SPARK PROGRAMMING CHEAT SHEET-kirupagaran.com

Transformations (return new RDDs – Lazy)	
Where	**Function**
RDD	map(function)
RDD	filter(function)
OrderedRDD Functions	filterByRange(lower, upper)
RDD	flatMap(function)
RDD	mapPartitions(function)
RDD	mapPartitionsWithIndex(function)
RDD	sample(withReplacement, fraction, seed)
RDD	union(otherDataset)
RDD	intersection(otherDataset)
RDD	distinct([numTasks])
RDD	cartesian(otherDataset)
RDD	pipe(command, [envVars])
RDD	coalesce(numPartitions)
RDD	repartition(numPartitions)
PairRDD Functions	groupByKey([numTasks])
PairRDD Functions	reduceByKey(function, [numTasks])
PairRDD Functions	aggregateByKey(zeroValue)(seqOp, combOp, [numTasks])
OrderedRDD Functions	sortByKey([ascending], [numTasks])
PairRDD Functions	join(otherDataset, [numTasks])
PairRDD Functions	cogroup(otherDataset, [numTasks])
OrderedRDD Functions	repartitionAndSortWithinPartitions(partitioner)
Actions (return values – NOT Lazy)	
Where	**Function**
RDD	reduce(function)
RDD	collect()
RDD	count()
RDD	countByValue()
RDD	first()
RDD	take(n)
RDD	takeSample(withReplacement, num, [seed])
RDD	takeOrdered(n, [ordering])
RDD	saveAsTextFile(path)
SequenceFileRDD Functions	saveAsSequenceFile(path) (Java and Scala)
RDD	saveAsObjectFile(path) (Java and Scala)
PairRDD Functions	countByKey()
RDD	foreach(function)
Persistence Methods	
Where	**Function**
RDD	cache()
RDD	persist([Storage Level])
RDD	unpersist()
RDD	checkpoint()
Additional Transformation and Actions	
Where	**Function**
SparkContext	doubleRDDToDoubleRDDFunctions
SparkContext	numericRDDToDoubleRDDFunctions
SparkContext	rddToPairRDDFunctions
SparkContext	hadoopFile()
SparkContext	hadoopRDD()
SparkContext	makeRDD()
SparkContext	parallelize()

SparkContext	textFile()
SparkContext	wholeTextFiles()
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.
Streaming Transformations	
Where	**Function**
DStream	window(windowLength, slideInterval)
DStream	countByWindow(windowLength, slideInterval)
DStream	reduceByWindow(function, windowLength, slideInterval)
PairDStream Functions	reduceByKeyAndWindow(function, windowLength, slideInterval, [numTasks])
PairDStream Functions	reduceByKeyAndWindow(function, invFunc, windowLength, slideInterval, [numTasks])
DStream	countByValueAndWindow(windowLength, slideInterval, [numTasks])
DStream	transform(function)
PairDStream Functions	updateStateByKey(function)
RDD Persistence	
Storage Level	**Meaning**
MEMORY_ONLY (default level)	Store RDD as deserialized Java objects. If the RDD does not 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, etc...	Same as the levels above, but replicate each partition on two cluster nodes.
Shared Data	
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.	
Language	**Create, Evaluate**
Scala	val broadcastVar = sc.broadcast(Array(1, 2, 3))
	broadcastVar.value
Java	Broadcast<int[]> broadcastVar = sc.broadcast(new int[] {1, 2, 3});

	broadcastVar.value();
Python	broadcastVar = sc.broadcast([1, 2, 3])
	broadcastVar.value

Accumulators: Accumulators are variables that are only "added" to through an associative operation and can therefore be efficiently supported in parallel.

Language	**Create, Add, Evaluate**
Scala	val accum = sc.accumulator(0, My Accumulator)
	sc.parallelize(Array(1, 2, 3, 4)).foreach(x => accum += x)
	accum.value
Java	Accumulator<Integer> accum = sc.accumulator(0);
	sc.parallelize(Arrays.asList(1, 2, 3, 4)).foreach(x -> accum.add(x))
	accum.value();
Python	accum = sc.accumulator(0)
MLlib Reference	
Topic	**Description**
Data types	Vectors, points, matrices.
Basic Statistics	Summary, correlations, sampling, testing and random data.
Classification and regression	Includes SVMs, decision trees, naïve Bayes, etc...
Collaborative filtering	Commonly used for recommender systems.
Clustering	Clustering is an unsupervised learning 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.