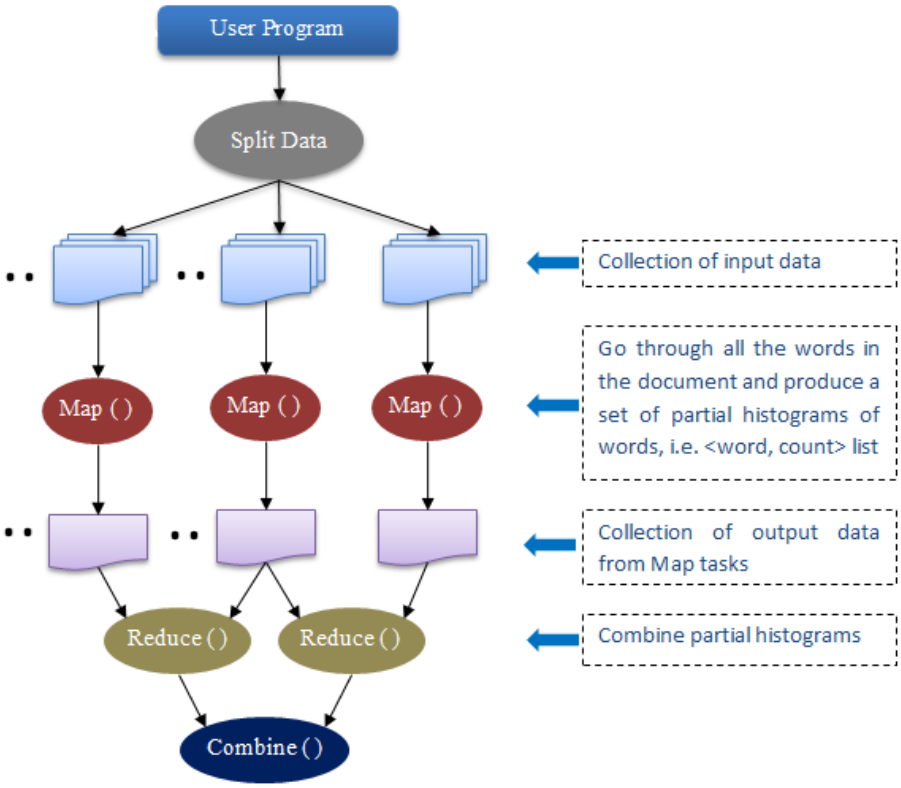
Map Reduce : Word Count

Task Requirement:

Write an application that takes input file(space seperated) from hdfs, the application should find word count on top of file and store the output to hdfs.

A MapReduce job usually splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically both the input and the output of the job are stored in a file-system.



Map Reduce work Flow

Hadoop Map Reduce: WordCount

A MapReduce job usually splits the input data-set into independent chunks, here the input data sets are splitted by 'space', which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Reduce task counts the number of occurence of each word and finally stores the output into FileSystem.

Map Function:

```
public void map(LongWritable key, Text value,OutputCollector<Text, IntWritable> output, Reporter r) throws IOException {
        String s = value.toString();
                for(String word:s.split(" ")){
                   if(word.length()>0){
                      output.collect(new Text(word), new IntWritable(1));
                   }
          }
    }
```

The map function split the data by space ,OutputCollector collects the output of map function i.e intermediate output. which is passed to reducer by map Reduce framework. Reporter report progress and update counters, status information etc.

Reduce Function:

```
public void reduce(Text key, Iterator<IntWritable> values,OutputCollector<Text, IntWritable> output, Reporter r)
throws IOException {
              int count = 0;
             while (values.hasNext()) {
             IntWritable i = values.next();
            count += i.get();
             }
          output.collect(key, new IntWritable(count));
      }
```

Reduce function takes the intermediate output and counts the no of times the word occurs. OutputCollector collects the output of reduce function.

Driver program:

Although the mapper and reducer implementations are all we need to perform the MapReduce job, there is one more piece of code necessary in MapReduce: the driver which communicates with the Hadoop framework and specifies the configuration elements required to run a MapReduce job. This involves aspects such as telling Hadoop which Mapper and Reducer classes to use, where to find the input data and in what format, and where to place the output data and how to format it.

```
JobConf conf = new JobConf(WordCount.class);
FileInputFormat.setInputPaths(conf, new Path(args[0]));
FileOutputFormat.setOutputPath(conf, new Path(args[1]));
conf.setMapperClass(WordMapper.class);
conf.setReducerClass(WordReducer.class);
conf.setMapOutputKeyClass(Text.class);
conf.setMapOutputValueClass(IntWritable.class);
conf.setOutputKeyClass(Text.class);
conf.setOutputValueClass(IntWritable.class);
JobClient.runJob(conf);
```

JobConf is the primary interface for a user to describe a map-reduce job to the hadoop framework, JobClient provides facilities to submit jobs, track their progress etc. runjob(), submits the job and returns only after the job has completed.