Big Data
Rethink Algos and Architecture

Scott Marsh

Manager R&D

Personal Lines Auto Pricing

Agenda

- History
- Map Reduce
- Algorithms

History

- Google talks about their solutions to their problems
  - Map Reduce: http://research.google.com/archive/mapreduce.html
  - Google File System: http://research.google.com/archive/gfs.html
  - Big Table: http://research.google.com/archive/bigtable.html
- Yahoo says “Me too! Let’s Share!”
  - Reimplements in Java and Open Sources the code
  - Calls it Hadoop – Named after a stuffed elephant
  - Two components: HDFS & MR
- Facebook and others start building on top of HDFS/MR
  - Write a SQL to MR compiler on Hadoop called HIVE

Why HDFS

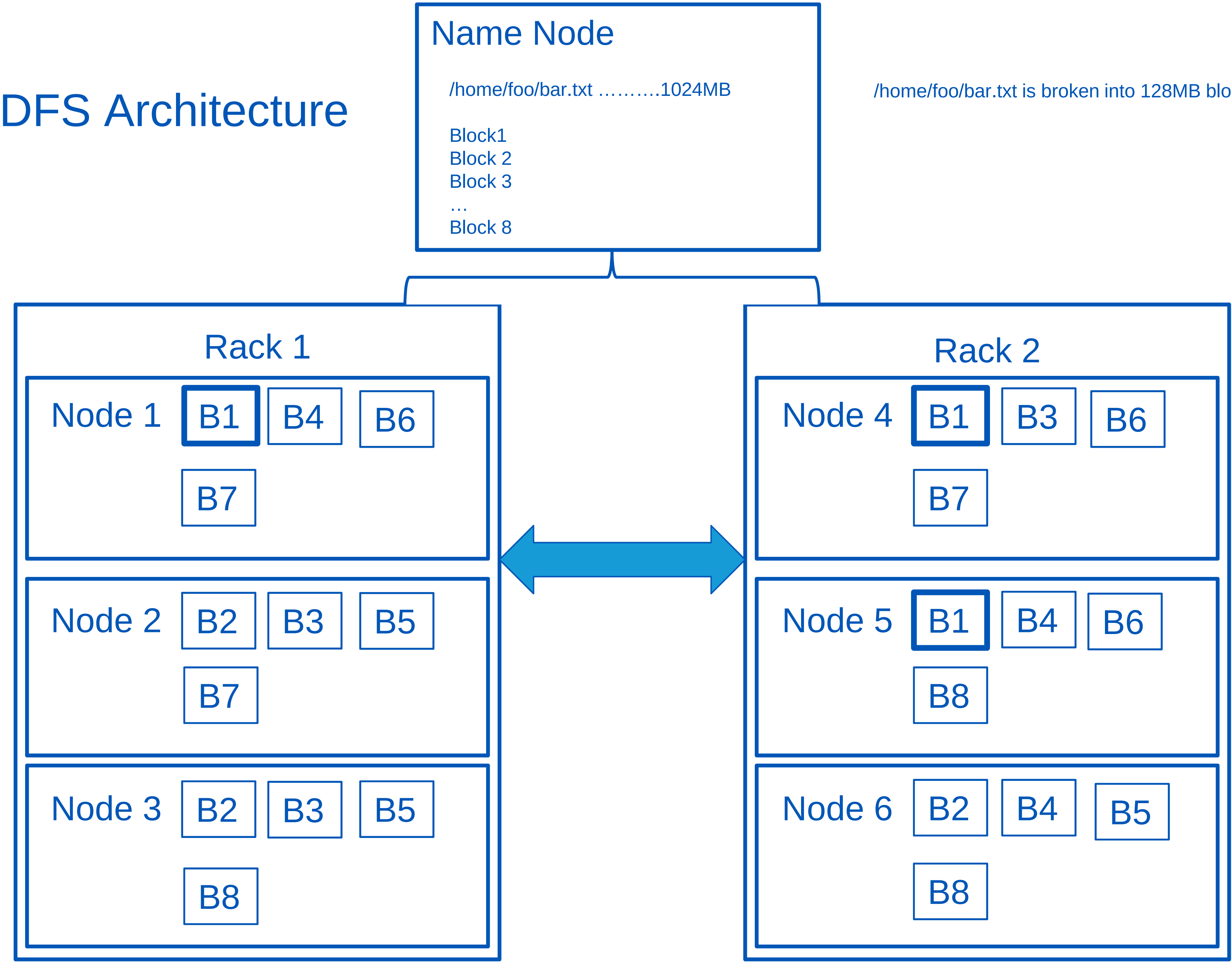
- Annualized Failure rate of a hard drive? ~3%
- How many machines? 4,500
- How many hard drives per machine? 4
- How many total hard drives? 18,000
- How many hard drives do I expect to lose per year? 540
  - Per week? ~10
- Houston we have a problem! Have to plan to lose data every day!

How to reduce the probability of failure?

- Replication to the rescue
- HDFS implements a default replication factor for data of 3X
- Even with a large cluster the probability of failure of 3 nodes within the timeframe that it take Hadoop to re-replicate is very low.
- It is left as an exercise to the reader:

  Assuming it takes Hadoop 1 minute to recognize and re-replicate the missing data to a new node what is the annualized probability of data loss? Assume the cluster has the same characteristics as the original Yahoo cluster.
- Bottom line: Replication fixes data loss worries at the software level

HDFS Architecture



Name Node

/home/foo/bar.txt ……….1024MB

Block1
Block 2
Block 3
…
Block 8

/home/foo/bar.txt is broken into 128MB blocks

Rack 1

Node 1 B1 B4 B6 B7

Node 2 B2 B3 B5 B7

Node 3 B2 B3 B5 B8

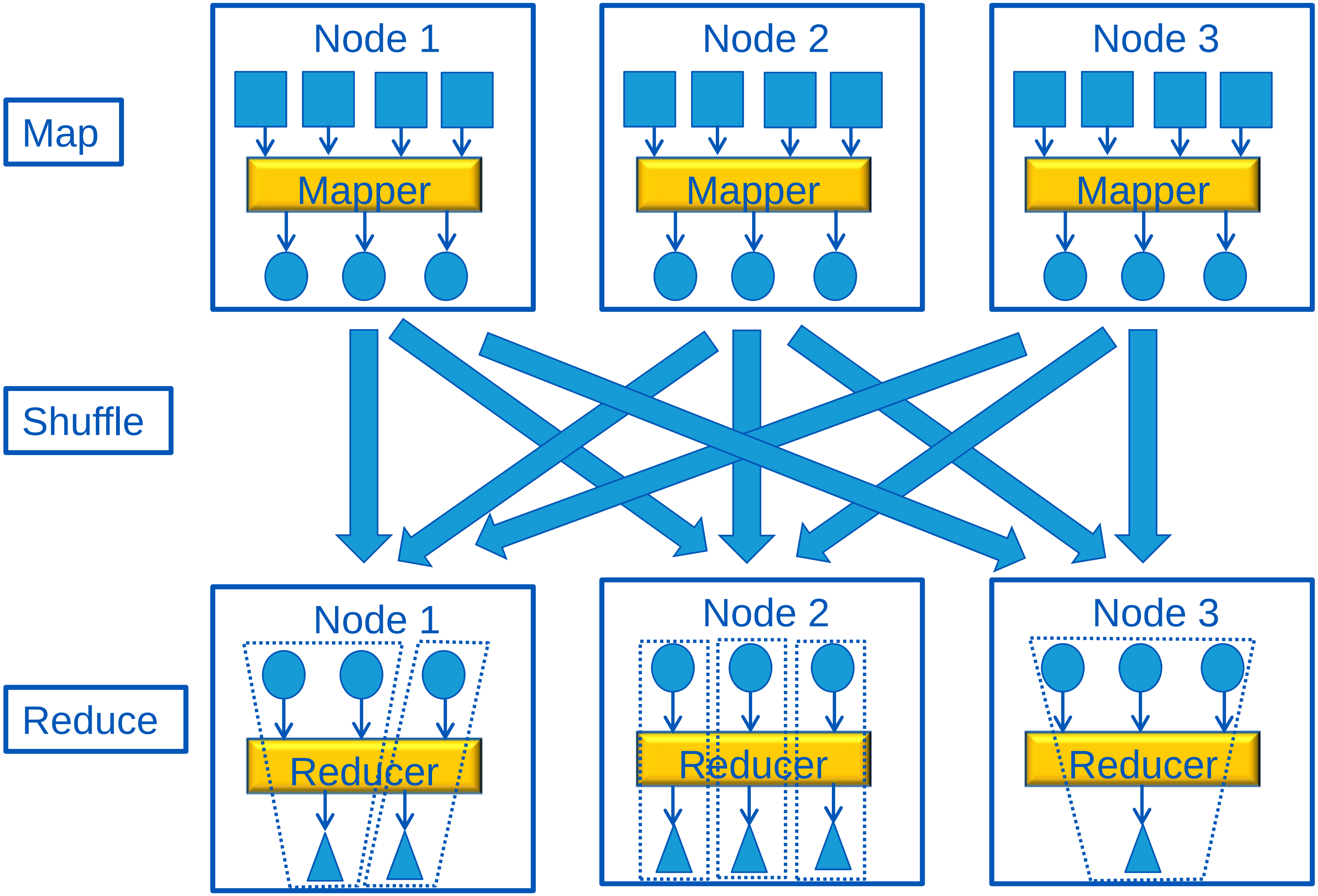
Rack 2

Node 4 B1 B3 B6 B7

Node 5 B1 B4 B6 B8

Node 6 B2 B4 B5 B8

MapReduce



Map

Shuffle

Reduce

Node 1

Mapper

Node 2

Mapper

Node 3

Mapper

Node 1

Reducer

Node 2

Reducer

Node 3

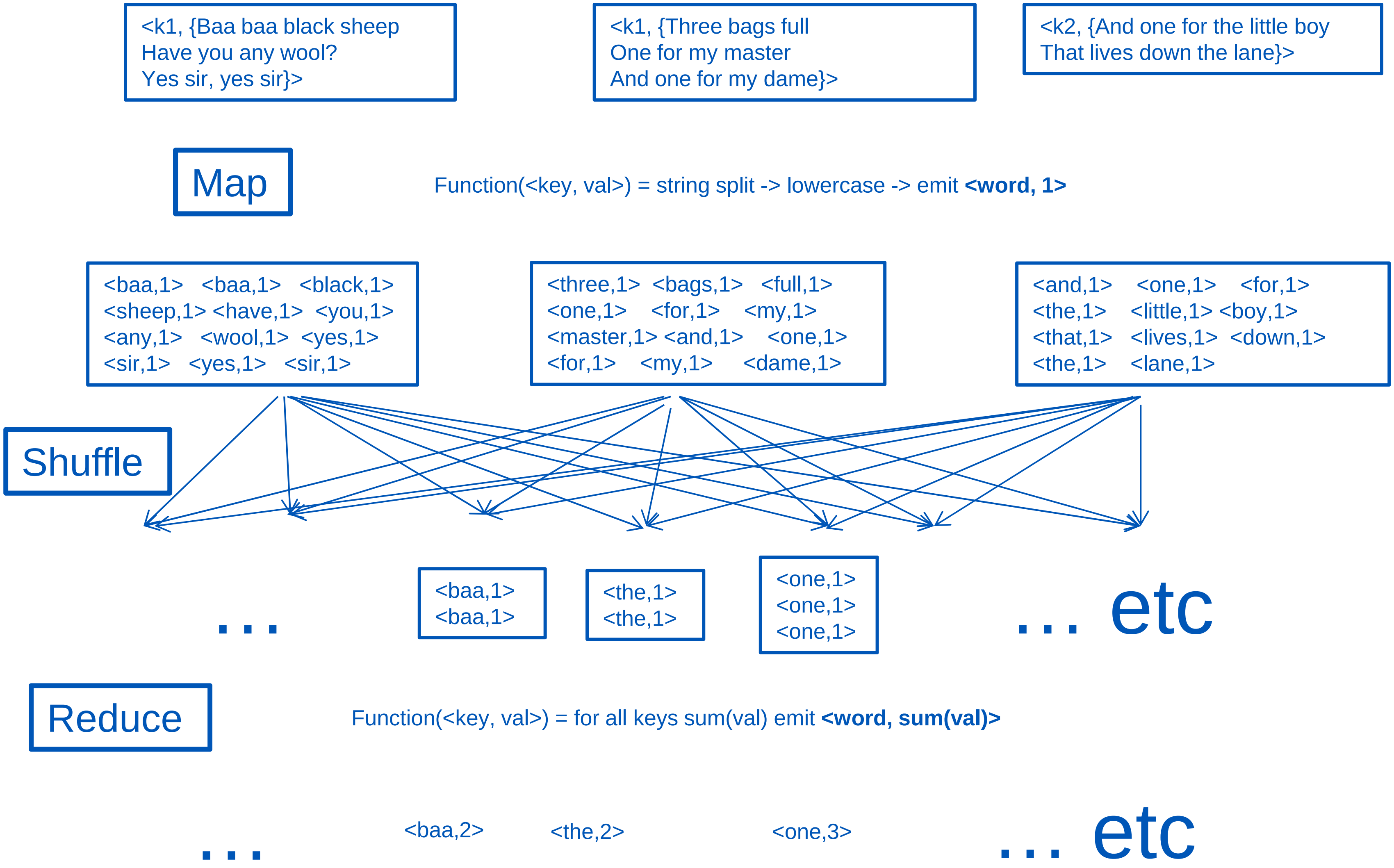
Reducer

Map Reduce More Concretely

- Mappers receive local data
- The mapping function is the same function for each piece of data it receives
- The output is a set of tuples <key, value>
- These <k,v> pairs are shuffled across the network s.t. all of the <k,v> pairs with the same key are received by the same reducer
- The reducer then runs a function over the set of <k,v> pairs also emits a new set of <k,v> pairs which are then stored on HDFS on the local node

Is your head spinning, did I just speak in Greek?

Map Reduce an Example (Wordcount)



<k1, {Baa baa black sheep
Have you any wool?
Yes sir, yes sir}>

<k1, {Three bags full
One for my master
And one for my dame}>

<k2, {And one for the little boy
That lives down the lane}>

Map

Function(<key, val>) = string split -> lowercase -> emit **<word, 1>**

<baa,1> <baa,1> <black,1>
<sheep,1> <have,1> <you,1>
<any,1> <wool,1> <yes,1>
<sir,1> <yes,1> <sir,1>

<three,1> <bags,1> <full,1>
<one,1> <for,1> <my,1>
<master,1> <and,1> <one,1>
<for,1> <my,1> <dame,1>

<and,1> <one,1> <for,1>
<the,1> <little,1> <boy,1>
<that,1> <lives,1> <down,1>
<the,1> <lane,1>

Shuffle

…

<baa,1>
<baa,1>

<the,1>
<the,1>

<one,1>
<one,1>
<one,1>

… etc

Reduce

Function(<key, val>) = for all keys sum(val) emit **<word, sum(val)>**

… <baa,2> <the,2> <one,3> … etc

Why Should I Care About MapReduce?

- Why should I want to constrain myself to a narrow programming pattern of Maps and Reduces?

	Single Machine	Cluster
Storage	Raid 0 SSD	HDD
# of Machines	1	50
# of Drives / Machine	1	8
Total Drives	1	400
Cost / GB Storage	Expensive	Cheap
Throughput GB/s	0.8	16
Time to Read 10TB	210 Min	10 Min

- It takes too long to bring the data to the program, so flip the paradigm, bring the program to the data!
- Data locality is important! Move the computation close to where the data is stored.

What do I get if I am willing to adopt Map Reduce?

“Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program’s execution across a set of machines, handling machine failures, and managing the required inter-machine communication. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system.”

-Jeffrey Dean and Sanjay Ghemawat

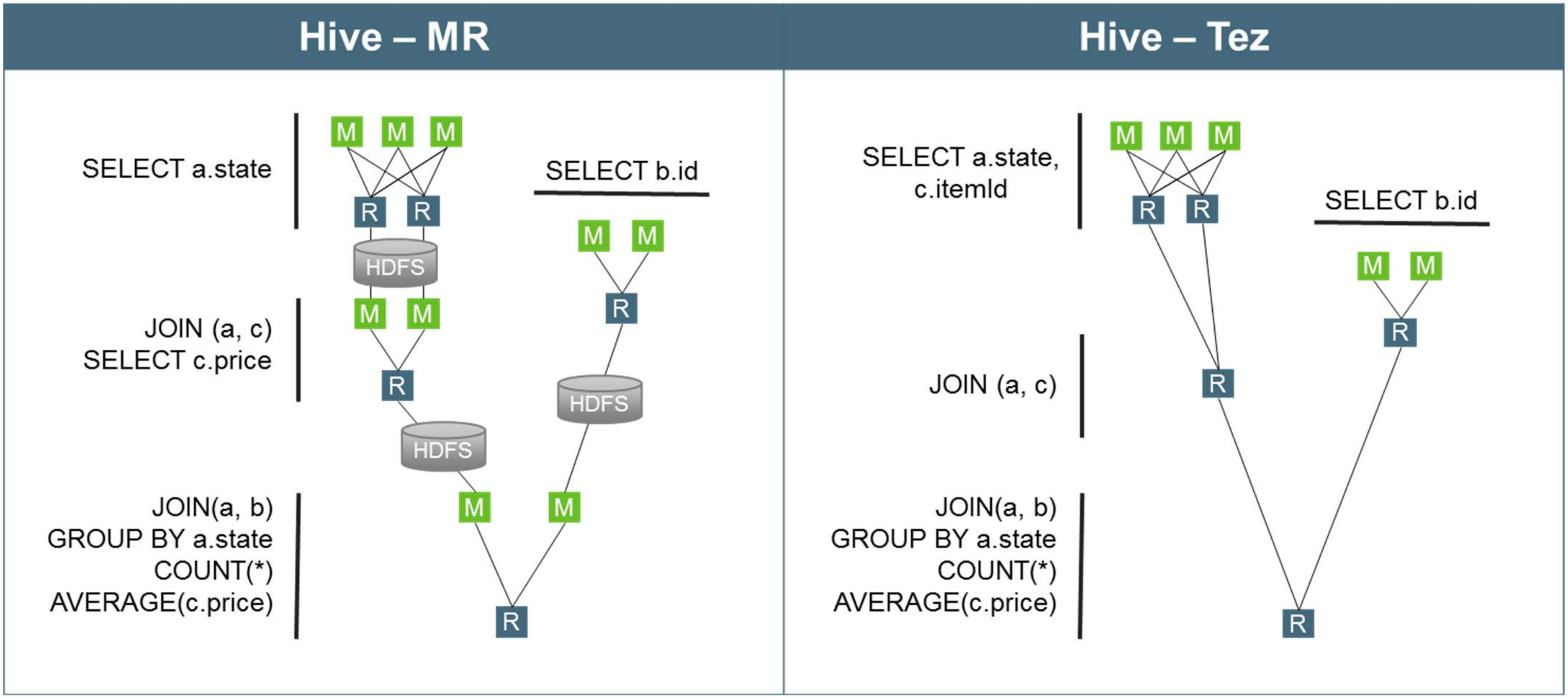
I don’t know, this still sounds too hard!

- Hadoop and MR are just the “operating system” not the ecosystem
- I know SQL, I don’t want to learn Java to write job on Hadoop
  - Ah, **HIVE** is a SQL to Map Reduce Compiler
- I already know Python, R, Insert your favorite language here
  - Ah, Let me introduce you to the **Streaming** Interface

The ecosystem is BIG and it is changing fast

- Buzz Words
  - YARN (I got a bunch of stuff to run on this cluster, how do I keep the kids in the back seat from punching each other)
  - Spark (More liberal programming construction / stays in memory)
  - Tez (Why do I keep writing this stuff to disk just to read it in the next phase)
  - MLLIB (Let’s predict stuff!)
  - HBASE (NOSQL)

Map Reduce versus Tez

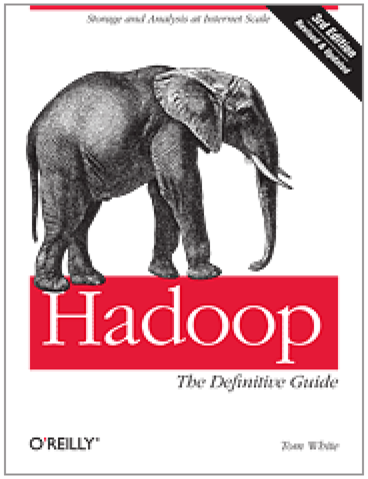


Real World Use Case

- NY Times
- Archive of images of every page of every issues of the newspaper between 1851–1980
- 4TB raw data
- Wanted to convert to PDFs
- 100 Node Hadoop Cluster in 24 hours
- Generated 1.5TB of PDF output

I want to learn more, where to I go next

- Download a VM (both the major commercial Hadoop vendors have a prepackage machine image you can run)
- Read “The Definitive Guide to Hadoop”
- Download the Airlines Dataset, learn how to process it with HIVE*



* http://randyzwitch.com/big-data-hadoop-amazon-ec2-cloudera-part-1/

Questions