CHEATSHEET ON BIG DATA TECHNOLOGIES

Introduction to Big Data Technologies:

- **Definition:**

  Big data is defined as extraordinarily big and complicated datasets that cannot be properly handled, processed, or analyzed using conventional data processing tools and methodologies. The term "big data" refers to not only the size of a dataset, but also its volume, velocity, diversity, validity, and value. These datasets often have the following characteristics:

  1. Volume: Big data refers to massive amounts of data that transcend the storage and processing capabilities of typical databases and technologies. Big data collections might be terabytes, petabytes, or even larger.

  2. Velocity: Data is generated at a quick rate in a variety of formats, including streaming data, sensor data, social media updates, and others. Processing this data in real-time or near-real-time is critical for gaining important insights and making informed decisions.

  3. Variety: Big data refers to a wide range of data types, forms, and sources, including structured data (e.g., databases, spreadsheets), unstructured data (e.g., text, photos, videos), and semi-structured data. This variation complicates data administration and analysis.

  4. Veracity: Big data may have concerns with data quality, reliability, and correctness. Veracity refers to the data's dependability and the extent to which it may be used to make educated decisions.

  5. Value: The basic goal of big data analytics is to extract meaningful insights and value from data. This could include detecting patterns, trends, correlations, and anomalies to help drive corporate decisions, optimize operations, improve customer experiences, or develop new goods and services.

- ## Importance of big data technologies in handling large volumes of data Hadoop

The development of open source frameworks, such as Hadoop (and more recently, Spark) was essential for the growth of big data because they make big data easier to work with and cheaper to store. In the years since then, the volume of big data has skyrocketed.

Hadoop, a key component in big data, manages massive data volumes by distributing storage and processing across commodity technology. Its scalability provides cost effectiveness, while fault tolerance and parallel processing improve dependability and performance. Hadoop supports a wide range of data formats and interfaces with a large ecosystem of real-time analytics tools such as Apache Spark and Kafka. Its versatility supports a wide range of analytics jobs, from querying to predictive modeling, allowing organizations to gain important insights from structured, semi-structured, and unstructured data. Hadoop's significance stems from its capacity to efficiently manage enormous amounts of data, enabling informed decision-making and innovation in today's data-driven environment.

- ## Overview of Hadoop ecosystems components:

Hadoop is a framework that uses distributed storage and parallel processing to store and manage big data. It is the software most used by data analysts to handle big data, and its market size continues to grow. There are three components of Hadoop:

- Hadoop HDFS - Hadoop Distributed File System (HDFS) is the storage unit.
- Hadoop MapReduce - Hadoop MapReduce is the processing unit.
- Hadoop YARN - Yet Another Resource Negotiator (YARN) is a resource management unit.

1. **HDFS:** The conventional strategy involved storing all data in a single central database. With the rise of big data, a single database was unable to handle the workload. The solution was to store the large volume of data in a distributed manner. Data was broken up and assigned to numerous distinct databases. HDFS is a particularly built file system for storing large datasets with commodity hardware, storing information in multiple formats on different devices.

There are two components in HDFS:
NameNode - NameNode is the master daemon. There is only one active NameNode. It manages the DataNodes and stores all the metadata.

DataNode - DataNode is the slave daemon. There can be multiple DataNodes. It stores the actual data.

So, we spoke of HDFS storing data in a distributed fashion, but did you know that the storage system has certain specifications? HDFS splits the data into multiple blocks, defaulting to a maximum of 128 MB. The default block size can be changed depending on the processing speed and the data distribution.

2. **YARN (Yet Another Resource Negotiator):**

   YARN is an acronym for Yet Another Resource Negotiator. It handles the cluster of nodes and acts as Hadoop's resource management unit. YARN allocates RAM, memory, and other resources to different applications.

   YARN has two components :

   ResourceManager (Master) - This is the master daemon. It manages the assignment of resources such as CPU, memory, and network bandwidth.

   NodeManager (Slave) - This is the slave daemon, and it reports the resource usage to the Resource Manager.

3. **MapReduce:**

   Hadoop data processing is built on MapReduce, which processes large volumes of data in a parallelly distributed manner.

   As we can see, we have large amounts of data that must be processed in order to provide an output. Initially, input data is partitioned into input splits. The first phase is the Map phase, in which data from each split is used to generate output values. The shuffle and sort phase takes the result of the mapping phase and groups it into blocks of comparable data. Finally, the output values of the shuffling phase are aggregated. It then returns one output value.

- ## Use cases:

Batch Processing:

Scenario: A retail company needs to analyze sales data collected over the past month to identify trends, optimize inventory, and generate reports for business stakeholders.
Solution: Using Hadoop's MapReduce framework, the organization can run batch processing operations on massive amounts of sales data stored in HDFS. MapReduce divides data into smaller bits and distributes processing across numerous nodes in the cluster, allowing parallel execution. This approach allows for more effective analysis of previous sales data, allowing the company to gain insights, estimate demand, and make data-driven decisions to improve business operations.

Distributed Storage:

Scenario: A financial institution requires a scalable and fault-tolerant storage solution to manage vast amounts of transaction data generated daily from multiple sources.
Solution: The organization can store transactional data across dispersed clusters of commodity hardware by implementing Hadoop's HDFS. HDFS uses multiple node data replication to provide high availability and fault tolerance. Nodes can be added to the cluster to enable seamless scalability as data quantities increase. The organization can effectively manage and analyze big datasets while maintaining data integrity and dependability thanks to this distributed storage architecture.

Distributed Computation:

Scenario: A healthcare organization needs to analyze medical imaging data to identify anomalies, diagnose diseases, and support medical research projects.
Solution: The company can use distributed compute jobs to process and analyze medical imaging data stored in HDFS by utilizing Hadoop's YARN framework. YARN allows for the effective use of computing resources throughout the cluster by dynamically allocating cluster resources to compute workloads according to their needs. The company can expedite the analysis of medical imaging data, promote cooperative research endeavors, and enhance patient care results by allocating computation jobs among several nodes.

Spark:

- ## Introduction to Apache Spark Framework:

  The Apache Spark project promises "lightning fast cluster computing." It is currently the most active Apache project, with a flourishing open-source community.

  A more versatile and speedier data processing platform is offered by Spark. Programmes can run up to 100 times faster in memory or 10 times faster on disc with Spark compared with Hadoop. By finishing the 100 TB Daytona GraySort competition three times faster on a tenth of the computers needed, Spark overtook Hadoop last year. It also became the quickest open source engine for sorting a petabyte.

  Spark facilitates faster code writing by providing access to more than 80 high-level operators.

  Additional key features of Spark include:

  - ❖ Currently provides APIs in Scala, Java, and Python, with support for other languages (such as R) on the way
  - ❖ Integrates well with the Hadoop ecosystem and data sources (HDFS, Amazon S3, Hive, HBase, Cassandra, etc.)
  - ❖ Can run on clusters managed by Hadoop YARN or Apache Mesos, and can also run standalone

- ## Features:

  In-Memory Processing: Utilizes in-memory computing to speed up data processing.

  Example:

```
# Python code snippet demonstrating in-memory processing in Spark
data = spark.read.csv("data.csv")  # Read data into Spark DataFrame
data.cache()  # Cache data in memory for faster access
```

  Support for Multiple Languages: Provides APIs in Scala, Python, and Java for programming Spark applications.

  Example:

```
// Scala code snippet demonstrating Spark application in Scala
val data = spark.read.csv("data.csv")  // Read data into DataFrame
```

```
data.show()  // Display data using Scala API
```

- **Use cases: Real-time data processing, machine learning, graph processing**

  Real-Time Data Processing: Process streaming data in real-time for analytics and decision-making.

  Example:
  ```
  # Python code snippet demonstrating real-time data processing in Spark
  streaming_data = spark.readStream.format("kafka").option("kafka.bootstrap.servers", "localhost:9092").load()
  ```

  Machine Learning: Build and deploy scalable machine learning models on large datasets.

  Example:
  ```
  # Python code snippet demonstrating machine learning with Spark MLlib
  from pyspark.ml.classification import LogisticRegression
  lr = LogisticRegression(featuresCol='features', labelCol='label')
  model = lr.fit(training_data)
  ```

  Graph Processing: Perform graph analytics and computations on large-scale graphs.

  Example:
  ```
  // Scala code snippet demonstrating graph processing with Spark GraphX
  import org.apache.spark.graphx._
  val graph: Graph[Int, Int] = GraphLoader.edgeListFile(spark, "graph.txt")
  ```

Hive:

- **Overview of Apache Hive:** Apache Hive is open-source data warehouse software designed to read, write, and manage large datasets extracted from the Apache Hadoop Distributed File System (HDFS) , one aspect of a larger Hadoop Ecosystem. With extensive Apache Hive documentation and continuous updates, Apache Hive continues to innovate data processing in an ease-of-access way.

  This was created mainly on two aspects,

1. **SQL:** Use SQL with minimal disruption or retraining compared to others.
2. It provided a centralized metadata store (Hadoop based) of all the datasets in the organization.

- **Features:**

  SQL-like Query Language (HiveQL): Allows users to write queries in a familiar SQL syntax, making it accessible to users with SQL skills.

  Example:

```
-- SQL-like query in HiveQL
SELECT * FROM table_name WHERE column_name = 'value';
```

  Data Warehousing Capabilities: Supports data warehousing functionalities such as partitioning, bucketing, and indexing to optimize query performance and enhance data organization.

  Example:

```
-- Example of partitioning table in Hive
CREATE TABLE sales_data (
    product_id INT,
    sales_amount DECIMAL
) PARTITIONED BY (date STRING);
```

- **Use Cases:**

  Data Analysis: Hive is commonly used for data analysis tasks, allowing users to run complex queries on large datasets to derive insights and make data-driven decisions.

  Example:

```
-- Example of data analysis query in Hive
SELECT product_category, SUM(sales_amount) AS total_sales
FROM sales_data
GROUP BY product_category;
```

  Ad-Hoc Querying: Users can perform ad-hoc queries on diverse datasets stored in Hadoop, without the need for pre-defined schemas or data models.

  Example:

```
-- Example of ad-hoc query in Hive
SELECT * FROM log_data WHERE event_type = 'error' LIMIT 10;
```

ETL (Extract, Transform, Load): Hive facilitates ETL processes by enabling users to extract data from various sources, transform it using HiveQL expressions, and load it into target tables.

Example:

```
-- Example of ETL query in Hive
INSERT INTO target_table
SELECT column1, column2, column3
FROM source_table
WHERE condition = 'value';
```

Comparison of Big Data Technologies

- **Performance: Speed, scalability:**

Speed:

➔ **Apache Spark:**

1. Offers high-speed processing through in-memory computing.
2. Utilizes RDDs (Resilient Distributed Datasets) and DAG (Directed Acyclic Graph) execution for efficient computation.
3. Suitable for iterative algorithms and interactive queries.

➔ **Apache Hadoop:**

1. Generally slower due to disk-based processing.
2. Splits data into smaller chunks (map) and processes them in parallel before combining results (reduce).
3. Can handle large-scale batch processing effectively but may face performance bottlenecks with extremely large datasets due to disk I/O.

Scalability:

➔ **Apache Spark:**
Highly scalable, thanks to its in-memory processing and efficient data caching mechanisms.
Scales horizontally by adding more nodes to the cluster, offering linear scalability.

➔ **Apache Hadoop:**
Scalable but may face limitations with extremely large datasets due to disk I/O.
Scales by adding more nodes to the cluster but may not exhibit linear scalability in all cases.

- **Ease of use: Programming model, learning curve:**
    a. **Programming Model:**
    Apache Spark: Provides a more user-friendly programming model with higher-level APIs (e.g., DataFrames, Spark SQL), reducing the complexity of writing distributed applications.
    Apache Hadoop (MapReduce): Requires a lower-level programming model, making it more challenging for developers to write and debug MapReduce programs.

    b. **Learning Curve:**
    Apache Spark: Has a steeper learning curve than traditional Hadoop MapReduce due to its rich ecosystem and diverse APIs, but offers higher productivity once mastered.
    Apache Hadoop (MapReduce): Relatively simpler to learn for developers familiar with Java and MapReduce concepts, but can be cumbersome for complex data processing tasks.

- **Use case suitability: Batch processing vs. real-time processing**

Batch Processing:

1. Slow processing data after chunks accumulation.
2. Use cases are Massive with low variations.
3. Overhead is low bulk, processing minimizes overheads.
4. The complexity was moderate.
5. Low flexibility, changes after batch finishes.

Real time processing:

1. Immediate processing of individual data points.
2. It uses irregular intervals.
3. Overhead is moderate due to constant overhead initiation.
4. Complexity is high.
5. High flexibility.

Best practices and Tips:

- **Optimizing Performance:**

  Apache Spark:

  1. Use in-memory caching.
  2. Optimize parallelism.
  3. Employ efficient transformations.
  4. Monitor and tune resource allocation.

  Apache Hadoop (MapReduce):

  1. Optimize input/output formats.
  2. Compress data.
  3. Tune cluster settings.
  4. Utilize combiners and reducers.

- **Considerations for Choosing Technology Stack:**

  ### Batch Processing:

  1. Apache Hadoop (MapReduce) for traditional batch processing.
  2. Apache Spark for faster processing and iterative algorithms.

  ### Real-Time Processing:

  1. Apache Spark Streaming for micro-batch processing.
  2. Apache Flink for event-driven processing.
  3. Apache Kafka for distributed streaming.

  ### Data Warehousing and Analytics:

  1. Apache Hive for SQL-like querying.
  2. Apache Impala for interactive SQL queries.
  3. Cloud-based solutions like Amazon Redshift or Google BigQuery for scalability and performance.

Resources for further learning:

Hadoop:

Online Tutorials and Documentation:

Official Apache Hadoop Documentation: Hadoop Apache Documentation
Hadoop Tutorial by TutorialsPoint: Hadoop Tutorial

Hortonworks Hadoop Tutorials: Hortonworks Tutorials

Books:
"Hadoop: The Definitive Guide" by Tom White
"Hadoop in Practice" by Alex Holmes
"Hadoop Operations" by Eric Sammer

Community Forums and Discussion Groups:
Apache Hadoop Mailing Lists: Hadoop Mailing Lists
Stack Overflow: Hadoop Tag
Hadoop User Groups (HUGs) in various cities and regions.

Spark:

Online Tutorials and Documentation:
Official Apache Spark Documentation: Spark Apache Documentation
Spark Tutorial by TutorialsPoint: Spark Tutorial
Databricks Spark Guides: Databricks Spark Guides

Books:
"Learning Spark: Lightning-Fast Data Analysis" by Holden Karau, Andy Konwinski, Patrick Wendell, and Matei Zaharia
"Spark: The Definitive Guide" by Bill Chambers and Matei Zaharia

Community Forums and Discussion Groups:
Apache Spark User Mailing List: Spark User Mailing List
Stack Overflow: Spark Tag

Hive:

Online Tutorials and Documentation:
Official Apache Hive Documentation: Hive Apache Documentation
Hive Tutorial by TutorialsPoint: Hive Tutorial
Hortonworks Hive Tutorials: Hortonworks Tutorials

Books:
"Programming Hive" by Edward Capriolo, Dean Wampler, and Jason Rutherglen

Community Forums and Discussion Groups:
Apache Hive User Mailing List: Hive User Mailing List

Conclusion:

In this cheat sheet, we explored key big data technologies including Apache Hadoop, Apache Spark, and Apache Hive, highlighting their features, use cases, and comparison factors. Apache Hadoop, with its MapReduce framework, excels in batch processing tasks, while Apache Spark offers faster processing speeds and scalability through in-memory computing, making it suitable for real-time processing and iterative algorithms. Apache Hive provides SQL-like querying capabilities for data warehousing and analytics use cases.

We discussed optimization strategies for each technology, emphasizing the importance of understanding their performance characteristics and choosing the right technology stack based on specific use case requirements. Mastering these big data technologies is crucial for organizations dealing with large-scale data processing tasks, enabling them to derive valuable insights, make informed decisions, and stay competitive in today's data-driven world. By leveraging the capabilities of Hadoop, Spark, and Hive, businesses can efficiently manage, analyze, and derive actionable insights from their vast datasets, driving innovation and success in the digital era.

References:

1. simplilearn.com/tutorials/hadoop-tutorial
2. toptal.com/spark/introduction-to-apache-spark
3. databricks.com/glossary/apache-hive
4. geeksforgeeks.org/difference-between-batch-processing-and-real-time-processing-system/
5. harperdb.io/post/real-time-vs-batch-processing-vs-stream-processing