cloudera
Ask Bigger Questions

Introduction to Apache Hadoop and its Ecosystem

Mark Grover | Budapest Data Forum
June 5th, 2015
@mark_grover
github.com/markgrover/hadoop-intro-fast

Schedule and Logistics

- Facilities
- Questions
- Schedule (times are approximate)

Time	Event
9:00 – 10:20	Tutorial
10:20 – 10:30	Break
10:30 – 12:00	Tutorial

About the Presentation…

- What's ahead
  - Fundamental Concepts
  - HDFS: The Hadoop Distributed File System
  - Data Processing with MapReduce
  - The Hadoop Ecosystem
  - Hadoop Clusters: Past, Present, and Future
  - Conclusion + Q&A

Fundamental Concepts

Why the World Needs Hadoop

Volume

- Every day…
  - More than 1.5 billion shares are traded on the NYSE
  - Facebook stores 2.7 billion comments and Likes
- Every minute…
  - Foursquare handles more than 2,000 check-ins
  - TransUnion makes nearly 70,000 updates to credit files
- Every second…
  - Banks process more than 10,000 credit card transactions

Velocity

- We are generating data faster than ever
  - Processes are increasingly automated
  - People are increasingly interacting online
  - Systems are increasingly interconnected

Variety

- We're producing a variety of data, including
  - Audio
  - Video
  - Images
  - Log files
  - Web pages
  - Product ratings
  - Social network connections
- Not all of this maps cleanly to the relational model

Big Data Can Mean Big Opportunity

- One tweet is an anecdote
  - But a million tweets may signal important trends
- One person's product review is an opinion
  - But a million reviews might uncover a design flaw
- One person's diagnosis is an isolated case
  - But a million medical records could lead to a cure

We Need a System that Scales

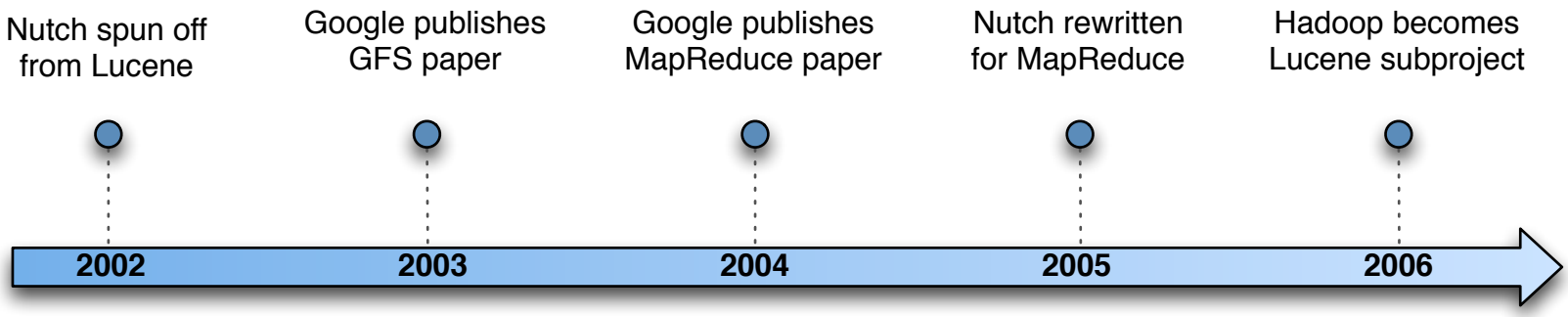
- Too much data for traditional tools
- Two key problems
  - How to reliably store this data at a reasonable cost
  - How to we process all the data we've stored

What is Apache Hadoop?

- Scalable data storage and processing
  - Distributed and fault-tolerant
  - Runs on standard hardware
- Two main components
  - Storage: Hadoop Distributed File System (HDFS)
  - Processing: MapReduce
- Hadoop *clusters* are composed of computers called *nodes*
  - Clusters range from a single node up to several thousand nodes

How Did Apache Hadoop Originate?

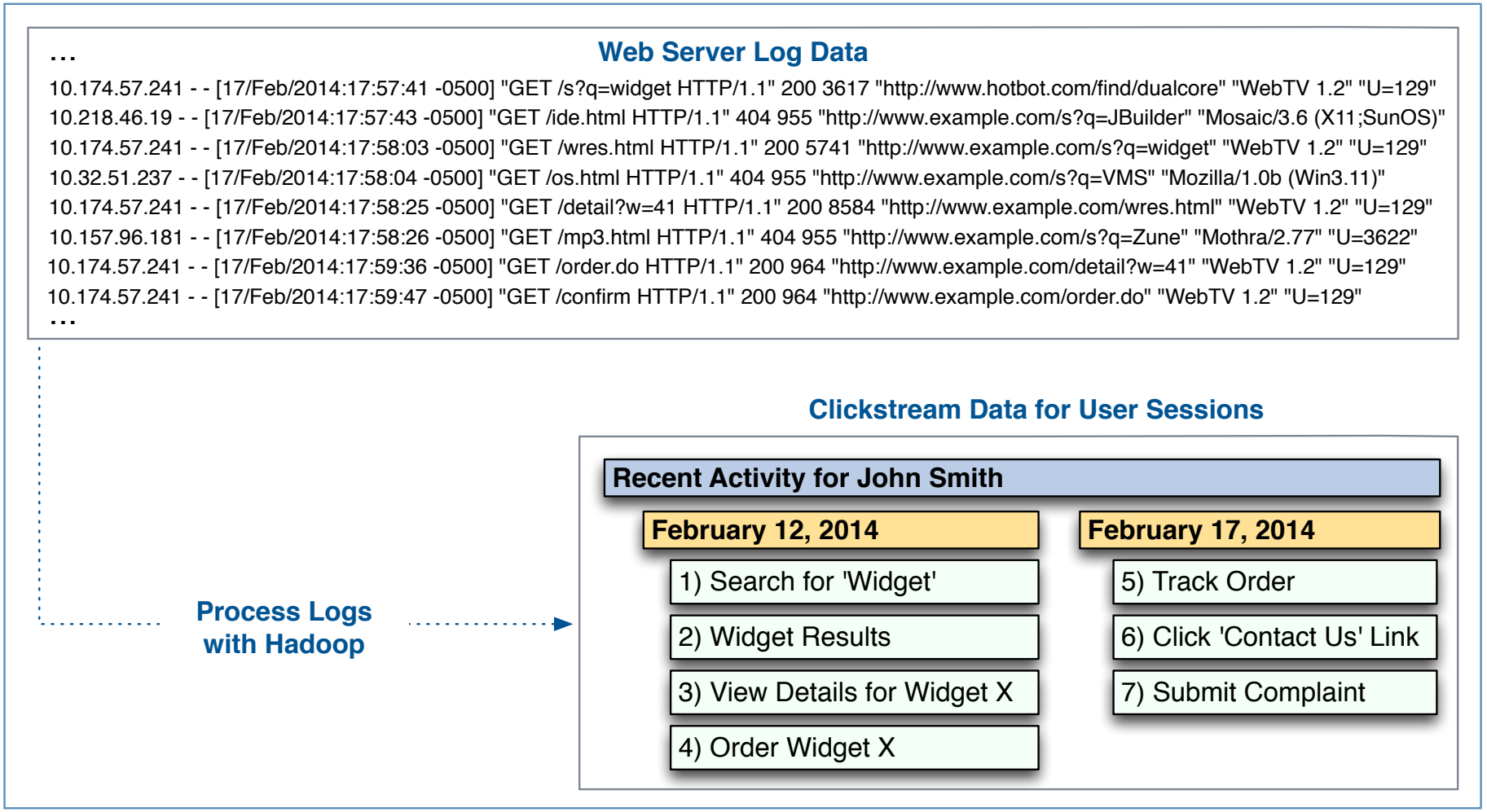
- Heavily influenced by Google's architecture
  - Notably, the Google Filesystem and MapReduce papers
- Other Web companies quickly saw the benefits
  - Early adoption by Yahoo, Facebook and others



How Are Organizations Using Hadoop?

- Let's look at a few common uses…

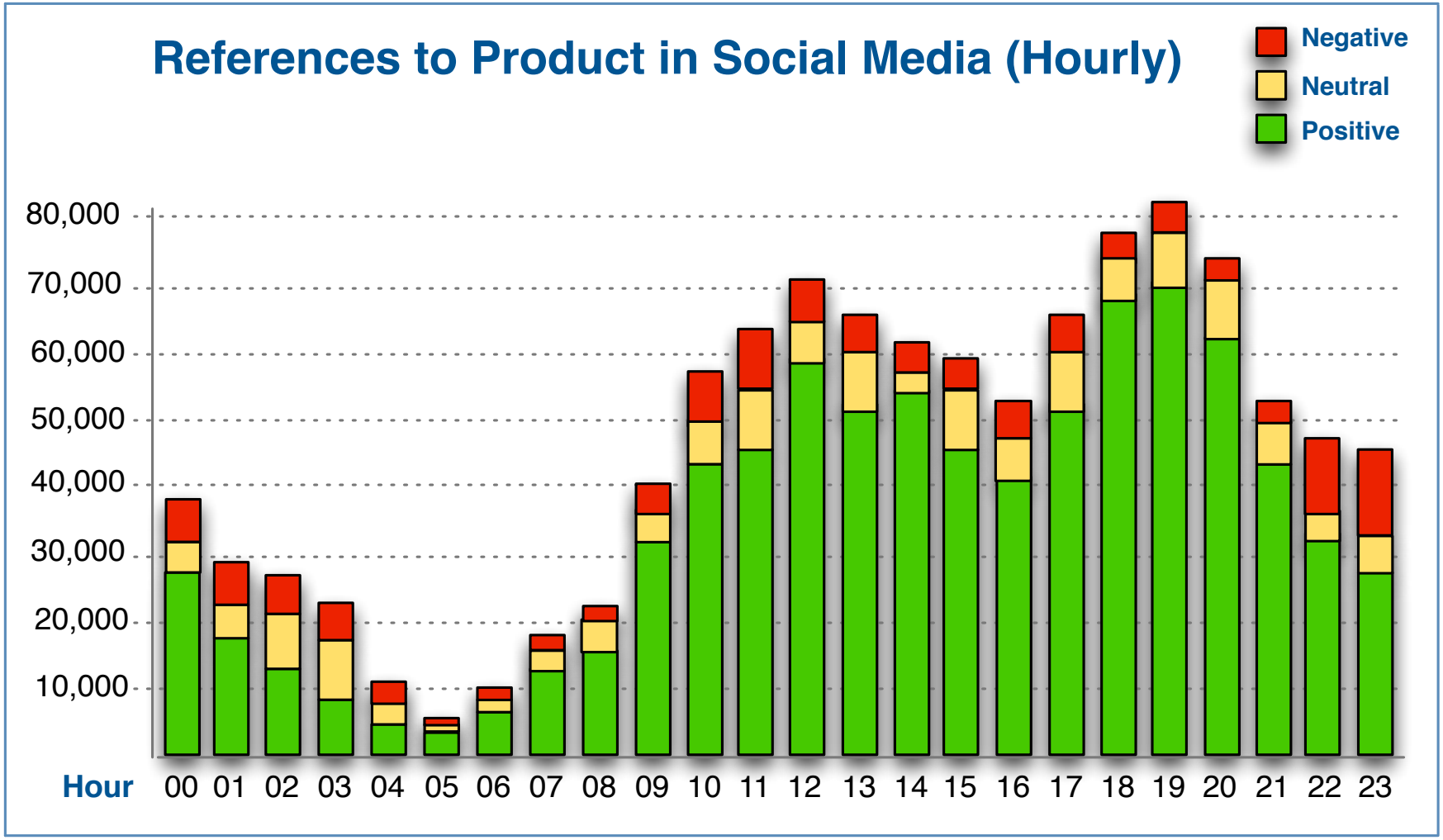
Hadoop Use Case: Log Sessionization



Hadoop Use Case: Customer Analytics

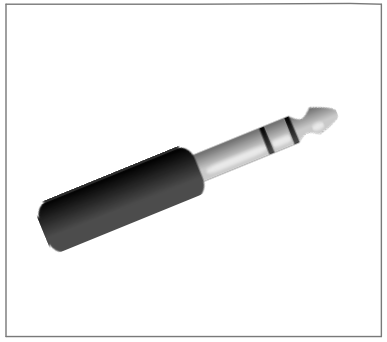
Example Inc. Public Web Site (February 9 - 15)					
Category	**Unique Visitors**	**Page Views**	**Average Time on Page**	**Bounce Rate**	**Conversion Rate**
Television	1,967,345	8,439,206	17 seconds	23%	51%
Smartphone	982,384	3,185,749	23 seconds	47%	41%
MP3 Player	671,820	2,174,913	42 seconds	61%	12%
Stereo	472,418	1,627,843	26 seconds	74%	19%
Monitor	327,018	1,241,837	37 seconds	56%	17%
Tablet	217,328	816,545	53 seconds	48%	28%
Printer	127,124	535,261	34 seconds	27%	64%

Hadoop Use Case: Sentiment Analysis



Hadoop Use Case: Product Recommendations

Products Recommended for You



Audio Adapter - Stereo

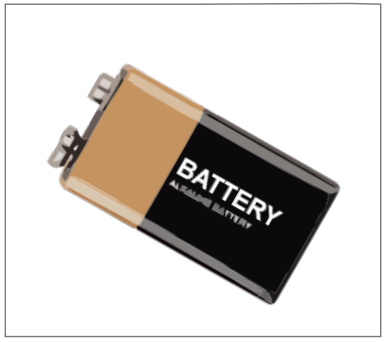
$4.99

(341 ratings)

Over-the-Ear Headphones

$29.99

(1,672 ratings)



9-volt Alkaline Battery

$1.79

(847 ratings)

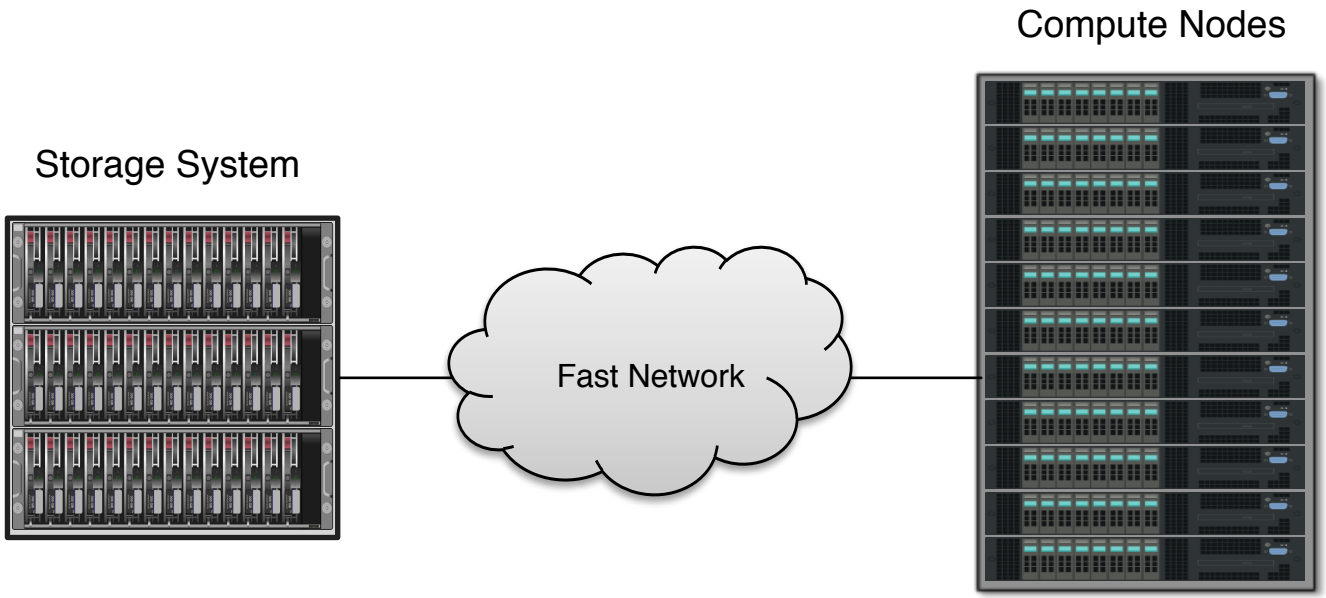
"In pioneer days they used oxen for heavy pulling, and when one ox couldn't budge a log, we didn't try to grow a larger ox"

— Grace Hopper, early advocate of distributed computing

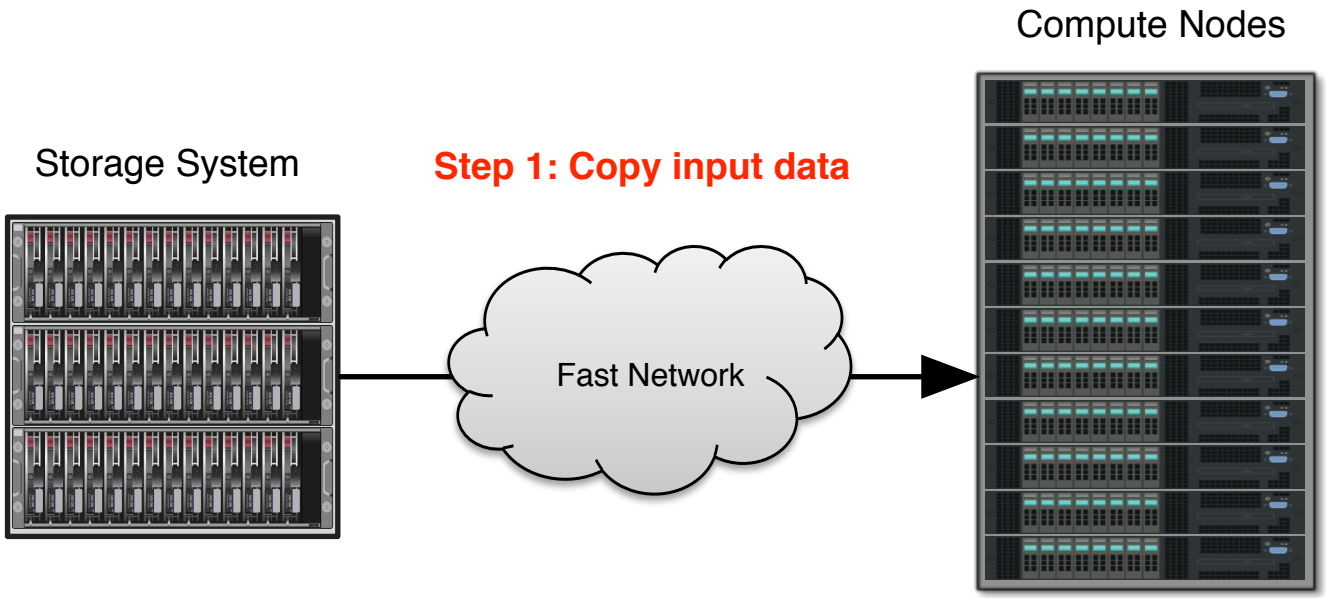
Comparing Hadoop to Other Systems

- Monolithic systems don't scale
- Modern high-performance computing systems are distributed
  - They spread computations across many machines in parallel
  - Widely-used used for scientific applications
  - Let's examine how a typical HPC system works

Architecture of a Typical HPC System

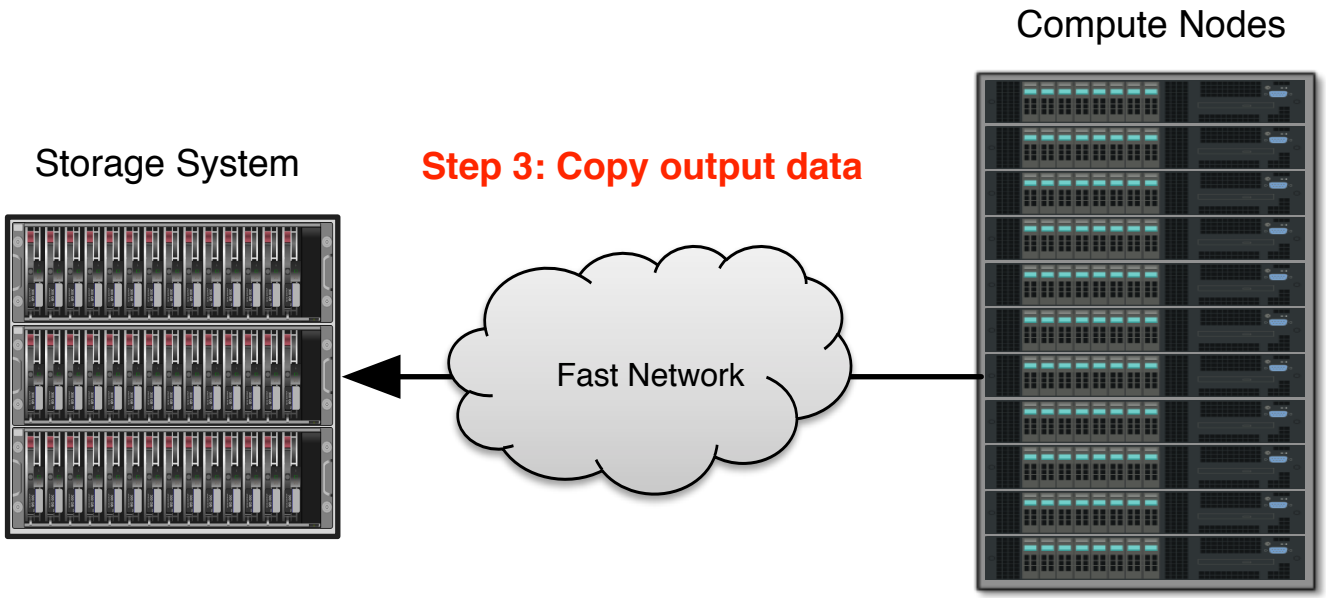


Architecture of a Typical HPC System



Architecture of a Typical HPC System

Architecture of a Typical HPC System

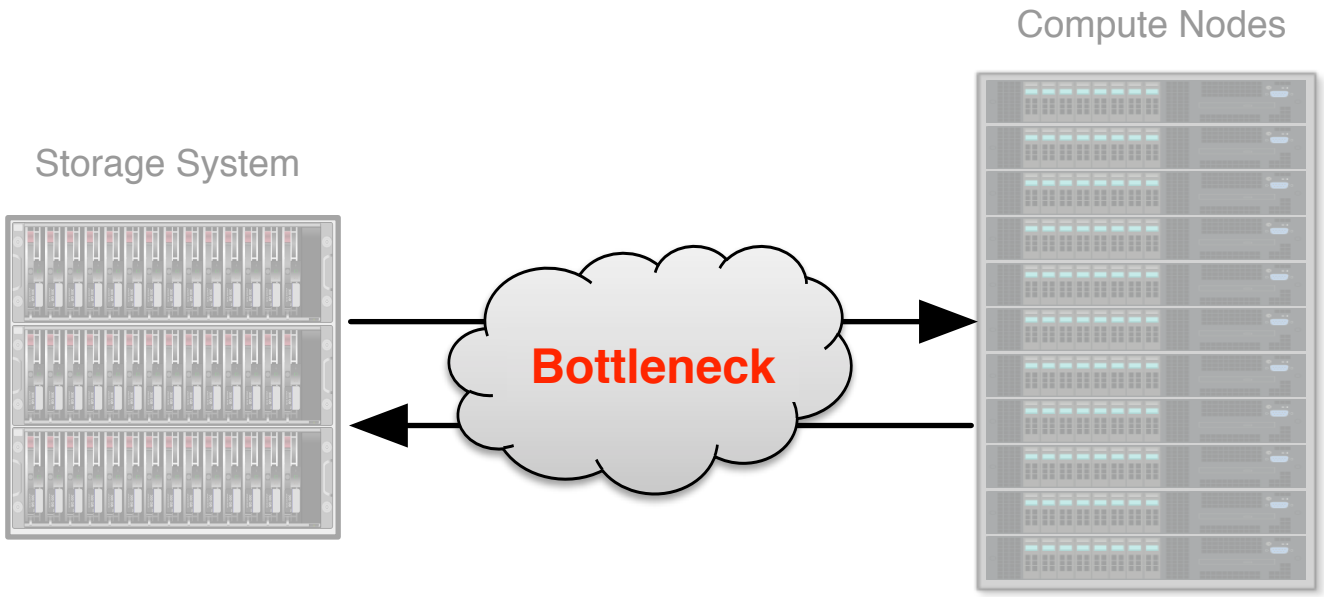


You Don’t Just Need Speed…

- The problem is that we have way more data than code

```
$ du -ks code/
1,087

$ du -ks data/
854,632,947,314
```

You Need Speed *At Scale*



Hadoop Design Fundamental: Data Locality

- This is a hallmark of Hadoop's design
  - Don't bring the data to the computation
  - Bring the computation to the data
- Hadoop uses the same machines for storage and processing
  - Significantly reduces need to transfer data across network

Other Hadoop Design Fundamentals

- Machine failure is unavoidable – embrace it
  - Build reliability into the system
- "More" is usually better than "faster"
  - Throughput matters more than latency

HDFS

The Hadoop Distributed Filesystem

HDFS: Hadoop Distributed File System

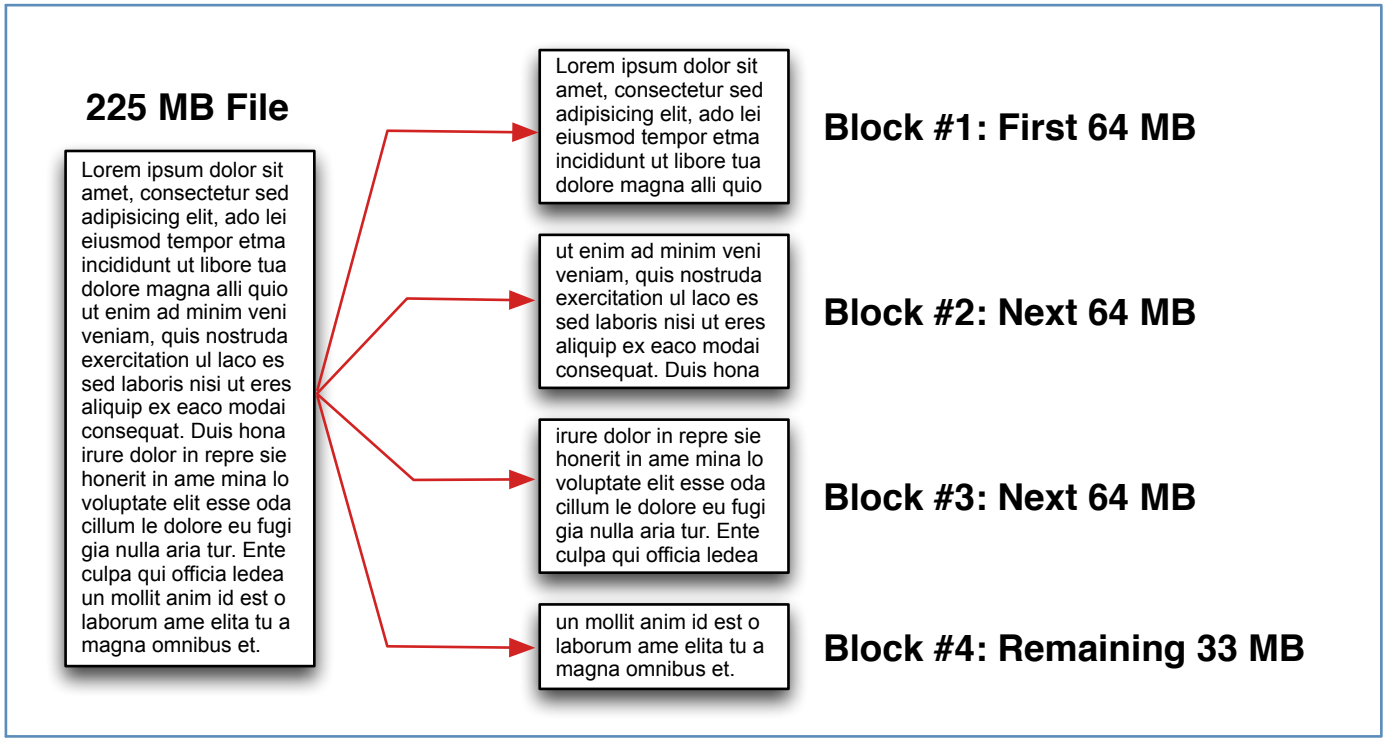
- Inspired by the Google File System
  - Reliable, low-cost storage for massive amounts of data
- Similar to a UNIX filesystem in some ways
  - Hierarchical
  - UNIX-style paths (e.g., `/sales/alice.txt`)
  - UNIX-style file ownership and permissions

HDFS: Hadoop Distributed File System

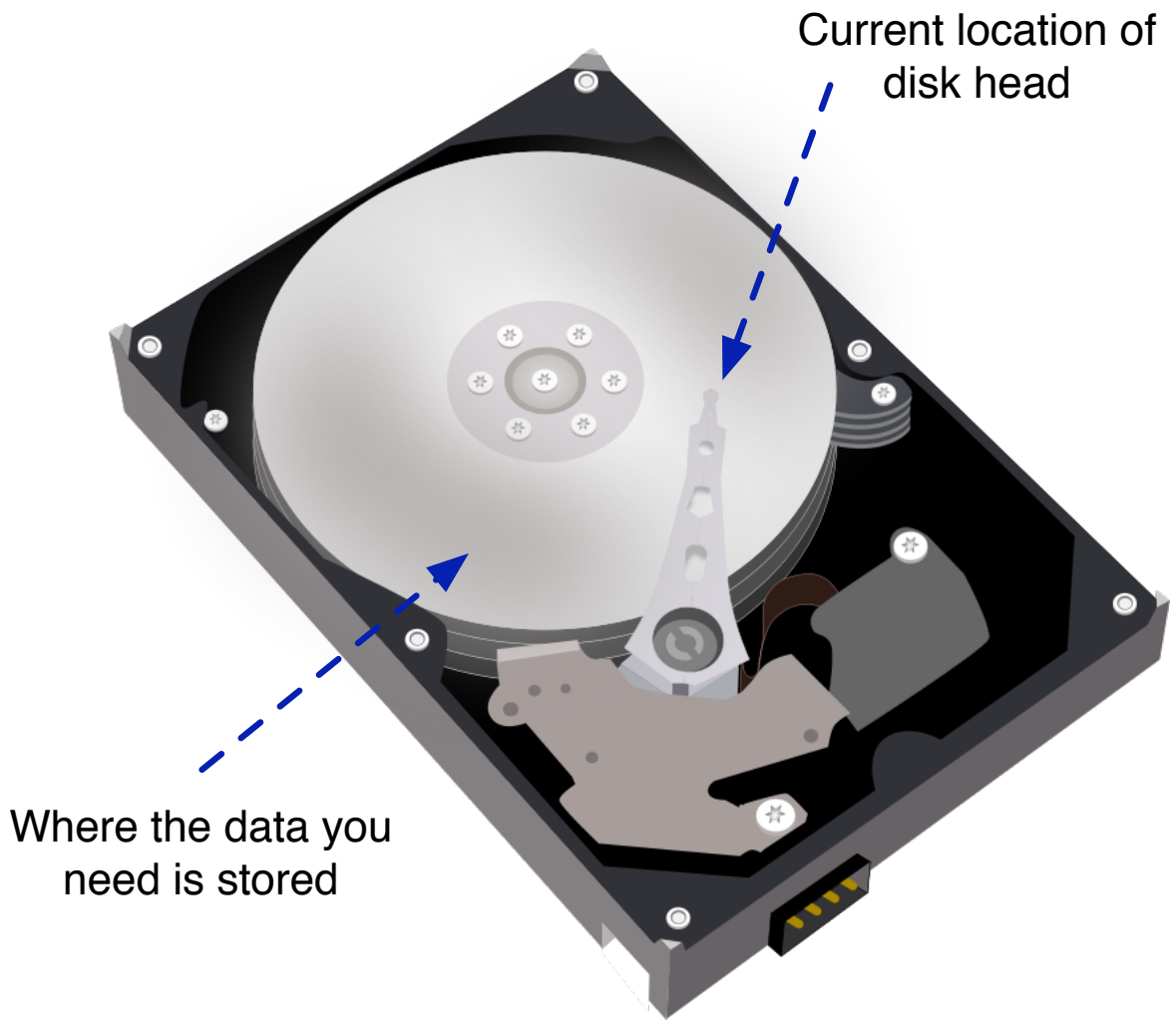
- There are also some major deviations from UNIX filesystems
- Highly-optimized for processing data with MapReduce
  - Designed for sequential access to large files
  - Cannot modify file content once written
- It's actually a user-space Java process
  - Accessed using special commands or APIs
- No concept of a current working directory

HDFS Blocks

- Files added to HDFS are split into fixed-size blocks
  - Block size is configurable, but defaults to 64 megabytes



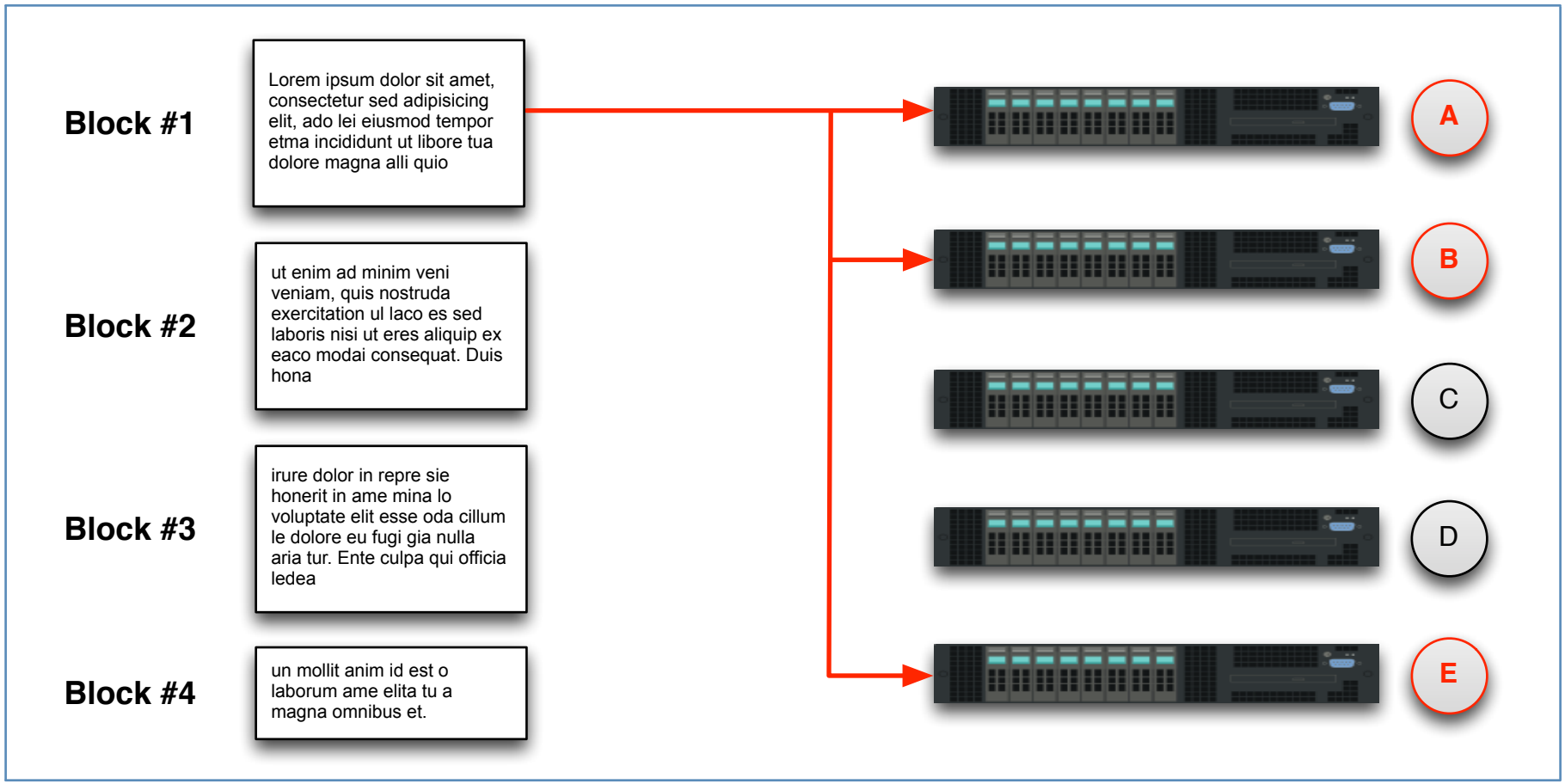
Why Does HDFS Use Such Large Blocks?



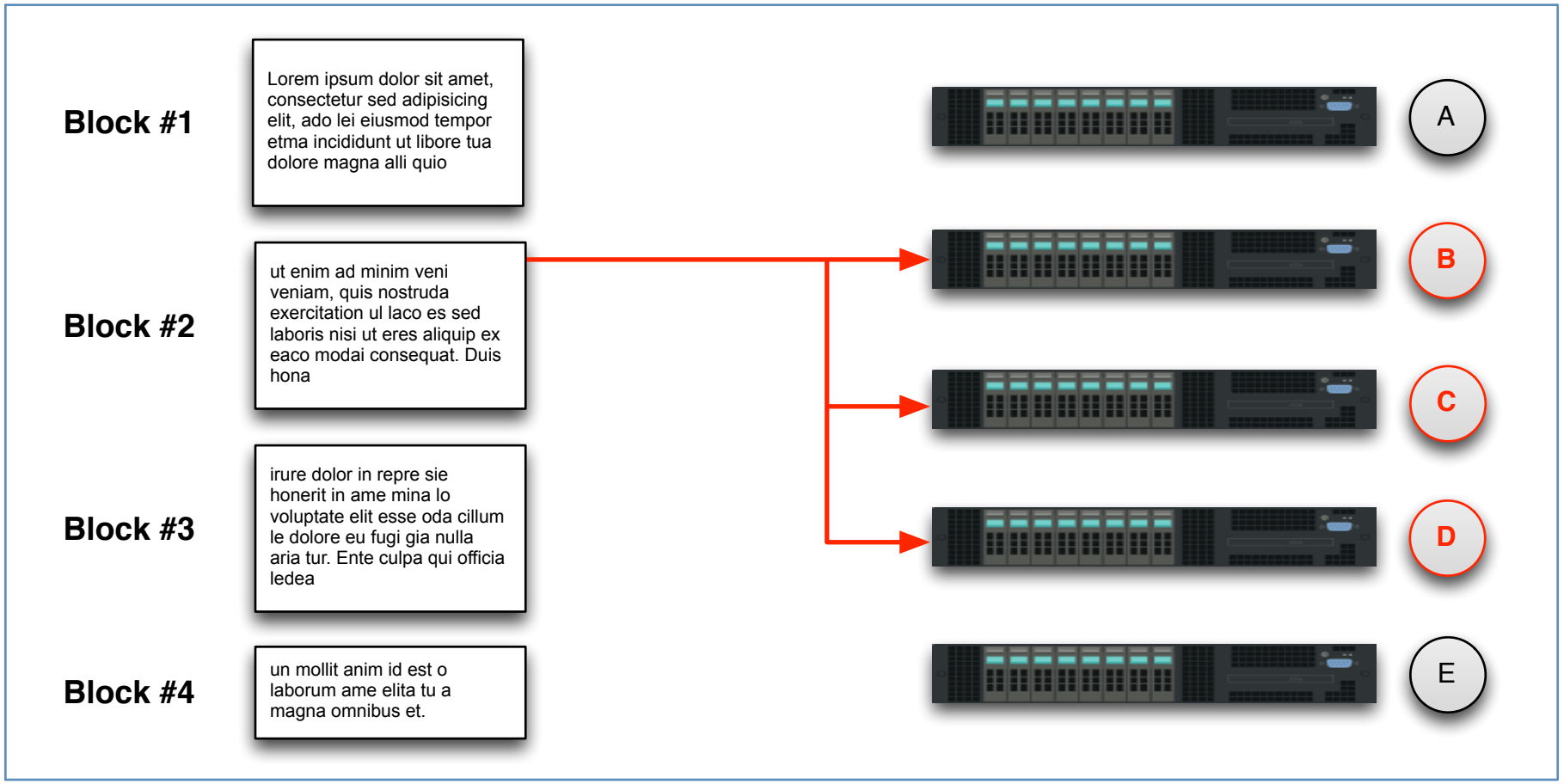
HDFS Replication

- Each block is then replicated across multiple nodes
  - Replication factor is also configurable, but defaults to three
- Benefits of replication
  - Availability: data isn't lost when a node fails
  - Reliability: HDFS compares replicas and fixes data corruption
  - Performance: allows for data locality
- Let's see an example of replication…

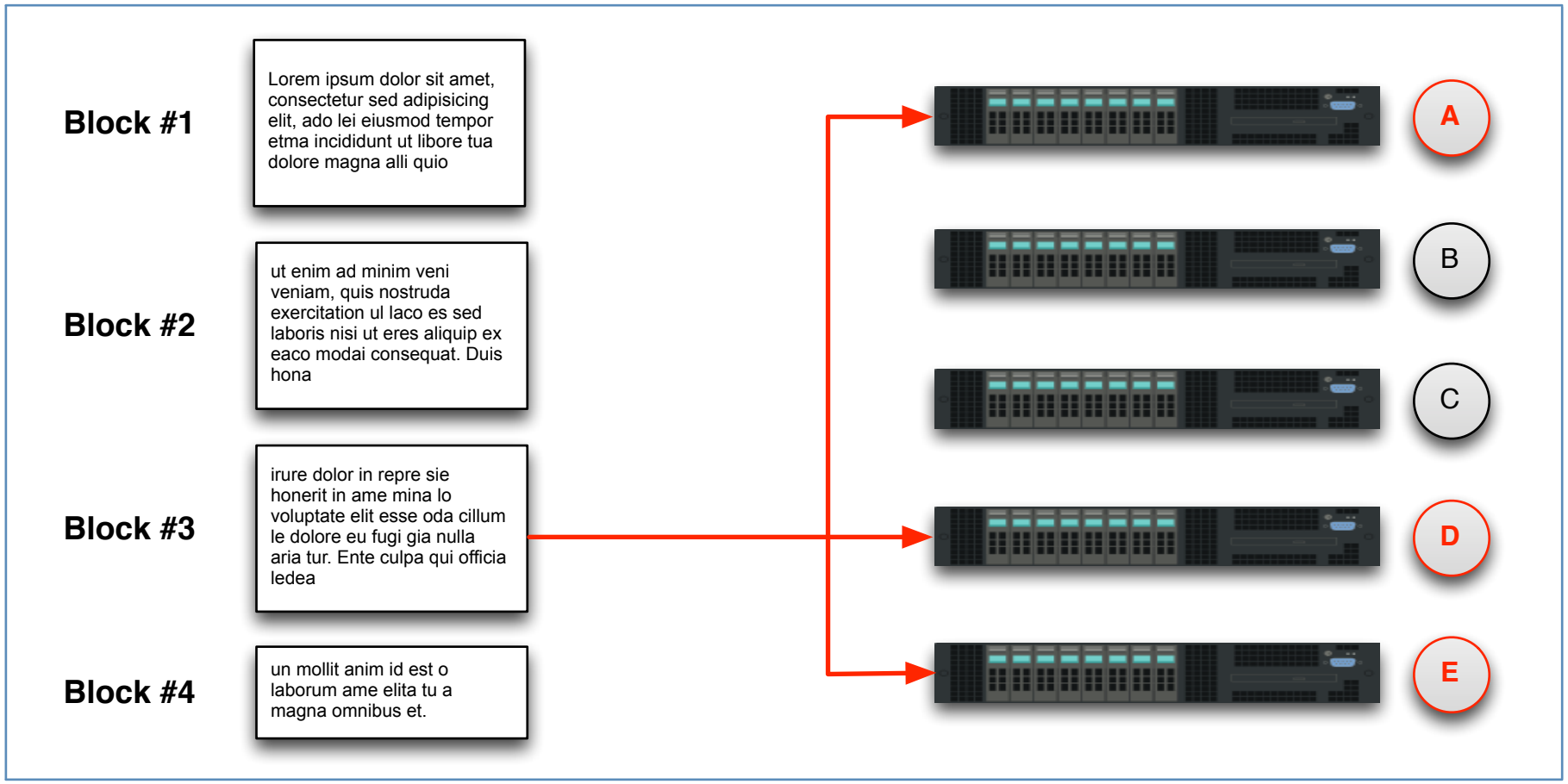
HDFS Replication (cont'd)



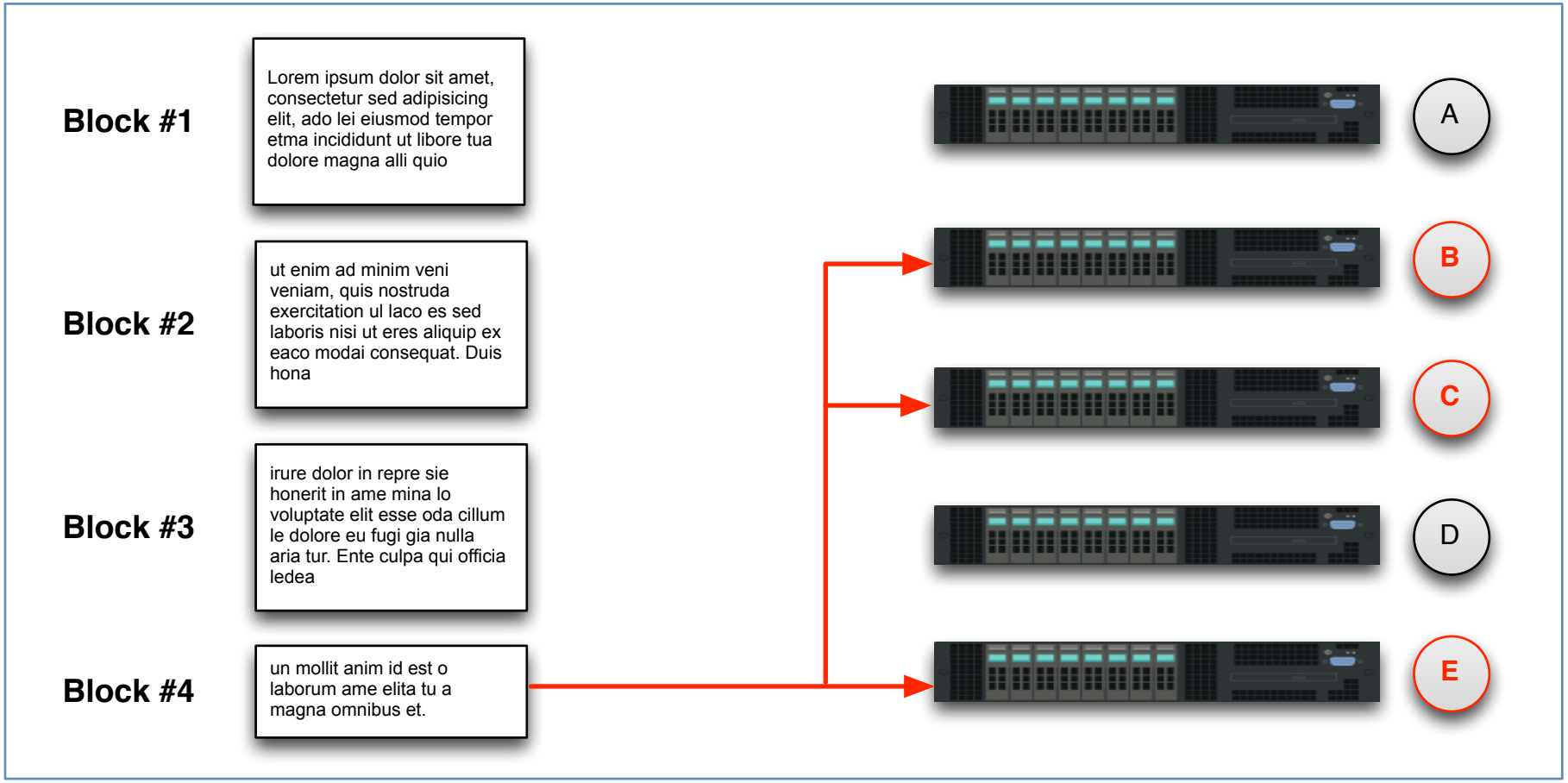
HDFS Replication (cont'd)



HDFS Replication (cont'd)



HDFS Replication (cont'd)



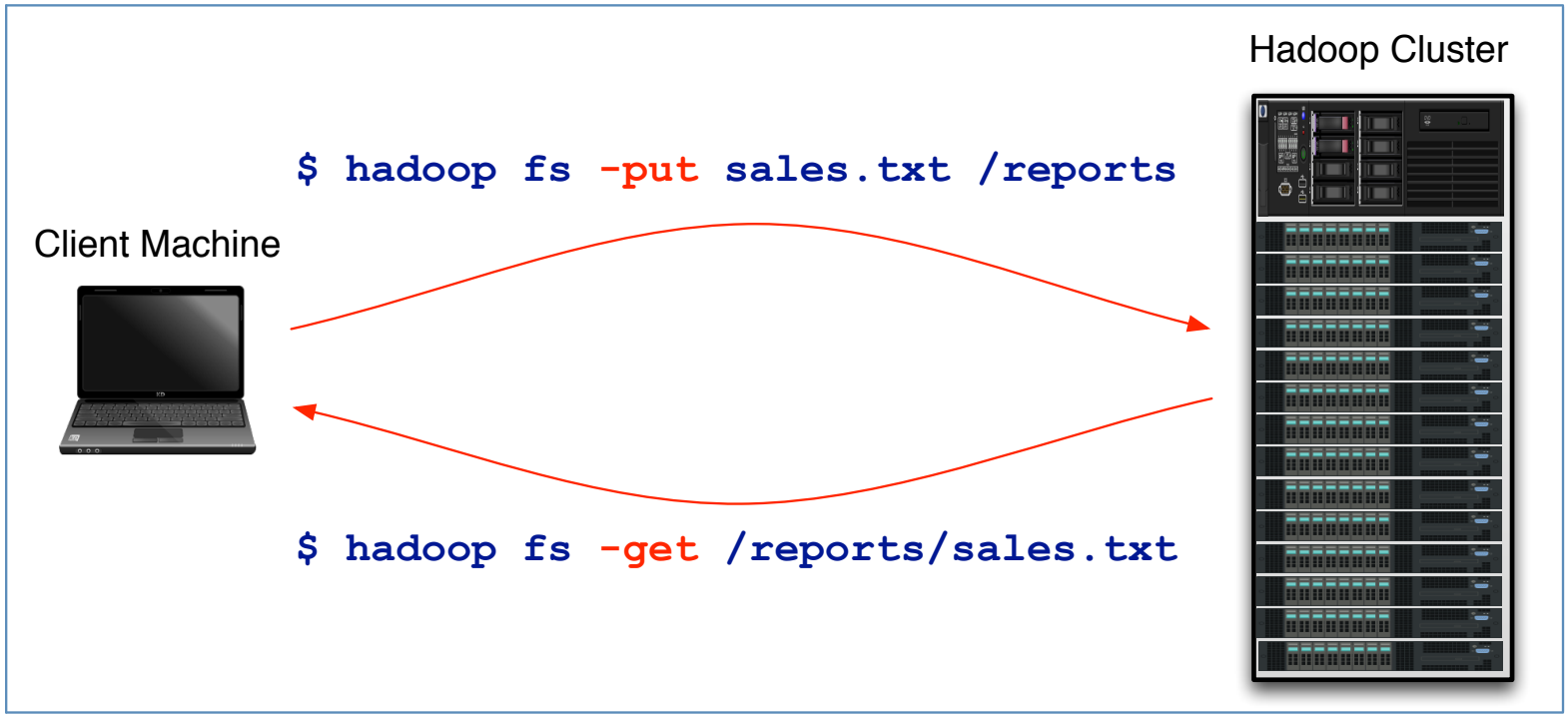
Accessing HDFS via the Command Line

- Users typically access HDFS via the `hadoop fs` command
  - Actions specified with subcommands (prefixed with a minus sign)
  - Most are similar to corresponding UNIX commands

```
$ hadoop fs -ls /user/tomwheeler

$ hadoop fs -cat /customers.csv

$ hadoop fs -rm /webdata/access.log

$ hadoop fs -mkdir /reports/marketing
```

Copying Local Data To and From HDFS

- Remember that HDFS is distinct from your local filesystem
  - `hadoop fs -put` copies local files ***to*** HDFS
  - `hadoop fs -get` fetches a local copy of a file ***from*** HDFS



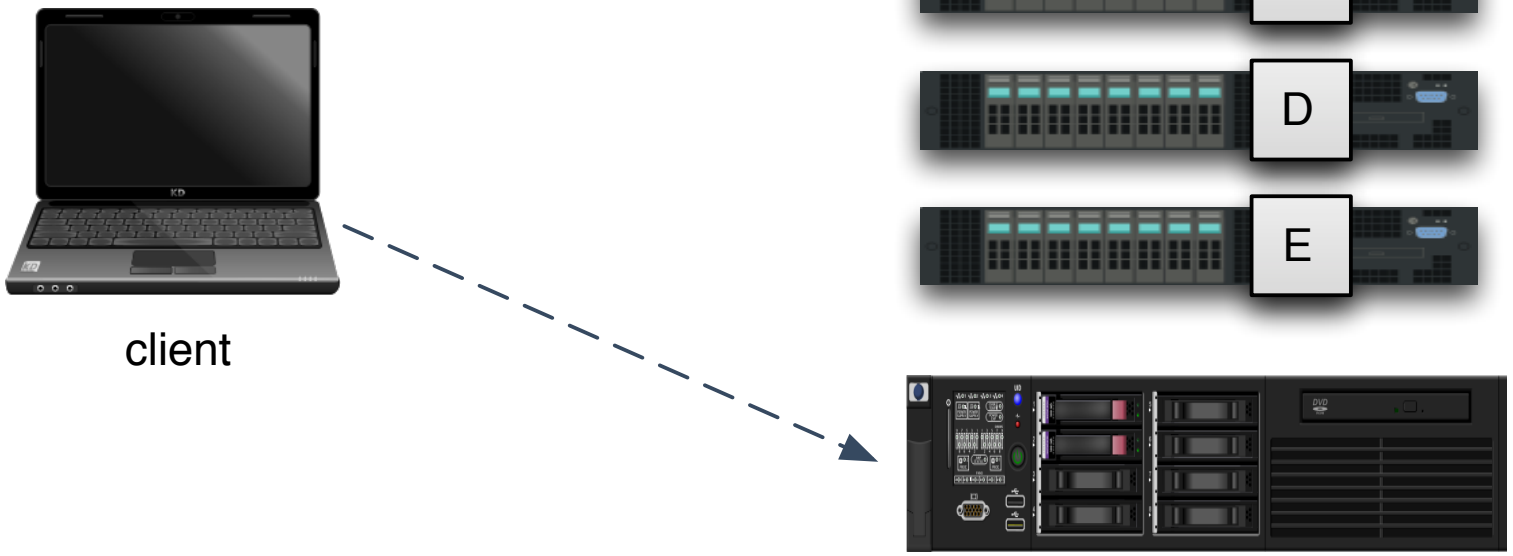
HDFS Daemons

- There are two daemon processes in HDFS
- NameNode (master)
  - Exactly one active NameNode per cluster
  - Manages namespace and metadata
- DataNode (slave)
  - Many per cluster
  - Performs block storage and retrieval

How File Reads Work

Step 1:

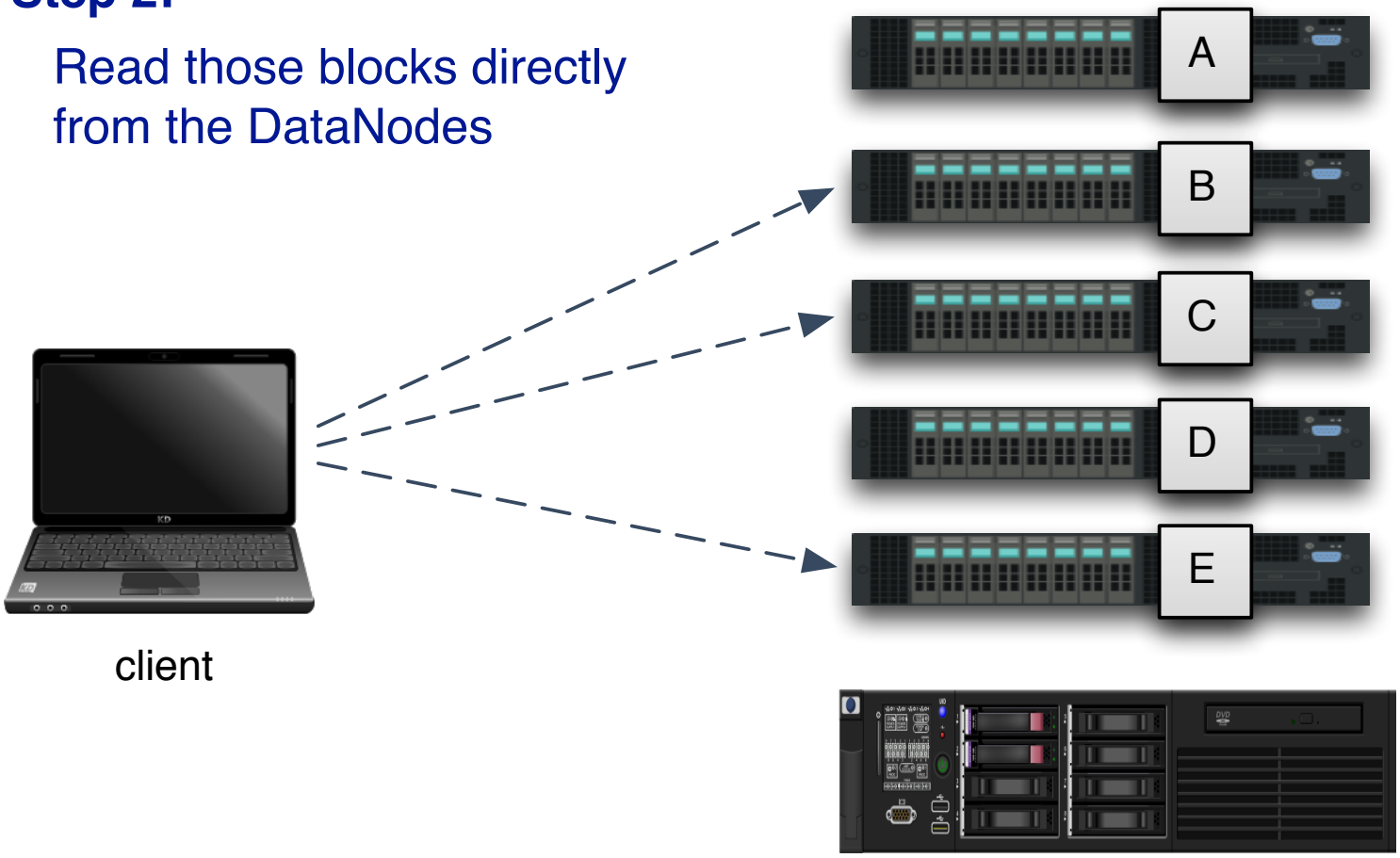
Get block locations from the NameNode



How File Reads Work (cont'd)

Step 2:

Read those blocks directly from the DataNodes



HDFS Demo

TODO: provide VM and instructions for demo

- I will now demonstrate the following
  1. How to list the contents of a directory
  2. How to create a directory in HDFS
  3. How to copy a local file to HDFS
  4. How to display the contents of a file in HDFS
  5. How to remove a file from HDFS

Data Processing with MapReduce

A Scalable Data Processing Framework

What is MapReduce?

- MapReduce is a programming model
  - It's a way of processing data
  - You can implement MapReduce in any language

Understanding Map and Reduce

- You supply two functions to process data: Map and Reduce
  - Map: typically used to transform, parse, or filter data
  - Reduce: typically used to summarize results
- The Map function always runs first
  - The Reduce function runs afterwards, but is optional
- Each piece is simple, but can be powerful when combined

MapReduce Benefits

- Scalability
  - Hadoop divides the processing job into individual tasks
  - Tasks execute in parallel (independently) across cluster
- Simplicity
  - Processes one record at a time
- Ease of use
  - Hadoop provides job scheduling and other infrastructure
  - Far simpler for developers than typical distributed computing

MapReduce in Hadoop

- MapReduce processing in Hadoop is *batch-oriented*
- A MapReduce *job* is broken down into smaller *tasks*
  - Tasks run concurrently
  - Each processes a small amount of overall input
- MapReduce code for Hadoop is usually written in Java
  - This uses Hadoop's API directly
- You can do basic MapReduce in other languages
  - Using the *Hadoop Streaming* wrapper program
  - Some advanced features require Java code

MapReduce Example in Python

- The following example uses Python
  - Via Hadoop Streaming
- It processes log files and summarizes events by type
  - I'll explain both the data flow and the code

Job Input

- Here's the job input

```
2013-06-29 22:16:49.391 CDT INFO "This can wait"
2013-06-29 22:16:52.143 CDT INFO "Blah blah blah"
2013-06-29 22:16:54.276 CDT WARN "This seems bad"
2013-06-29 22:16:57.471 CDT INFO "More blather"
2013-06-29 22:17:01.290 CDT WARN "Not looking good"
2013-06-29 22:17:03.812 CDT INFO "Fairly unimportant"
2013-06-29 22:17:05.362 CDT ERROR "Out of memory!"
```

- Each map task gets a chunk of this data to process
  - Typically corresponds to a single block in HDFS

Python Code for Map Function

```
1  #!/usr/bin/env python
2
3  import sys
4
5  levels = ['TRACE', 'DEBUG', 'INFO',
6            'WARN', 'ERROR', 'FATAL']
7
8  for line in sys.stdin:
9     fields = line.split()
10
11    level = fields[3].upper()
12
13    if level in levels:
14       print "%s\t1" % level
```

Define list of known log levels

Read records from standard input. Use whitespace to split into fields.

Extract "level" field and convert to uppercase for consistency.

If it matches a known level, print it, a tab separator, and the literal value 1 (since the level can only occur once per line)

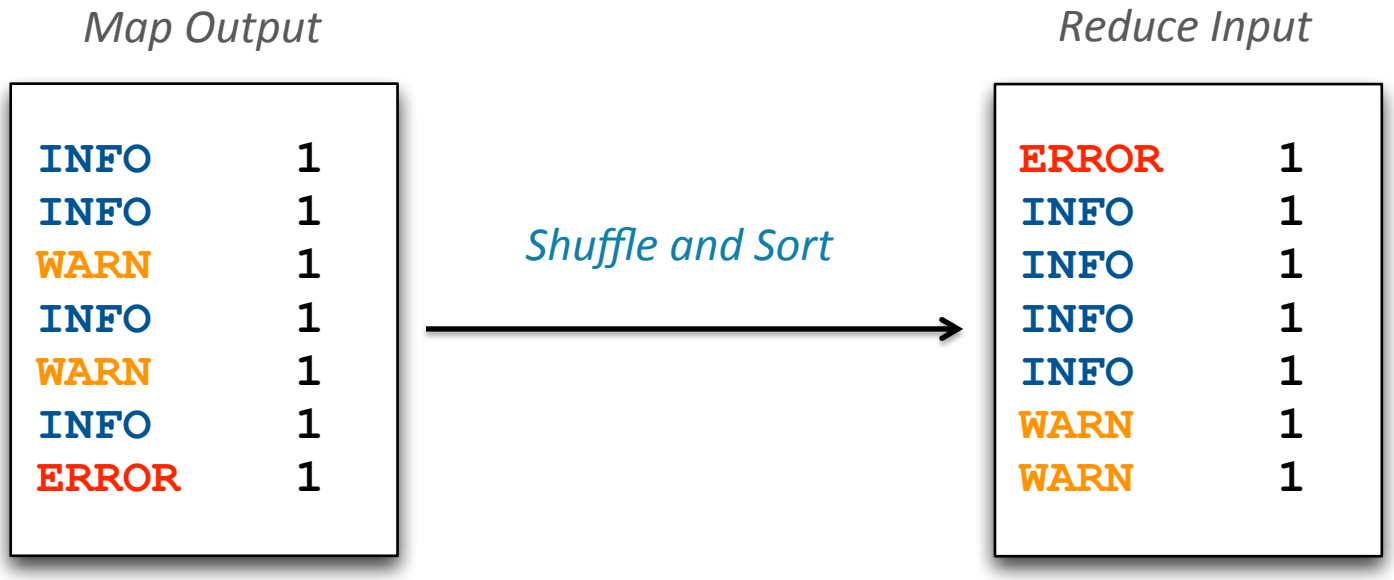
Output of Map Function

- The map function produces key/value pairs as output

```
INFO    1
INFO    1
WARN    1
INFO    1
WARN    1
INFO    1
ERROR   1
```

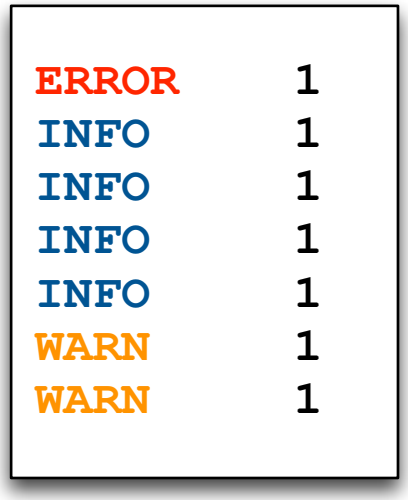
The "Shuffle and Sort"

- Hadoop *automatically* merges, sorts, and groups map output
  - The result is passed as input to the reduce function
  - More on this later…



Input to Reduce Function

- Reduce function receives a key and all values for that key



- Keys are always passed to reducers in sorted order
  - Although not obvious here, values are unordered

Python Code for Reduce Function

```
1  #!/usr/bin/env python
2
3  import sys
4
5  previous_key = None
6  sum = 0
7
8  for line in sys.stdin:
9      key, value = line.split()
10
11     if key == previous_key:
12         sum = sum + int(value)
13         # continued on next slide
```

Initialize loop variables

Extract the key and value passed via standard input

If key unchanged, increment the count

Python Code for Reduce Function

```
14      # continued from previous slide
15      else:
16          if previous_key:
17              print '%s\t%i' % (previous_key, sum)
18
19          previous_key = key
20          sum = 1
21
22  print '%s\t%i' % (previous_key, sum)
```

If key changed, print data for old level

Start tracking data for the new record

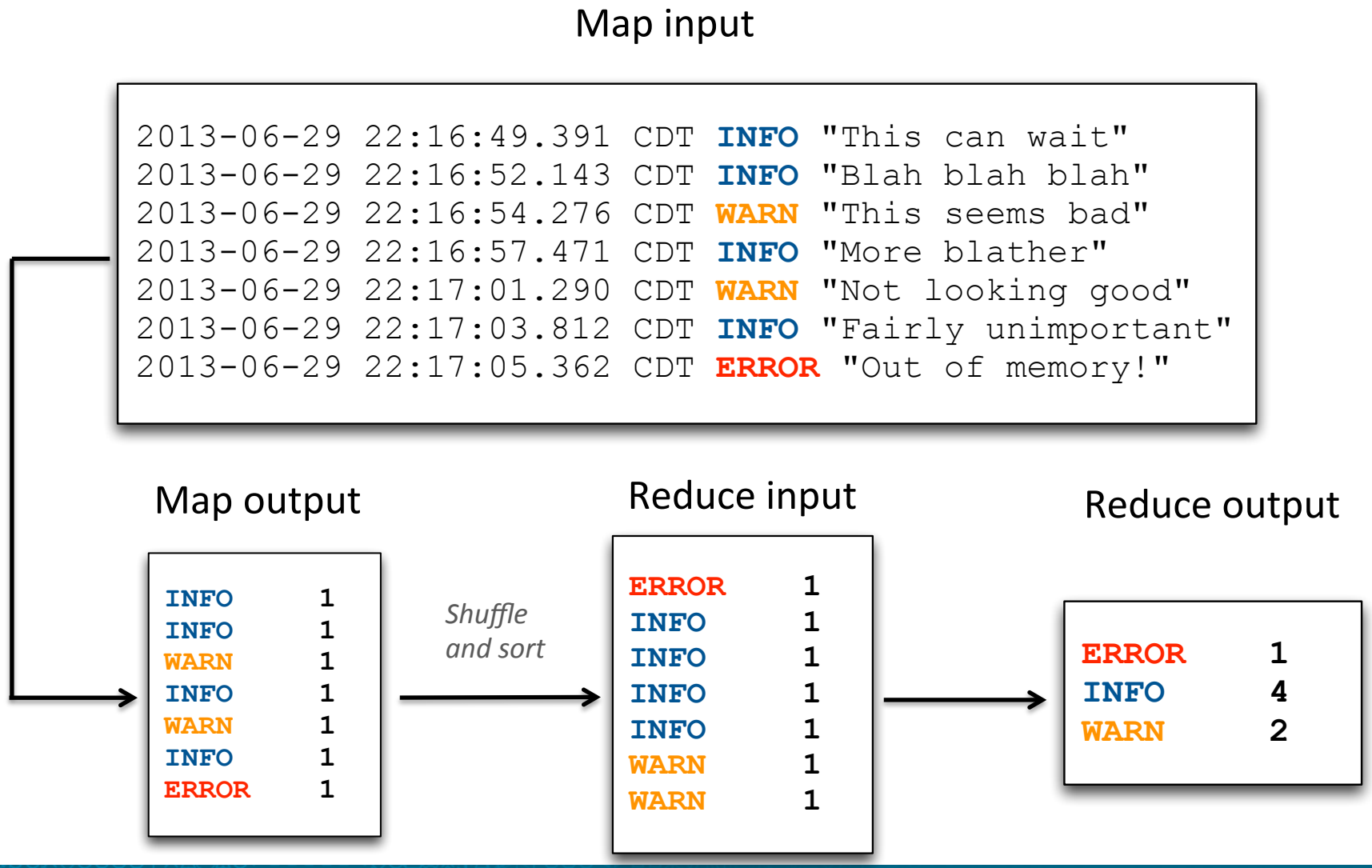
Print data for the final key

Output of Reduce Function

- Its output is a sum for each level

```
ERROR    1
INFO     4
WARN     2
```

Recap of Data Flow



How to Run a Hadoop Streaming Job

- I'll demonstrate this now…

TODO: provide VM and instructions for demo

MapReduce Daemons

- There are two daemon processes in MapReduce
- JobTracker (master)
  - Exactly one active JobTracker per cluster
  - Accepts jobs from client
  - Schedules and monitors tasks on slave nodes
  - Reassigns tasks in case of failure
- TaskTracker (slave)
  - Many per cluster
  - Performs the shuffle and sort
  - Executes map and reduce tasks

The Hadoop Ecosystem

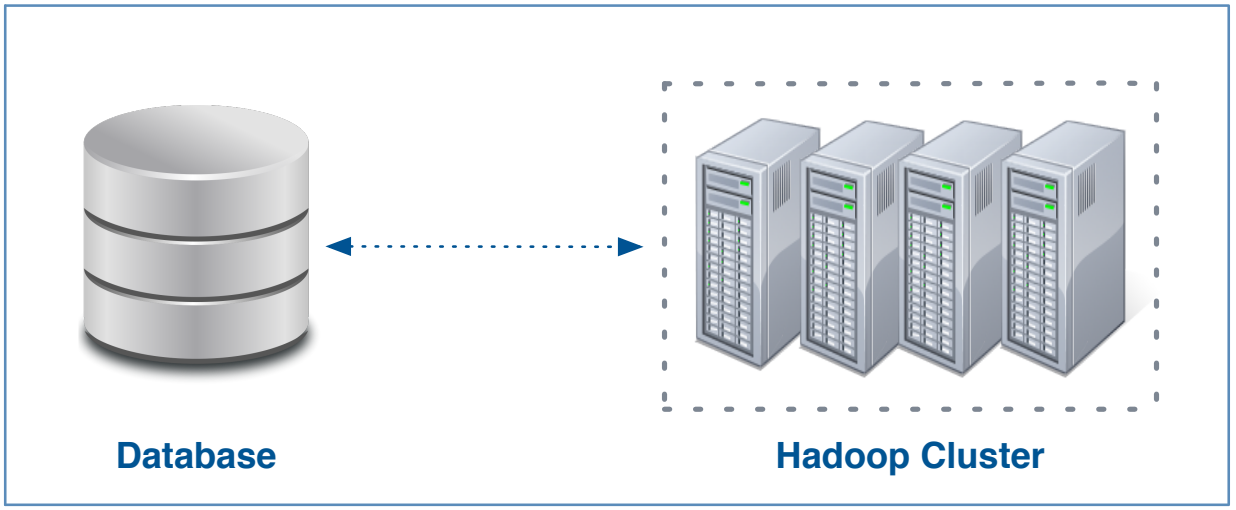
Open Source Tools that Complement Hadoop

The Hadoop Ecosystem

- "Core Hadoop" consists of HDFS and MapReduce
  - These are the kernel of a much broader platform
- Hadoop has many related projects
  - Some help you integrate Hadoop with other systems
  - Others help you analyze your data
- These are not considered “core Hadoop”
  - Rather, they’re part of the *Hadoop ecosystem*
  - Many are also open source Apache projects

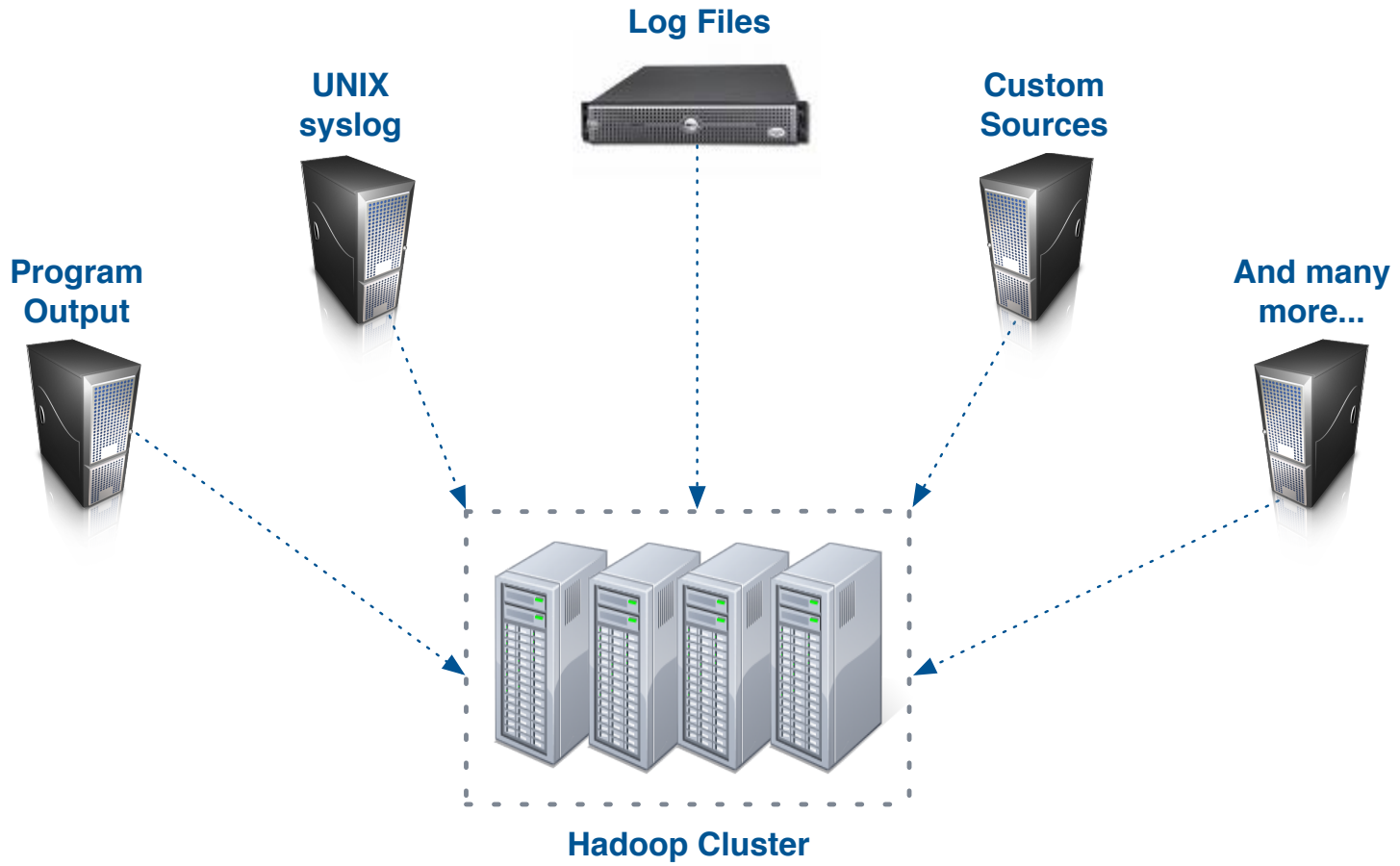
Apache Sqoop

- Sqoop exchanges data between RDBMS and Hadoop
- Can import entire DB, a single table, or a table subset into HDFS
  - Does this very efficiently via a Map-only MapReduce job
  - Can also export data from HDFS back to the database



Apache Flume

- **Flume imports data into HDFS *as it is being generated* by various sources**



Apache Pig

- Pig offers high-level data processing on Hadoop
  - An alternative to writing low-level MapReduce code

```
people = LOAD '/data/customers' AS (cust_id, name);
orders = LOAD '/data/orders' AS (ord_id, cust_id, cost);
groups = GROUP orders BY cust_id;
totals = FOREACH groups GENERATE group, SUM(orders.cost) AS t;
result = JOIN totals BY group, people BY cust_id;
DUMP result;
```

- Pig turns this into MapReduce jobs that run on Hadoop

Apache Hive

- Hive is another abstraction on top of MapReduce
  - Like Pig, it also reduces development time
  - Hive uses a SQL-like language called HiveQL


```
  SELECT customers.cust_id, SUM(cost) AS total
    FROM customers
    JOIN orders
      ON customers.cust_id = orders.cust_id
GROUP BY customers.cust_id
ORDER BY total DESC;
```

Apache Mahout

- Mahout is a scalable machine learning library
- Support for several categories of problems
  - Classification
  - Clustering
  - Collaborative filtering
  - Frequent itemset mining
- Many algorithms implemented in MapReduce
  - Can parallelize inexpensively with Hadoop

Apache HBase

- HBase is a NoSQL database built on top of Hadoop
- Can store massive amounts of data
  - Gigabytes, terabytes, and even petabytes of data in a table
  - Tables can have many thousands of columns
- Scales to provide very high write throughput
  - Hundreds of thousands of inserts per second

Cloudera Impala

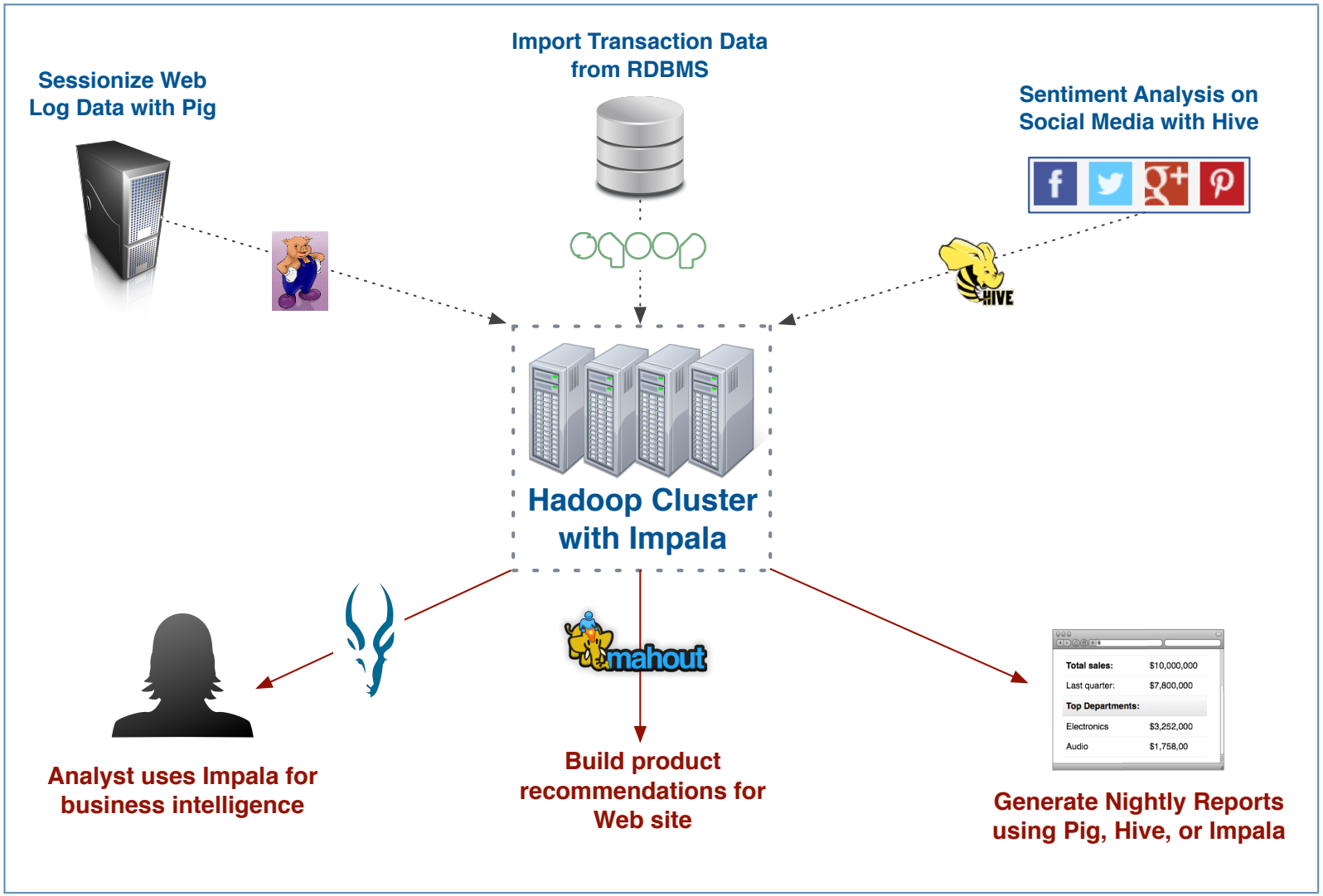
- Massively parallel SQL engine which runs on a Hadoop cluster
  - Inspired by Google's Dremel project
  - Can query data stored in HDFS or HBase tables
- High performance
  - Typically > 10 times faster than Pig or Hive
  - Query syntax virtually identical to Hive / SQL
- Impala is 100% open source (Apache-licensed)

Querying Data with Hive and Impala

- I'll demonstrate this now…

TODO: provide VM and instructions for demo

Visual Overview of a Complete Workflow



Hadoop Distributions

- Conceptually similar to a Linux distribution
- These distributions include a stable version of Hadoop
  - Plus ecosystem tools like Flume, Sqoop, Pig, Hive, Impala, etc.
- Benefits of using a distribution
  - Integration testing helps ensure all tools work together
  - Easy installation and updates
  - Compatibility certification from hardware vendors
  - Commercial support
- Apache Bigtop – upstream distribution for many commercial distributions

Cloudera's Distribution (CDH)

- Cloudera's Distribution including Apache Hadoop (CDH)
  - The most widely used distribution of Hadoop
  - Stable, proven and supported environment
- Combines Hadoop with many important ecosystem tools
  - Including all those I've mentioned
- How much does it cost?
  - It's completely free
  - Apache licensed – it's 100% open source too

Hadoop Clusters

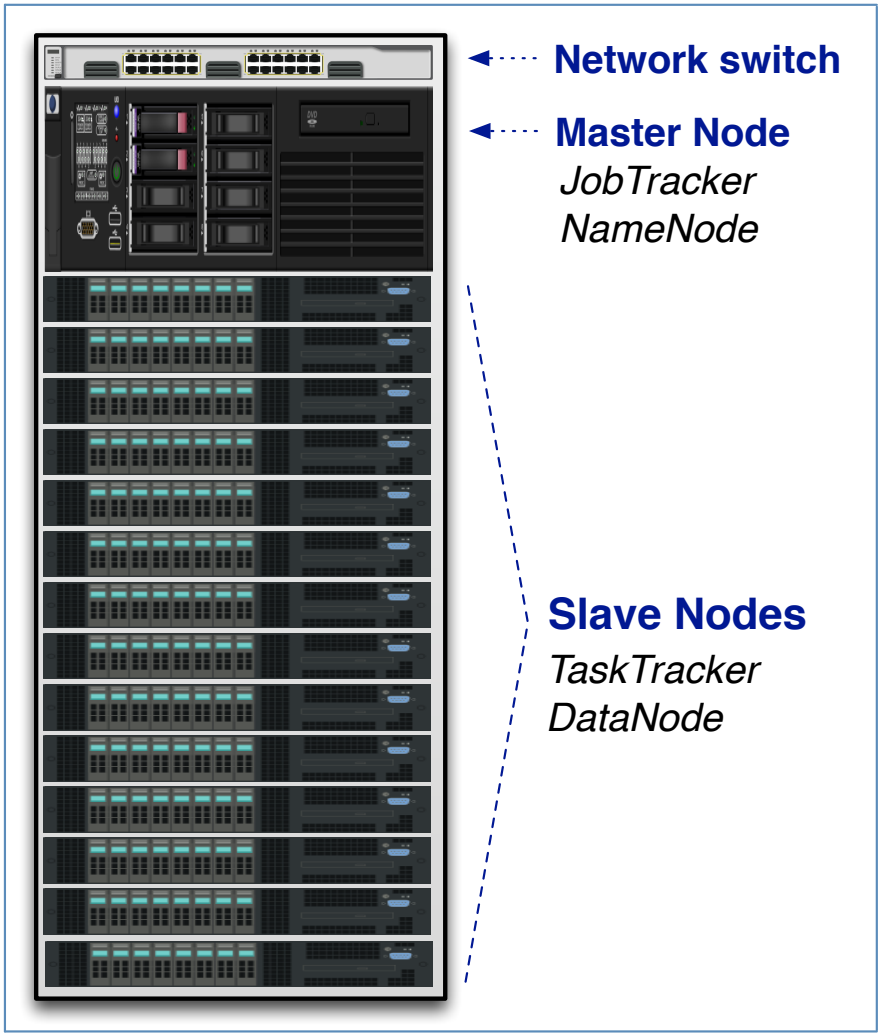
Past, Present, and Future

Hadoop Cluster Overview

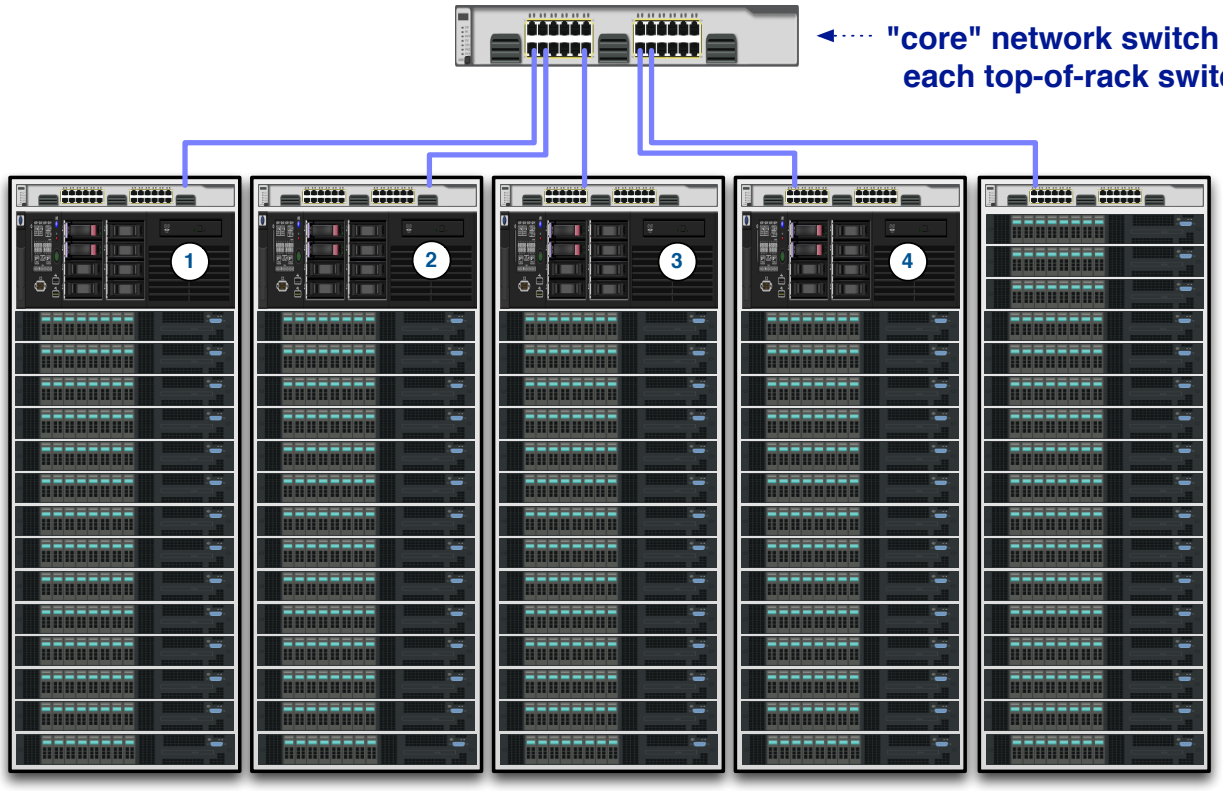
- A cluster is made up of nodes
  - A node is simply a (typically rackmount) server
  - There may be a few – or a few thousand – nodes
  - Most are *slave* nodes, but a few are *master* nodes
  - Every node is responsible for both storage and processing
- Nodes are connected together by network switches
- Slave nodes do not use RAID
  - Block splitting and replication is built into HDFS
- Nearly all production clusters run Linux

Anatomy of a Small Hadoop Cluster

- Typically consists of industry-standard rackmounted servers
- JobTracker and NameNode might run on same server
- TaskTracker and DataNode are always co-located on each slave node for data locality



Anatomy of a Large Cluster



1. Master (active NameNode)
2. Master (standby NameNode)
3. Master (active JobTracker)
4. Master (standby JobTracker)

Build, Buy, or Use the Cloud?

- There are several ways to run Hadoop at scale
  - Build your own cluster
  - Buy a pre-configured cluster from hardware vendor
  - Run Hadoop in the cloud
    - Private cloud: virtualized hardware in your data center
    - Public cloud: on a service like Amazon EC2
- Let's cover the pros, cons, and concerns…

Build versus Buy

- Pros for building your own
  - Select whichever components you like
  - Can reuse components you already have
  - Avoid reliance on a single vendor
- Pros for buying pre-configured cluster
  - Vendor tests all the components together (certification)
  - Avoids “blame the other vendor” during support calls
  - May actually be less expensive due to economies of scale

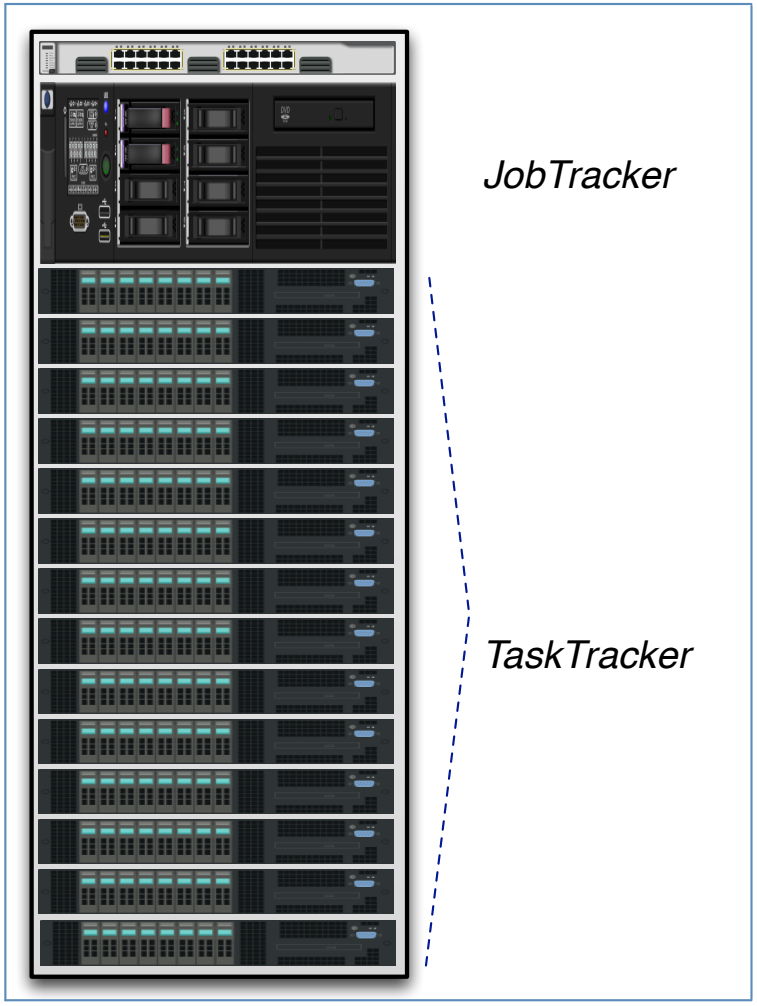
Host in the Cloud?

- Good choice when need for cluster is sporadic
  - And amount of data stored / transferred is relatively low

	Your Own Cluster	Hosted in the Cloud
Hardware Cost		X
Staffing Cost		X
Power / HVAC		X
Storage Cost	X	
Bandwidth Cost	X	
Performance	X	

Hadoop's Current Data Processing Architecture

- Hadoop's facilities for data processing are based on the MapReduce framework
- Works well, but there are two important limitations
  - Only one active JobTracker per cluster (scalability)
  - MapReduce is not an ideal fit for all processing needs (flexibility)



YARN: Next Generation Processing Architecture

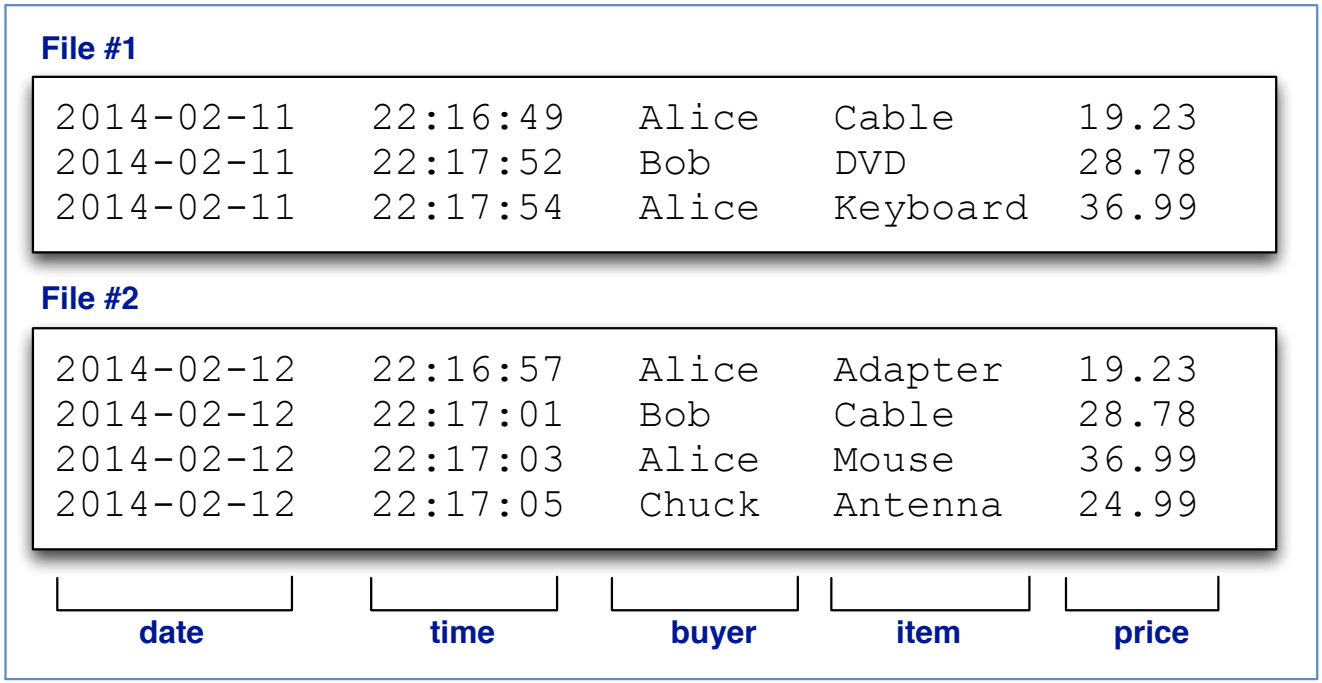
- YARN generalizes scheduling and resource allocation
- Improves scalability
  - Names and roles of daemons have changed
  - Many responsibilities previously associated with JobTracker are now delegated to slave nodes
- Improves flexibility
  - Allows processing frameworks other than MapReduce
  - Example: Apache Giraph (graph processing framework)
- Maintains backwards compatibility
  - MapReduce is also supported by YARN
  - Requires no change to existing MapReduce jobs

Hadoop File Formats

- Hadoop supports a number of input/output formats, including
  - Free-form text
  - Delimited text
  - Several specialized formats for efficient storage
    - Sequence files
    - Avro
    - RCFile
    - Parquet
  - Also possible to add support for custom formats

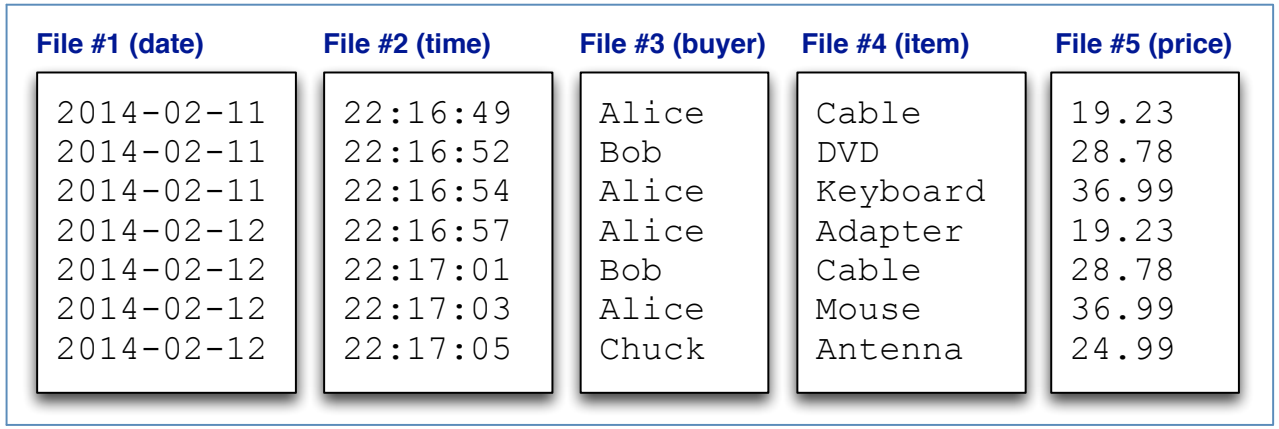
Typical Dataset Example

- Most of these formats store data as rows of fields
  - Each row contains all fields for a single record
  - Additional files contain additional records in the same format



The Parquet File Format

- Parquet is a new high-performance file format
  - Originally developed by engineers from Cloudera and Twitter
  - Open source, with an active developer community
- Can store each column in its own file
  - Allows for much better compression due to similar values
  - Reduces I/O when only a subset of columns are needed

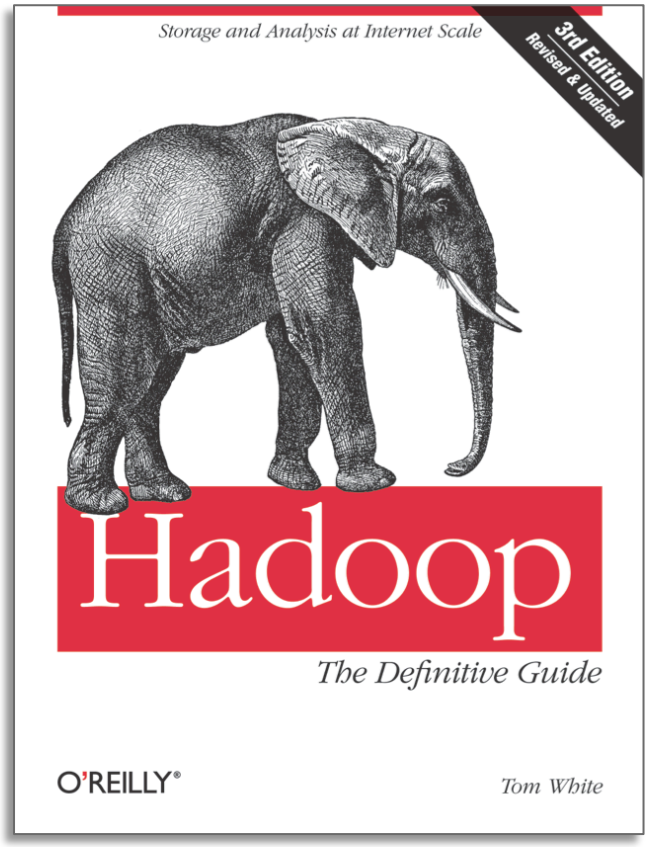


Conclusion

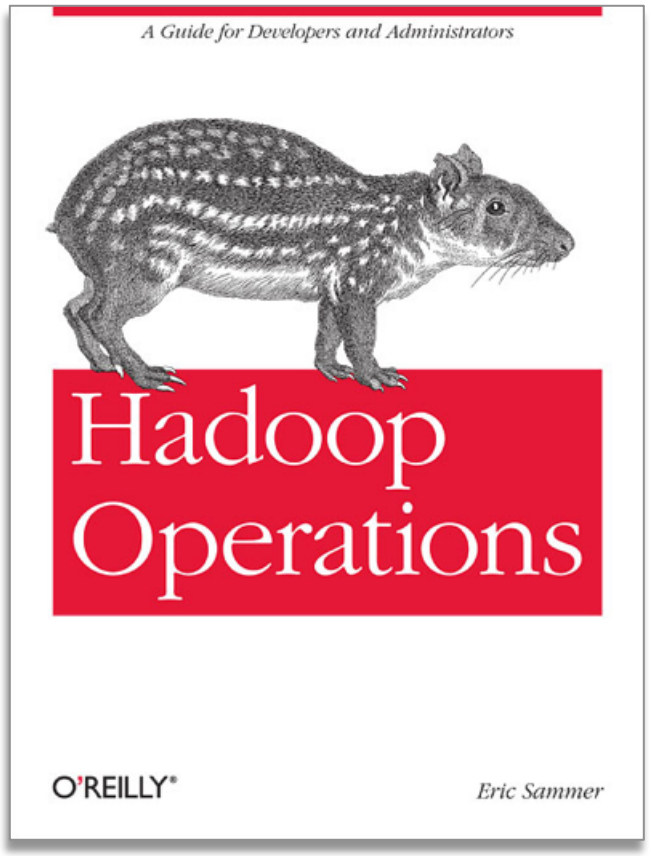
Key Points

- We're generating massive volumes of data
  - This data can be extremely valuable
  - Companies can now analyze what they previously discarded
- Hadoop supports large-scale data storage and processing
  - Heavily influenced by Google's architecture
  - Already in production by thousands of organizations
  - HDFS is Hadoop's storage layer
  - MapReduce is Hadoop's processing framework
- Many ecosystem projects complement Hadoop
  - Some help you to integrate Hadoop with existing systems
  - Others help you analyze the data you've stored

Highly Recommended Books



Author: Tom White
ISBN: 1-449-31152-0

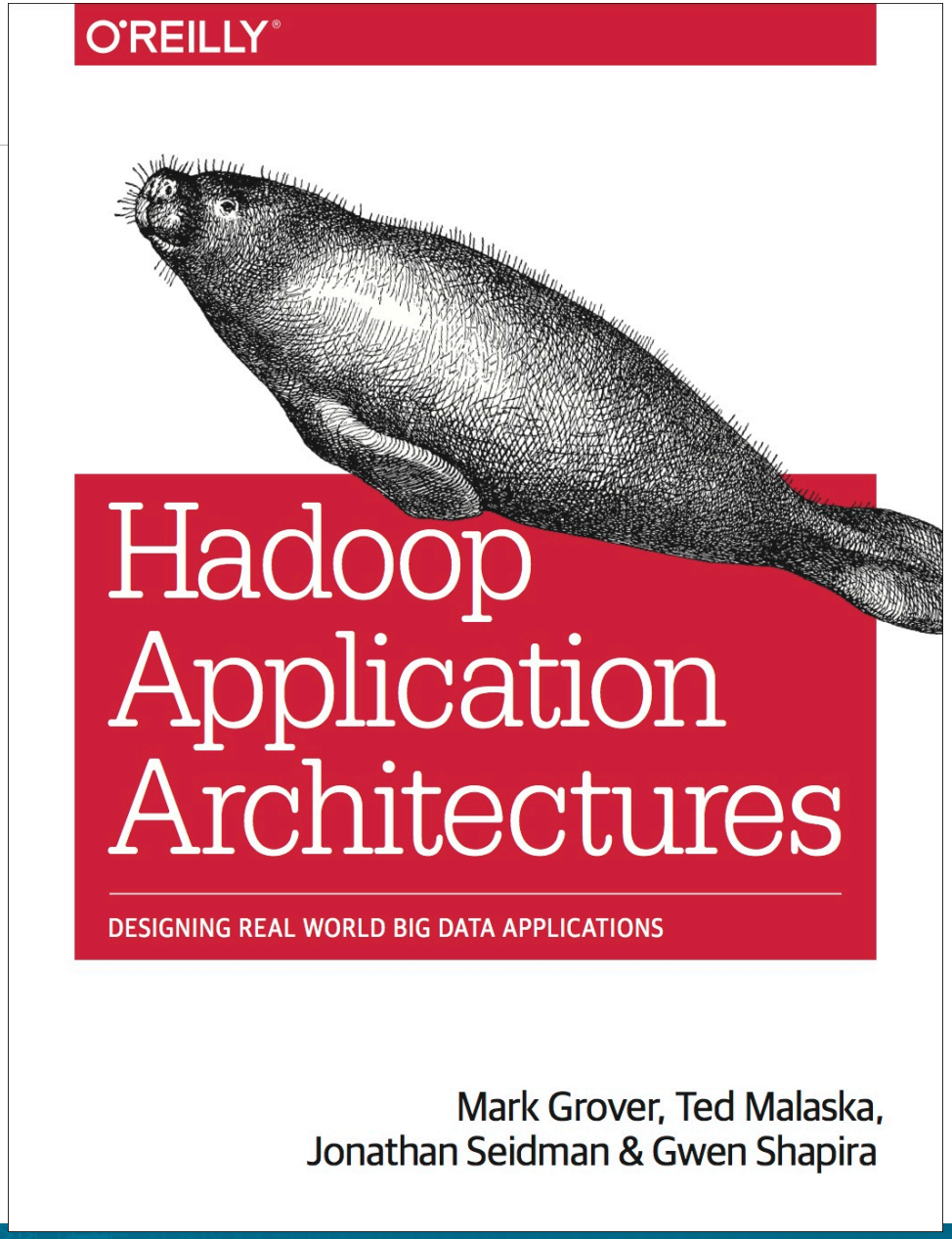


Author: Eric Sammer
ISBN: 1-449-32705-2

My book

@hadooparchbook

hadooparchitecturebook.com



About Cloudera

- Helps companies profit from their data
  - Founded by experts from Facebook, Google, Oracle, and Yahoo
- We offer products and services for large-scale data analysis
  - Software (CDH distribution and Cloudera Manager)
  - Consulting and support services
  - Training and certification
- Active developers of open source “Big Data” software
  - Staff includes committers to every single project I’ll cover today

Questions?

- Thank you for attending!
- I'll be happy to answer any additional questions now…
- Want to learn even more?
  - Cloudera training: developers, analysts, sysadmins, and more
  - Offered in more than 50 cities worldwide, and online too!
  - See http://university.cloudera.com/ for more info
- Demo and slides at github.com/markgrover/hadoop-intro-fast
- Twitter: mark_grover
- Survey page: tiny.cloudera.com/mark