LESSON 1

워드카운트 소스코드

워드카운트 소스코드

예제코드(2.7.2)

예제위치

$HADOOP_HOME/share/hadoop/mapreduce/sources

```
$ cd  $HADOOP_HOME/share/hadoop/mapreduce/sources
$ jar xvf hadoop-mapreduce-examples-2.7.2-sources.jar
$ nano org/apache/hadoop/examples/WordCount.java
```

워드카운트 소스코드

예제코드(2.7.2)

```
/**
 * Licensed to the Apache Software Foundation (ASF) under one
 * or more contributor license agreements.  See the NOTICE file
 * distributed with this work for additional information
 * regarding copyright ownership.  The ASF licenses this file
 * to you under the Apache License, Version 2.0 (the
 * "License"); you may not use this file except in compliance
 * with the License.  You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package org.apache.hadoop.examples;

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.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.FileInputFormat;
```

워드카운트 소스코드

예제코드(2.7.2)

```
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

public class WordCount {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }

  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context
                       ) throws IOException, InterruptedException {
```

워드카운트 소스코드

예제코드(2.7.2)

```
                    Context context
                    ) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }

  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: wordcount <in> [<in>...] <out>");
      System.exit(2);
    }
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    for (int i = 0; i < otherArgs.length - 1; ++i) {
      FileInputFormat.addInputPath(job, new Path(otherArgs[i]));
    }
    FileOutputFormat.setOutputPath(job,
      new Path(otherArgs[otherArgs.length - 1]));
```

워드카운트 소스코드

예제코드(2.7.2)

```
    FileOutputFormat.setOutputPath(job,
      new Path(otherArgs[otherArgs.length - 1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

워드카운트 소스코드

워드카운트 1.2.1과 2.7.2 버전 비교

$ diff WordCount.java.1.2.1 WordCount.java.2.7.2

```
<
69,70c71,72
<     if (otherArgs.length != 2) {
<       System.err.println("Usage: wordcount <in> <out>");
---
>     if (otherArgs.length < 2) {
>       System.err.println("Usage: wordcount <in> [<in>...] <out>");
73c75
<     Job job = new Job(conf, "word count");
---
>     Job job = Job.getInstance(conf, "word count");
80,81c82,86
<     FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
<     FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
---
>     for (int i = 0; i < otherArgs.length - 1; ++i) {
>       FileInputFormat.addInputPath(job, new Path(otherArgs[i]));
>     }
>     FileOutputFormat.setOutputPath(job,
>       new Path(otherArgs[otherArgs.length - 1]));
```

사실상 동일 = 코드는 변화가 거의 없음

워드카운트 소스코드

Sample Mapper

```
 public static class MyMapper extends
Mapper<K1,V1,K2,V2> {
   K2 k2 = new K2();
   V2 v2 = new V2();

   publc void map(K1 key, V1 value, Context context) {
   ...
   context.write(k2, v2);
  }
}
```

워드카운트 소스코드

Sample Reducer

```
 public static class MyReducer extends
Reducer<K2,V2,K3,V3> {
   K3 k3 = new K3();
   V3 v3 = new V3();

   publc void reduce(K2 key, Iterable<V2> values, Context
context) {
     ...
     context.write(k3, v3);
  }
}
```

워드카운트 소스코드

Sample Main

```
public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = new Job(conf, “MyJob”);

        // 각종 초기화작업
        // 맵클래스 지정
        // 리듀스클래스 지정
        // 입력파일 위치 지정
        // 출력파일 저장될 위치 지정

        job.waitForCompletion(true);
}
```

워드카운트 소스코드

변수 타입

- Writable Interface

  write (serialize될 경우 호출)
  readFields(deserialize될 경우 호출)

- WritableComparable Interface

  compareTo (순서를 지정할 경우-키 소팅용)

- 기본 제공 타입

  Text IntWritable
  LongWritable FloatWritable
  BooleanWritable ArrayWritable NullWritable

워드카운트 소스코드

입력포맷

입력포맷 클래스

TextInputFormat
- extends FileInputFormat
- text file, .gz extension
- ₩r₩n(라인) 하나가 하나의 레코드
- 키는 파일 Offset, 타입 LongWritable
- 밸류는 라인 한 줄 전체, 타입 Text

KeyValueTextInputFormat
- 키, 밸류 사이에 ₩t(탭)
- 키, 밸류 타입 Text

출력 포맷 TextOutputFormat

워드카운트 소스코드

컴바이너

- Mini Reducer or Local Reducer
- 태스크의 출력에 간이 리듀스를 적용해 리듀스가 처리해야 하는 데이터의 크기를 줄이는 역할
- 교환과 결합의 법칙이 만족하는 잡이면 적용 가능

- job.setCombinerClass(Reducer.class)
- 적용했는지 여부는 시스템 카운터의 값을 확인하면 됨

워드카운트 소스코드

맵 태스크 수의 결정방식

- getSplits 메소드
  InputSplit들의 리스트 리턴
- InputSplit마다 맵 태스크가 할당됨
- 입력 파일의 수
- 입력 파일의 크기
  데이터 블록(기본 64MB)마다 맵태스크 할당
  데이터 블럭이 InputSplit
- 입력 포맷의 변수
  압축파일의 경우 전체를 하나의 맵 태스크에 할당
  (SequenceFileInputFormat의 경우 예외)
  isSplitable

워드카운트 소스코드

Reduce Class

- run, setup, cleanup
- 리듀스 입력
  맵단의 출력에 따라 결정
- 리듀스 출력
  TextOutputFormat/SequenceFileOutputFormat
  Customization -> Job.setOutputFormatClass
  FileOutputFormat.setOutputPath
  출력타입

  setOutputKeyClass/setOutputValueClass
  리듀스 태스크의 수 : Job.setNumReduceTasks

워드카운트 소스코드

출력 포맷

- TextOutputFormat
  - 출력레코드 하나가 한 라인
  - 키/밸류는 TAB으로 분리
  - 결과파일 압축가능
    - TextOutputFormat.setCompressOutput(job.true);
    - TextOutputFormat.setOutputCompressorClass(job, GzipCodec.class)
    - cf. LZO codec, Snappy

워드카운트 소스코드

Counter

- context.getCounter(“Error”, “No numeric ID”).increment(1);
- Max 120개(CounterExceededException)
  mapreduce.job.counters.limit in mapred-site.xml 나
  job.getConfiguration().setInt(“mapreduce.job.counters.limit”, 200);
- Counter Group
  system counters
  “Job Counters”, “File Output Format Counters”, “FileSystemCounters”

워드카운트 소스코드

WordCount Build

- sudo apt-get install maven2
- wget https://s3.amazonaws.com/hadoopkr/source.tar.gz
- tar xvfz source.tar.gz
- cd chapter4/WordCount
- mvn install
- sjha@ubuntu-server:~/source/chapter4/WordCount$ ls
  README.txt pom.xml src target
  sjha@ubuntu-server:~/source/chapter4/WordCount/target$ ls target
  WordCount-1.0-SNAPSHOT.jar classes maven-archiver surefire surefire-reports test-classes
- hadoop jar WordCount-1.0-SNAPSHOT.jar WordCount /input/README.txt /output2
- sjha@ubuntu-server:~/hadoop-1.0.3$ hadoop fs -cat /output2/part-r-00000 | more

워드카운트 소스코드

Test Data

```
wget http://s3.amazonaws.com/hadoopkr/data.tar.gz
tar xvfz data.tar.gz
cd data
hadoop fs -mkdir /input/wex
hadoop fs -copyFromLocal * /input/wex
hadoop fs -ls /input/wex
```

실습동영상

LESSON 2

Tip : 한글처리는 어떻게 하나?

Tip : 한글처리는 어떻게 하나?

- 한글은 일반적인 영어와 다르게 적용
- '나는', '나를' 은 같은 단어임
  - '나'(어간) + '는'(조사) / '나'(어간) + '를'(조사)
- 한글 문장은 전처리가 필요하다
  - 형태소분석기가 필요
  - 오픈소스/상용 형태소 분석기
    - 은전한닢(http://eunjeon.blogspot.kr/) 등