MapReduce

More Detailed MapReduce Dataflow

- The number of **reducers** can be not only single one but also multiple. When it happens, the map tasks partition their output:
  1. One partition for each reduce task as the output from map tasks are partitioned into one large key-value pairs and are merged altogether.
  2. For each key-value pairs output from map tasks, the key-value pairs with the same key have to be partitioned into the same partition.

     For example, we have three pairs output p1-<k1, v1>, p2-<k1, v2>, p3-<k2, v3> p1 and p2 have to be partitioned into the same partition while p3 does not have to. Otherwise, in the different reducers, v1 and v2 will not be able to do the arithmetic operation.
  3. Partitioning can be controlled by a user-defined partitioning function.

Shuffle

Shuffle is the process of data redistribution where,

- Guarantee each reducer obtains all values associated with the same key (mentioned above).
- Essentiall process for all of the operations requiring grouping (e.g., word count, compute avg)

Shuffle in Hadoop

- Happens between each **Map** and **Reduce** phase
- Use Shuffle and Sort mechanism (output of each Mapper are sorted by the key)
- Use combiner to reduce the amount of data shuffled
  - Combiner combines key-value pairs with the same key in each partition (reduce by key)
  - Might need to combine the extra value (e.g., total amount for average)
  - HAVE TO BE DONE BY USER

Shuffle in Spark

- Triggere by certain operations
  - Distinct, Join, Repartition, all *By,* ByKey
  - Hash/Sort shuffle
  - [Exclude] Tungsten shuffle-sort

Hash Shuffle

- Data are initially hash partitioned on the **Map** side
- Then there are total F many files are created to store the partitioned data portion

$$\text{F (\# of files created)} = \text{\# of mappers} \times \text{\# of reducers}$$

- Use *Consolidate Files* to reduce the number of files created

  from

$$M \times R$$

  to

$$\text{\# Executor} \times \frac{\#Core}{\#CPU} \times \text{\# Reducer}$$