

# 과제2-2

## 목표

1. invertedIndex 분산처리를 해보자

## 간단설명

hadoop을 이용하여 각 문서에 어느 위치에 어느 단어가 있는 지 알려주는 목차를 만들었다

## 결과

```
hadoop@ubuntu:~/Project$ hdfs dfs -cat inverted_test_out/part-r-00000 | more
All :wordcount_test:5257
Almighty :wordcount_test:183
Americans :wordcount_test:5729
Americans :wordcount_test:5729, wordcount_test:5638
Americans: :wordcount_test:6812
Americas.: :wordcount_test:2921
And :wordcount_test:5812
And :wordcount_test:5812, wordcount_test:2931
And :wordcount_test:5812, wordcount_test:2931, wordcount_test:6794
And :wordcount_test:5812, wordcount_test:2931, wordcount_test:6794, wordcount_test:427
But :wordcount_test:4301
But :wordcount_test:4301, wordcount_test:1927
But :wordcount_test:4301, wordcount_test:1927, wordcount_test:2735
But :wordcount_test:4301, wordcount_test:1927, wordcount_test:2735, wordcount_test:5463
But :wordcount_test:4301, wordcount_test:1927, wordcount_test:2735, wordcount_test:5463, word
count_test:3819
Divided :wordcount_test:1509
East :wordcount_test:6223
Finally, :wordcount_test:7828
Finally, :wordcount_test:7828, wordcount_test:3389
For :wordcount_test:316
For :wordcount_test:316, wordcount_test:151
For :wordcount_test:316, wordcount_test:151, wordcount_test:3782
I :wordcount_test:155
I :wordcount_test:155, wordcount_test:6468
I :wordcount_test:155, wordcount_test:6468, wordcount_test:6524
If :wordcount_test:2419
In :wordcount_test:5482
In :wordcount_test:5482, wordcount_test:6332
Let :wordcount_test:2814
Let :wordcount_test:2814, wordcount_test:4338
Let :wordcount_test:2814, wordcount_test:4338, wordcount_test:710
Let :wordcount_test:2814, wordcount_test:4338, wordcount_test:710, wordcount_test:4265
Let :wordcount_test:2814, wordcount_test:4338, wordcount_test:710, wordcount_test:4265, wordcount_test:4439
Let :wordcount_test:2814, wordcount_test:4338, wordcount_test:710, wordcount_test:4265, wordcount_test:4439, wordcount_test:1147
Let :wordcount_test:2814, wordcount_test:4338, wordcount_test:710, wordcount_test:4265, wordcount_test:4439, wordcount_test:1147, wordcount_test:4864
Let :wordcount_test:2814, wordcount_test:4338, wordcount_test:710, wordcount_test:4265, wordcount_test:4439, wordcount_test:1147, wordcount_test:4864, wordcount_test:4653
Nor :wordcount_test:5318
North :wordcount_test:6206
South, :wordcount_test:6216
The :wordcount_test:283
The :wordcount_test:283, wordcount_test:5789
The :wordcount_test:283, wordcount_test:5789, wordcount_test:6625
To :wordcount_test:1388
To :wordcount_test:1388, wordcount_test:2122
--More--
```

## code

```
package ssafy;

import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileSplit;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
```

```

public class InvertedIndex {
    /*
    Object, Text : input key-value pair type (always same (to get a line of input
file))
    Text, IntWritable : output key-value pair type
    */
    public static class TokenizerMapper
        extends Mapper<Object,Text,Text,Text> {
        private Text word = new Text();
        private Text pos = new Text();
        private String filename;
        protected void setup(Context context) throws IOException,
InterruptedException{
            filename =((FileSplit)context.getInputSplit()).getPath().getName();
        }

        // map function (Context -> fixed parameter)
        public void map(Object key, Text value, Context context)
            throws IOException, InterruptedException {

            // value.toString() : get a line
            StringTokenizer itr = new StringTokenizer(value.toString(),"
,true);

            long p = ((LongWritable)key).get();
            while ( itr.hasMoreTokens() ) {
                String token = itr.nextToken();
                word.set(token.trim());
                if (! token.equals(" ")) {
                    pos.set(filename+": "+p);
                    context.write(word,pos);
                }
                p+= token.length();
            }
        }
    }

    /*
    Text, IntWritable : input key type and the value type of input value list
    Text, IntWritable : output key-value pair type
    */
    public static class ConcatenatorReducer
        extends Reducer<Text,Text,Text,Text> {

        // variables
        private Text list = new Text();

        // key : a distinct word
        // values : Iterable type (data list)
        public void reduce(Text key, Iterable<Text> values, Context context)
            throws IOException, InterruptedException {
            String s = new String();
            int comma = 0;

            for ( Text val : values ) {
                if(comma ==0){
                    comma = 1;

```

```

        s +=(":"+val.toString());
    } else {
        s +=(",    "+val.toString());
    }
    list.set(s);
    context.write(key,list);
}
}
}

/* Main function */
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    String[] otherArgs = new
GenericOptionsParser(conf,args).getRemainingArgs();
    if ( otherArgs.length != 2 ) {
        System.err.println("Usage: <in> <out>");
        System.exit(2);
    }
    /*FileSystem hdfs = FileSystem.get(conf);
    Path output = new Path(otherArgs[1]);
    if (hdfs.exists(output))
        hdfs.delete(output,true);
    */

    Job job = new Job(conf,"word count");
    job.setJarByClass(InvertedIndex.class);

    // let hadoop know my map and reduce classes
    job.setMapperClass(TokenizerMapper.class);
    job.setReducerClass(ConcatenatorReducer.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);

    // set number of reduces
    job.setNumReduceTasks(2);

    // set input and output directories
    FileInputFormat.addInputPath(job,new Path(otherArgs[0]));
    FileOutputFormat.setOutputPath(job,new Path(otherArgs[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1 );
}
}

```