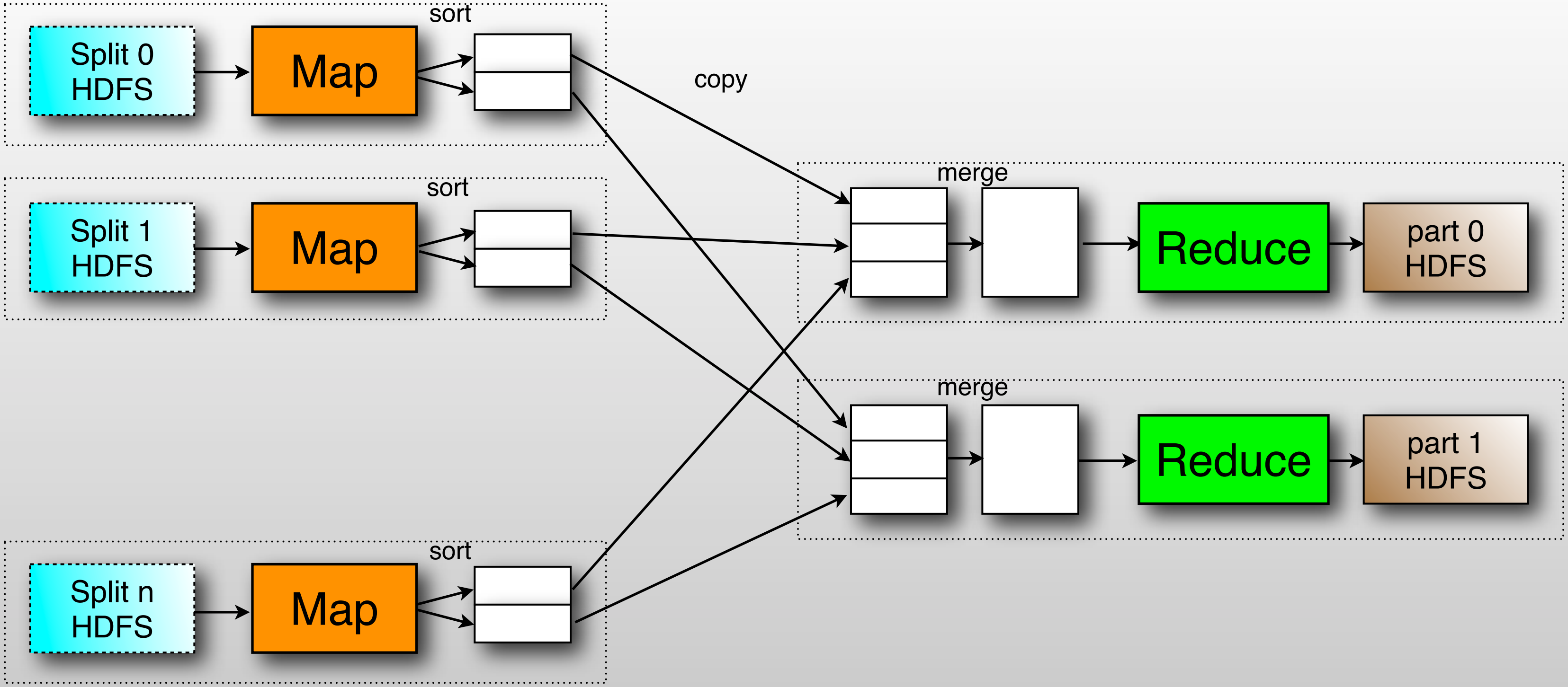
Reduce

```
Con:2
Lab:2
Lib:2
```

- Extrapolate to PB of data on thousands of nodes

MapReduce

- Distributed Example



Sunday, 30 May 2010

MapReduce can be good for:

- “Embarrassingly Parallel” problems
- Semi-structured or unstructured data
- Index generation
- Log analysis
- Statistical analysis of patterns in data
- Image processing
- Generating map tiles
- Data Mining
- Much, much more

MapReduce is not be good for:

- Real-time or low-latency queries
- Some graph algorithms
- Algorithms that can’t be split into independent chunks
- Some types of joins*
- Not a replacement for RDBMS

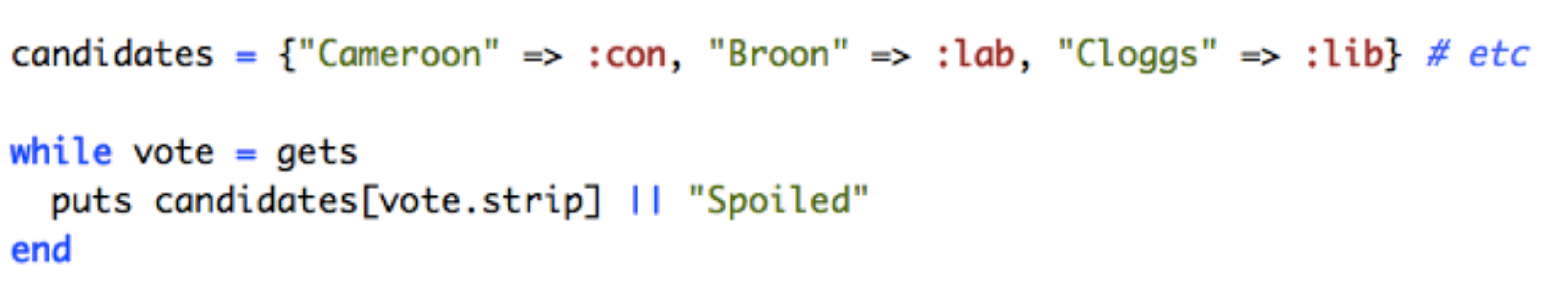
* Can be tricky to write unless you use an abstraction e.g. Pig, Hive

Writing MapReduce Programs

- Java
- Pipes (C++, sockets)
- **Streaming**
- Frameworks, e.g. wukong(ruby), dumbo(python)
- JVM languages e.g. JRuby, Clojure, Scala
- Cascading.org
- Cascalog
- **Pig**
- **Hive**

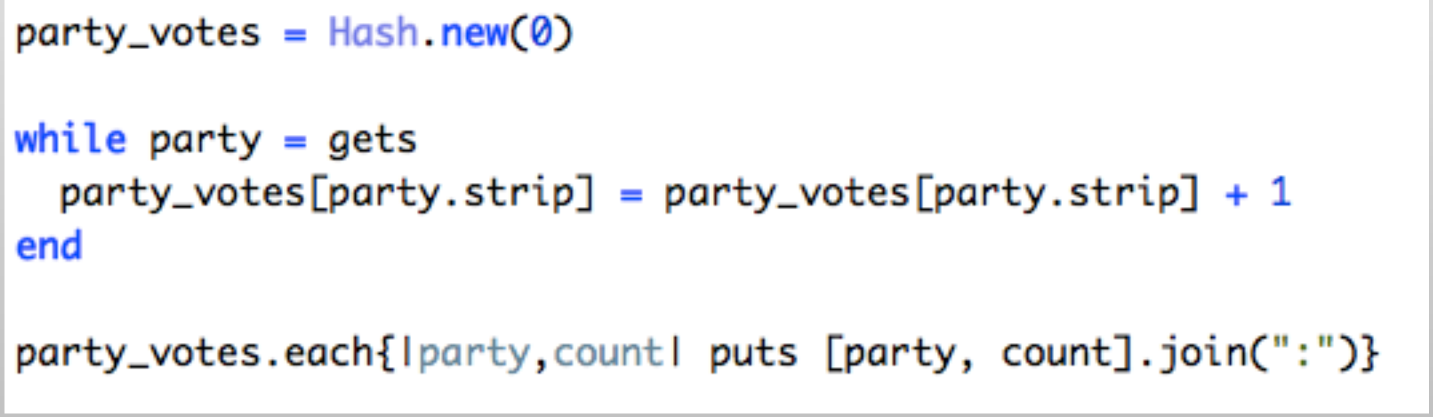
Streaming Example (ruby)

- mapper.rb



```ruby
candidates = {"Cameroon" => :con, "Broon" => :lab, "Cloggs" => :lib} # etc

while vote = gets
  puts candidates[vote.strip] || "Spoiled"
end
```

- reducer.rb

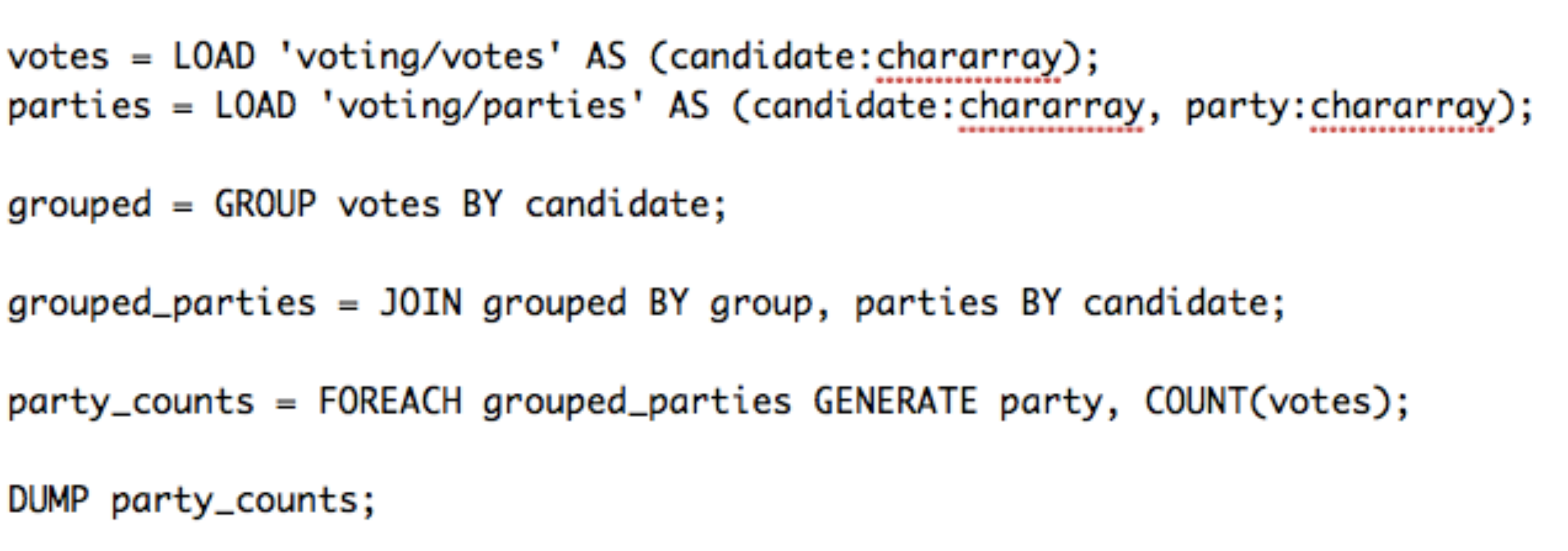


```ruby
party_votes = Hash.new(0)

while party = gets
  party_votes[party.strip] = party_votes[party.strip] + 1
end

party_votes.each{|party,count| puts [party, count].join(":")}
```

Pig

- High level language for writing data analysis programs
- Runs MapReduce jobs
- Joins, grouping, filtering, sorting, statistical functions
- User-defined functions
- Optional schemas
- Sampling
- Pig Latin similar to imperative language, define steps to run

Pig Example

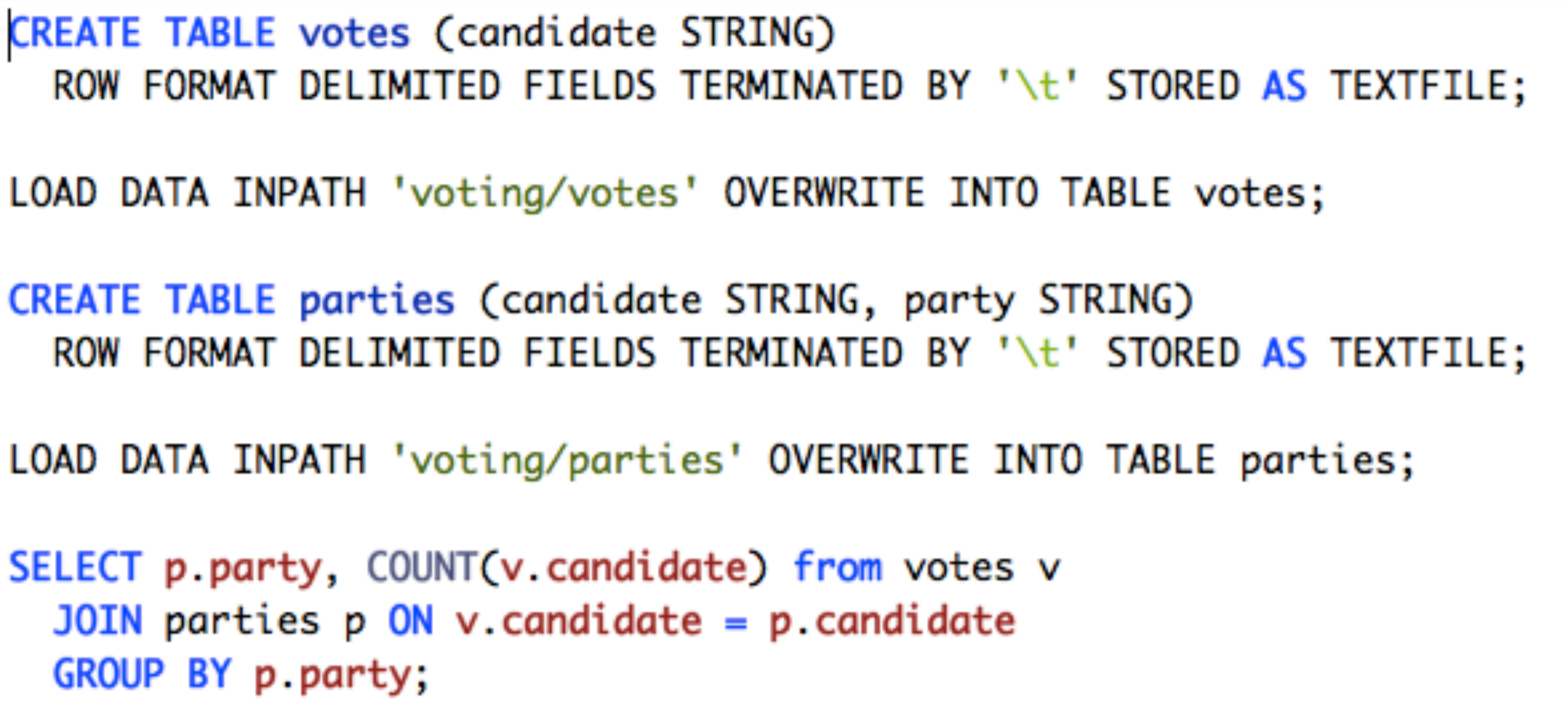


```
votes = LOAD 'voting/votes' AS (candidate:chararray);
parties = LOAD 'voting/parties' AS (candidate:chararray, party:chararray);

grouped = GROUP votes BY candidate;

grouped_parties = JOIN grouped BY group, parties BY candidate;

party_counts = FOREACH grouped_parties GENERATE party, COUNT(votes);

DUMP party_counts;
```

Hive

- Data warehousing and querying
- HiveQL - SQL-like language for querying data
- Runs MapReduce jobs
- Joins, grouping, filtering, sorting, statistical functions
- Partitioning of data
- User-defined functions
- Sampling
- Declarative syntax

Hive Example



```
CREATE TABLE votes (candidate STRING)
  ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' STORED AS TEXTFILE;

LOAD DATA INPATH 'voting/votes' OVERWRITE INTO TABLE votes;

CREATE TABLE parties (candidate STRING, party STRING)
  ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' STORED AS TEXTFILE;

LOAD DATA INPATH 'voting/parties' OVERWRITE INTO TABLE parties;

SELECT p.party, COUNT(v.candidate) from votes v
  JOIN parties p ON v.candidate = p.candidate
  GROUP BY p.party;
```

Getting Started

- http://hadoop.apache.org
- Cloudera Distribution (VM, source, rpm, deb)
- Elastic MapReduce

- Cloudera VM
- Pseudo-distributed cluster

Learn More

- http://hadoop.apache.org
- Books

- Mailing Lists
- Commercial Support & Training, e.g. Cloudera

Related

- Cassandra 0.6 has Hadoop integration - run MapReduce jobs against data in Cassandra
- NoSQL DBs with MapReduce functionality include CouchDB, MongoDB, Riak and more
- RDBMS with MapReduce include Aster, Greenplum, HadoopDB and more

Gavin Heavyside

gavin.heavyside@journeydynamics.com

www.journeydynamics.com