Spark Cheat Sheet

Basic Transformations (return new RDDs – Lazy)

Where	Function	Description
RDD	map(function)	Return a new distributed dataset formed by passing each element of the source through a function.
RDD	filter(function)	Return a new dataset formed by selecting those elements of the source on which function returns true.
OrderedRDD Functions	filterByRange(lower, upper)	Returns an RDD containing only the elements in the inclusive range lower to upper.
RDD	flatMap(function)	Similar to map, but each input item can be mapped to 0 or more output items (so function should return a Seq rather than a single item).
RDD	mapPartitions(function)	Similar to map, but runs separately on each partition of the RDD.
RDD	mapPartitions WithIndex(function)	Similar to mapPartitions, but also provides function with an integer value representing the index of the partition.
RDD	sample(withReplacement, fraction, seed)	Sample a fraction of the data, with or without replacement, using a given random number generator seed.
RDD	union(otherDataset)	Return a new dataset that contains the union of the elements in the datasets.
RDD	intersection(otherDataset)	Return a new RDD that contains the intersection of elements in the datasets.
RDD	distinct([numTasks])	Return a new dataset that contains the distinct elements of the source dataset.
RDD	cartesian(otherDataset)	When called on datasets of types T and U, returns a dataset of (T,U) pairs (all pairs of elements).
RDD	pipe(command, [envVars])	Pipe each partition of the RDD through a shell command, e.g. a Perl or bash script.
RDD	coalesce(numPartitions)	Decrease the number of partitions in the RDD to numPartitions. Useful for running operations more efficiently after filtering down a large dataset.
RDD	repartition(numPartitions)	Reshuffle the data in the RDD randomly to create either more or fewer partitions and balance it across them. This always shuffles all data over the network.
PairRDD Functions	groupByKey([numTasks])	Returns a dataset of (K,Iterable<V>) pairs. Use reduceByKey or aggregateByKey to perform an aggregation (such as a sum or average).
PairRDD Functions	reduceByKey(function, [numTasks])	Returns a dataset of (K, V) pairs where the values for each key are aggregated using the given reduce function.
PairRDD Functions	aggregateByKey(zeroValue) (seqOp, combOp, [numTasks])	Returns a dataset of (K, U) pairs where the values for each key are aggregated using the given combine functions and a neutral "zero" value. Allows an aggregated value type that is different than the input value type.
OrderedRDD	sortByKey([ascending], [numTasks])	Returns a dataset of (K, V) pairs sorted by keys in ascending or descending order, as specified in the boolean ascending argument.
PairRDD Functions	join(otherDataset, [numTasks])	When called on datasets of type (K, V) and (K, W), returns a dataset of (K, (V, W)) pairs with all pairs of elements for each key. Outer joins are supported through leftOuterJoin, rightOuterJoin, and fullOuterJoin.
PairRDD Functions	cogroup(otherDataset, [numTasks])	When called on datasets of type (K, V) and (K, W), returns a dataset of (K, (Iterable<V>, Iterable<W>)) tuples.
OrderedRDD Functions	repartitionAndSortWithin Partitions(partitioner)	Repartition the RDD according to the given partitioner and, within each resulting partition, sort records by their keys. More efficient than calling repartition and then sorting.

Shared Variables	
Accumulators :	"Accumulators are variables that are only "added" to through an associative operation and can therefore be effciently supported in parallel.
Broadcast Variables :	"Broadcast variables allow the programmer to keep a read-only variable cached on each machine rather than shipping a copy of it with tasks."

Actions

Where	Function	Description
PairRDD Functions	countByKey()	Only available on RDDs of type (K, V). Returns a hashmap of (K, Int) pairs with the count of each key.

Basic Actions (return values – NOT Lazy)

Where	Function	Description
RDD	reduce(function)	Aggregate the elements of the dataset using a function (which takes two arguments and returns one).
RDD	collect()	Return all the elements of the dataset as an array at the driver program. Best used on sufficiently small subsets of data.
RDD	count()	Return the number of elements in the dataset.
RDD	countByValue()	Return the count of each unique value in this RDD as a local map of (value, count) pairs.
RDD	first()	Return the first element of the dataset (similar to take(1)).
RDD	take(n)	Return an array with the first n elements of the dataset.
RDD	takeSample(with Replacement, num, [seed])	Return an array with a random sample of num elements of the dataset.
RDD	takeOrdered (n, [ordering])	Return the first n elements of the RDD using either their natural order or a custom comparator.
RDD	saveAsTextFile(path)	Write the elements of the dataset as a text. Spark will call toString on each element to convert it to a line of text in the file.
SequenceFileRDD Functions	saveAsSequenceFile (path) (Java and Scala)	Write the elements of the dataset as a Hadoop SequenceFile in a given path. For RDDs of key-value pairs that use Hadoop's Writable interface.
RDD	saveAsObjectFile(path) (Java and Scala)	Write the elements of the dataset in a simple format using Java serialization, which can then be loaded using SparkContext. objectFile().
PairRDD Functions	countByKey()	Only available on RDDs of type (K, V). Returns a hashmap of (K, Int) pairs with the count of each key.
RDD	foreach(function)	Run a function on each element of the dataset. This is usually done for side effects such as updating an Accumulator.

Spark SQL	Spark SQL is a Spark module for structured data processing. Spark SQL is to execute SQL querie	
Dataset	A Dataset is a distributed collection of provides the benefits of RDDs (strong typing, ability to use powerful lambda functions) with the benefits of Spark SQL's optimized execution engine.	
DataFrame	DataFrame is a Dataset organized into named columns. It is conceptually equivalent to a table in a relational database or a data frame in R/Python, but with richer optimizations under the hood	
SparkSession	"import org.apache.spark. sql.SparkSession val spark = SparkSession .builder() .appName("Spark SQL basic example") .config("spark.some.config.option", "some-value") .getOrCreate()	The entry point into all functionality in Spark is the SparkSession class

RDD Persistence

Storage Level	Purpose
MEMORY_ONLY (default level)	Store RDD as deserialized Java objects. If the RDD does ot fit in memory, some partitions will not be cached and will be recomputed on the fly when needed.
MEMORY_AND_DISK	Store RDD as deserialized Java objects. If the RDD does not fit in memory, store the partitions that don't fit on disk, and load them when they're needed.
MEMORY_ONLY_SER	Store RDD as serialized Java objects. Generally more space-efficient than deserialized objects, but more CPU-intensive to read.
MEMORY_AND_DISK_SER	Similar to MEMORY_ONLY_SER, but spill partitions that don't fit in memory to disk instead of recomputing them on the fly each time they're needed.
DISK_ONLY	Store the RDD partitions only on disk.
MEMORY_ONLY_2, MEMORY_AND_DISK_2	Same as the levels above, but replicate each partition on two cluster nodes.

Streaming Transformations

Where	Function	Description
DStream	countByWindow(windowLength, slideInterval)	Return a sliding window count of elements in the stream.
DStream	reduceByWindow(function, windowLength, slideInterval)	Return a new single-element stream, created by aggregating elements in the stream over a sliding interval using function.
PairDStream Functions	reduceByKeyAndWindow(function, windowLength, slideInterval, [numTasks])	Returns a new DStream of (K, V) pairs where the values for each key are aggregated using the given reduce function over batches in a sliding window.
PairDStream Functions	reduceByKeyAndWindow(function, invFunc, windowLength, slideInterval, [numTasks])	A more efficient version of the above reduceByKeyAndWindow(). Only applicable to those reduce functions which have a corresponding "inverse reduce" function.
DStream	countByValueAndWindow (windowLength, slideInterval, [numTasks])	Returns a new DStream of (K, Long) pairs where the value of Checkpointing must be enabled for using this operation. each key is its frequency within a sliding window.
DStream	transform(function)	The transform operation (along with its variations like transformWith) allows arbitrary RDD-to-RDD functions to be applied on a Dstream.
PairDStream Functions	updateStateByKey(function)	The updateStateByKey operation allows you to maintain arbitrary state while continuously updating it with new information.

Spark Dataset

Where	Function	Description
Dataset/Dataframe	select (scala.collection. Seq<Column> cols)	Selects a set of column based expressions
Dataset/Dataframe	groupBy (Column... cols)	Groups the Dataset using the specified columns, so we can run aggregation on them
Dataset/Dataframe	sql	Executes a SQL query using Spark, returning the result as a DataFrame. The dialect that is used for SQL parsing can be configured with 'spark.sql.dialect'
Dataframe/SQL	Window	Window functions allow users of Spark SQL to calculate results such as the rank of a given row or a moving average over a range of input rows (API supported : ranking functions, analytic functions, aggregate functions) e.g. rank, denseRank, denseRank, ntile, rowNumber, cumeDist, firstValue, lastValue, lag, lead, sum, avg, min, max and count
Dataset/Dataframe	flatMap (scala.Function1<T,scala. collection.TraversableOnce<U>> func, Encoder<U> evidence$8)	Returns a new Dataset by first applying a function to all elements of this Dataset, and then flattening the results
Dataset/Dataframe	UDF	A collection of methods for registering user-defined functions (UDF)
Dataset/Dataframe	withColumn (String colName, Column col)	Returns a new Dataset by adding a column or replacing the existing column that has the same name.
Dataset/Dataframe	withColumnRenamed (String existing Name, String newName)	Returns a new Dataset with a column renamed. This is a no-op if schema doesn't contain existingName.
Dataset/Dataframe	joinWith (Dataset<U> other, Column condition)	Using inner equi-join to join this Dataset returning a Tuple2 for each pair where condition evaluates to true
Dataset/Dataframe	union (Dataset<T> other)	Returns a new Dataset containing union of rows in this Dataset and another Dataset
Dataset/Dataframe	write ()	Interface for saving the content of the non-streaming Dataset out into external storage
Dataset/Dataframe	filter (Column condition)	Filters rows using the given condition.
Dataset/Dataframe	where (Column condition)	Filters rows using the given condition.
Dataset/Dataframe	sort (Column... sortExprs)	Returns a new Dataset sorted by the given expressions
RelationalGrouped	cube (scala.collection. Seq<Column> cols)	Create a multi-dimensional cube for the current Dataset using Dataset the specified columns, so we can run aggregation on them
Dataset/Dataframe	stat ()	Returns a DataFrameStatFunctions for working statistic functions support
Dataset/Dataframe	printSchema ()	Prints the schema to the console in a nice tree format
Dataset/Dataframe	repartition (int numPartitions, scala.collection.Seq<Column > partitionExprs)	Returns a new Dataset partitioned by the given partitioning expressions into numPartitions.

Spark MLlib

Topic	Description
Data types	Vectors, points, matrices
Basic Statistics	Summary, correlations, sampling, testing and random data
Classifcation and regression	Includes SVMs, decision trees, naïve Bayes, etc...
Collaborative fltering	Commonly used for recommender systems
Clustering	Clustering is an unsupervised earning approach
Dimensionality reduction	"Dimensionality reduction is the process of reducing the number of variables under consideration"
"Feature extraction and transformation"	"Used in selecting a subset of relevant features (variables, predictors) for use in model construction"
Frequent pattern mining	Mining is usually among the first steps to analyze a large-scale dataset
Optimization	Different optimization methods can have different convergence guarantees
PMML model export	MLlib supports model export to Predictive Model Markup Language

Persist Methods

Where	Function	Description
RDD	cache()	Call cache on RDDs to avoid unnecessary recomputation. NOTE: This is the same as persist(MEMORY_ONLY).
RDD	persist([Storage Level])	Persist this RDD with the default storage level.
RDD	unpersist()	Mark the RDD as non-persistent, and remove its blocks from memory and disk.
RDD	checkpoint()	Save to a file inside the checkpoint directory and all references to its parent RDDs will be removed.

Extended RDDs w/ Custom Transformations and Actions

RDD Name	Description
CoGroupedRDD	A RDD that cogroups its parents. For each key k in parent RDDs, the resulting RDD contains a tuple with the list of values for that key.
EdgeRDD	Storing the edges in columnar format on each partition for performance. It may additionally store the vertex attributes associated with each edge.
JdbcRDD	An RDD that executes an SQL query on a JDBC connection and reads results. For usage example, see test case JdbcRDDSuite.
ShuffledRDD	The resulting RDD from a shuffle.
VertexRDD	Ensures that there is only one entry for each vertex and by pre-indexing the entries for fast, efficient joins.

Key-Value Operations | Trasformations

Where	Function	Description
PairRDD Functions	groupByKey([numTasks])	Returns a dataset of (K,Iterable<V>) pairs. Use `reduceByKey or aggregateByKey to perform an aggregation (such as a sum or average).
PairRDD Functions	reduceByKey(function, [numTasks])	Returns a dataset of (K, V) pairs where the values for each key are aggregated using the given reduce function.
PairRDD Functions	aggregateByKey(zeroValue)(seqOp, combOp, [numTasks])	Returns a dataset of (K, U) pairs where the values for each key are aggregated using the given combine functions and a neutral "zero" value. Allows an aggregated value type that is different than the input value type.
PairRDD Functions	join(otherDataset, [numTasks])	When called on datasets of type (K, V) and (K, W), returns a dataset of (K, (V, W)) pairs with all pairs of elements for each key. Outer joins are supported through leftOuterJoin, rightOuterJoin, and fullOuterJoin.
PairRDD Functions	cogroup(otherDataset, [numTasks])	When called on datasets of type (K, V) and (K, W), returns a dataset of (K, (Iterable<V>, Iterable<W>)) tuples.

RDD:	"Resilient Distributed Datasets, - Immutable collections of objects spread across a cluster • Built through parallel transformations (map, filter, etc) • Automatically rebuilt on failure • Controllable persistence (e.g. caching in RAM)"
Operations:	Trasformations: Lazy operations to build RDDs from other RDDs Actions: Return a result or write it to storage