Day 4 – RDD process in Spark

1. RDDs support two types of operations: transformations, which create a new dataset from an existing one, and actions, which return a value to the driver program after running a computation on the dataset. For example, map is a transformation that passes each dataset element through a function and returns a new RDD representing the results. On the other hand, reduce is an action that aggregates all the elements of the RDD using some function and returns the final result to the driver program (although there is also a parallel reduceByKey that returns a distributed dataset).
2. All transformations in Spark are lazy, in that they do not compute their results right away. Instead, they just remember the transformations applied to some base dataset (e.g. a file). The transformations are only computed when an action requires a result to be returned to the driver program. This design enables Spark to run more efficiently. For example, we can realize that a dataset created through map will be used in a reduce and return only the result of the reduce to the driver, rather than the larger mapped dataset.
3. By default, each transformed RDD may be recomputed each time you run an action on it. However, you may also persist an RDD in memory using the persist (or cache) method, in which case Spark will keep the elements around on the cluster for much faster access the next time you query it. There is also support for persisting RDDs on disk, or replicated across multiple nodes.
4. Why Spark RDD?
   1. Spark makes use of data abstraction through RDDs to achieve faster and more efficient performance than Hadoop’s MapReduce.
   2. RDDs support in-memory processing.  Accessing data from memory is 10 to 100 times faster than accessing data from a network or disk.  Data access from disk often occurs in Hadoop’s MapReduce-based processing.
   3. In addition to performance gains, working through an abstraction layer provides a convenient and consistent way for developers and engineers to work with a variety of data sets.
5. When to use Spark RDDs?
   1. RDDs are utilized to perform computations on an RDD dataset through Spark Actions such as a count or reduce when answering questions such as “how many times did xyz happen?” or “how many times did xyz happen by location?”
   2. Often, RDDs are transformed into new RDDs in order to better prepare datasets for future processing downstream in the processing pipeline.  To reuse a previous example, let’s say you want to examine North America customer data and you have an RDD of all worldwide customers in memory.  It could be beneficial from a performance perspective to create a new RDD for North America only customers instead of using the much larger RDD of all worldwide customers.
   3. Depending on the Spark operating environment and RDD size, RDDs should be cached (via cache function) or persisted to disk when there is an expectation for the RDD to be utilized more than once.