Map-Reduce With Hadoop

Announcement 1/2

- Assignments, in general:
  - Autolab is not secure and assignments aren't designed for adversarial interactions
  - Our policy: **deliberately "gaming" an autograded assignment is considered cheating.**
    - The default penalty for cheating is failing the course.
  - Getting perfect test scores should not be possible: you're either cheating, or it's a bug.

Announcement 2/2

- Really a correction….

The Naïve Bayes classifier – v2

- You have a *train* dataset and a *test* dataset
- Initialize an "event counter" (hashtable) C
- For each example *id, y,* $x_1,\ldots,x_d$ in *train:*
  - C("*Y*=ANY") ++; C("*Y*=*y*") ++
  - For *j* in *1..d*:
    - C("*Y*=*y* ^ *X*=x_j") ++
- For each example *id, y,* $x_1,\ldots,x_d$ in *test:*
  - For each *y'* in *dom(Y):*
    - Compute log Pr(*y'*,$x_1,\ldots,x_d$) =

where:
$q_j = 1/|V|$
$q_y = 1/|dom(Y)|$
$m=1$

$$= \left(\sum_j \log \frac{C(X = x_j \wedge Y = y') + mq_x}{C(X = ANY \wedge Y = y') + m} \right) + \log \frac{C(Y = y') + mq_y}{C(Y = ANY) + m}$$

  - Return the best *y'*

The Naïve Bayes classifier – v2

- You have a *train* dataset and a *test* dataset
- Initialize an "event counter" (hashtable) C
- For each example *id, y,* $x_1,\ldots,x_d$ in *train:*
  - C("*Y*=ANY") ++; C("*Y*=*y*") ++
  - For *j* in *1..d*:
    - C("*Y*=*y* ^ *X*=x_j") ++
- For each example *id, y,* $x_1,\ldots,x_d$ in *test:*
  - For each *y'* in *dom(Y):*
    - Compute log Pr(*y'*,$x_1,\ldots,x_d$) =

where:
$q_j = 1/|V|$
$q_y = 1/|dom(Y)|$
$mq_x = 1$

$$= \left(\sum_j \log \frac{C(X = x_j \wedge Y = y') + mq_x}{C(X = ANY \wedge Y = y') + m} \right) + \log \frac{C(Y = y') + mq_y}{C(Y = ANY) + m}$$

  - Return the best *y'*

Today: from stream+sort to hadoop

- Looked at algorithms consisting of
  - Sorting (to organize messages)
  - Streaming (low-memory, line-by-line) file transformations ("map" operations)
  - Streaming "reduce" operations, like summing counts, that input files sorted by keys and operate on contiguous runs of lines with the same keys

- ➔ All our algorithms *could* be expressed as sequences of map-sort-reduce triples (allowing identity maps and reduces) operating on sequences of key-value pairs
- ➔ Likewise all the abstraction operations we discussed can be implemented as map-sort-reduce triples
- ➔ To parallelize: we can look at parallelizing these …

Recap: abstractions

// just a single streaming transform – ie, a map
// without a sort/reduce

table2 = MAP table1 TO λ row : f(row))
table2 = FILTER table1 BY λ row : f(row))
table2 = FLATMAP table1 TO λ row : f(row))

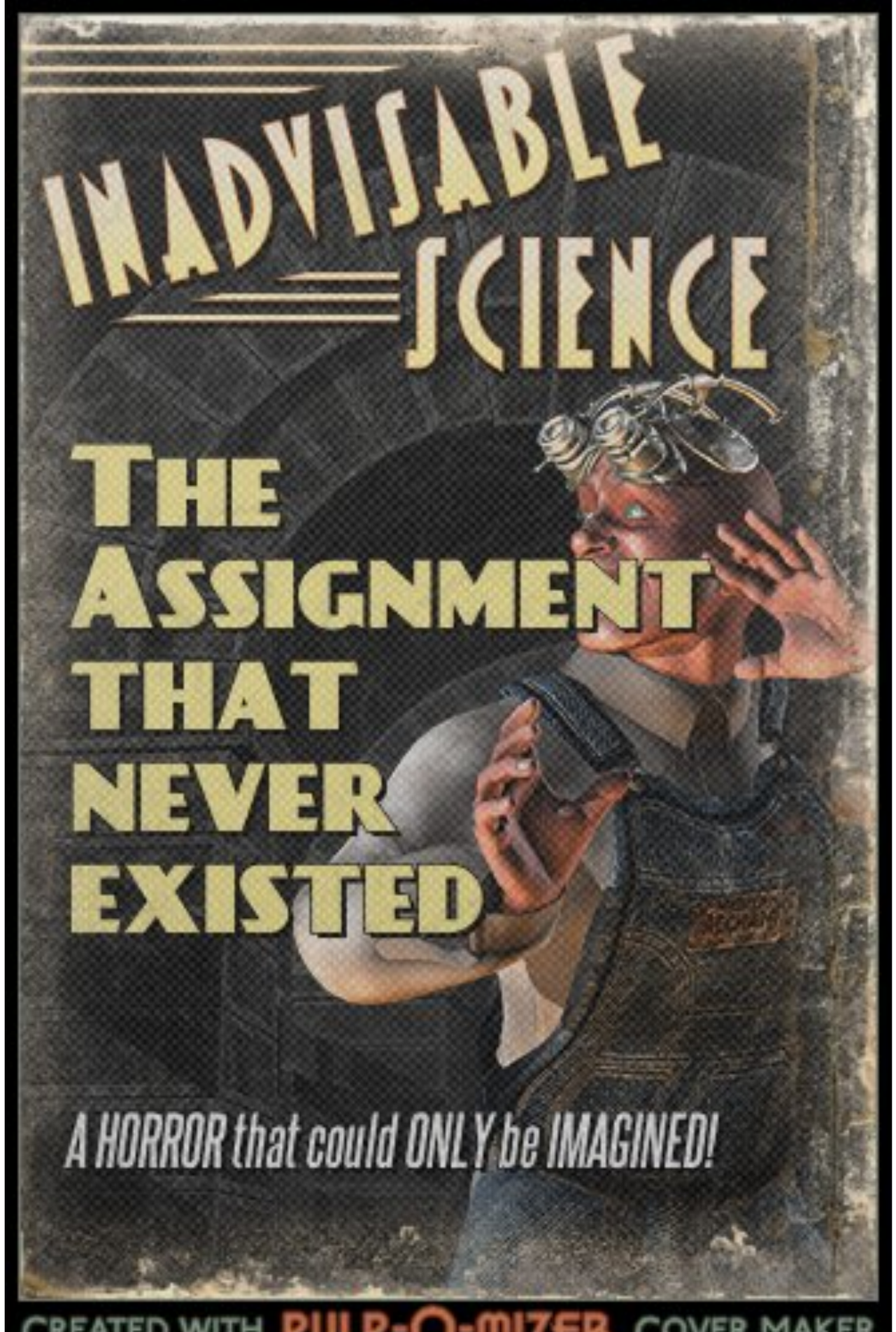
// map|sort|reduce
table2 = GROUP table1 BY λ row : f(row)

// map1+map2 | sort | reduce
table3 = JOIN table1 BY f1, table2 BY f2

Today: from stream+sort to hadoop

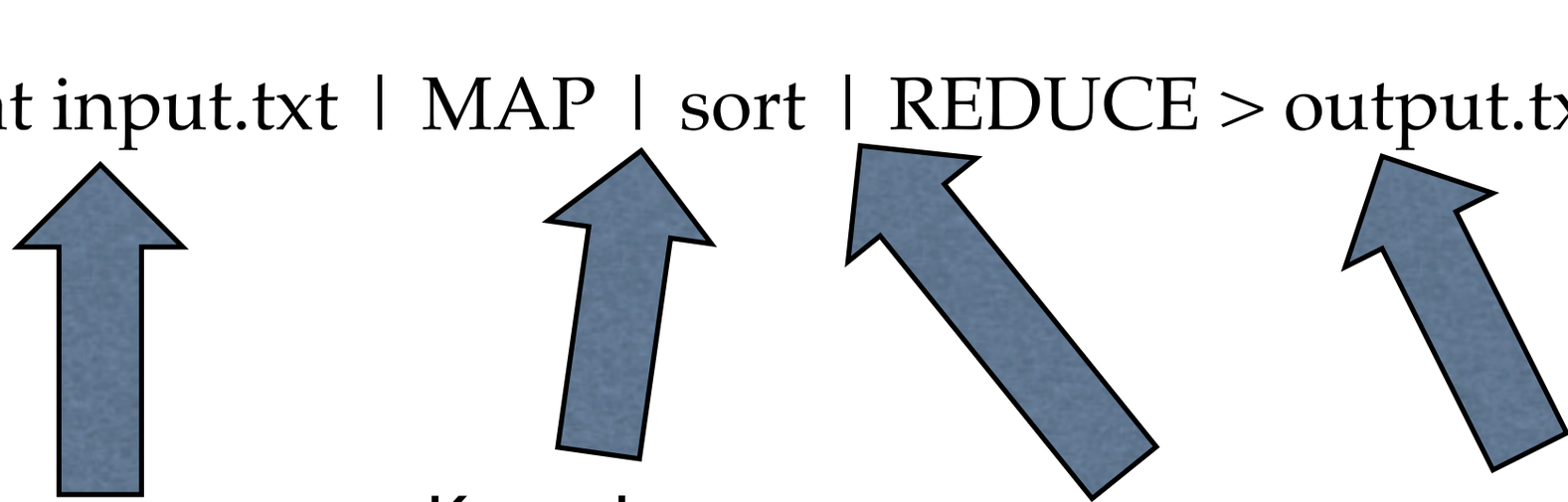
- Important point:
  - Our code is *not* CPU-bound
  - It's I/O bound
  - To speed it up, we need to add more *disk drives, not more CPUs*.
  - Example: finding a particular line in 1 TB of data

- ➔ Our algorithms *could* be expressed as sequences of map-sort-reduce triples (allowing identity maps and reduces) operating on sequences of key-value pairs
- ➔ To parallelize we can look at parallelizing these …


INADVISABLE SCIENCE
THE ASSIGNMENT THAT NEVER EXISTED
A HORROR that could ONLY be IMAGINED!
CREATED WITH PULP-O-MIZER COVER MAKER

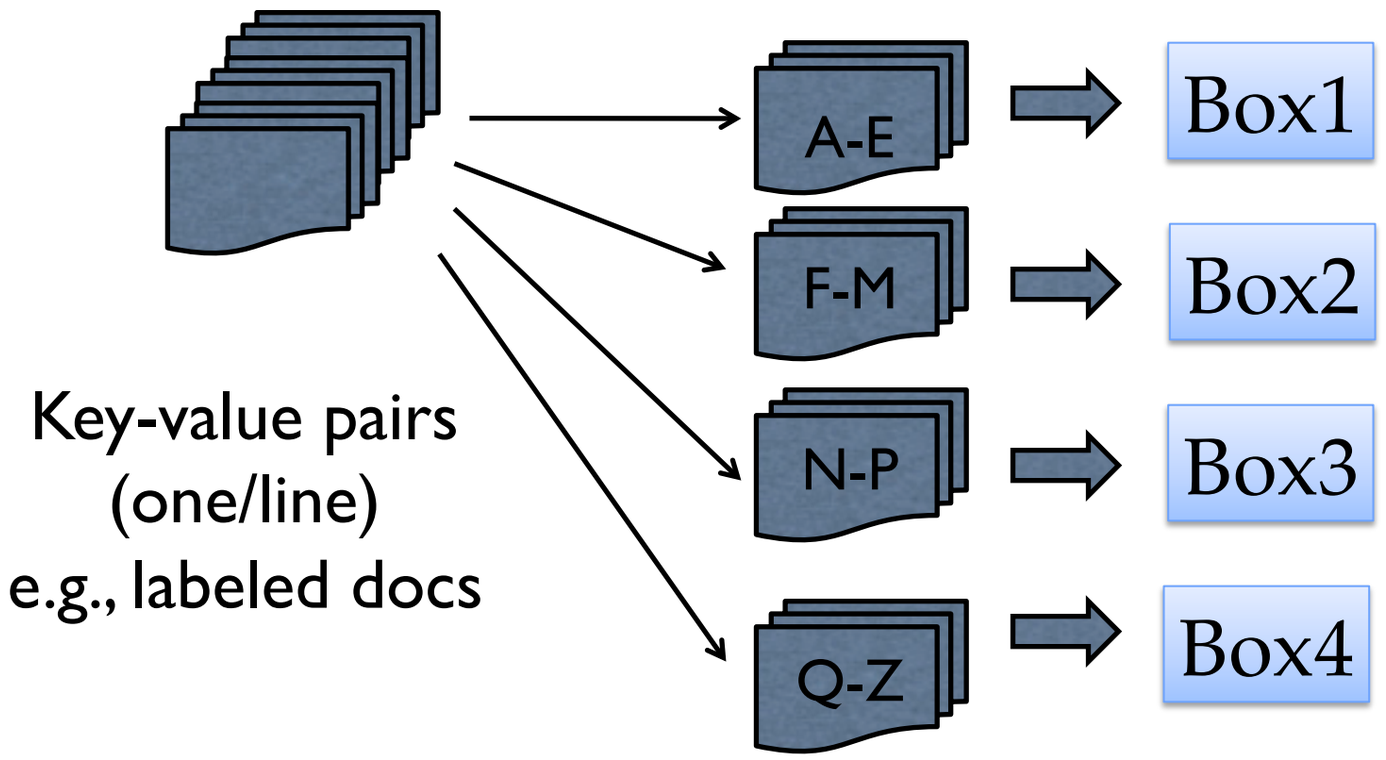
Write code to run assignment 1 in parallel

- What infrastructure would you need?
- How could you run a generic "stream-and-sort" algorithm in parallel?
- cat input.txt | MAP | sort | REDUCE > output.txt



How would you run assignment 1 in parallel?

- What infrastructure would you need?
- How could you run a generic "stream-and-sort" algorithm in parallel?
  - cat input.txt | MAP | sort | REDUCE > output.txt



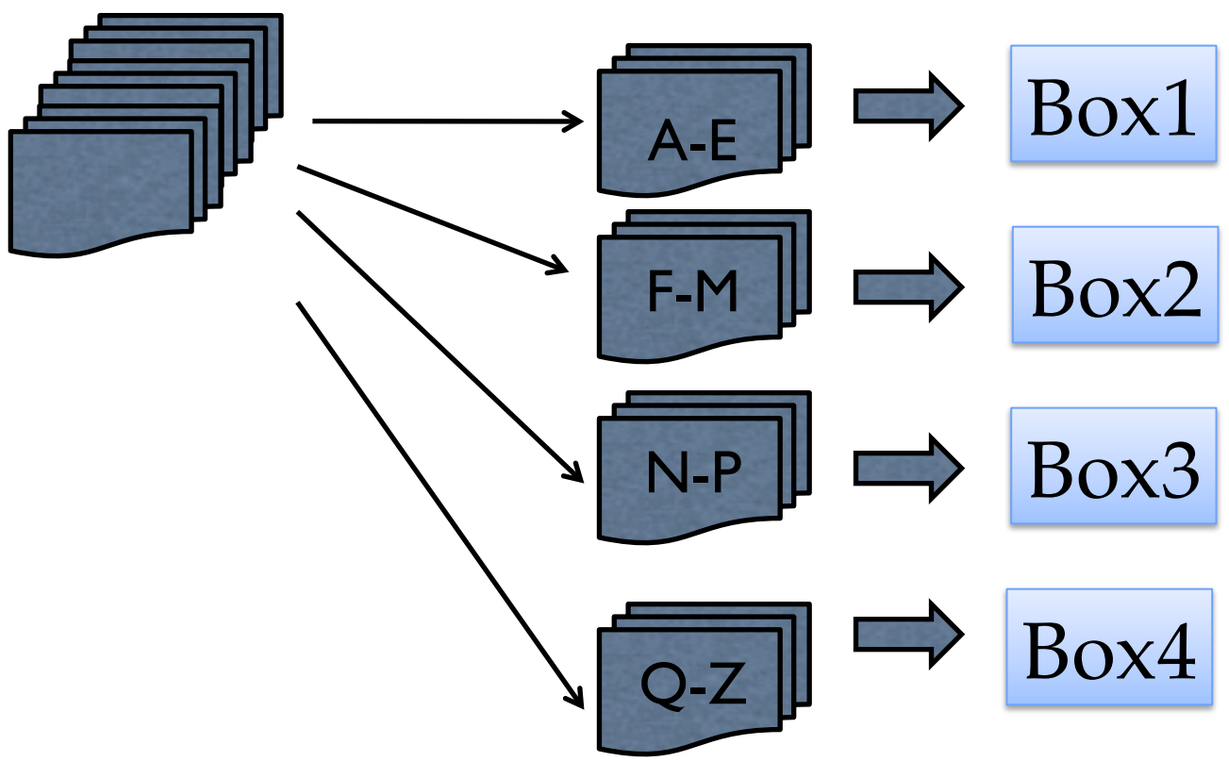
How would you run assignment 1 in parallel?

- What infrastructure w

- How could you run a parallel?

  - cat input.txt | MAP | sort | REDUCE > output.txt

•Open sockets to receive data to ~wcohen/kludge/mapinput.txt on each of the K boxes
•For each key,val pair:
  • Send key,val pair to boxFor (key)

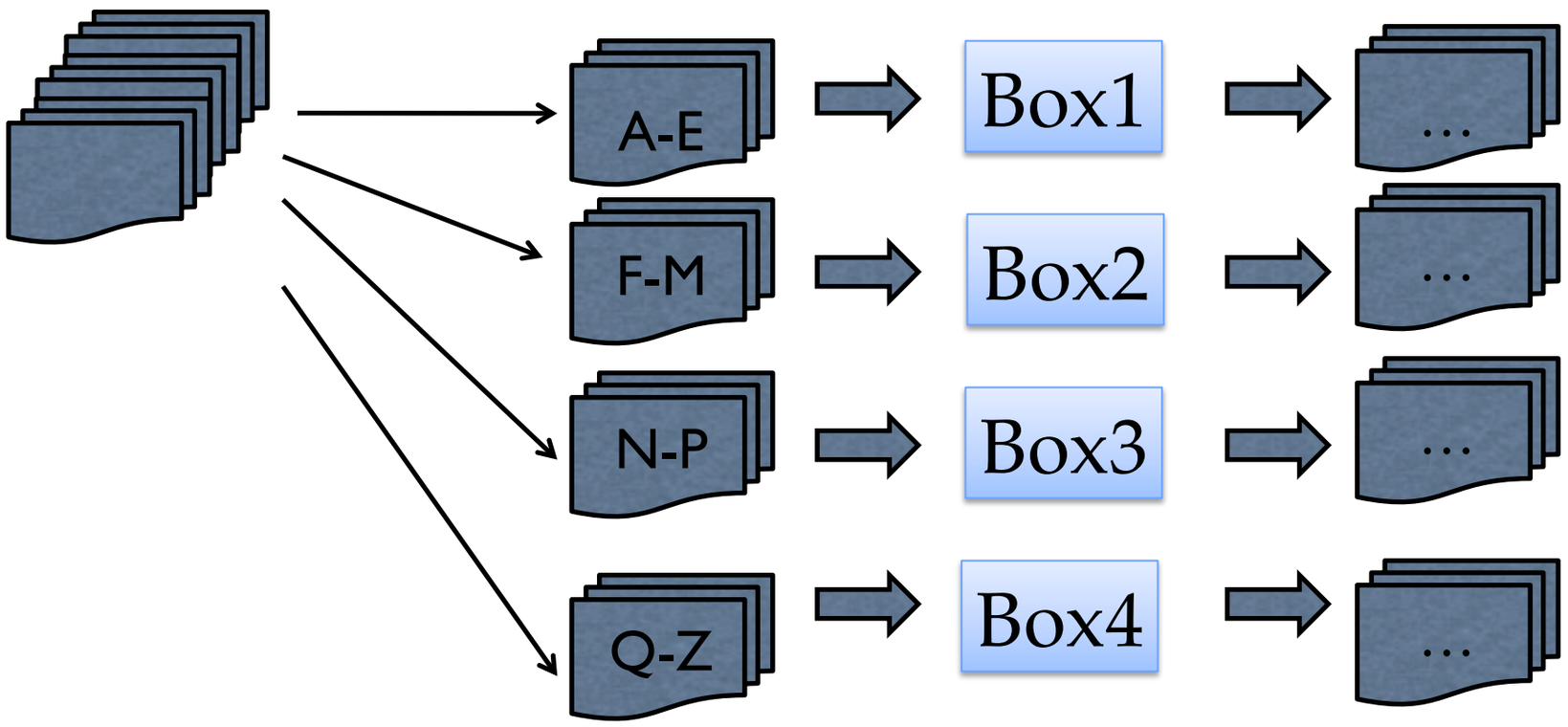


Step 1: split input data, by key, into "shards" and ship each shard to a different box

How would you run assignment 1 in parallel?

- What
- How
  - parall
    - cat input.txt | MAP | sort | REDUCE > output.txt

•Open sockets to receive data to boxk:/kludge/mapin.txt on each of the K boxes
•For each key,val pair in input.txt:
  • Send key,val pair to boxFor (key)
• Run K processes: rsh boxk 'MAP < mapin.txt > mapout.txt'



Step 2: run the maps in parallel

•Open sockets to receive data to boxk:/kludge/mapin.txt on each of the K boxes

•For each key,val pair in input.txt:

- Send key,val pair to socket[boxFor (key)]

• Run K processes: rsh … 'MAP < ….> …' to completion

• On each box:

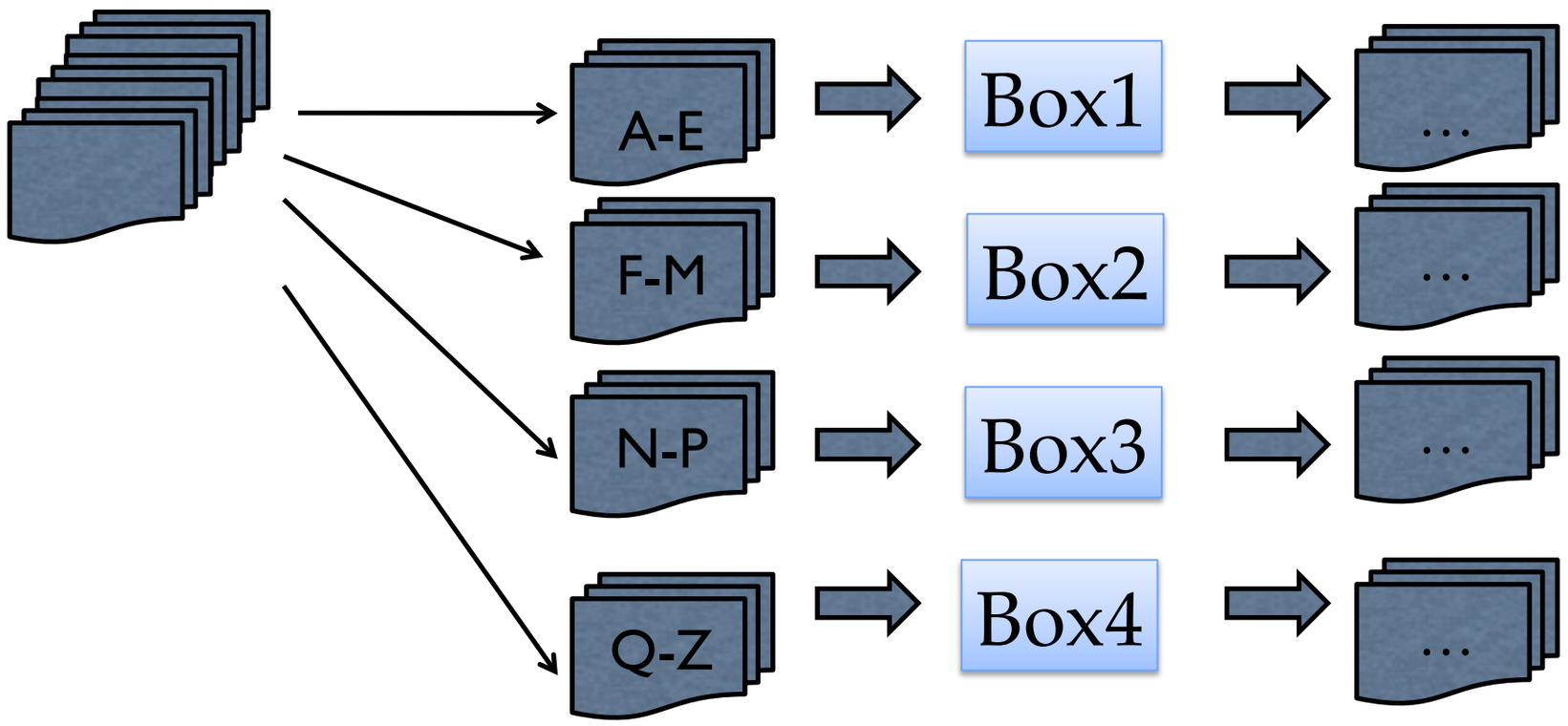
•Open sockets to receive **and sort** data to boxk:/kludge/redin.txt on each of the K boxes

•For each key,val pair in mapout.txt:

- Send key,val pair to socket[boxFor (key)]

t

cat input.txt | MAP | sort | REDUCE > output.txt



Step 3: redistribute the map output

•Open sockets to receive data to boxk:/kludge/mapin.txt on each of the K boxes

•For each key,val pair in input.txt:

- Send key,val pair to socket[boxFor (key)]

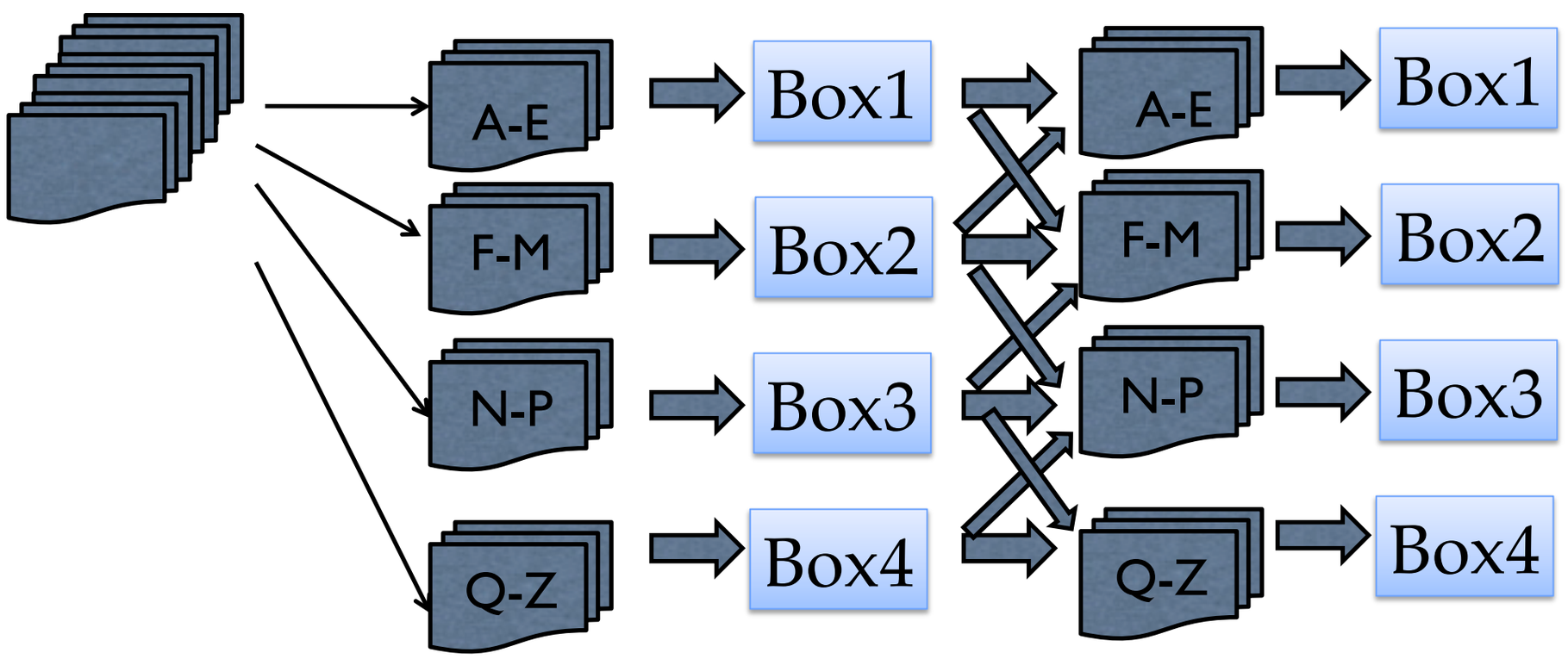
• Run K processes: rsh MAP …

• On each box:

•Open sockets to receive **and sort** data to boxk:/kludge/redin.txt on each of the K boxes

•For each key,val pair in mapout.txt:

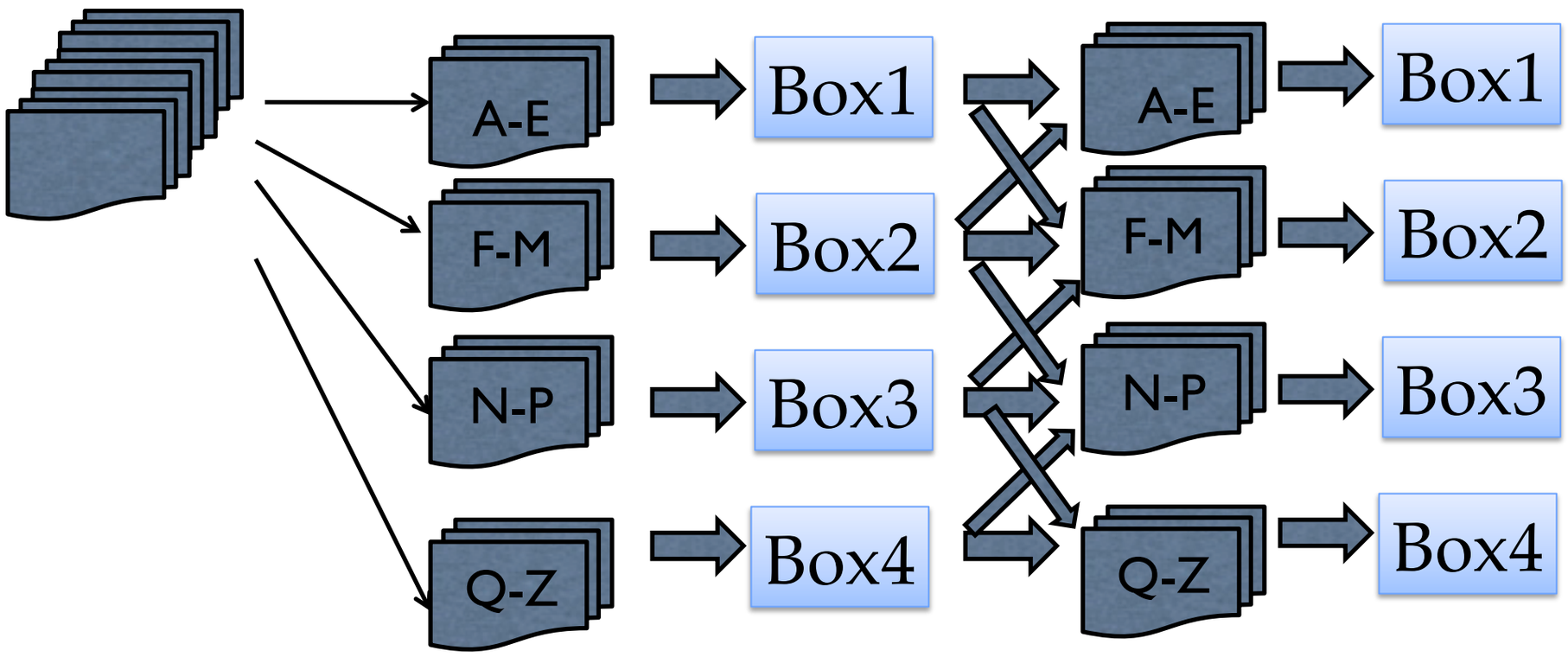
- Send key,val pair to socket[boxFor (key)]



Step 3: redistribute the map output

•Open sockets to receive data to boxk:/kludge/mapin.txt on each of the K boxes
•For each key,val pair in input.txt:
  • Send key,val pair to socket[boxFor (key)]
• Run K processes: rsh MAP < mapin.txt > mapout.txt
• Shuffle the data back to the right box
• Do the same steps for the reduce processes

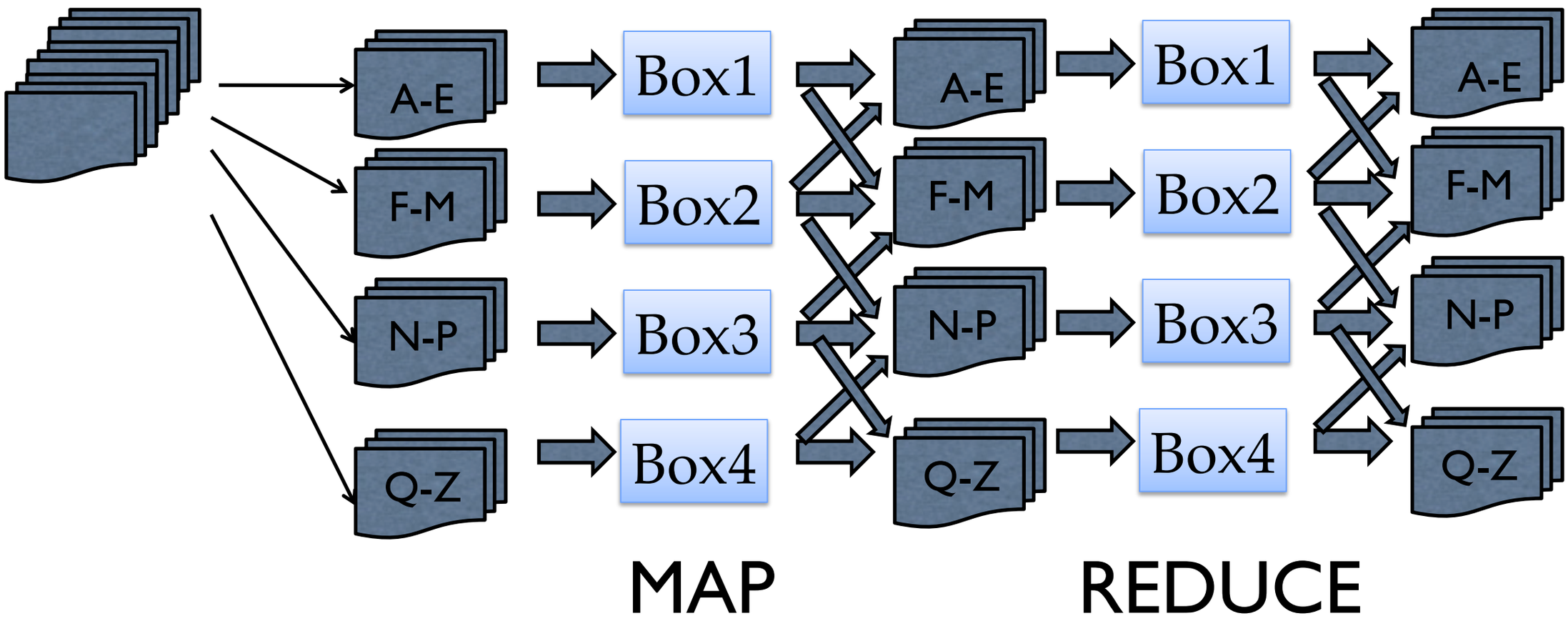
- How could you run a generic "stream-and-sort" algorithm in parallel?
  - cat input.txt | MAP | sort | REDUCE > output.txt

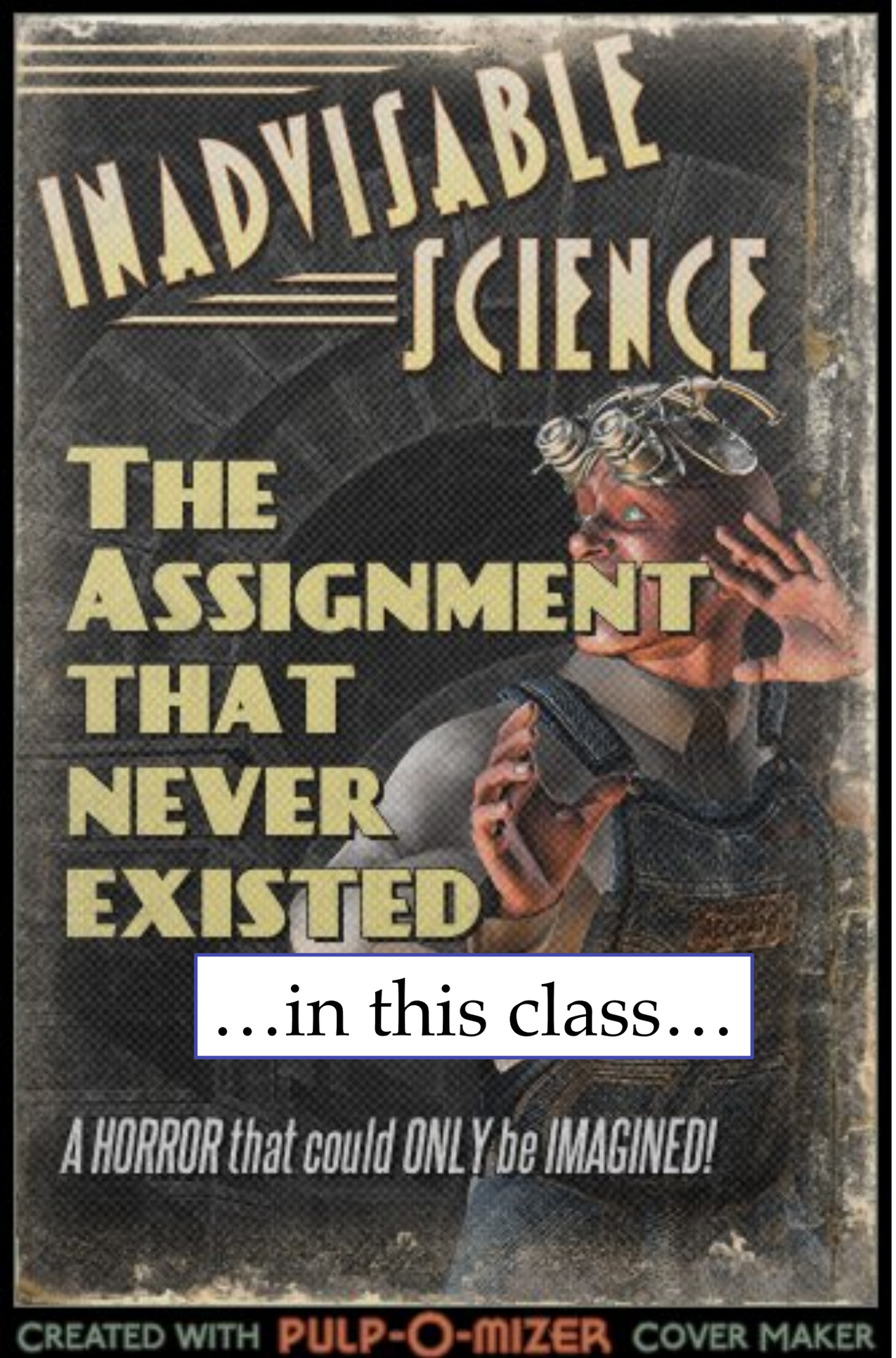


Step 4: run the reduce processes in parallel

•Open sockets to receive data to boxk:/kludge/mapin.txt on each of the K boxes
•For each key,val pair in input.txt:
  • Send key,val pair to socket[boxFor (key)]
• Run K processes: rsh MAP < mapin.txt > mapout.txt
• Shuffle the data back to the right box
• Do the same steps for the reduce process
  • (If the keys for reduce process don't change, you don't need to reshuffle them)

- cat input.txt | MAP | sort | REDUCE > output.txt

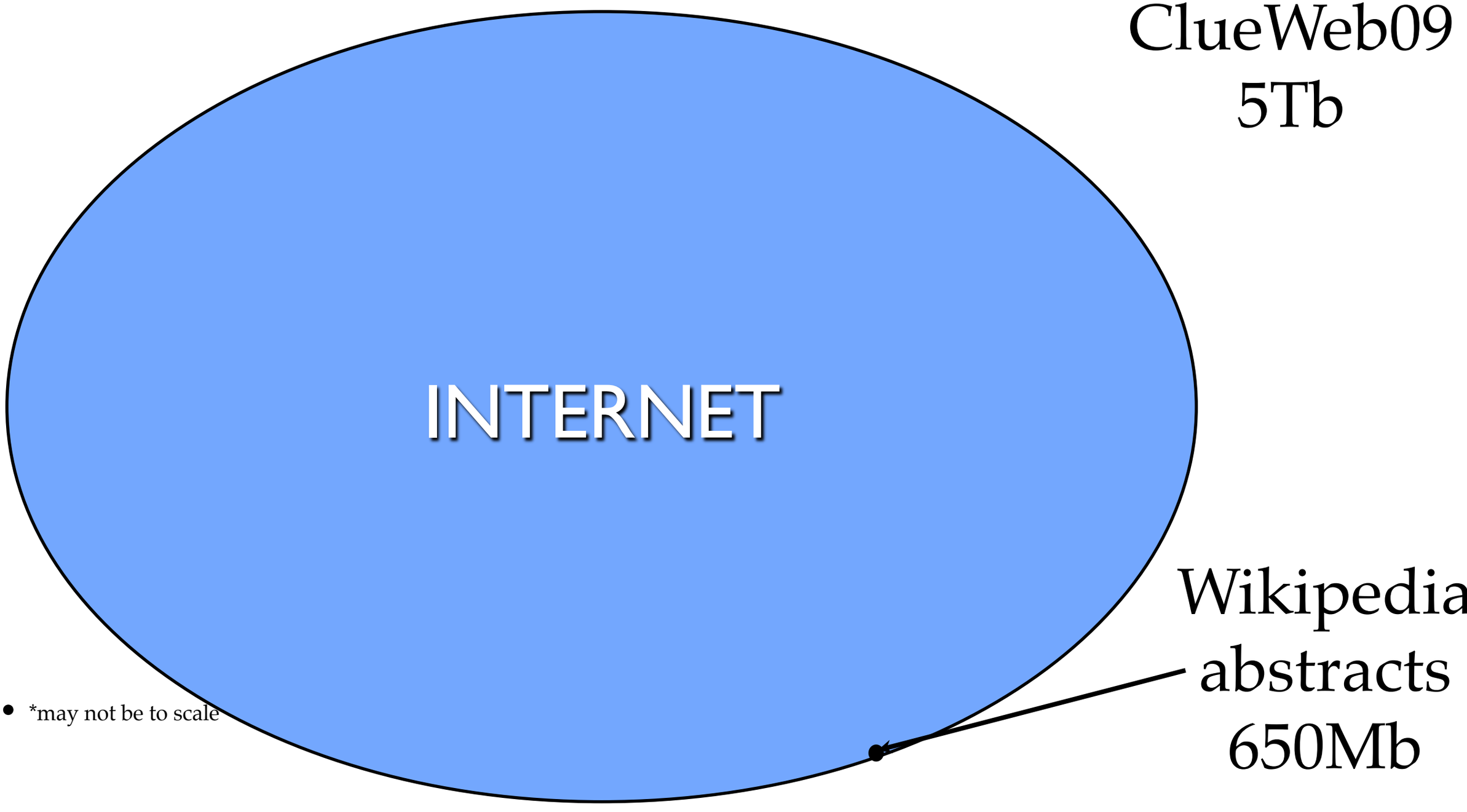




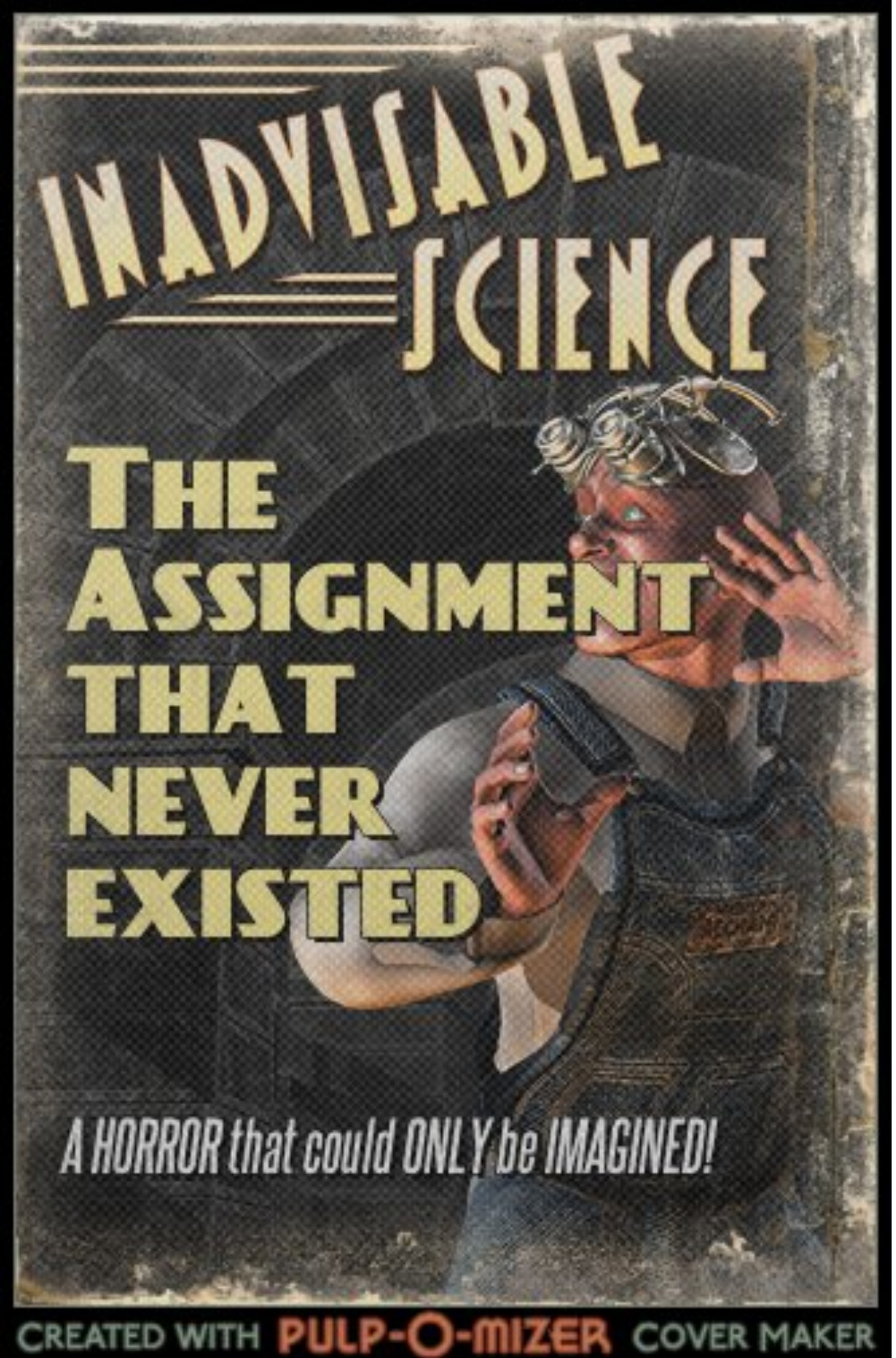
1. This would be pretty systems-y (remote copy files, waiting for remote processes, …)
2. It would take work to make it useful….

Motivating Example

- Wikipedia is a very small part of the internet*

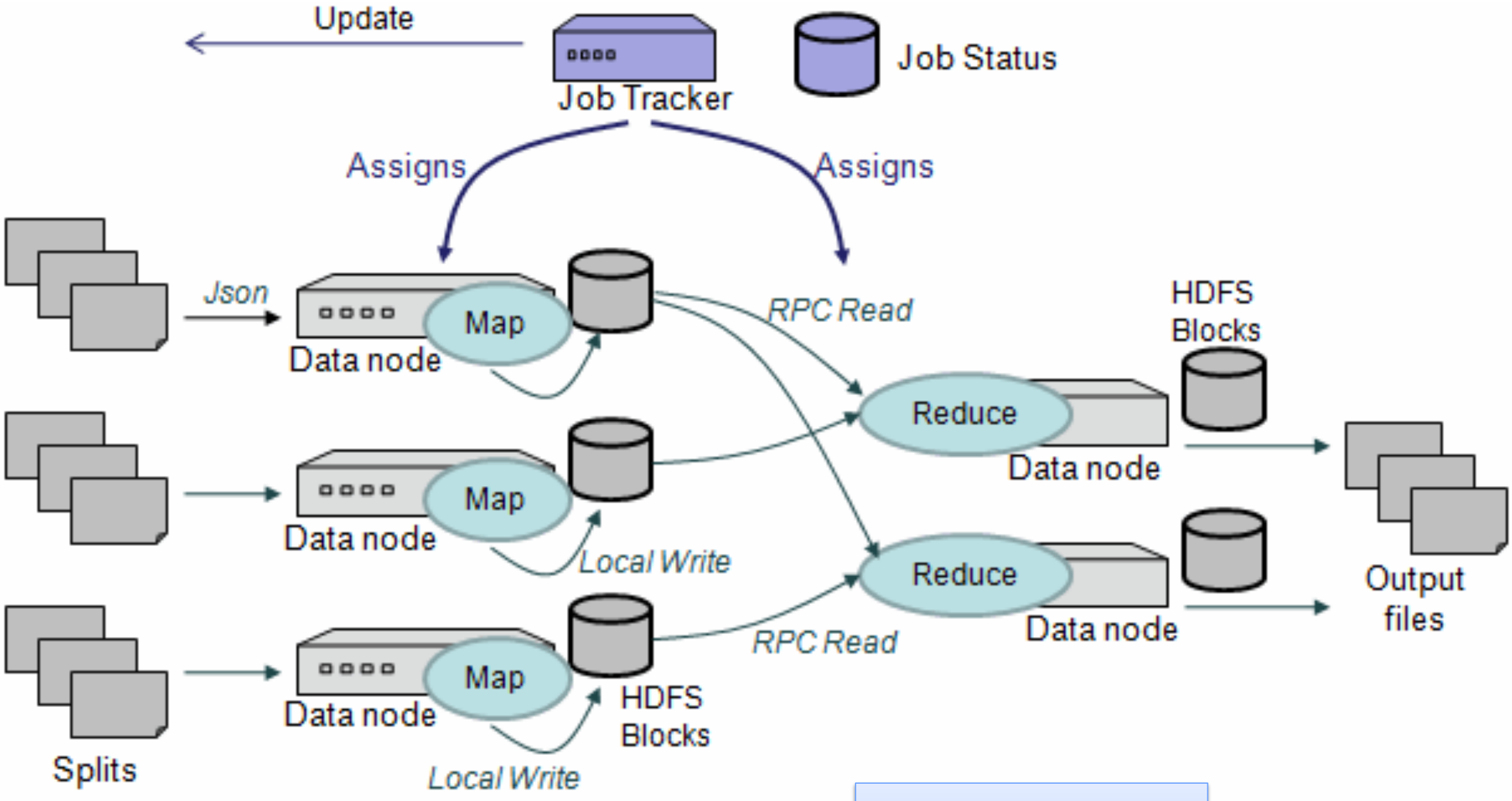


- *may not be to scale



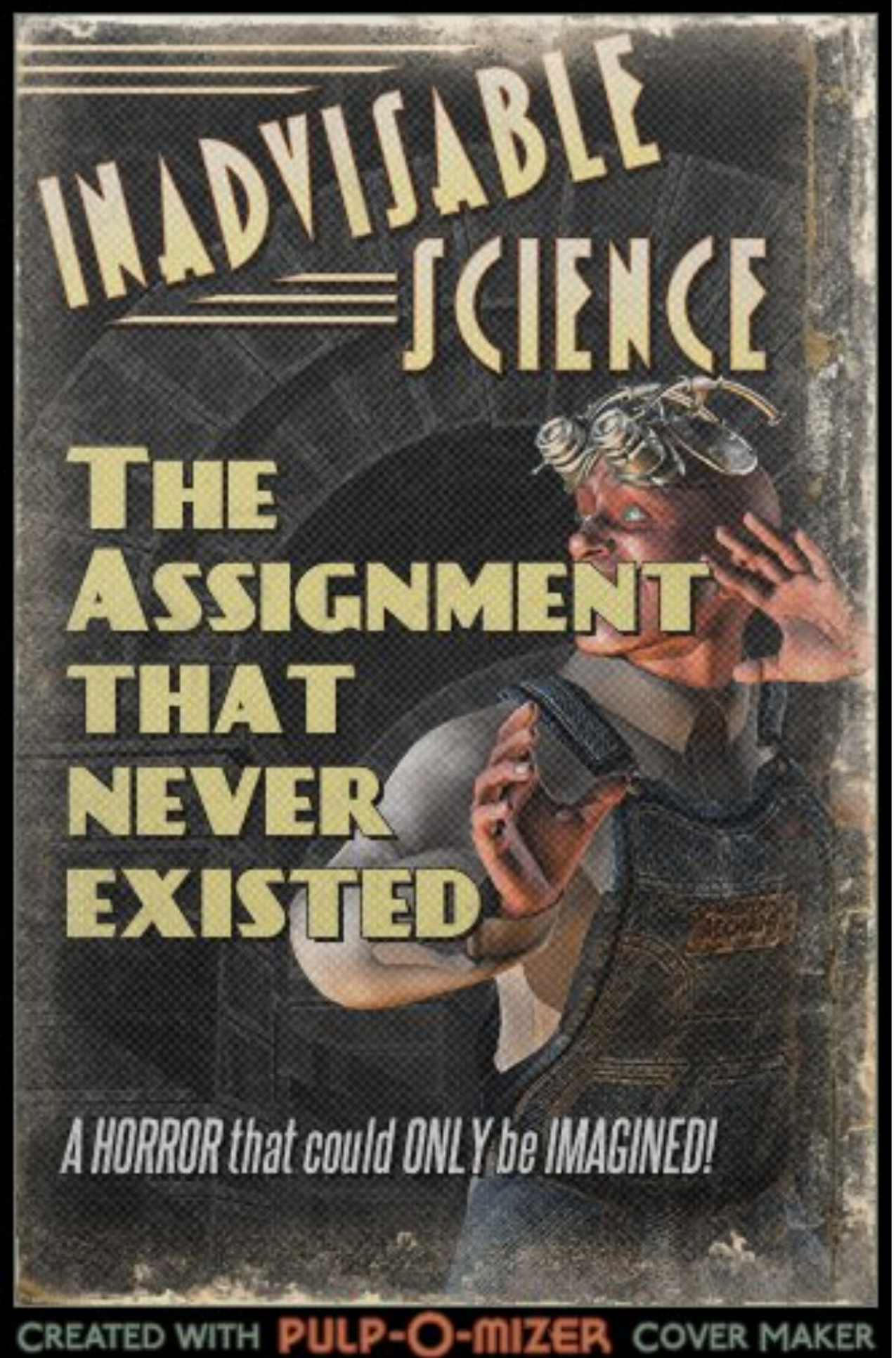
1. This would be pretty systems-y (remote copy files, waiting for remote processes, …)
2. It would take work to make run for 500 jobs

- Reliability: *Replication, restarts, monitoring jobs,…*
- Efficiency: *load-balancing, reducing file/network i/o, optimizing file/network i/o, …*
- Useability: *stream defined datatypes, simple reduce functions, ….*



Event Counting on Subsets of Documents

Summing Counts



1. This would be pretty systems-y (remote copy files, waiting for remote processes, …)
2. It would take work to make run for 500 jobs

- Reliability: *Replication, restarts, monitoring jobs,…*
- Efficiency: *load-balancing, reducing file/network i/o, optimizing file/network i/o, …*
- Useability: *stream defined datatypes, simple reduce functions, ….*

Parallel and Distributed Computing: MapReduce

- pilfered from: Alona Fyshe

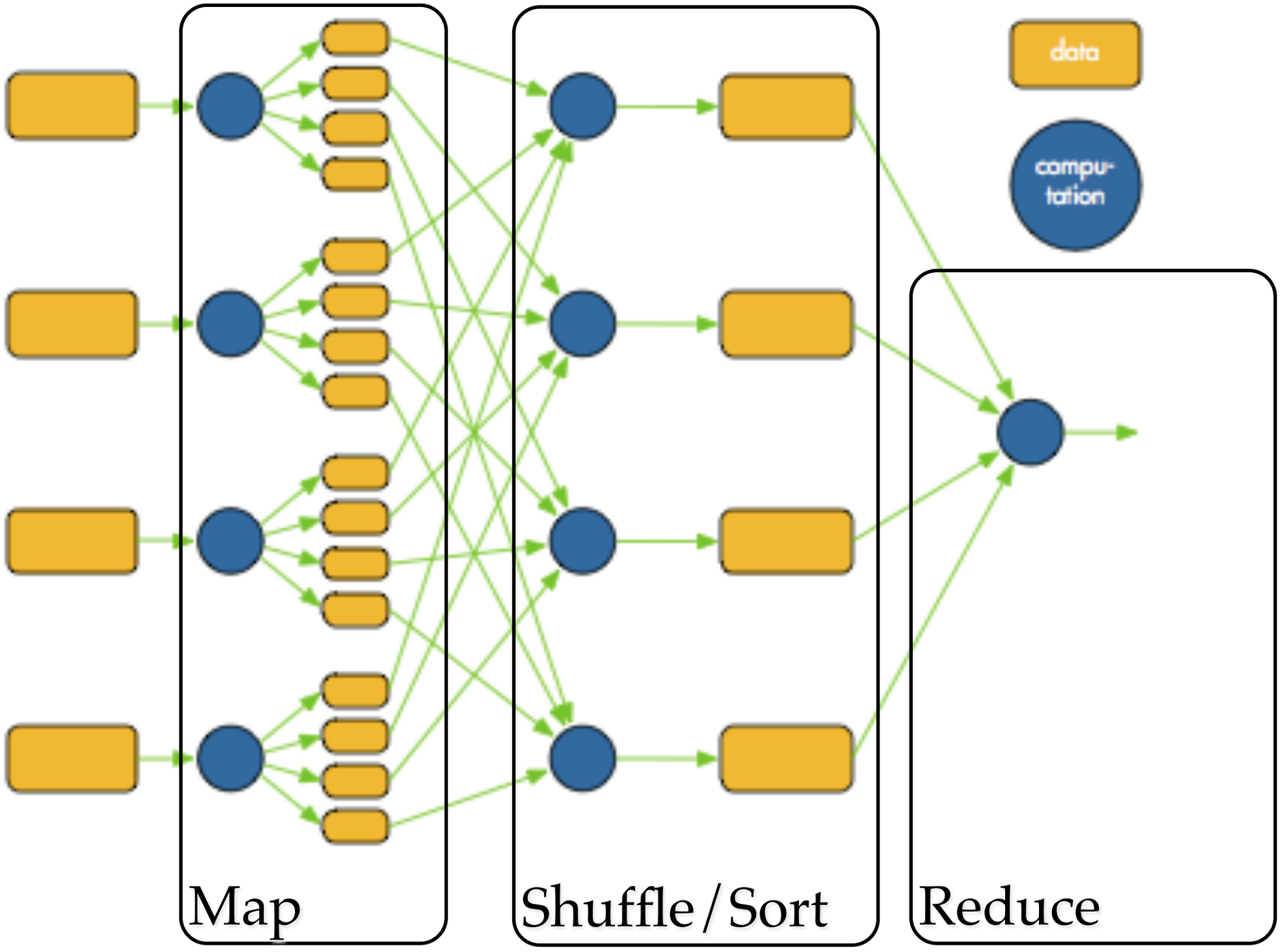
Inspiration not Plagiarism

- This is not the first lecture ever on Mapreduce
- I borrowed from Alona Fyshe and she borrowed from:
  - Jimmy Lin
    - http://www.umiacs.umd.edu/~jimmylin/cloud-computing/SIGIR-2009/Lin-MapReduce-SIGIR2009.pdf
  - Google
    - http://code.google.com/edu/submissions/mapreduce-minilecture/listing.html
    - http://code.google.com/edu/submissions/mapreduce/listing.html
  - Cloudera
    - http://vimeo.com/3584536

Surprise, you mapreduced!

- Mapreduce has three main phases
  - Map (send each input record to a key)
  - Sort (put all of one key in the same place)
    - handled behind the scenes
  - Reduce (operate on each key and its set of values)
- Terms come from functional programming:
  - map(lambda x:x.upper(),["william","w","cohen"])➔['WILLIAM', 'W', 'COHEN']
  - reduce(lambda x,y:x+"-"+y,["william","w","cohen"])➔”william-w-cohen”

Mapreduce overview



Mapreduce: slow motion

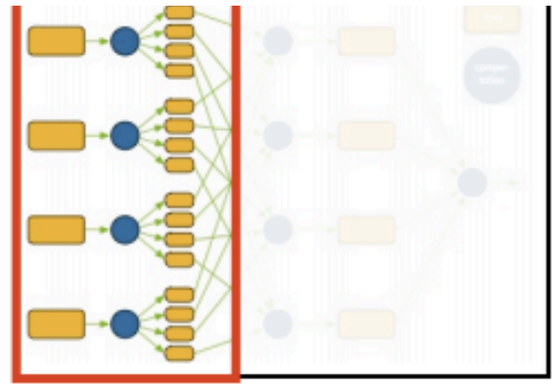
- The canonical mapreduce example is word count
- Example corpus:

Joe likes toast

Jane likes toast with jam

Joe burnt the toast

MR: slow motion: Map



Input

Joe likes toast

Map 1

Jane likes toast with jam

Map 2

Joe burnt the toast

Map 3

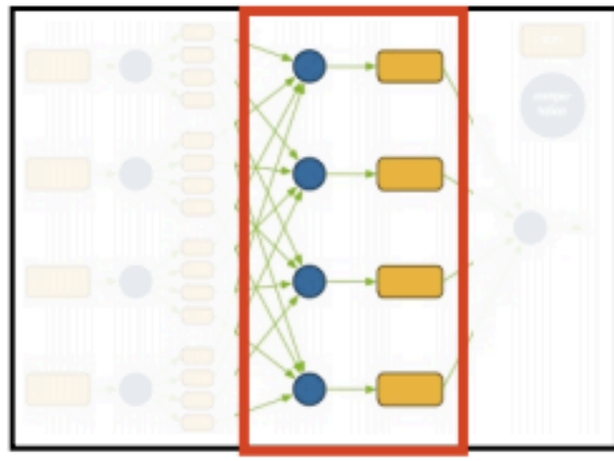
Output

Joe	1
likes	1
toast	1

Jane	1
likes	1
toast	1
with	1
jam	1

Joe	1
burnt	1
the	1
toast	1

MR: slow motion: Sort



Input

Joe	1
likes	1
toast	1

Jane	1
likes	1
toast	1
with	1
jam	1

Joe	1
burnt	1
the	1
toast	1

Output

Joe	1
Joe	1

Jane	1

likes	1
likes	1

toast	1
toast	1
toast	1

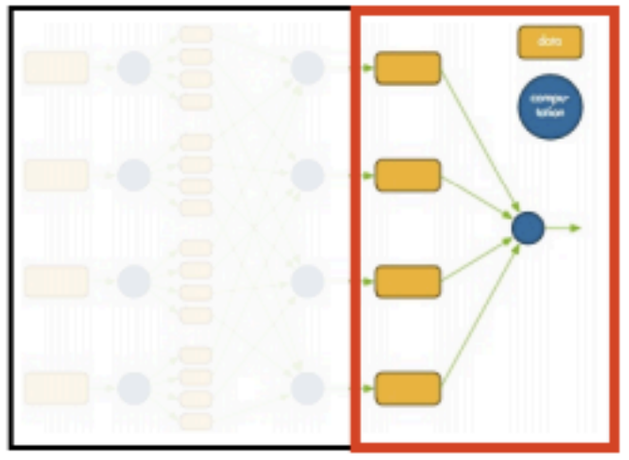
with	1

jam	1

burnt	1

the	1

MR: slow mo: Reduce



Input

Joe	1	Reduce 1
Joe	1	
Jane	1	Reduce 2
likes	1	Reduce 3
likes	1	
toast	1	Reduce 4
toast	1	
toast	1	
with	1	Reduce 5
jam	1	Reduce 6
burnt	1	Reduce 7
the	1	Reduce 8

Output

Joe	2
Jane	1
likes	2
toast	3
with	1
jam	1
burnt	1
the	1

Distributing NB

- Questions:
  - How will you know when each machine is done?
    - Communication overhead
  - How will you know if a machine is dead?

Failure

- How big of a deal is it really?
  - A huge deal. In a distributed environment disks fail ALL THE TIME.
  - Large scale systems must assume that any process can fail at any time.
  - It may be much cheaper to make the software run reliably on unreliable hardware than to make the hardware reliable.
- Ken Arnold (Sun, CORBA designer):
- Failure is the defining difference between distributed and local programming, so you have to design distributed systems with the expectation of failure. Imagine asking people, "If the probability of something happening is one in 10^{13}, how often would it happen?" Common sense would be to answer, "Never." That is an infinitely large number in human terms. But if you ask a physicist, she would say, "All the time. In a cubic foot of air, those things happen all the time."

Well, that's a pain

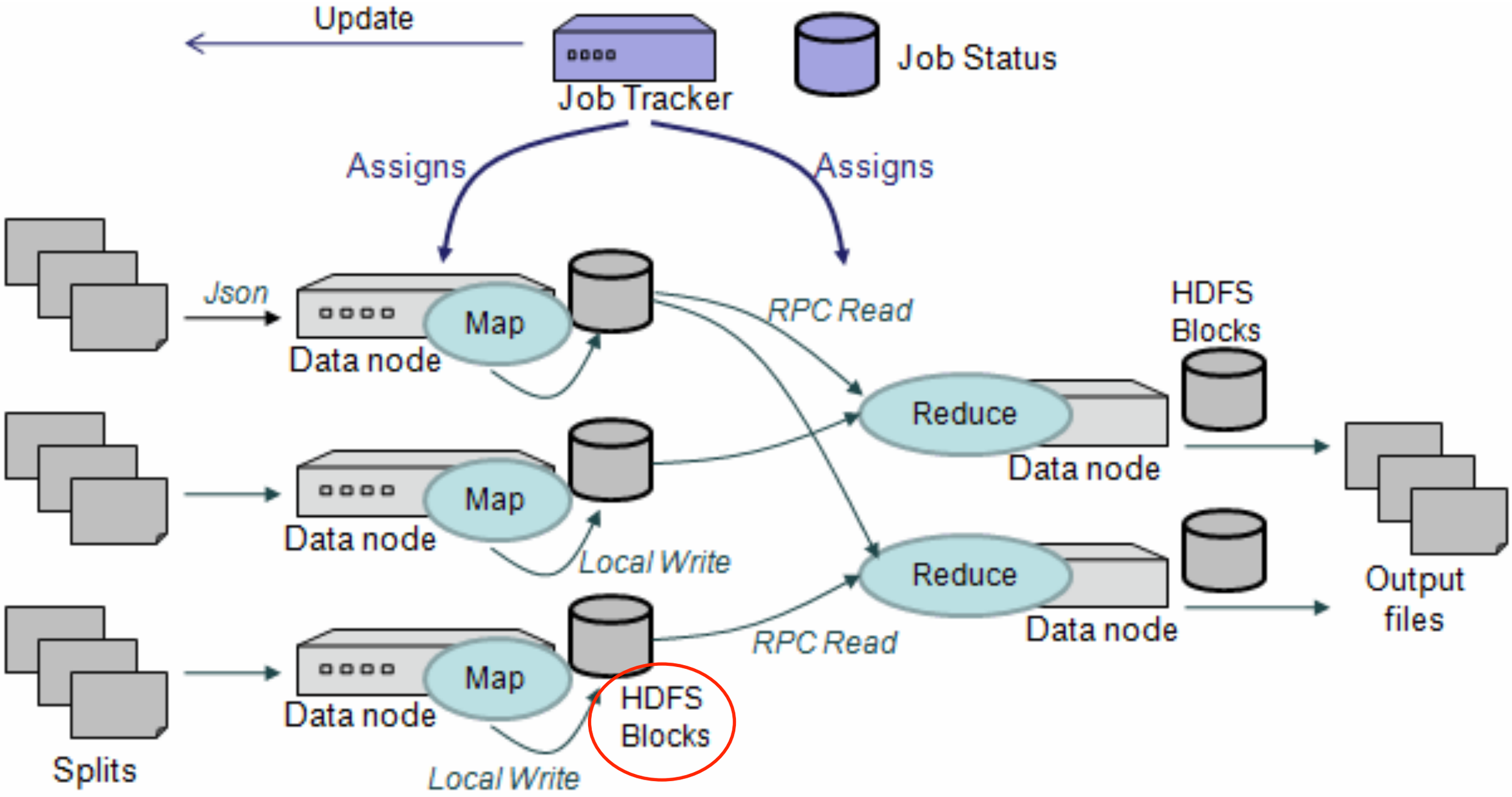
- What will you do when a task fails?

Well, that's a pain

- What's the difference between slow and dead?
  - Who cares? Start a backup process.
    - If the process is slow because of machine issues, the backup may finish first
    - If it's slow because you poorly partitioned your data... waiting is your punishment

What else is a pain?

- Losing your work!
- If a disk fails you can lose some intermediate output
    - Ignoring the missing data could give you wrong answers
- Who cares? if I'm going to run backup processes I might as well have backup copies of the intermediate data also

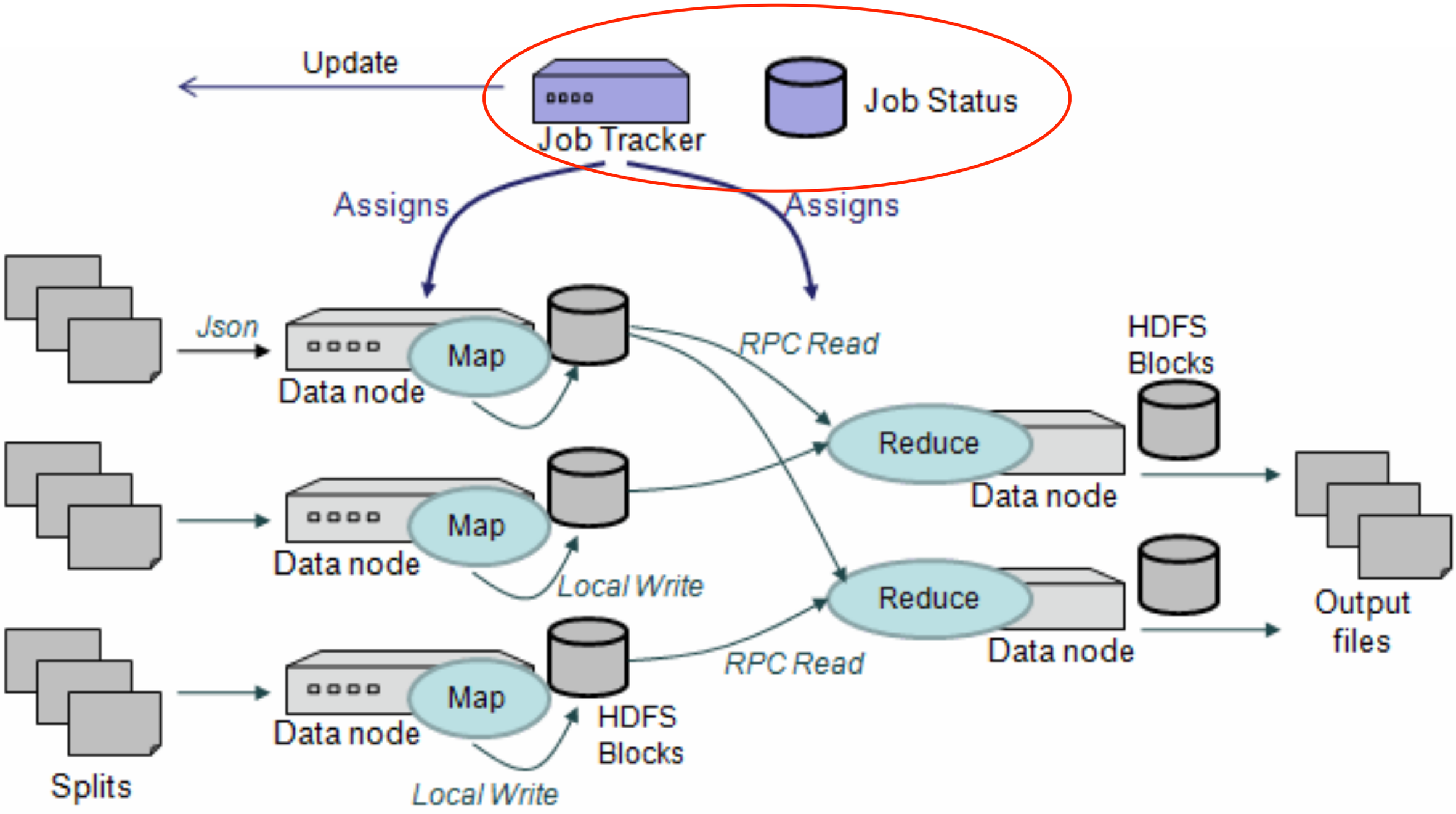

Update
Job Tracker
Job Status
Assigns
Assigns
Json
Map
Data node
RPC Read
HDFS
Blocks
Reduce
Data node
Map
Data node
Local Write
Reduce
Data node
Output
files
Map
Data node
HDFS
Blocks
RPC Read
Splits
Local Write

HDFS: The Hadoop File System

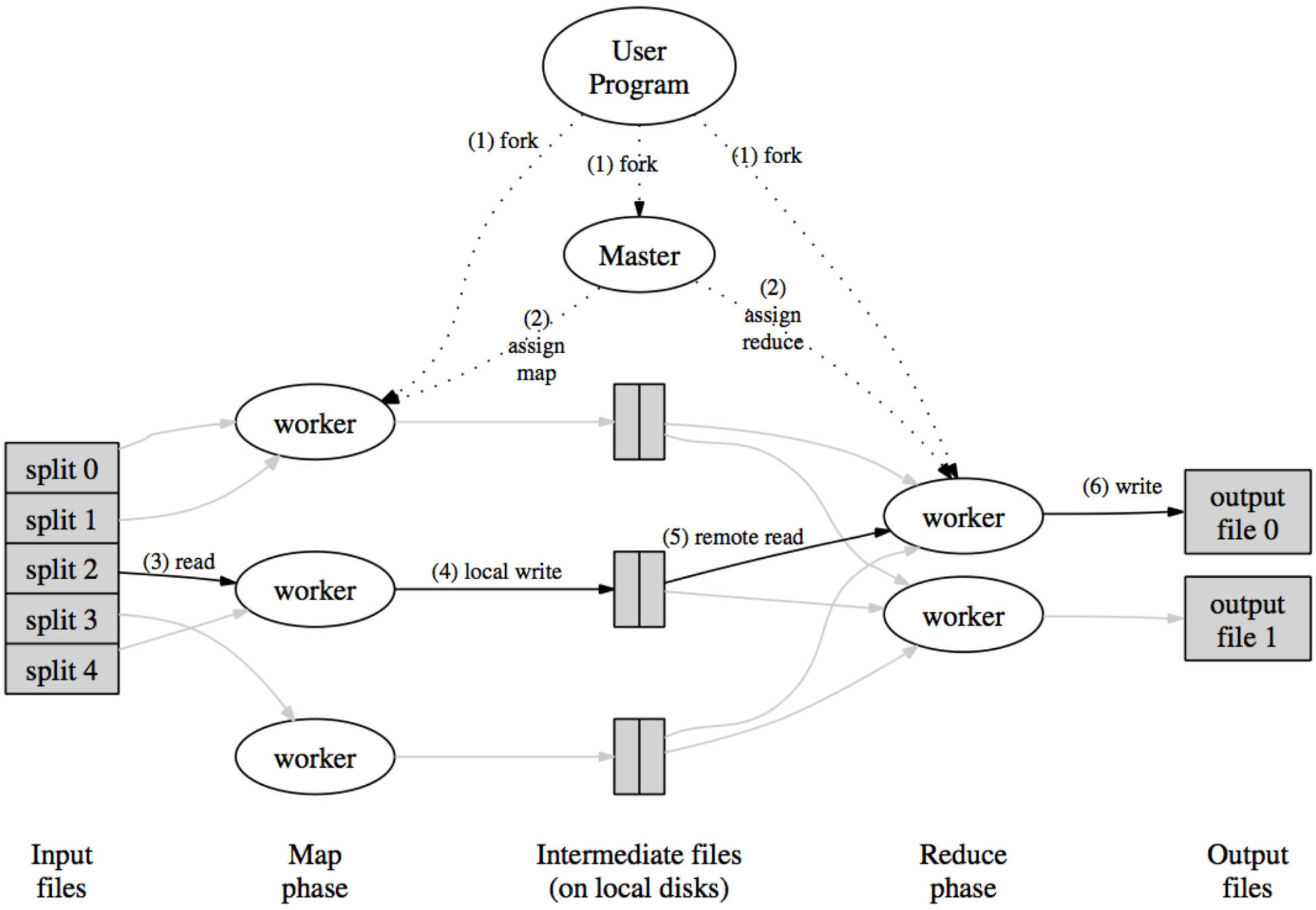
- Distributes data across the cluster
    - distributed file *looks like* a directory with shards as files inside it
        - makes an effort to run processes *locally* with the data
- Replicates data
    - default 3 copies of each file
- Optimized for streaming
    - really really big “blocks”

```
$ hadoop fs -ls rcv1/small/sharded
Found 10 items
-rw-r--r--   3 …  606405 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00000
-rw-r--r--   3 … 1347611 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00001
-rw-r--r--   3 …  939307 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00002
-rw-r--r--   3 … 1284062 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00003
-rw-r--r--   3 … 1009890 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00004
-rw-r--r--   3 … 1206196 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00005
-rw-r--r--   3 … 1384658 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00006
-rw-r--r--   3 … 1299698 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00007
-rw-r--r--   3 …  928752 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00008
-rw-r--r--   3 …  806030 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00009

$ hadoop fs -tail rcv1/small/sharded/part-00005
weak as the arrival of arbitraged cargoes from the West has put the local market under pressure…
M14,M143,MCAT    The Brent crude market on the Singapore International …
```


Update
Job Tracker
Job Status
Assigns
Assigns
Json
Map
Data node
RPC Read
HDFS
Blocks
Reduce
Data node
Map
Data node
Local Write
Reduce
Data node
Output
files
Map
Data node
HDFS
Blocks
RPC Read
Splits
Local Write

MR Overview


User Program
(1) fork
(1) fork
(1) fork
Master
(2) assign map
(2) assign reduce
worker
worker
worker
worker
worker
split 0
split 1
split 2
split 3
split 4
(3) read
(4) local write
(5) remote read
(6) write
output file 0
output file 1
Input files
Map phase
Intermediate files (on local disks)
Reduce phase
Output files

Hadoop job_201301231150_0778 on hadoopjt

User: wcohen
Job Name: streamjob6055532903853567038.jar
Job File: hdfs://hdfsname.opencloud/l/a2/scratch/hadoop-data/global/mapred/system/job_201301231150_0778/job.xml
Job Setup: Successful
Status: Failed
Started at: Wed Jan 30 11:46:47 EST 2013
Failed at: Wed Jan 30 11:47:28 EST 2013
Failed in: 41sec
Job Cleanup: Successful
Black-listed TaskTrackers: 2
Job Scheduling information: 5 running map tasks using 5 map slots, 0 running reduce tasks using 0 reduce slots.

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	10	0	0	0	10	35 / 5
reduce	00%	10	0	0	0	10	0 / 0

	Counter	Map	Reduce	Total
Job Counters	Rack-local map tasks	0	0	38
	Launched map tasks	0	0	40
	Data-local map tasks	0	0	2
	Failed map tasks	0	0	1

Map Completion Graph - close

Hadoop map task list for job_201301231150_0778 on hadoop

All Tasks

Task	Complete	Status	Start Time	Finish Time	Errors
task_201301231150_0778_m_000000	0.00%		30-Jan-2013 11:47:01	30-Jan-2013 11:47:25 (24sec)	java.lang.RuntimeException: PipeMa at org.apache.hadoop.strea at org.apache.hadoop.strea at org.apache.hadoop.strea at org.apache.hadoop.mapre at org.apache.hadoop.strea at org.apache.hadoop.mapre at org.apache.hadoop.mapre at org.apache.hadoop.mapre java.lang.RuntimeException: PipeMa at org.apache.hadoop.strea at org.apache.hadoop.strea at org.apache.hadoop.strea at org.apache.hadoop.strea at org.apache.hadoop.mapre at org.apache.hadoop.strea at org.apache.hadoop.mapre at org.apache.hadoop.mapre java.lang.RuntimeException: PipeMa at org.apache.hadoop.strea at org.apache.hadoop.strea at org.apache.hadoop.strea at org.apache.hadoop.strea at org.apache.hadoop.mapre at org.apache.hadoop.strea at org.apache.hadoop.mapre at org.apache.hadoop.mapre java.lang.RuntimeException: PipeMa at org.apache.hadoop.strea

Job job_201301231150_0778

All Task Attempts

Task Attempts	Machine	Status	Progress	Start Time	Finish Time	Error
attempt_201301231150_0778_m_000000_0	/default-rack/cloud3u12.opencloud	FAILED	0.00%	30-Jan-2013 11:47:01	30-Jan-2013 11:47:06 (4sec)	java.
attempt_201301231150_0778_m_000000_1	/default-rack/cloud2u28.opencloud	FAILED	0.00%	30-Jan-2013 11:47:07	30-Jan-2013 11:47:11 (4sec)	java.

me	Errors	Task Logs	Counters	Actions
2013	java.lang.RuntimeException: PipeMapRed.waitOutputThreads(): subprocess failed w de 1 at org.apache.hadoop.streaming.PipeMapRed.waitOutputThreads(PipeMapRed.j 1) at org.apache.hadoop.streaming.PipeMapRed.mapRedFinished(PipeMapRed.java: at org.apache.hadoop.streaming.PipeMapper.close(PipeMapper.java:132) at org.apache.hadoop.mapred.MapRunner.run(MapRunner.java:57) at org.apache.hadoop.streaming.PipeMapRunner.run(PipeMapRunner.java:36) at org.apache.hadoop.mapred.MapTask.runOldMapper(MapTask.java:358) at org.apache.hadoop.mapred.MapTask.run(MapTask.java:307) at org.apache.hadoop.mapred.Child.main(Child.java:170)	Last 4KB Last 8KB All	1	
2013	java.lang.RuntimeException: PipeMapRed.waitOutputThreads(): subprocess failed with code 1 at org.apache.hadoop.streaming.PipeMapRed.waitOutputThreads(PipeMapRed.java:311) at org.apache.hadoop.streaming.PipeMapRed.mapRedFinished(PipeMapRed.java:540) at org.apache.hadoop.streaming.PipeMapper.close(PipeMapper.java:132) at org.apache.hadoop.mapred.MapRunner.run(MapRunner.java:57) at org.apache.hadoop.streaming.PipeMapRunner.run(PipeMapRunner.java:36) at org.apache.hadoop.mapred.MapTask.runOldMapper(MapTask.java:358) at org.apache.hadoop.mapred.MapTask.run(MapTask.java:307) at org.apache.hadoop.mapred.Child.main(Child.java:170)	Last 4KB Last 8KB All	1	
2013	java.lang.RuntimeException: PipeMapRed.waitOutputThreads(): subprocess failed with code 1 at org.apache.hadoop.streaming.PipeMapRed.waitOutputThreads(PipeMapRed.java:311) at org.apache.hadoop.streaming.PipeMapRed.mapRedFinished(PipeMapRed.java:540) at org.apache.hadoop.streaming.PipeMapper.close(PipeMapper.java:132) at org.apache.hadoop.mapred.MapRunner.run(MapRunner.java:57) at org.apache.hadoop.streaming.PipeMapRunner.run(PipeMapRunner.java:36) at org.apache.hadoop.mapred.MapTask.runOldMapper(MapTask.java:358) at org.apache.hadoop.mapred.MapTask.run(MapTask.java:307) at org.apache.hadoop.mapred.Child.main(Child.java:170)	Last 4KB Last 8KB All	1	

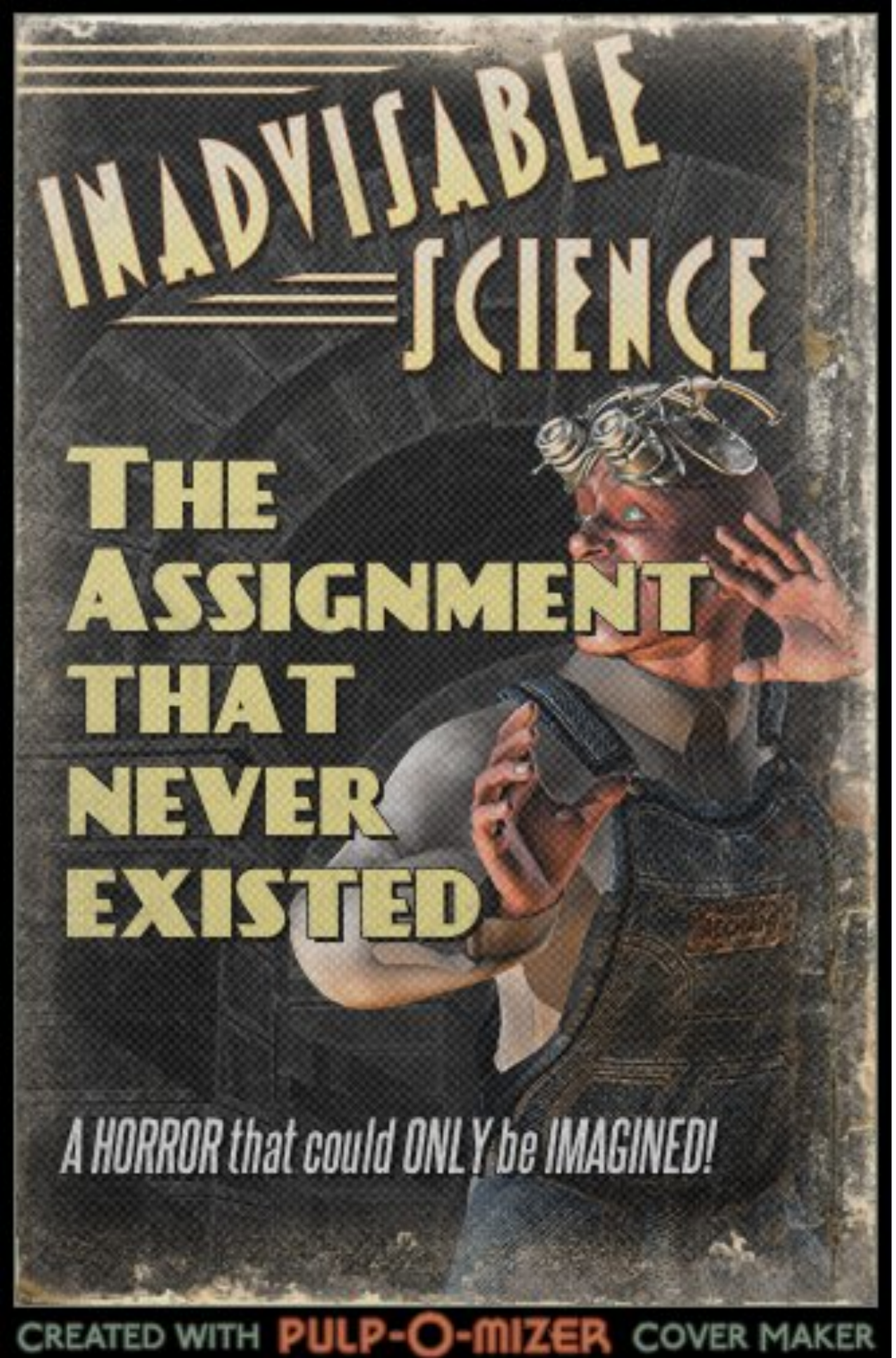
Task Logs: 'attempt_201301231150_0778_m_000000_0'

stdout logs

---

stderr logs

```
Exception in thread "main" java.lang.NoClassDefFoundError: com/wcohen/StreamNB
Caused by: java.lang.ClassNotFoundException: com.wcohen.StreamNB
        at java.net.URLClassLoader$1.run(URLClassLoader.java:202)
        at java.security.AccessController.doPrivileged(Native Method)
        at java.net.URLClassLoader.findClass(URLClassLoader.java:190)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:306)
        at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:301)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:247)
Could not find the main class: com.wcohen.StreamNB.  Program will exit.
```



1. This would be pretty systems-y (remote copy files, waiting for remote processes, …)
2. It would take work to make work for 500 jobs

- Reliability: *Replication, restarts, monitoring jobs,…*
- Efficiency: *load-balancing, reducing file/network i/o, optimizing file/network i/o, …*
- Useability: *stream defined datatypes, simple reduce functions, ….*

Map reduce with Hadoop streaming

Breaking this down…

- What actually is a key-value pair? How do you interface with Hadoop?
- One very simple way: Hadoop's *streaming* interface.
  - Mapper outputs key-value pairs as:
    - One pair per line, key and value tab-separated
- Reduced reads in data in the same format
  - Lines are sorted so lines with the same key are adjacent.

An example:

- SmallStreamNB.java and StreamSumReducer.java:
  - the code you just wrote.

annotated-log.txt

```
wcohen@shell2:~/naive-bayes-demo$ ls -l
total 1170892
-rw-r--r--. 1 wcohen lcs       1288 Feb  5 11:16 Makefile
-rw-r--r--. 1 wcohen lcs  118656462 Jan 22  2013 RCV1.full_test.txt
-rw-r--r--. 1 wcohen lcs 1065782405 Jan 22  2013 RCV1.full_train.txt
-rw-r--r--. 1 wcohen lcs    1198026 Jan 22  2013 RCV1.small_test.txt
-rw-r--r--. 1 wcohen lcs   10812609 Jan 22  2013 RCV1.small_train.txt
-rw-r--r--. 1 wcohen lcs      18771 Jan 22  2013 RCV1.very_small_test.txt
-rw-r--r--. 1 wcohen lcs      99397 Jan 22  2013 RCV1.very_small_train.txt
-rw-r--r--. 1 wcohen lcs       7721 Jan 30  2013 SmallStreamNB.java
-rw-r--r--. 1 wcohen lcs       4162 Jan 26  2013 StreamNB.java
-rw-r--r--. 1 wcohen lcs        772 Jan 29  2013 StreamSumReducer.java
drwxr-xr-x. 3 wcohen lcs       4096 Jan 25  2013 classes
-rw-r--r--. 1 wcohen lcs      10802 Jan 30  2013 nb.jar
-rw-r--r--. 1 wcohen lcs        576 Jan 28  2013 notes.txt
wcohen@shell2:~/naive-bayes-demo$ cat Makefile
--:---  annotated-log.txt   Top L1     (Text)----------------------------------
```

To run locally:

```
test-small: small-events.txt nb.jar
        time java -cp nb.jar com.wcohen.SmallStreamNB \
        RCV1.small_test.txt MCAT,CCAT,GCAT,ECAT 2000 < small-events.txt \
        | cut -f3 | sort | uniq -c

small-events.txt:  nb.jar
        time java -cp nb.jar com.wcohen.SmallStreamNB \
        < RCV1.small_train.txt | sort -k1,1 \
        | java -cp nb.jar com.wcohen.StreamSumReducer> small-events.txt
```

To train with streaming Hadoop you do this:

```
STRJAR = /usr/lib/hadoop/contrib/streaming/hadoop-streaming-1.2.0.1.3.0.0-107.jar

small-events-hs:
        hadoop fs -rmr rcv1/small/events
        time hadoop jar $(STRJAR) \
           -input rcv1/small/sharded -output rcv1/small/events \
        -mapper 'java -Xmx512m -cp ./lib/nb.jar com.wcohen.StreamNB' \
        -reducer 'java -Xmx512m -cp ./lib/nb.jar com.wcohen.StreamSumReducer' \
        -file nb.jar -numReduceTasks 10
```

But first you need to get your code and data to the "Hadoop file system"

```
hadoop fs -rmr rcv1/small/events
Moved to trash: hdfs://t1disc-pnn:8020/user/wcohen/rcv1/small/events
time hadoop jar /usr/lib/hadoop/contrib/streaming/hadoop-streaming-1.2.0.1.3.0.0-107.jar  \
          -input rcv1/small/sharded -output rcv1/small/events \
          -mapper 'java -Xmx512m -cp ./lib/nb.jar com.wcohen.StreamNB' \
          -reducer 'java -Xmx512m -cp ./lib/nb.jar com.wcohen.StreamSumReducer' \
          -file nb.jar -numReduceTasks 10
packageJobJar: [nb.jar, /tmp/hadoop-wcohen/hadoop-unjar89357193914132497 32/] [] /tmp/streamjob
l
14/02/05 11:24:56 INFO lzo.GPLNativeCodeLoader: Loaded native gpl library
14/02/05 11:24:56 INFO lzo.LzoCodec: Successfully loaded & initialized native-lzo library [hac
08101c2729dc0c9ff3]
14/02/05 11:24:56 WARN snappy.LoadSnappy: Snappy native library is available
14/02/05 11:24:56 INFO util.NativeCodeLoader: Loaded the native-hadoop library
14/02/05 11:24:56 INFO snappy.LoadSnappy: Snappy native library loaded
14/02/05 11:24:56 INFO mapred.FileInputFormat: Total input paths to process : 10
14/02/05 11:24:57 INFO streaming.StreamJob: getLocalDirs(): [/l/a/hadoop/mapred, /l/b/hadoop/m
doop/mapred]
14/02/05 11:24:57 INFO streaming.StreamJob: Running job: job_201312100900_1189
14/02/05 11:24:57 INFO streaming.StreamJob: To kill this job, run:
14/02/05 11:24:57 INFO streaming.StreamJob: /usr/lib/hadoop/libexec/../bin/hadoop job  -Dmapre
mu.local:50300 -kill job_201312100900_1189
14/02/05 11:24:57 INFO streaming.StreamJob: Tracking URL: http://t1disc-jt.disc.pdl.cmu.local:
312100900_1189
14/02/05 11:24:58 INFO streaming.StreamJob:  map 0%  reduce 0%
14/02/05 11:25:09 INFO streaming.StreamJob:  map 20%  reduce 0%
14/02/05 11:25:14 INFO streaming.StreamJob:  map 57%  reduce 0%
14/02/05 11:25:15 INFO streaming.StreamJob:  map 77%  reduce 0%
14/02/05 11:25:16 INFO streaming.StreamJob:  map 79%  reduce 0%
14/02/05 11:25:17 INFO streaming.StreamJob:  map 79%  reduce 4%
14/02/05 11:25:18 INFO streaming.StreamJob:  map 80%  reduce 6%
14/02/05 11:25:19 INFO streaming.StreamJob:  map 70%  reduce 7%
14/02/05 11:25:24 INFO streaming.StreamJob:  map 80%  reduce 7%
14/02/05 11:25:25 INFO streaming.StreamJob:  map 80%  reduce 10%
14/02/05 11:25:26 INFO streaming.StreamJob:  map 90%  reduce 17%
14/02/05 11:25:27 INFO streaming.StreamJob:  map 100%  reduce 20%
14/02/05 11:25:28 INFO streaming.StreamJob:  map 100%  reduce 22%
14/02/05 11:25:31 INFO streaming.StreamJob:  map 100%  reduce 34%
14/02/05 11:25:32 INFO streaming.StreamJob:  map 100%  reduce 38%
14/02/05 11:25:33 INFO streaming.StreamJob:  map 100%  reduce 40%
14/02/05 11:25:34 INFO streaming.StreamJob:  map 100%  reduce 60%
14/02/05 11:25:35 INFO streaming.StreamJob:  map 100%  reduce 87%
14/02/05 11:25:36 INFO streaming.StreamJob:  map 100%  reduce 100%
14/02/05 11:25:37 INFO streaming.StreamJob: Job complete: job_201312100900_1189
14/02/05 11:25:37 INFO streaming.StreamJob: Output: rcv1/small/events
2.59user 0.13system 0:42.14elapsed 6%CPU (0avgtext+0avgdata 314512maxresident)k
0inputs+1080outputs (0major+25504minor)pagefaults 0swaps
```

To train with streaming Hadoop:

- First, you need to prepare the corpus by splitting it into *shards*
- … and distributing the shards to different machines:

```
wcohen@shell2:~/naive-bayes-demo$ hadoop fs --help
-help: Unknown command
Usage: java FsShell
           [-ls <path>]
           [-lsr <path>]
           [-du <path>]
           [-dus <path>]
           [-count[-q] <path>]
           [-mv <src> <dst>]
           [-cp <src> <dst>]
           [-rm [-skipTrash] <path>]
           [-rmr [-skipTrash] <path>]
           [-expunge]
           [-put <localsrc> ... <dst>]
           [-copyFromLocal <localsrc> ... <dst>]
           [-moveFromLocal <localsrc> ... <dst>]
           [-get [-ignoreCrc] [-crc] <src> <localdst>]
           [-getmerge <src> <localdst> [addnl]]
```

```
wcohen@shell2:~/naive-bayes-demo$ hadoop fs -ls rcv1/small/sharded
Found 10 items
-rw-r--r--   3 wcohen supergroup     606405 2013-06-08 01:45 /user/wcohen/rcv1/small/sharded/part-00000
-rw-r--r--   3 wcohen supergroup    1347611 2013-06-08 01:45 /user/wcohen/rcv1/small/sharded/part-00001
-rw-r--r--   3 wcohen supergroup     939307 2013-06-08 01:45 /user/wcohen/rcv1/small/sharded/part-00002
-rw-r--r--   3 wcohen supergroup    1284062 2013-06-08 01:45 /user/wcohen/rcv1/small/sharded/part-00003
-rw-r--r--   3 wcohen supergroup    1009890 2013-06-08 01:45 /user/wcohen/rcv1/small/sharded/part-00004
-rw-r--r--   3 wcohen supergroup    1206196 2013-06-08 01:45 /user/wcohen/rcv1/small/sharded/part-00005
-rw-r--r--   3 wcohen supergroup    1384658 2013-06-08 01:45 /user/wcohen/rcv1/small/sharded/part-00006
-rw-r--r--   3 wcohen supergroup    1299698 2013-06-08 01:45 /user/wcohen/rcv1/small/sharded/part-00007
-rw-r--r--   3 wcohen supergroup     928752 2013-06-08 01:45 /user/wcohen/rcv1/small/sharded/part-00008
-rw-r--r--   3 wcohen supergroup     806030 2013-06-08 01:45 /user/wcohen/rcv1/small/sharded/part-00009
```

```
wcohen@shell2:~/naive-bayes-demo$ hadoop fs -tail rcv1/small/sharded/part-00005
weak as the arrival of arbitraged cargoes from the West has put the local market under pressure. In Singapor
e, May swaps fell to $21.30/$21.50 per barrelin late trade on Thursday from $21.70/$21.90 on Wednesday. Whil
e in Tokyo, first-half June open-spec naphtha was assessed at $207.00/$208.00 per tonne, compared with late
Wednesday's $213.00/$214.00. Added to these factors, traders said that a few petrochemicals were due to go i
nto turnaround in the next few weeks which would further dampen demand. -- Melanie Goodfellow, London Newsro
om, +44 171 542 7714.
M14,M143,MCAT    The Brent crude market on the Singapore International Monetary Exchange (SIMEX) will be clo
sed on Friday and Monday, SIMEX officials said on Tuesday. The closure will mark the corresponding closure o
n Friday and Monday of the Brent market on the International Petroleum Exchange (IPE) in London. SIMEX Brent
 operates a mutual offset system with the IPE in London, so SIMEX tends to close in line with the U.K.. --Si
ngapore Newsroom (+65 870 3081)
```

To train with streaming Hadoop:

- One way to shard text:
  - hadoop fs -put LocalFileName HDFSName
  - then run a streaming job with ‘cat’ as mapper and reducer
  - and specify the number of shards you want with option -numReduceTasks

To train with streaming Hadoop:

- Next, prepare your code for upload and distribution to the machines cluster

```
nb.jar: StreamSumReducer.java StreamNB.java SmallStreamNB.java
        javac -d classes StreamSumReducer.java StreamNB.java SmallStreamNB.java
        jar -cvf nb.jar -C classes .
```

To train with streaming Hadoop:

- Next, prepare your code for upload and distribution to the machines cluster

```
nb.jar: StreamSumReducer.java StreamNB.java SmallStreamNB.java
        javac -d classes StreamSumReducer.java StreamNB.java SmallStreamNB.java
        jar -cvf nb.jar -C classes .
```

Now you can run streaming Hadoop:

```
STRJAR = /usr/lib/hadoop/contrib/streaming/hadoop-streaming-1.2.0.1.3.0.0-107.jar

small-events-hs:
        hadoop fs -rmr rcv1/small/events
        time hadoop jar $(STRJAR) \
           -input rcv1/small/sharded -output rcv1/small/events \
        -mapper 'java -Xmx512m -cp ./lib/nb.jar com.wcohen.StreamNB' \
        -reducer 'java -Xmx512m -cp ./lib/nb.jar com.wcohen.StreamSumReducer' \
        -file nb.jar -numReduceTasks 10
```

“Real” Hadoop

- Streaming is simple but
  - There’s no typechecking of inputs/outputs
  - You need to parse strings a lot
  - You can’t use compact binary encodings
  - …
  - basically you have limited *control* over the messages you’re sending
    - i/o costs = O(message size) often dominates

MR code: Word count Main

```
public static void main(String[] args) throws Exception {

 Configuration conf = new Configuration();

 Job job = new Job(conf, "wordcount");

 job.setMapperClass(Map.class);
 job.setReducerClass(Reduce.class);

 job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);

     job.setOutputKeyClass(Text.class);

    job.setOutputValueClass(IntWritable.class);

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    job.waitForCompletion(true);
}
```

others:

- KeyValueInputFormat
- SequenceFileInputFormat

MR Code: Word Count Map

```
  public static class Map extends 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, Context context) throws <stuff> {
    String line = value.toString();
    StringTokenizer tokenizer = new StringTokenizer(line);
    while (tokenizer.hasMoreTokens()) {
        word.set(tokenizer.nextToken());
        context.write(word, one);
    }
  }
}
```

MR code: Word count Reduce

```
  public static class Reduce extends Reducer<Text, IntWritable, Text,
  IntWritable> {

public void reduce(Text key, Iterable<IntWritable> values, Context context)
  throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values) {
        sum += val.get();
    }
    context.write(key, new IntWritable(sum));
  }
}
```

Is any part of this wasteful?

- Remember - moving data around and writing to/reading from disk are very expensive operations
- No reducer can start until:
    - all mappers are done
    - data in its partition has been sorted

How much does buffering help?

BUFFER_SIZE	Time	Message Size
none		1.7M *words*
100	47s	1.2M
1,000	42s	1.0M
10,000	30s	0.7M
100,000	16s	0.24M
1,000,000	13s	0.16M
limit		0.05M

Combiners

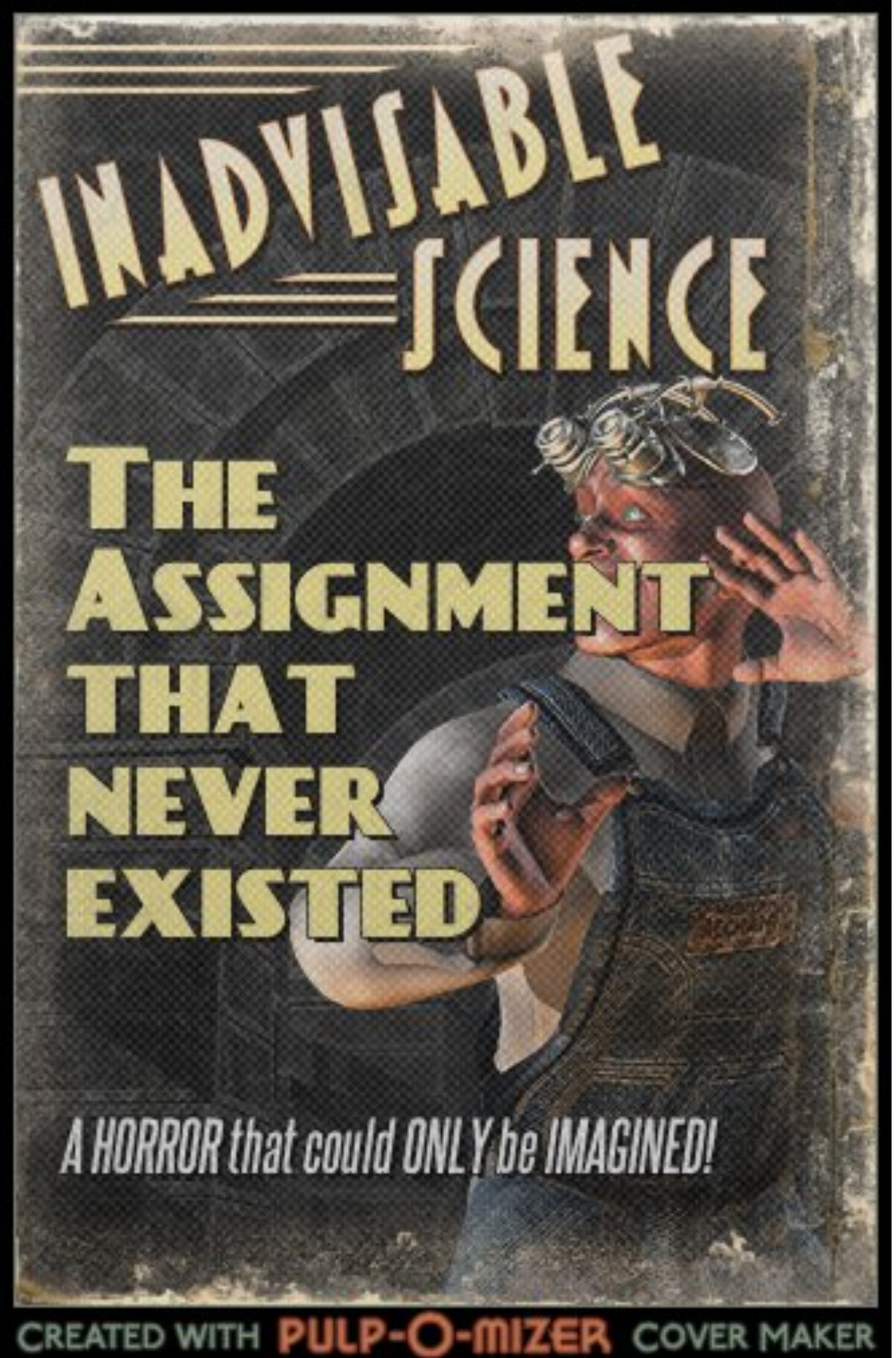
- Sits between the map and the shuffle
  - Do some of the reducing while you're waiting for other stuff to happen
  - Avoid moving all of that data over the network
- Only applicable when
  - order of reduce values doesn't matter
  - effect is cumulative

```
    public static class Combiner extends Reducer<Text, IntWritable, Text,
    IntWritable> {

    public void reduce(Text key, Iterable<IntWritable> values, Context context)
      throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        context.write(key, new IntWritable(sum));
    }
}
```

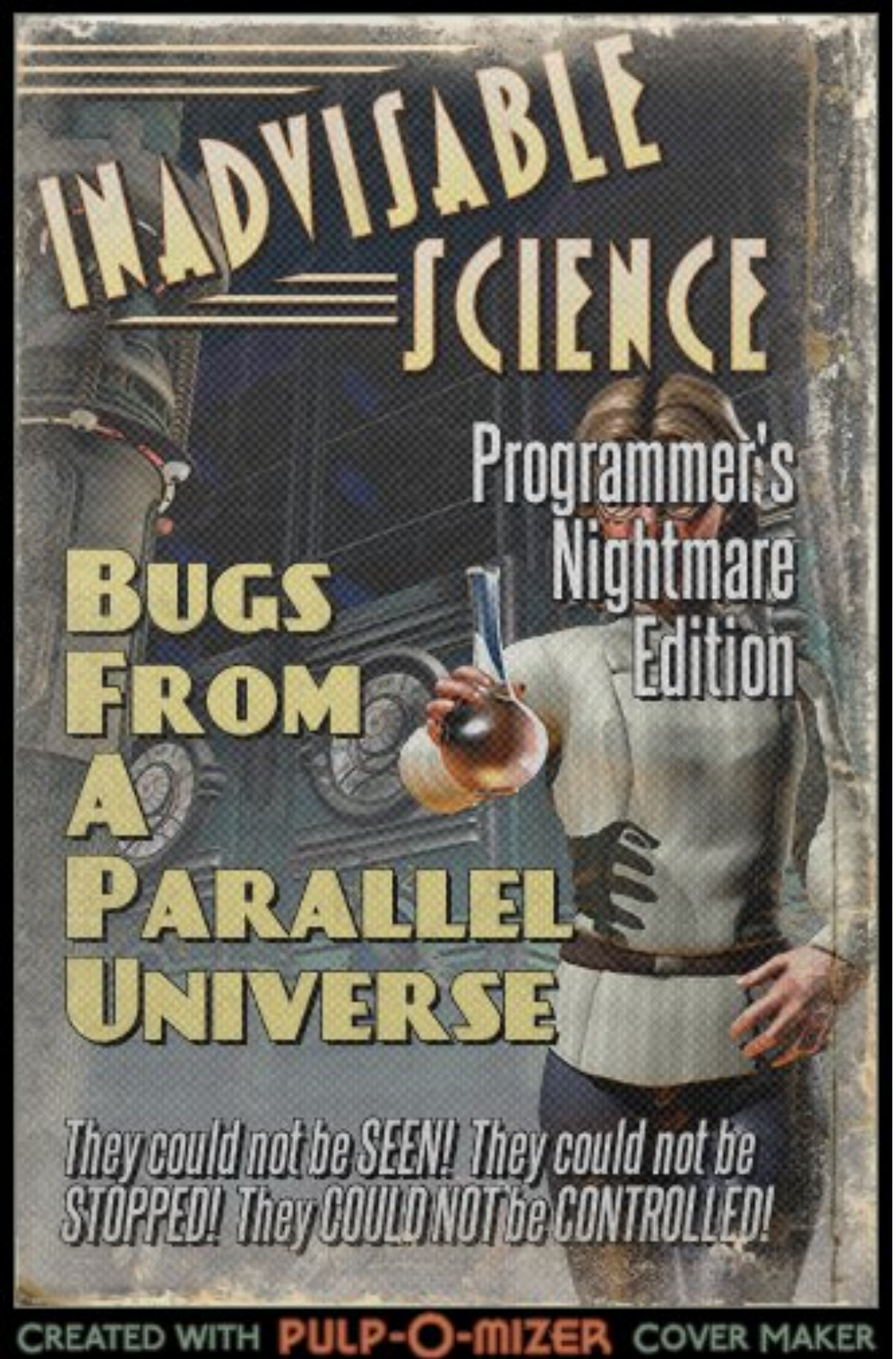
Deja vu: Combiner = Reducer

- Often the combiner is the reducer.
  - like for word count
  - but not always



1. This would be pretty systems-y (remote copy files, waiting for remote processes, …)
2. It would take work to make work for 500 jobs

- Reliability: *Replication, restarts, monitoring jobs,…*
- Efficiency: *load-balancing, reducing file/network i/o, optimizing file/network i/o, …*
- Useability: *stream defined datatypes, simple reduce functions, ….*


INADVISABLE SCIENCE
Programmer's Nightmare Edition
BUGS FROM A PARALLEL UNIVERSE
They could not be SEEN! They could not be STOPPED! They COULD NOT be CONTROLLED!
CREATED WITH PULP-O-MIZER COVER MAKER

Some common pitfalls

- You have no control over the order in which reduces are performed
- You have “no” control over the order in which you encounter reduce values
    - More on this later
- The only ordering you should assume is that Reducers always start after Mappers

Some common pitfalls

- You should assume your Maps and Reduces will be taking place on different machines with different memory spaces
- Don't make a static variable and assume that other processes can read it
  - They can't.
  - It appear that they can when run locally, but they can't
  - No really, don't do this.

Some common pitfalls

- Do not communicate between mappers or between reducers
    - overhead is high
    - you don't know which mappers/reducers are actually running at any given point
    - there's no easy way to find out what machine they're running on
        - because you shouldn't be looking for them anyway

When mapreduce doesn't fit

- The beauty of mapreduce is its separability and independence
- If you find yourself trying to communicate between processes
  - you're doing it wrong
    - or
  - what you're doing is not a mapreduce

When mapreduce doesn't fit

- Not everything is a mapreduce
- Sometimes you need more communication
  - We'll talk about other programming paradigms later

What's so tricky about MapReduce?

- Really, nothing. It's easy.
- What's often tricky is figuring out how to write an *algorithm* as a series of *map-reduce* substeps.
  - How and when do you parallelize?
  - When should you even try to do this? when should you use a different model?
- Last few lectures we've stepped through a few algorithms as examples
  - good exercise: think through some alternative implementations in map-reduce

Thinking in Mapreduce

- A new task: Word co-occurrence statistics (simplified)
- Input:
    - Sentences
- Output:
    - P(Word B is in sentence | Word A started the sentence)

Thinking in mapreduce

- We need to calculate
- P(B in sentence | A started sentence) =
  - P(B in sentence & A started sentence)/P(A started sentence)=
    - count<A,B>/count<A,*>

Word Co-occurrence: Solution 1

- The Pairs paradigm:
  - For each sentence, output a pair
  - E.g Map("Machine learning for big data") creates:
    - <Machine, learning>:1
    - <Machine, for>:1
    - <Machine, big>:1
    - <Machine, data>:1
    - <Machine,*>:1

Word Co-occurrence: Solution 1

- Reduce would create, for example:
  - <Machine, learning>:10
  - <Machine, for>:1000
  - <Machine, big>:50
  - <Machine, data>:200
  - ...
  - <Machine,*>:12000

Word Co-occurrence: Solution 1

- P(B in sentence | A started sentence) =
  - P(B in sentence & A started sentence)/P(A started sentence)=
    - <A,B>/<A,*>

- Do we have what we need?
  - Yes!

Word Co-occurrence: Solution 1

- But wait!
  - There's a problem .... can you see it?

Word Co-occurrence: Solution 1

- Each reducer will process all counts for a <word1,word2> pair
- We need to know <word1,*> at the same time as <word1,word2>
- The information is in different reducers!

Word Co-occurrence: Solution 1

- Solution 1 a)
  - Make the first word the reduce key
  - Each reducer has:
    - key: word_i
    - values: <word_i,word_j>....<word_i,word_b>.....<word_i,*>....

Word Co-occurrence: Solution 1

- Now we have all the information in the same reducer
- But, now we have a new problem, can you see it?

- Hint: remember - we have no control over the order of values

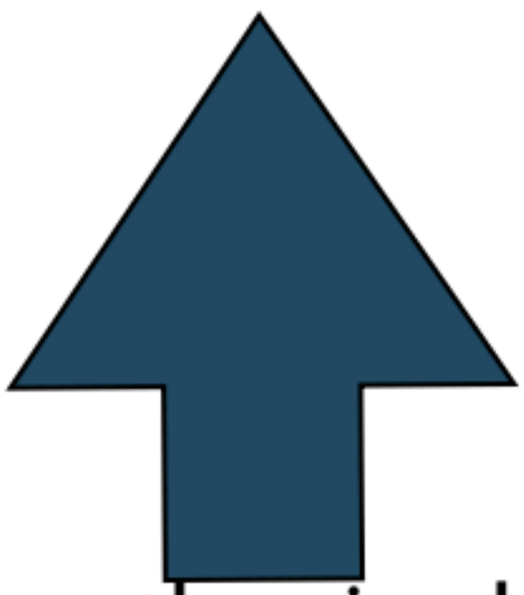
Word Co-occurrence: Solution 1

- There could be too many values to hold in memory
- We need <word_i,*> to be the first value we encounter
- Solution 1 b):
  - Keep <word_i,word_j> as the reduce key
  - Change the way Hadoop does its partitioning.

Word Co-occurrence: Solution 1

```
  /**

 * Partition based on the first part of the pair.
 */
public static class FirstCommaPartitioner extends Partitioner<Text, Text> {
    @Override
    public int getPartition(Text key, Text value, int numPartitions) {
        String[] s = key.toString().split(",");
        return (s[0].hashCode() & Integer.MAX_VALUE) % numPartitions;
    }
}
```



Removes the sign bit, if set

Word Co-occurrence: Solution 1

- Ok cool, but we still have the same problem.
  - The information is all in the same reducer, but we don't know the order
- But now, we have all the information we need in the reduce key!

Word Co-occurrence: Solution 1

- We can use a custom comparator to sort the keys we encounter in the reducer
    - One way: custom key class which implements WriteableComparable

- Aside: if you use tab-separated values and Hadoop streaming you can create a streaming job where (for instance) field 1 is the partition key, and the lines are sorted by fields 1 & 2.

Word Co-occurrence: Solution 1

- Now the order of key, value pairs will be as we need:
    - <Machine,*>:12000
    - <Machine, big>:50
    - <Machine, data>:200
    - <Machine, for>:1000
    - <Machine, learning>:10
        - ...
- P("big" in sentence | "Machine" started sentence) = 50/12000

Word Co-occurrence: Solution 2

- The Stripes paradigm
- For each sentence, output a key, record pair
  - E.g Map(“Machine learning for big data”) creates:
    - <Machine>:<*:1,learning:1, for:1, big:1, data:1>
  - E.g Map(“Machine parts are for machines”) creates:
    - <Machine>:<*:1,parts:1,are:1, for:1,machines:1>

Word Co-occurrence: Solution 2

- Reduce combines the records:
  - E.g Reduce for key <Machine> receives values:
    - <*:1,learning:1, for:1, big:1, data:1>
    - <*:1,parts:1,are:1, for:1,machines:1>
  - And merges them to create
  - <*:2,learning:1, for:2, big:1, data:1,parts:1,are:1,machines:1>

Word Co-occurrence: Solution 2

- This is nice because we have the * count already created
  - we just have to ensure it always occurs first in the record

Word Co-occurrence: Solution 2

- There is a really big (ha ha) problem with this solution
  - Can you see it?

- The value may become too large to fit in memory

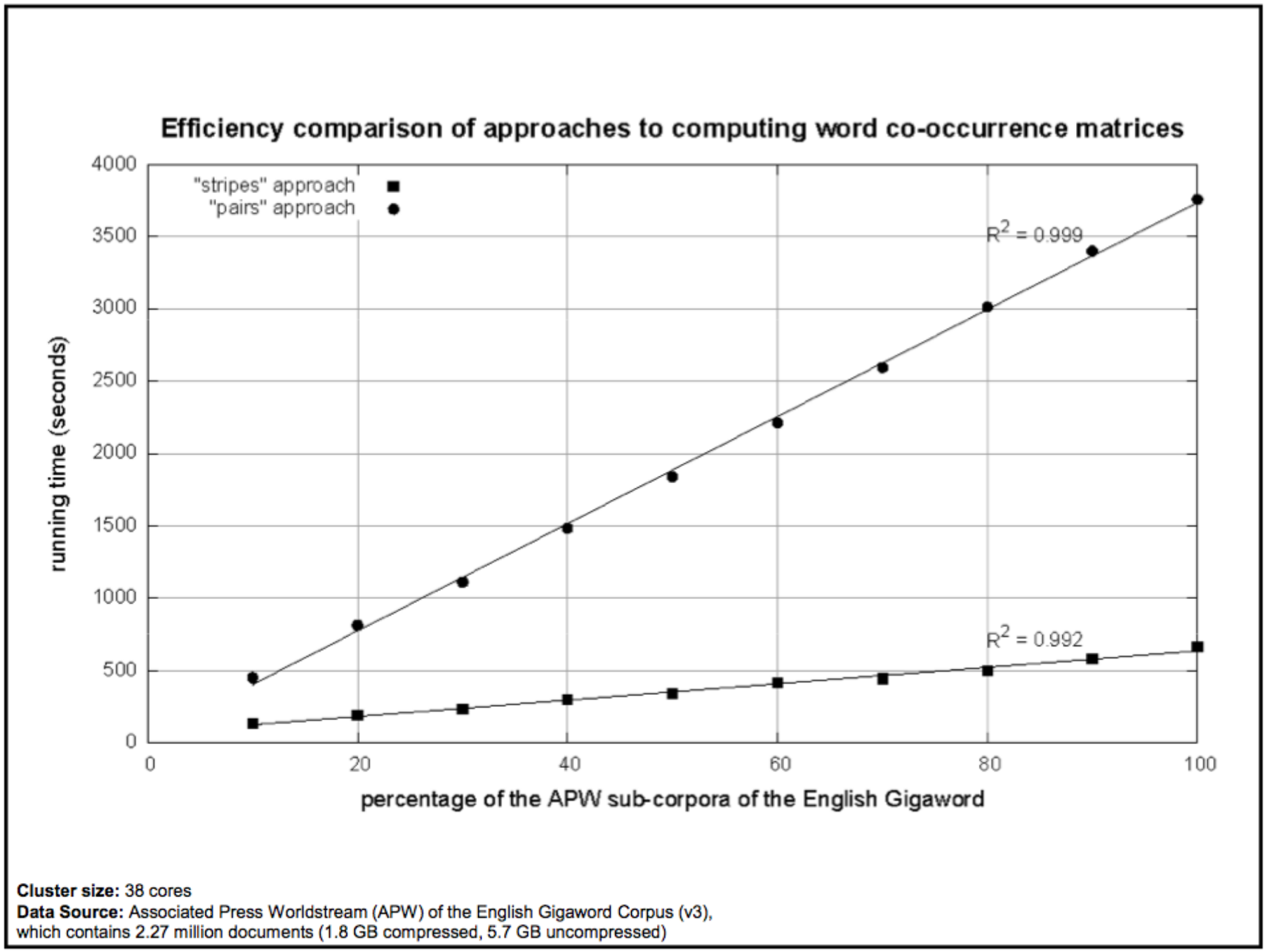
Performance

- IMPORTANT
  - You may not have room for all reduce values in memory
    - In fact you should PLAN not to have memory for all values
    - Remember, small machines are much cheaper
      - you have a limited budget

Performance

- Which is faster, stripes vs pairs?
  - Stripes has a bigger value per key
  - Pairs has more partition/sort overhead

Performance



Cluster size: 38 cores
Data Source: Associated Press Worldstream (APW) of the English Gigaword Corpus (v3), which contains 2.27 million documents (1.8 GB compressed, 5.7 GB uncompressed)

Conclusions

- Mapreduce
  - Can handle big data
  - Requires minimal code-writing
  - Real algorithms are typically a *sequence* of map-reduce steps