(Mis)Use of Hadoop

John Langford

NYU Large Scale Learning Class, February 19, 2013

(Post presentation version)

The data problem

Traditional high performance computing is FLoating Operations Per Second. http://top500.org/lists/2012/11/
Titan http://www.olcf.ornl.gov/titan/: 17.6PFlops, 8.2MW
Flops first, network second, data is irrelevant.
Primary use = simulations.

The data problem

Traditional high performance computing is FLoating Operations Per Second. http://top500.org/lists/2012/11/
Titan http://www.olcf.ornl.gov/titan/: 17.6PFlops, 8.2MW
Flops first, network second, data is irrelevant.
Primary use = simulations.
Data is different.
$\Rightarrow$ the need to store information. With large amounts of data, errors should be assumed. Use replication to zero out the chance of error.
$\Rightarrow$ the need for quick access. No single machine can handle all data requests. Use locality to minimize bandwidth.

Hadoop

1. System for datacentric computing.

Hadoop

1. System for datacentric computing.
2. Distributed File System + Map Reduce + Advanced components

Hadoop

1. System for datacentric computing.
2. Distributed File System + Map Reduce + Advanced components
3. Java's first serious use as an OS

Hadoop

1. System for datacentric computing.
2. Distributed File System + Map Reduce + Advanced components
3. Java's first serious use as an OS
4. Open Source clone of GFS + Map Reduce system.

Hadoop

1. System for datacentric computing.
2. Distributed File System + Map Reduce + Advanced components
3. Java's first serious use as an OS
4. Open Source clone of GFS + Map Reduce system.
5. Yahoo!'s bulk data processing system.

Hadoop

1. System for datacentric computing.
2. Distributed File System + Map Reduce + Advanced components
3. Java's first serious use as an OS
4. Open Source clone of GFS + Map Reduce system.
5. Yahoo!'s bulk data processing system.
6. The craze at Strata (= data business conference). Nearly every company is interoperating with, extending, and/or using Hadoop.

The NYU Hadoop cluster

~92 machines
8 cores @ 2Ghz to 2.5Ghz / machine
16GB RAM / machine
1Gb/s network card
~100TB storage (in Hadoop).
A low end Hadoop cluster, of most use as a shared datarich environment.

The NYU Hadoop cluster

~92 machines
8 cores @ 2Ghz to 2.5Ghz / machine
16GB RAM / machine
1Gb/s network card
~100TB storage (in Hadoop).
A low end Hadoop cluster, of most use as a shared datarich environment.
Access directions for NYU students:
`ssh <netid>@hpc.nyu.edu`
`ssh dumbo.es.its.nyu.edu`
Take 10 minutes to setup ssh tunneling:
https://wikis.nyu.edu/display/NYUHPC/SCP+through+SSH+Tunneling

The NYU Hadoop cluster

~92 machines
8 cores @ 2Ghz to 2.5Ghz / machine
16GB RAM / machine
1Gb/s network card
~100TB storage (in Hadoop).
A low end Hadoop cluster, of most use as a shared datarich environment.
Access directions for NYU students:
`ssh <netid>@hpc.nyu.edu`
`ssh dumbo.es.its.nyu.edu`
Take 10 minutes to setup ssh tunneling:
https://wikis.nyu.edu/display/NYUHPC/SCP+through+SSH+Tunneling
For nonNYU students, you can experiment with Hadoop easily using AWS.

Hadoop Distributed File System (HDFS)

Half of Hadoop is HDFS. It's the better half.

Hadoop Distributed File System (HDFS)

Half of Hadoop is HDFS. It's the better half.
All data is stored 3 times $\Rightarrow$

1. robust to failures. How many random failures required to lose info?
2. 1/3 storage.
3. You don't "backup" a Hadoop cluster.
4. Multiple sources for any one piece of data.

Hadoop Distributed File System (HDFS)

Half of Hadoop is HDFS. It's the better half.
All data is stored 3 times $\Rightarrow$

1. robust to failures. How many random failures required to lose info?
2. 1/3 storage.
3. You don't "backup" a Hadoop cluster.
4. Multiple sources for any one piece of data.

Files are stored in 64MB chunks ("shards").

1. Sequential reads are fast. 100MB/s disk requires 0.64s to read a chunk but only .01 s to start reading it.
2. Not made for random reads.
3. Not for small files.

Hadoop Distributed File System (HDFS)

Half of Hadoop is HDFS. It's the better half.
All data is stored 3 times $\Rightarrow$

1. robust to failures. How many random failures required to lose info?
2. 1/3 storage.
3. You don't "backup" a Hadoop cluster.
4. Multiple sources for any one piece of data.

Files are stored in 64MB chunks ("shards").

1. Sequential reads are fast. 100MB/s disk requires 0.64s to read a chunk but only .01 s to start reading it.
2. Not made for random reads.
3. Not for small files.

No support for file modification.
$\Rightarrow$ HDFS is in it's own namespace.
$\Rightarrow$ Need new comands.

HDFS ops

Execute:

```
echo alias hfs= 'hadoop fs ' >> .bashrc
source .bashrc
```

Common commands:

1. `hfs` See available commands.
2. `hfs -help` more command details.
3. `hfs -ls [<path>]` List files
4. `hfs -cp <src> <dst>` Copy stuff
5. `hfs -mkdir <path>` Create path
6. `hfs -rm <path>` remove a file
7. `hfs -chmod <path>` Modify permissions.
8. `hfs -chown <path>` Modify owner.

HDFS ops

Execute:

```
echo alias hfs= 'hadoop fs ' >> .bashrc
source .bashrc
```

Common commands:

1. `hfs` See available commands.
2. `hfs -help` more command details.
3. `hfs -ls [<path>]` List files
4. `hfs -cp <src> <dst>` Copy stuff
5. `hfs -mkdir <path>` Create path
6. `hfs -rm <path>` remove a file
7. `hfs -chmod <path>` Modify permissions.
8. `hfs -chown <path>` Modify owner.

Remote access commands:

1. `hfs -cat <src>` Cat contents to stdout.
2. `hfs -copyFromLocal <localsrc> <dst>` Copy stuff
3. `hfs -copyToLocal <src> <localdst>` Copy stuff

File system is browsable. For NYU: `http://babar:50070/`

Hadoop Map-Reduce

An example of "Bulk Synchronous Parallel" data processing.
Map (Programming language ideal): A function $f : A \rightarrow B$.

Reduce (Programming language ideal): A function $g : BxB \rightarrow B$.

Hadoop Map-Reduce

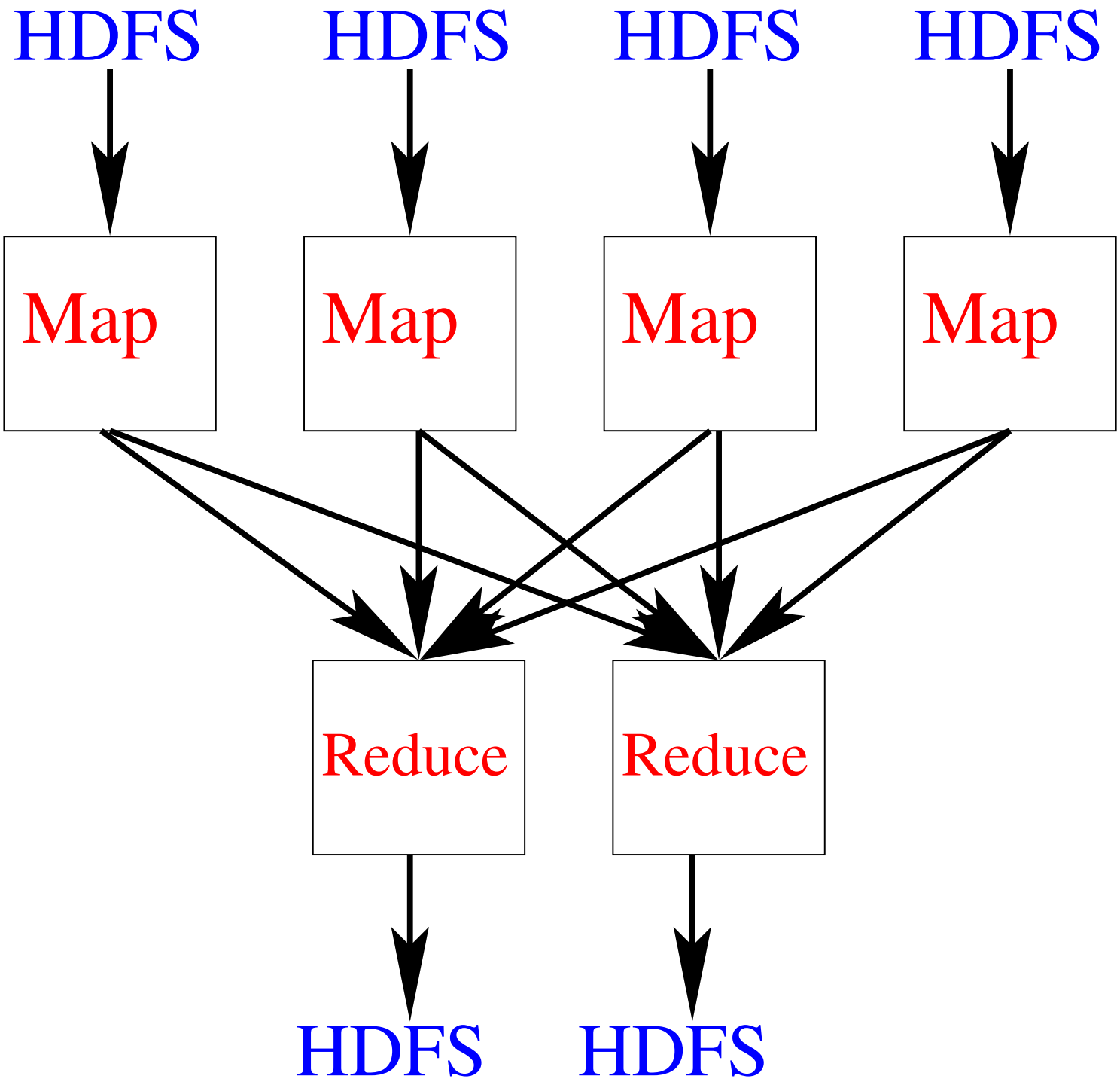
An example of "Bulk Synchronous Parallel" data processing.
Map (Programming language ideal): A function $f : A \rightarrow B$.
Map (Hadoop ideal): A function $f : A^* \rightarrow B^*$

In between Map and Reduce is sort(B^*) which partitions elements across multiple reducers.
Reduce (Programming language ideal): A function $g : BxB \rightarrow B$.
Reduce (Hadoop ideal): A function $g : B^* \rightarrow C$.

A, B, C are often (but not always) line oriented.

Hadoop Map-Reduce

An example of "Bulk Synchronous Parallel" data processing.
Map (Programming language ideal): A function $f : A \to B$.
Map (Hadoop ideal): A function $f : A^* \to B^*$
Map (Real implementation): Any program consuming A^* and outputting B^*.
In between Map and Reduce is sort(B^*) which partitions elements across multiple reducers.
Reduce (Programming language ideal): A function $g : BxB \to B$.
Reduce (Hadoop ideal): A function $g : B^* \to C$.
Reduce (Real Implementation): Any program consuming B^* and outputing C.
A, B, C are often (but not always) line oriented.


HDFS
HDFS
HDFS
HDFS
Map
Map
Map
Map
Reduce
Reduce
HDFS
HDFS

Hadoop Streaming

Hadoop streaming = use any program written in any language for mapreduce operations.
Execute:

```
echo "export HAS=/usr/lib/hadoop/contrib/streaming
export HSJ=hadoop-streaming-1.0.3.16.jar
alias hjs= 'hadoop jar $(HAS)/$(HSJ) '" >> .bashrc
source .bashrc
```

Hadoop Streaming

Hadoop streaming = use any program written in any language for mapreduce operations.
Execute:

```
echo "export HAS=/usr/lib/hadoop/contrib/streaming
export HSJ=hadoop-streaming-1.0.3.16.jar
alias hjs= 'hadoop jar $(HAS)/$(HSJ) '" >> .bashrc
source .bashrc
```

Get the first example from every mapped chunk in rcv1.

```
echo ``cat > temp; head -n 1 temp'' > header
hjs -input /user/jl5386/rcv1.train.txt -output
headresults -mapper header -reducer cat -file header
hfs -cat headresults/part-00000 | wc -l
```

= number of mappers

Guess what it does

```
echo ‘‘cut -d ’ ’ -f 1 | grep 1’’ > cutter
echo wc -l > counter
hjs -input /user/jl5386/rcv1.train.txt -output
countres -mapper cutter -reducer counter -file cutter
-file counter
hfs -cat countres/part-00000
```

Guess what it does

```
echo grep 326:  > grepper
echo wc -l > counter
hjs -input /user/jl5386/rcv1.train.txt -output fcount
-mapper grepper -reducer counter -file grepper -file
counter
hfs -cat fcount/part-00000
```

Guess what it does

```
echo ‘‘cut -d ’ ’ -f 1 | sort -u’’ > cutsort
echo sort -u > sorter
hjs -input /user/jl5386/rcv1.train.txt -output labels
-mapper cutsort -reducer sorter -file cutsort -file
sorter
hfs -cat labels/part-00000
```

Hadoop job control

Watch what is happening with job tracker URL (given on job launch).

```
hadoop job -list
```

```
hadoop job -kill <id>
```

Abusing Hadoop Streaming

Hadoop streaming makes Hadoop into a general purpose job submission system.

```
hjs -Dmapred.task.timeout=600000000
-Dmapred.job.map.memory.mb=3000 -input <yourdata>
-output <finaloutput> -mapper cat -reducer
<yourprogram> -file <yourprogram>
```

Abusing Hadoop Streaming

Hadoop streaming makes Hadoop into a general purpose job submission system.

```
hjs -Dmapred.task.timeout=600000000
-Dmapred.job.map.memory.mb=3000 -input <yourdata>
-output <finaloutput> -mapper cat -reducer
<yourprogram> -file <yourprogram>
```

Why Hadoop for job control?

1. map can be handy for selecting a subset or different data for features or examples.
2. much better bandwidth limits.

Parallel Learning for Parameter Search

Hadoop streaming makes Hadoop into a general purpose job submission system.

```
for a in 0.1 0.3 1 3 10; do

echo ./vw -l $a > job_$a

hjs -Dmapred.task.timeout=600000000
-Dmapred.job.map.memory.mb=3000 -input <yourdata>
-output output_$a -mapper cat -reducer job_$a -file
job_$a  -file vw
done
```

Parallel Learning for Speed

The next lecture.

More Hadoop things

PIG(Y!) is an SQL→MapReduce compiler

Zookeeper(Y!) is a system for sharing small amounts of info.

Hive(Facebook): Much faster data query and exploration. ...

References

[Hadoop Tutorial]
http://developer.yahoo.com/hadoop/tutorial/
[GFS] Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung, "The Google file system", SOSP '03.
[MapReduce] Jeffrey Dean, Sanjay Ghemawat MapReduce: simplified data processing on large clusters, Communications of the ACM, Volume 51 Issue 1, January 2008, Pages 107-113.

In general, MapReduce is an example of bulk synchronous parallel computation—there are many other papers.