



ESS 1000 – Big Data Essentials Slide Guide

Spring 2016 – Version 5.1.0

For use with the following courses:

ESS 1000 – Big Data Essentials (comprises ESS 100, ESS 101, ESS 102)

ESS 100 – Introduction to Big Data

ESS 101 – Apache Hadoop Essentials

ESS 102 – MapR Converged Data Platform Essentials

This Guide is protected under U.S. and international copyright laws, and is the exclusive property of MapR Technologies, Inc.

© 2016, MapR Technologies, Inc. All rights reserved. All other trademarks cited here are the property of their respective owners.





ESS 1000 – Big Data Essentials

Lesson 0: Get Started

Spring 2016 v5.1

Welcome to ESS 1000, Big Data Essentials.

Learning Goals



Learning Goals



- Define Key Concepts of Big Data
- Describe the Big Data Pipeline
- Describe Key Features of the Apache Hadoop Ecosystem
- Describe Key Features of the MapR Converged Data Platform

When you have finished this course, you will be able to:

- Define key concepts of big data
- Describe the big data pipeline
- Describe key features of the Apache Hadoop ecosystem
- Describe key features of the MapR Converged Data Platform

Prerequisites

- **Required:**
 - Internet access
 - Basic computer skills
- **Recommended:**
 - Basic data analysis skills, such as SQL
 - Basic programming skills, such as Java
 - Basic administrative skills, such as Linux

There are no labs or technical prerequisites for this course.

Internet access and basic computer skills are required to view online course materials and complete the online quizzes.

It is helpful if you are familiar with basic data analysis, programming, or computer administration concepts. It is recommended that you have basic familiarity with one or more of SQL, Java, or Linux.



Next Steps

Lesson 1: Introduction to Big Data

ESS100 – Big Data Essentials

Continue on to Lesson 1: Introduction to Big Data.



ESS 100 – Introduction to Big Data

Lesson 1: Introduction to Big Data

Spring 2016 v5.1

Welcome to ESS 100, Lesson 1: Introduction to Big Data.

Learning Goals



Learning Goals



- 1.1 Define Big Data
- 1.2 Summarize the History of Big Data Computing
- 1.3 Define Key Terms in Big Data Computing

When you have finished with this lesson, you will be able to:

- define big data,
- summarize the history of big data computing,
- and define key terms in big data computing.

Learning Goals



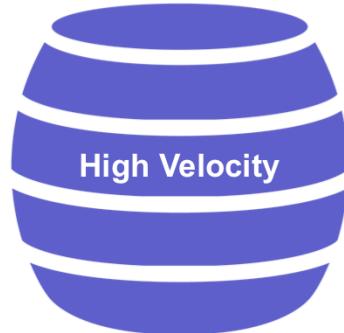
1.1 Define Big Data

- 1.2 Summarize the History of Big Data Computing
- 1.3 Define Key Terms in Big Data Computing

Let's start by defining big data.

What is "Big Data"?

- Data is information stored on a computer
- Big data is difficult to describe, store, or process
- Big data is data with one or more of the characteristics below:



Data includes numbers, text, images, audio, video, or any other kind of information you might store on your computer.

Big data is data which is characterized by one or more of the following characteristics: high volume, high variety, or high velocity.

Volume, velocity, and variety are sometimes called "the 3 V's of big data."

Volume: Amount

Reference: <https://open.nasa.gov/blog/2012/10/04/what-is-nasa-doing-with-big-data-today/>

MapR Academy

© 2016 MapR Technologies L1-6

Volume refers to the amount of data generated.

Traditional data analysis models typically require servers with large storage capacities and with massive computing power. This model does not scale well: to increase your processing power, you need to keep investing in what is often expensive proprietary hardware.

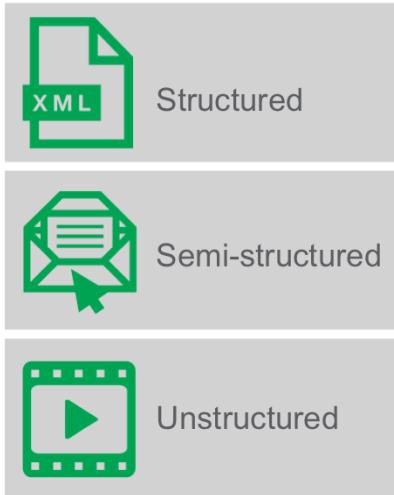
NASA is one of many companies collecting data at an unprecedented rate. At the end of 2014, NASA gathered approximately 1.73 gigabytes of data every few seconds, with the data collection rate growing exponentially.

This is a high volume of data. It can be difficult to store a high volume of data.

Variety: Type



WIKIPEDIA
The Free Encyclopedia



Variety refers to the types of data generated.

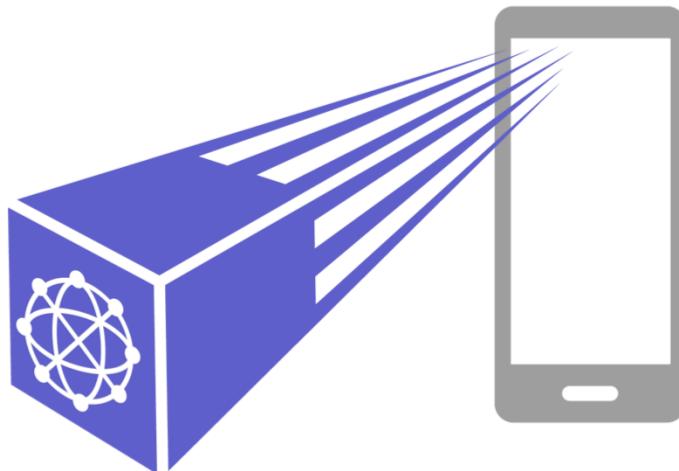
The traditional data model requires development of a schema to hold the data. In order to create a schema, you need to know the format of the data that will be collected.

Data might be structured like XML files, semi-structured like emails, or unstructured like video files.

For example, Wikipedia includes more than just text data: it contains hyperlinks, images, sound files, and many other data types. You may not know how every page is structured.

Having several different types of data, especially unstructured data, is high variety. It can be difficult to describe high-variety data.

Velocity: Speed



Velocity refers to the speed at which data is generated.

Traditional data analysis models were designed for batch processing: collect the entire set of data, then process it and get it into the database. With big data, you may want to perform real-time analysis of the data as it comes in, so you can react quickly to the information.

For example, networked sensors connected to the Internet of Things may generate thousands of data points per second. Unlike NASA or Wikipedia, whose data can be processed later, data from smartphones and other networked sensors must be processed in real time.

This is high velocity data. It can be difficult to process high-velocity data.

Knowledge Check



Knowledge Check



Office Supply Co. needs to manage online customer orders, which are coming in **every few seconds**. The order files themselves, however, are relatively **small** and regularly **structured**. This is a big data problem because:

- A. The orders have high variety, but not velocity or volume.
- B. The orders have high velocity, but not variety or volume.
- C. The orders have high volume, but not velocity or variety.
- D. This is not a big data problem, because it lacks all 3 V's.

Knowledge Check



Office Supply Co. needs to manage online customer orders, which are coming in every few seconds. The order files themselves, however, are relatively small and regularly structured. This is a big data problem because:

- A. The orders have high variety, but not velocity or volume.
- B. The orders have high velocity, but not variety or volume.**
- C. The orders have high volume, but not velocity or variety.
- D. This is not a big data problem, because it lacks all 3 V's.

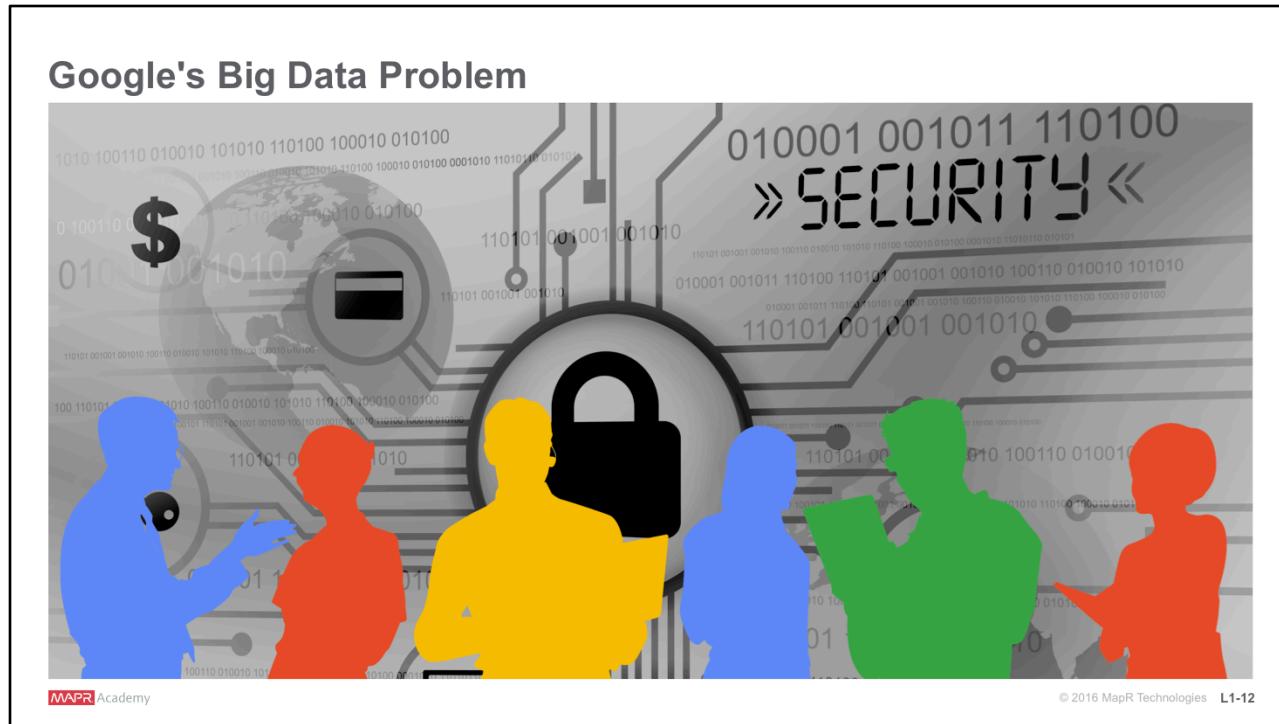
Answer: A

The orders are coming in very fast; this is an example of high velocity.

Because the order files are small, there is not high volume.

Because the order files are regularly structured, there is not high variety.

A big data problem can be characterized by one or more of the 3 V's. High velocity is sufficient for this to be a big data problem.



Google came in as the 19th web search engine in 1996. To try to gain an advantage in their searches, Google wanted to create a catalog of the entire Internet. In order to be successful, they had to meet the challenges presented by the 3 V's in an innovative way.

Google's Big Data Problem

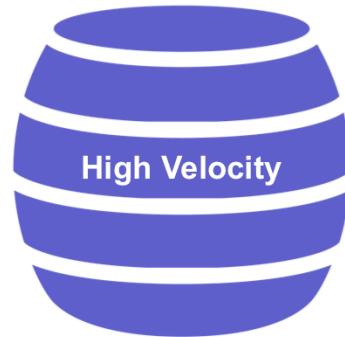
- Every word on every web page contributes to a **high volume** of data



First, there's the sheer volume of the data they wanted to collect. Imagine how much data we're talking about here. Cataloging every word on every page across the entire Internet. This is a vast amount of data that would cripple any standard database table.

Google's Big Data Problem

- Every word on every web page contributes to a **high volume** of data
- New websites with dynamic updates create **high-velocity** data



Next, there's the velocity at which this data is collected. With billions of pages across the Internet, millions of new changes are made every day. Keeping up with this level of content requires the ability to ingest and process data at a staggering rate.

Google's Big Data Problem

- Every word on every web page contributes to a **high volume** of data
- New websites with dynamic updates create **high-velocity** data
- An unstructured combination of text, images, and movies is **high-variety** data



In addition, there's the variety of data that is being collected. Databases expect a certain kind of data, organized in a certain way. The content of images, sound bites, and movies can't be cataloged this way. Even if a site does have easily organized content, there's no way of knowing how the data from one site compares to another. A typical address form can have different fields organized in different ways from one site to the next. All of these different kinds of data from all of these different sites cannot be stored in a single, traditional database that describes every piece of data.

Now we've seen how the 3 V's and presented challenges to Google. In the next section, we'll see how Google solved these problems.

Class Discussion



Class Discussion



Describe your own big data problem.

Think about which of the 3 V's of big data apply to your situation.

What features make your data difficult to describe, store, or process?

Describe your own big data problem.

Think about which of the 3 V's of Big Data apply to your situation.

What features make your data difficult to describe, store, or process?

<https://community.mapr.com/thread/10154>

Learning Goals



Learning Goals



- 1.1 Define Big Data
- 1.2 Summarize the History of Big Data Computing**
- 1.3 Define Key Terms in Big Data Computing

Next, we'll explore the history of big data computing.



Google certainly had a big challenge on their hands, but they didn't let these 3 V's stop them. They knew that they couldn't take this on with traditional, relational databases. They needed something new.

Google Solution

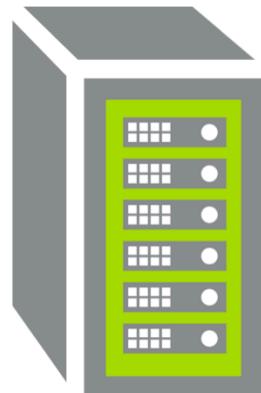
Google File System, Bigtable, and MapReduce work together in a group of interconnected, inexpensive computers



They tackled the big data problem with a trio of components: the Google File System, Bigtable, and MapReduce. This trio of components work together in a group of interconnected, inexpensive computers. This was revolutionary, since previous big data solutions required expensive, proprietary hardware.

Google Solution: Cluster

Cluster: a collection of computers



A collection of computers is known as a cluster.

Google Solution: Node

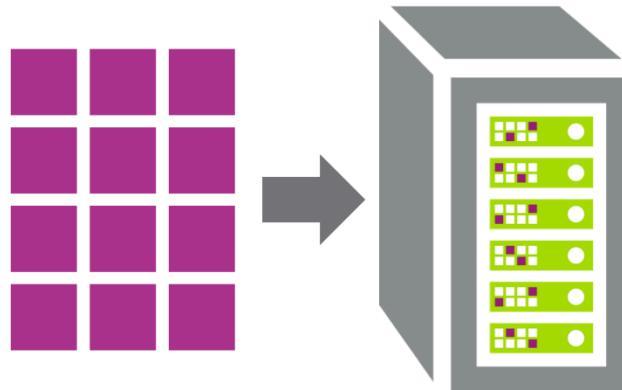
Node: individual machine in a cluster



Each individual machine is a node. Rather than spend a lot of money on the biggest, fastest machines they could find, Google designed clusters that were made up of commodity hardware that can be easily replaced when one of them breaks down.

Google File System (GFS)

GFS breaks up files into chunks and distributes them across different nodes in a cluster



First, Google created the Google File System or GFS. GFS breaks up files into chunks and distributes them across the nodes of a cluster. These chunks are replicated onto different nodes, to protect the data from the loss of a node.

Bigtable

	A	B	C	D
1	Name	Email	Purchase	Time
2	Ahi	tahi@my-co.com	12	2010-03-16 19:06:01
3	Becker	becker@mapr.com	6	2015-05-05 03:21:12
4	Carlisle	carlisle@tcs.net	12	2014-12-12 09:35:03
5	Carlisle	carlisle@tcs.net	12	2015-01-25 11:30:22
6	Cavalero	dancav@abc123.com	6	2016-04-16 07:55:52

Bigtable is a database system that uses GFS to store and retrieve data. Despite its name, Bigtable is not just a very large table.

Bigtable

	A	B	C	D
1	Name	Email	Purchase	Time
2	Ahi	tahi@my-co.com	12	2010-03-16 19:06:01
3	Becker	becker@mapr.com	6	2015-05-05 03:21:12
4	Carlisle	carlisle@tcs.net	12	2014-12-12 09:35:03
5	Carlisle	carlisle@tcs.net	12	2015-01-25 11:30:22
6	Cavalero	dancav@abc123.com	6	2016-04-16 07:55:52

A Bigtable maps the data it stores using a row key,

Bigtable

	A	B	C	D
1	Name	Email	Purchase	Time
2	Ahi	tahi@my-co.com	12	2010-03-16 19:06:01
3	Becker	becker@mapr.com	6	2015-05-05 03:21:12
4	Carlisle	carlisle@tcs.net	12	2014-12-12 09:35:03
5	Carlisle	carlisle@tcs.net	12	2015-01-25 11:30:22
6	Cavalero	dancav@abc123.com	6	2016-04-16 07:55:52

a column key,

Bigtable

	A	B	C	D
1	Name	Email	Purchase	Time
2	Ahi	tahi@my-co.com	12	2010-03-16 19:06:01
3	Becker	becker@mapr.com	6	2015-05-05 03:21:12
4	Carlisle	carlisle@tcs.net	12	2014-12-12 09:35:03
5	Carlisle	carlisle@tcs.net	12	2015-01-25 11:30:22
6	Cavalero	dancav@abc123.com	6	2016-04-16 07:55:52

and a timestamp.

Bigtable

	A	B	C	D
1	Name	Email	Purchase	Time
2	Ahi	tahi@my-co.com	12	2010-03-16 19:06:01
3	Becker	becker@mapr.com	6	2015-05-05 03:21:12
4	Carlisle	carlisle@tcs.net	12	2014-12-12 09:35:03
5	Carlisle	carlisle@tcs.net	12	2015-01-25 11:30:22
6	Cavalero	dancav@abc123.com	6	2016-04-16 07:55:52

This allows the same information to be captured over time without overwriting any pre-existing entries.

Bigtable

	A	B	C	D
1	Name	Email	Purchase	Time
2	Ahi	tahi@my-co.com	12	2010-03-16 19:06:01
3	Becker	becker@mapr.com	6	2015-05-05 03:21:12
4	Carlisle	carlisle@tcs.net	12	2014-12-12 09:35:03
5	Carlisle	carlisle@tcs.net	12	2015-01-25 11:30:22
6	Cavalero	dancav@abc123.com	6	2016-04-16 07:55:52

The rows are then partitioned into subtables called tablets, which are distributed across the cluster.

Bigtable was designed to handle enormous amounts of data, with the ability to add new nodes to the cluster without the need to reconfigure any of the existing files.

MapReduce

- Parallel processing paradigm
- Two steps: `map()` and `reduce()`

As the third piece of the puzzle, a parallel processing paradigm called MapReduce was leveraged to process the data stored in GFS. The name, MapReduce, is taken from the names of the two steps in the process.

History of MapReduce

Square Function

- Applies same logic to each value, one value at a time
- Emits result for each value

```
(map square '(1 2 3 4))=(1 4 9 16)
```

Addition Function

- Applies same logic to all the values taken together

```
(reduce + '(1 4 9 16))=30
```

While the MapReduce process has been made famous by Apache Hadoop, it is hardly a new idea. In fact, many common tasks like sorting and folding laundry can be thought of as examples of the MapReduce process.

The names map and reduce can be seen used in programming at least as far back as the 70s in Lisp. In this example we see how Lisp uses the MapReduce model, first using a map of the square function on an input list of 1 to 4. The square function, since it is mapped, will apply to each of the inputs and produce a single output per input, in this case 1, 4, 9, and 16. The addition function reduces the list and produces a single output of 30, which is the sum of the inputs.

History of MapReduce

The diagram illustrates the evolution of search engines over time. At the top, five boxes represent search engines from 1994 to 1996: Infoseek (Jan), Webcrawler (Apr), Excite (Apr), Yahoo! Search (Dec), and Altavista (May). Below these, a large box for 1997 (Sept) contains the Google logo. The timeline is indicated by a horizontal bar at the bottom labeled "1997 (SEPT)".

1994 (JAN) **infoseek**
proof of intelligent life on the net...
excite

1994 (APR) **WEBCRAWLER**
LIGHTNING FAST WEB SEARCH
YAHOO! SEARCH

1995 (DEC) **altavista**

1996 (MAY) **Inktomi**

1997 (APR) **Ask**

1997 (SEPT) **Google**

MAPR Academy © 2016 MapR Technologies L1-33

Google then leveraged the power of MapReduce to dominate a crowded search engine market. They came in as the 19th web search engine, and within just a couple of years, established their dominance, which they still enjoy today.

Google used MapReduce in a variety of ways to improve the web search experience. It was used to help index page content, and as the power behind their PageRank algorithm, it was used to determine the position of pages in a resulting search.

History of MapReduce

```
map(String key, String value):
    //key: document name
    //value: document contents
    for each word w in value:
        EmitIntermediate(w,"1");

reduce(String key, Iterator values):
    //key: a word
    //values: a list of counts
    int result = 0;
    for each v in values:
        result +=ParseInt(v);
        Emit(AsString(result));
```

Reference: MapReduce: Simplified Data Processing on Large Clusters Jeffrey Dean (jeff@Google.com) & Sanjay Ghemawat (sanjay@google.com), Google, Inc.

This example shows the MapReduce algorithm that Google used to perform word counts on web pages. The map method takes as input a key and a value, where the key represents the name of a document and the value is the contents of the document. The map method loops through each word in the document, and returns a 2-tuple consisting of the word and a 1.

The reduce method takes as input a key and a list of values, where the key represents a word. The list of values is the list of counts for that word. In this example, the value is 1. The reduce method loops through the counts and sums them. When the loop is done, the reduce method returns a 2-tuple consisting of the word and its total count.

Apache Hadoop



Over a span of a couple of years, Google Labs released papers describing the parts of their big data solution. From these, Doug Cutting began developing a project at Yahoo!, which was later open sourced into the Apache Foundation project called Hadoop, named after the toy elephant of Mr. Cutting's son.

Major Components of Hadoop



MapReduce	MapReduce	MapReduce
Bigtable	HBase	HBase/MapR-DB
GFS	HDFS	HDFS/MapR-FS

Apache Hadoop is comprised of three components, similar to those in the original Google solution, that also work on a collection of commodity hardware nodes.

The bottom layer in Hadoop is HDFS, or Hadoop Distributed File System. HDFS breaks up files into chunk sizes similar to GFS, and distributes them across the nodes of the cluster. Like Google Bigtable, data can be stored in a tabular format in HDFS using an application called HBase. Hadoop also leverages MapReduce to process data stored in HDFS.

Major Components of the MapR Converged Data Platform



MapReduce	MapReduce	MapReduce
Bigtable	HBase	HBase/MapR-DB
GFS	HDFS	HDFS/MapR-FS

MapR then further advanced the concepts in Hadoop and created a faster, more stable version of these three pieces. The MapR Converged Data Platform performs all of the same functions as the original Google solution to the big data problem, and is completely compatible with Hadoop and all of the applications associated with the Hadoop ecosystem.

The advances introduced by MapR have created a faster, more reliable, enterprise converged data platform that is designed for the rigorous standards of business, small and large alike.

Knowledge Check



Knowledge Check



How was Google able to solve its big data problem? Check all that apply.

- A. By organizing data into columns and rows, using Bigtable
- B. By storing data on a Hadoop Distributed File System
- C. By splitting data into manageable chunks, using MapReduce
- D. By cataloging only the plain text data on the Internet

Knowledge Check



How was Google able to solve its big data problem? Check all that apply.

- A. By organizing data into columns and rows, using Bigtable**
- B. By storing data on a Hadoop Distributed File System
- C. By splitting data into manageable chunks, using MapReduce**
- D. By cataloging only the plain text data on the Internet

Answer: A & C

Learning Goals



Learning Goals



- 1.1 Define Big Data
- 1.2 Summarize the History of Big Data Computing
- 1.3 Define Key Terms in Big Data Computing**

Finally, we'll learn some of the key terms commonly used in big data computing.

How Big is Big Data?



Megabyte:
1,000 kilobytes



Gigabyte:
1,000 megabytes

Most people work with megabytes and gigabytes.

A small image or audio file might contain about one megabyte of information. This is also about the amount of information a 3 1/2-inch floppy disks held in the early 1990s.

An hour-long music CD contains about one gigabyte of information. Modern smartphones typically hold several gigabytes of storage.

How Big is Big Data?



Terabyte:
1,000 gigabytes

A terabyte is about 1000 gigabytes. Some personal computers may have this much storage. The terabyte is the level where big data problems typically start to emerge. The size of a Hadoop cluster might be measured in terabytes.

How Big is Big Data?



Petabyte: 1,000 terabytes

Exabyte: 1,000 petabytes

A petabyte is 1000 terabytes, and an exabyte is 1000 petabytes. Here is where big data truly shines. Many Hadoop clusters will contain petabytes, or even exabytes, of data.

What Types of Data Are There?

This is a string of characters

1
12345678
12.34

01/01/01
12:30:01
12.34



Strings,
characters

Integers,
floats

Dates,
timestamps,
money

Images,
sounds,
videos

In addition to the size of your data, it is important to consider the types of data you are working with when planning a big data project.

Most people who have worked with programming languages are already familiar with simple data types, such as strings, characters, integers, and floats.

What Types of Data Are There?

This is a string of characters

1
12345678
12.34

01/01/01
12:30:01
12.34



Strings,
characters

Integers,
floats

Dates,
timestamps,
money

Images,
sounds,
videos

Other numeric data types include dates, timestamps, and money. These are special data types, since they often function differently than normal numbers and must be formatted in certain ways.

What Types of Data Are There?

This is a string of characters

1
12345678
12.34

01/01/01
12:30:01
12.34



Strings,
characters

Integers,
floats

Dates,
timestamps,
money

Images,
sounds,
videos

More complex data types also exist. For example, you may have images, sounds, or a combination of images and sounds in a video file. These data types are unstructured, and may be difficult to process using traditional database tools.

What Types of Files Are There?



Simple
Structured



Simple
Semi-Structured



Complex
Structured



Complex
Semi-Structured

In addition to data types, you should consider how your data is stored. There are many file types, but a typical database may contain simple, structured files like spreadsheets saved as a CSV file. There are also more complex structured files, such as parquet files. Data can also be organized in simple, semi-structured files like HBase tables. Finally, some data is stored in complex, semi-structured files like JSON.

Knowledge Check



Knowledge Check



How large is the typical Hadoop cluster?

- A. Several gigabytes
- B. Several terabytes
- C. Several petabytes
- D. Several exabytes

Knowledge Check



How large is the typical Hadoop cluster?

- A. Several gigabytes
- B. Several terabytes
- C. Several petabytes
- D. Several exabytes

Answer: any of these. Volume typically becomes a big data problem after several petabytes or exabytes. However, velocity or variety means you may have a big data problem without these large volumes.

How is Data Organized?



Physical Storage:
Racks and nodes



Logical Storage:
Partitions and topologies

Once you know how big your data is, and what type your data is, the next step is to figure out how it is organized.

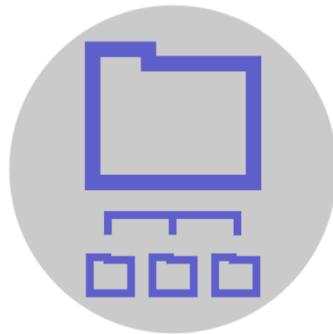
It is important to keep in mind the difference between physical storage and logical storage.

Physical storage refers to the actual real-world location of your data. This might be several racks of servers in a data warehouse. To protect your data, you want to replicate it to nodes on different racks. That way, if a single rack fails, there are copies of your data still available on other racks.

How is Data Organized?



Physical Storage:
Racks and servers.



Logical Storage:
Partitions and nodes.

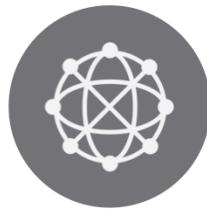
Logical storage refers to how your operating system organizes your data. This includes the topology of your file system, and how data is partitioned across nodes.

When choosing how to organize your data, keep in mind both how it is logically stored and physically stored.

What Common Terms Are Used in Big Data?



ETL:
Extract, Transform, Load



IoT:
Internet-of-Things



SQL:
Structured Query Language

Now that you're in the big data world, you'll encounter many terms and acronyms. We'll discuss a few of the most common ones.

Extract, transform, load, or ETL, refers to the process of getting data ready to use. Once data is created, it must go through the ETL process before it can be analyzed.

What Common Terms Are Used in Big Data?



ETL:
Extract, Transform, Load



IoT:
Internet-of-Things



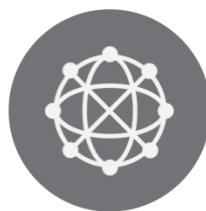
SQL:
Structured Query Language

The Internet-of-Things, or IoT, is a collection of networked sensors. These include personal sensors on gadgets like smartphones and fitness monitors, as well as sensors in smart cars, factory production lines, and credit card readers. IoT is a major source of big data.

What Common Terms Are Used in Big Data?



ETL:
Extract, Transform, Load



IoT:
Internet-of-Things



SQL:
Structured Query Language

Structured Query Language, or SQL (commonly pronounced *sequel*) is a language commonly used by data analysts to explore data.

In addition to SQL, there is also NoSQL, which stands for not-only SQL.

To learn more common terms you'll encounter in the world of big data, download the glossary attached to this course on learn.mapr.com.

Knowledge Check



Knowledge Check



Match the acronym with its definition.

SQL

Network of data-generating sensors

IoT

Process for getting data ready for analysis

ETL

Programming language used by analysts

SQL :: a programming language used by analysts

IoT :: a network of data-generating sensors

ETL :: a process for getting data ready for analysis



Next Steps

Lesson 2: The Big Data Pipeline

ESS 100 – Introduction to Big Data

Congratulations! You have completed ESS 100, lesson 1. Continue on to lesson 2, to learn about The Big Data Pipeline.



ESS 100 – Introduction to Big Data

Lesson 2: The Big Data Pipeline

Spring 2016 v5.1

Welcome to ESS 100, Lesson 2: The Big Data Pipeline.

Learning Goals



Learning Goals



- 2.1 Organize the Steps in the Data Pipeline
- 2.2 Explain the Role of Administrators
- 2.3 Explain the Role of Developers
- 2.4 Explain the Role of Data Analysts

When you have finished with this lesson, you will be able to:

- organize the steps in the data pipeline,
- explain the role of administrators,
- explain the role of developers,
- and explain the role of data analysts.

Learning Goals



2.1 Organize the Steps in the Data Pipeline

- 2.2 Explain the Role of Administrators
- 2.3 Explain the Role of Developers
- 2.4 Explain the Role of Data Analysts

Let's start by organizing steps in the data pipeline.

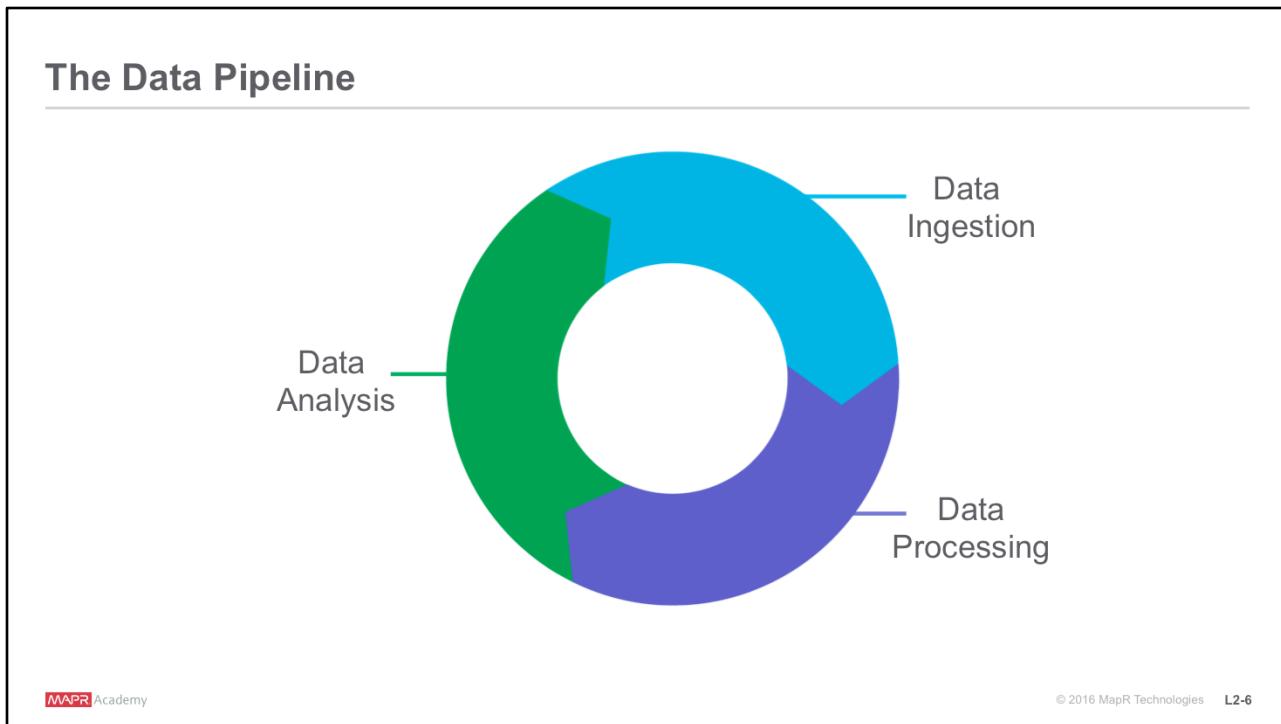
The Data Pipeline



In general, a basic data pipeline looks something like this diagram. Data from your sources must be ingested, processed, stored, and analyzed.

There are many tools, processes, and people involved in each step of the data pipeline.

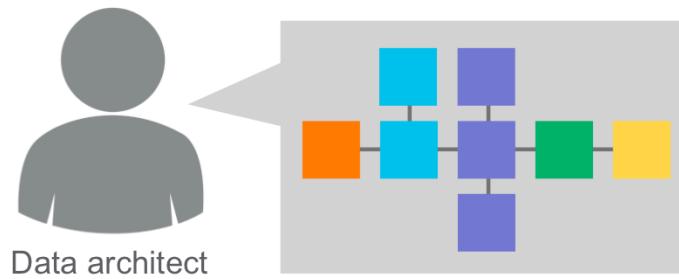
In this lesson, we'll cover how to create a data pipeline, including hiring the right people and deploying the right tools for your big data project.



It is also important to keep in mind that the data pipeline is not necessarily linear. The output of an analysis might be the input for your next project.

Define Your Big Data Project

Define the project pipeline



The first step in the data pipeline is to define your project. A data architect can help define your project and map out the data pipeline. Try to clearly define your goals before you begin.

Define Your Big Data Project

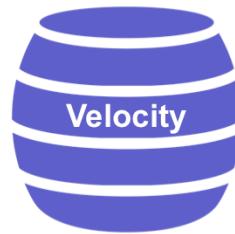
Why are you collecting data?



How much data is collected?



What kind of data is collected?



How is your data collected?

Consider how and why you are collecting data.

Think about the 3 V's of big data before you start collecting data.

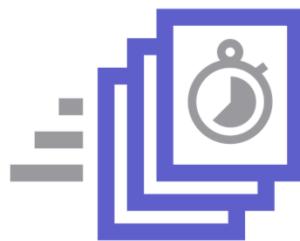
What is the velocity of your data? Can it be processed in batches, or do you need to stream it in real time?

What is the variety of your data? Is it structured or unstructured? Will you need to normalize it in some way?

What is the volume of your data? How much storage and bandwidth should you plan for?

Data Collection: The 3 Vs

Velocity:
Streaming vs. batch



Analyze historical
trends, no
immediate action

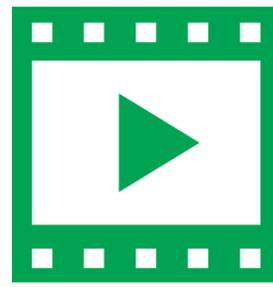
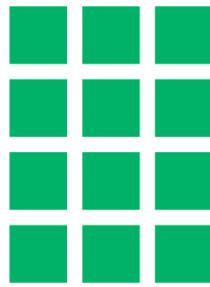
Data can come from many different sources. Networked sensors connected to the Internet of Things might provide streaming data. Streaming data is dynamic and is often produced and analyzed in near real time.

On the other hand, repositories of historical data available for download from a central server might provide batch data. Batch data is great for analyzing historical trends or large quantities of data that do not require immediate action.

Streaming data is generally high-velocity. If you do not need real-time results or lack network bandwidth to handle high-velocity data, consider using batch processes instead.

Data Collection: The 3 Vs

Variety:
Structured vs. unstructured



The next thing you should consider is the type of data you are collecting. Is it structured, semi-structured, or unstructured? What file format is it saved as? Is it a JSON file, plain text, numeric data, or something else?

Whether you need to convert data types or clean your data, the type of data can affect how you plan your ETL process.

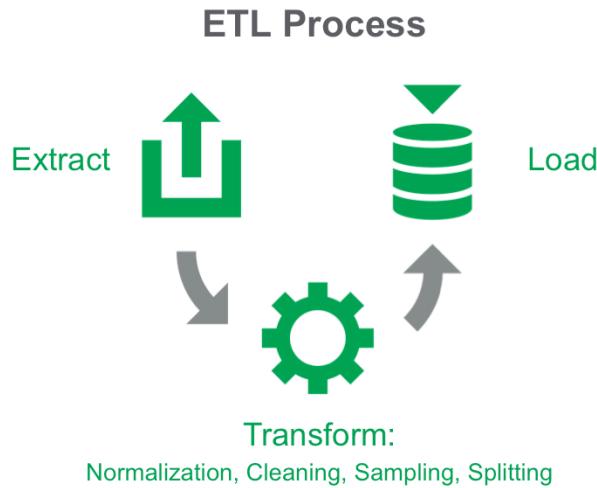
Data Collection: The 3 Vs

Volume:
Size of data

$$\sim 4 \times \text{Data} = \text{Storage Space}$$

Finally, consider the size of your data. Will the size of your data grow over time, or are you storing historical data long term? Keep in mind that replicating your data for fault tolerance and keeping track of metadata takes up additional space. Therefore, you may need about four times as much storage space as you have data. This much space will accommodate three replicas for fault tolerance, as well as space for your operating system and metadata files.

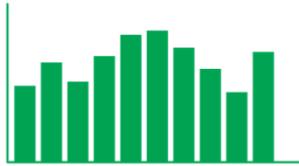
Data Processing



Once you have a big data project and have collected some data, it must be processed. This is often referred to as Extract, Transform, and Load or ETL.

Depending on your project, the ETL process may include normalization or cleaning of the data. It may also involve sampling the data, such as splitting data into test and training groups for machine learning.

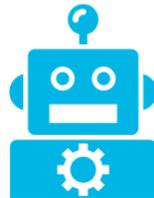
Data Analysis



Graphs, charts



Decisions, alerts



Machine learning

Finally, once your data has been collected and processed, you can begin gaining insights from it. This might include graphs and charts you use for presentations, automated decisions and alerts built in to your applications, or machine learning algorithms.

Knowledge Check



Knowledge Check



What does ETL stand for?

- A. Estimated Time to Load
- B. Escape, Tabulate, Log
- C. Extract, Transform, Load
- D. Explain Table Logs

Knowledge Check



What does ETL stand for?

- A. Estimated Time to Load
- B. Escape, Tabulate, Log
- C. Extract, Transform, Load**
- D. Explain Table Logs

Answer: C

Learning Goals



Learning Goals



2.1 Organize the Steps in the Data Pipeline

2.2 Explain the Role of Administrators

2.3 Explain the Role of Developers

2.4 Explain the Role of Data Analysts

Next, we'll learn about administrators.

Administrators

Install and maintain hardware and software



Administrators are responsible for many aspects of installing and maintaining the hardware and software used in the big data pipeline. This includes preparing nodes, adding and removing users, configuring security, testing performance against benchmarks, upgrading software, disaster recovery planning, and day-to-day problem solving.

The Data Pipeline



Administrators set up the servers where you ingest and store your data. They also install the software needed to process and analyze your data.

Administrator Skills

What makes a good administrator?



What makes a good administrator?

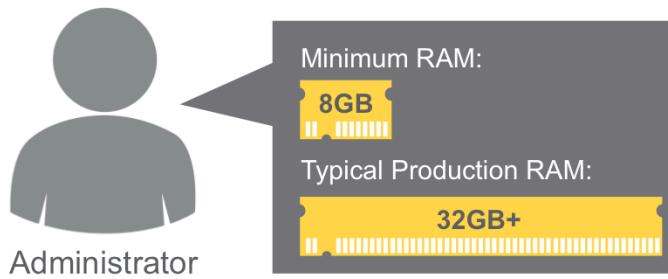
Administrators are generally comfortable working either with a command line interface or with a graphical user interface, and often must use both.

Administrators are also familiar with several different operating systems, including a variety of flavors of Linux like Red Hat, Ubuntu, CentOS, or SUSE.

They also have a breadth of knowledge of the kinds of hardware and software used by people in their organizations.

Manage Physical Storage

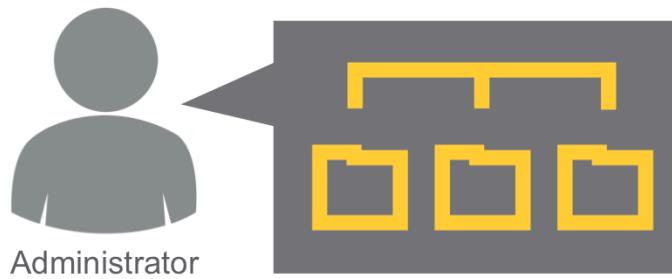
Choose and install hardware



Administrators manage physical storage. They choose and install hardware, and distribute your data across different racks or data centers. For example, an administrator might advise you that the minimum RAM required for a Hadoop cluster is 8 gigabytes, but that typical production will require 32 gigabytes or more.

Manage Logical Storage

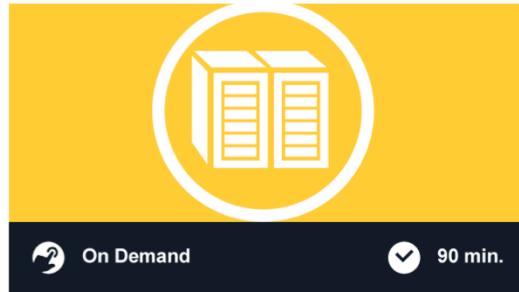
Organize data in topologies



Administrators also manage logical storage. They organize your data into topologies. For example, an administrator might advise you to keep data in the Marketing topology separate from the Engineering topology, so that users who can access one cannot access the other.

Administrator Courses on MapR Academy

- ADM 200 – Install a MapR Cluster
- ADM 201 – Configure a MapR Cluster
- ADM 202 – Data Access and Protection
- ADM 203 – Cluster Maintenance



If you are interested in learning more about how to administer a big data platform, consider taking courses from the ADM series on learn.mapr.com.

Knowledge Check



Knowledge Check



Which of the following are responsibilities of administrators? (Select all that apply)

- A. Normalizing data for a machine learning algorithm
- B. Installing a new program for everyone in an organization
- C. Creating infographics or other presentations
- D. Implementing security protocols for a database

Knowledge Check



Which of the following are responsibilities of administrators? (Select all that apply)

- A. Normalizing data for a machine learning algorithm
- B. **Installing a new program for everyone in an organization**
- C. Creating infographics or other presentations
- D. **Implementing security protocols for a database**

Learning Goals



Learning Goals

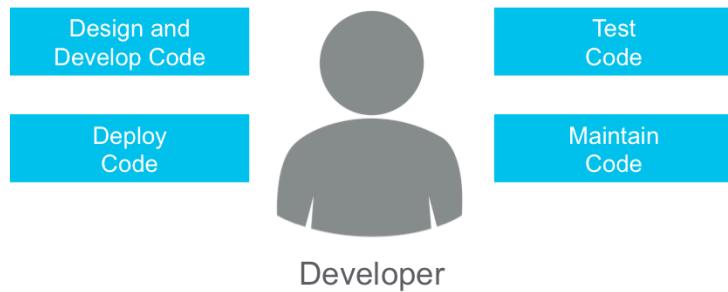


- 2.1 Organize the Steps in the Data Pipeline
- 2.2 Explain the Role of Administrators
- 2.3 Explain the Role of Developers**
- 2.4 Explain the Role of Data Analysts

Next, we'll learn about developers.

Developers

Develop programs to ingest and process data



Developers are responsible for developing programs to ingest and process data. This includes designing, developing, deploying, testing, and maintaining code, typically in Java, Python, or Scala.

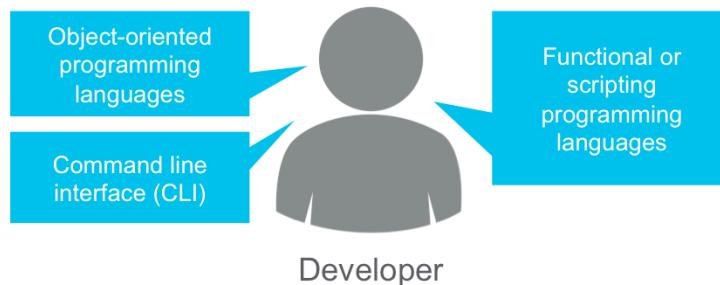
The Data Pipeline



Developers write programs that help with the ETL process. They convert data into formats which can be stored on your cluster and analyzed for results.

Developer Skills

What makes a good developer?



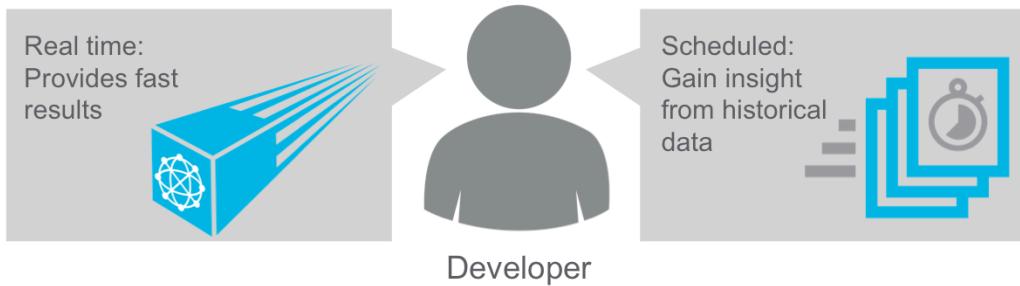
What makes a good developer?

Developers are often proficient in a variety of programming languages. They might know one or more object-oriented programming languages, such as Java or C++. They might also know several functional or scripting languages, such as Python, JavaScript, or Scala.

Like administrators, developers are usually comfortable using command line tools.

Ingest Data

Write programs that move data from source to cluster

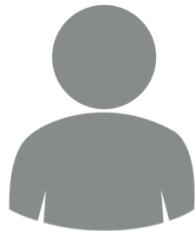


One thing developers do is write programs that ingest data. Data ingestion involves moving data from the source to the cluster, where it is stored and processed.

There are two major types of data ingestion: streaming and batch. Streamed data happens in real time. It can be resource intensive, but provides results fast, which is great for traffic and weather updates or fraud detection. Batched data, on the other hand, can be scheduled. You can run batches at night or on weekends, when less resources are being actively used. Batch data is great for gaining insight from historical data.

Process Data

Process data for easy use by data analysts



Developer

Convert:

Example: Split data into different samples for machine learning

Clean:

Example: Remove material from data sets

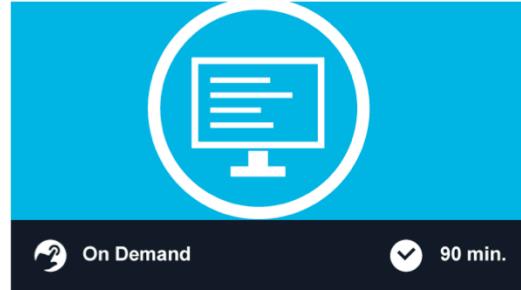
Normalize:

Example: Write script converting all strings to lowercase

Developers also process data, so that it is easier for data analysts to work with. This includes cleaning data, for example by removing unnecessary material from data sets. Scripts can also be written to normalize data, such as converting all strings to lowercase. Developers might also help split data into different samples for machine learning, or convert it into a particular structure for querying.

Developer Courses on MapR Academy

- DEV 301 – Developing Hadoop Applications
- DEV 320 – Apache HBase Data Model
- DEV 350 – MapR Streams Essentials
- DEV 360 – Apache Spark Essentials



If you are interested in learning more about how to build your own big data applications, consider taking some of these courses from the DEV series on learn.mapr.com.

Knowledge Check



Knowledge Check



Which of the following are responsibilities of developers? (Select all that apply)

- A. Building and testing a new ingestion application
- B. Installing and configuring a database
- C. Querying data to find correlations
- D. Maintaining legacy code

Knowledge Check



Which of the following are responsibilities of developers? (Select all that apply)

- A. **Building and testing a new ingestion application**
- B. Installing and configuring a database
- C. Querying data to find correlations
- D. **Maintaining legacy code**

Learning Goals



Learning Goals

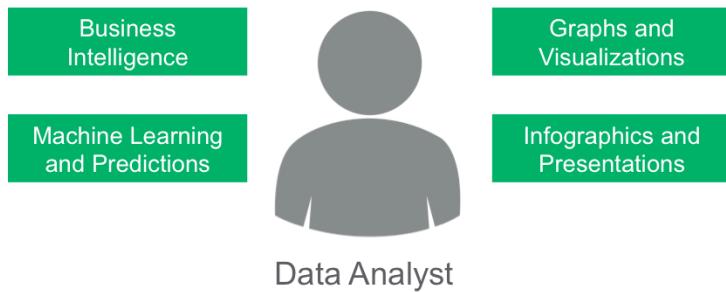


- 2.1 Organize the Steps in the Data Pipeline
- 2.2 Explain the Role of Administrators
- 2.3 Explain the Role of Developers
- 2.4 Explain the Role of Data Analysts**

Finally, we'll learn about data analysts.

Data Analysts

Responsible for analyzing data



Data analysts are responsible for analyzing data. This includes data mining, extraction, normalization, filtering, aggregation, querying, interpreting, graphing, and making predictions. They provide companies with business intelligence, as well as use visualization tools to make infographics and presentations that summarize their findings.

The Data Pipeline



Data analysts are generally involved in the final step of the big data pipeline.

Data Analyst Skills

What makes a good data analyst?



What makes a good data analyst?

Data analysts are often proficient in a few functional or scripting languages like Python, R, or MySQL. They usually have a strong background in statistics and are experts in a particular field, such as finance, medicine, or natural language processing. They may also be able to make infographics and presentations using office software like Microsoft PowerPoint or visualization tools like Tableau.

Analyze Data

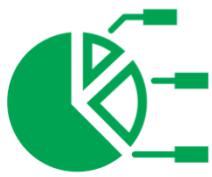
Query and analyze data to find interesting insights



Data analysts query and analyze data to find interesting insights. They might correlate several variables, like the release of a new product with social media sentiment. They can also use historical data to predict future trends. Exploring the data often requires skills like writing SQL queries while analysis requires advanced knowledge of statistical models.

Present Data

Present data in accessible ways



Infographics



Written Reports



Statistical
Significance



Action Items and
Insights

In addition to exploring data and creating statistical models, data analysts often must present their data in accessible ways. This may include an infographic, a written report, an explanation of statistical significance, or a list of action items and insights for a company based on their interpretation of the data.

Data Analyst Courses on MapR Academy

- DA 410 – Apache Drill Essentials
- DA 440 – Apache Hive Essentials
- DA 450 – Apache Pig Essentials



If you are interested in learning more about how extract insights from big data, consider taking courses from the Data Analyst series on learn.mapr.com.

Knowledge Check



Knowledge Check



Which of the following are responsibilities of analysts? (Select all that apply)

- A. Writing a script to automate an ETL process
- B. Predicting user behavior using machine learning
- C. Presenting complex data using a pie chart
- D. Giving security permissions to a new user

Knowledge Check



Which of the following are responsibilities of analysts? (Select all that apply)

- A. Writing a script to automate an ETL process
- B. **Predicting user behavior using machine learning**
- C. **Presenting complex data using a pie chart**
- D. Giving security permissions to a new user



Next Steps

Lesson 3: Core Elements of Apache Hadoop

ESS 101 – Core Elements of Apache Hadoop

Congratulations! You have completed ESS 100, lesson 2. Continue on to ESS 101, lesson 3, to learn about Apache Hadoop.



MAPR Academy

ESS 101 – Apache Hadoop Essentials

Lesson 3: Core Elements of Apache Hadoop

Spring 2016 v5.1

Welcome to ESS 101, Lesson 3: Core Elements of Apache Hadoop.

Learning Goals



Learning Goals



- 3.1 Compare and Contrast Local and Distributed File Systems
- 3.2 Explain Data Management in the Hadoop File System
- 3.3 Summarize the MapReduce Algorithm

When you have finished with this lesson, you will be able to:

- compare and contrast local and distributed file systems,
- explain data management in the Hadoop file system,
- and summarize the MapReduce algorithm.

Learning Goals



- 3.1 Compare and Contrast Local and Distributed File Systems**
- 3.2 Explain Data Management in the Hadoop File System
- 3.3 Summarize the MapReduce Algorithm

Let's start by learning about local and distributed file systems.

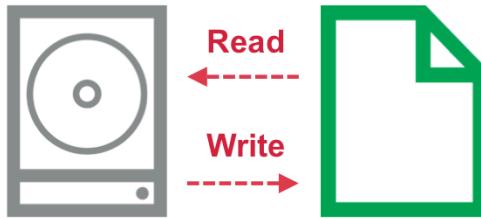
ReadWrite File Systems



You're watching this training session right now on a web browser [or PowerPoint presentation]. A web browser [or PowerPoint presentation] is a file. It's an application, or an executable file, but it is still a file that is stored on the hard drive of the computer that you are using.

ReadWrite File Systems

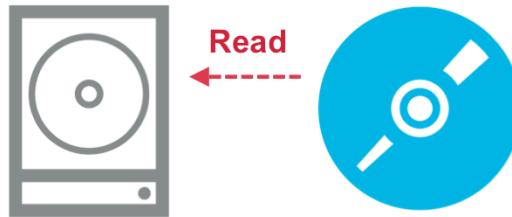
Examples: Hierarchical File System (HFS),
New Technology File System (NTFS)



The files on your local hard drive are stored using a hard drive file system, such as HFS or NTFS. Most of us have no idea how these file systems work, but we don't have to. We just need to know that we can open files, create files, save files, modify files, and delete files. Since you can both read existing files, as well as create new files or write to existing files, your local hard drive file system is considered a read/write file system.

Read-Only File Systems

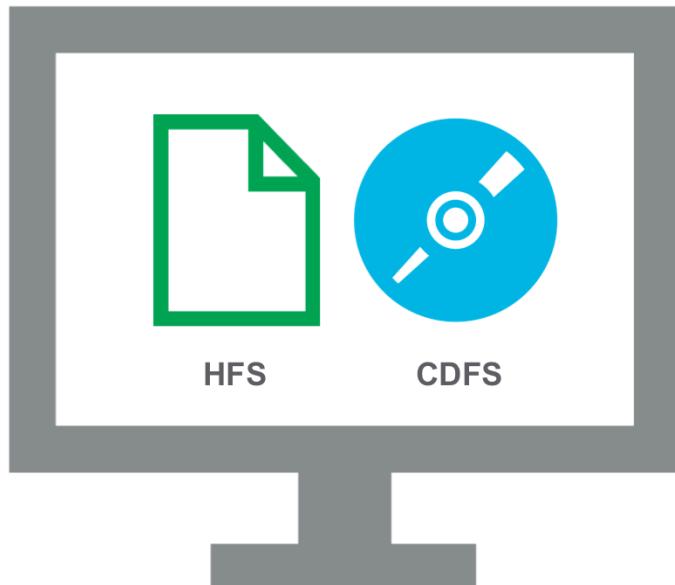
Compact Disc File System (CDFS)



When you pop in a CD-ROM or Blu-ray disc, you can access the files in the disc's file system, such as CDFS. CDFS uses the same utilities as those in your hard drive. This set of commonly used file system utilities are part of the POSIX standards.

As with HFS or NTFS, you don't have to know how CDFS works to use the files stored on the CD or Blu-ray. When working with these files, however, you cannot modify files, create new ones, nor delete them. You can only read from files that are stored on the disc. Therefore, CDFS is considered a read-only file system.

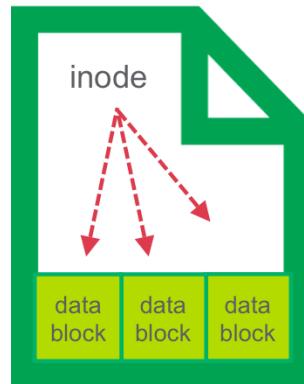
Local File Systems



HFS and CDFS are both local to your laptop, and we call them local file systems.

Local File Systems

- **inode:** stores metadata and pointers to the data blocks the file is stored in
- **data blocks:** store the actual contents of the file



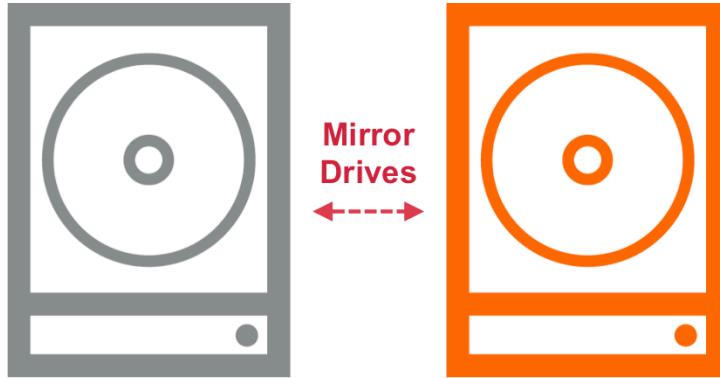
Each file in a file system is comprised of an inode and a set of data blocks. The inode stores metadata about the file, such as the file type, permissions, owner, file name, and the time it was last modified. The inode also stores pointers to the data blocks the file is stored in. The data blocks for a file store the actual contents of the file.

Common Problems of Local File Systems: HD Failure



What happens when the hard drive on your laptop crashes? You lose all of the files on the hard drive.

Common Problems of Local File Systems: HD Failure



To prevent data loss when a drive crashes, you can install a second local drive and mirror the two. That way, if one disk fails, you have an exact replica of the entire file system on the other drive. All of your music, photos, and working files are available as if nothing happened. You can then go replace the failed drive at your convenience.

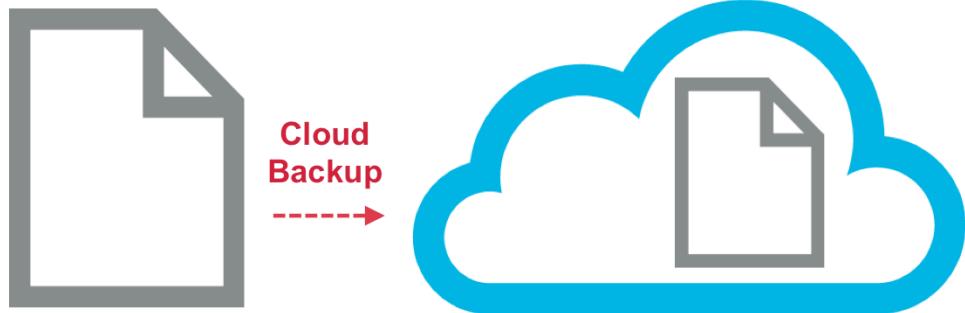
Local mirrors are fast and easy, and if you have a failure you can keep on going as though nothing happened.

Common Problems of Local File Systems: Lost



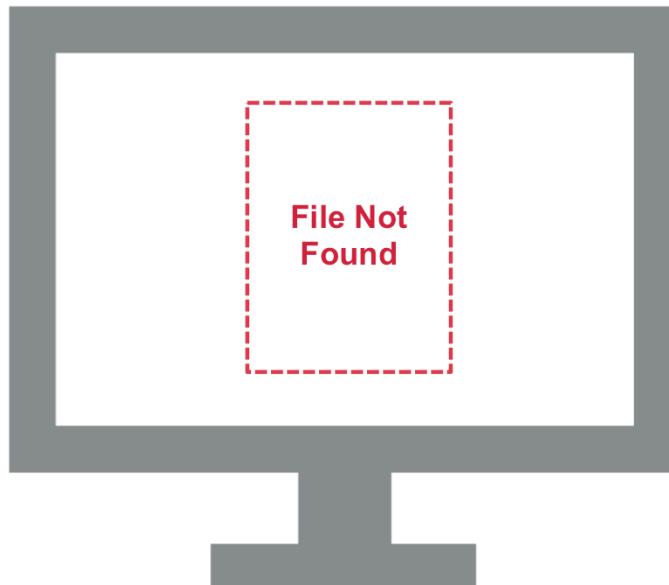
What if someone breaks into your car and steals your computer? Mirroring will not help, as both disks are part of the same computer and are now in the hands of the thief.

Common Problems of Local File Systems: Lost



You can mirror important files in the cloud. That way, when disaster strikes your computer, you can retrieve the most important data quickly. Further, your cloud provider likely has local and remote mirrors of all their customer data, in case a disaster strikes them.

Common Problems of Local File Systems: Human Error



What about simple human error? Anyone who has worked around a computer for any considerable length of time has accidentally deleted an important file or directory. Having a mirror copy, local or remote, will not help, because both mirrors are exact replicas of each other. Deleting from one deletes from the other.

Common Problems of Local File Systems: Human Error



What we should do in this case is to make regular backups of our important files onto a backup drive. That way, when any disaster strikes, natural or man-made, we can restore the files from the backup. How safe your data is depends on how frequent your "regular" backups are. A nightly backup would mean you would lose at most the last 24 hours of work.

Common Problems of Local File Systems: Human Error



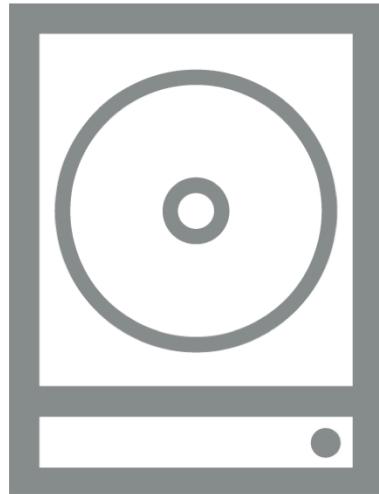
It is common for an operating system to allow you to make incremental backups as often as every hour, protecting you from accidental deletion of files. This way, the first incremental backup usually copies all of the data that you want to back up.

Common Problems of Local File Systems: Human Error



The next backup, however, will only copy any files that are new, or have been modified since the last backup. Since we are usually only saving a small subset of our data, incremental backups are much more efficient in both the space and time required to store them. In addition, we can go back in time to any backup, and recover our drive to match when that backup was made.

Common Problems of Local File Systems: Space

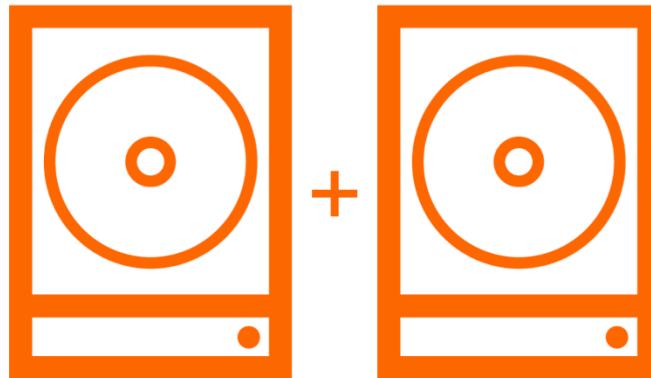


Another common experience is running out of space on your local drive. This could be on your personal drive because you have too many home movies of your kids and pets, or on your work drive because it gets filled up with archived project files. Drives fill up quickly, and we often want to continue to access the old data, while giving ourselves more room for new data.

Common Problems of Local File Systems: Space

Redundant Array of Inexpensive Disks (RAID)

RAID-0: Space = Sum of All Disks



Generally, our local operating system will have a volume manager that will allow us to grow our file system on-the-fly. For a desktop computer, or a set of servers at the office, we can often just add a new drive and run some commands to extend the file system onto that new hard drive. Now we're good to go. The volume manager implements what is called RAID, which stands for a redundant array of inexpensive disks. This particular type of RAID, where the amount of available space is equal to the sum of all disks in the RAID, is called a RAID-0.

Common Problems of Local File Systems: Space

Redundant Array of Inexpensive Disks (RAID)
RAID-1: Space = One of Two Disks



The local mirror we discussed earlier is called RAID-1. Since the data is mirrored on both disks, the amount of space available to a RAID-1 is equal to only one of the two disks.

Knowledge Check





Knowledge Check

How can you protect against these common problems that local file systems face?

Match the problem with its solution.

Human Error	Cloud or external mirror
Insufficient Space	Local or internal mirror
Theft or Loss	Incremental external backups
Disk Failure	Add a new disk to the system

theft or loss :: cloud or external mirror

disk failure :: local or internal mirror

human error :: incremental external backups

insufficient space :: add a new disk to the system

Learning Goals



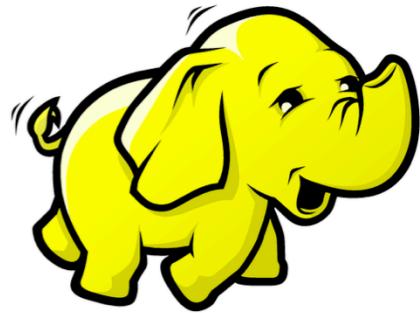
Learning Goals



- 3.1 Compare and Contrast Local and Distributed File Systems
- 3.2 Explain Data Management in the Hadoop File System**
- 3.3 Summarize the MapReduce Algorithm

Next, we'll learn about data management in the Hadoop file system.

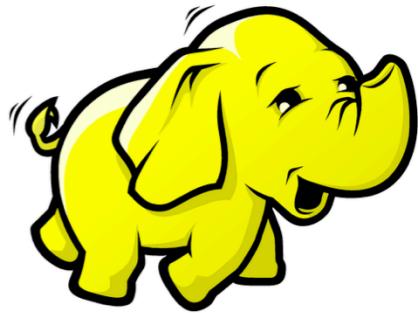
What is Hadoop?



Open-source framework for
distributed storage/processing

Apache Hadoop is an open-source framework designed for distributed storage and processing of very large data sets.

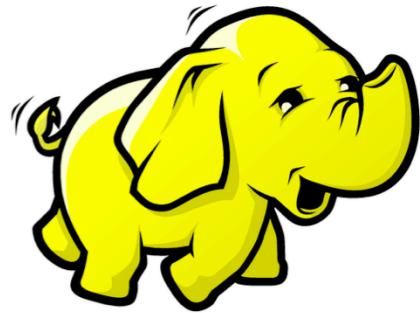
What is Hadoop?



Built from
commodity hardware

Hadoop uses mid-grade machines that are easily replaced; no proprietary hardware is needed. All the nodes in a cluster can provide both storage and analytic power. With Hadoop, you do not need a single, massively powerful machine. Instead, you have a lot of mid-grade machines that distribute the load. If a node fails, the cluster keeps running.

What is Hadoop?



Designed to
tolerate failures

Finally, Hadoop is highly fault-tolerant, with built-in redundancy. If a disk or node fails, the remaining machines take over the work with no loss of data.

Hadoop Distributed File System (HDFS)

Data is replicated three times, by default



Distributed file systems, also called clustered systems or parallel systems, can help solve many of the problems facing local file systems.

In the Hadoop Distributed File System, or HDFS, data is replicated three times by default.

Hadoop Distributed File System

At least one copy is on a different **physical** disk



In HDFS, at least one copy of the data is placed on a different disk from the original copy. This way, if the hardware fails, there is still another copy available.

Hadoop Distributed File System

Physical disks are organized into nodes and clusters



Disks



Node



Cluster

Physical disks are organized into nodes and clusters.

Hadoop Distributed File System

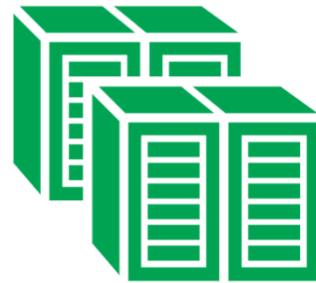
A node is a collection of disks,
and a cluster is a collection of nodes



Disks



Node

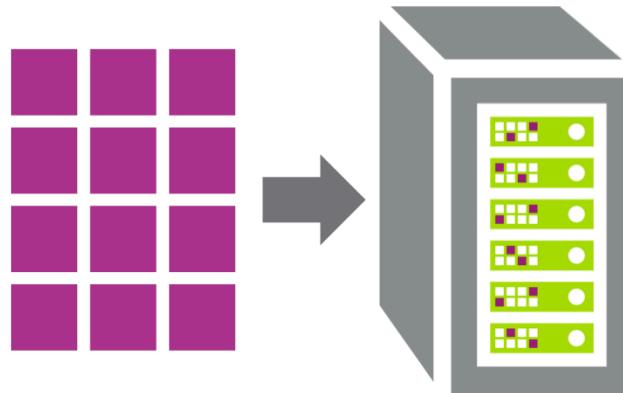


Cluster

A node is a collection of disks, and a cluster is a collection of nodes.
If you need additional space, simply add more nodes.

Hadoop Distributed File System

Large files are split into blocks



In HDFS, large files are split into blocks. Blocks are how much data is replicated, read, or written at a time. Each block of data is replicated three times, by default. This way, even if one node fails, there are still two copies.

For example, if you have a file that is one terabyte in size, HDFS will shard the file and store it across multiple nodes.

Knowledge Check



Knowledge Check



How are distributed file systems, like HDFS, different from local file systems, like the one on your personal computer? Check all that apply.

- A. They automatically replicate data for fault tolerance
- B. They distribute data in small pieces
- C. They do not use physical storage, since they are on the cloud
- D. They are read-only, to prevent user error with read/write access

Answers: A & B

Knowledge Check



How are distributed file systems, like HDFS, different from local file systems, like the one on your personal computer? Check all that apply.

- A. They automatically replicate data for fault tolerance**
- B. They distribute data in small pieces**
- C. They do not use physical storage, since they are on the cloud
- D. They are read-only, to prevent user error with read/write access

Answers: A & B

Definitions



Logical Topology



Physical Topology



Permissions



Compression



Hard and Soft Quotas



Replication

When managing data in a distributed file system like HDFS, you will need to be familiar with some of the following basic concepts: topology, permissions, compression, quotas, and replication.

We'll go over each of these concepts in detail.

Definitions: Topology



Logical Topology



Physical Topology



Permissions



Compression



Hard and Soft Quotas



Replication

Topology refers to how data is arranged. This might be how files are logically arranged, as in the hierarchy of folders on your hard drive.

Definitions: Topology



Logical Topology



Physical Topology



Permissions



Compression



Hard and Soft Quotas



Replication

It can also refer to how data is physically arranged, as in how the disks are arranged in a server room.

You can configure different topologies to enforce that certain data resides in certain physical locations, such as racks and nodes. In this way, you can design your data to exploit locality of reference for performance, availability, and any other arbitrary criteria that suit you.

Definitions: Permissions



Logical Topology



Physical Topology



Permissions



Compression



Hard and Soft Quotas



Replication

Permissions refers to who has access to different parts of your file system.

Definitions: Compression



Logical Topology



Physical Topology



Permissions



Compression



Hard and Soft Quotas



Replication

Compression is the process of making a file smaller in size. A zip folder is an example of a compressed file.

Definitions: Quotas



Logical Topology



Physical Topology



Permissions



Compression



Hard and Soft Quotas



Replication

A quota is the upper bound for disk space use. A user might have a hard quota, which prohibits any additional writing, or a soft quota, which sends an alert when reached.

Definitions: Replication



Logical Topology



Physical Topology



Permissions



Compression



Hard and Soft Quotas



Replication

Data is automatically replicated in HDFS. For example, if your replication factor is three, HDFS will store the original file, plus two copies.

Knowledge Check



Knowledge Check



By default, which of the following is automatic when data is written to HDFS?

- A. Compression
- B. Mirroring
- C. Replication
- D. Snapshots

Knowledge Check



By default, which of the following is automatic when data is written to HDFS?

- A. Compression
- B. Mirroring
- C. Replication**
- D. Snapshots

Answer: C

Learning Goals

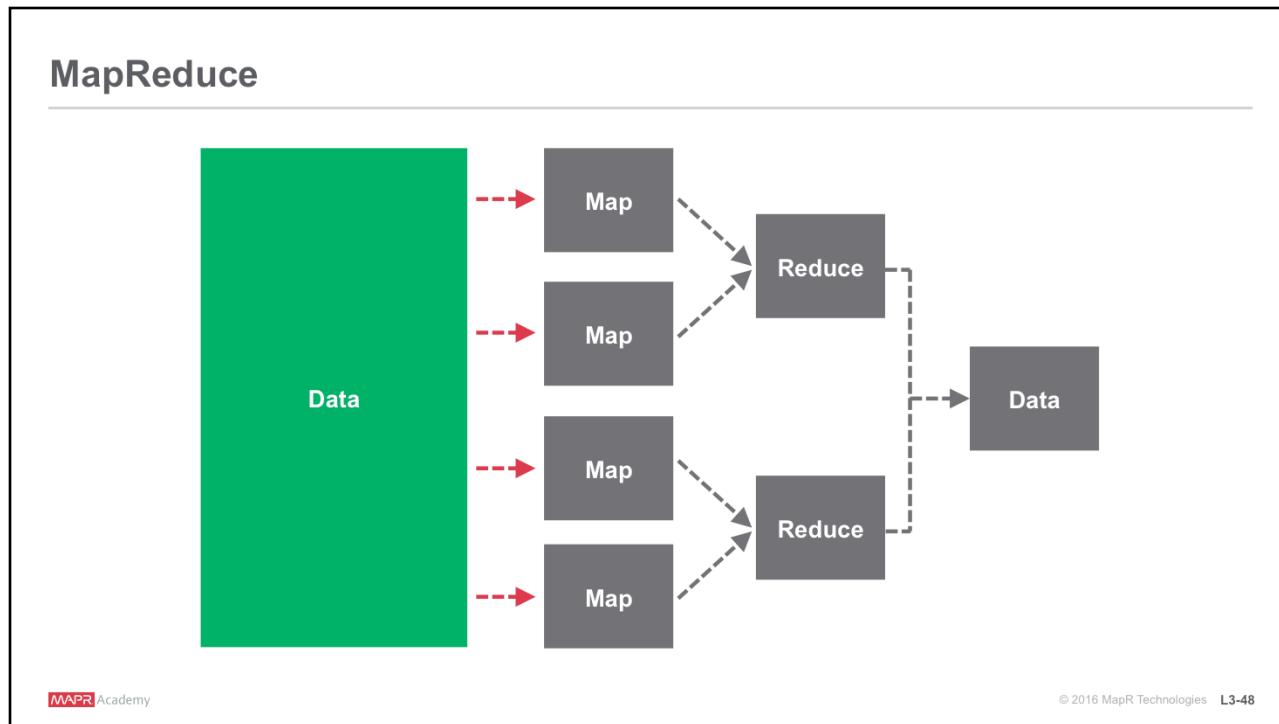


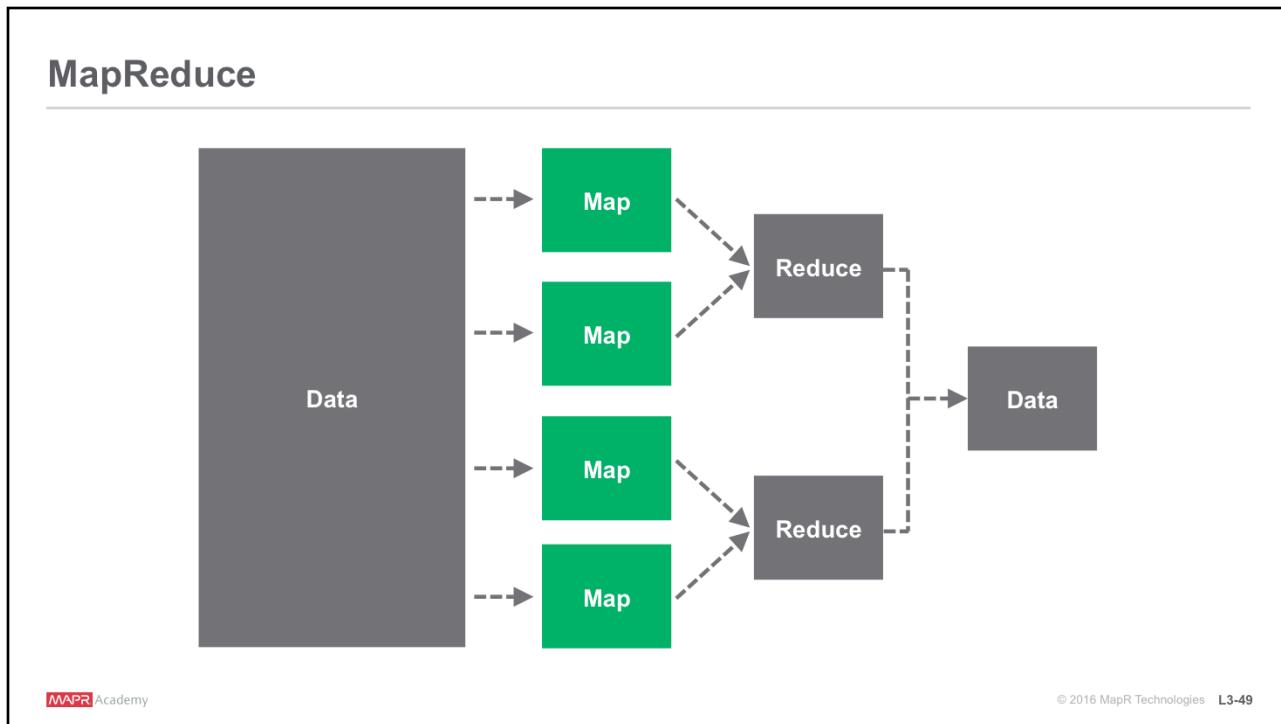
Learning Goals



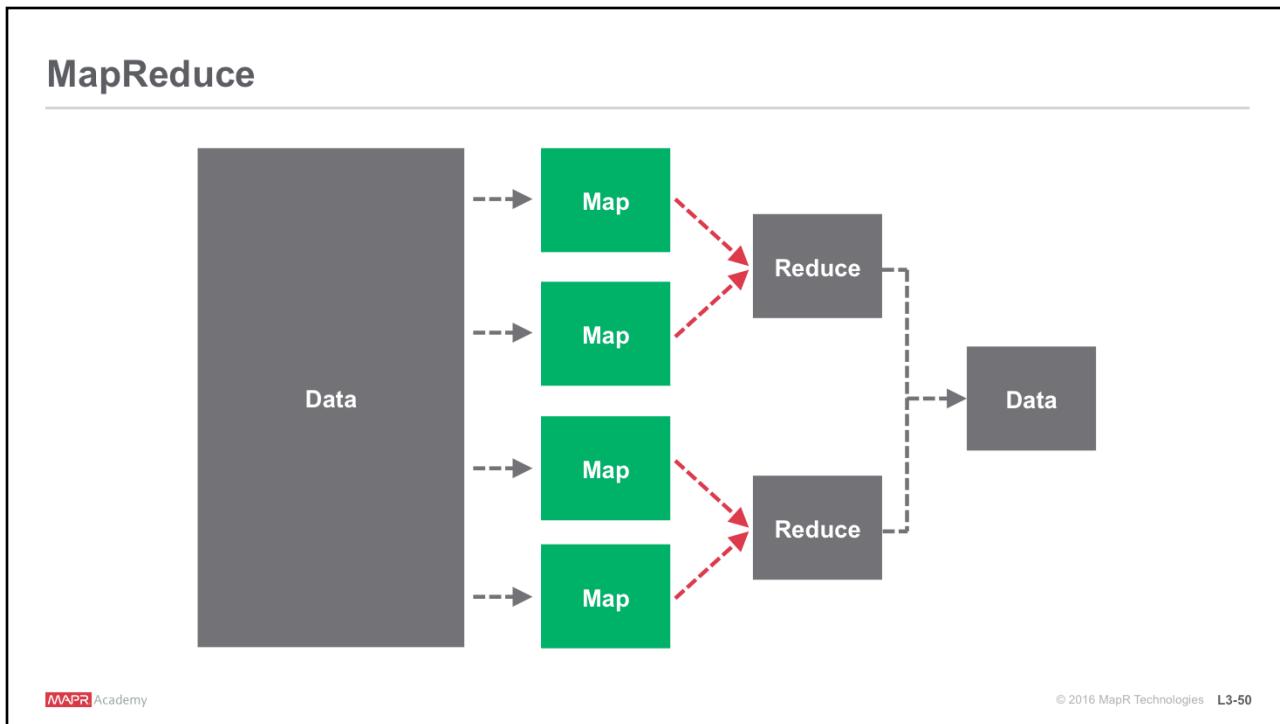
- 3.1 Compare and contrast local and distributed file systems
- 3.2 Explain data management in the Hadoop file system
- 3.3 Summarize the MapReduce algorithm**

Next, we'll learn about how the MapReduce algorithm works.

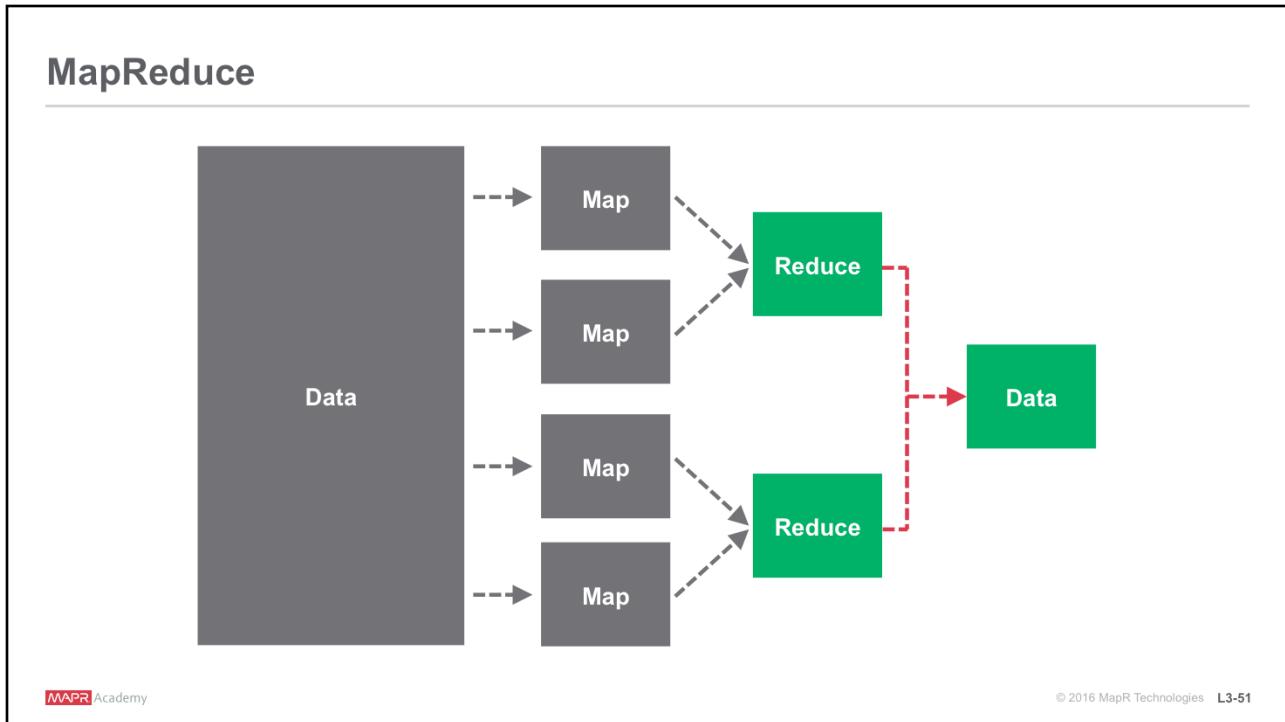




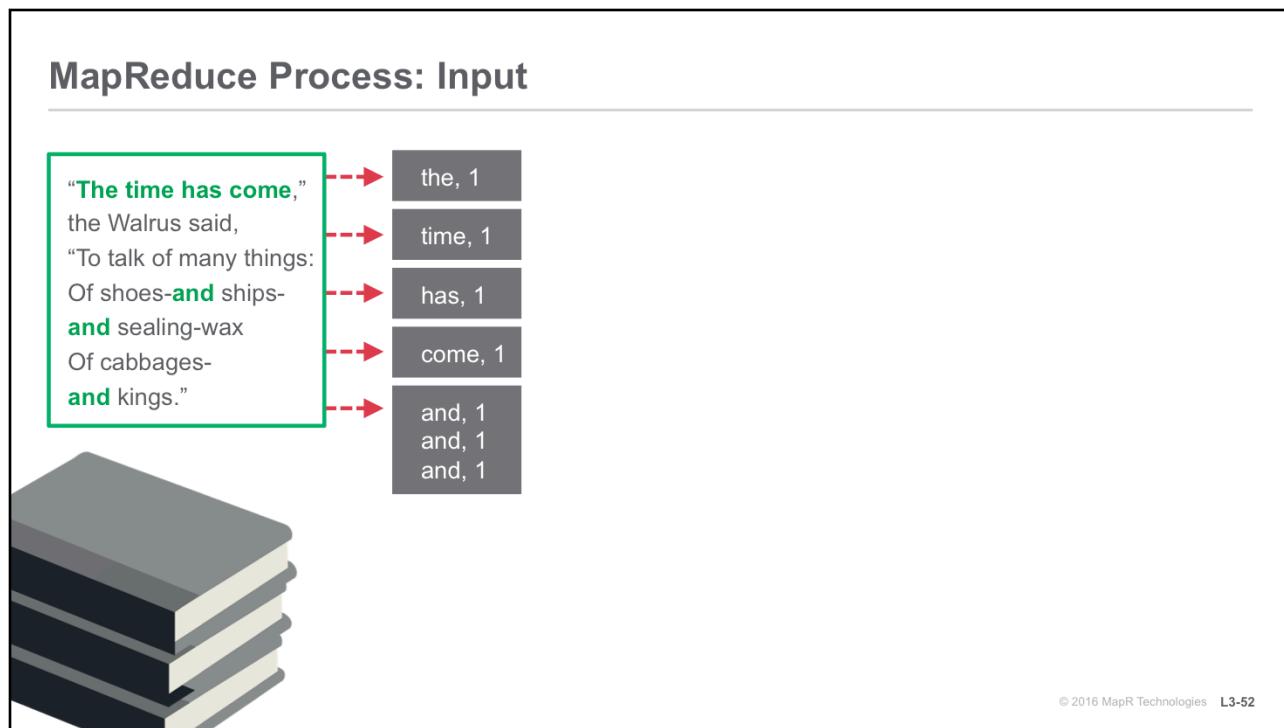
and a map function is applied to each of these splits to produce key-value pairs.



In the shuffle phase, the framework sorts all the key/value pairs from the mappers and partitions them among the reducers.



In the reduce phase a reduce function is applied to each of these partitions. MapReduce is a divide-and-conquer approach that breaks a single, mammoth job into a set of small, manageable tasks.

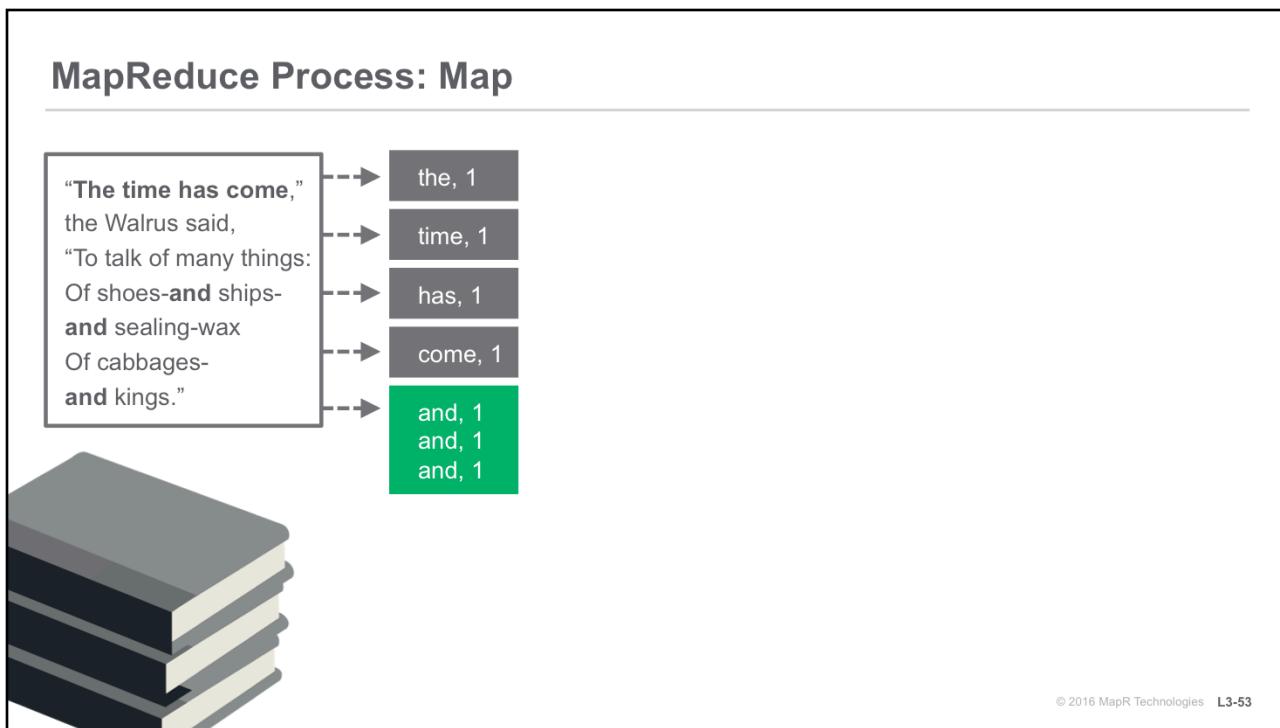


Word count is the "Hello World" of the MapReduce algorithm. Let's look at word count and see how the MapReduce process we just described works.

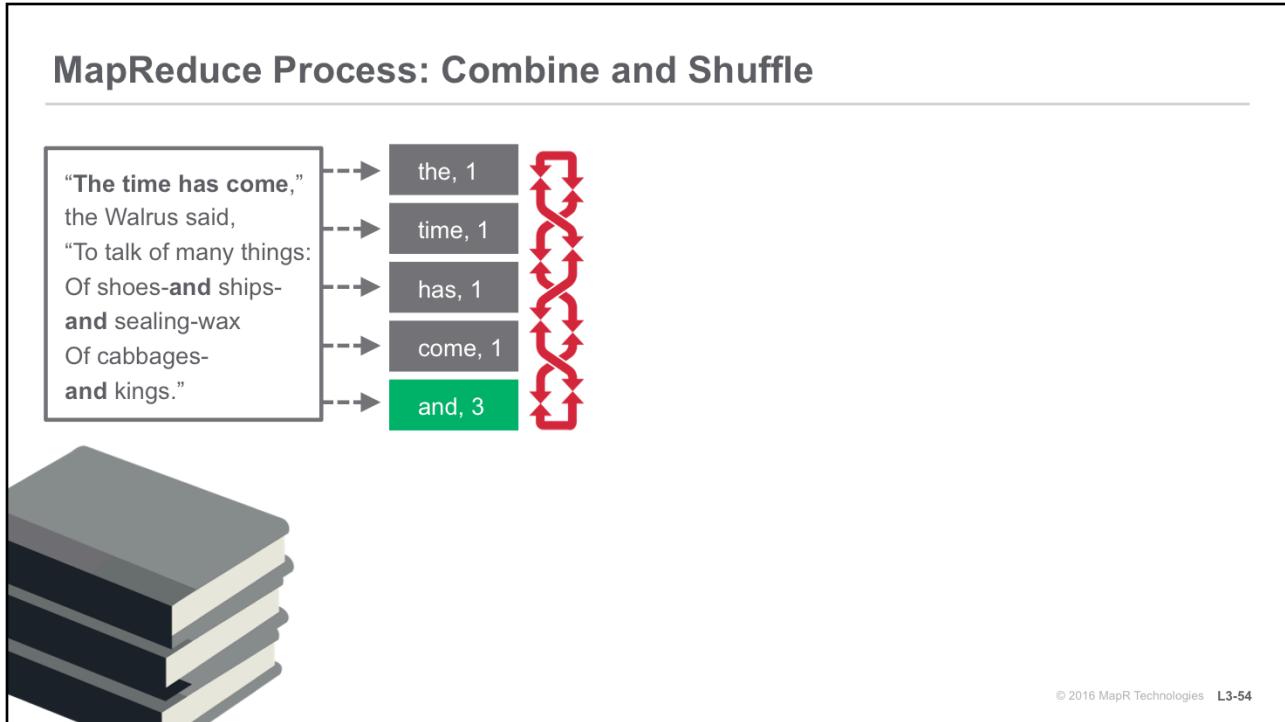
As input, let's say we have all of Lewis Carroll's books, and we want to count the occurrence of each word across all of the books.

First, Hadoop will divide the data files into input splits, and divide the splits among the data nodes.

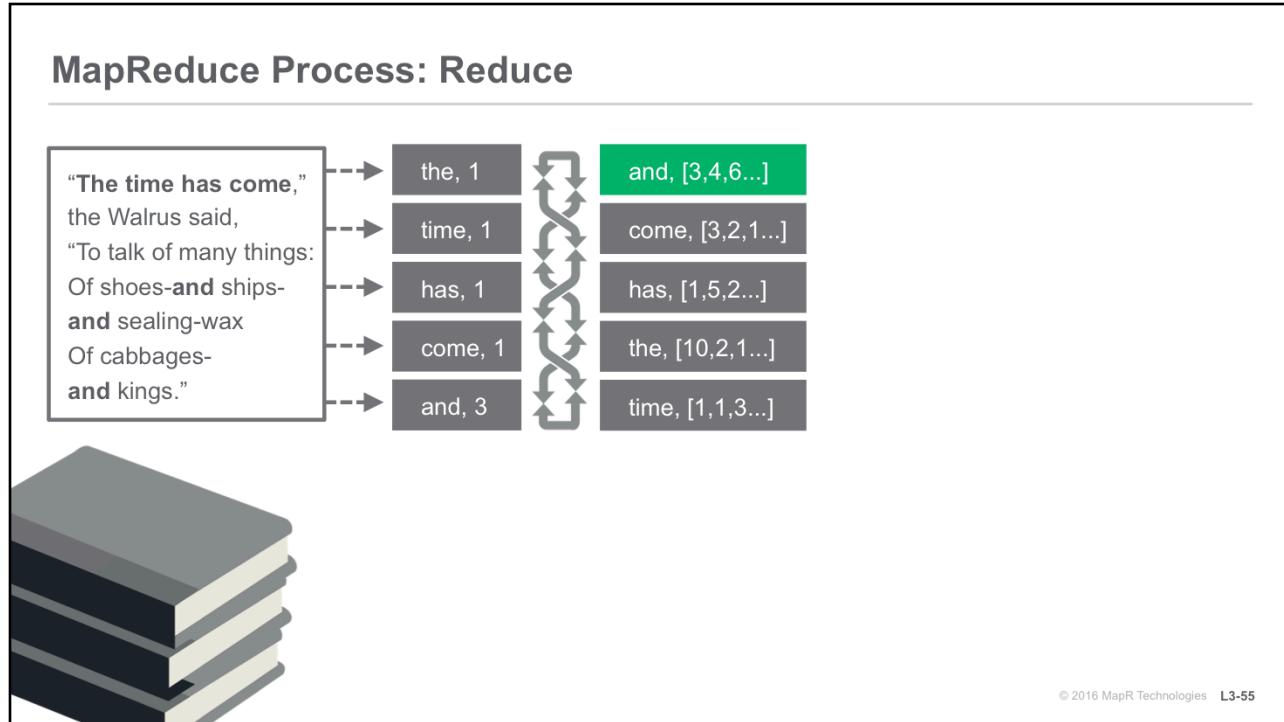
One of the nodes will contain Tweedledee's poem, "The time has come to talk of many things. Of shoes, and ships, and sealing-wax. Of cabbages and kings."



The map method tokenizes the input string, and outputs a key-value pair for every word. The key is the word, and the value is simply 1. As you can see, the word “and” shows up three times, and it produces three distinct key-value pairs.



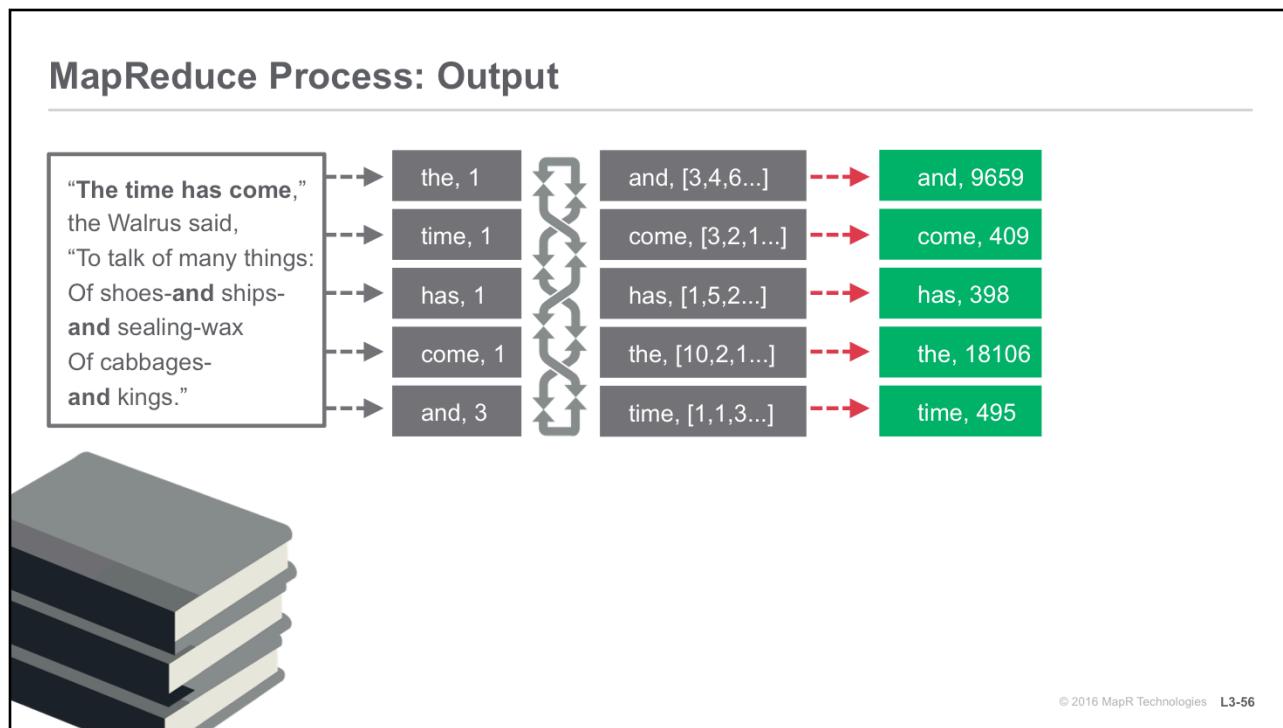
The combiner aggregates multiple instances of the same key coming out of the map into a subtotal. So in our case, the three instances of "and, 1" get combined into a record of value "and, 3" which then goes through the shuffle.



The framework sorts records by key and, for each key, sends all records to a particular reducer.

This reducer gets the keys: and, come, has, the, and time. You can see that the combined value of 3 for "and" shows up here from our first mapper. The other values for "and" come into this same reducer from the other map tasks.

Then begins the reduce phase, which sums up the values from each of the mappers.

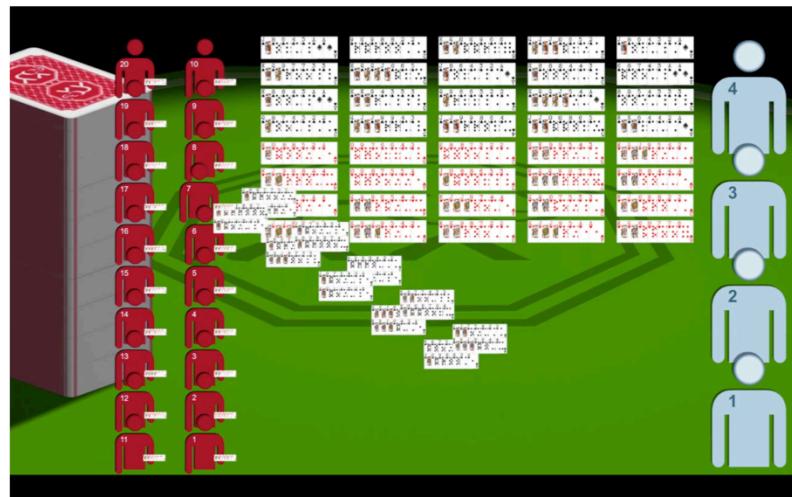


Finally, the framework gathers the output and deposits it in the file system where we can read it. We have now used MapReduce to create a list showing the number of times each word is used in the complete collection of Lewis Carroll's works.

Review



Review



https://www.youtube.com/watch?v=A_8d55ZfWo8

Watch this video for another explanation of how the MapReduce algorithm works.

Knowledge Check



Knowledge Check



What are the phases of the MapReduce process, in the correct order?

- A. map, split, reduce
- B. map, shuffle, reduce
- C. split, map, reduce
- D. shuffle, map, reduce

Knowledge Check



What are the phases of the MapReduce process, in the correct order?

- A. map, split, reduce
- B. map, shuffle, reduce**
- C. split, map, reduce
- D. shuffle, map, reduce

Answer: B



Next Steps

Lesson 4: The Apache Hadoop Ecosystem

ESS 101 – Apache Hadoop Essentials

Congratulations! You have completed ESS 101: Lesson 3. Continue on to ESS 101: Lesson 4, to learn about the Apache Hadoop ecosystem.



ESS 101 – Apache Hadoop Essentials

Lesson 4: The Apache Hadoop Ecosystem

Spring 2016 v5.1

Welcome to ESS 101, Lesson 4, The Apache Hadoop Ecosystem.

Learning Goals



Learning Goals



- 4.1 Define the Apache Hadoop ecosystem
- 4.2 Administration: Apache ZooKeeper, YARN
- 4.3 Ingestion: Apache Flume, Apache Oozie, Apache Sqoop
- 4.4 Processing: Apache Spark, Apache HBase, Apache Pig
- 4.5 Analysis: Apache Hive, Apache Drill, Apache Mahout

When you have finished with this lesson, you will be able to:

- Define the Hadoop ecosystem,
- Define the following ecosystem components:
 - ZooKeeper, YARN,
 - Flume, Oozie, Sqoop,
 - Spark, HBase, Pig,
 - Hive, Drill, and Mahout.

Learning Goals



Learning Goals



4.1 Define the Apache Hadoop ecosystem

- 4.2 Administration: Apache ZooKeeper, YARN
- 4.3 Ingestion: Apache Flume, Apache Oozie, Apache Sqoop
- 4.4 Processing: Apache Spark, Apache HBase, Apache Pig
- 4.5 Analysis: Apache Hive, Apache Drill, Apache Mahout

First, we'll define the Hadoop ecosystem.

Review



Review



In Lesson 2, we introduced the data pipeline.

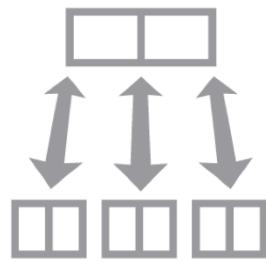
First, data must be created and stored. This data is then ingested and processed, so it can finally be analyzed for useful insights.

In this lesson, we'll take a closer look at the tools used for each of these steps in the data pipeline.

Core Elements of Hadoop



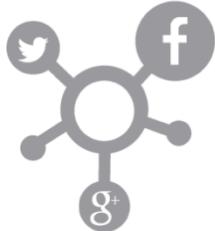
HDFS



MapReduce

The two core elements of Hadoop, HDFS and MapReduce, are used for storing and processing data, but that's just beginning of what Hadoop has to offer as a platform.

A Variety of Data



Unstructured



Semi-Structured



Structured

Data comes in a variety of structured and unstructured formats, and Hadoop can be used with a number of different sources of data. With the need to process a variety of data sources, several applications have been developed to ingest these different types of data into a Hadoop cluster.

The Hadoop Ecosystem



Some of these data processing applications create what's known as the Hadoop ecosystem. Here we see a few of the tools in the Hadoop ecosystem: Spark, Hive, Pig, and Mahout. We'll cover more ecosystem components in this lesson, but there are even more which are beyond the scope of this course.

Knowledge Check



Knowledge Check



What is the Hadoop ecosystem?

- A. A method for making predictions from data
- B. A method of organizing data on a platform
- C. A set of tools for ingesting and processing data
- D. A set of tools for making MapReduce more efficient

Knowledge Check



What is the Hadoop ecosystem?

- A. A method for making predictions from data
- B. A method of organizing data on a platform
- C. A set of tools for ingesting and processing data**
- D. A set of tools for making MapReduce more efficient

Answer: C

Learning Goals



Learning Goals



4.1 Define the Apache Hadoop ecosystem

4.2 Administration: Apache ZooKeeper, YARN

4.3 Ingestion: Apache Flume, Apache Oozie, Apache Sqoop

4.4 Processing: Apache Spark, Apache HBase, Apache Pig

4.5 Analysis: Apache Hive, Apache Drill, Apache Mahout

Let's start by learning about some of the tools used for administration.

Administration

Responsible for many aspects of cluster maintenance



Recall from Lesson 2: administrators are responsible for many aspects of cluster maintenance. One of the primary open source tools Hadoop administrators use is ZooKeeper. Administrators should also be aware of which version of MapReduce or YARN runs on their clusters.

Apache ZooKeeper



- **Coordinates other services**
 - Maintains configuration information
 - Monitors cluster heartbeats
- **Runs on an odd number of nodes**
 - Requires a majority for tie-breakers
- **Starts before other services**

Apache ZooKeeper is a centralized service that maintains configuration information, naming, distributed synchronization, heartbeats and group services for all of the servers. Hadoop uses ZooKeeper to coordinate between services running across multiple nodes.

ZooKeeper runs on an odd number of control nodes, because it needs one member to be the tie-breaker, or leader, among its members. A quorum, or majority, of ZooKeepers must be up for the cluster to run.

ZooKeeper should always be the first service that is started, as the other services in the Hadoop Ecosystem depend on it for coordination.

You can learn more about ZooKeeper by completing the lab activities in ADM 200: Cluster Administration, on learn.mapr.com.

YARN



- Hadoop 2.0 introduced YARN
- "Yet Another Resource Negotiator"
- Allows other applications to run on Hadoop

Originally, MapReduce was the only application run on a Hadoop cluster. Because of this, MapReduce and Hadoop are sometimes thought of as the same thing. This isn't true: Hadoop is the storage and processing paradigm, and MapReduce is a popular application that runs on a Hadoop cluster.

With the advent of Hadoop 2.0, YARN was introduced. YARN stands for Yet Another Resource Negotiator, and allows other applications to run on a Hadoop cluster.

Most Hadoop clusters use YARN to manage the many components of the Hadoop ecosystem, including MapReduce and other tools like Apache Spark.

Learning Goals



Learning Goals

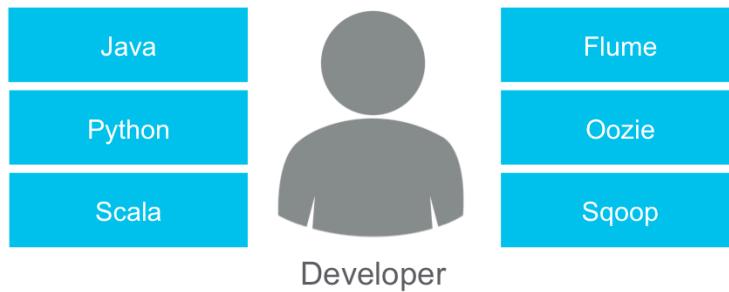


- 4.1 Define the Apache Hadoop ecosystem
- 4.2 Administration: Apache ZooKeeper, YARN
- 4.3 Ingestion: Apache Flume, Apache Oozie, Apache Sqoop**
- 4.4 Processing: Apache Spark, Apache HBase, Apache Pig
- 4.5 Analysis: Apache Hive, Apache Drill, Apache Mahout

Next we'll learn about some of the tools used for data ingestion.

Developers

Responsible for data ingestion



Recall from Lesson 2: developers are responsible for designing, developing, deploying, and maintaining code, typically in programming languages like Java, Python, or Scala. Developers use tools like Flume, Oozie, and Sqoop for data ingestion.

Apache Flume



- Ingests data into Hadoop cluster
- Commonly used with log files

Apache Flume is a reliable, scalable, open source service tool for ingesting streaming data into a Hadoop cluster.

Flume is commonly used to stream events from an external source, such as syslog logs, web server logs, or social media feeds. It then delivers this data to a Hadoop cluster.

Apache Sqoop



- Transfers data between HDFS and external data stores
- Can be used with RDBMS and NoSQL

Apache Sqoop is a utility that transfers data between an external data store and a Hadoop cluster. Sqoop is a command line tool that can be used with both RDBMS and NoSQL data.

You can invoke Sqoop from any client, including your laptop. Sqoop will then call map-only MapReduce jobs to import and/or export data between the Hadoop cluster and the external data stores. Using MapReduce provides parallelization and fault tolerance, and data can be stored in Hive tables or HBase.

Apache Oozie



The image cannot be displayed. Your computer may not have enough memory to open the image, or the image may have been corrupted. Restart your computer, and then open the file again. If the red x still appears, you may have to delete the image and then insert it again.

- Schedules workflows
- Manages multiple jobs

Apache Oozie is a scalable and extensible scheduling system for creating Hadoop workflows. Hadoop workflows are often quite complex, involving multiple applications running in series, parallel, or both. Oozie can be used to manage multiple types of jobs within a workflow, including MapReduce, Pig, Hive, and Sqoop, as well as arbitrary scripts and Java programs.

Knowledge Check



Knowledge Check



Which of the following tools is commonly used to load log files into a Hadoop cluster?

- A. Flume
- B. Oozie
- C. YARN
- D. Sqoop

Knowledge Check



Which of the following tools is commonly used to load log files into a Hadoop cluster?

- A. Flume
- B. Oozie
- C. YARN
- D. Sqoop

Answer: A

Learning Goals



Learning Goals

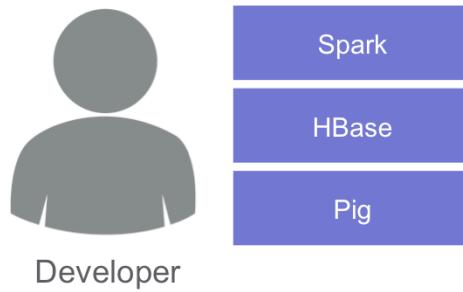


- 4.1 Define the Apache Hadoop ecosystem
- 4.2 Administration: Apache ZooKeeper, YARN
- 4.3 Ingestion: Apache Flume, Apache Oozie, Apache Sqoop
- 4.4 Processing: Apache Spark, Apache HBase, Apache Pig**
- 4.5 Analysis: Apache Hive, Apache Drill, Apache Mahout

Next we'll learn about some of the tools used by developers for data processing.

Developers

Responsible for data processing



In addition to data ingestion, developers use tools like Spark, HBase, and Pig for data processing.

Apache Spark



- **Iterative, in-memory jobs**
 - Cached and fault-tolerant
- **Written in Java, Scala, or Python**

Apache Spark is a framework for performing general data analytics on a distributed computing cluster like Hadoop. Spark caches data sets to provide in-memory computation. These in-memory, iterative jobs mean faster processing than MapReduce for complex analyses like machine learning.

Spark can also process structured data from Hive, Flume, or your own custom data sources, through Spark Streaming or Spark SQL.

Spark applications can be written in Java, Scala or Python, making it an easily accessible resource for data analysts.

You can try out Spark by completing the lab activities in DEV 360: Apache Spark Essentials, on learn.mapr.com.

Apache HBase



- **NoSQL database**
 - Capture large amounts of data
 - Store sparse data with inconsistent values
 - Continuous data with read/write access
- **Written in Java**

Apache HBase is an open-source, distributed, versioned, NoSQL database modeled after Google's Bigtable.

HBase is designed to handle enormous amounts of inconsistent data, and is especially suited for:

- Capturing millions or even billions of rows and columns worth of data from something like systems metrics or user clicks
- Storing sparse data such as chats or emails that has inconsistent values across columns
- Continuously captured data, where random read/write access is needed, such as a web application back-end or search index

Developers primarily write HBase applications in Java.

You can try out HBase by completing the lab activities in DEV 320: Apache HBase Data Model and Architecture, on learn.mapr.com.

Apache Pig



- Includes scripting language called Pig Latin
- Transform and analyze large datasets
- Write functions in Python, Java, JavaScript, Ruby

Apache Pig is a platform for analyzing data. Pig consists of a data flow scripting language called “Pig Latin,” and an infrastructure for converting these scripts into a sequence of MapReduce programs. By using MapReduce, Pig Latin scripts can be used with a Hadoop platform to perform transformations on very large and/or complex data sets.

Developers can write functions for Pig in Python, Java, JavaScript, or Ruby.

You can try out Pig by completing the lab activities in DA 450: Apache Pig Essentials, on learn.mapr.com.

Knowledge Check



Knowledge Check



Which of the following ecosystem tools is commonly used for iterative processing?

- A. Spark
- B. Pig
- C. HBase
- D. Oozie

Knowledge Check



Which of the following ecosystem tools is commonly used for iterative processing?

- A. Spark
- B. Pig
- C. HBase
- D. Oozie

Answer: A

Learning Goals



Learning Goals

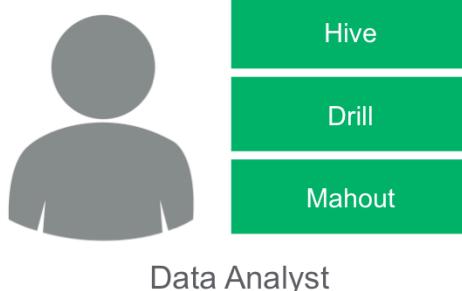


- 4.1 Define the Apache Hadoop ecosystem
- 4.2 Administration: Apache ZooKeeper, YARN
- 4.3 Ingestion: Apache Flume, Apache Oozie, Apache Sqoop
- 4.4 Processing: Apache Spark, Apache HBase, Apache Pig
- 4.5 Analysis: Apache Hive, Apache Drill, Apache Mahout**

Next we'll learn about some of the tools used by data analysts.

Analysts and Scientists

Responsible for analyzing data



Data Analyst

Data analysts and data scientists are responsible for analyzing data. This includes data mining, extraction, normalization, filtering, aggregation, querying, interpreting, graphing, and making predictions using tools like machine learning. Analysts and scientists use ecosystem tools like Hive, Drill, and Mahout.

Apache Hive



- Stores data in HCatalog and Hive Metastore
- Hive Query Language (HQL)
 - SQL-like query language
- Uses MapReduce

Apache Hive is a data warehouse infrastructure built on top of Hadoop. Hive stores data in tables using HCatalog and tracks this data using the Hive Metastore. Hive uses a SQL-like query language called HiveQL. HiveQL provides a way for analysts already familiar with SQL to easily create MapReduce programs, and to execute queries on data stored on Hadoop.

You can try out Hive by completing the lab activities in DA 440: Apache Hive Essentials, on learn.mapr.com.

Apache Drill



- Structured, semi-structured, and unstructured data
- Multiple data sources and data types
- ANSI-SQL compliant

Apache Drill is a query engine for big data exploration. Drill can perform dynamic queries on structured, semi-structured, and unstructured data on HDFS or Hive tables using ANSI-SQL.

Drill is very flexible, and can query a variety of data from multiple data sources. This makes it an excellent tool for exploring unknown data. Unlike Hive, Drill does not use MapReduce and is ANSI-SQL compliant, making it faster and easier to use.

You can try out Drill by completing the lab activities in DA 410: Apache Drill Essentials, on learn.mapr.com.

Apache Mahout



- Clustering
- Classification
- Collaborative filtering

Apache Mahout is a suite of scalable machine learning libraries. It supports clustering, classification, and collaborative filtering. These machine learning algorithms can use large amounts of data to make predictions, such as the recommendations made by Netflix or Amazon.

Mahout also has low-level matrix math libraries for those who do want to write their own algorithms. Mahout has traditionally been run using MapReduce, but support for Spark has been added to Mahout.

Knowledge Check



Knowledge Check



What advantages does Apache Drill have over Apache Hive?

- A. Drill is ANSI-SQL compliant while Hive is not
- B. Drill works on HDFS while Hive does not
- C. Drill uses MapReduce while Hive does not
- D. Drill is a newer technology and Hive is no longer maintained

Knowledge Check



What advantages does Apache Drill have over Apache Hive?

- A. **Drill is ANSI-SQL compliant while Hive is not**
- B. Drill works on HDFS while Hive does not
- C. Drill uses MapReduce while Hive does not
- D. Drill is a newer technology and Hive is no longer maintained

Answer: A

Review



Review



- ZooKeeper: <https://zookeeper.apache.org/>
- YARN: <http://hadoop.apache.org/>
- Flume: <http://flume.apache.org/>
- Oozie: <http://oozie.apache.org/>
- Sqoop: <http://sqoop.apache.org/>
- Spark: <http://spark.apache.org/>
- HBase: <https://hbase.apache.org/>
- Pig: <https://pig.apache.org/>
- Hive: <https://hive.apache.org/>
- Drill: <https://drill.apache.org/>
- Mahout: <http://mahout.apache.org/>

You can learn more about each of these open source Apache Hadoop ecosystem tools by visiting these webpages.



Next Steps

Lesson 5: Solving Big Data Problems with Apache Hadoop

ESS 101 – Apache Hadoop Essentials

Congratulations! You have completed ESS 101: Lesson 4. Continue on to ESS 101: Lesson 5, to learn about some use cases for Apache Hadoop.



MAPR Academy

ESS 101 – Apache Hadoop Essentials

Lesson 5: Solving Big Data Problems With Apache Hadoop

Spring 2016 v5.1

Welcome to ESS 101, Lesson 5: Solving Big Data Problems With Apache Hadoop.

Learning Goals



Learning Goals



Summarize the Following Use Cases:

- 5.1 Data Warehouse Optimization
- 5.2 Recommendation Engine
- 5.3 Large-scale Log Analysis

When you have finished with this lesson, you will be able to summarize the following use cases:

- Data warehouse optimization
- Recommendation engine
- Large scale log analysis

Review



Review



We discussed this workflow in Lesson 2, when we learned about the data pipeline. In Lesson 4, we introduced several parts of the Hadoop ecosystem, based on roles like administration, ingestion, processing, and analysis. This represents the typical workflow for Hadoop:

- Data is created and stored on a database
- Data is ingested into a Hadoop cluster using one or more of the ecosystem components
- Data is then processed, again using one or more components of the Hadoop ecosystem
- The results are then analyzed for insights

In this lesson, we'll take a closer look at how this works in real-world use cases.

Learning Goals



Learning Goals



Summarize the Following Use Cases:

5.1 Data Warehouse Optimization

5.2 Recommendation Engine

5.3 Large-scale Log Analysis

Let's start by learning about data warehouse optimization.

What is a Data Warehouse?

Centralized repository storing data from a variety of sources



The enterprise data warehouse is a centralized repository for data.

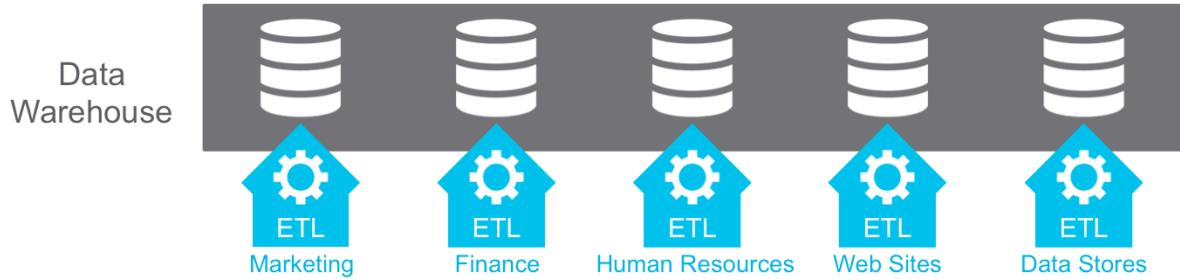
A data warehouse stores data from multiple sources. Data might come from internal operations systems like marketing, finance and human resources data, or external data from sources like web sites or external data stores.

The data warehouse provides a single home for all of these disparate sources of data, allowing the data analyst to query, or otherwise process, any or all of this data as though it was a single source.

To help with querying and processing, data warehouses store data in a structured format.

Data Warehouse ETL

ETL process becomes more taxing to data warehouse
as velocity, variety, and volume increase



In most cases, the data being ingested needs to be converted to match the schema of the data warehouse before it can be stored and processed. Many data warehouses do this through a process called Extract, Transform and Load, or ETL. The data is extracted from its source, transformed into the data warehouse structure, and then loaded into the warehouse resources.

Each different data source requires its own script or API to convert its raw data into the data warehouse schema. The ETL process has become more taxing to the data warehouse as the velocity, variety, and volume of data has increased.

Hadoop and Data Warehouse Retail Example

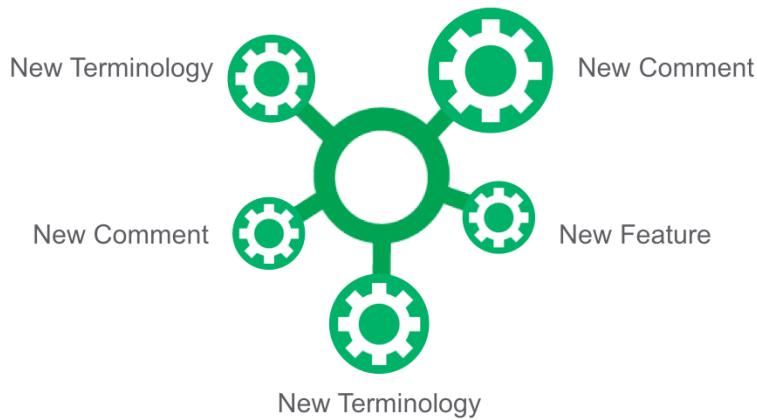


As an example, let's consider the needs of a large retail merchant who is trying gain a better understanding of what is happening with their products. They want to track metadata for each item, as well as build correlations to customer ratings and sentiment analysis.

Product description data such as size, weight, color, manufacturer, and other such information, may be stored in a structured or semi-structured format, such as a JSON file.

Social media data can be used for customer sentiment analysis. However, it may be semi-structured or unstructured, making it difficult to store and process.

Hadoop and Data Warehouse Retail Example



There will likely be frequent changes to the social media content as customers write new comments. Even the product information may change, as new features are added or there are changes in the terminology about the product.

Trying to handle these frequent changes in an enterprise data warehouse can cause costly delays, as the data would have to be transformed to store each time there is a change.

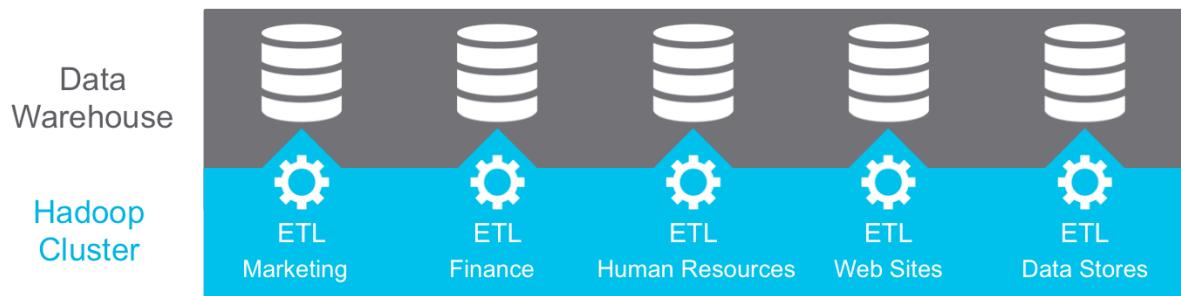
Data Warehouse Optimization

How do we build a **cost-effective, scalable solution** to consume big data while taking advantage of **legacy query work and skills** our data analysts have developed?

How do we build a cost-effective, scalable solution, then, so that we can consume big data while taking advantage of our legacy query work and the skills that our data analysts have developed?

Hadoop and Data Warehouses

Hadoop separates ETL from the DW

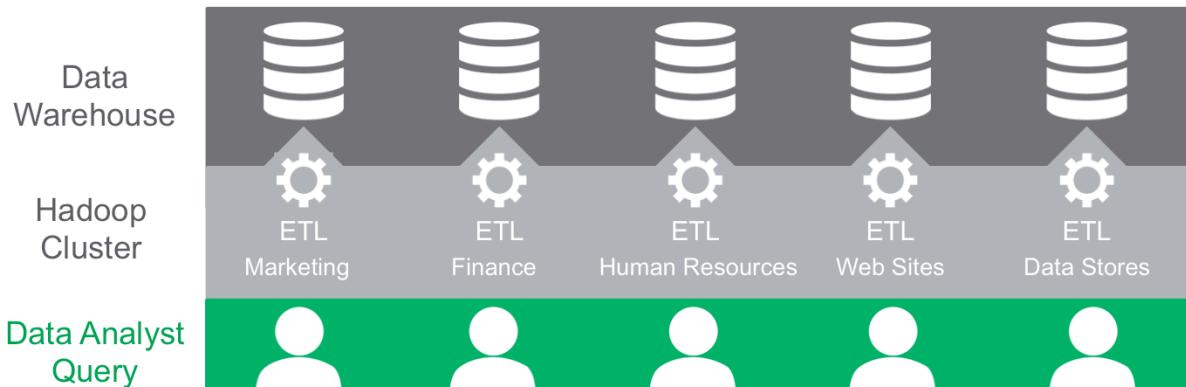


Hadoop provides the solution we need, by separating the extract and transform services from the data warehouse. This allows the data warehouse to focus on storing data. With tools like Apache Flume and Apache Pig handling the extraction and transformation stages, our data warehouse only needs to store the refined data, greatly reducing the cost of our expensive warehouse resources.

Hadoop easily and quickly consumes multiple forms of structured, unstructured, flat and streaming data, and already has all of the necessary tools to transform these different data types so that it can be delivered to the data warehouse in the proper structure.

Hadoop and Data Warehouses

Data analysts continue to query data warehouse using legacy tools and code



MAPR Academy © 2016 MapR Technologies L5-14

Our data analysts can continue to query the data warehouse using their legacy tools and code, without the need for a steep learning curve of new products.

In addition, Hadoop can consume data sources that were previously unavailable to the data warehouse because they could not be transformed using traditional methods. This gives our analysts access to data they could not look at previously, unlocking new insights.

When we upload transformed data into our data warehouse, the ingested data can still be kept on the Hadoop cluster in its original format. This means we have the freedom to come back to this data and process it in a different way, then re-ingest it into our warehouse.

Review



Review



Hadoop and Data Warehouses

- Ingests and stores data in original format
- Provides more cost effective storage of raw data
- Saves data for future use (instead of use and discard)

Let's review.

Hadoop ingests and stores the data in its original format. We can then transform the data when we are ready, and upload the transformed data to our enterprise warehouse for use.

The Hadoop system provides much more cost effective long term storage of the raw data. This allows the data to be saved for future use, instead of using and discarding it, as is typical with a traditional data warehouse model.

You can delay making decisions about how to use the data, which further supports multi-purpose use of large data sets. You protect your options by preserving the raw data and using it in different ways as the need arises.

Knowledge Check



Knowledge Check



What is one benefit of migrating data from a traditional SQL-based data warehouse to a Hadoop cluster?

- A. Hadoop automatically converts all data to SQL format
- B. Hadoop allows analysts and developers to use legacy code
- C. Hadoop clusters keep data clean by deleting old files
- D. Hadoop keeps costs down by limiting maximum file size

Knowledge Check



What is one benefit of migrating data from a traditional SQL-based data warehouse to a Hadoop cluster?

- A. Hadoop automatically converts all data to SQL format
- B. Hadoop allows analysts and developers to use legacy code**
- C. Hadoop clusters keep data clean by deleting old files
- D. Hadoop keeps costs down by limiting maximum file size

Answer: B

Hadoop keeps your data in its original format, and performs any processing you might perform on high-variety data.

Hadoop allows you to use legacy code, reducing the learning curve for developers and analysts.

Hadoop does not delete old files or limit your file size, because it provides massively scalable storage for high-volume data.

Learning Goals



Learning Goals



Summarize the Following Use Cases:

5.1 Data Warehouse Optimization

5.2 Recommendation Engine

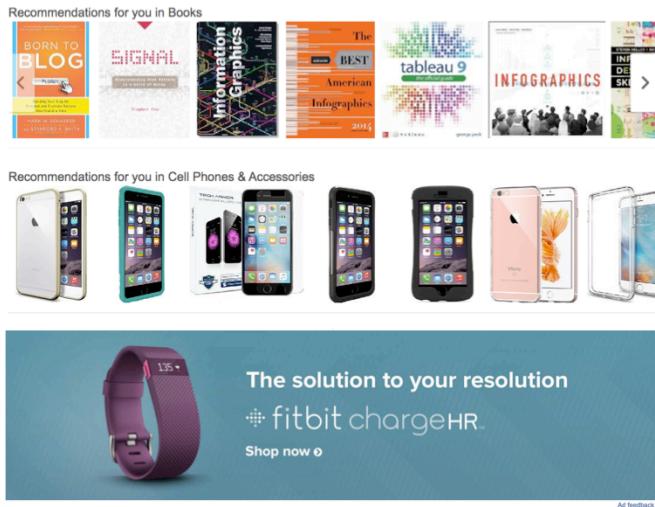
5.3 Large-scale Log Analysis

Next we'll learn about another use case: building a recommendation engine.

Gaining Value Through Recommendation

Recommendations:

- Provide suggestions on what to read, watch, buy
- Increase customer engagement, satisfaction, retention



It's very likely that you have seen suggested items listed when shopping online at a site like Amazon.

Recommendations provide your customers with suggestions about what they might like to read, to watch, or to buy as well as adding variability and relevance to the experience. Recommendations make each visit to a website fresh and interesting, thereby increasing customer engagement, satisfaction, and retention.

If the suggestions appeal to your customer, then the recommendation system is doing its job. To be successful, recommendations must be delivered in a timely manner and appeal to the specific user's tastes.

Recommender Workflow

The diagram illustrates the Recommender Workflow. On the left, a blue robot icon with a gear on its chest points to three examples of recommendation interfaces. The top example shows 'Recommendations for you in Books' with book covers for 'BORN TO BLOG', 'SIGNAL', 'Information Graphics', and 'The BEST American Infographics'. The middle example shows 'Recommendations for you in Cell Phones & Accessories' with images of various iPhone models and cases. The bottom example is an advertisement for the 'fitbit chargeHR' with the text 'The solution to your resolution' and 'Shop now'.

MAPR Academy

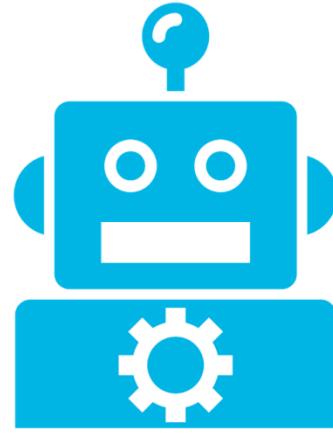
© 2016 MapR Technologies L5-23

So, how do we know what to recommend?

We need to build a machine learning model that can accurately predict what each individual will want.

Machine Learning Algorithms

- Classification
- Clustering
- Collaborative filtering



Recommendation engines can get quite complicated. They filter recommended items in a number of ways, and perform calculations offline, all to achieve a recommendation that is likely to lead to another sale.

Most recommendation engines use some sort of machine learning algorithm. The most common machine learning algorithms are classification, clustering, and collaborative filtering.

Machine Learning Algorithms: Classification

Supervised machine learning that takes a set of known classes and learns how to classify new records based on that information

Example: Spam Detection

"Pre-approved" and "\$\$\$" are in the "spam" class



Classification takes a set of data with known labels and learns how to label new records based on that information. The algorithm identifies which category or class an item belongs to, based on labeled examples of known items.

Because classification works with a pre-defined set of outputs, it is considered a "supervised" machine learning algorithm. The data scientist decides the classes in advance.

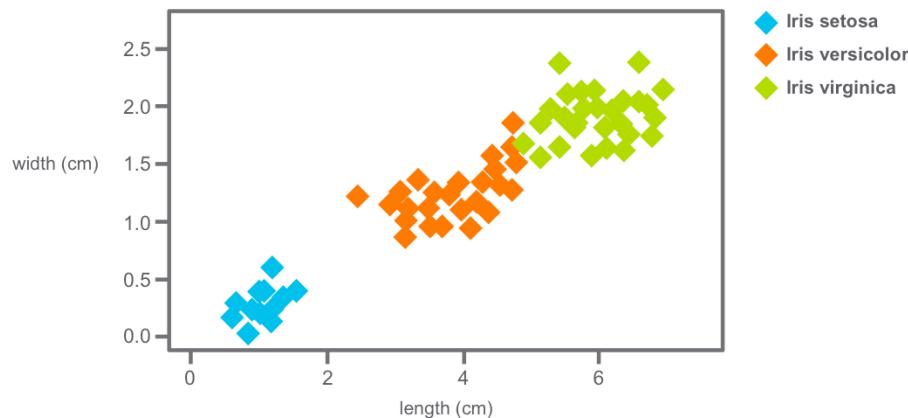
Classification can decide whether or not an email is spam based on emails already classified as "spam" or "not-spam." Classification can also categorize a transaction as fraud or not fraud, and categorize social media feeds based on positive or negative sentiment.

Machine Learning Algorithms: Clustering

Unsupervised machine learning that groups similar objects

Example: Group Species

Clusters of iris species based on petal length and width



In clustering, an algorithm groups inputs into clusters by analyzing similarities between input examples. Clustering uses unsupervised algorithms, which do not have the outputs in advance. No known classes are used as a reference, as with a supervised algorithm like classification. The algorithm decides the clusters by itself.

Clustering can solve for many problems, such as:

- grouping similar species of plants or animals
- anomaly detection, such as fraud detection
- and text categorization, such as sorting books into genres

Machine Learning Algorithms: Collaborative Filtering

C.compares preference data from different users to create a recommendation model

	Ted	Sue	Bob
Jaws			
Star Wars			
Indiana Jones			

Finally, a collaborative filtering algorithm compares preference data from different users, to create a model that can be used for recommendations or predictions.

In the example shown, Ted likes Jaws, Star Wars, and Indiana Jones. Sue likes Star Wars and Indiana Jones. Bob likes Star Wars. To recommend a movie to Bob, we calculate that users who liked Star Wars also liked Indiana Jones, so Indiana Jones is a possible recommendation for Bob.

Recommender Workflow

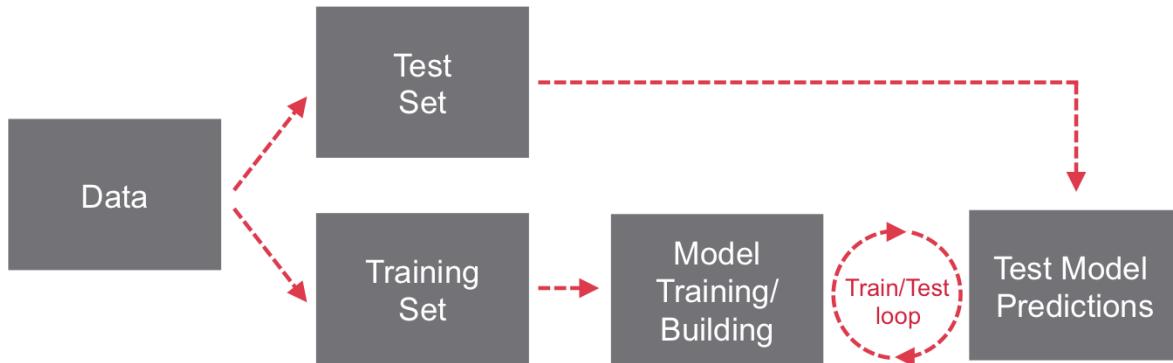
How do we know what to recommend?
Crowd behavior helps understand individual behavior



So how do we know what to recommend? First, we observe what a number of people do, and observe how they interact with various items. We observe what they buy, what they read or watch, and which links they click on a webpage.

We look for a pattern of interaction between a customer and various items or events. Once we have collected a large enough sample of these user behavior histories, we have the raw data on which to model recommendations to predict what an individual will want, using collaborative filtering.

Using Hadoop to Build our Recommender



We will use Hadoop as the core of a two-stage recommendation engine. In this design, a very large set of past user behavior histories will be loaded into our Hadoop cluster offline. The data is split into two sets: test and training.

These data sets will be used to train our collaborative filtering model. Then, we can use this data to predict what a new user might like, using a machine learning tool like Apache Spark or Apache Mahout. As more and more users add data, the algorithm loops, continuously improving the recommendation accuracy.

Knowledge Check



Knowledge Check



What type of machine learning algorithm is used for recommendation?

- A. Clustering
- B. Correlations
- C. Classification
- D. Collaborative filtering

Answer: D

Knowledge Check



What type of machine learning algorithm is used for recommendation?

- A. Clustering
- B. Correlations
- C. Classification
- D. Collaborative filtering**

Answer: D

Learning Goals



Learning Goals



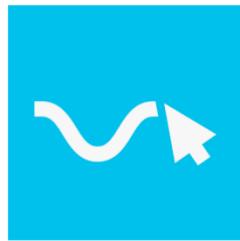
Summarize the Following Use Cases:

- 5.1 Data Warehouse Optimization
- 5.2 Recommendation Engine
- 5.3 Large-scale Log Analysis**

Next we'll learn about large scale log analysis.

Log Analysis Use Case

Log Data Producing Systems



Clickstream Data
Log Files



Set-Top Box
Streaming Logs

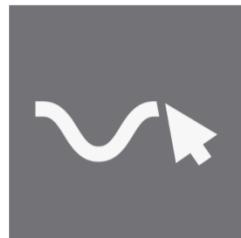


Server Logs
User Activity

There are many different systems that produce log data, most of which are reported in some form of text. For example, clickstream data that captures users' website activity is often stored in log files,

Log Analysis Use Case

Log Data Producing Systems



Clickstream Data
Log Files



Set-Top Box
Streaming Logs

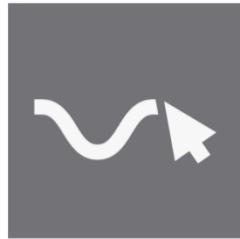


Server Logs
User Activity

and set-top boxes for cable companies report a stream of logs while the box is in use.

Log Analysis Use Case

Log Data Producing Systems



Clickstream Data
Log Files



Set-Top Box
Streaming Logs



Server Logs
User Activity

Server logs are common in almost every computer system: recording events such as users logging onto the system or the launch of an application, as well as errors and other system or application events.

This data can be used to detect security or fraud breaches in your networks.

Log Analysis Use Case

Organizations need a new approach to log analysis, to analyze and store large volumes of historical logs cost-effectively



Discarding data loses
valuable future insights



Difficult to process
nested data

Traditionally, server logs are collected over a period of time, analyzed in real time, and then discarded as the data grows very large, very quickly. Discarding this data poses a potential problem to the organization, however, as valuable future insights will be lost.

Server logs are often saved in a nested format, organized by date and time of the event. This makes it efficient, because meaningful groups of information are stored together. However, it is difficult to process nested data with analytics tools like SQL.

Organizations need a new approach to log analysis, to analyze and store large volumes of historical logs cost-effectively.

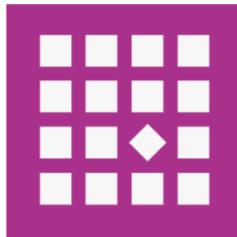
Improve Security Breach Detection

Hadoop ecosystem tools can handle high velocity, variety, volume data in batch or real-time



The Hadoop ecosystem tools like Apache Drill can analyze these large amounts of structured, semi-structured and unstructured log data, from multiple security sources, in batch and real time, in a cost-effective manner.

Improve Security Breach Detection



Anomaly
detection



Suspicious
behavior



Unauthorized
access



Forensic
analysis

Once our data is stored on our Hadoop cluster, we can then further leverage the data with machine learning and statistical analysis to provide actionable intelligence to our organization, such as:

- Identify anomalies and correlate events indicating a threat or attack
- Detect suspicious user behaviors or transactions
- Identify unauthorized access from devices over a network
- Perform forensic analysis or investigations on historical data

Knowledge Check



Knowledge Check



Why is Hadoop an effective platform for capturing and analyzing log data?

- A. Hadoop can ingest a wide variety of incoming data
- B. Hadoop can store the raw data cost-effectively
- C. Hadoop can process the data in a number of ways
- D. All of the above

Answer: D

Knowledge Check



Why is Hadoop an effective platform for capturing and analyzing log data?

- A. Hadoop can ingest a wide variety of incoming data
- B. Hadoop can store the raw data cost-effectively
- C. Hadoop can process the data in a number of ways
- D. **All of the above**

Answer: D



Next Steps

Lesson 6: MapR-FS

ESS 102 – MapR Converged Data Platform Essentials

Congratulations! You have completed ESS 101: Lesson 5. Continue on to ESS 102: Lesson 6, to learn about MapR-FS.



ESS 102 – MapR Converged Data Platform Essentials

Lesson 6: MapR-FS

Spring 2016 v5.1

Welcome to ESS 102, Lesson 6: MapR File System.

Learning Goals



Learning Goals



- 6.1 Review Key Components of HDFS
- 6.2 Describe Key Components of MapR-FS
- 6.3 Compare and Contrast MapR-FS and HDFS

In this lesson, you will:
review key components of HDFS,
describe key components of MapR-FS,
and compare and contrast MapR-FS and HDFS.

Learning Goals



- 6.1 Review Key Components of HDFS**
- 6.2 Describe Key Components of MapR-FS
- 6.3 Compare and Contrast MapR-FS and HDFS

Let's start by reviewing the Hadoop Distributed File System.

The Hadoop Strategy

- Distribute data
- Distribute computation
- Tolerate faults

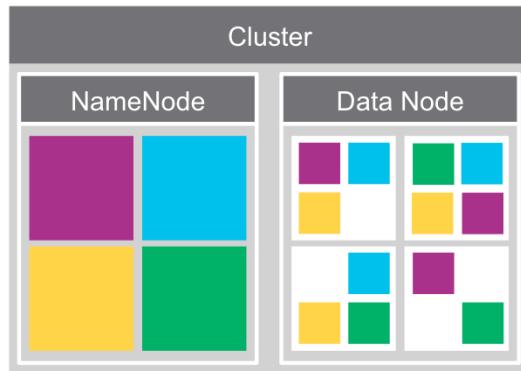
The Hadoop strategy is to:

1. Distribute data
2. Distribute computation
3. Tolerate faults

The Hadoop Strategy

Distribute data

Metadata managed by NameNode



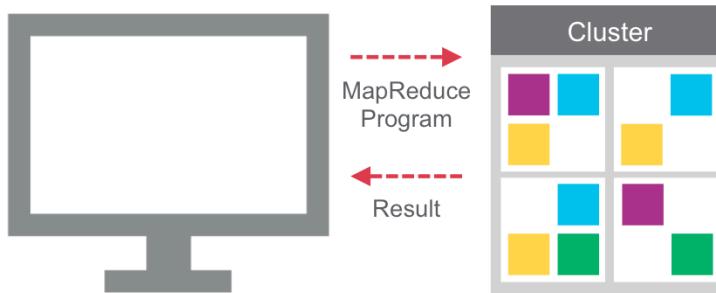
Data is distributed across nodes. HDFS splits data into blocks of 128 megabytes, and distributes these blocks across different locations throughout your cluster. Files are automatically distributed as they are written.

In HDFS, metadata is managed by the NameNode. Before any operations can be performed on data stored in HDFS, an application must contact the NameNode.

The Hadoop Strategy

Distribute computation

MapReduce sends resources to the data



The MapReduce process distributes computation. Once data is split, different nodes can perform computations in parallel.

Every task is completely independent - it doesn't require any output or information from any other task. This is why, if a task fails, it can simply be re-launched without impacting anything else.

The MapReduce paradigm is based on sending computational resources to where data resides whenever possible. This is referred to as data locality. Much of the computation happens on nodes with data, which minimizes network traffic.

The Hadoop Strategy

Tolerate faults

Data stored on any node gets replicated multiple times across cluster. No single point of failure



By distributing both data and computation Hadoop is fault-tolerant. This eliminates single points of failure which could cause a disaster in your cluster.

The cluster is a collection of many servers with multiple disks. The Distributed File System makes use of the disks, while MapReduce or other tasks make use of the processors on each server.

Data stored on any node gets replicated multiple times across the cluster. These replicas prevent data loss. If one node fails, other nodes can continue processing the data. If a MapReduce or other task fails, another one can be scheduled.

Limitations of HDFS

Aspect	Limitation
Reliability	NameNode may cause performance bottlenecks
Mutability	Write once, read many; data immutable
Block size	Same size for I/O, sharding and replication
POSIX semantics	Must use 'hadoop fs' to access data
Availability	Inconsistent snapshots and no built-in mirroring capability
Performance	Written in Java and runs on file system

This table lists a few limitations of HDFS.

- The single NameNode maintains metadata information for all the physical data blocks that comprise the files. This can create performance bottlenecks.
- Data in HDFS is immutable. If the source data changes, the data must be appended to existing data, or else reloaded into the cluster.
- The same block size, while configurable, is used to define the I/O size, the sharding size, and replication. This one-size-fits-all approach does not exploit the inherent differences between these requirements.
- HDFS does not support standard POSIX file semantics. You must use the 'hadoop fs' command in order to read and write the data. Users of HDFS must learn and incorporate this into their data flows.
- HDFS has limited support for snapshots or remote mirrors, both of which enhance the availability and usability of the data set.
- Since HDFS is written in Java, it is slower than file systems written and compiled into machine code.

Class Discussion



Class Discussion



Discuss some of the limitations of HDFS, such as performance bottlenecks. What features are important for your big data projects?

Learning Goals



Learning Goals



- 6.1 Review Key Components of HDFS
- 6.2 Describe Key Components of MapR-FS**
- 6.3 Compare and Contrast MapR-FS and HDFS

Now, let's learn more about MapR-FS.

The MapR-FS Strategy

- Distribute data
- Distribute computation
- Tolerate faults
- More efficient than Hadoop

MapR-FS builds on the HDFS strategy. It still distributes data, distributes computation, and tolerates faults, but is more efficient than Hadoop.

MapR-FS Architectural Components

Feature	Description
Storage Pool	Group of disks to which MapR-FS writes
Container	Logical entity stores files and directories
CLDB	Service tracks location of containers
Volume	Management entity stores and organizes containers
Snapshot	Read-only image of volume at specific point in time
Direct Access NFS™	Enables POSIX-compliant access to data on cluster

There are many architectural components which are unique to the MapR File System, which overcome some of the weaknesses of HDFS. We will go over each of these in more detail next.

Storage Pools

A group of physical disks. Default: three

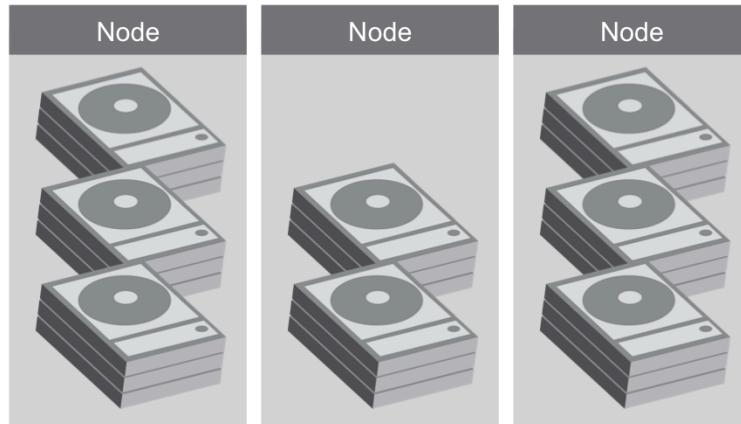


A storage pool is made of several disks grouped together by MapR-FS. This grouping is logical and is different from a RAID. The default number of disks in a storage pool is three.

MapR-FS writes data by striping it across the disks in a storage pool.

Storage Pools

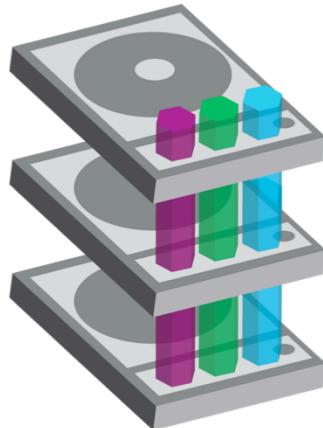
Up to 36 storage pools per node



Each node contains one or more storage pools, and all storage pools across all nodes combine into the overall storage available to the MapR-FS.

Storage Pools

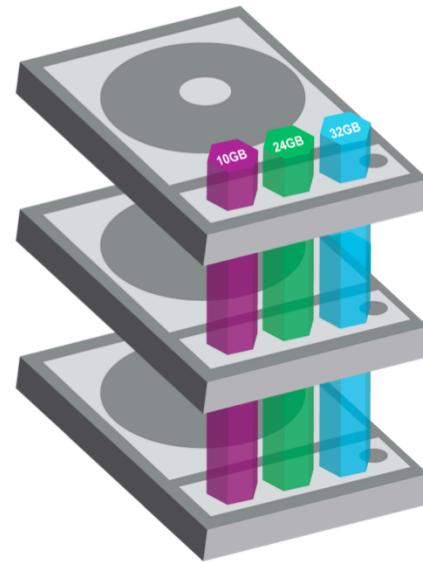
Chunks are written to containers



When data is split, each chunk is striped across storage pools and replicated to other nodes. When data is replicated, MapR-FS will attempt to write the replicas to different storage pools. In other words, MapR-FS tries to keep your data distributed as much as possible.

Containers

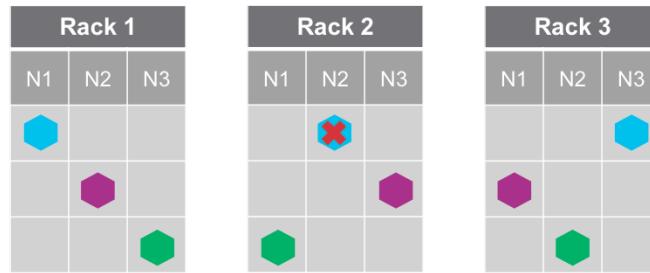
- Data is sharded into 256 MB chunks
- Chunks are written to containers that are typically 10-32 GB
- Containers are striped across storage pools
- Storage pools hold many containers



When data is written to MapR-FS, it is sharded into chunks. The default chunk size is 256 Megabytes. Chunks are striped across storage pools in a series of blocks, into logical entities called containers. Striping the data across multiple disks allows data to be written faster, because the file will be split across the three physical disks in a storage pool, but remain in one logical container.

Containers

Each container is replicated to different racks

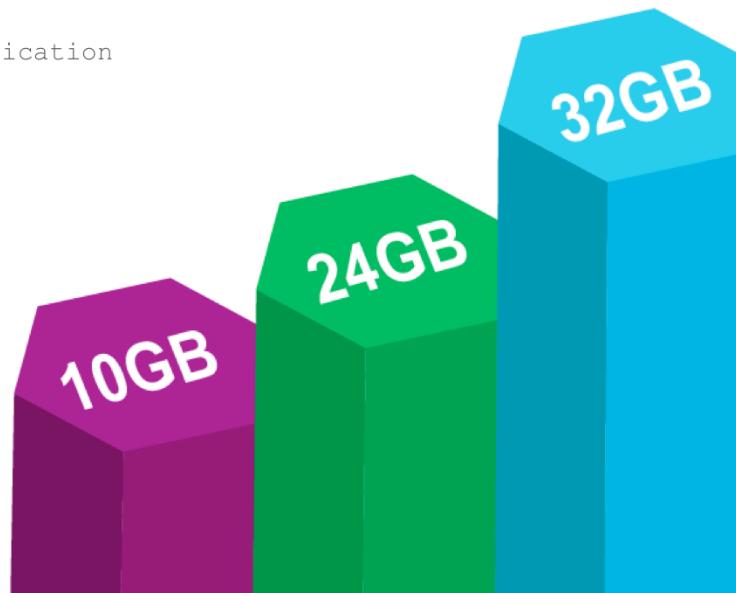


MapR-FS attempts to replicate data to different nodes, in different racks. Replicating to different nodes protects against the loss of a single node, while replicating to different racks means that the data will still be available if a rack experiences a failure, such as a power outage.

In this example, we see that each container is replicated to different racks.

Containers

- Default replication factor: 3
 - `cldb.volumes.default.replication`
- Default size: 32 gigabytes
 - `cldb.container.sizemb`



MAPR Academy

Replication factor is controlled by the `cldb.volumes.default.replication` parameter. You can also adjust this using the MapR Control System GUI.

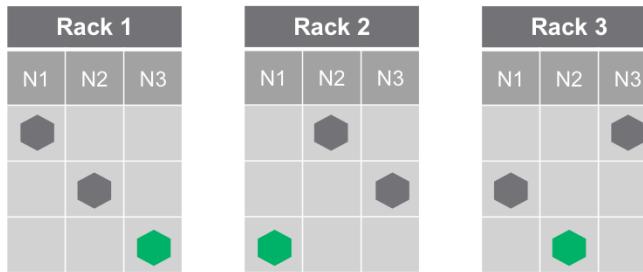
By default, the container size is 32 GB, but a container only occupies disk space when an application or program writes to it. Large containers allow for greater scaling and allocation of space in parallel without bottlenecks.

Container size is controlled by the `cldb.container.sizemb` parameter. Again, you can also adjust this using the MapR Control System GUI.

Volumes

Logically organize data in a cluster.
Control disk usage, replication, snapshots, mirrors

Volume includes all: 



Containers are grouped into volumes. A volume is a logical organization of data that is spread throughout the cluster. All containers and their replicas are distributed across the nodes that the volume can write to.

Volumes are used to enforce disk usage limits, set replication levels, define snapshots and mirrors, and establish ownership and accountability. Since a container always belongs to exactly one volume, that container's replicas all belong to the same volume as well.

In this image, the green hexagons represent containers in a volume.

CLDB

Container Location Database (CLDB) locates containers.
CLDB information is stored on disk in a container and replicated

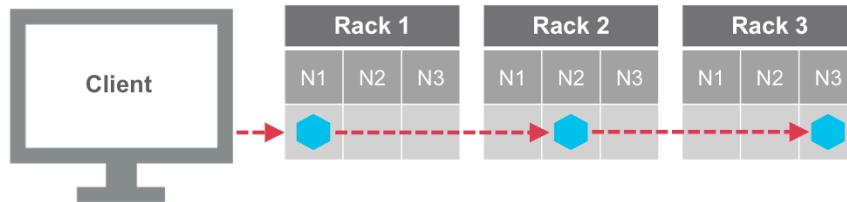


The Container Location Data Base (CLDB) is a management service that locates containers and the root of volumes.

The CLDB keeps track of the locations of containers, as well as the replication chain order.

Replication

Copies of data at the container level
Automatic; controlled by CLDB

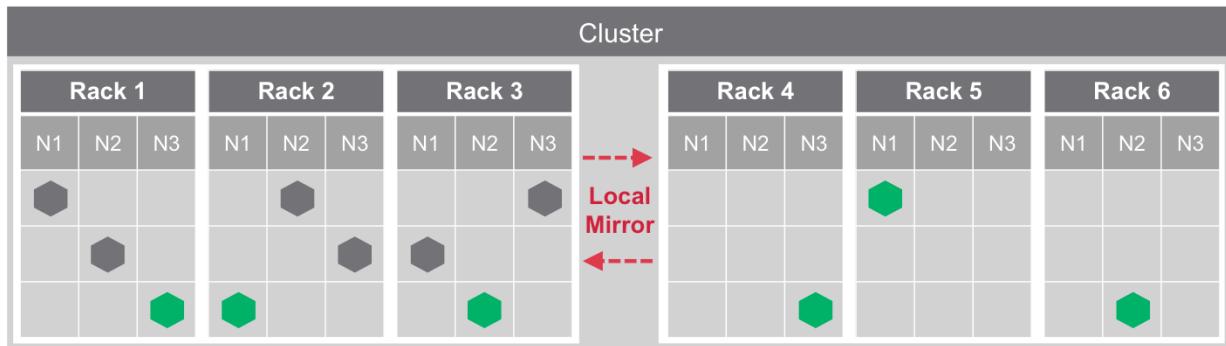


MapR-FS automatically replicates containers across different nodes on the cluster to preserve data. Container replication creates multiple synchronized copies of the data across the cluster for failover. Each container has a replication chain managed by CLDB.

By default, the replication factor is three, but is configurable at a per volume level. Data is first written to the master container, then replicated to other containers.

Mirrors

Copies of data at the volume level
Can be manual or scheduled

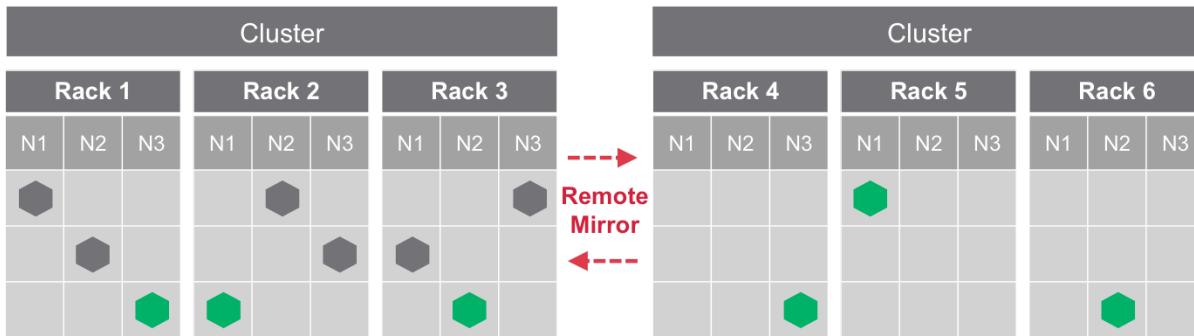


Mirrors are also copies of data, but at the volume level, rather than the container level. Mirrors can be performed manually, or you can schedule them.

A mirror volume is a read-only physical copy of a source volume that is promotable to read/write at any time for disaster recovery.

Mirrors

Copies of data at the volume level
Can be manual or scheduled

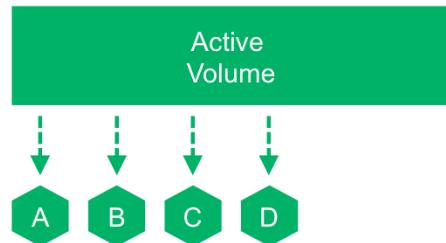


Local mirrors are copied to the same cluster, while remote mirrors are copied to a different cluster. You can mirror data between clusters, data centers, or between on-premise and public cloud infrastructures.

The initial mirroring operation copies the entire source volume. Subsequent mirrors only update the differences between the source volume and the mirror volume. The mirroring operation never consumes all of the available network bandwidth..

Snapshots

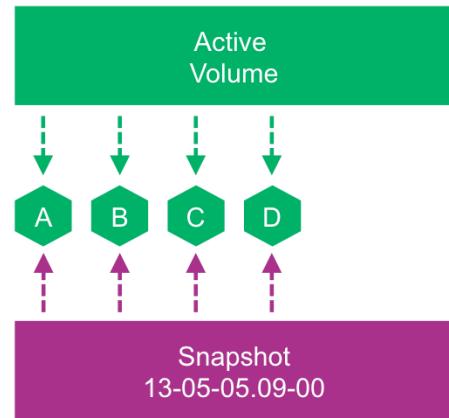
- Points to data at the volume level
 - Does not actually copy data
 - Protects against user error
- Can be manual or scheduled



Snapshots allow you to create point-in-time versions of data while maintaining the on-going consistency of a live volume.

Snapshots

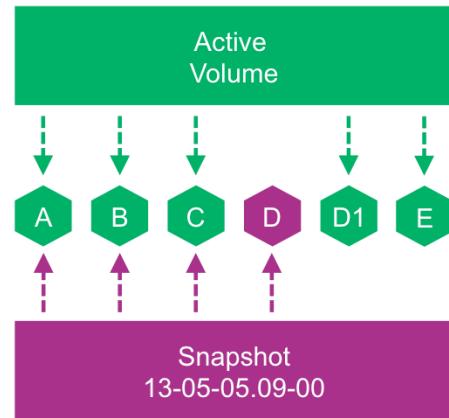
- Points to data at the volume level
 - Does not actually copy data
 - Protects against user error
- Can be manual or scheduled



When a snapshot is taken it does not actually copy data, it only points to it. This is why snapshots are very fast.

Snapshots

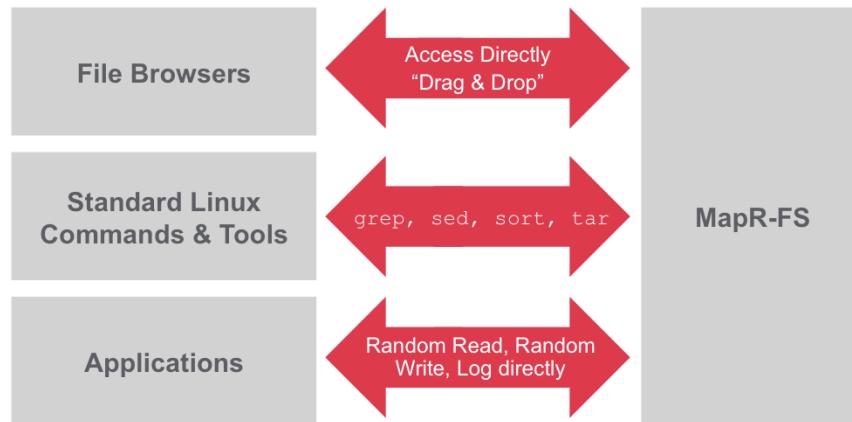
- Points to data at the volume level
 - Does not actually copy data
 - Protects against user error
- Can be manual or scheduled



Snapshots provide a level of protection against user error, and they also allow you to return to a version of a data set at any time. Snapshots can be manual, or recurring snapshots can be set automatically based on a defined schedule.

Direct Access NFS™

POSIX compliant; access cluster data directly using familiar tools



Direct Access NFS allows applications to read and write cluster data directly. This means that MapR-FS can directly process data in an existing NFS File System.

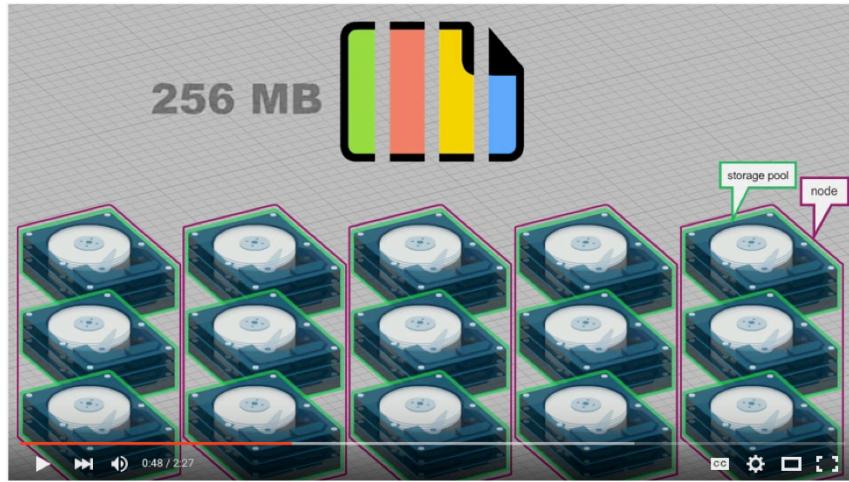
All this is possible because MapR-FS is POSIX-compliant. Your developers and administrators can use familiar POSIX commands to navigate the file system and move, copy, or delete data.

Direct Access NFS also allows any file-based application to access the cluster with no changes or rewrites required. This means your existing Hadoop, SQL, or ODBC applications will work on MapR.

Review



Review



<https://www.youtube.com/watch?v=TOFQq9PHJHk>

Watch this video to review some of the core components of distributed file systems, and how they work on MapR-FS.

Knowledge Check



Knowledge Check



What is one feature of MapR-FS that protects against user error?

- A. POSIX-compliance
- B. Snapshots
- C. Sharding of data
- D. CLDB metadata

Knowledge Check



What is one feature of MapR-FS that protects against user error?

- A. POSIX-compliance
- B. Snapshots**
- C. Sharding of data
- D. CLDB metadata

Answer: B

Snapshots, replication, and mirroring all help keep MapR-FS fault-tolerant.

Learning Goals



Learning Goals



- 6.1 Review Key Components of HDFS
- 6.2 Describe Key Components of MapR-FS
- 6.3 Compare and Contrast MapR-FS and HDFS**

Now, we'll compare and contrast MapR-FS with HDFS.

HDFS and MapR-FS



MapReduce	MapReduce	MapReduce
Bigtable	HBase	HBase/MapR-DB
GFS	HDFS	HDFS/MapR-FS

MapR-FS is a distributed file system. It is the underlying file system for the MapR Converged Data Platform. It supports all the features of local file systems that we discussed in lesson 3, including read/write access, local and remote mirroring, as well as the ability to grow the file system on the fly. Applications that are written for HDFS will also run on MapR-FS. MapR-FS has the same benefits as HDFS, like fault-tolerance, but has been designed to overcome the limitations of HDFS.

Let's take a look at how MapR-FS differs from HDFS.

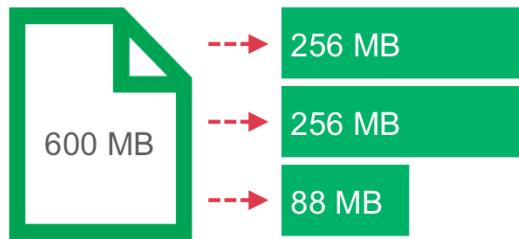
Storage Component Sizes

Storage Components	MapR-FS
Size of file shard	256 MB chunk
Unit of replication	32 GB container
Unit of file I/O	8 KB block

A major difference between HDFS and MapR-FS is the size of different storage components. This table shows the default sizes for shards, replication, and file I/O.

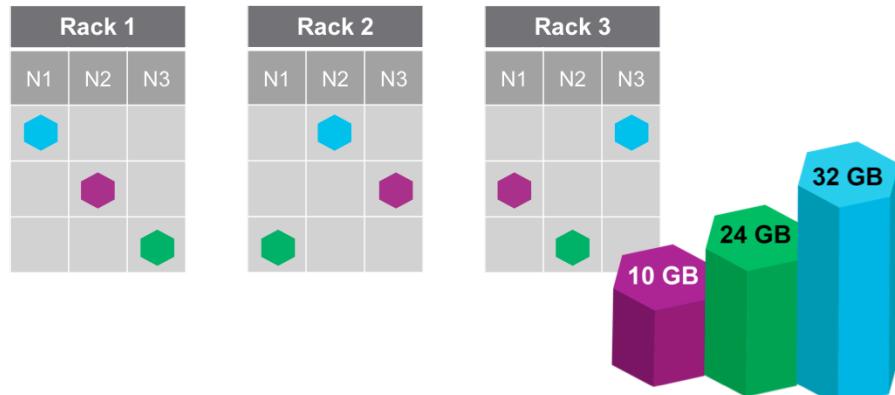
HDFS uses a single block size for sharding, replication, and file I/O. By default, an HDFS block is 128 MB. This can be set when the cluster is configured, but cannot be changed after that.

Storage Component Sizes: Chunks



MapR-FS on the other hand, uses a larger chunk size for file sharding. This shard size is also used in applications like MapReduce and Apache Spark for file partitioning. The default size is 256 MB, but it can be larger. This large file size allows for fast sequential reads in parallel. Having a larger shard size also means that the metadata footprint associated with files is smaller. This permits users to store a larger number of files and make fewer metadata lookups when accessing these files.

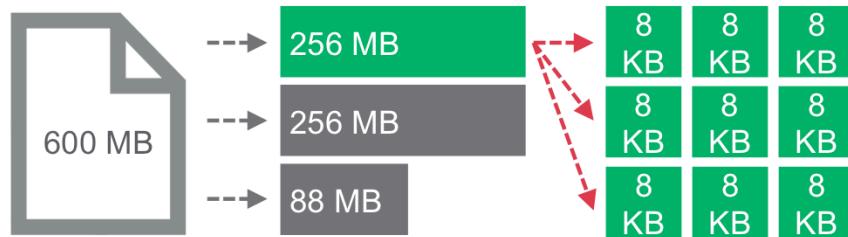
Storage Component Sizes: Containers



MapR-FS replicates containers, which are much larger than HDFS blocks. Having a larger replication unit size means that fewer metadata requests are made. With MapR, the CLDB only has to locate containers, not files and directories. This allows for faster performance.

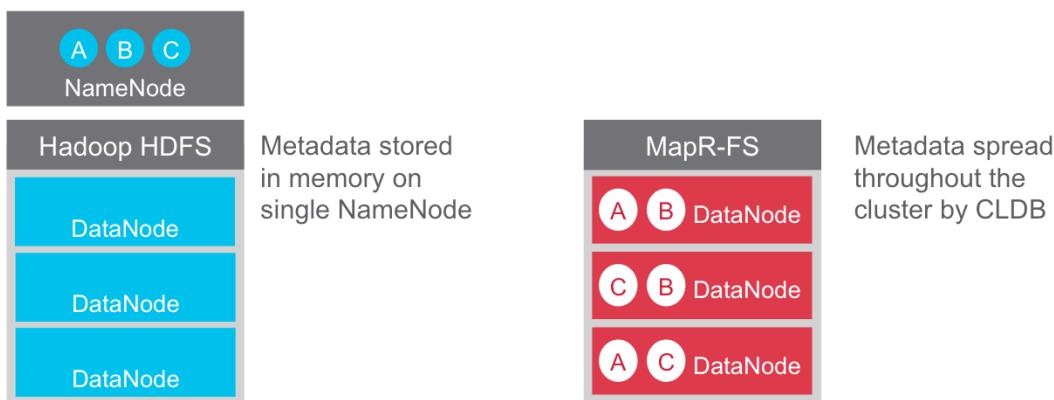
Remember, a container only occupies disk space when data is written to it. This means up to 32GB will be replicated, but not all containers will be at their maximum capacity.

Storage Component Sizes: File I/O



Finally, MapR-FS allows updates to files in increments as small as 8K. Having a smaller I/O size reduces overhead on the cluster, which allows for snapshots, and is one of the reasons that MapR-FS is randomly read/write, even during ingestion. In HDFS, however, files are append-only because of the large I/O size.

Storing and Retrieving Metadata



Another difference between Apache Hadoop and the MapR-FS is in the storing and retrieval of file metadata.

In Apache Hadoop, the NameNode tracks blocks and their locations on data nodes in the cluster. The NameNode is the single namespace for the entire cluster, and the starting point for all file system operations such as reading, writing, or listing files and directories.

MapR-FS distributes and replicates the namespace information throughout the cluster, in the same way that data is replicated. Each volume has a name container, which contains the metadata for the files in that volume. The CLDB service typically runs on multiple nodes in the cluster. CLDB is used to locate the name container for the volume, and the client connects to the name container to access the file metadata.

Since HDFS keeps all of its metadata in a single location, it faces performance bottlenecks. Because MapR-FS distributes its metadata, performance is faster and data is more secure.

Availability and Data Protection	
HDFS	MapR-FS
Replication <ul style="list-style-type: none">• Protect from hardware failures• At least one replica on a different rack	Replication <ul style="list-style-type: none">• Protect from hardware failures• At least one replica on a different rack
	Consistent Snapshots <ul style="list-style-type: none">• Protect from user and application errors• Point-in-time recovery
	Mirrors of Volumes <ul style="list-style-type: none">• Development or production• Disaster recovery back up

Both HDFS and MapR-FS use replication for high availability and fault tolerance. Replication protects from hardware failures. File chunks, table regions and metadata are automatically replicated. There is generally at least one replica on a different rack.

However, MapR-FS provides additional options for fault-tolerance. Snapshots protect against user errors or application failures. Snapshots allow point-in-time recovery. You can read files and tables directly from a snapshot.

MapR-FS also provides the option of mirroring entire volumes. Mirroring provides additional remote disaster recovery back-ups, as well as local load balancing.

These additional options for fault-tolerance make MapR-FS more reliable.

Accessing the File System

HDFS

- Does not support standard POSIX file semantics
- Must use the `hadoop fs` command to use legacy tools re-written with Hadoop

MapR-FS

- Supports standard POSIX semantics
- MapR Direct Access NFS™ adds full r/w access, security, and speed
- Can use legacy applications, tools, and browsers
- Can use standard Hadoop API to access data on MapR-FS from a system outside the cluster

MapR provides a full range of access. If an application is written to run on Hadoop HDFS, it will run on the MapR Platform. With MapR, you also have the ability to run other non-Hadoop applications directly on data from the MapR cluster by making use of standard NFS commands. With MapR, you do not have to re-write legacy code to work with Hadoop. You can use familiar tools such as Linux commands, Python, or Java to work directly with the MapR cluster.

Scalability and Performance

HDFS

- NameNode coordinates files and directories – affects performance
- 100 million files per cluster
- Written in Java

MapR-FS

- Metadata for files and directories fully distributed
- Only limited by disk space available
- Written in C

In HDFS, NameNodes can lead to single point of failure and performance bottlenecks. MapR-FS avoids this problem by fully distributing the metadata for file and directories.

In HDFS, NameNodes scale up to 100 million files per cluster. In MapR, there are no limitations. You can create files as long as there is disk space.

HDFS is written in Java, while MapR-FS written in C. Being written in C means less garbage collection for the operating system, which translates to faster performance.

Review



Review



- **MapR-FS unique architectural components**
 - CLDB
 - Storage pools
 - Containers
 - Volumes
 - Snapshots
 - Direct Access NFS
 - Separate block, container, and file I/O sizes
- <https://www.youtube.com/watch?v=xUihiT9RODk>
- <https://www.youtube.com/watch?v=TiqA9ybgewk>

Let's review.

MapR-FS is an implementation of HDFS, with the addition of some key architectural components which are designed to overcome the limitations of open source Hadoop. These components include the CLDB, storage pools, containers, volumes, snapshots, Direct Access NFS, and separate sizes for blocks, containers, and file I/O.

Watch these videos for another look at some of the similarities and differences between HDFS and MapR-FS.

Knowledge Check



Knowledge Check



Why is MapR-FS faster than HDFS?

- A. Smaller file I/O size
- B. POSIX compliant
- C. Written in C
- D. No NameNode

Knowledge Check



Why is MapR-FS faster than HDFS?

- A. Smaller file I/O size
- B. POSIX compliant
- C. Written in C
- D. No NameNode

Answer: A, B, C, D



Next Steps

Lesson 7: MapR-DB

ESS 102 – MapR Converged Data Platform Essentials

Congratulations! You have completed ESS 102: Lesson 6. Continue on to ESS 102: Lesson 7, to learn about the MapR Database.



ESS 102 – MapR Converged Data Platform Essentials

Lesson 7: MapR-DB

Spring 2016 v5.1

Welcome to ESS 102, Lesson 7: MapR-DB.

Learning Goals



Learning Goals



- 7.1 Compare and Contrast MapR-DB, HBase, and RDBMS
- 7.2 Describe Components and Features of MapR-DB
- 7.3 Understand Table Replication in MapR-DB

When you have finished with this lesson, you will be able to:
describe key components of MapR-DB,
compare and contrast MapR-DB, HBase, and traditional databases,
and understand table replication in MapR-DB.

Learning Goals



7.1 Compare and Contrast MapR-DB, HBase, and RDBMS

7.2 Describe Components and Features of MapR-DB

7.3 Understand Table Replication in MapR-DB

First, we'll learn about the similarities and differences between MapR-DB, HBase, and traditional databases.

HBase and MapR-DB

Google

 hadoop

MAPR®

MapReduce	MapReduce	MapReduce
Bigtable	HBase	HBase/MapR-DB
GFS	HDFS	HDFS/MapR-FS

MapR-DB is a NoSQL database that is part of the MapR Converged Data Platform. Applications that are written for Apache HBase will also run on MapR-DB.

Let's take a look at how MapR-DB and HBase differ from the traditional RDBMS.

Traditional RDBMS: Cost and Scalability

Must upgrade to get additional storage or processing power



Relational databases, or RDBMS, have been around for decades and are the standard for traditional data storage and processing.

To increase the storage or processing power, you might need to buy faster, larger, more expensive machines. Because of this, it becomes expensive and complex to handle the demands of big data.

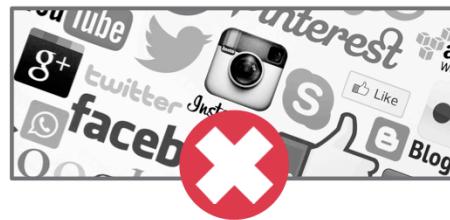
Traditional RDBMS: Schema

Defined Schema

First Name	Last Name	State
John	Doe	IA
Mary	Smith	CO
George		WA



Semi-structured or
Unstructured Data



A relational database also requires a defined schema, which makes it difficult to store and process semi-structured or unstructured data like JSON files.

RDBMS vs. Apache HBase

Feature	RDBMS	Apache HBase
Size	Terabytes	Petabytes
Scaling	Nonlinear	Linear
Fault Tolerance	Low	High
Schema	Static	Dynamic
Updates	In place	Cell versioning

Apache HBase running on a Hadoop cluster overcomes some limitations of relational databases. This table compares HBase with RDBMS.

HBase was designed to run on a cluster of commodity hardware, so it can easily handle petabytes of data, and scales linearly. To increase storage or processing power, add a few more nodes. In general, data stored on Hadoop is more fault tolerant than data stored on traditional databases. HBase does not require a static schema, and can work with all types of data. Finally, HBase uses cell versioning, so you can store multiple versions of the data without losing previous information.

Apache HBase Features

Compaction:

- faster data reads
- introduces latency

RegionServer:

- controls operations, like compaction
- complex administration process

Although HBase has many advantages, but there are some disadvantages as well.

For example, HBase performs an operation called compaction. Compaction is used in HBase to re-write multiple files into fewer larger files. Fewer files means faster reads. However, merging the files requires a large amount of time and computational resources. Additionally, the HBase cluster may be inaccessible during major compactions.

HBase also has a service called a RegionServer, which runs on each node in the Hadoop cluster. The RegionServer can crash, forcing administrators to manually control some operations, like compactions, splits, and merges. Basic administration is complex: renaming a table requires copying all the data. Backing up a cluster is a complex process.

MapR-DB vs. HBase

No Compaction

- MapR-DB uses fewer computational resources
- new data can be merged without re-writing files

No RegionServer

- controls operations, like compaction
- complex administration process

MapR-DB is fast and reliable, without the need for compacting files.

Since MapR-FS is read/write, it does not require compaction to increase performance. New data can be written or merged into existing files without having to overwrite old data. Since MapR-DB does not use compaction, there are no delays.

MapR-DB is optimized to offer a high level of availability for distributed storage. Since there is no RegionServer, there are also fewer administrative processes to manage, and better protection of data. MapR recovers from failure very fast and provides snapshots and mirrors for data recovery.

MapR-DB Advantages

Feature	MapR-DB	HBase
99.999% High Availability	✓	✗
Instant Recovery from Failures	✓	✗
Continuous Low Latency (No Compaction Delays)	✓	✗
Less Administration (No Processes to Manage, Self-tuning)	✓	✗
Online Data Protection (Snapshots, Mirroring)	✓	✗
Scalability (Number of Tables Supported)	Trillions	Hundreds

MapR-DB is optimized to offer a high level of availability for distributed storage.

MapR recovers from failure very fast and provides snapshots and mirrors for data recovery.

Since MapR-DB does not use compaction, there are no delays.

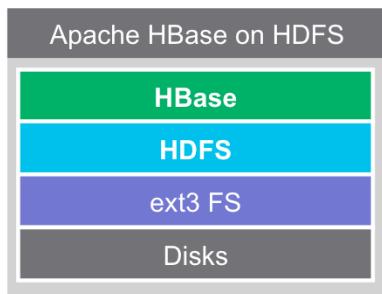
There are also fewer administrative processes to manage.

Features like snapshots and mirrors provide better protection of data.

Finally, MapR-DB is massively scalable. It can support trillions of tables. Unlike HBase, which supports about three column families, MapR-DB can support up to 64 column families in a table.

Standard Hadoop with HBase

- Disks with some file system
- HDFS interacts with file system via JVM
- HBase interacts with HDFS via JVM



This diagram shows the standard Apache HBase on Apache Hadoop application stack.

We start with physical disks, formatted with a file system such as ext3. HDFS runs on top of the underlying file system. HDFS does not interact directly with the disks – it goes through a Java Virtual Machine (JVM) in order to read and write files on the file system.

Finally, HBase sits on top of HDFS. The HBase API is used to communicate with HDFS, via JVM.

MapR-FS with HBase

MapR-FS:

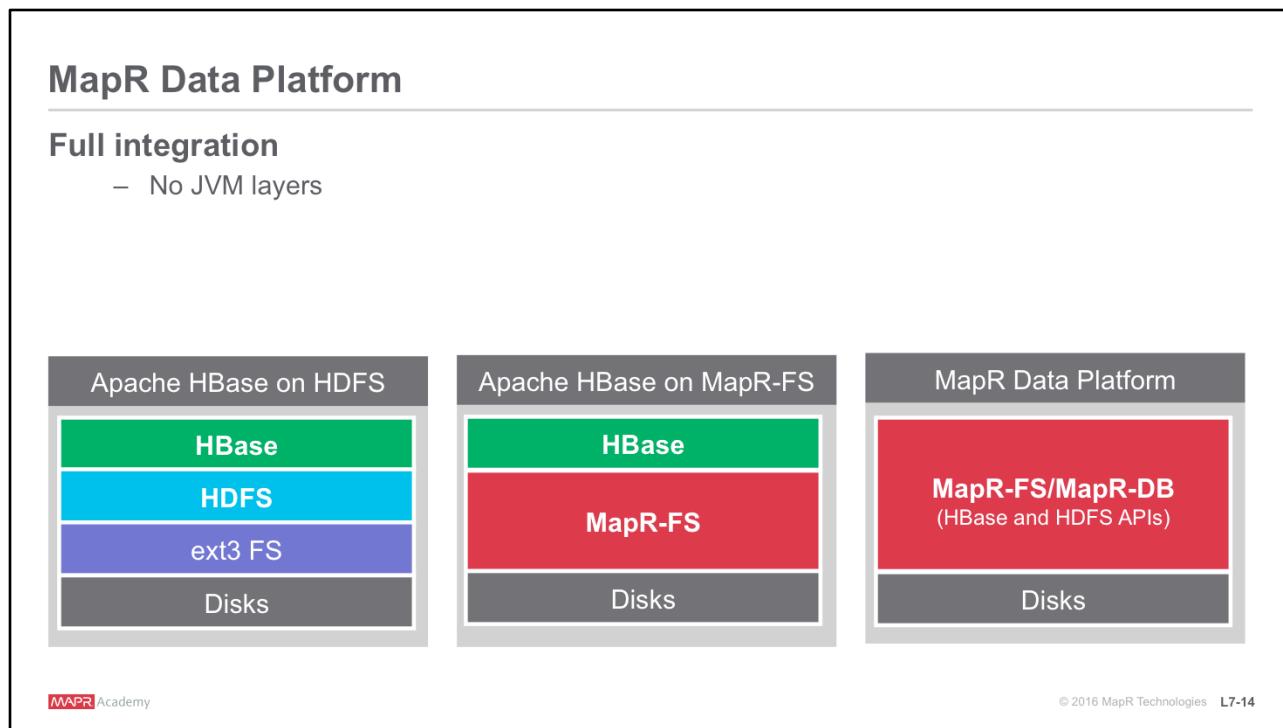
- Replaces other file systems
- Is POSIX-compliant; no JVM needed to interact with disks
- Uses HDFS API; HBase applications will work



This diagram shows how Apache HBase works with MapR-FS.

Since MapR-FS is a POSIX-compliant, read/write file system, it can act as the underlying file system – replacing ext3 and HDFS, for example. This eliminates one layer of complexity, as MapR-FS is working directly with the disks.

HBase then sits on top of MapR-FS, just as it does with HDFS. MapR-FS is compatible with the HDFS API, so any HBase applications you have already written for HDFS will just work with MapR-FS.



In this diagram, HBase has been replaced by MapR-DB. MapR-DB is integrated into MapR-FS, providing seamless communication. MapR-DB is API-compatible with HBase, so existing applications written for HBase will just work with MapR-DB.

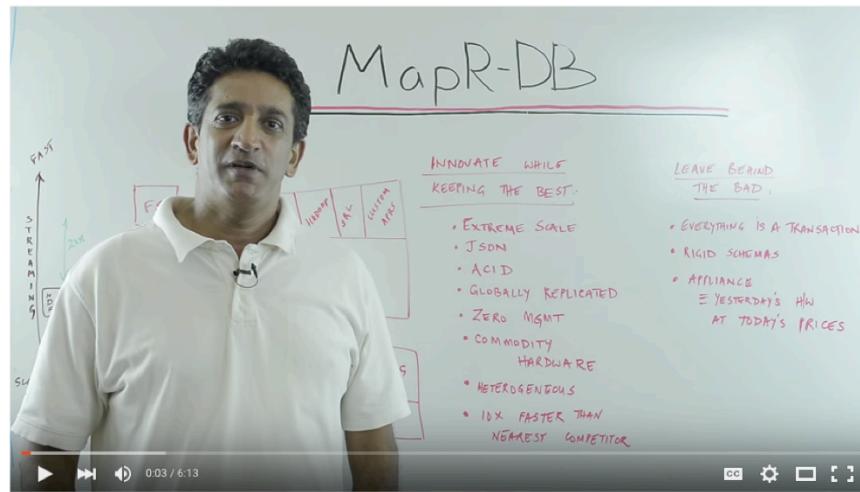
Using MapR-FS and MapR-DB provides a tightly integrated solution that reduces complexity and increases performance, without needing to rewrite existing HDFS or HBase applications.

You can move and delete tables and files using Direct Access NFS or hadoop commands. Direct Access NFS allows you to use standard POSIX commands to navigate data in MapR-FS and MapR-DB.

Review



Review



<https://www.youtube.com/watch?v=DR32AOJwG4Q>

Watch this video to learn a little bit about the motivation behind MapR-DB.

Knowledge Check





Knowledge Check

Match the feature of MapR-DB with its benefit.

Scalability

no compaction required

Low Latency

tablets replicated in MapR-FS containers

High Availability

compatible with HBase and Hadoop API

Legacy Apps

supports trillions of rows

Scalability :: supports trillions of rows

Low Latency :: no compaction required

High Availability :: tablets replicated in MapR-FS containers

Legacy apps :: compatible with HBase and Hadoop API

Learning Goals



Learning Goals



7.1 Compare and Contrast MapR-DB, HBase, and RDBMS

7.2 Describe Components and Features of MapR-DB

7.3 Understand Table Replication in MapR-DB

Next, we'll describe some of the components and features of MapR-DB.

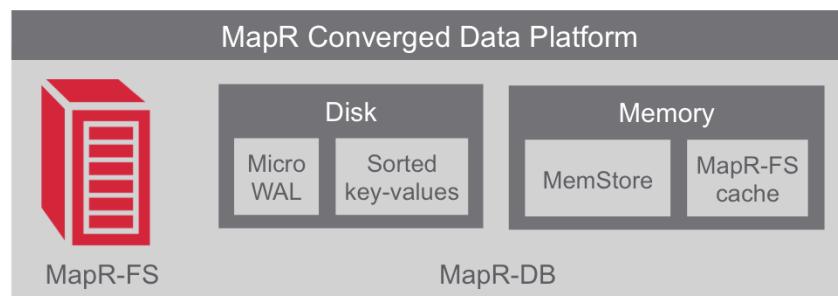
What is MapR-DB?

- NoSQL database on Hadoop
- Part of the MapR Converged Data Platform
- Uses HBase API

MapR-DB is a NoSQL database on Hadoop and part of the MapR Converged Data Platform. MapR-DB provides the same API as Apache HBase.

MapR-DB Architecture

MapR-DB takes advantage of the features of MapR-FS

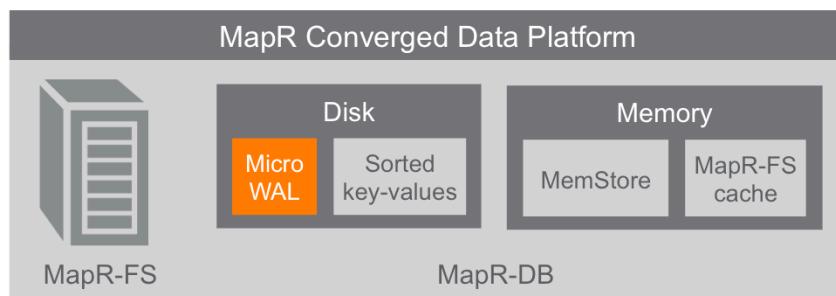


Here are some of the components of MapR-DB.

First, MapR-DB takes advantage of the read/write file system of MapR-FS. All the features of MapR-FS, like replication and volume permissions, are available in MapR-DB.

MapR-DB Architecture

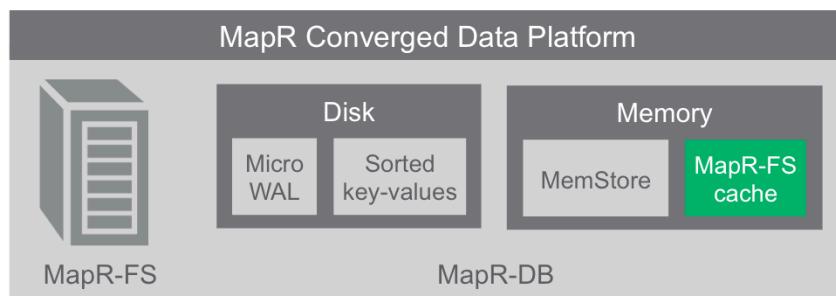
Small write-ahead-logs (WALs) for faster performance



Second, it uses very small write-ahead-logs, or WALs, for faster recovery. These WALs are stored on disk.

MapR-DB Architecture

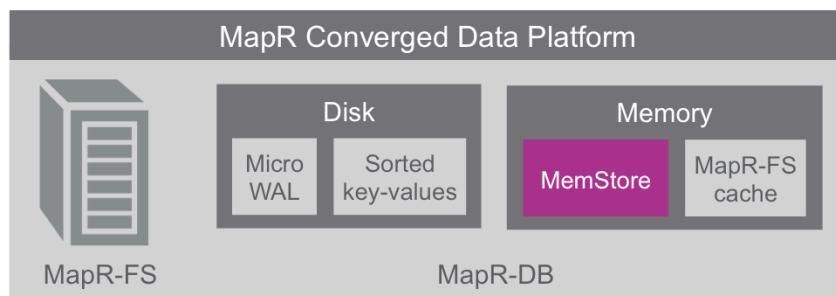
Uses MapR-FS Cache for in-memory computations



Third, it uses the MapR-FS Cache for in-memory computations, for very fast reads and writes. The file system cache is used by both tables and regular files.

MapR-DB Architecture

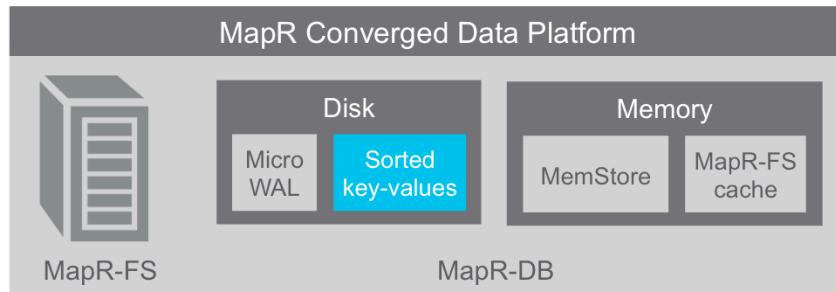
Stores key-value updates in the MemStore



The MemStore stores sorted key-values in memory. Like HBase, there is one MemStore per column family. However, MapR-DB supports more column families than HBase.

MapR-DB Architecture

Stores sorted key-values on disk



Finally, it stores sorted key-value pairs on disk in files on MapR-FS.

Logical Model of MapR-DB Tables

Feature	Description
Rowkey	Identifies row in table; max supported row key size = 64KB
Row	Spans one or more column families and qualifiers; max supported size of row = 2 GB
Column Family	A group of column qualifiers; MapR-DB tables support up to 64 column families
Column Qualifier	Identifies a column within a column family
Value	Data written to a cell in a row

The logical model of a MapR-DB table is similar to that of an HBase table. Keys identify the rows in a table and can be up to 64 kilobytes.

Rows span one or more column families and can contain up to two gigabytes of data.

A column family is a key associated with a set of column qualifiers. MapR-DB tables support up to 64 column families.

Column qualifiers identify a column within a column family.

A value is the data written to a cell in a row.

You can learn more about the data model of HBase and MapR-DB tables in DEV 320 and DEV 325 on learn.mapr.com.

Logical Model of MapR-DB Tables

MapR-DB tables:

- Similar to Apache HBase schema design
- Integrate data from MapR-DB with MapR-FS

MapR-DB tablets:

- Logical structures equivalent to HBase regions
- Treated like other data in MapR-FS

	ID	Address			Name	
		Street	City	Zip Code	Last	First
Tablet 1	2413	Main St		94105	Singh	
	...					
Tablet 2	2481	First St			Johnson	Mike
	3524					
Tablet 3	3601		Grand Ave	Chicago	60707	Garcia Raul
	4012					
	...	Park Dr		02215	Weber	
	4119					

Schemas for MapR-DB tables follow the same general principles as schemas for standard Apache HBase tables.

Since MapR-DB tables can use up to 64 column families, you can make more extensive use of the advantages of column families, which can lead to more optimized querying.

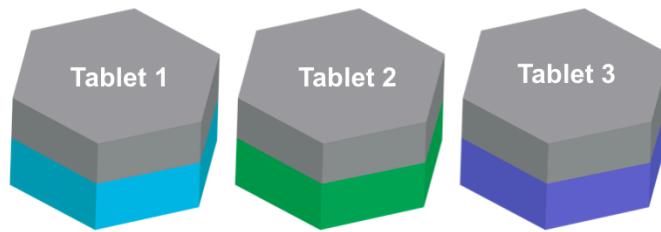
One difference between MapR-DB and HBase is the terminology used with tables. In MapR-DB, tables are broken up into tablets, and are replicated across MapR-FS nodes.

In HBase, tables are broken into regions, and are replicated across region servers. Both tablets and regions are logical organizations of data.

These tablets are stored in the same namespace as other files in MapR-FS. You can use snapshots, replication, mirroring, and volume controls on your tables and tablets, just like you can with any other kind of data.

Benefits of Tablets Inside Containers

- Tablets live inside containers
- Tablets make clusters
 - Highly available
 - Efficient



There are two major advantages to storing regions in containers: scalability of clusters and high availability for your data.

Each tablet of a table is stored in one container.

Containers are replicated and distributed to different nodes in the same cluster. When updates are applied to the tablet, the update is also applied to the replicas of that container. This adds to the high availability of your data.

The location of containers is tracked by that cluster's CLDB, which makes accessing your data faster and more efficient. Since the number of updates necessary for CLDB is much lower than for Apache NameNode, the CLDB is smaller and less data has to travel on your network.

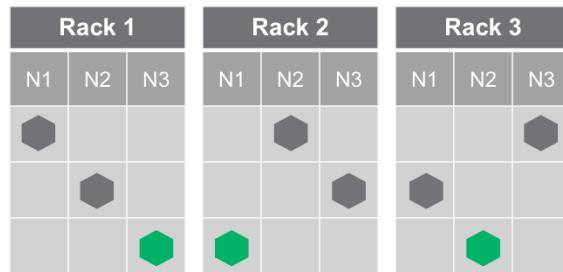
Advantage of Containers in Volumes

- Isolate sensitive data or applications
- Isolate work environments: different users and applications
- Run multiple jobs with different requirements without conflict

/user/john = 



/projects/ads = 



As mentioned in lesson 6, containers reside in volumes. One feature of MapR is the volume management capability which logically segments your data. This allows applications that require different data sets or user permissions to reside in the same cluster.

There are many reasons to organize your MapR-DB tables into volumes. For example, you can use data placement to keep personally identifiable information on nodes that have encrypted drives.

You can also isolate work environments for different database users or applications. For example, you might place MapR-DB tables on specific hardware for better performance, or restrict access to certain users.

Some applications may require different quotas or access privileges on the same cluster. You can run multiple jobs with different requirements without conflict using volumes.

Learning Goals



Learning Goals

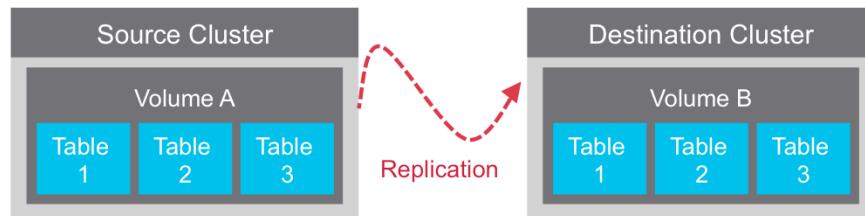


- 7.1 Compare and Contrast MapR-DB, HBase, and RDBMS
- 7.2 Describe Components and Features of MapR-DB
- 7.3 Understand Table Replication in MapR-DB**

Finally, we'll learn a little bit about table replication in MapR-DB.

Table Replication

Source tables reside on source cluster
Replica tables stored on the destination cluster



Replication is one way MapR ensures high availability of your data.

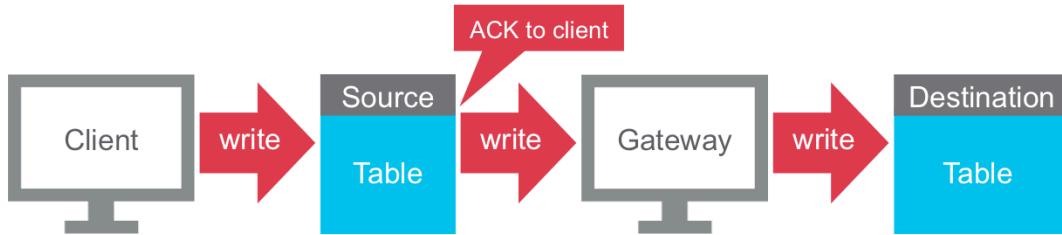
Tables that data are replicated from are called source tables, and they reside on your source cluster.

When MapR replicates a table, the copy is called a replica. These replicas are stored on the destination cluster.

A single cluster can be both a source cluster and a destination cluster, depending on how replication is configured. There are many different ways to configure table replication; we'll go over a few of them next.

Replication Mode: Asynchronous

- Default replication mode
- Confirmation sent to client after source table updated
- Well suited for clusters that are geographically separated

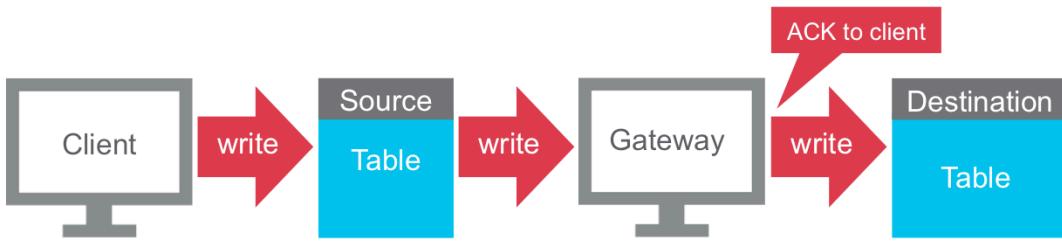


Asynchronous replication is the default replication mode in MapR-DB. When an update is sent, confirmation is sent to the client after the source table is updated. The update is then replicated in the background.

This type of replication is well suited for clusters that are geographically separated in wide-area networks.

Replication Mode: Synchronous

- Confirmation sent to client only after:
 - Change sent to all container copies in source cluster
 - Change sent to a gateway in destination cluster
- Well suited for creating a backup of data for disaster recovery



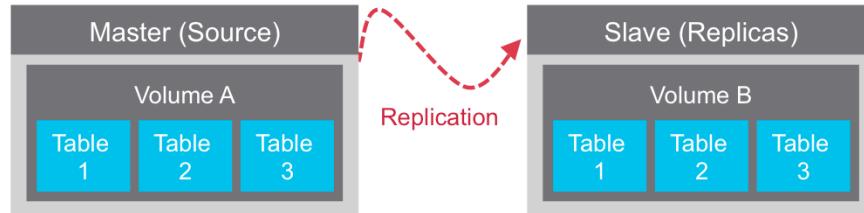
In synchronous mode, MapR-DB confirms to client applications that changes have been applied only after these two conditions are met:

- The change was sent to all of the container copies in the local cluster, and
- The change was sent to a gateway in the destination cluster. This operation takes place only after the first.

Synchronous replication is especially well suited for creating a backup of your data for disaster recovery.

Replication Topology: Master-Slave

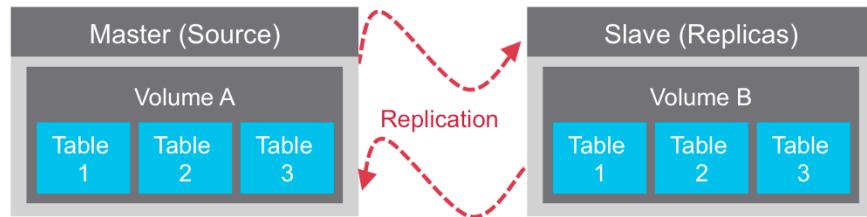
- Changes made to source propagated to replica
- Changes made to replica **not** propagated back to source
- Tables may not be identical



In this figure, we see an example of master-slave replication. Changes made to the master or source tables are copied to the slave or replica tables. However, if you change the replica tables, these are not updated on the source tables.

Replication Topology: Multi-Master

- Changes made to either table propagated to the other
- Tables always identical



In multi-master replication, there are two identical copies of a table, often on different clusters. Applications can change any copy of the table, and MapR-DB replicates the changes to the other copy.

Knowledge Check





Knowledge Check

Match the statement to the replication type:

Asynchronous Mode

Changes made to source and replica tables will be propagated to each other

Synchronous Mode

Changes made to replica tables will not be propagated to the source

Master-Slave Topology

Changes are confirmed once they have been written to the source and sent to the destination gateway

Multi-Master Topology

Changes are confirmed once they have been written to the source

Asynchronous Mode :: Changes are confirmed once they have been written to the source.

Synchronous Mode :: Changes are confirmed once they have been written to the source and sent to the destination gateway.

Master-Slave Topology :: Changes can be made to both source and replica tables, but changes made to the replica will not be propagated to the source.

Multi-Master Topology :: Changes can be made to both source and replica tables, and changes made in one will always be propagated to the other.



Next Steps

Lesson 8: MapR Streams

ESS 102 – MapR Converged Data Platform Essentials

Congratulations! You have completed ESS 102: Lesson 7. Continue on to ESS 102: Lesson 8, to learn about MapR Streams.



ESS 102 – MapR Converged Data Platform Essentials

Lesson 8: MapR Streams

Spring 2016 v5.1

Welcome to ESS 102, Lesson 8: MapR Streams.

Learning Goals



Learning Goals



- 8.1 Compare and Contrast Real-Time and Batch Processing
- 8.2 Describe Key Components of MapR Streams

When you have finished with this lesson, you will be able to:
compare and contrast real-time and batch processing,
and describe key components of MapR Streams.

Learning Goals

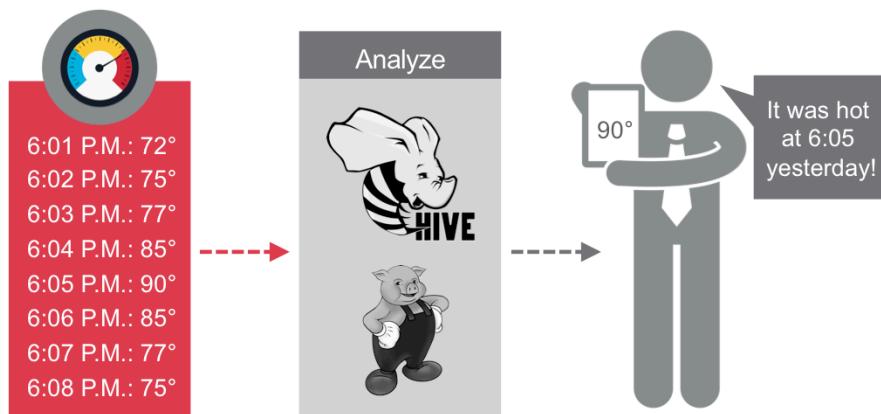


8.1 Compare and Contrast Real-Time and Batch Processing

8.2 Describe Key Components of MapR Streams

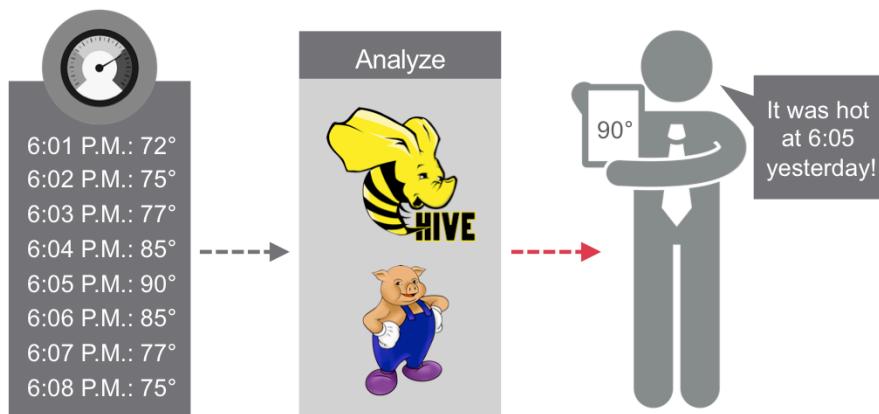
First, we'll learn about the differences between real-time and batch processing.

Batch Processing with HDFS



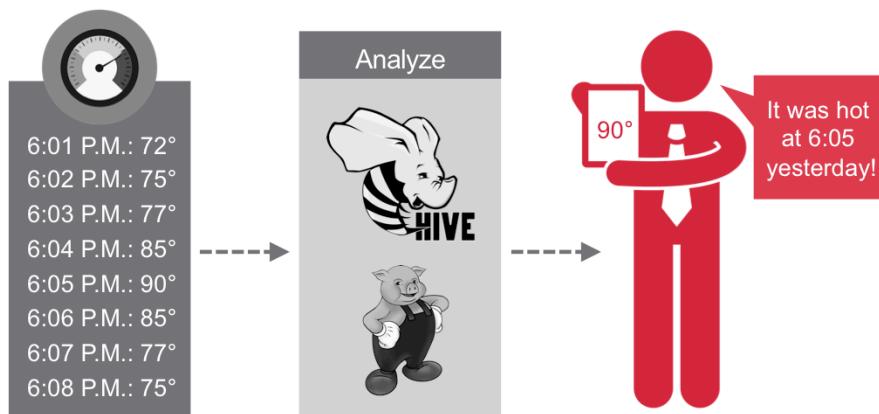
Big data is typically sent in batches to HDFS,

Batch Processing with HDFS



It is then analyzed by distributed processing frameworks like MapReduce, Hive or Pig.

Batch Processing with HDFS



Batch processing can give great insights into things that happened in the past, but lacks the ability to answer the question of "what is happening right now?"

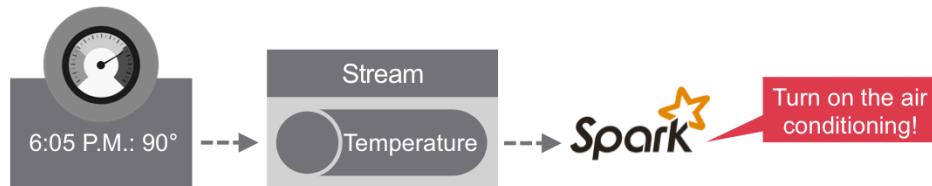
Analyze Data

What if you need to analyze data as it arrives?



What if you want to analyze data as it arrives?

Event Processing with MapR Streams



It is becoming important to process events as they arrive for real-time insights. Analyzing data as it arrives requires several distributed applications to be linked together in real-time.

In this example, MapR Streams helps provide real-time insights based on sensor data, notifying the user to turn on the air conditioning!

Why MapR Streams?

Many big data sources are event-oriented



Trigger Events:

- Stock Prices
- User Activity
- Sensor Data

Many big data sources are event-oriented. For example, stock prices, user activity, and sensor data from an oil rig are trigger events.

Why MapR Streams?

Today's applications need to:

- process high-velocity data as soon as possible
- handle high-volume workloads

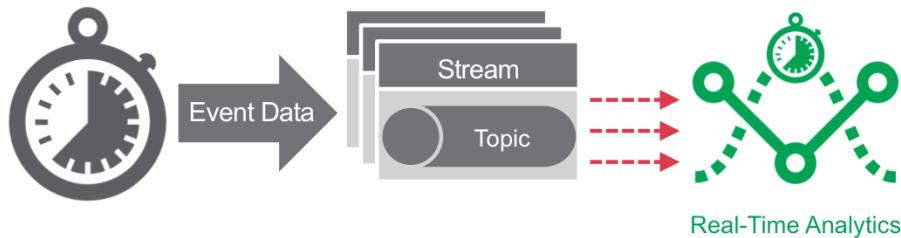


Today's applications need to process this high-velocity, high-volume data in real-time to gain insights.

Why MapR Streams?

Today's applications need to:

- process high-velocity data as soon as possible
- handle high-volume workloads
- provide real-time results with extremely low latency



MapR Streams adds event streaming to the MapR platform, allowing events to be ingested, moved, and processed as they happen, as part of a real-time data pipeline.

Knowledge Check



Knowledge Check



When should you use batch processing?

- A. You have a large volume of data
- B. You need results in real time
- C. You are processing historical data
- D. Your data source is the Internet of Things

Knowledge Check



When should you use batch processing?

- A. You have a large volume of data
- B. You need results in real time
- C. You are processing historical data**
- D. Your data source is the Internet of Things

Both batch and streaming is appropriate for large volumes of data.

Batch processing is great for processing historical data.

If your data source is high-velocity, like the Internet of Things, or you need results in real time, you should use MapR Streams.

Learning Goals



Learning Goals



8.1 Compare and Contrast Real-Time and Batch Processing

8.2 Describe Key Components of MapR Streams

Now let's learn about how MapR Streams works.

What is MapR Streams?

- A global publish/subscribe streaming system for big data
- Built on top of the MapR Platform

MapR Converged Data Platform

Utility-Grade Platform Services



MapR Streams provides global, Internet-of-Things-scale publish-subscribe event streaming integrated into the MapR platform

What is MapR Streams?

- A global publish/subscribe streaming system for big data
- Built on top of the MapR Platform

MapR Converged Data Platform

Utility-Grade Platform Services

Web-Scale Storage

MapR-FS

Database

MapR-DB

Event Streaming

MapR Streams

High Availability

Real Time

Unified Security

Multi-tenancy

Disaster Recovery

Global Namespace

—alongside MapR's distributed file system (MapR-FS)

What is MapR Streams?

- A global publish/subscribe streaming system for big data
- Built on top of the MapR Platform

MapR Converged Data Platform

Utility-Grade Platform Services

Web-Scale Storage

MapR-FS

Database

MapR-DB

Event Streaming

MapR Streams

High Availability

Real Time

Unified Security

Multi-tenancy

Disaster Recovery

Global Namespace

and NoSQL database (MapR-DB)–

What is MapR Streams?

- A global publish/subscribe streaming system for big data
- Built on top of the MapR Platform

MapR Converged Data Platform

Utility-Grade Platform Services

Web-Scale Storage

MapR-FS

Database

MapR-DB

Event Streaming

MapR Streams

High Availability

Real Time

Unified Security

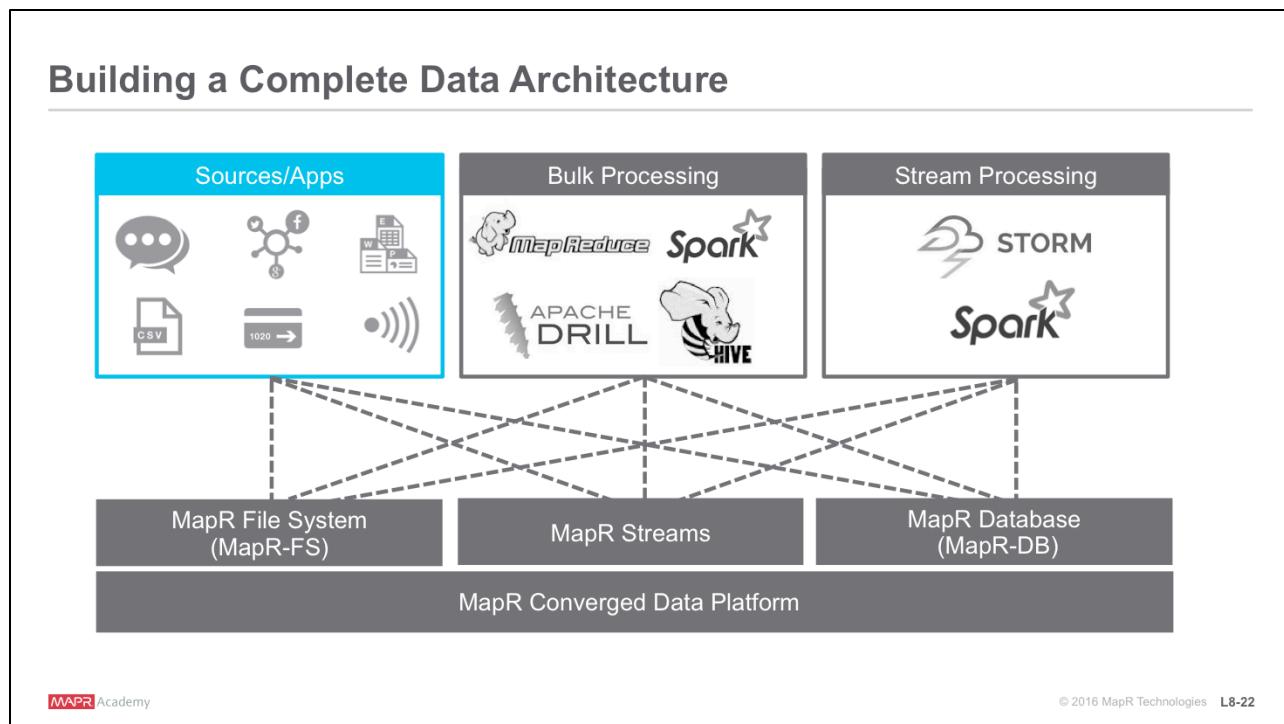
Multi-tenancy

Disaster Recovery

Global Namespace

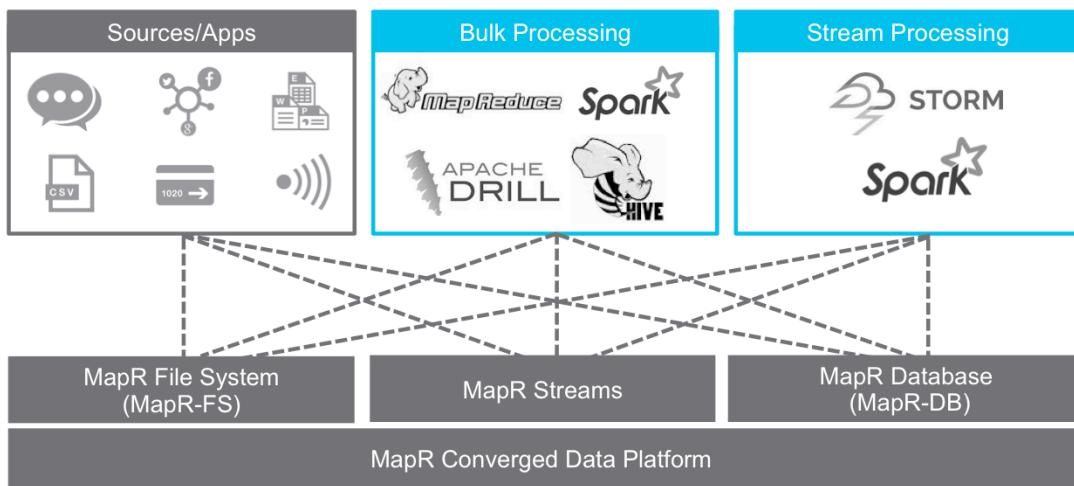
creating the industry's first Converged Data Platform.

The MapR Converged Data Platform integrates Apache Hadoop, Apache Spark, real-time database capabilities, global event streaming, and big data enterprise storage, for developing and running innovative data applications.

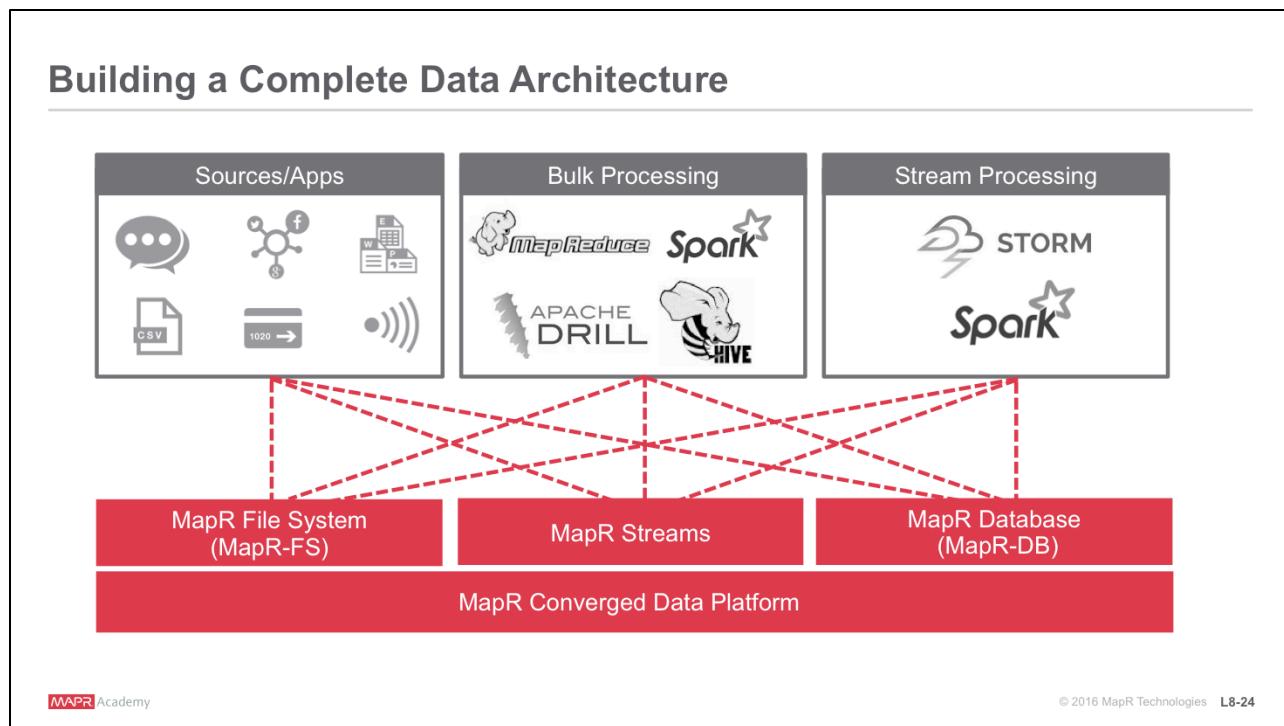


A complete big data pipeline includes many different components. First, you need data sources. In our pipeline, this includes the oil well sensors and other data-generating applications known as producers.

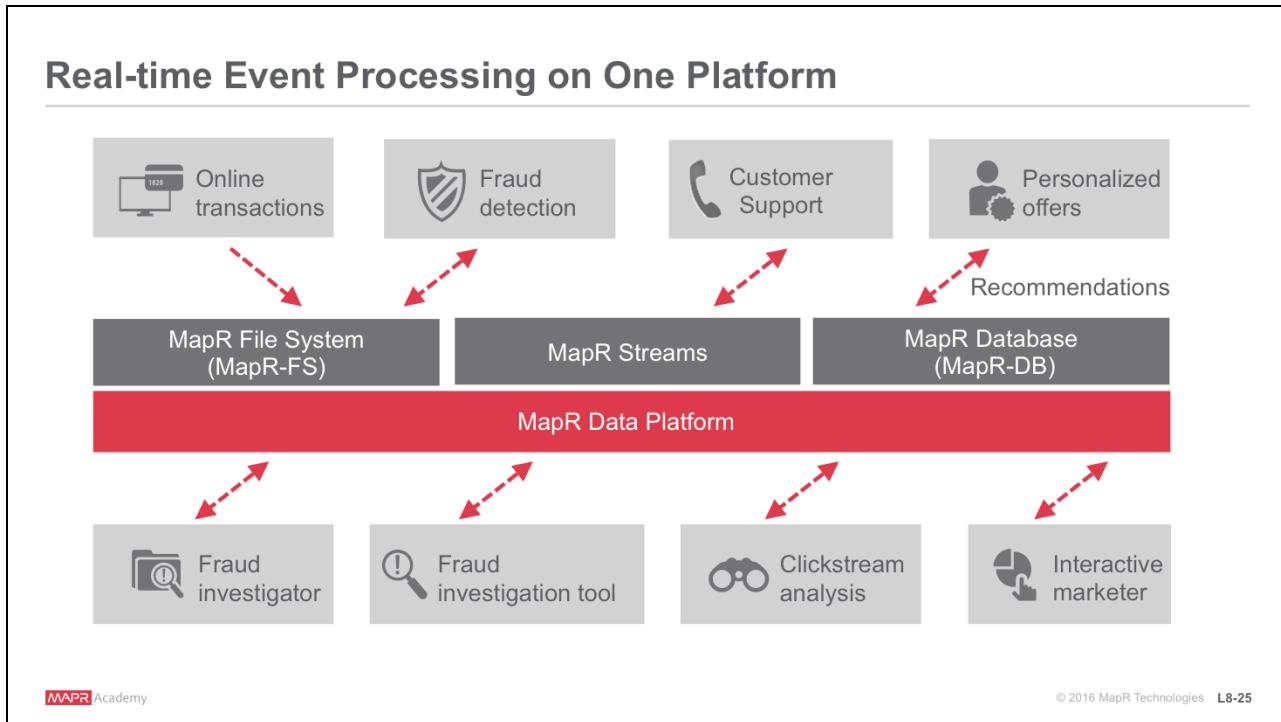
Building a Complete Data Architecture



A big data architecture will also include stream processing applications like Apache Storm and Apache Spark, as well as bulk processing applications like Apache Hive or Apache Drill. It may also include end user applications like dashboards to display data or issue alerts.



All of these components must interact with each other. MapR Streams acts as a message bus between these components. MapR Streams manages sending and receiving data between the many components of a complete data architecture. There are several advantages of having MapR Streams on the same cluster as all the other components. For example, maintaining only one cluster means less infrastructure to provision, manage, and monitor. Likewise, having producers and consumers on the same cluster means fewer delays related to copying and moving data between clusters, and between applications.

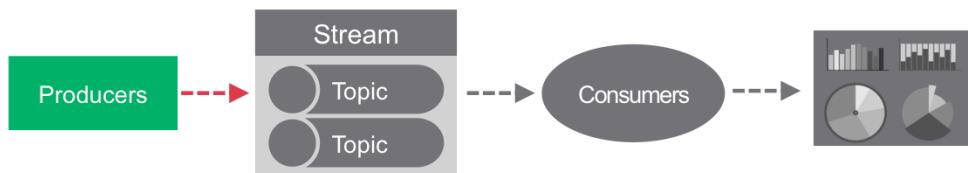


MapR Streams enables a faster closed-loop process between operational applications and analytics by reliably running both operational and analytical applications on one platform.

In this example, the clickstream data from online shopping is being used simultaneously by several different applications. The browsing behavior from the online shoppers is the primary data source. This data is analyzed by several different applications, including fraud models and recommendation tools, all on a single cluster.

How Does MapR Streams Work?

Producers publish messages to a topic

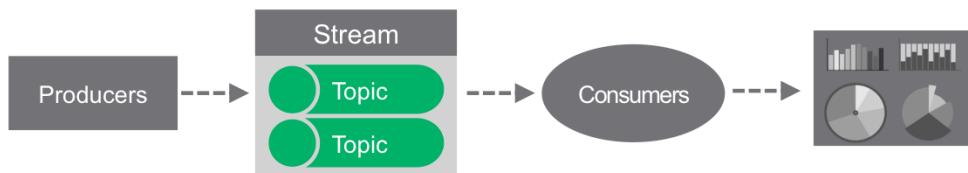


Producer applications generate data. This might be sensors like oil rigs which are connected to the Internet of Things.

Producer applications publish messages to topics.

How Does MapR Streams Work?

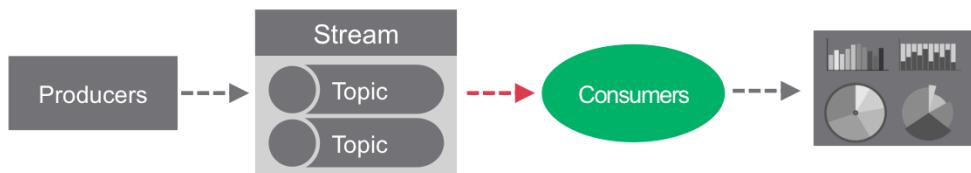
Topics organize events into categories



Topics organize events into categories. Topics are partitioned for scalability and replicated for reliability.

How Does MapR Streams Work?

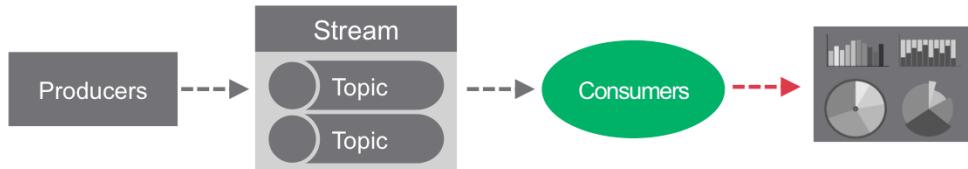
Consumers subscribe to topics with Kafka API



Consumers can subscribe to a topic using industry standard Kafka APIs.

How Does MapR Streams Work?

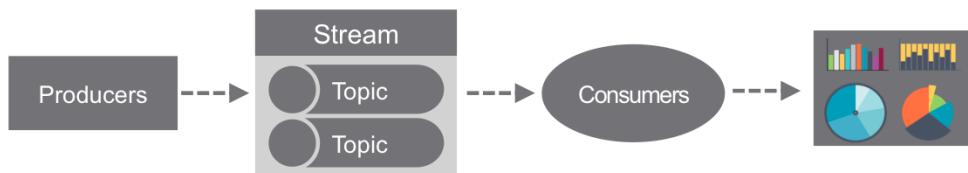
MapR Streams delivers to consumers in real-time



MapR Streams handles requests for messages from consumers for topics that they are subscribed to. Global delivery happens in real time.

How Does MapR Streams Work?

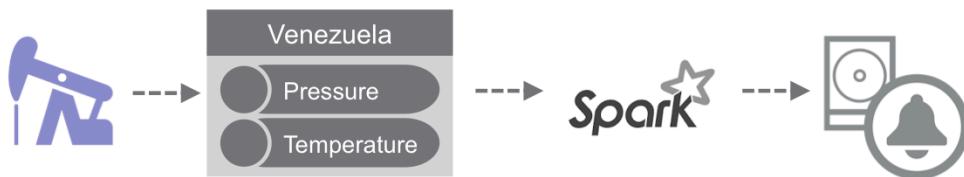
Consumers process data at their own pace



Finally, consumer applications can read those messages at their own pace. All messages published to MapR Streams are persisted, allowing future consumers to “catch-up” on processing, and analytics applications to process historical data.

How Does MapR Streams Work?

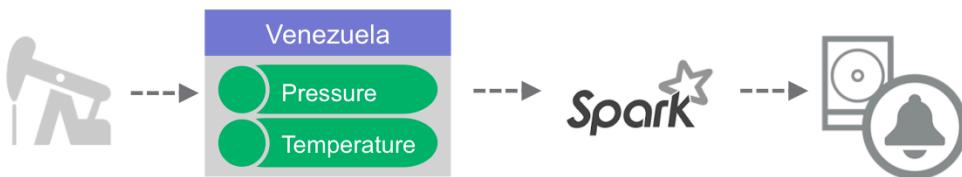
Build a complete data pipeline



MapR Streams can be part of a complete data pipeline. In this example, sensors from oil rigs send data. These sensors are the data producers.

How Does MapR Streams Work?

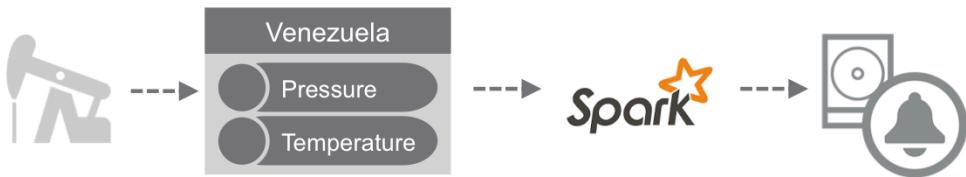
Build a complete data pipeline



This data is published to a stream. In this example, our stream is called Venezuela. The stream contains topics that organize the data into categories: pressure and temperature.

How Does MapR Streams Work?

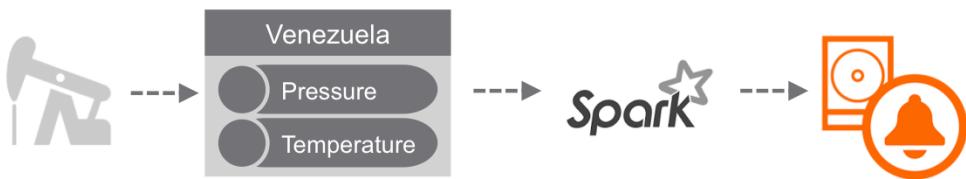
Build a complete data pipeline



An Apache Spark consumer application subscribes to the data.

How Does MapR Streams Work?

Build a complete data pipeline



If the pressure or temperature become too high or low, an alert is issued.

Knowledge Check



Knowledge Check



How does MapR Streams primarily organize data?

- A. Topics
- B. Partitions
- C. Replication
- D. Subscriptions

Knowledge Check



How does MapR Streams primarily organize data?

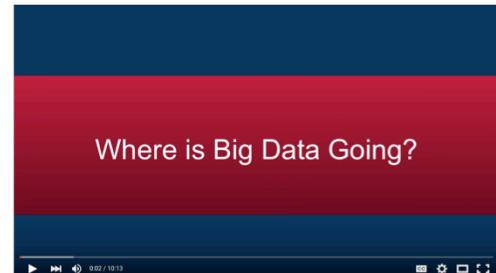
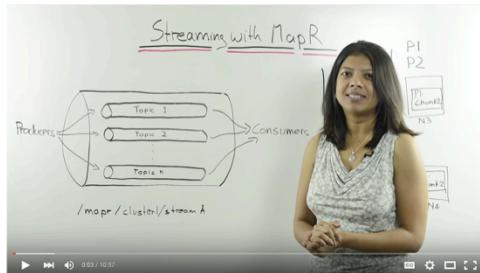
- A. Topics
- B. Partitions
- C. Replication
- D. Subscriptions

Answer: A

Review



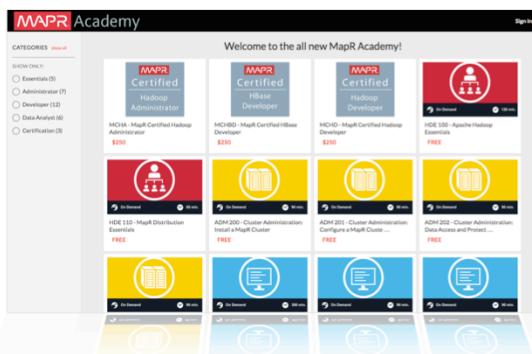
Review



<https://www.youtube.com/watch?v=KS2cB54OiUg>

Watch these short videos to learn a little bit about the history of big data, open source Hadoop, and the MapR Converged Data Platform.

Congratulations!



You have completed this course.

Register for more at: learn.mapr.com

Congratulations! You have completed ESS 102: Lesson 8. Find more courses for developers, data analysts, and administrators at learn.mapr.com!