The respective architectures of Hadoop and Spark, how these big data frameworks compare in multiple contexts and scenarios that fit best with each solution.

Hadoop and Spark, both developed by the Apache Software Foundation, are widely used open-source frameworks for big data architectures. Each framework contains an extensive ecosystem of open-source technologies that prepare, process, manage and analyze big data sets.

What is Apache Hadoop?

Apache Hadoop is an open-source software utility that allows users to manage big data sets (from gigabytes to petabytes) by enabling a network of computers (or "nodes") to solve vast and intricate data problems. It is a highly scalable, cost-effective solution that stores and processes structured, semi-structured and unstructured data (e.g., Internet clickstream records, web server logs, IoT sensor data, etc.).

Benefits of the Hadoop framework include the following:

- Data protection amid a hardware failure
- Vast scalability from a single server to thousands of machines
- Real-time analytics for historical analyses and decision-making processes

What is Apache Spark?

Apache Spark — which is also open source — is a data processing engine for big data sets. Like Hadoop, Spark splits up large tasks across different nodes. However, it tends to perform faster than Hadoop and it uses random access memory (RAM) to cache and process data instead of a file system. This enables Spark to handle use cases that Hadoop cannot.

Benefits of the Spark framework include the following:

- A unified engine that supports SQL queries, streaming data, machine learning (ML) and graph processing

Can be 100x faster than Hadoop for smaller workloads via in-memory processing, disk data storage, etc.

APIs designed for ease of use when manipulating semi-structured data and transforming data

The Hadoop ecosystem

Hadoop supports advanced analytics for stored data (e.g., predictive analysis, data mining, machine learning (ML), etc.). It enables big data analytics processing tasks to be split into smaller tasks. The small tasks are performed in parallel by using an algorithm (e.g., MapReduce), and are then distributed across a Hadoop cluster (i.e., nodes that perform parallel computations on big data sets).

The Hadoop ecosystem consists of four primary modules:

Hadoop Distributed File System (HDFS): Primary data storage system that manages large data sets running on commodity hardware. It also provides high-throughput data access and high fault tolerance.

Yet Another Resource Negotiator (YARN): Cluster resource manager that schedules tasks and allocates resources (e.g., CPU and memory) to applications.

Hadoop MapReduce: Splits big data processing tasks into smaller ones, distributes the small tasks across different nodes, then runs each task.

Hadoop Common (Hadoop Core): Set of common libraries and utilities that the other three modules depend on.

The Spark ecosystem

Apache Spark, the largest open-source project in data processing, is the only processing framework that combines data and artificial intelligence (AI). This enables users to perform large-scale data transformations and analyses, and then run state-of-the-art machine learning (ML) and AI algorithms.

The Spark ecosystem consists of five primary modules:

Spark Core: Underlying execution engine that schedules and dispatches tasks and coordinates input and output (I/O) operations.

Spark SQL: Gathers information about structured data to enable users to optimize structured data processing.

Spark Streaming and Structured Streaming: Both add stream processing capabilities. Spark Streaming takes data from different streaming sources and divides it into micro-batches for a continuous stream. Structured Streaming, built on Spark SQL, reduces latency and simplifies programming.

Machine Learning Library (MLlib): A set of machine learning algorithms for scalability plus tools for feature selection and building ML pipelines. The primary API for MLlib is DataFrames, which provides uniformity across different programming languages like Java, Scala and Python.

GraphX: User-friendly computation engine that enables interactive building, modification and analysis of scalable, graph-structured data.

Comparing Hadoop and Spark

Spark is a Hadoop enhancement to MapReduce. The primary difference between Spark and MapReduce is that Spark processes and retains data in memory for subsequent steps, whereas MapReduce processes data on disk. As a result, for smaller workloads, Spark's data processing speeds are up to 100x faster than MapReduce.

Furthermore, as opposed to the two-stage execution process in MapReduce, Spark creates a Directed Acyclic Graph (DAG) to schedule tasks and the orchestration of nodes across the Hadoop cluster. This task-tracking process enables fault tolerance, which reapplies recorded operations to data from a previous state.

Let's take a closer look at the key differences between Hadoop and Spark in six critical contexts:

Performance: Spark is faster because it uses random access memory (RAM) instead of reading and writing intermediate data to disks. Hadoop stores data on multiple sources and processes it in batches via MapReduce.

Cost: Hadoop runs at a lower cost since it relies on any disk storage type for data processing. Spark runs at a higher cost because it relies on in-memory computations for real-time data processing, which requires it to use high quantities of RAM to spin up nodes.

Processing: Though both platforms process data in a distributed environment, Hadoop is ideal for batch processing and linear data processing. Spark is ideal for real-time processing and processing live unstructured data streams.

Scalability: When data volume rapidly grows, Hadoop quickly scales to accommodate the demand via Hadoop Distributed File System (HDFS). In turn, Spark relies on the fault tolerant HDFS for large volumes of data.

Security: Spark enhances security with authentication via shared secret or event logging, whereas Hadoop uses multiple authentication and access control methods. Though, overall, Hadoop is more secure, Spark can integrate with Hadoop to reach a higher security level.

Machine learning (ML): Spark is the superior platform in this category because it includes MLlib, which performs iterative in-memory ML computations. It also includes tools that perform regression, classification, persistence, pipeline construction, evaluation, etc.

Misconceptions about Hadoop and Spark

Common misconceptions about Hadoop

Hadoop is cheap: Though it's open source and easy to set up, keeping the server running can be costly. When using features like in-memory computing and network storage, big data management can cost up to $5,000 USD.

Hadoop is a database: Though Hadoop is used to store, manage and analyze distributed data, there are no queries involved when pulling data. This makes Hadoop a data warehouse rather than a database.

Hadoop does not help SMBs: "Big data" is not exclusive to "big companies". Hadoop has simple features like Excel reporting that enable smaller companies to harness its power. Having one or two Hadoop clusters can greatly enhance a small company's performance.

Hadoop is hard to set up: Though Hadoop management is difficult at the higher levels, there are many graphical user interfaces (GUIs) that simplify programming for MapReduce.

Common misconceptions about Spark

Spark is an in-memory technology: Though Spark effectively utilizes the least recently used (LRU) algorithm, it is not, itself, a memory-based technology.

Spark always performs 100x faster than Hadoop: Though Spark can perform up to 100x faster than Hadoop for small workloads, according to Apache, it typically only performs up to 3x faster for large ones.

Spark introduces new technologies in data processing: Though Spark effectively utilizes the LRU algorithm and pipelines data processing, these capabilities previously existed in massively parallel processing (MPP) databases. However, what sets Spark apart from MPP is its open-source orientation.

Hadoop and Spark use cases

Based on the comparative analyses and factual information provided above, the following cases best illustrate the overall usability of Hadoop versus Spark.

Hadoop use cases

Hadoop is most effective for scenarios that involve the following:

- Processing big data sets in environments where data size exceeds available memory
- Batch processing with tasks that exploit disk read and write operations
- Building data analysis infrastructure with a limited budget
- Completing jobs that are not time-sensitive
- Historical and archive data analysis

Spark use cases

Spark is most effective for scenarios that involve the following:

- Dealing with chains of parallel operations by using iterative algorithms
- Achieving quick results with in-memory computations
- Analyzing stream data analysis in real time
- Graph-parallel processing to model data
- All ML applications