**Data Engineering Notes**

*Created: 2024-10-30*

*Updated: 2024-11-01*

**References:**

* <https://www.coursera.org/learn/spark-hadoop-snowflake-data-engineering>
* <https://learn.microsoft.com/en-us/training/modules/use-apache-spark-azure-databricks/>
* ChatGPT 4o mini
* Google search

***Apache Hadoop***: open source ecosystem of software enabling parallel processing of big data.

* Hadoop Distributed File System (HDFS): storage system.
* MapReduce: framework within Hadoop that:
  + maps (distributes) tasks across a cluster of computers that store intermediate results (key-value pair) on disk (slower than in-memory).
  + Intermediate results are grouped by keys and sent to the cluster of computers (parallel processing) responsible for reducing (performing user reduction function) and results are combined.

***Apache Spark***: built on top of Hadoop but stores intermediate results in-memory instead of on disk. Leverages parallelism for task completion via horizontal scaling (more nodes).

* Each worker/executor node runs a JVM (multi-threaded). Each has multiple slots based on #cores #cpus of node.
* Parallelized jobs are broken down into stages to be performed in order.

***Azure Databricks***: parallelized data processing on Apache Spark clusters.

* A notebook instance (*SparkSession* object) controls the driver node which distributes work across worker nodes.

***Resilient Distributed Datasets (RDD)***: immutable fundamental data structure in Spark that facilitates distributed computing.

* Resilient: fault tolerant.
* Distributed: RDDs distributed across nodes in Spark clusters.
* Operations:
  + Transformation (lazy, triggered with action): creates new RDDs after applying transformation e.g., filter.
    - Shuffle is a transformation operation that writes data to disk then re-partitions data. Shuffle defines stage boundaries.
  + Action: return result from computation, e.g., count.
* In pyspark shell (sc: spark context, spark: spark session):
  + rdd = sc.parallelize(list(range(15)) # create RDD
  + rdd2 = rdd.map(lambda x: x\*2) #transformation
  + rdd3 = rdd.filter(lambda x: x%3==0) #lazy transformation
  + rdd3.count() #action triggers computation

***Spark SQL DataFrames***: DataSet with named columns.

* Distributed collection of data.
* Abstraction of RDDs.
* Optimized for structured data manipulation.
* Works with (semi)-structured data: CSV, Apache Avro, ORC, Parquet, JSON, relational DBs.
* PySpark integrates with Spark SQL can be used for ETL (Extract, Transform, Load) pipelines.
* Spark SQL is a module within Spark providing SQL capabilities for structured data.
* PySpark is a library allowing Spark to be used in Python.

**Big Data Storage Architecture**

* ***Shared everything***: database installed on the server that runs computations. Efficient for small data.
* ***Shared disk***: computation done on separate nodes, but all share one storage. Nodes may need to wait for each other to write to disk, reducing efficiency.
* ***Shared nothing***: each computation node has its own storage.

***Snowflake***: big data processing platform built on top of Aws, Azure, GC.

* Architecture: persistent data stored in shared storage and each processing node has its own local storage.
* Layers:
  + Database storage: stores compressed and encrypted data that can only be accessed by Snowflake SQL.
  + Compute layer: virtual warehouses that can be scaled both horizontally and vertically containing node clusters from a cloud provider, e.g., standard, Snowpark (for large memory tasks).
    - ***Virtual warehouses***: independent MPP (massive parallel processor) clusters.
  + Cloud services (top most layer): interface to other layers, handles authentication, security, query optimization. Run Snowflake SQL “cursor.execute()”
* Python connector (snowflake.connector, snowflake.connector.pandas): run Snowflake SQL and pull push data to Snowflake.

***Snowsight***: UI tool built on top of Snowflake (data warehousing platform) to enhance data visualization, exploration, analysis, etc.