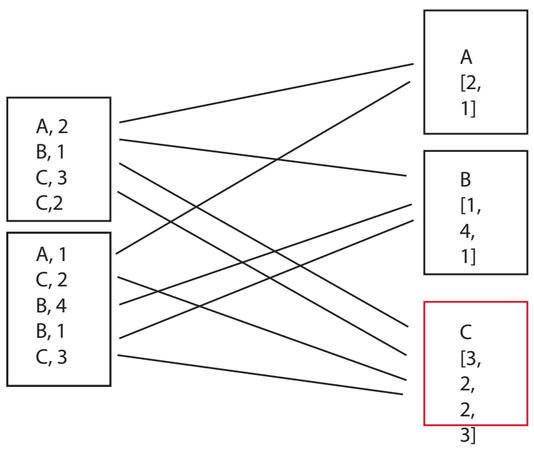
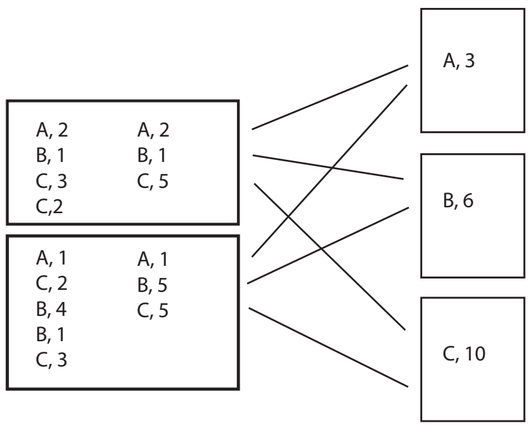
A few common aggreation methods for Spark RDDs.

groupByKey()
https://spark.apache.org/docs/latest/api/python/_modules/pyspark/rdd.html#RDD.groupByKey

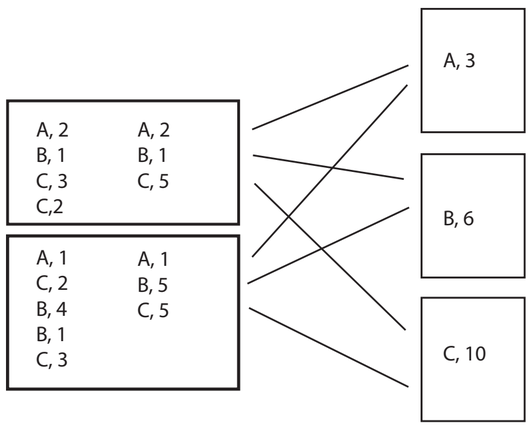


Warning: if there is extreme key skew, all values for a given key might not fit on a worker

reduceByKey(Func)
https://spark.apache.org/docs/latest/api/python/_modules/pyspark/rdd.html#RDD.reduceByKey



combineByKey(createCombiner, mergeValue, mergeCombiners)
https://spark.apache.org/docs/latest/api/python/_modules/pyspark/rdd.html#RDD.combineByKey



- C{createCombiner}, which turns a V into a C (e.g., creates a one-element list)

- C{mergeValue}, to merge a V into a C (e.g., adds it to the end of a list)

- C{mergeCombiners}, to combine two C's into a single one (e.g., merges the lists)

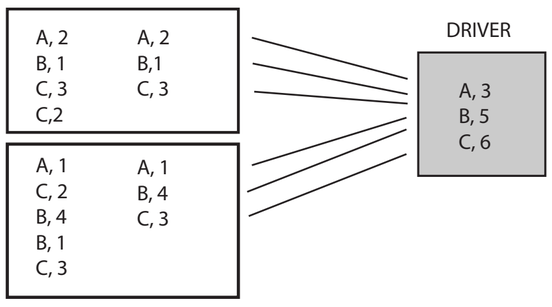
aggregateByKey(zeroValue, seqOp, combOp)
https://spark.apache.org/docs/latest/api/python/_modules/pyspark/rdd.html#RDD.aggregateByKey

start value,
seqOp -> within partition function,
combOp -> across partition function

ex: aggregateByKey(0,max,add)

foldByKey(zeroValue, Func)
https://spark.apache.org/docs/latest/api/python/_modules/pyspark/rdd.html#RDD.foldByKey

Calls combineByKey, but allows us to use a zero value which can be added to the result an arbitrary number of times, and must not change the result (eg. 0 for addition, 1 for multiplication)



Warning: if the results from the executors are too large, they can take down the driver with an `OutOfMemoryError`

treeAggregate(zeroValue, seqOp, combOp, depth)
https://spark.apache.org/docs/latest/api/python/_modules/pyspark/rdd.html#RDD.treeAggregate

Same as aggregate except it "pushes down" some of the subaggregations (creating a tree from executor to executor) before performing final aggregations on the driver.