(Big) Data Engineering In Depth

Moustafa Mahmoud
Data Solution Architect

Chapter: Apache Spark

Chapter Objectives

- Understand the fundamentals of Apache Spark.

Chapter Objectives

- Understand the fundamentals of Apache Spark.
- Build data processing applications using Apache Spark.

Chapter Objectives

- Understand the fundamentals of Apache Spark.
- Build data processing applications using Apache Spark.
- Understand Apache Spark APIs (RDD, DataFrames, and Datasets).

Chapter Objectives

- Understand the fundamentals of Apache Spark.
- Build data processing applications using Apache Spark.
- Understand Apache Spark APIs (RDD, DataFrames, and Datasets).
- Optimize and tune Apache Spark applications.

Chapter Objectives

- Understand the fundamentals of Apache Spark.
- Build data processing applications using Apache Spark.
- Understand Apache Spark APIs (RDD, DataFrames, and Datasets).
- Optimize and tune Apache Spark applications.
- Build streaming applications using Apache Spark.

Chapter Objectives

- Understand the fundamentals of Apache Spark.
- Build data processing applications using Apache Spark.
- Understand Apache Spark APIs (RDD, DataFrames, and Datasets).
- Optimize and tune Apache Spark applications.
- Build streaming applications using Apache Spark.
- Build scalable machine learning applications using Apache Spark MLlib.

Chapter Objectives

- Understand the fundamentals of Apache Spark.
- Build data processing applications using Apache Spark.
- Understand Apache Spark APIs (RDD, DataFrames, and Datasets).
- Optimize and tune Apache Spark applications.
- Build streaming applications using Apache Spark.
- Build scalable machine learning applications using Apache Spark MLlib.
- Deploy Apache Spark in production environments.

Chapter Objectives

- Understand the fundamentals of Apache Spark.
- Build data processing applications using Apache Spark.
- Understand Apache Spark APIs (RDD, DataFrames, and Datasets).
- Optimize and tune Apache Spark applications.
- Build streaming applications using Apache Spark.
- Build scalable machine learning applications using Apache Spark MLlib.
- Deploy Apache Spark in production environments.

Section: Course References

Course References

- Learning Spark, 2nd Edition By Jules S. Damji, Brooke Wenig, Tathagata Das, Denny Lee.

Course References

- Learning Spark, 2nd Edition By Jules S. Damji, Brooke Wenig, Tathagata Das, Denny Lee.
- Spark: The Definitive Guide By Bill Chambers, Matei Zaharia.

Course References

- Learning Spark, 2nd Edition By Jules S. Damji, Brooke Wenig, Tathagata Das, Denny Lee.
- Spark: The Definitive Guide By Bill Chambers, Matei Zaharia.
- Random blog posts.

Course References

- Learning Spark, 2nd Edition By Jules S. Damji, Brooke Wenig, Tathagata Das, Denny Lee.
- Spark: The Definitive Guide By Bill Chambers, Matei Zaharia.
- Random blog posts.

Section: Course Prerequisites

Course Prerequisites

Before beginning this course, participants should have:

- Experience in programming using Python.

Course Prerequisites

Before beginning this course, participants should have:

- Experience in programming using Python.
- Basic programming skills using shell.

Course Prerequisites

Before beginning this course, participants should have:

- Experience in programming using Python.
- Basic programming skills using shell.
- An understanding of MapReduce foundations and Hive. Garage Education YouTube Playlist

Course Prerequisites

Before beginning this course, participants should have:

- Experience in programming using Python.
- Basic programming skills using shell.
- An understanding of MapReduce foundations and Hive. Garage Education YouTube Playlist

Section: Python vs Scala

Python vs Scala

- Python is widely used with numerous tools and libraries available.

Python vs Scala

- Python is widely used with numerous tools and libraries available.
- Python is easier to learn than Scala; however, Scala might be more intuitive for those who prefer functional programming.

Python vs Scala

- Python is widely used with numerous tools and libraries available.
- Python is easier to learn than Scala; however, Scala might be more intuitive for those who prefer functional programming.
- Finding Python developers is generally easier for companies than finding Scala developers.

Python vs Scala

- Python is widely used with numerous tools and libraries available.
- Python is easier to learn than Scala; however, Scala might be more intuitive for those who prefer functional programming.
- Finding Python developers is generally easier for companies than finding Scala developers.
- Initially, Scala offered better performance in Apache Spark, but this advantage has diminished over time, with no significant performance differences now.

Python vs Scala

- Python is widely used with numerous tools and libraries available.
- Python is easier to learn than Scala; however, Scala might be more intuitive for those who prefer functional programming.
- Finding Python developers is generally easier for companies than finding Scala developers.
- Initially, Scala offered better performance in Apache Spark, but this advantage has diminished over time, with no significant performance differences now.
- PySpark and Scala share the same Spark concepts, allowing for interchangeable use of examples from both languages without affecting learning.

Python vs Scala

- Python is widely used with numerous tools and libraries available.
- Python is easier to learn than Scala; however, Scala might be more intuitive for those who prefer functional programming.
- Finding Python developers is generally easier for companies than finding Scala developers.
- Initially, Scala offered better performance in Apache Spark, but this advantage has diminished over time, with no significant performance differences now.
- PySpark and Scala share the same Spark concepts, allowing for interchangeable use of examples from both languages without affecting learning.

Chapter: Introduction to Apache Spark

Section: History of Apache Spark

History of Apache Spark

- Apache Spark was initiated at UC Berkeley in 2009, leading to the publication of Spark: Cluster Computing with Working Sets in 2010 by Matei Zaharia et al.

History of Apache Spark

- Apache Spark was initiated at UC Berkeley in 2009, leading to the publication of Spark: Cluster Computing with Working Sets in 2010 by Matei Zaharia et al.

- Spark was developed to improve processing efficiency over Hadoop MapReduce, which struggled with iterative tasks because it launched separate jobs and reloaded data for each one. This was particularly important for machine learning algorithms that need multiple data passes.

History of Apache Spark

- Apache Spark was initiated at UC Berkeley in 2009, leading to the publication of Spark: Cluster Computing with Working Sets in 2010 by Matei Zaharia et al.

- Spark was developed to improve processing efficiency over Hadoop MapReduce, which struggled with iterative tasks because it launched separate jobs and reloaded data for each one. This was particularly important for machine learning algorithms that need multiple data passes.

- Initially, Spark was designed for batch applications, but it quickly expanded to include streaming, SQL analytics, graph processing, and machine learning.

History of Apache Spark

- Apache Spark was initiated at UC Berkeley in 2009, leading to the publication of Spark: Cluster Computing with Working Sets in 2010 by Matei Zaharia et al.

- Spark was developed to improve processing efficiency over Hadoop MapReduce, which struggled with iterative tasks because it launched separate jobs and reloaded data for each one. This was particularly important for machine learning algorithms that need multiple data passes.

- Initially, Spark was designed for batch applications, but it quickly expanded to include streaming, SQL analytics, graph processing, and machine learning.

History of Apache Spark

- By 2013, the project had more than 100 contributors, and now it has over 2,000 contributors with more than 39,000 commits. It has been donated to the Apache Software Foundation, guaranteeing its future as a vendor-independent project.

History of Apache Spark

- By 2013, the project had more than 100 contributors, and now it has over 2,000 contributors with more than 39,000 commits. It has been donated to the Apache Software Foundation, guaranteeing its future as a vendor-independent project.

- Key milestones in its development are Spark 1.0 in 2014, Spark 2.0 in 2016, and Spark 3.0 in 2020, highlighting its evolution and broad acceptance.

History of Apache Spark

- By 2013, the project had more than 100 contributors, and now it has over 2,000 contributors with more than 39,000 commits. It has been donated to the Apache Software Foundation, guaranteeing its future as a vendor-independent project.

- Key milestones in its development are Spark 1.0 in 2014, Spark 2.0 in 2016, and Spark 3.0 in 2020, highlighting its evolution and broad acceptance.

Section: About Databricks

About Databricks

- Databricks, founded by the early AMPlab team, joined the community to further develop Spark.

About Databricks

- Databricks, founded by the early AMPlab team, joined the community to further develop Spark.
- The organization was founded to deliver a comprehensive analytics platform, streamlining Spark's application in big data processing and analytics.

About Databricks

- Databricks, founded by the early AMPlab team, joined the community to further develop Spark.
- The organization was founded to deliver a comprehensive analytics platform, streamlining Spark's application in big data processing and analytics.
- Databricks has bridged the gap between academic research and enterprise applications through its managed cloud service and significant contributions to the Spark project.

About Databricks

- Databricks, founded by the early AMPlab team, joined the community to further develop Spark.
- The organization was founded to deliver a comprehensive analytics platform, streamlining Spark's application in big data processing and analytics.
- Databricks has bridged the gap between academic research and enterprise applications through its managed cloud service and significant contributions to the Spark project.

Beyond Apache Spark

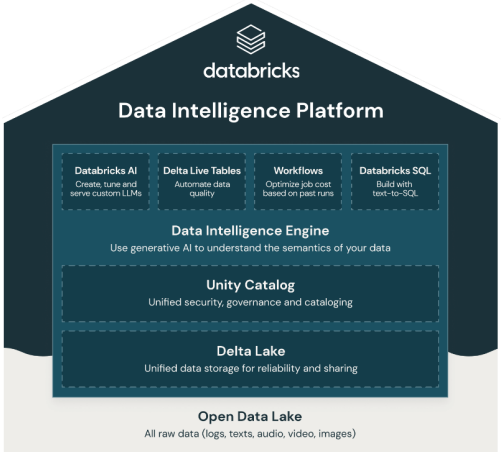


Photo copied from
https://www.datanami.com

Figure F-1: Databricks Analytics Platform

Do You Need Databricks to Work with Spark?

Do You Need Databricks to Work with Spark?

Do You Need Databricks to Work with Spark?

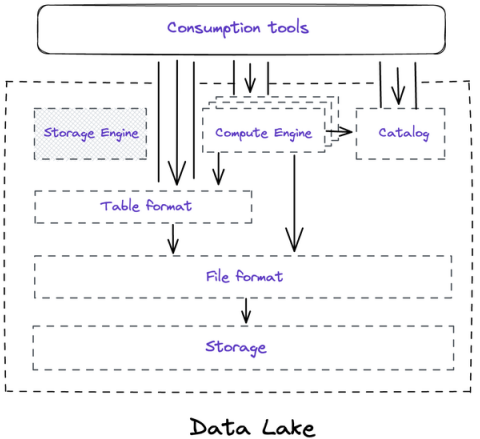
The answer is NO!

Do You Need Databricks to Work with Spark?

- Spark workloads can be run both on the Cloud and on-premise.
  - **Cloud:** Choose any preferred cloud provider, like AWS, GCP, or Azure.
  - **On-Premise:** Deploy on Kubernetes or utilize big data distributions such as Cloudera.
  - **Serverless Platforms:** For efficient resource management, consider options like:
    - **AWS:** EMR Serverless or AWS Glue for auto-scaling and managed services.
    - **GCP:** Dataproc for integrated analytics and machine learning.
    - **Azure:** Azure Synapse for big data and advanced analytics.

Section: Apache Spark in Data Platforms

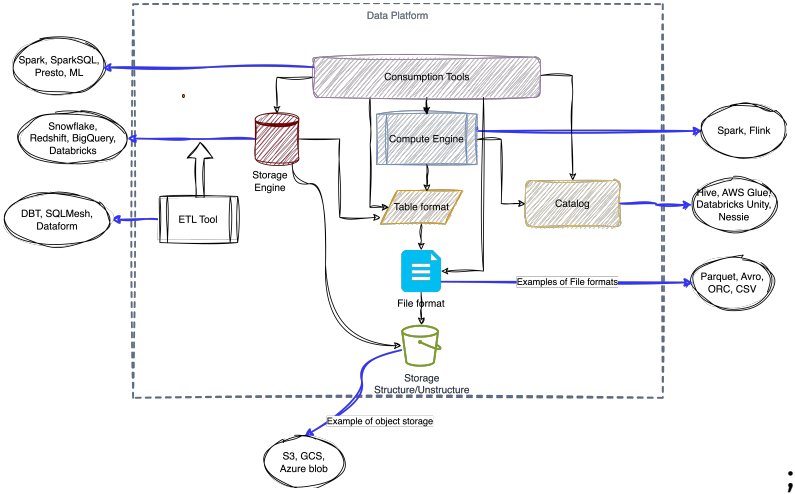
Technical components in a data lake



Apache Iceberg: The Definitive Guide:
Data Lakehouse Functionality,
Performance, and Scalability
on the Data Lake
PUBLISHED BY: O'Reilly Media, Inc.

Figure F-2: Technical components in a data lake

Technical components in a data lake



;

Section: Running Spark

Running Spark for Beginners

- **Databricks Community Edition:** The simplest option for Spark beginners. A free version that's easy to use for learning and small projects.

Running Spark for Beginners

- **Databricks Community Edition:** The simplest option for Spark beginners. A free version that's easy to use for learning and small projects.
- **Install Spark Locally:** For hands-on experience with Spark's core features on your own machine.

Running Spark for Beginners

- **Databricks Community Edition:** The simplest option for Spark beginners. A free version that's easy to use for learning and small projects.
- **Install Spark Locally:** For hands-on experience with Spark's core features on your own machine.
- **Spark on Docker:** For a flexible, containerized environment that can replicate a production setup.

Note: For those new to Spark, starting with the Databricks Community Edition is highly recommended due to its user-friendly interface and comprehensive documentation.

Demo!

Section: From MapReduce to Apache Spark

The basic idea of MapReduce

- Assume we need to launch a high-throughput bulk-production sandwich shop.

[1]This example taken from https://reberhardt.com/cs110/summer-2018/lecture-notes/lecture-14/

The basic idea of MapReduce

- Assume we need to launch a high-throughput bulk-production sandwich shop.
- This sandwich has a lot of raw ingredients, and our target is to produce the sandwich as quickly as possible.

[1]This example taken from https://reberhardt.com/cs110/summer-2018/lecture-notes/lecture-14/

The basic idea of MapReduce

- Assume we need to launch a high-throughput bulk-production sandwich shop.
- This sandwich has a lot of raw ingredients, and our target is to produce the sandwich as quickly as possible.
- To make the production very quickly we need to distribute the tasks between the *workers*.

[1]This example taken from https://reberhardt.com/cs110/summer-2018/lecture-notes/lecture-14/

The basic idea of MapReduce

We break this into three stages

- Map.

[1]This example taken from https://reberhardt.com/cs110/summer-2018/lecture-notes/lecture-14/

The basic idea of MapReduce

We break this into three stages

- Map.
- Shuffle/Group (Mapper Intermediates).

[1]This example taken from https://reberhardt.com/cs110/summer-2018/lecture-notes/lecture-14/

The basic idea of MapReduce

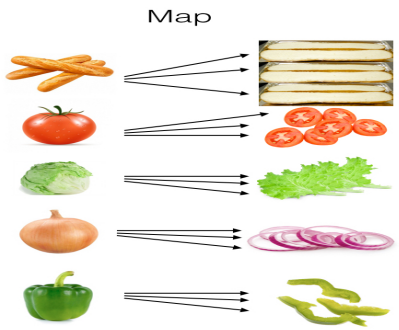
We break this into three stages

- Map.
- Shuffle/Group (Mapper Intermediates).
- Reduce

[1]This example taken from https://reberhardt.com/cs110/summer-2018/lecture-notes/lecture-14/

Map

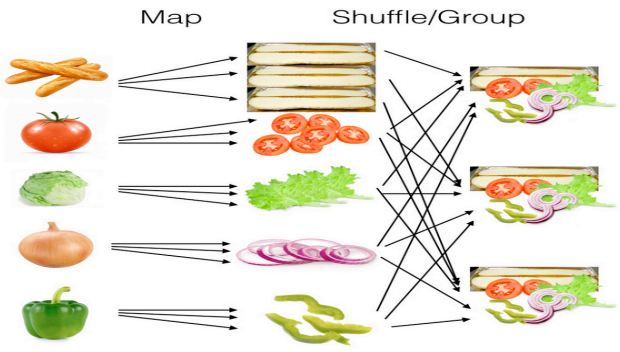
We distribute our raw ingredients amongst the workers.



[1] This example taken from https://reberhardt.com/cs110/summer-2018/lecture-notes/lecture-14/

Shuffle/Group

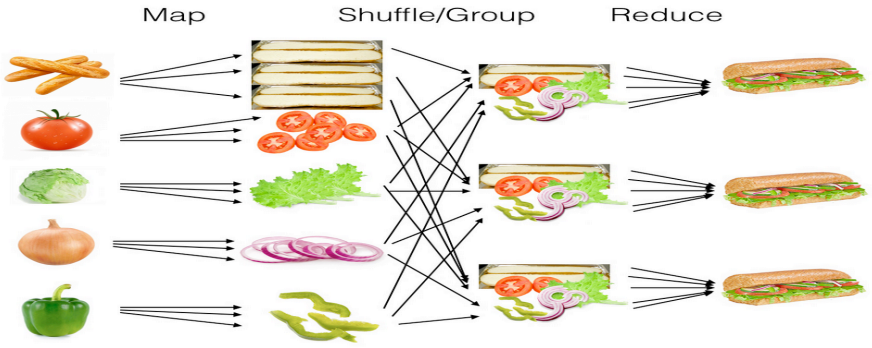
We will organise and group the processed ingredients into piles, so that making a sandwich becomes easy.



[1] This example taken from https://reberhardt.com/cs110/summer-2018/lecture-notes/lecture-14/

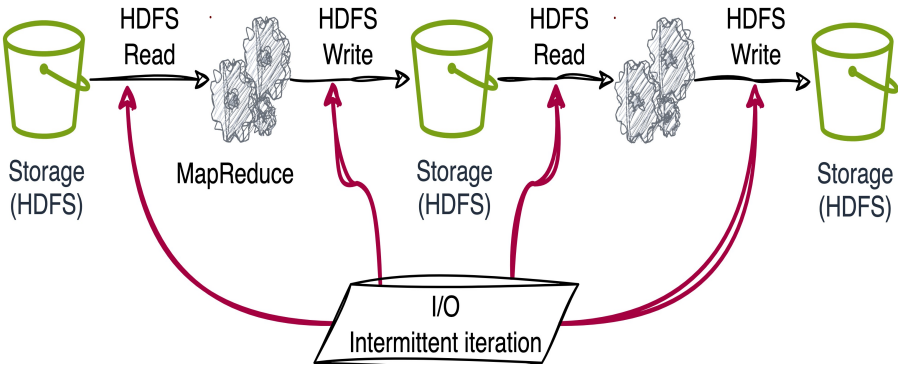
Reduce

we'll combine the ingredients into a sandwich



[1] This example taken from https://reberhardt.com/cs110/summer-2018/lecture-notes/lecture-14/

Map Reduce Bottelneck



Spark Motivation

- In-Memory Processing.
- Resilient Distributed Datasets (RDDs).
- Optimized Execution.
- Caching.
- Advanced Optimization.

Section: What is Apache Spark

What is Apache Spark

- Speed.

What is Apache Spark

- Speed.
- Ease of use.

What is Apache Spark

- Speed.
- Ease of use.
- Modularity.

What is Apache Spark

- Speed.
- Ease of use.
- Modularity.
- Extensibility.

What is Apache Spark

- Speed.
- Ease of use.
- Modularity.
- Extensibility.

Chapter: Apache Spark Distributed Execution

Apache Spark Distributed Execution

- Spark driver.

Apache Spark Distributed Execution

- Spark driver.
- Spark session.

Apache Spark Distributed Execution

- Spark driver.
- Spark session.
- Cluster manager.

Apache Spark Distributed Execution

- Spark driver.
- Spark session.
- Cluster manager.
- Spark executors.

Apache Spark Distributed Execution

- Spark driver.
- Spark session.
- Cluster manager.
- Spark executors.
- Deployment mode.

Apache Spark Distributed Execution

- Spark driver.
- Spark session.
- Cluster manager.
- Spark executors.
- Deployment mode.
- Data partition.

Apache Spark Distributed Execution

- Spark driver.
- Spark session.
- Cluster manager.
- Spark executors.
- Deployment mode.
- Data partition.

Section: Further Readings and Assignment

Thank you for watching!

See you in the next video ☺