MapReduce

Combiners

Combiners are very powerful in that they reduce the IO between the Mappers and Reducers.

1. The input/output key and value types must match the output types of your Mapper.

2. Combiners can only get its input from one Mapper.

3. Combiners can only be used on the functions that are commutative and associative.
    1. The "Commutative Laws" say we can **swap numbers** over and still get the same answer ...

       $a + b = b + a$

       $a \times b = b \times a$

    2. The "Associative Laws" say it doesn't matter how we group the numbers (i.e. which we calculate first) ...

       $(a + b) + c = a + (b + c)$

       $(a \times b) \times c = a \times (b \times c)$

Combiners

For example, imagine your map tasks find the maximum temperature for a given year :

Node 1's Map output:
(1971, 20)
(1971, 05)
(1971, 30)

Node 2's Map output:
(1971, 10)
(1971, 15)

The reduce function would get this input after the shuffle phase:
(1971, [05, 10, 15, 20, 30])
and the reduce function would output:
(1971, 30)

But if you used a combiner, the reduce function would have gotten smaller input to work with after the shuffle phase:
(1971, [30, 15])
and the output from Reduce would be the same.

Hadoop OLD API and NEW API

Prior to Hadoop 0.20.x, a Map class had to extend a MapReduceBase and implement a Mapper as such:

```
public static class Map extends MapReduceBase implements Mapper {
...
}
```

and similarly, a map function had to use an OutputCollector and a Reporter object to emit **(key,value)** pairs and send progress updates to the main program. A typical map function looked like:

```
public void map(K1, V1, OutputCollector o, Reporter r) throws IOException {
  ...
  output.collect(key,value);
}
```

Package
old API uses *org.apache.hadoop.mapred*

Hadoop OLD API and NEW API

With the new Hadoop API, here is how a Map class is defined :

```
public class MapClass extends Mapper {
  ...
}
```

Package
new API uses *org.apache.hadoop.mapreduce*

and a map function uses Context objects to emit records and send progress updates. A typical map function is now defined as:

```
public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
  ...
  context.write(key,value);
}
```

All of the changes for a Mapper above go the same way for a Reducer.

Hadoop OLD API and NEW API

Earlier, a map reduce job was configured through a **JobConf** object and the job control was done using an instance of **JobClient**. The main body of a driver class used to look like:

```
JobConf conf = new JobConf(Driver.class);
conf.setPropertyX(..);
conf.setPropertyY(..);
...
...
...
JobClient.runJob(conf);
```

Hadoop OLD API and NEW API

In the new Hadoop API, the same functionality is achieved as follows:

```
Configuration conf = new Configuration();
Job job = new Job(conf);
job.setJarByClass(Driver.class);
job.setPropertyX(..);
job.setPropertyY(..);
...
...
...
job.waitForCompletion(true);
```

Hadoop OLD API and NEW API

OutPut file Name

In the new API map outputs are named ***part-m-nnnnn***, and reduce outputs are named ***part-r-nnnnn*** (where nnnnn is an integer designating the part number, starting from zero).

In the old API both map and reduce outputs are named ***part-nnnnn***