WQD7007 Big Data Management

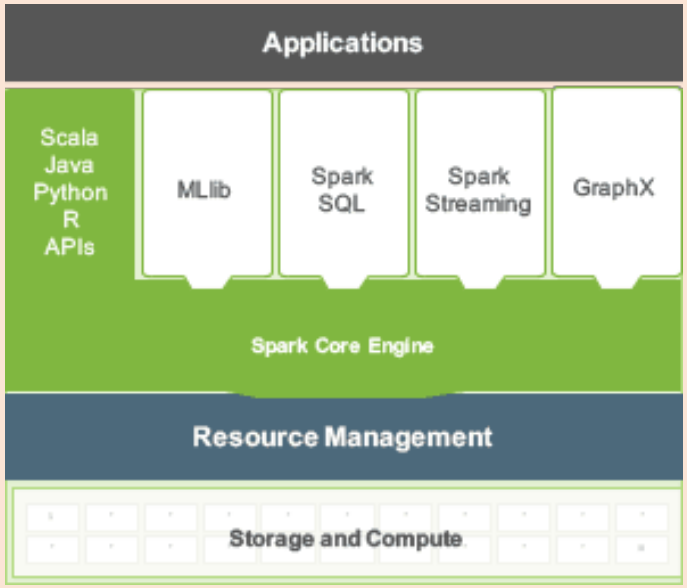
Spark

Spark

- Spark adds **in-Memory Compute** for ETL, Machine Learning and Data Science Workloads to Hadoop
  - fast, in-memory data processing engine with elegant and expressive development APIs to allow data workers to efficiently execute streaming, machine learning or **SQL workloads that require fast iterative access** to datasets.
- With Spark running on Apache Hadoop YARN, developers everywhere can now create applications to exploit Spark's power, derive insights, and enrich their data science workloads within a single, shared dataset in Hadoop.

Spark

- Apache Spark consists of Spark Core and a set of libraries. The core is the distributed execution engine and the Java, Scala, and Python APIs offer a platform for distributed ETL application development.
- Additional libraries, built atop the core, allow diverse workloads for streaming, SQL, and machine learning.



Spark

- Spark is designed for data science and its abstraction makes data science easier.
  1. Data scientists commonly use **machine learning** – a set of techniques and algorithms that can learn from data.
     - These algorithms are often iterative, and Spark's ability to cache the dataset in memory greatly speeds up such iterative data processing, making Spark an ideal processing engine for implementing such algorithms.
     - Spark also includes MLlib, a library that provides a growing set of machine algorithms for common data science techniques: Classification, Regression, Collaborative Filtering, Clustering and Dimensionality Reduction.

Spark

- Spark is designed for data science and its abstraction makes data science easier.
  2. Spark's ML Pipeline API is a high level abstraction to model an entire data science workflow.
     - The ML pipeline package in Spark models a typical machine learning workflow and provides abstractions like Transformer, Estimator, Pipeline & Parameters. This is an abstraction layer that makes data scientists more productive.

Use Cases for Spark

Insurance	Optimize their claims reimbursements process by using Spark's machine learning capabilities to process and analyze all claims.
Healthcare	Build a Patient Care System using Spark Core, Streaming and SQL.
Retail	Use Spark to analyze point-of-sale data and coupon usage.
Internet	Use Spark's ML capability to identify fake profiles and enhance products matches that they show their customers.
Banking	Use a machine learning model to predict the profile of retail banking customers for certain financial products.
Government	Analyze spending across geography, time and category.
Scientific Research	Analyze earthquake events by time, depth, geography to predict future events.
Investment Banking	Analyze intra-day stock prices to predict future price movements.
Geospatial Analysis	Analyze Uber trips by time and geography to predict future demand and pricing.
Twitter Sentiment Analysis	Analyze large volumes of Tweets to determine positive, negative or neutral sentiment for specific organizations and products.
Airlines	Build a model for predicting airline travel delays.
Devices	Predict likelihood of a building exceeding threshold temperatures.

Storm?

- A system for processing streaming data in real time
  - adds reliable real-time data processing capabilities to Enterprise Hadoop.
  - Storm on YARN is powerful for scenarios **requiring real-time analytics, machine learning and continuous monitoring of operations.**
  - Storm integrates with YARN via Apache Slider
  - YARN manages Storm while also considering cluster resources for data governance, security and operations components of a modern data architecture.

Storm

- Storm is effective in distributed real-time computation system for processing large volumes of high-velocity data.
  - Storm is extremely fast, with the ability to **process over a million records per second per node** on a cluster of modest size. Enterprises harness this speed and combine it with other data access applications in Hadoop to prevent undesirable events or to optimize positive outcomes.
  - Some of specific new business opportunities include: real-time customer service management, data monetization, operational dashboards, or cyber security analytics and threat detection.

Use cases for Storm

	"Prevent" Use Cases	"Optimize" Use Cases
Financial Services	•Securities fraud •Operational risks & compliance violations	•Order routing •Pricing
Telecom	•Security breaches •Network outages	•Bandwidth allocation •Customer service
Retail	•Shrinkage •Stock outs	•Offers •Pricing
Manufacturing	•Preventative maintenance •Quality assurance	•Supply chain optimization •Reduced plant downtime
Transportation	•Driver monitoring •Predictive maintenance	•Routes •Pricing
Web	•Application failures •Operational issues	•Personalized content

Storm VS Spark

	Storm	Spark
Programming Language Options	Creation of Storm applications is possible in Java, Clojure, and Scala	Creation of Spark applications is possible in Java, Scala, Python & R.
Fault tolerance	It is designed with fault-tolerance at its core. As if the process fails, supervisor process will restart it automatically. Because ZooKeeper handles the state management.	It is also fault tolerant in nature. Spark handles restarting workers by resource managers, such as Yarn, Mesos or its Standalone Manager.
Latency	It provides better latency with fewer restrictions.	Latency is less good than a storm.
Ease of Operability	It is not easy to deploy/install storm through many tools and deploys the cluster. It depends on Zookeeper cluster.	Spark is fundamental execution framework for streaming. Hence, it should be easy to feed up spark cluster of YARN.
Low development cost	We cannot use same code base for stream processing and batch processing	We can use same code base for stream processing as well as batch processing

Source: https://techvidvan.com/tutorials/apache-storm-vs-spark-streaming/

Using spark to do wordcount

- Example from here. Start the spark shell by typing `spark-shell` in terminal.

- Step 1: load input text file from local directory. Text file example can be downloaded from here.

```
val inputfile = sc.textFile (“10538-0.txt")
```

*You can display `inputfile` content by:

- inputfile.collect().foreach(println)
- Inputfile.take(5).foreach(println)

Using spark to do wordcount

- Example from here. Start the spark shell by typing `spark-shell` in terminal.

- Step 2: count word by word, using space as delimiter:

```
val counts = inputfile. flatMap (line =>
line. Split (" ")).map (word => (word,
1)).reduceByKey (_+_)
```

Using spark to do wordcount

- Example from here. Start the spark shell by typing `spark-shell` in terminal.

- Step 3: save the outcome in a file.

```
counts.saveAsTextFile ("output")
```

- How to do machine learning? Good example here.

What is not covered but is important?

- Spark2, pyspark, spark in zeppellin
- Kafka
- Connect mongodb to ubuntu
- Access control in Linux