



Introduction to Apache Hadoop





Chapter 1

Introduction

Introduction



Course Logistics

About Apache Hadoop

About Cloudera

Conclusion

Logistics

- Course start and end time
- Breaks
- Restrooms

About Your Instructor

- A brief introduction to your instructor

Introduction

Course Logistics

 **About Apache Hadoop**

About Cloudera

Conclusion

Facts about Apache Hadoop

- Open source (using the Apache license)
- Around 40 core Hadoop committers from ~10 companies
 - Cloudera, Yahoo!, Facebook, Apple, and more
- Hundreds of contributors writing features, fixing bugs
- Many related projects, applications, tools, etc.

A large ecosystem



Who uses Hadoop?



The New York Times



eHarmony®



Autodesk



amazon.com



JPMORGAN CHASE & Co.



Vendor integration



Introduction

Course Logistics

About Apache Hadoop



About Cloudera

Conclusion

About Cloudera

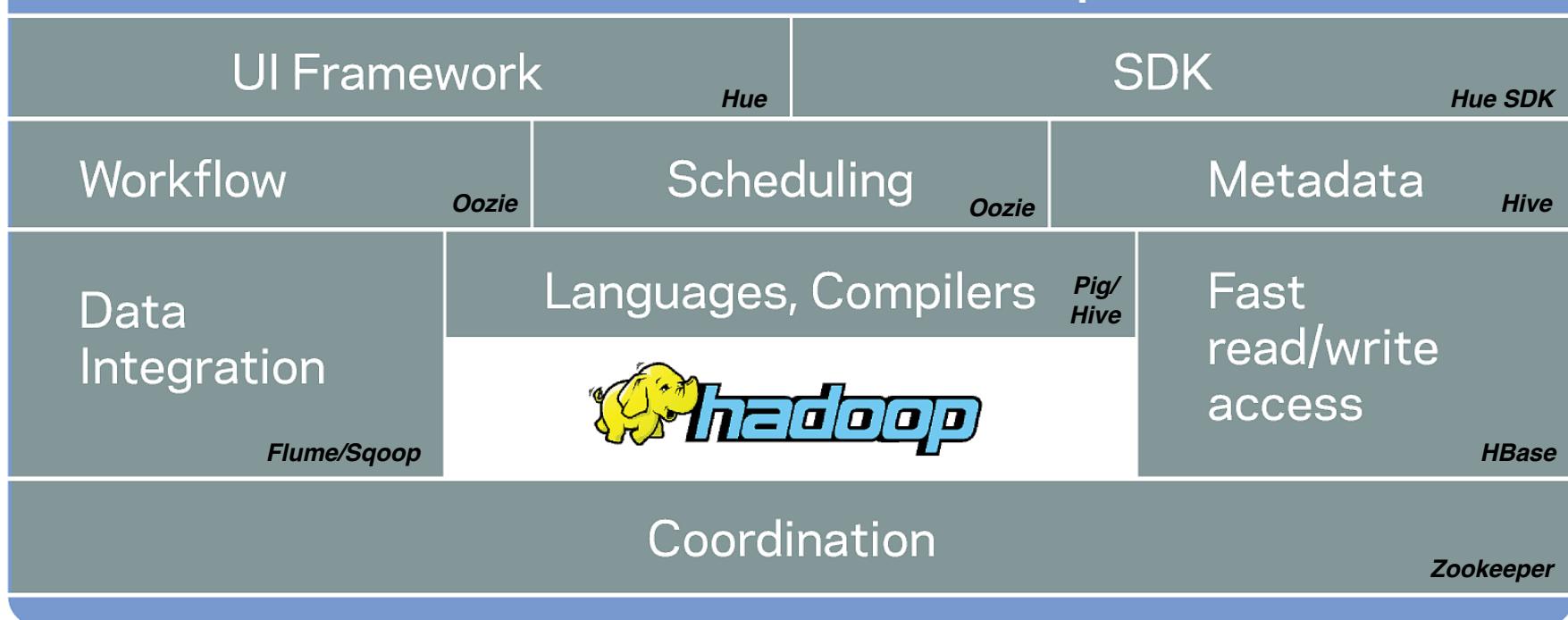
- **Cloudera is “The commercial Hadoop company”**
- **Founded by leading experts on Hadoop from Facebook, Google, Oracle and Yahoo**
- **Provides consulting and training services for Hadoop users**
- **Staff includes several committers to Hadoop projects**

Cloudera Software (All Open-Source)

- **Cloudera's Distribution including Apache Hadoop (CDH)**
 - A single, easy-to-install package from the Apache Hadoop core repository
 - Includes a stable version of Hadoop, plus critical bug fixes and solid new features from the development version
- **Components**
 - Apache Hadoop
 - Apache Hive
 - Apache Pig
 - Apache HBase
 - Apache Zookeeper
 - Flume, Hue, Oozie, and Sqoop

A Coherent Platform

Cloudera's Distribution for Hadoop



Cloudera Manager

- **Cloudera Manager, Free Edition**

- The fastest, easiest way to install, configure and manage your Hadoop cluster
- Installs CDH and management agents on each machine
- Configuration is performed from a central location
 - No need to edit configuration files on each individual machine in the cluster
- Supports clusters of up to 50 nodes

- **Cloudera Manager, full version**

- Supports unlimited nodes in the cluster
- Includes powerful, best-of-breed cluster monitoring tools
- Provided as part of Cloudera Enterprise

Cloudera Enterprise

- **Cloudera Enterprise**

- Complete package of software and support
- Built on top of CDH
- Includes the full edition of Cloudera Manager
 - Central cluster configuration
 - Powerful cluster monitoring
 - Alerting
 - Resource consumption tracking
 - LDAP integration
 - Much more
- Phone and e-mail support for the entire Hadoop stack
 - Including priority bugfixes

Introduction

Course Logistics

About Apache Hadoop

About Cloudera



Conclusion

Conclusion

- Apache Hadoop is a fast-growing data framework
- Cloudera's Distribution including Apache Hadoop offers a free, cohesive platform that encapsulates:
 - Data integration
 - Data processing
 - Workflow scheduling
 - Monitoring



Chapter 2

The Motivation For Hadoop

The Motivation For Hadoop

In this chapter you will learn:

- **What problems exist with ‘traditional’ large-scale computing systems**
- **What requirements an alternative approach should have**
- **How Hadoop addresses those requirements**

The Motivation For Hadoop



Problems with Traditional Large-Scale Systems

Requirements for a New Approach

Hadoop!

Conclusion

Traditional Large-Scale Computation

- Traditionally, computation has been processor-bound
 - Relatively small amounts of data
 - Significant amount of complex processing performed on that data
- For decades, the primary push was to increase the computing power of a single machine
 - Faster processor, more RAM



Distributed Systems

- **Moore's Law:** roughly stated, processing power doubles every two years
- Even that hasn't always proved adequate for very CPU-intensive jobs
- **Distributed systems evolved to allow developers to use multiple machines for a single job**
 - MPI
 - PVM
 - Condor

MPI: Message Passing Interface

PVM: Parallel Virtual Machine

Distributed Systems: Problems

- **Programming for traditional distributed systems is complex**
 - Data exchange requires synchronization
 - Finite bandwidth is available
 - Temporal dependencies are complicated
 - It is difficult to deal with partial failures of the system
- **Ken Arnold, CORBA designer:**
 - “Failure is the defining difference between distributed and local programming, so you have to design distributed systems with the expectation of failure”
 - Developers spend more time designing for failure than they do actually working on the problem itself

CORBA: Common Object Request Broker Architecture

Distributed Systems: Data Storage

- Typically, data for a distributed system is stored on a SAN
- At compute time, data is copied to the compute nodes
- Fine for relatively limited amounts of data

SAN: Storage Area Network

The Data-Driven World

- **Modern systems have to deal with far more data than was the case in the past**
 - Organizations are generating huge amounts of data
 - That data has inherent value, and cannot be discarded
- **Examples:**
 - Yahoo – over 170PB of data
 - Facebook – over 30PB of data
 - eBay – over 5PB of data
- **Many organizations are generating data at a rate of terabytes per day**

Data Becomes the Bottleneck

- **Getting the data to the processors becomes the bottleneck**
- **Quick calculation**
 - Typical disk data transfer rate: 75MB/sec
 - Time taken to transfer 100GB of data to the processor: approx 22 minutes!
 - Assuming sustained reads
 - Actual time will be worse, since most servers have less than 100GB of RAM available
- **A new approach is needed**

The Motivation For Hadoop

Problems with Traditional Large-Scale Systems



Requirements for a New Approach

Hadoop!

Conclusion

Partial Failure Support

- **The system must support partial failure**
 - Failure of a component should result in a graceful degradation of application performance
 - Not complete failure of the entire system

Data Recoverability

- **If a component of the system fails, its workload should be assumed by still-functioning units in the system**
 - Failure should not result in the loss of any data

Component Recovery

- **If a component of the system fails and then recovers, it should be able to rejoin the system**
 - Without requiring a full restart of the entire system

Consistency

- Component failures during execution of a job should not affect the outcome of the job

Scalability

- **Adding load to the system should result in a graceful decline in performance of individual jobs**
 - Not failure of the system
- **Increasing resources should support a proportional increase in load capacity**

The Motivation For Hadoop

Problems with Traditional Large-Scale Systems

Requirements for a New Approach



Hadoop!

Conclusion

Hadoop's History

- **Hadoop is based on work done by Google in the early 2000s**
 - Specifically, on papers describing the Google File System (GFS) published in 2003, and MapReduce published in 2004
- **This work takes a radical new approach to the problem of distributed computing**
 - Meets all the requirements we have for reliability, scalability etc
- **Core concept: distribute the data as it is initially stored in the system**
 - Individual nodes can work on data local to those nodes
 - No data transfer over the network is required for initial processing

Core Hadoop Concepts

- **Applications are written in high-level code**
 - Developers do not worry about network programming, temporal dependencies etc
- **Nodes talk to each other as little as possible**
 - Developers should not write code which communicates between nodes
 - ‘Shared nothing’ architecture
- **Data is spread among machines in advance**
 - Computation happens where the data is stored, wherever possible
 - Data is replicated multiple times on the system for increased availability and reliability

Hadoop: Very High-Level Overview

- When data is loaded into the system, it is split into ‘blocks’
 - Typically 64MB or 128MB
- Map tasks (the first part of the MapReduce system) work on relatively small portions of data
 - Typically a single block
- A master program allocates work to nodes such that a Map task will work on a block of data stored locally on that node
 - Many nodes work in parallel, each on their own part of the overall dataset

Fault Tolerance

- If a node fails, the master will detect that failure and re-assign the work to a different node on the system
- Restarting a task does not require communication with nodes working on other portions of the data
- If a failed node restarts, it is automatically added back to the system and assigned new tasks
- If a node appears to be running slowly, the master can redundantly execute another instance of the same task
 - Results from the first to finish will be used

The Motivation For Hadoop

Problems with Traditional Large-Scale Systems

Requirements for a New Approach

Hadoop!



Conclusion

The Motivation For Hadoop

In this chapter you have learned:

- **What problems exist with ‘traditional’ large-scale computing systems**
- **What requirements an alternative approach should have**
- **How Hadoop addresses those requirements**



Chapter 3

Hadoop: Basic Concepts

Hadoop: Basic Concepts

In this chapter you will learn

- What Hadoop is
- What features the Hadoop Distributed File System (HDFS) provides
- The concepts behind MapReduce
- How a Hadoop cluster operates

Hadoop: Basic Concepts



What Is Hadoop?

The Hadoop Distributed File System (HDFS)

How MapReduce works

Anatomy of a Hadoop Cluster

Conclusion

Hadoop Components

- **Hadoop consists of two core components**
 - The Hadoop Distributed File System (HDFS)
 - MapReduce Software Framework
- **There are many other projects based around core Hadoop**
 - Often referred to as the ‘Hadoop Ecosystem’
 - Pig, Hive, HBase, Flume, Oozie, Sqoop, etc
 - Many are discussed later in the course
- **A set of machines running HDFS and MapReduce is known as a Hadoop Cluster**
 - Individual machines are known as nodes
 - A cluster can have as few as one node, as many as several thousands
 - More nodes = better performance!

Hadoop Components: HDFS

- **HDFS, the Hadoop Distributed File System, is responsible for storing data on the cluster**
- **Data files are split into blocks and distributed across multiple nodes in the cluster**
- **Each block is replicated multiple times**
 - Default is to replicate each block three times
 - Replicas are stored on different nodes
 - This ensures both reliability and availability

Hadoop Components: MapReduce

- **MapReduce is the system used to process data in the Hadoop cluster**
- **Consists of two phases: Map, and then Reduce**
- **Each Map task operates on a discrete portion of the overall dataset**
 - Typically one HDFS data block
- **After all Maps are complete, the MapReduce system distributes the intermediate data to nodes which perform the Reduce phase**
 - Much more on this later!

Hadoop: Basic Concepts

What Is Hadoop?

 **The Hadoop Distributed File System (HDFS)**

How MapReduce works

Anatomy of a Hadoop Cluster

Conclusion

HDFS Basic Concepts

- **HDFS is a filesystem written in Java**
 - Based on Google's GFS
- **Sits on top of a native filesystem**
 - ext3, xfs etc
- **Provides redundant storage for massive amounts of data**
 - Using cheap, unreliable computers

HDFS Basic Concepts (cont'd)

- **HDFS performs best with a ‘modest’ number of large files**
 - Millions, rather than billions, of files
 - Each file typically 100Mb or more
- **Files in HDFS are ‘write once’**
 - No random writes to files are allowed
- **HDFS is optimized for large, streaming reads of files**
 - Rather than random reads

How Files Are Stored

- **Files are split into blocks**
- **Data is distributed across many machines at load time**
 - Different blocks from the same file will be stored on different machines
 - This provides for efficient MapReduce processing (see later)
- **Blocks are replicated across multiple machines, known as DataNodes**
 - Default replication is three-fold
 - i.e., each block exists on three different machines
- **A master node called the NameNode keeps track of which blocks make up a file, and where those blocks are located**
 - Known as the metadata

Getting Data in and out of HDFS

- **Hadoop API**

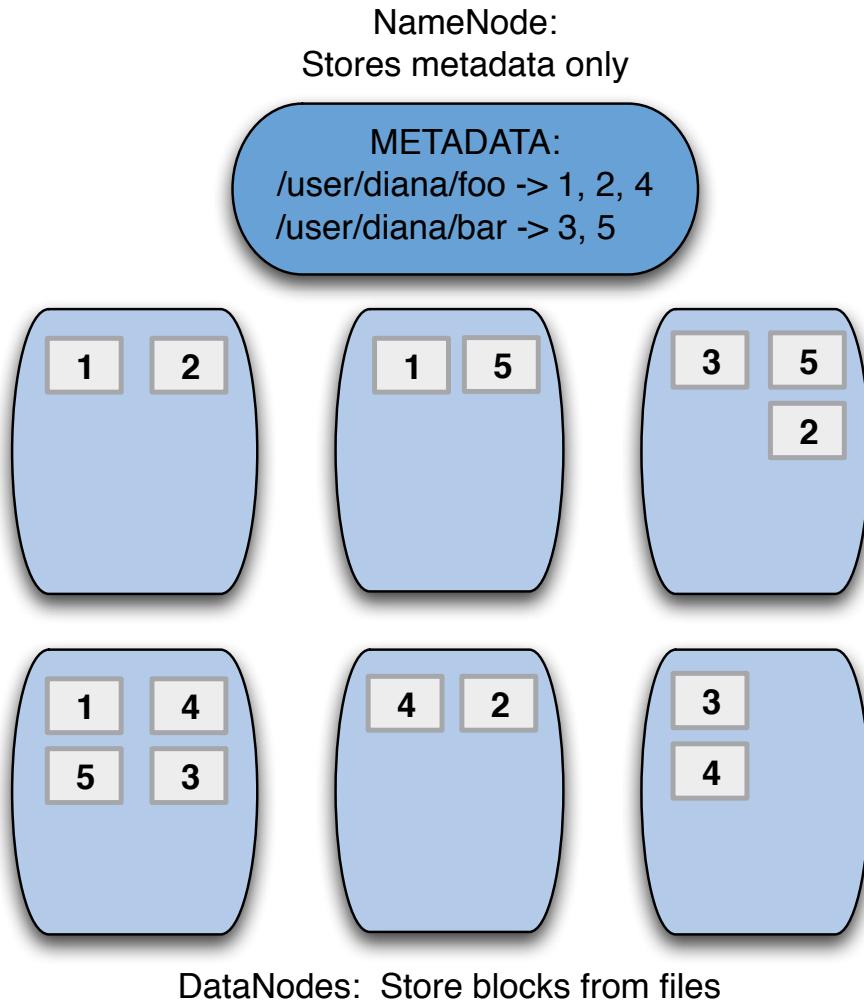
- Use `hadoop fs` to work with data in HDFS
- `hadoop fs -copyFromLocal local_dir /hdfs_dir`
- `hadoop fs -copyToLocal /hdfs_dir local_dir`

- **Ecosystem Projects**

- Flume
 - Collects data from log generating sources (e.g., Websites, syslogs, STDOUT)
- Sqoop
 - Extracts and/or inserts data between HDFS and RDBMS

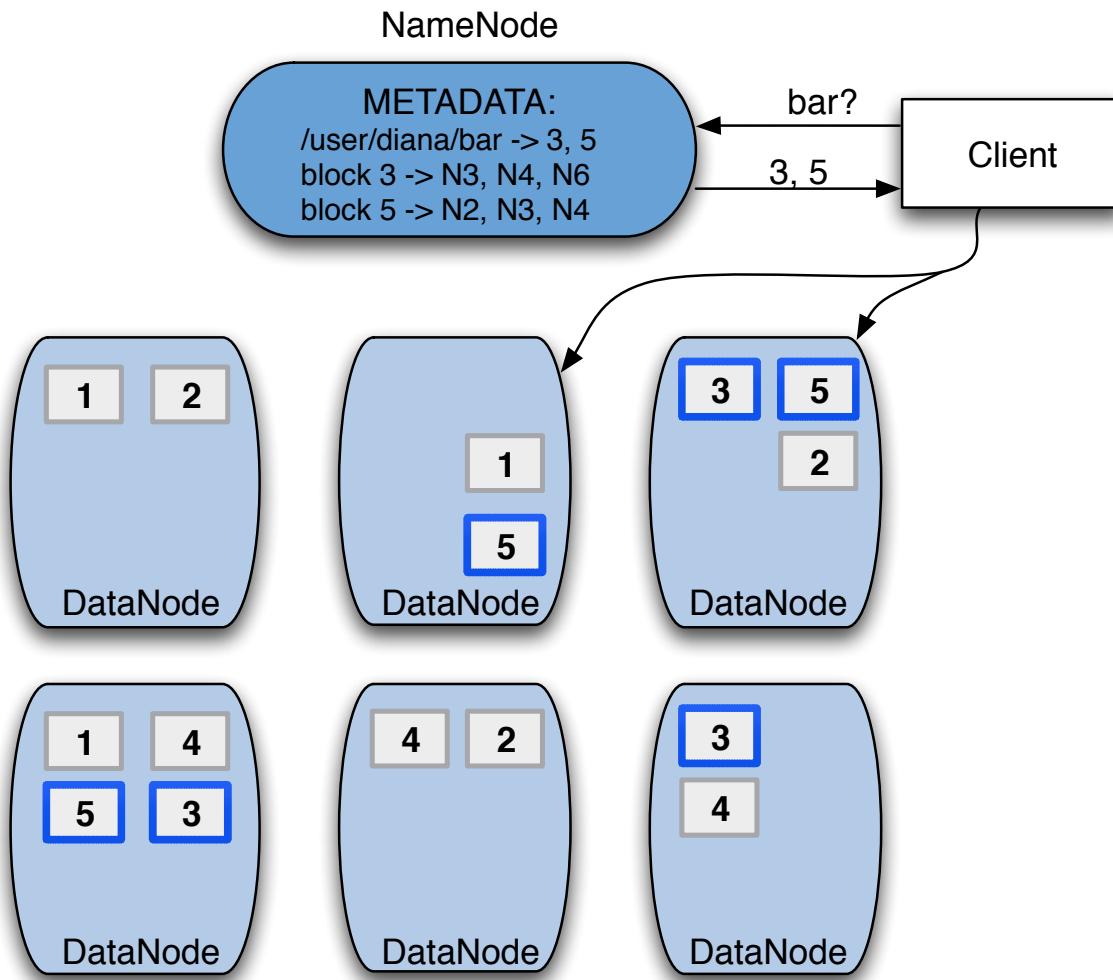
- **Business Intelligence Tools**

How Files Are Stored: Example



- **NameNode holds metadata for the data files**
- **DataNodes hold the actual blocks**
 - Each block is replicated three times on the cluster

HDFS: Points To Note



- **When a client application wants to read a file:**

- It communicates with the NameNode to determine which blocks make up the file, and which DataNodes those blocks reside on
- It then communicates directly with the DataNodes to read the data

Hadoop: Basic Concepts

What Is Hadoop?

The Hadoop Distributed File System (HDFS)

 **How MapReduce works**

Anatomy of a Hadoop Cluster

Conclusion

What Is MapReduce?

- **MapReduce is a method for distributing a task across multiple nodes**
- **Each node processes data stored on that node**
 - Where possible
- **Consists of two phases:**
 - Map
 - Reduce

Features of MapReduce

- **Automatic parallelization and distribution**
- **Fault-tolerance**
- **Status and monitoring tools**
- **A clean abstraction for programmers**
 - MapReduce programs are usually written in Java
- **MapReduce abstracts all the ‘housekeeping’ away from the developer**
 - Developer can concentrate simply on writing the Map and Reduce functions

MapReduce: The Mapper

- Hadoop attempts to ensure that Mappers run on nodes which hold their portion of the data locally, to avoid network traffic
 - Multiple Mappers run in parallel, each processing a portion of the input data
- The Mapper reads data in the form of key/value pairs
- It outputs zero or more key/value pairs

```
map(in_key, in_value) ->  
    (inter_key, inter_value) list
```

MapReduce: The Mapper (cont'd)

- **The Mapper may use or completely ignore the input key**
 - For example, a standard pattern is to read a line of a file at a time
 - The key is the byte offset into the file at which the line starts
 - The value is the contents of the line itself
 - Typically the key is considered irrelevant
- **If it writes anything at all out, the output must be in the form of key/value pairs**

MapReduce Example: Word Count

- Count the number of occurrences of each word in a large amount of input data

```
Map(input_key, input_value)
    foreach word w in input_value:
        emit(w, 1)
```

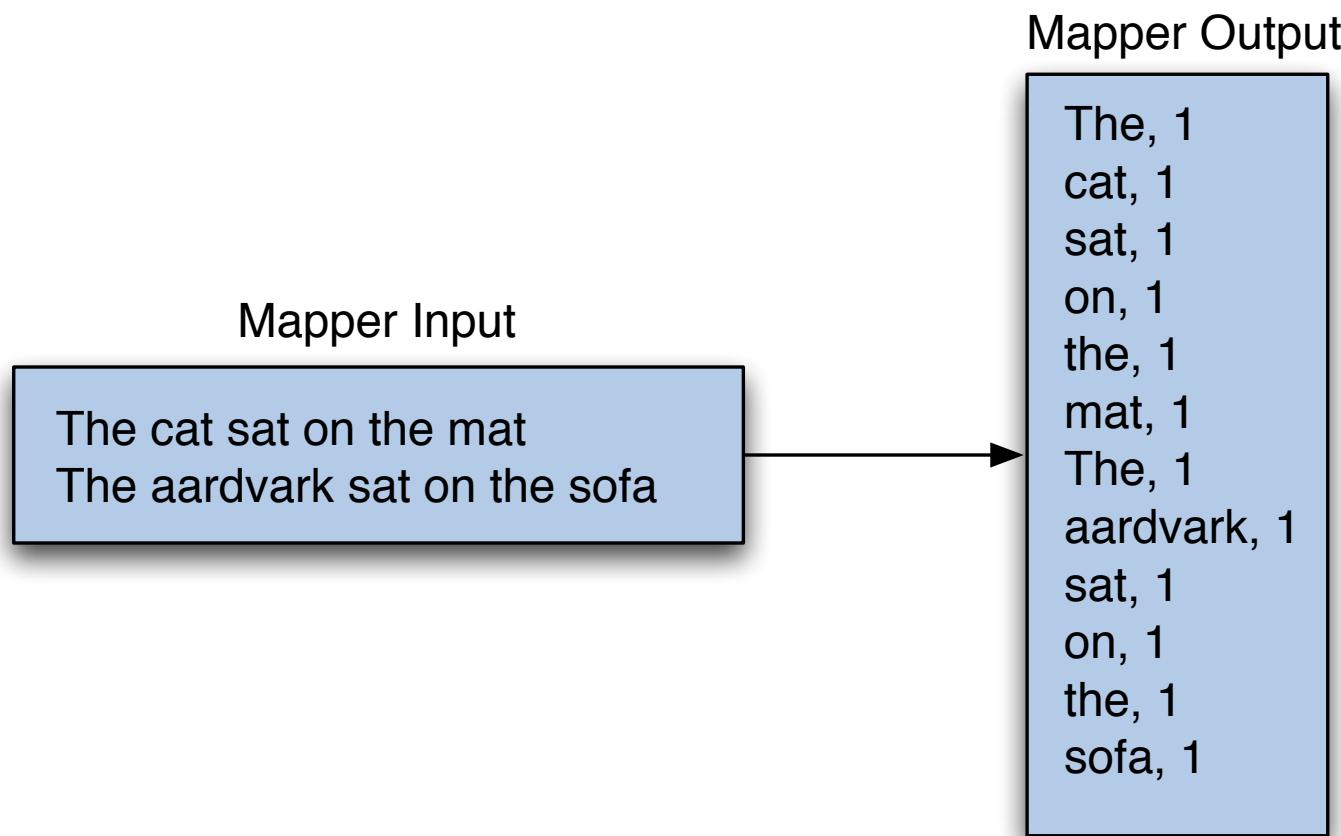
- Input to the Mapper

```
(3414, 'the cat sat on the mat')
(3437, 'the aardvark sat on the sofa')
```

- Output from the Mapper

```
('the', 1), ('cat', 1), ('sat', 1), ('on', 1),
('the', 1), ('mat', 1), ('the', 1), ('aardvark', 1),
('sat', 1), ('on', 1), ('the', 1), ('sofa', 1)
```

Map Phase



MapReduce: The Reducer

- After the Map phase is over, all the intermediate values for a given intermediate key are combined together into a list
- This list is given to a Reducer
 - There may be a single Reducer, or multiple Reducers
 - All values associated with a particular intermediate key are guaranteed to go to the same Reducer
 - The intermediate keys, and their value lists, are passed to the Reducer in sorted key order
 - This step is known as the ‘shuffle and sort’
- The Reducer outputs zero or more final key/value pairs
 - These are written to HDFS
 - In practice, the Reducer usually emits a single key/value pair for each input key

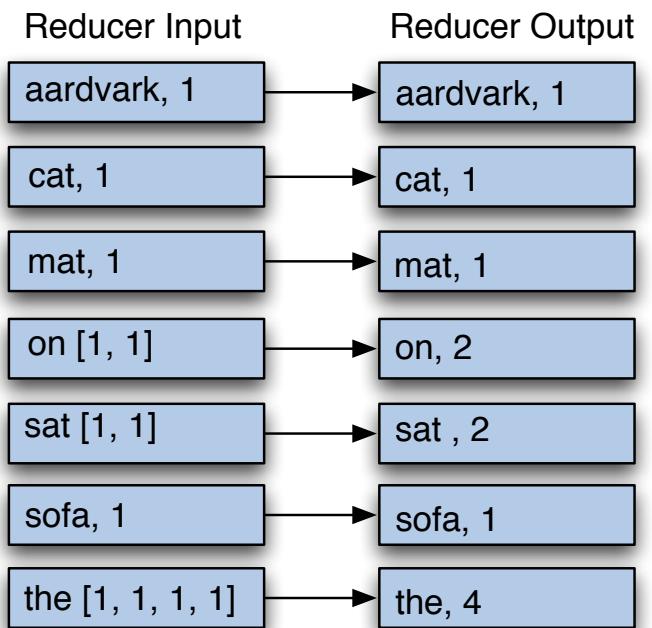
Example Reducer: Sum Reducer

- Add up all the values associated with each intermediate key:

```
reduce(output_key,  
       intermediate_vals)  
  set count = 0  
  foreach v in intermediate_vals:  
    count += v  
  emit(output_key, count)
```

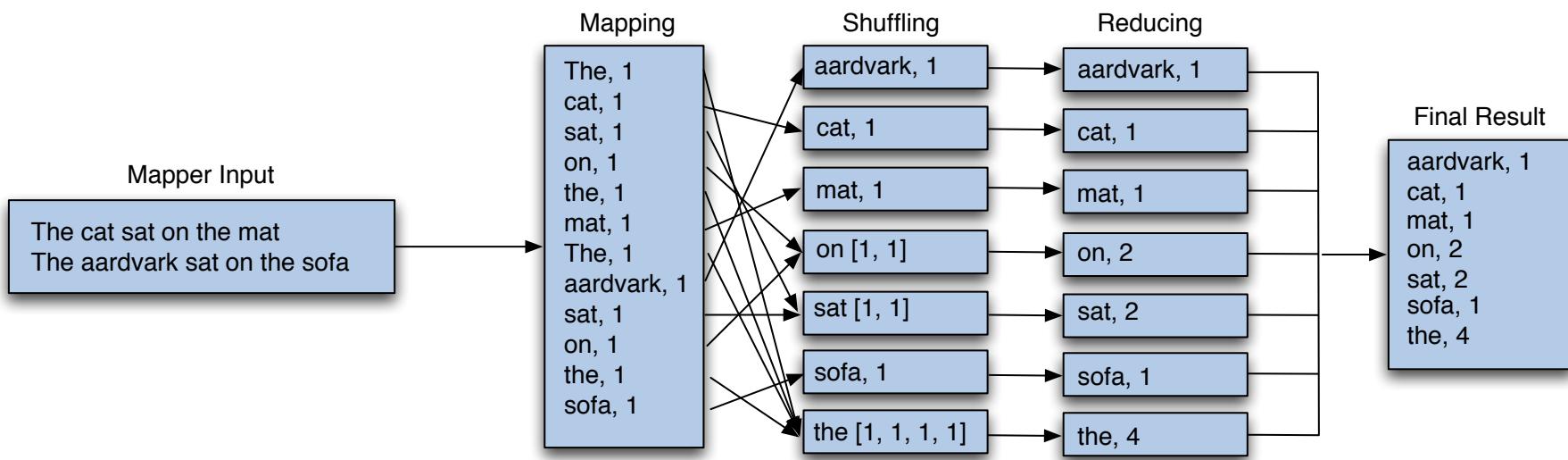
- Reducer output:

```
('aardvark', 1)  
('cat', 1)  
('mat', 1)  
('on', 2)  
('sat', 2)  
('sofa', 1)  
('the', 4)
```



Putting It All Together

The overall word count process



Why Do We Care About Counting Words?

- **Word count is challenging over massive amounts of data**
 - Using a single compute node would be too time-consuming
 - Using distributed nodes require moving data
 - Number of unique words can easily exceed the RAM
 - Would need a hash table on disk
 - Would need to partition the results (sort and shuffle)
- **Fundamentals of statistics often are simple aggregate functions**
- **Most aggregation functions have distributive nature**
 - e.g., max, min, sum, count
- **MapReduce breaks complex tasks down into smaller elements which can be executed in parallel**

Hadoop: Basic Concepts

What Is Hadoop?

The Hadoop Distributed File System (HDFS)

How MapReduce works

 **Anatomy of a Hadoop Cluster**

Conclusion

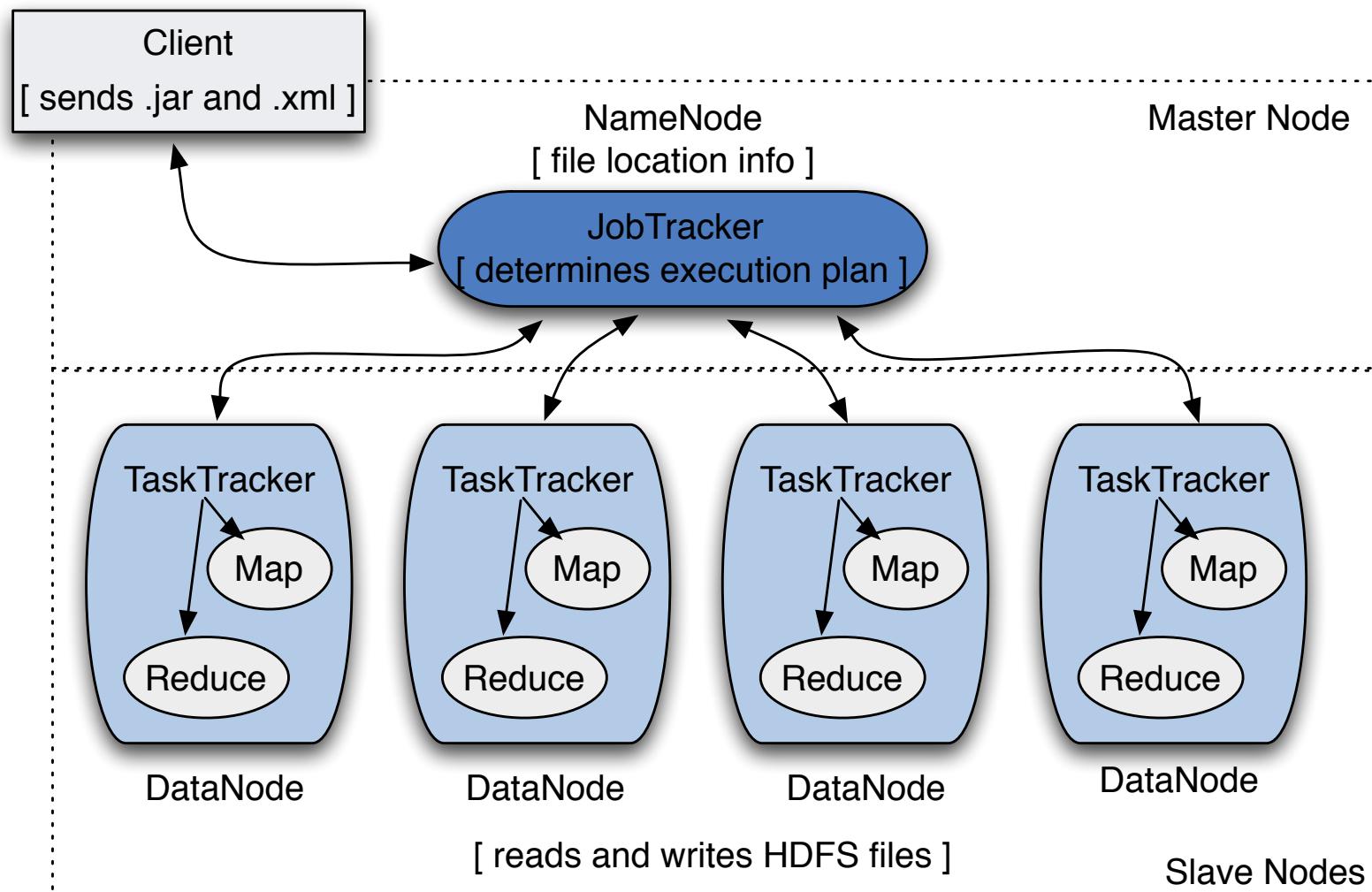
Installing A Hadoop Cluster

- **Cluster installation is usually performed by the system administrator**
 - Cloudera offers a Hadoop for System Administrators course specifically aimed at those responsible for commissioning and maintaining Hadoop clusters
- **Easiest way to download and install Hadoop is by using Cloudera's Distribution including Apache Hadoop (CDH)**
 - Vanilla Hadoop plus many patches, backports of future features
 - Current version has over 900 patches applied
 - Full documentation available at <http://cloudera.com>

The Five Hadoop Daemons

- **Hadoop is comprised of five separate daemons:**
- **NameNode**
 - Holds the metadata for HDFS
- **Secondary NameNode**
 - Performs housekeeping functions for the NameNode
 - Is not a backup or hot standby for the NameNode!
- **DataNode**
 - Stores actual HDFS data blocks
- **JobTracker**
 - Manages MapReduce jobs, distributes individual tasks to...
- **TaskTracker**
 - Responsible for instantiating and monitoring individual Map and Reduce tasks

Basic Cluster Configuration



Mitigating Risk

- **Single Points of Failure**
 - NameNode
 - Secondary NameNode
 - Jobtracker
- **Automatic restart and Failover not yet supported**
- **Mitigating Risk**
 - Carrier-grade Hardware
 - Hot Standbys
 - Monitoring

Hadoop: Basic Concepts

What Is Hadoop?

The Hadoop Distributed File System (HDFS)

How MapReduce works

Anatomy of a Hadoop Cluster



Conclusion

Conclusion

In this chapter you have learned

- **What Hadoop is**
- **What features the Hadoop Distributed File System (HDFS) provides**
- **The concepts behind MapReduce**
- **How a Hadoop cluster operates**



Chapter 4

Hadoop Solutions

Hadoop Solutions

In this chapter you will learn:

- **The most common problems Hadoop can solve**
- **The types of analytics often performed with Hadoop**
- **Where the data comes from**
- **The benefits of analyzing data with Hadoop**
- **How some real-world companies use Hadoop**

Hadoop Solutions



Eight Common Hadoop-able Problems

Orbitz Hadoop Use Case

Other Examples

Conclusion

Where Does Data Come From?

- **Science**

- Medical imaging, sensor data, genome sequencing, weather data, satellite feeds, etc.

- **Industry**

- Financial, pharmaceutical, manufacturing, insurance, online, energy, retail data

- **Legacy**

- Sales data, customer behavior, product databases, accounting data, etc.

- **System Data**

- Log files, health & status feeds, activity streams, network messages, Web Analytics, intrusion detection, spam filters

Analyzing Data: The Challenges

- Huge volumes of data
- Mixed sources result in many different formats
 - XML
 - CSV
 - EDI
 - LOG
 - Objects
 - SQL
 - Text
 - JSON
 - Binary
 - Etc.

What is Common Across Hadoop-able Problems?

- **Nature of the data**

- Complex data
- Multiple data sources
- Lots of it



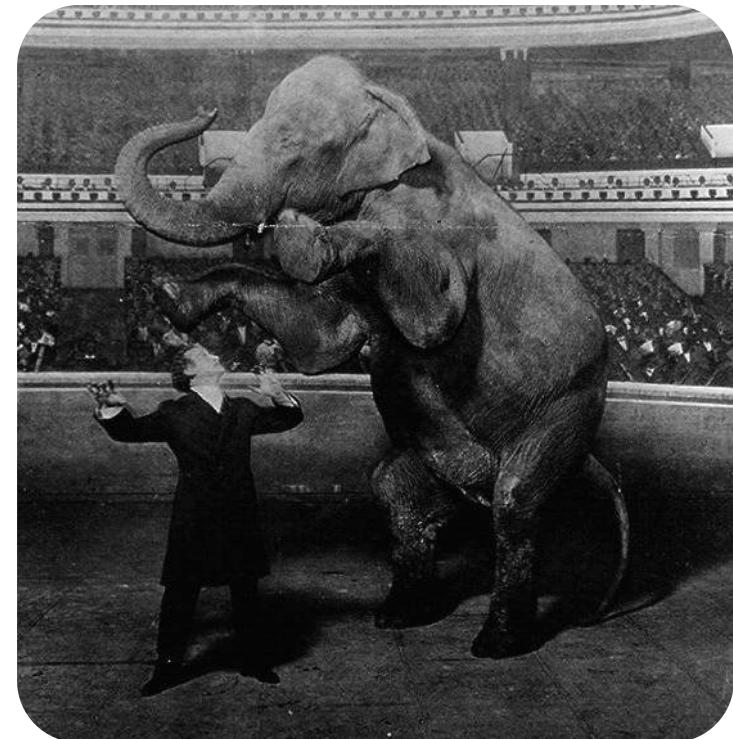
- **Nature of the analysis**

- Batch processing
- Parallel execution
- Spread data over a cluster of servers and take the computation to the data



Benefits of Analyzing With Hadoop

- Previously impossible/impractical to do this analysis
- Analysis conducted at lower cost
- Analysis conducted in less time
- Greater flexibility
- Linear scalability



What Analysis is Possible With Hadoop?

- Text mining
- Index building
- Graph creation and analysis
- Pattern recognition
- Collaborative filtering
- Prediction models
- Sentiment analysis
- Risk assessment

Eight Common Hadoop-able Problems

- 1. Modeling true risk**
- 2. Customer churn analysis**
- 3. Recommendation engine**
- 4. PoS transaction analysis**
- 5. Analyzing network data to predict failure**
- 6. Threat analysis**
- 7. Search quality**
- 8. Data “sandbox”**

1. Modeling True Risk

Challenge:

- **How much risk exposure does an organization really have with each customer?**
 - Multiple sources of data and across multiple lines of business

Solution with Hadoop:

- **Source and aggregate disparate data sources to build data picture**
 - e.g. credit card records, call recordings, chat sessions, emails, banking activity
- **Structure and analyze**
 - Sentiment analysis, graph creation, pattern recognition

Typical Industry:

- Financial Services (banks, insurance companies)



2. Customer Churn Analysis

Challenge:

- Why is an organization really losing customers?
 - Data on these factors comes from different sources

Solution with Hadoop:

- Rapidly build behavioral model from disparate data sources
- Structure and analyze with Hadoop
 - Traversing
 - Graph creation
 - Pattern recognition

Typical Industry:

- Telecommunications, Financial Services



3. Recommendation Engine/Ad Targeting

Challenge:

- **Using user data to predict which products to recommend**

Solution with Hadoop:

- **Batch processing framework**
 - Allow execution in parallel over large datasets
- **Collaborative filtering**
 - Collecting ‘taste’ information from many users
 - Utilizing information to predict what similar users like

Typical Industry

- Ecommerce, Manufacturing, Retail
- Advertising



4. Point of Sale Transaction Analysis

Challenge:

- **Analyzing Point of Sale (PoS) data to target promotions and manage operations**
 - Sources are complex and data volumes grow across chains of stores and other sources

Solution with Hadoop:

- **Batch processing framework**
 - Allow execution in parallel over large datasets
- **Pattern recognition**
 - Optimizing over multiple data sources
 - Utilizing information to predict demand

Typical Industry:

- Retail



5. Analyzing Network Data to Predict Failure

Challenge:

- **Analyzing real-time data series from a network of sensors**
 - Calculating average frequency over time is extremely tedious because of the need to analyze terabytes

Solution with Hadoop:

- **Take the computation to the data**
 - Expand from simple scans to more complex data mining
- **Better understand how the network reacts to fluctuations**
 - Discrete anomalies may, in fact, be interconnected
- **Identify leading indicators of component failure**

Typical Industry:

- Utilities, Telecommunications, Data Centers



6. Threat Analysis/Trade Surveillance

Challenge:

- Detecting threats in the form of fraudulent activity or attacks
 - Large data volumes involved
 - Like looking for a needle in a haystack

Solution with Hadoop:

- Parallel processing over huge datasets
- Pattern recognition to identify anomalies,
 - i.e., threats

Typical Industry:

- Security, Financial Services,
General: spam fighting, click fraud



7. Search Quality

Challenge:

- **Providing real time meaningful search results**

Solution with Hadoop:

- **Analyzing search attempts in conjunction with structured data**
- **Pattern recognition**
 - Browsing pattern of users performing searches in different categories

Typical Industry:

- Web, Ecommerce

8. Data “Sandbox”

Challenge:

- **Data Deluge**

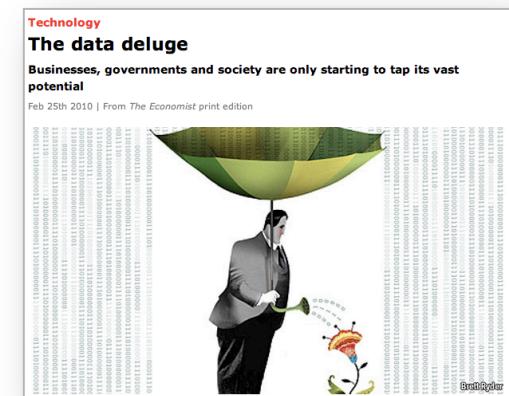
- Don't know what to do with the data or what analysis to run

Solution with Hadoop:

- “Dump” all this data into an HDFS cluster
- Use Hadoop to start trying out different analysis on the data
- See patterns to derive value from data

Typical Industry:

- Common across all industries



Hadoop Solutions

Eight Common Hadoop-able Problems



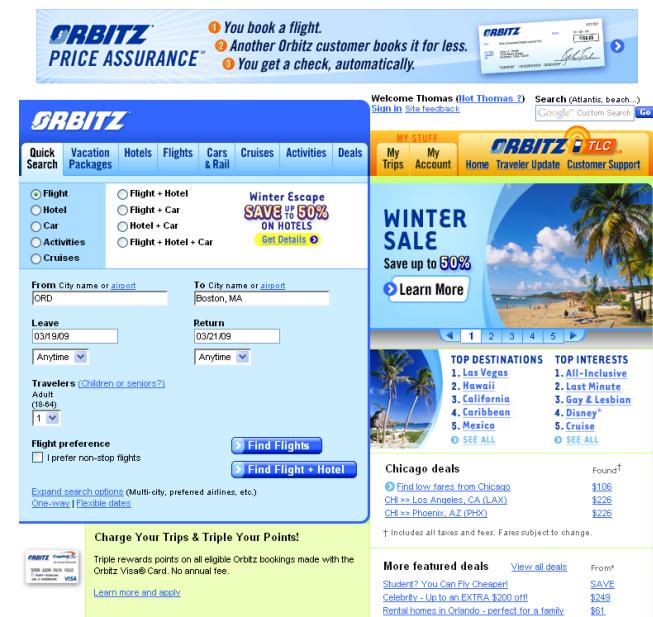
Orbitz Hadoop Use Case

Other Examples

Conclusion

Who Is Orbitz?

- **Orbitz Worldwide**
- **Orbitz started in 1999, Orbitz site launched in 2001**
 - Leading online travel consumer brands including Orbitz, Cheaptickets, The Away Network, ebookers and HotelClub.
- **Business to business services**
 - Orbitz Worldwide Distribution provides hotel booking capabilities to a number of leading carriers such as Amtrak, Delta, LAN, KLM, Air France
 - Orbitz for Business provides corporate travel services to a number of Fortune 100 clients



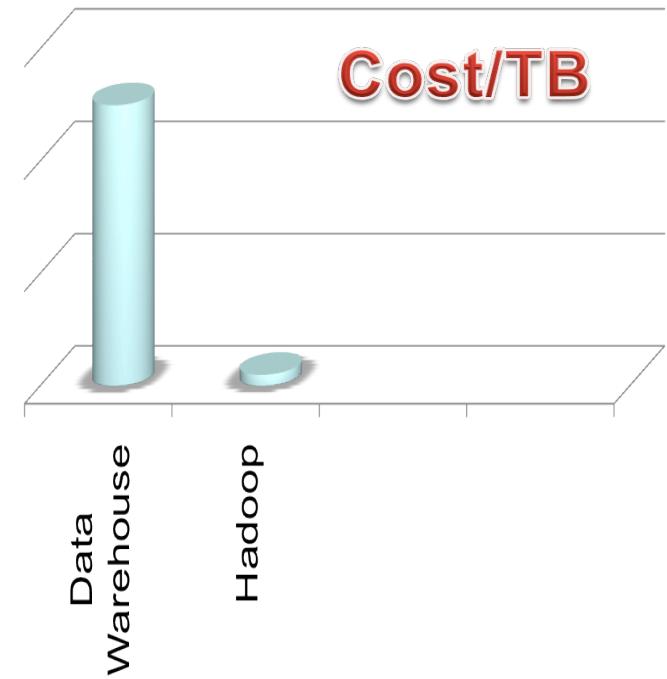
Why Is Orbitz Using Hadoop?

- **Challenge:**

- Orbitz performs millions of searches and transactions daily, which leads to hundreds of gigabytes of log data every day
- Not all of that data has value (i.e., it is logged for historic reasons)
- Much is quite valuable
- Want to capture even more data

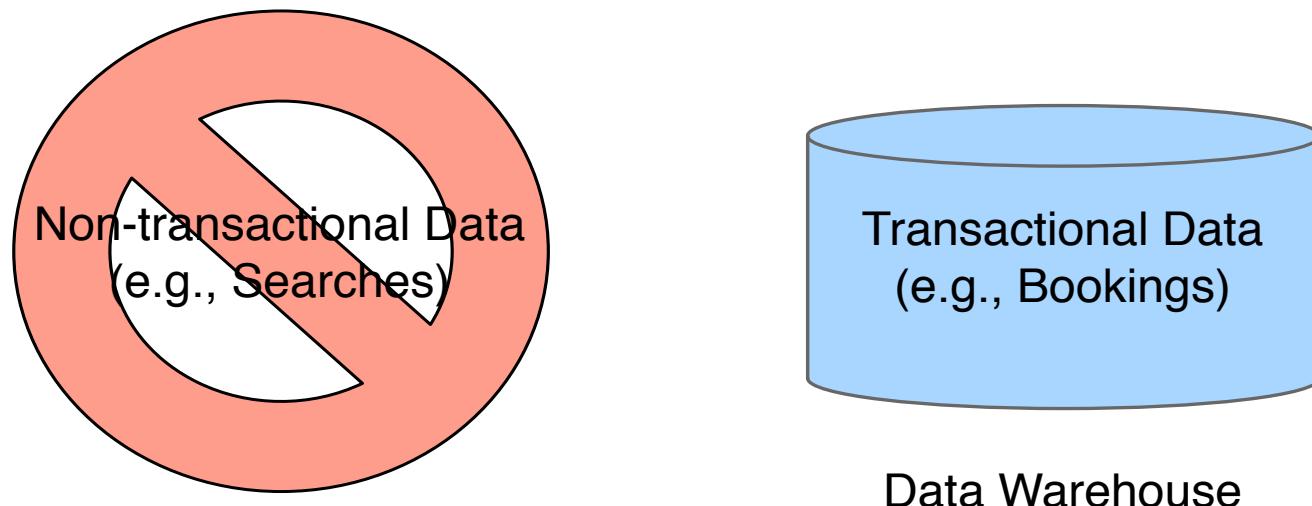
- **Solution with Hadoop:**

- Hadoop provides Orbitz with efficient, economical, scalable, and reliable storage and processing of these large amounts of data
- Hadoop places no constraints on how data is processed



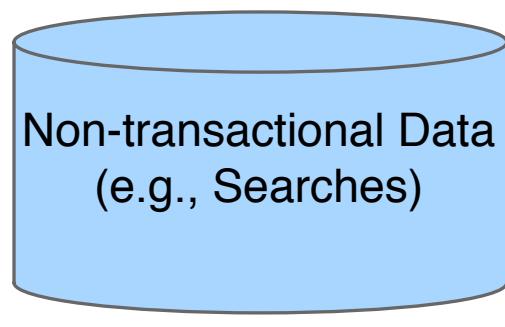
Before Hadoop

- Orbitz's data warehouse contains a full archive of all transactions
 - Every booking, refund, cancellation etc.
- Non-transactional data was thrown away because it was uneconomical to store

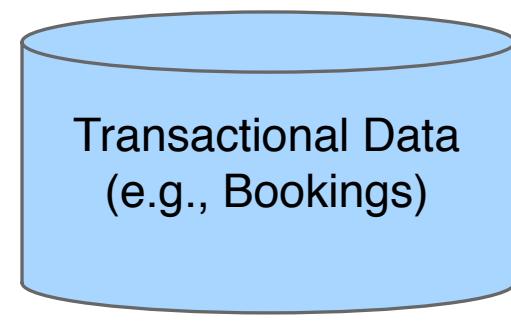


After Hadoop

- Hadoop was deployed late 2009/early 2010 to begin collecting this non-transactional data. Orbitz has been using CDH for that entire period with great success.
- Much of this non-transactional data is contained in web analytics logs.



Hadoop



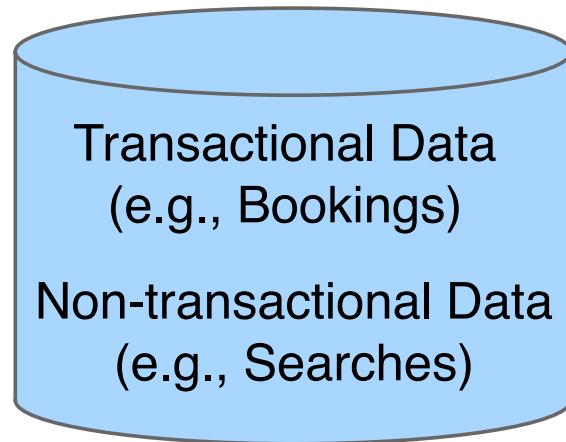
Data Warehouse

What Now?

- **Access to this non-transactional data enables a number of applications...**
 - Optimizing hotel search
 - E.g., optimize hotel ranking and show consumers hotels more closely matching their preferences
 - User specific product Recommendations
 - Web page performance tracking
 - Analyses to optimize search result cache performance
 - User segments analysis, which can drive personalization
 - E.g., Safari users click on hotels with higher mean and median prices as opposed to other users.

Next Steps

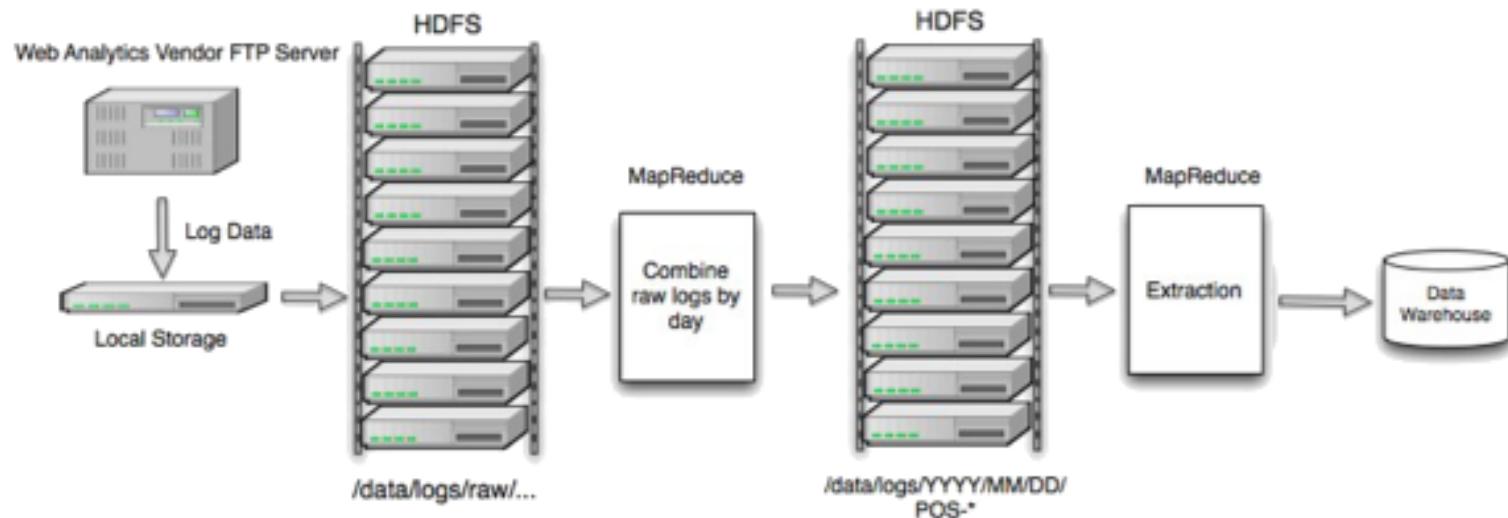
- Most of these efforts are driven by development teams
- Next challenge is to make data more available to the rest of the organization
- The goal is a unified view of the data, allowing Orbitz to use the power of its existing tools for reporting and analysis.
 - E.g., JasperSoft, Pentaho, Informatica, MicroStrategy, etc.



Data Warehouse

How Orbitz Is Aggregating Data For Import

- Typical processing flow for large volumes of non-transactional data being collected at Orbitz
 - Converts large volumes of un-structured data into structured data
 - Structured data can then be queried, extracted, or exported into the data warehouse



Hadoop Solutions

Ten Common Hadoop-able Problems

Orbitz Hadoop Use Case



Other Examples

Conclusion

Case Study: A Major National Bank

- **Background**

- 100M customers
- Relational data: 2.5B records/month
 - Card transactions, home loans, auto loans, etc.
 - Data volume growing by TB+/year
- Needs to incorporate non-relational data as well
 - Web clicks, check images, voice data

- **Uses Hadoop to**

- Identify credit risk, fraud
- Proactively manage capital

Case study: Netflix

- **Before Hadoop**
 - Nightly processing of logs
 - Imported into a database
 - Analysis/BI
- **As data volume grew, it took more than 24 hours to process and load a day's worth of logs**
- **Today, an hourly Hadoop job processes logs for quicker availability to the data for analysis/BI**
- **Currently ingesting approx. 1TB/day**

Case Study: Hadoop as cheap storage

- **Yahoo**
 - Before Hadoop: 1 million for 10 TB storage
 - With Hadoop: \$1 million for 1 PB of storage
- **Other Large Company**
 - Before Hadoop: \$5 million to store data in Oracle
 - With Hadoop: \$240k to store the data in HDFS
- **Facebook**
 - Hadoop as unified storage

Hadoop Solutions

10 Common Hadoop-able Problems

Orbitz Hadoop Use Case

Other Examples



Conclusion

Conclusion

- **Hadoop has become a valuable business intelligence tool, and will become an increasingly important part of a BI infrastructure**
- **Hadoop won't replace your EDW**
 - But any organization with a large EDW should at least be exploring Hadoop as a complement to its BI infrastructure
- **Use Hadoop to offload the time and resource intensive processing of large data sets so you can free up your data warehouse to serve user needs**

Conclusion (cont'd)

- **Data is big and getting bigger**
- **Data is often unstructured or complex**
- **Hadoop is used to retrieve value out of data**
- **Examples are Orbitz, eBay, Netflix, eHarmony, etc.**
- **Benefits of Hadoop:**
 - Handles less structured data
 - Return On Byte
 - Lower TCO



Chapter 5

The Hadoop Ecosystem

The Hadoop Ecosystem

In this chapter you will learn

- What other projects exist around core Hadoop
- The differences between Hive and Pig
- When to use HBase
- How Flume is typically deployed
- What other ecosystem projects exist

The Hadoop Ecosystem



Introduction

Hive and Pig

HBase

Flume

Other Ecosystem Projects

Conclusion

Introduction

- **The term ‘Hadoop’ is taken to be the combination of HDFS and MapReduce**
- **There are numerous other projects surrounding Hadoop**
 - Typically referred to as the ‘Hadoop Ecosystem’
 - Most are incorporated into Cloudera’s Distribution Including Apache Hadoop (CDH)
- **All use either HDFS, MapReduce, or both**

The Hadoop Ecosystem

Introduction

 **Hive and Pig**

HBase

Flume

Other Ecosystem Projects

Conclusion

Hive and Pig

- Although MapReduce is very powerful, it can also be complex to master
- Many organizations have business or data analysts who are skilled at writing SQL queries, but not at writing Java code
- Many organizations have programmers who are skilled at writing code in scripting languages
- Hive and Pig are two projects which evolved separately to help such people analyze huge amounts of data via MapReduce
 - Hive was initially developed at Facebook, Pig at Yahoo!
- Cloudera offers a two-day course, *Cloudera Training For Apache Hive and Pig*

Hive and Pig

- What is Hive?
 - An SQL-like interface to Hadoop

```
SELECT * FROM purchases WHERE price > 100 GROUP BY  
storeid
```

- What is Pig?
 - A dataflow language for transforming large data sets

```
purch = LOAD "/user/dave/purchases" AS (itemID,  
                                         price, storeID, purchaserID);  
bigticket = FILTER purchases BY price > 10000;  
....
```

Hive vs. Pig

	Hive	Pig
Language	HiveQL (SQL-like)	Pig Latin, a dataflow language
Schema	Table definitions that are stored in a metastore	A schema is optionally defined at runtime. Metastore coming soon
Programmatic access	JDBC, ODBC	PigServer (Java API)

The Hadoop Ecosystem

Introduction

Hive and Pig

 **HBase**

Flume

Other Ecosystem Projects

Conclusion

HBase: ‘The Hadoop Database’

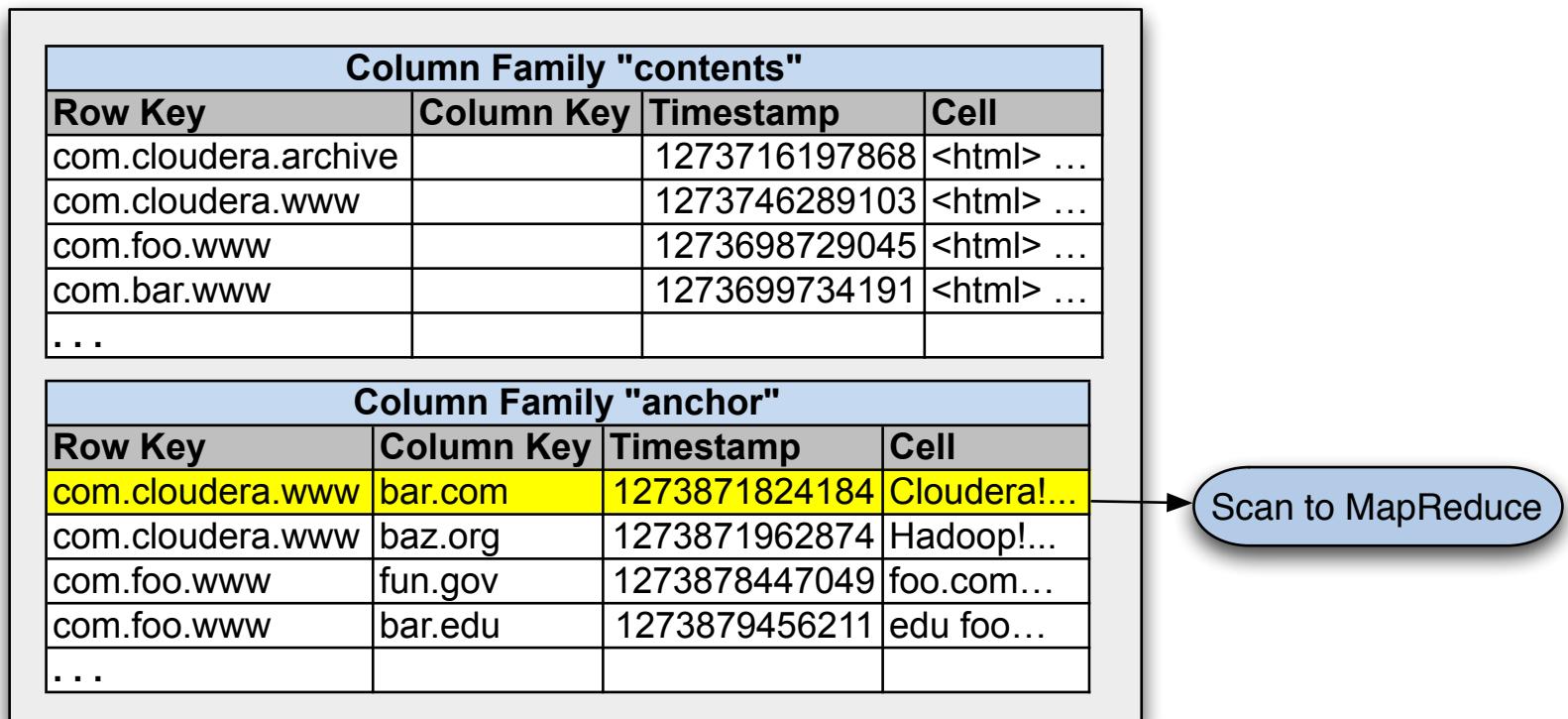
- **HBase is a column family-store database layered on top of HDFS**
 - Based on Google’s Big Table
 - Provides interactive access to data
- **Can store massive amounts of data**
 - Multiple Terabytes, up to Petabytes of data
- **High Write Throughput**
 - Scales up to millions of writes per second
- **Copes well with sparse data**
 - Tables can have many thousands of columns
 - Even if a given row only has data in a few of the columns
- **Has a constrained access model**
 - Limited to lookup of a row by a single key
 - No transactions
 - Single row operations only

HBase vs A Traditional RDBMS

	RDBMS	HBase
Data layout	Row or column-oriented	Column Family-oriented
Transactions	Yes	Single row only
Query language	SQL	get/put/scan
Security	Authentication/Authorization	TBD
Indexes	Yes	Row-key only
Max data size	TBs	PB+
Read/write throughput limits	1000s queries/second	Millions of queries/second

HBase Data as Input to MapReduce Jobs

- Rows from an HBase table can be used as input to a MapReduce job
 - Each row is treated as a single record
 - MapReduce jobs can sort/search/index/query data in bulk



HBase Use Case

■ WorldLingo

- Hardware: 44 servers (each server has two dual-core CPUs, 2TB storage, 48GB RAM)
- Two separate Hadoop/HBase clusters with 22 nodes each.
- Hadoop is primarily used to run HBase and Map/Reduce jobs scanning over the HBase tables to perform specific tasks.
- HBase is used as a scalable and fast storage back end for millions of documents.
- Store 12 million documents with a target of 450 million in the near future.

When To Use HBase

- **Use HBase if...**

- You need random write, random read, or both (but not neither)
- You need to do many thousands of operations per second on multiple TB of data
- Your access patterns are well-known and simple

- **Don't use HBase if...**

- You only append to your dataset, and tend to read the whole thing
- You primarily do ad-hoc analytics (ill-defined access patterns)
- Your data easily fits on one beefy node

The Hadoop Ecosystem

Introduction

Hive and Pig

HBase

 **Flume**

Other Ecosystem Projects

Conclusion

What Is Flume?

- **Flume is a distributed, reliable, available service for efficiently moving large amounts of data as it is produced**
 - Ideally suited to gathering logs from multiple systems and inserting them into HDFS as they are generated
- **Developed in-house by Cloudera, and released as open-source software**
 - Now an Apache Incubator project
- **Design goals:**
 - Reliability
 - Scalability
 - Manageability
 - Extensibility

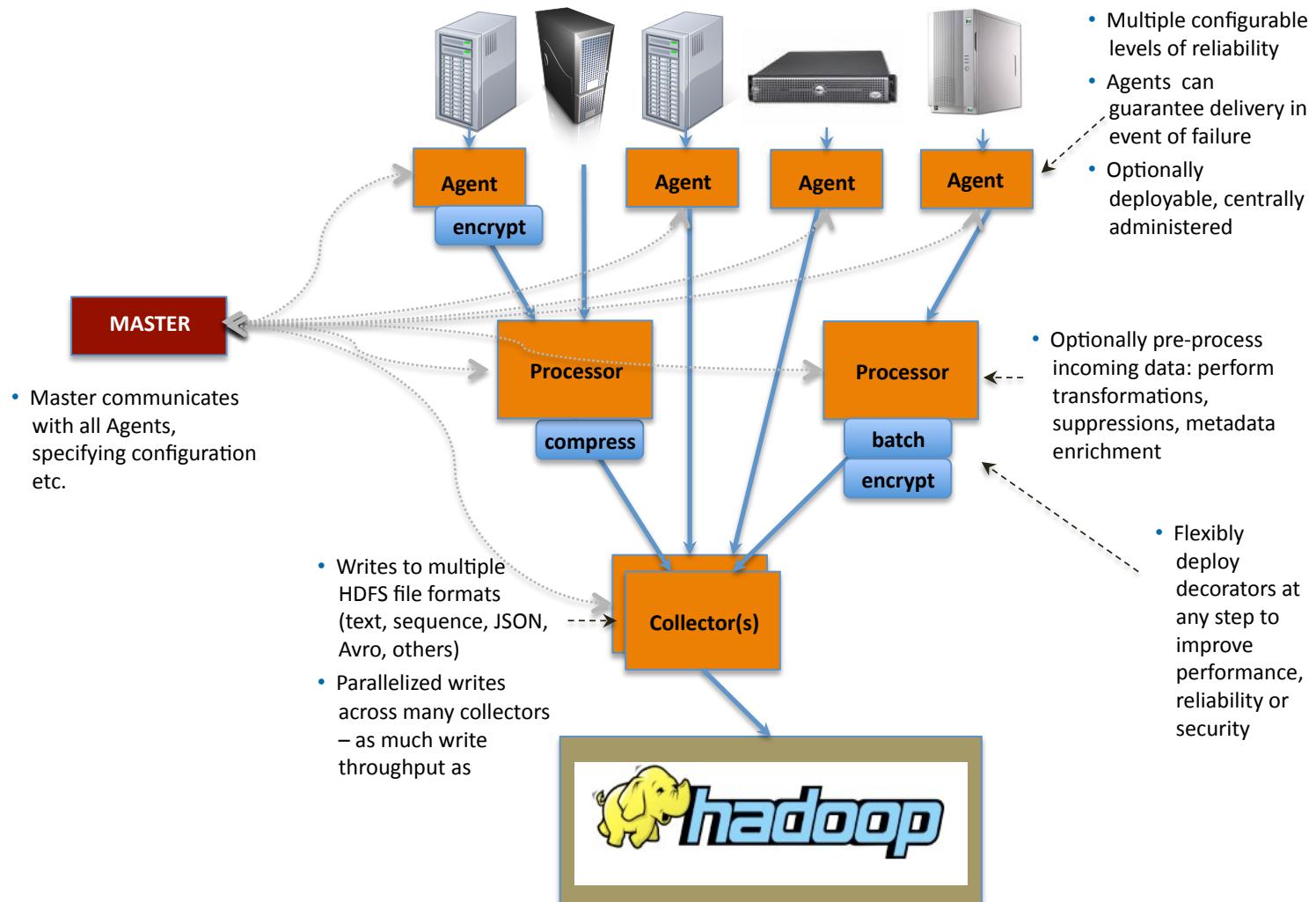
Flume's Design Goals

- **Flume is designed to continue delivering events in the face of system component failure**
- **Flume scales horizontally to support scalability**
 - As load increases, more machines can be added to the configuration
- **Flume provides a central Master controller for manageability**
 - Administrators can monitor and reconfigure data flows on the fly
- **Flume can be extended by adding connectors to existing storage layers or data platforms**
 - General sources already provided include data from files, syslog, and standard output (stdout) from a process
 - General endpoints already provided include files on the local filesystem or in HDFS
 - Other connectors can be added using Flume's API

Flume: General System Architecture

- **The Master holds configuration information for each Node, plus a version number for that node**
 - Version number is associated with the Node's configuration
- **Nodes communicate with the Master every five seconds**
 - Node passes its version number to the Master
 - If the Master has a later version number for the Node, it tells the Node to reconfigure itself
 - The Node then requests the new configuration information from the Master, and dynamically applies that new configuration

Flume: High-Level Overview



The Hadoop Ecosystem

Introduction

Hive and Pig

HBase

Flume

 **Other Ecosystem Projects**

Conclusion

Sqoop: Retrieving Data From RDBMSs

- **Sqoop: SQL to Hadoop**
- **Extracts data from RDBMSs and inserts it into HDFS**
 - Also works the other way around
- **Command-line tool which works with any RDBMS**
 - Optimizations available for some specific RDBMSs
- **Generates Writeable classes for use in MapReduce jobs**
- **Developed at Cloudera, released as Open Source**
 - Now an Apache Incubator project

Sqoop: Custom Connectors

- Cloudera has partnered with other vendors and developers to develop connectors between their applications and HDFS
 - MySQL
 - Postgres
 - Netezza
 - Teradata
 - Oracle (partnered with Quest Software)
- These connectors are made freely available as they are released
 - Not open-source, but free to use
- Support is available as part of Cloudera Enterprise

Oozie

- **Oozie provides a way for developers to define an entire workflow**
 - Comprised of multiple MapReduce jobs
- **Allows some jobs to run in parallel, others to wait for the output of a previous job**
- **Workflow definitions are written in XML**

Hue

- **Graphical front-end to developer and administrator functionality**
 - Uses a Web browser as its front-end
- **Developed by Cloudera, released as Open Source**
- **Extensible**
 - Publically-available API
- **Cloudera Enterprise includes extra functionality**
 - Advanced user management
 - Integration with LDAP, Active Directory
 - Accounting
 - Cluster Monitoring

Hue (cont'd)

The screenshot shows the Hue web interface running in Mozilla Firefox. The title bar reads "Hue - Mozilla Firefox". The address bar shows the URL "http://localhost:8088/#". The browser tabs include "Hue", "/hadoop", and "Upload Files". The main content area displays the "Flows : Flume" monitoring dashboard. It features two flows: "process-monitoring" and "syslog-test-flow". Each flow card provides metrics for nodes, unhealthy nodes, bytes in/out, and events in/out. The "process-monitoring" flow has 7 healthy nodes and 4 unhealthy nodes. The "syslog-test-flow" has 5 healthy nodes and 3 unhealthy nodes. The interface includes a toolbar at the bottom with various icons.

Flow	Nodes	Unhealthy Nodes	Bytes In (kb/s)	Bytes Out (kb/s)	Events In	Events Out
process-monitoring	7	4	Upward trend	Upward trend	Upward trend	Upward trend
syslog-test-flow	5	3	Upward trend	Upward trend	Upward trend	Upward trend

The Hadoop Ecosystem

Introduction

Hive and Pig

HBase

Flume

Other Ecosystem Projects



Conclusion

Conclusion

In this chapter you have learned

- **What other projects exist around core Hadoop**
- **The differences between Hive and Pig**
- **When to use HBase**
- **How Flume is typically deployed**
- **What other ecosystem projects exist**



Chapter 6

Conclusion

Conclusion

In this course, you have learned:

- **Why Hadoop is needed**
- **The basic concepts of HDFS and MapReduce**
- **What sort of problems can be solved with Hadoop**
- **What other projects are included in the Hadoop ecosystem**

Conclusion (cont'd)

- **Thank you for coming!**
- **Cloudera offers a wide range of courses including**
 - Developer training
 - System administrator training
 - Hive and Pig training
 - HBase training
- **Please see <http://university.cloudera.com> for more details**
- **If you have any further questions, please feel free to contact us via <http://cloudera.com>**