**Real-World Applications**

**Question 1: There are two datasets: A is the large one, B is small enough to fit in the memory of the cluster node. What type of join do you choose to make their intersection A&B?**

**Records in A: keyA, valueA**

**Records in B: keyB, valueB**

**Records in the result:**

**key (=keyA=keyB), valueA, valueB**

Map

**Question 2: There are two datasets: A is the large one, B is small enough to fit in the memory of the cluster node. What type of join do you choose to make the difference B\A (records from B not found in A)?**

**A: keyA, valueA**

**B: keyB, valueB**

**Result: keyB, null, valueB**

Reduce

**Question 3: What parameters do you specify in the Hadoop Streaming command to perform Secondary sort (i.e. sort by two fields of the key, partition by the first field)?**

-D stream.num.map.output.key.fields=2

-D mapred.output.key.comparator.class=org.apache.hadoop.mapred.lib.KeyFieldBasedComparator

-D mapred.text.key.comparator.options=-k1,2

-D mapred.text.key.partitioner.options=-k1,1

-partitioner org.apache.hadoop.mapred.lib.KeyFieldBasedPartitioner

**Question 4: When is Secondary Sort really useful?**

When you join two datasets with a Reduce-side join and one of them has many records with repeating keys

When you want to avoid containers in memory on the reducers and therefore decrease the memory required by your tasks.

**Question 5: In what type of join could this script be used?**

**for line in sys.stdin:**

**key, tag, value = line.strip().split('\t', 2)**

**if key != current\_key:**

**if current\_value is not None:**

**print “%s\t%s” % (current\_key, current\_value)**

**current\_key = key**

**current\_value = None**

**if tag == 0:**

**continue**

**current\_value = min(current\_value, value) if current\_value is not None else value**

**if current\_value is not None:**

**print “%s\t%s” % (current\_key, current\_value)**

Reduce