맵리듀스 프로그램

●프로그램 요소 - 파일 시스템 클래스

●org.apache.hadoop.fs.FileSystem
● 하둡은 자바 어플리케이션에서 하둡의 파일 시스템을 접근할 수 있도록 라이브러리를 제공합니다.
● 로컬 파일 시스템이나 HDFS나 어떤 파일 시스템을 사용하든 반드시 FileSystem 클래스로 파일에 접근해야 합니다.

❖ 객체 생성
➢ Configuration conf = new Configuration();
➢ FileSystem hdfs = Filesystem.get(conf);
➢ Path path = new Path("파일 시스템 경로");

실행] hadoop jar filewrite.jar hadoop.test.SinglefileWriteRead

● 프로그래밍 - Mapper

●매퍼 클래스 작성
● Mapper 클래스를 상속받고 map() 메서드 정의
● map() 메서드 인자로 넘겨진 key와 value를 분석하여 context 객체를 통해 출력 (write) 합니다.

```
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);
     }
   }
 }
```

● 프로그래밍 - Reducer

●리듀서 클래스 작성
● Reduce 클래스를 상속받고 reduce() 메서드 정의
● 맵퍼의 결과가 셔플되고 각 키의 값은 Iterable 객체로 생성되어 리듀서의 입력으로 됩니다.
● reduce() 메서드 인자로 넘겨진 Iterable value를 카운트 하여 context 객체를 통해 출력합니다.

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

● 프로그래밍 - Driver

●드라이버 클래스 작성
● Job 객체 생성 (Job job = new Job(“WordCount”);
● 맵리듀스 실행정보 설정
● 맵리듀스 잡 실행

```
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> <out>");
     System.exit(2);
   }
   Job job = new Job(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);
   FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
   FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
   System.exit(job.waitForCompletion(true) ? 0 : 1);
 }
```

맵리듀스 실습 예제

●맵리듀스 실습 데이터 정보

- apat63_99.txt
Variable Name Variable type Characters Contents
patent numeric 7 Patent Number
gyear numeric 12 Grant Year
gdate numeric 12 Grant Date
appyear numeric 12 Application Year
country Character 3 Country of First Inventor
postate numeric 3 State of First Inventor (if US)
assignee numeric 12 Assignee Identifier (missing 1963-1967)
asscode numeric 12 Assignee Type
claims numeric 12 number of Claims
nclass numeric 12 Main Patent Class (3 digit)

- cite75_99.txt
Variable Name Variable type Characters Contents
citing numeric 7 Citing Patent Number
cited numeric 7 Cited Patent Number

●WordCount01.java (1/2)

```
package WordCount01;

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;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

public class WordCount01 {

 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();
```

●WordCount01.java (2/2)

```
    public void reduce(Text key, Iterable<IntWritable> values,
                       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> <out>");
      System.exit(2); 4

    }
    Job job = new Job(conf, "word count 01");
    job.setJarByClass(WordCount01.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
    FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

●WordCount02.java (1/2)

```
package WordCount02;

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

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;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

public class WordCount02 {

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

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

   public void map(Object key, Text value, Context context
                   ) throws IOException, InterruptedException {

       String[] citation = value.toString().split(",");
       context.write(new Text(citation[1]), 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 {
```

●WordCount02.java (2/2)

```
    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(); 6

  if (otherArgs.length != 2) {
    System.err.println("Usage: wordcount <in> <out>");
    System.exit(2);
  }
  Job job = new Job(conf, "word count 02");
  job.setJarByClass(WordCount02.class);
  job.setMapperClass(TokenizerMapper.class);
  job.setCombinerClass(IntSumReducer.class);
  job.setReducerClass(IntSumReducer.class);
  job.setOutputKeyClass(Text.class);
  job.setOutputValueClass(IntWritable.class);
  FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
  FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
  System.exit(job.waitForCompletion(true) ? 0 : 1);
 }
}
```

●MyJob01.java (1/2)

```
package WordCount02;

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

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;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

public class WordCount02 {

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

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

   public void map(Object key, Text value, Context context
                   ) throws IOException, InterruptedException {

       String[] citation = value.toString().split(",");
       context.write(new Text(citation[1]), 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 {
```

●MyJob01.java (2/2)

```
int count = 0;
        for (IntWritable val : values) {
            count += val.get();
        }

        context.write(key, new IntWritable(count));
    }
  }

  public int run(String[] args) throws Exception {
    Configuration conf = getConf();

    Job job = new Job(conf, "MyJob01");
    job.setJarByClass(MyJob01.class);

    Path in = new Path(args[0]);

    Path out = new Path(args[1]);
    FileInputFormat.setInputPaths(job, in);
    FileOutputFormat.setOutputPath(job, out);

    job.setMapperClass(MapClass.class);
    job.setReducerClass(Reduce.class);

    job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    System.exit(job.waitForCompletion(true)?0:1);

    return 0;
  }

  public static void main(String[] args) throws Exception {
    int res = ToolRunner.run(new Configuration(), new MyJob01(),
args);

    System.exit(res);
  }
}
```

●MyJob02.java (1/3)

```
package MyJob02;

import java.io.IOException;
import java.util.Iterator;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.IntWritable;
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;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class MyJob02 extends Configured implements Tool {
```

●MyJob02.java (2/3)

```
 public static class MapClass extends Mapper<LongWritable, Text, Text,
Text > {

       public void map(LongWritable key, Text value, Context context)
throws
IOException, InterruptedException {

          String[] citation = value.toString().split(",");
          String strKey = new Text(citation[4]) + "," + new
Text(citation[9]);
          context.write(new Text(strKey), new Text(citation[0]));
       }
   }

public static class Reduce extends Reducer<Text, Text, Text, Text> {

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

          String csv = "";

          for (Text val:values) {
             if (csv.length() > 0) csv += ",";
             csv += val.toString();
          }
          context.write(key, new Text(csv));
       }
   }
```

●MyJob02.java (3/3)

```
public int run(String[] args) throws Exception {
        Configuration conf = getConf();


        Job job = new Job(conf, "MyJob02");
        job.setJarByClass(MyJob02.class);

        Path in = new Path(args[0]);
        Path out = new Path(args[1]);
        FileInputFormat.setInputPaths(job, in);
        FileOutputFormat.setOutputPath(job, out);

        job.setMapperClass(MapClass.class);
        job.setReducerClass(Reduce.class);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);

        System.exit(job.waitForCompletion(true)?0:1);

        return 0;
    }

    public static void main(String[] args) throws Exception {
        int res = ToolRunner.run(new Configuration(), new MyJob02(),
args);

        System.exit(res);
    }
}
```

●프로그래밍 실행 방법

● 파일 HDFS에 업로드
- hadoop dfs -put cite.TXT /mapreduce/

● 실행
- hadoop jar WordCount01.jar WordCount01.WordCount01 /wordcount/cite.TXT /wordcount/r01/

● 실행 시 출력 로그
- 잡/리듀스 처리 과정이 로그로 출력 됨.
- 파일 출력 포멧 카운터
- 파일 시스템 카운터
- 파일 입력 포멧 카운터
- 맵리듀스 프레임웍 등

MEMO

❖

PIG 프로그램 실습 예제

● 소개 및 구조이해

➢Apache Pig는 Hadoop을 구성하는 하부 프로젝트 중 하나로써, high-level 언어로 구성된 대용량 데이터 셋 분석을 위한 데이터 분석 플랫폼이다. Pig는 대규모 병렬 처리에 대응 할수 있는 구조를 가지고 있으며, 이것은 대규모 데이터 셋을 다룰 수 있도록 하는 것이 두드러진 특징이다.
➢pig의 언어 계층은 Pig Latin이라는 텍스트 기반의 언어로 다음과 같은 특징이 있다

1. 데이터 흐름기반의 프로그래밍
2. 맵리듀스 변환 수행
3. 다양한 데이터 구조
4. 분산 실행 환경
5. 사용자 정의 함수
6. 오프라인 배치작업

MEMO

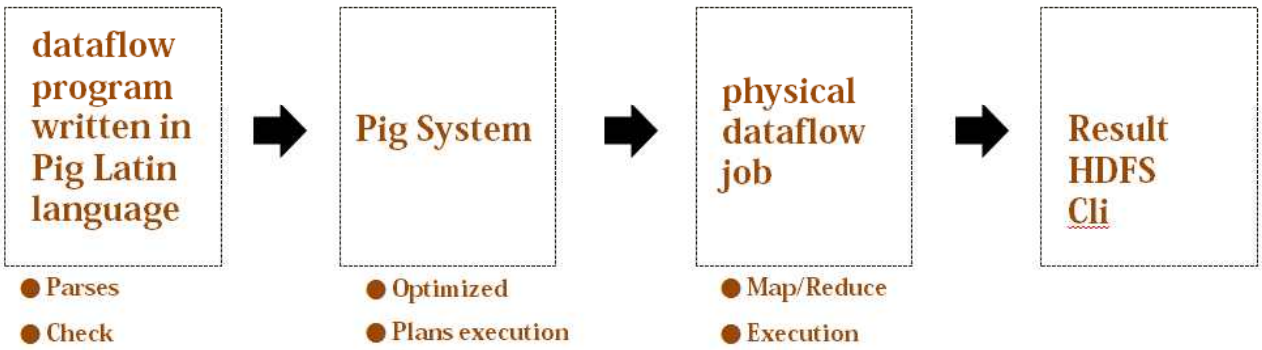
❖

●소개 및 구조이해

● 사용자 정의 함수 (User Defined functions)
➢저장, 로드 , 필터, 조인 과정을 사용자가 원하는 순서대로 변경이 가능하며, 사용자가 자신이 목적에 맞게 자신이 필요로 하는 함수를 구현할 수 있다.

● 쉬운 프로그래밍
➢복잡하게 구성된 데이터 분석 조작에 있어서도 데이터 흐름을 명시적으로 보여줄 수 있는 코드 작성과 관계형 데이터 처리스타일(filter, group, join , union) 등을 지원하여 사용하기 쉽다.

● 비용 최적화
➢시스템이 코드 실행을 자동으로 최적화 해주므로 사용자는 효율성을 생각하지 않고 프로그래밍 내용에만 집중 한다.



MEMO

● 클라이언트 (Pig Latin Language)
❖데이터 흐름 기반의 스크립트를 클라이언트 또는 Pig 파일로 작성하고 실행 시, 스크립트에 대한 구문분석 및
❖구문검사를 수행한다.

● Pig System
❖Pig는 자동으로 최적화를 위한 실행계획작성 및 최적화된 맵리듀스 파일을 생성한다.

● MapReduce Job
❖맵리듀스 프로그램을 자동 실행한다.

● Result
❖결과를 커맨드 인터페이스 또는 HDFS에 출력한다.

●Pig Data Model

➢Expression
➢t = (‘alice’ , { (‘and’, 1) (‘ipad’ , 2)},[‘age’ -> 20])

➢t는 f1 , f2 , f3라는 필드와 튜플로 구성 되어있다고 보자.

Expression Type	Example	Value for t
형태	Set	Independent of t
필드위치	$0	‘alice’
필드이름	f3	[‘age’->20]
백.튜플	f2.$0	{(and)(ipad)}
맵	f3#’age’	20
함수	SUM(f2.$1)	1 + 2 = 3
조건	f3#’age’>18 ? ‘a’:’b’	‘a’
flattening	flatten(f2)	‘and’,1 ₩n ‘ipad’,2

MEMO

❖

●활용 스크립트

➢ LOAD
: 파일 시스템에서 데이터 읽기

➢Expression
queries = load ‘query_log.txt’
USING myStorage(‘,’)
as (user , queryString, timestamp);

➢ STORE
파일 시스템에 데이터 쓰기
STORE 명령어는 피그가 실제 동작하도록 합니다.

➢Expression
STORE queries INTO ‘/output’
USING myStorage(‘,’);

➢ FOREACH … GENERATE
데이터 세트의 하나 이상의 레코드를 기록 또는 변경

➢Expression
forqueries = FOREACH queries GENERATE
user , queryString;

MEMO

❖

●활용 스크립트

➢ ▶ JOIN
연관된 키에 따라 둘 이상의 데이터 셋을 연결

➢Expression
joinqueries = JOIN queries1 BY queryString , queries2 BY queryString;

➢ ▶ GROUP
하나 이상의 레코드에서 동일한 키의 값을 가지는 레코드들의 집합

➢Expression
groupqueries = GROUP queries BY queryString;

➢ ▶ COGROUP
둘 이상의 데이터 셋에서 동일한 키의 값을 가지는 레코드들의 집합

➢Expression
cogroupqueries = COGROUP queries1 BY queryString , queries2 BY queryString;

MEMO

❖

●활용 스크립트

➢ ▶ DISTINCT
중복되는 레코드를 제거하여 기록

➢Expression
distinctqueries = DISTINCT queries;

➢ ▶ LIMIT
레코드의 수를 제한하여 기록

➢Expression
limitqueries = LIMIT queries 100;

➢ ▶ SPLIT
Filter 조건에 따라 2 개 이상의 데이터로 분할하여 기록

➢Expression
SPLIT queries INTO splitqueries1 IF queryString > 10 , splitqueries2 IF queryString <= 10;

➢ ▶ UNION
둘 이상의 데이터 셋을 하나의 데이터 셋으로 병합하여 기록

➢Expression
unionqueries = UNION queries1 , queries2;

●피그 명령어

명령어	설명
cat	한 개 이상의 파일 내용을 출력한다.
cd	현재의 경로를 변경한다.
copyFromLocal	로컬 파일을 하둡으로 복사한다.
copyToLocal	하둡 상의 파일을 로컬로 복사한다.
cp	파일 또는 디렉토리를 다른 경로에 복사한다.
ls	파일을 조회한다.
mkdir	새로운 디렉토리를 생성한다.
mv	파일 또는 디렉토리를 다른 경로로 이동시킨다.
pwd	현재의 실행 디렉토리의 경로를 출력한다.
rm	파일 또는 디렉토리를 삭제한다.
kill	맵리듀스 작업을 제거한다.
exec	백그라운드로 스크립트를 실행
help	사용 가능한 명령어와 옵션을 보여준다.
run	그런트 쉘에 있는 스크립트를 실행한다.
quit	명령처리기를 종료한다.
set	피그의 옵션을 설정한다.

MEMO

❖

●WordCount01

```
데이터 로드
c = load '/user/bdh/data27/cite.TXT' using
PigStorage(',') as (c1:int,
c2:int);

/** 카운트를 위한 컬럼 그룹화  */
g = group c by c1;

describe g;

/** group 데이터 수를 카운트  */
f = foreach g generate group, COUNT($1) as cnt;

/** 결과를 확인해 볼 수 있다. */
L = limit f 10;

Dump L;
```

MEMO

❖

●WordCount02

```
데이터 로드
c = load '/user/bdh/data27/cite.TXT' using
PigStorage(',') as (c1:int,
c2:int);

/** 카운트를 위한 컬럼 그룹화  */
g = group c by (c1, c2);

/** group 데이터 수를 카운트  */
f = foreach g generate flatten(group), COUNT($1) as
cnt;

/** 결과를 확인해 볼 수 있다. */
L = limit f 10;
Dump L;
```

MEMO

❖

●MY JOB 01

데이터 로드

```
a = load '/user/bdh/data27/apat.TXT' USING
PigStorage(',') as
(patent:int, gyear:int,  gdate:int, appyear:int,
country:chararray,
postate:int, assignee:int, asscode:int, claims:int,
nclass:int);


/** 그룹화*/
grunt> g = group a by (country, patent);
grunt> describe g;
g: {group: (country: chararray,patent: int),a: {(patent:
int,gyear:
int,gdate: int,appyear: int,country: chararray,postate:
int,assignee:
int,asscode: int,claims: int,nclass: int)}}

/** 카운트  */
f = foreach g generate flatten(group), COUNT($1) as
cnt;

/** 파일로 보기  */
STORE f INTO '/user/bdh/data27/f_tmp';
```

MEMO

❖

●MY JOB 02

데이터불러오기
b = load '/user/bdh/data27/f_tmp' USING
PigStorage('₩t') as
(country:chararray, patent:int, cnt:int);

/** 그룹화 */
g = group b by country;

/** 특허수 합 */
f = foreach g generate group , SUM(b.cnt) as sum;

/** 특허수로 정렬 */
o = order f by sum desc;

결과
l = limit o 100;
dump l;

MEMO

❖

●UDF 실습 예제(1)

```
import java.io.IOException;
import org.apache.pig.EvalFunc;
import org.apache.pig.data.Tuple;
import org.apache.pig.impl.util.WrappedIOException;
public class UPPER extends EvalFunc<String>{
  public String exec(Tuple input) throws IOException {
    if (input == null || input.size() == 0) return null;
    try{
      String str = (String)input.get(0);
        return str.toUpperCase();
    }catch(Exception e){
      throw WrappedIOException.wrap("Caught
exception processing input row ", e);
    }
  }
}
```

MEMO

❖

●UDF 실습 예제(2)

```
REGISTER /home/root/jar/PIG_UDF.jar;

DEFINE Up PIG_UDF();

b = load '/data/word.TXT' as (word:chararray);

c = foreach b generate Up(word);

dump c;
```

MEMO

❖