Apache Spark

cerenode

cerenode.io

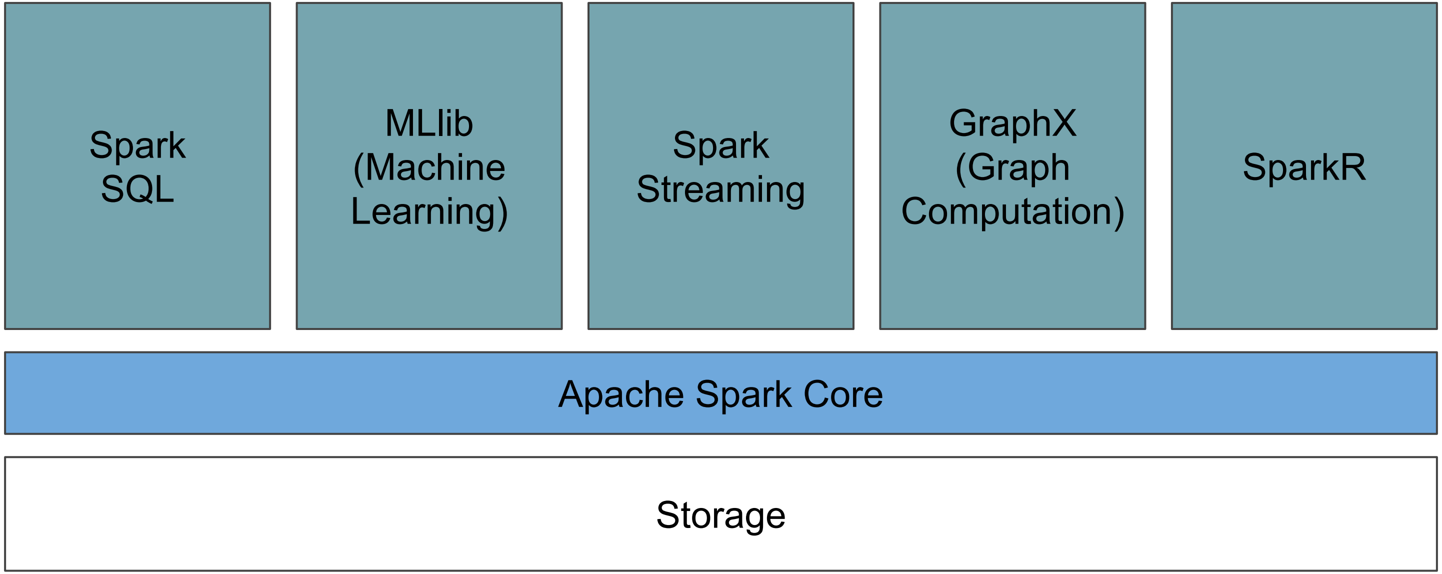
Apache Spark

- Unified analytics engine for large-scale data processing
- Achieves high performance for both batch and streaming data
- Write applications quickly in Java, Scala, Python, R, and SQL
- Runs on Hadoop, Apache Mesos, Kubernetes, standalone, or in the cloud
- Single library can perform SQL, graph analytics and streaming

Apache Spark vs Hadoop MapReduce

Spark	**MapReduce**
In - memory	Persists on disk after map and reduce operations
Faster	Slower
Lower Latency	Higher Latency
Based on Scala	Based on Java
Real time analysis possible	Real time analysis not possible

Components



Spark SQL

MLlib (Machine Learning)

Spark Streaming

GraphX (Graph Computation)

SparkR

Apache Spark Core

Storage

Apache Spark Core

- Central point of Spark
- Provides an execution platform for all the Spark applications
- Provides In-Memory computing and referencing datasets in external storage systems
- It is in charge of essential I/O functionalities
- Significant in programming and observing the role of the Spark Cluster
- Embedded with special collection called RDD

Spark SQL

- Enables users to run SQL/HQL queries
- Introduces a new data abstraction called SchemaRDD, which provides support for structured and semi-structured data
- Can perform extra optimization using the schema
- It offers to run unmodified queries up to 100 times faster on existing deployments

Spark Streaming

- Across live streaming, Spark Streaming enables a powerful interactive and data analytics application
- It ingests data in micro-batches and performs RDD transformations on those micro-batches of data
- Micro-batching is a technique that allows a process or task to treat a stream as a sequence of small batches of data

Phases of Streaming

- Gathering

  Provides two categories of built-in streaming sources

  - Basic Sources - file systems, socket connections
  - Advanced Sources - Kafka, Flume, Kinesis

- Processing
  - Data is processed using complex algorithms expressed with a high-level function

- Data Storage
  - Processed data is pushed out to file systems, databases, and live dashboards

MLlib

- Scalable Machine learning library that has both high-quality algorithm and high speed
- It is capable of in-memory data processing, that improves the performance of iterative algorithm drastically

GraphX

- Graph computation engine that enables to process graph data at scale
- Contains numerous operators in order to manipulate the graphs along with graph algorithms
- Clustering, classification, traversal, searching, and pathfinding is also possible in graphs
- Extends Spark RDD by a new Graph abstraction: a directed multigraph with properties attached to each vertex and edge

SparkR

- Allows data scientists to analyze large datasets
- Run jobs interactively on them from the R shell
- Main idea behind SparkR was to explore different techniques to integrate the usability of R with the scalability of Spark
- R package that gives light-weight frontend to use Apache Spark from R

RDD

- RDD stands for “Resilient Distributed Dataset”
- Fundamental data structure of Apache Spark
- Immutable collection of objects which computes on the different node of the cluster
- Each and every dataset in RDD is logically partitioned across many servers so that they can be computed on different nodes of the cluster
- Can also be cached and manually partitioned

Features of RDD

- In-memory computation
- Lazy Evaluations
- Fault Tolerance
- Immutability
- Partitioning
- Persistence
- Location Stickiness

Spark RDD Transformations

- Functions that take an RDD as the input and produce one or many RDDs as the output
- Transformations are lazy operations on an RDD in Apache Spark
- Narrow - Only a limited subset of partitions used to calculate the result
    - map, filter, flatMap
- Wide - Data required to compute the records in a single partition may live in many partitions of the parent RDD
    - intersection, distinct, groupByKey

Spark RDD Actions

- Returns final result of RDD computations
- RDD operations that produce non-RDD values
- Steps
  - Triggers execution using lineage graph to load the data into original RDD
  - Carry out all intermediate transformations
  - Return final results to Driver program or write it out to file system
- Examples
  - first, take, reduce, collect

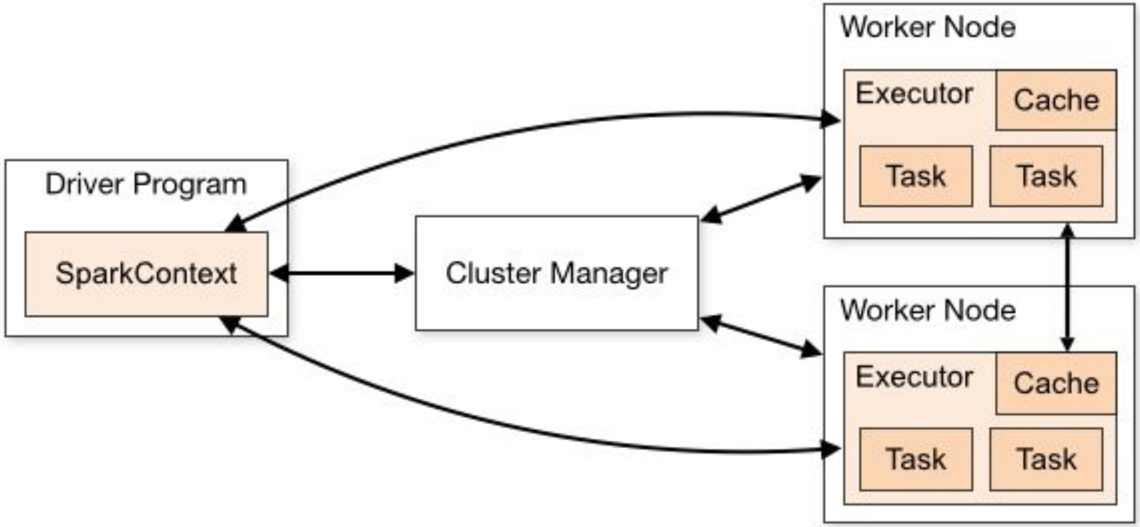
Limitation of Spark RDD

- No inbuilt optimization engine
- Handling structured data
- Performance limitation
- Storage limitation

How Spark Works?

- Spark uses master/slave architecture i.e. one central coordinator and many distributed workers
- The central coordinator is called the driver
- Drivers communicate with a potentially large number of distributed workers called executors
- Each executor is a separate process
- A Spark Application is a combination of driver and its own executors

Apache Spark on Clusters



Apache SparkContext

- It establishes a connection to the Spark Execution environment
- Used to create Spark RDDs, accumulators, and broadcast variables, access Spark services and run jobs
- The most important step of any Spark driver application is to generate SparkContext
- It allows your Spark Application to access Spark Cluster with the help of Resource Manager (Spark Standalone,YARN, Apache Mesos)

Functionalities of SparkContext

- Getting the current status of spark application
- Canceling the job
- Canceling the Stage
- Running job synchronously
- Running job asynchronously
- Accessing persistent RDD
- Accessing non persisting RDD

Spark Shell

- Spark application written in Scala
- Offers command line environment with auto-completion
- Extension of Scala REPL
- Automatic instantiation of SparkSession as spark and SparkContext as sc

Task

- A unit of work that is sent to the executor
- Same task is done over different partitions of RDD

Stage

- Stages are classified as computational boundaries
- All computation cannot be done in single stage
- It is achieved over many stages.

Job

- Parallel computation consisting of multiple tasks that get spawned in response to actions
- Each job gets divided into smaller sets of tasks called stages that depend on each other

Spark Application

- A self-contained computation that runs user-supplied code to compute a result
- Can have processes running on its behalf even when it’s not running a job
- Consists of multiple jobs

Apache Spark Driver

- main() method of the program runs in the driver
- Runs the user code that creates RDDs, and performs transformation and action, and also creates SparkContext
- The application is finished when the driver is terminated
- The two main key roles of drivers are:
    - Converting user program into the task
    - Scheduling task on the executor

Apache Spark Cluster Manager

- Spark relies on cluster manager to launch executors
- Jobs and action within a spark application are scheduled by Spark Scheduler in a FIFO fashion
- The scheduling can also be done in Round Robin fashion
- Resources used by a Spark application can be dynamically adjusted based on the workload

Apache Spark Executor

- Individual task in the given Spark job runs in the Spark executors
- Launched once in the beginning of Spark Application and run for the entire lifetime of an application
- Two main roles of the executors:
    - Runs the task that makes up the application and returns the result to the driver
    - Provide in-memory storage for RDDs that are cached by the user

Launching a Program in Spark

- Spark-submit is a program that can be used to launch application on a cluster
- can use all of Spark’s supported cluster managers
- Create jar of the project and run using spark-submit program

```
./bin/spark-submit \
  --class org.apache.spark.examples.SparkPi \
  --master local[8] \
  /path/to/examples.jar \
  100
```

Apache Spark on Clusters

- User submits an application using spark-submit
- main() method specified by the user is invoked and launches the driver program
- The driver program asks for the resources to the cluster manager that is required to launch executors
- The cluster manager launches executors on behalf of the driver program
- Based on the actions and transformation on RDDs, the driver sends work to executors in the form of tasks
- The executors process the task and the result is sent back to the driver through cluster manager