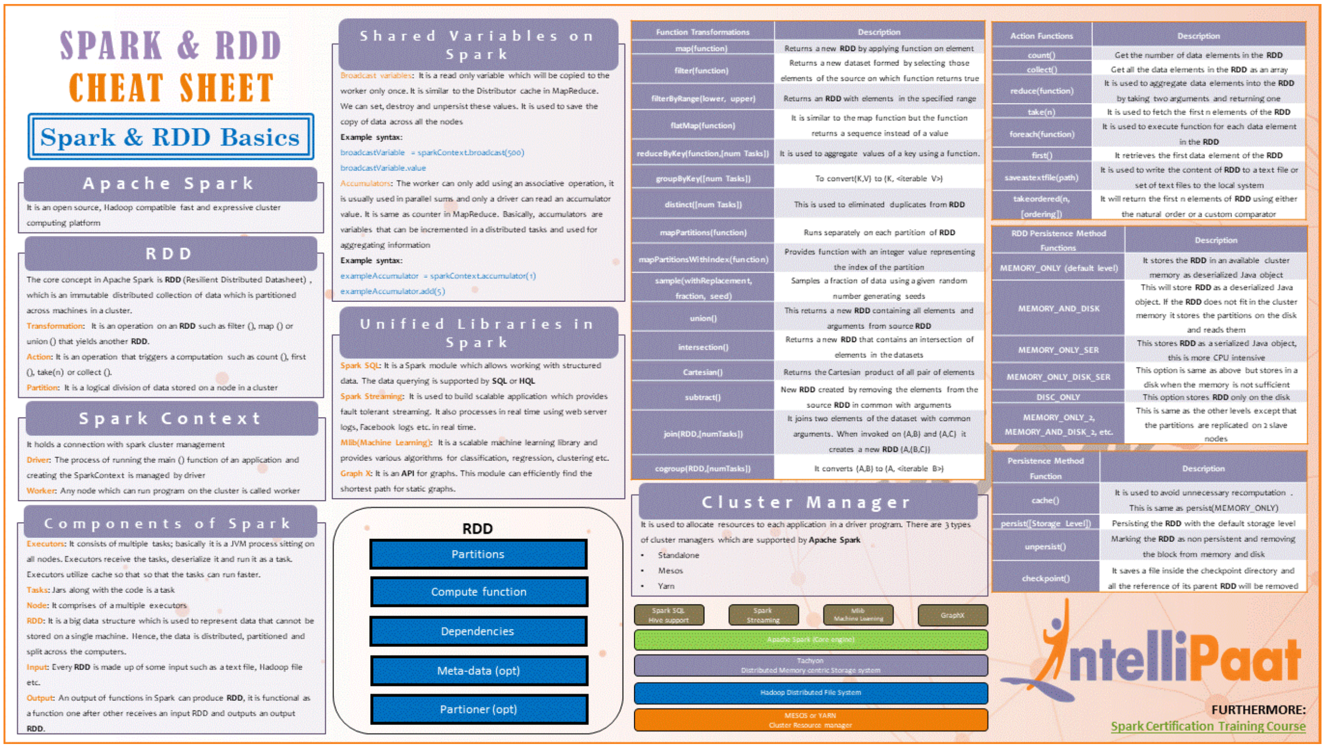
Spark and RDD Cheat Sheet

intellipaat.com/blog/tutorial/spark-tutorial/spark-and-rdd-cheat-sheet

Spark and RDD User Handbook

Are you a programmer experimenting in-memory computation on large clusters? If yes, then you must take Spark into your consideration. This Spark and RDD cheat sheet is designed for the one who has already started learning about memory management and using Spark as a tool. This sheet will be a handy reference for them.

You can also download the printable PDF of this Spark & RDD cheat sheet



SPARK & RDD
CHEAT SHEET

Spark & RDD Basics

Apache Spark

It is an open source, Hadoop compatible fast and expressive cluster computing platform

RDD

The core concept in Apache Spark is **RDD** (Resilient Distributed Datasheet) , which is an immutable distributed collection of data which is partitioned across machines in a cluster.

Transformation: It is an operation on an **RDD** such as filter (), map () or union () that yields another **RDD**.

Action: It is an operation that triggers a computation such as count (), first (), take(n) or collect ().

Partition: It is a logical division of data stored on a node in a cluster

Spark Context

It holds a connection with spark cluster management

Driver: The process of running the main () function of an application and creating the SparkContext is managed by driver

Worker: Any node which can run program on the cluster is called worker

Components of Spark

Executors: It consists of multiple tasks; basically it is a JVM process sitting on all nodes. Executors receive the tasks, deserialize it and run it as a task. Executors utilize cache so that so that the tasks can run faster.

Tasks: Jars along with the code is a task

Node: It comprises of a multiple executors

RDD: It is a big data structure which is used to represent data that cannot be stored on a single machine. Hence, the data is distributed, partitioned and split across the computers.

Input: Every **RDD** is made up of some input such as a text file, Hadoop file etc.

Output: An output of functions in Spark can produce **RDD**, it is functional as a function one after other receives an input RDD and outputs an output **RDD**.

Shared Variables on Spark

Broadcast variables: It is a read only variable which will be copied to the worker only once. It is similar to the Distributor cache in MapReduce. We can set, destroy and unpersist these values. It is used to save the copy of data across all the nodes

Example syntax:

```
broadcastVariable = sparkContext.broadcast(500)
broadcastVariable.value
```

Accumulators: The worker can only add using an associative operation, it is usually used in parallel sums and only a driver can read an accumulator value. It is same as counter in MapReduce. Basically, accumulators are variables that can be incremented in a distributed tasks and used for aggregating information

Example syntax:

```
exampleAccumulator = sparkContext.accumulator(1)
exampleAccumulator.add(5)
```

Unified Libraries in Spark

Spark SQL: It is a Spark module which allows working with structured data. The data querying is supported by **SQL** or **HQL**

Spark Streaming: It is used to build scalable application which provides fault tolerant streaming. It also processes in real time using web server logs, Facebook logs etc. in real time.

Mlib(Machine Learning): It is a scalable machine learning library and provides various algorithms for classification, regression, clustering etc.

Graph X: It is an **API** for graphs. This module can efficiently find the shortest path for static graphs.

RDD

Partitions

Compute function

Dependencies

Meta-data (opt)

Partioner (opt)

Function Transformations	Description
map(function)	Returns a new **RDD** by applying function on element
filter(function)	Returns a new dataset formed by selecting those elements of the source on which function returns true
filterByRange(lower, upper)	Returns an **RDD** with elements in the specified range
flatMap(function)	It is similar to the map function but the function returns a sequence instead of a value
reduceByKey(function,[num Tasks])	It is used to aggregate values of a key using a function.
groupByKey([num Tasks])	To convert(K,V) to (K, <iterable V>)
distinct([num Tasks])	This is used to eliminated duplicates from **RDD**
mapPartitions(function)	Runs separately on each partition of **RDD**
mapPartitionsWithIndex(function)	Provides function with an integer value representing the index of the partition
sample(withReplacement, fraction, seed)	Samples a fraction of data using a given random number generating seeds
union()	This returns a new **RDD** containing all elements and arguments from source **RDD**
intersection()	Returns a new **RDD** that contains an intersection of elements in the datasets
Cartesian()	Returns the Cartesian product of all pair of elements
subtract()	New **RDD** created by removing the elements from the source **RDD** in common with arguments
join(RDD,[numTasks])	It joins two elements of the dataset with common arguments. When invoked on (A,B) and (A,C) it creates a new **RDD** (A,(B,C))
cogroup(RDD,[numTasks])	It converts (A,B) to (A, <iterable B>)

Cluster Manager

It is used to allocate resources to each application in a driver program. There are 3 types of cluster managers which are supported by **Apache Spark**

- Standalone
- Mesos
- Yarn

Spark SQL
Hive support

Spark
Streaming

Mlib
Machine Learning

GraphX

Apache Spark (Core engine)

Tachyon
Distributed Memory centric Storage system

Hadoop Distributed File System

MESOS or YARN
Cluster Resource manager

Action Functions	Description
count()	Get the number of data elements in the **RDD**
collect()	Get all the data elements in the **RDD** as an array
reduce(function)	It is used to aggregate data elements into the **RDD** by taking two arguments and returning one
take(n)	It is used to fetch the first n elements of the **RDD**
foreach(function)	It is used to execute function for each data element in the **RDD**
first()	It retrieves the first data element of the **RDD**
saveastextfile(path)	It is used to write the content of **RDD** to a text file or set of text files to the local system
takeordered(n, [ordering])	It will return the first n elements of **RDD** using either the natural order or a custom comparator

RDD Persistence Method Functions	Description
MEMORY_ONLY (default level)	It stores the **RDD** in an available cluster memory as deserialized Java object
MEMORY_AND_DISK	This will store **RDD** as a deserialized Java object. If the **RDD** does not fit in the cluster memory it stores the partitions on the disk and reads them
MEMORY_ONLY_SER	This stores **RDD** as a serialized Java object, this is more CPU intensive
MEMORY_ONLY_DISK_SER	This option is same as above but stores in a disk when the memory is not sufficient
DISC_ONLY	This option stores **RDD** only on the disk
MEMORY_ONLY_2, MEMORY_AND_DISK_2, etc.	This is same as the other levels except that the partitions are replicated on 2 slave nodes

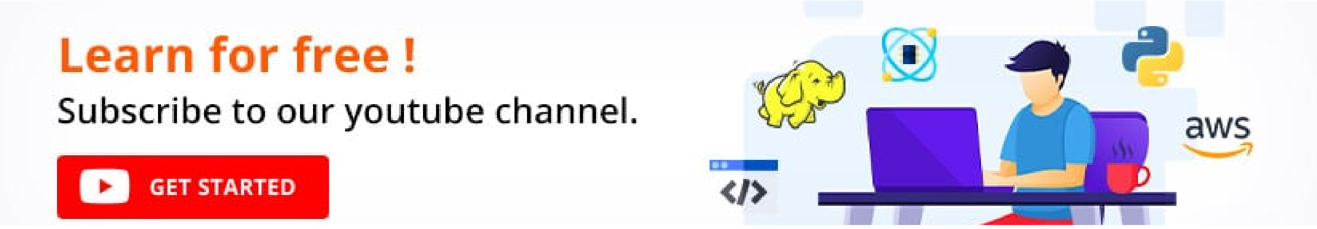
Persistence Method Function	Description
cache()	It is used to avoid unnecessary recomputation . This is same as persist(MEMORY_ONLY)
persist([Storage Level])	Persisting the **RDD** with the default storage level
unpersist()	Marking the **RDD** as non persistent and removing the block from memory and disk
checkpoint()	It saves a file inside the checkpoint directory and all the reference of its parent **RDD** will be removed

IntelliPaat

FURTHERMORE:
Spark Certification Training Course

Now, don’t worry if you are a beginner and have no idea about how Spark and RDD work. This cheat sheet includes all concepts you must know, from the basics, and will give you a quick reference to all of them.

Check out this insightful video on Spark Tutorial for Beginners:

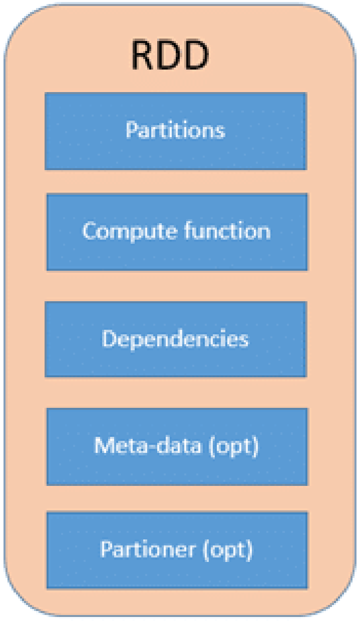


Key Concepts

- **Apache Spark:** It is an open-source, Hadoop-compatible, fast and expressive cluster computing platform.
- **Resilient Distributed Datasets (RDDs):** The core concept in Apache Spark is RDDs, which are the immutable distributed collections of data that is partitioned across machines in a cluster.
- **Transformation:** It is an operation performed on an RDD, such as filter(), map(), or union(), which yields another RDD.
- **Action:** It is an operation that triggers a computation such as count(), first(), take(n), or collect().
- **Partition:** It is a logical division of data stored on a node in a cluster.

RDD Components

- **SparkContext**: It holds a connection with Spark Cluster Management.
- **Driver**: The process of running the main() function of an application and creating the SparkContext is managed by the driver.
- **Worker**: Any node which can run the program on the cluster is called a worker.
- **Cluster Manager**: A cluster manager allocates resources to each application in a driver program. There are three types of cluster managers supported by Apache Spark:
    - **Standalone**
    - **Mesos**
    - **YARN**



Shared Variables in Spark

- **Broadcast variables**: Broadcast variables are read-only variables that will be copied to the worker only once. They are similar to the distributor cache in MapReduce. You can set, destroy, and unpersist these values. They are used to save the copies of data across all nodes.

  Example:

```
broadcastVariable = sparkContext.broadcast(500)
broadcastVariable.value
```

- **Accumulators**: The worker can only add using an associative operation; it is usually used in parallel sums, and only a driver can read an accumulator value. It is the same as the counter in MapReduce. Basically, accumulators are variables that can be incremented in distributed tasks and used for aggregating information.

  Example:

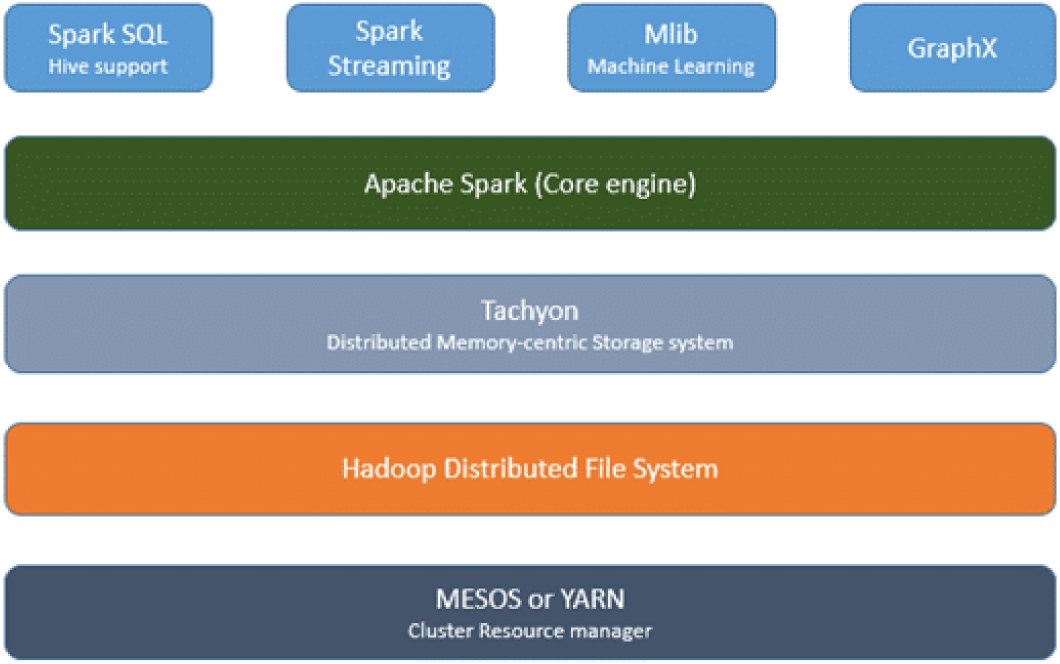
```
exampleAccumulator = sparkContext.accumulator(1)
exampleAccumulator.add(5)
```

Learn Apache Spark from Big Data and Spark Online Course in Hyderabad and be an Apache Spark Specialist!

Watch this Spark Video for Beginners:

Unified Libraries in Spark

- **Spark SQL**: It is a Spark module that allows working with structured data. Data querying is supported by SQL or HQL.
- **Spark Streaming**: It is used to build a scalable application that provides fault-tolerant streaming. It also processes using web server logs, Facebook logs, etc. in real time.
- **MLlib (Machine Learning)**: It is a scalable Machine Learning library and provides various algorithms for classification, regression, clustering, etc.
- **GraphX**: It is an API for graphs. This module can efficiently find the shortest path for static graphs.



Components of Spark

- **Executors**: Executors comprise multiple tasks; basically, it is a JVM process sitting on all nodes. Executors receive the tasks, deserialize them, and run them. Executors utilize cache so that the tasks can be run faster.
- **Tasks**: Jars, along with the code, are referred to as tasks.
- **Nodes**: Nodes consist of multiple executors.
- **RDDs:** RDD is a big data structure that is used to represent data, which cannot be stored on a single machine. Hence, the data is distributed, partitioned, and split across multiple computers.
- **Inputs:** Every RDD is made up of some inputs such as a text file, Hadoop file, etc.
- **Output:** The output of a function in Spark can produce an RDD; it is functional since a function, one after the other, receives an input RDD and outputs an output RDD.

Want to grasp detailed knowledge of Hadoop? Read this extensive Spark Tutorial!

Commonly Used Transformations

Function	Description
map(function)	Returns a new RDD by applying the function on each data element

filter(function)	Returns a new dataset formed by selecting those elements of the source on which the function returns true
filterByRange(lower, upper)	Returns an RDD with elements in the specified range, upper to lower
flatMap(function)	Similar to the map function but returns a sequence, instead of a value
reduceByKey(function,[num Tasks])	Aggregates the values of a key using a function
groupByKey([num Tasks])	Converts (K,V) to (K, <iterable V>)
distinct([num Tasks])	Eliminates duplicates from an RDD
mapPartitions(function)	Similar to map but runs separately on each partition of an RDD
mapPartitionsWithIndex(function)	Similar to the map partition but also provides the function with an integer value representing the index of the partition
sample(withReplacement, fraction, seed)	Samples a fraction of data using the given random number generating seeds
union()	Returns a new RDD containing all elements and arguments of the source RDD
intersection()	Returns a new RDD that contains an intersection of elements in the datasets
Cartesian()	Returns the Cartesian product of all pairs of elements
subtract()	Returns a new RDD created by removing the elements from the source RDD with common arguments
join(RDD,[numTasks])	Joins two elements of the dataset with common arguments; when invoked on (A,B) and (A,C), it creates a new RDD, (A,(B,C))
cogroup(RDD,[numTasks])	Converts (A,B) to (A, <iterable B>)

If you have any queries related to Spark and Hadoop, kindly refer to our Big Data Hadoop and Spark Community!

Commonly Used Actions

Function	Description
count()	Gets the number of data elements in an RDD
collect()	Gets all data elements of an RDD as an array
reduce(function)	Aggregates data elements into an RDD by taking two arguments and returning one
take(n)	Fetches the first *n* elements of an RDD
foreach(function)	Executes the function for each data element of an RDD
first()	Retrieves the first data element of an RDD
saveastextfile(path)	Writes the content of an RDD to a text file, or a set of text files, in the local system
takeordered(n, [ordering])	Returns the first *n* elements of an RDD using either the natural order or a custom comparator

Prepare yourself with these Apache Spark Interview Questions and Answers and excel in your career!

Persistence Methods

Function	Description
cache()	Avoids unnecessary recomputation; it is similar to persist(MEMORY_ONLY)
persist([Storage Level])	Persists an RDD with the default storage level
unpersist()	Marks an RDD as non-persistent and removes the block from memory and disk
checkpoint()	Saves a file inside the checkpoint directory and removes all the references of its parent RDD

RDD Persistence Methods

Function	Description

MEMORY_ONLY (default level)	Stores an RDD in an available cluster memory as a deserialized Java object
MEMORY_AND_DISK	Stores an RDD as a deserialized Java object; if the RDD does not fit in the cluster memory, it stores the partitions on the disk and reads them
MEMORY_ONLY_SER	Stores an RDD as a serialized Java object; it is more CPU intensive
MEMORY_ONLY_DISK_SER	Similar to the above but stores in a disk when the memory is not sufficient
DISC_ONLY	Stores an RDD only on the disk
MEMORY_ONLY_2, MEMORY_AND_ DISK_2, etc.	Similar to other levels, except that partitions are replicated on two slave nodes

Download a Printable PDF of this Cheat Sheet

With this, you have come to the end of the Spark and RDD Cheat Sheet. To get in-depth knowledge, check out our interactive, online Apache Spark Training that comes with 24/7 support to guide you throughout your learning period. Intellipaat’s Apache Spark training includes Spark Streaming, Spark SQL, Spark RDDs, and Spark Machine Learning libraries (Spark MLlib).

Intellipaat provides the most comprehensive Big Data and Spark Training in New York to fast-track your career!

Previous Next

Course Schedule

Name	Date	
Big Data Architect	2020-12-19 2020-12-20 (Sat-Sun) Weekend batch	View Details
Big Data Architect	2020-12-26 2020-12-27 (Sat-Sun) Weekend batch	View Details
Big Data Architect	2021-01-02 2021-01-03 (Sat-Sun) Weekend batch	View Details

Leave a Reply

Your email address will not be published. Required fields are marked *