

맵리듀스 응용 구축

순천향대학교 컴퓨터공학과
이 상 정

맵리듀스 응용 구축

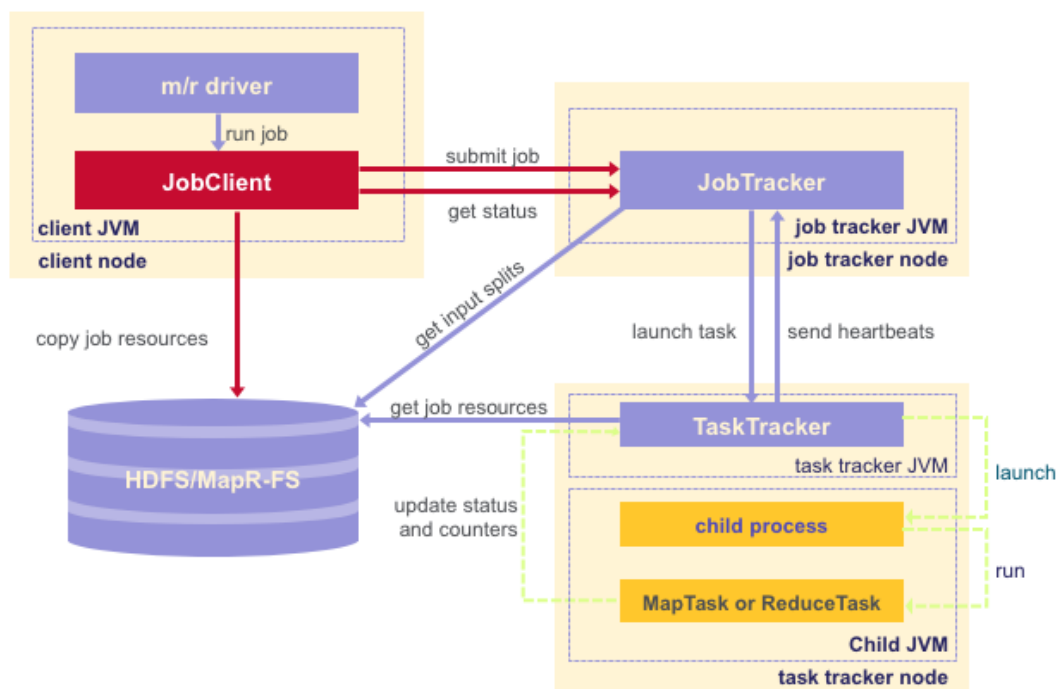
학습 내용

1. 작업 실행 프레임워크 (Job Execution Framework)
2. 맵리듀스 프로그램 작성
3. Receipts.jar 코드의 빌드와 실행
4. 맵리듀스 코드 실행 분석
5. WordCount 예

1. 작업 실행 프레임워크 (Job Execution Framework)

맵리듀스 응용 구축

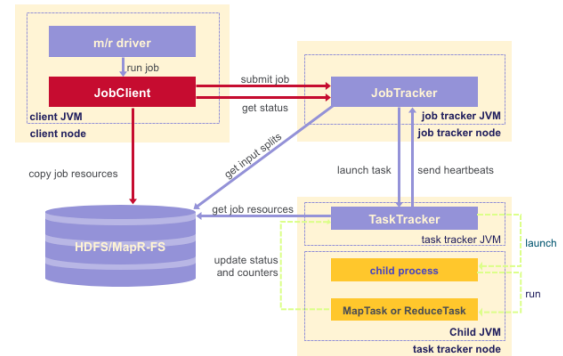
맵리듀스 작업 실행 프레임워크 (1)



맵리듀스 작업 실행 프레임워크 (2)

□ 하둡의 맵리듀스 작업 실행 과정

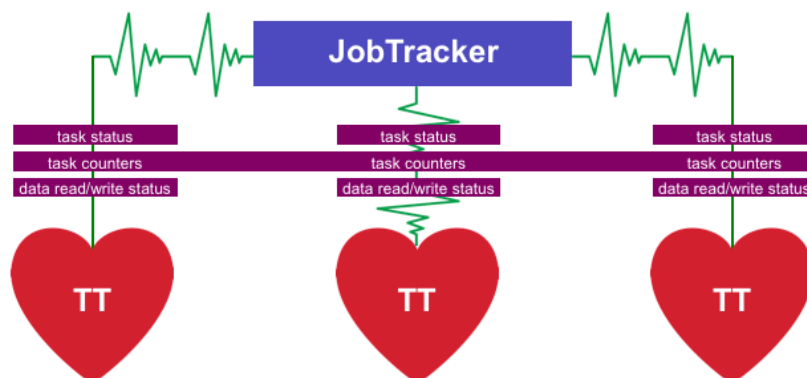
- 사용자가 **클라이언트 노드**에서 **JobClient** 객체를 생성하여 맵리듀스 프로그램 실행을 시작
- JobClient 객체가 **JobTracker**에 작업(job)을 제출
- JobTracker는 **Job** 객체를 생성하고, 적당한 **TaskTracker** 노드에 전송
- TaskTracker는 맵과 리듀스 작업을 차례로 수행하는 **자식 프로세스**를 생성하고 시작
- 작업 수행 중에 **TaskTracker**의 상태와 카운터 등이 계속 갱신



Heartbeat 적용

□ TaskTracker는 JobTracker에 주기적으로 heartbeats을 전송하여 살아 있음을 알림

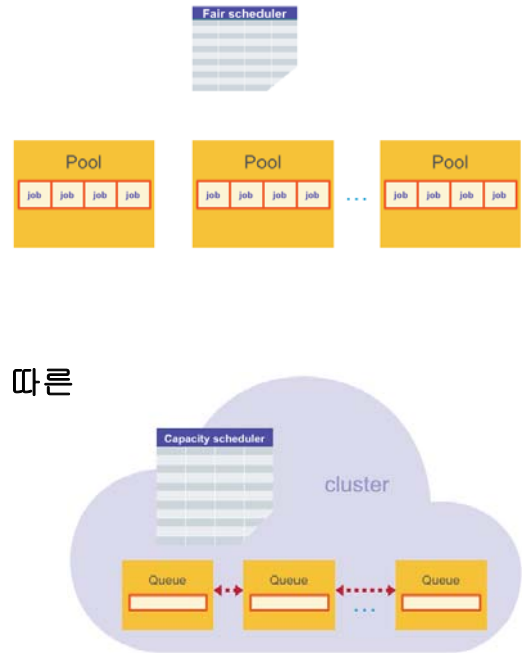
- heartbeats에는 **태스크의 상태**, **태스크 카운터**, **데이터의 읽기/쓰기 상태** 등이 포함
- 한 태스크 트랙커로 부터 heartbeats의 수신이 안되면 JobTracker는 다른 TaskTracker 노드에 중단된 작업을 재 스케줄
 - 중단된 노드는 다운된 것으로 표시하고 이 후 작업 스케줄 대상에서 제외



하둡 작업 스케줄링 (Job Scheduling)

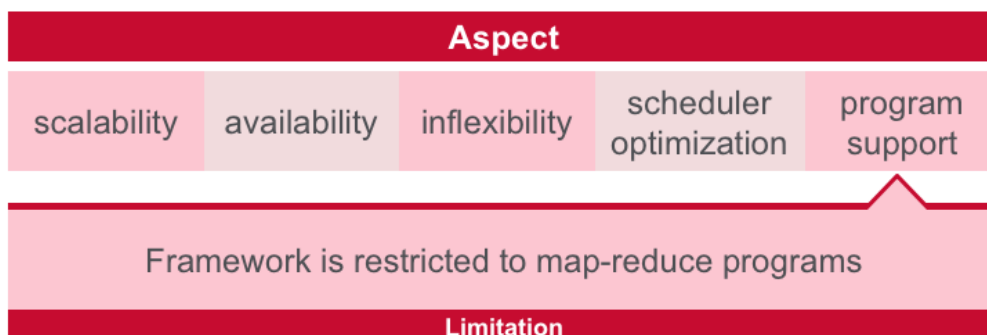
□ 하둡에는 2가지 스케줄러

- **페어 스케줄러 (Fair scheduler)**
 - 각 풀(pool)에 자원이 공평하게 분배
 - 각 사용자는 디폴트로 자신의 풀을 가짐
 - 선점(pre-emption)을 지원
 - 페이스북이 개발
- **커패시티 스케줄러 (Capacity scheduler)**
 - 관리자가 계층적 풀(큐)을 구성하여 조직에 따른 차별적인 접근을 반영
 - 각 큐 별로 이용할 수 있는 자원의 용량을 정하여 주면 그 용량에 맞게 자원을 할당
 - 자원-기반 스케줄링과 작업 우선순위 지원
 - 야후가 개발



하둡 실행 프레임워크의 제약

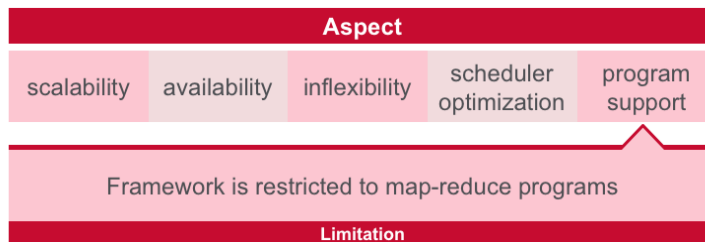
- 맵리듀스 v1 프레임워크는 여러 제약점이 있어서, 맵리듀스 v2 또는 **YARN(Yet Another Resource Negotiator)** 이 개발



YARN의 개발 배경

□ 기존의 맵리듀스 v1의 제약점

- 노드 당 실행하는 맵과 리듀스 슬롯의 설정이 고정 (inflexibility)
- 맵리듀스 작업만 지원하고, 맵리듀스가 아닌 응용은 지원되지 않음 (program support)
- 단일 JobTracker의 확장성에 제약이 있어 클러스터 당 최대 4000개 노드만 지원 (scalability)

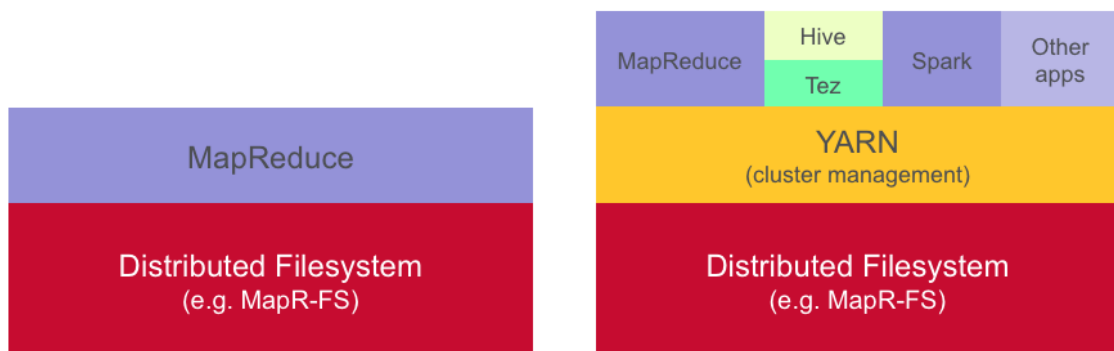


□ YARN은 위의 제약점을 해결하고, 맵리듀스 v1과 같은 API와 CLI 사용

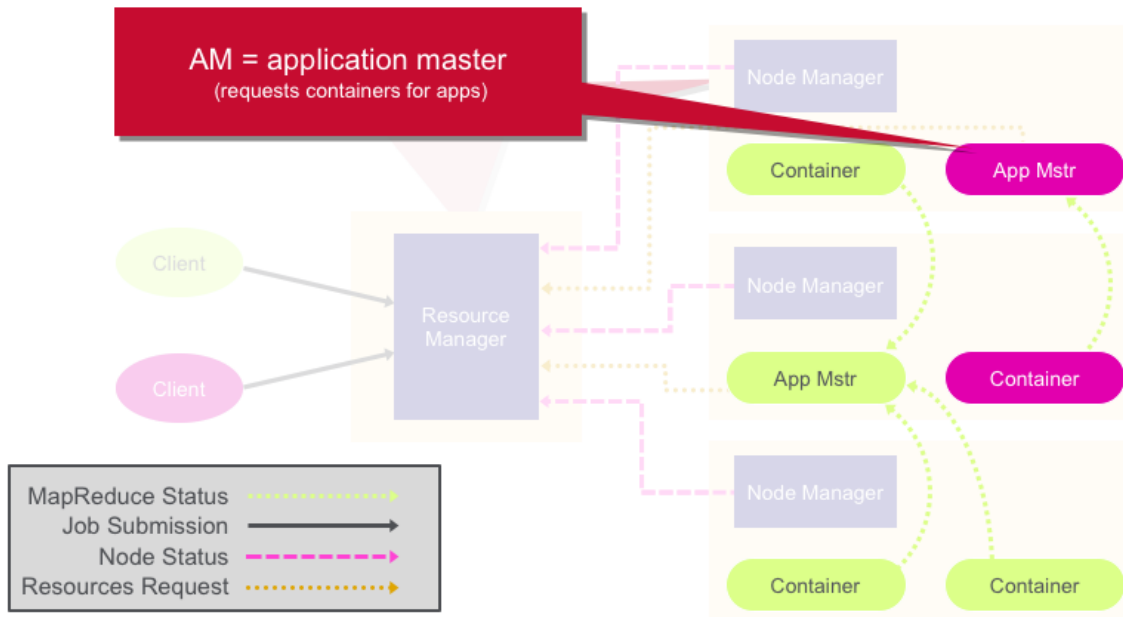
맵리듀스 v1과 YARN의 비교

□ YARN은 자원의 관리(resource management)와 작업의 관리(job management)를 분리

- 클러스터 자원의 관리와 작업 스케줄링(job scheduling)은 ResourceManager가 관리
- 자원의 협상(resource negotiation)과 작업 모니터링(job monitoring)은 클러스터에서 수행되는 각 응용의 ApplicationMaster가 관리



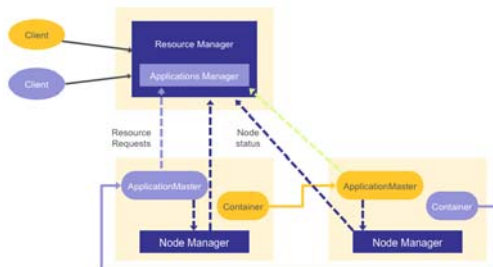
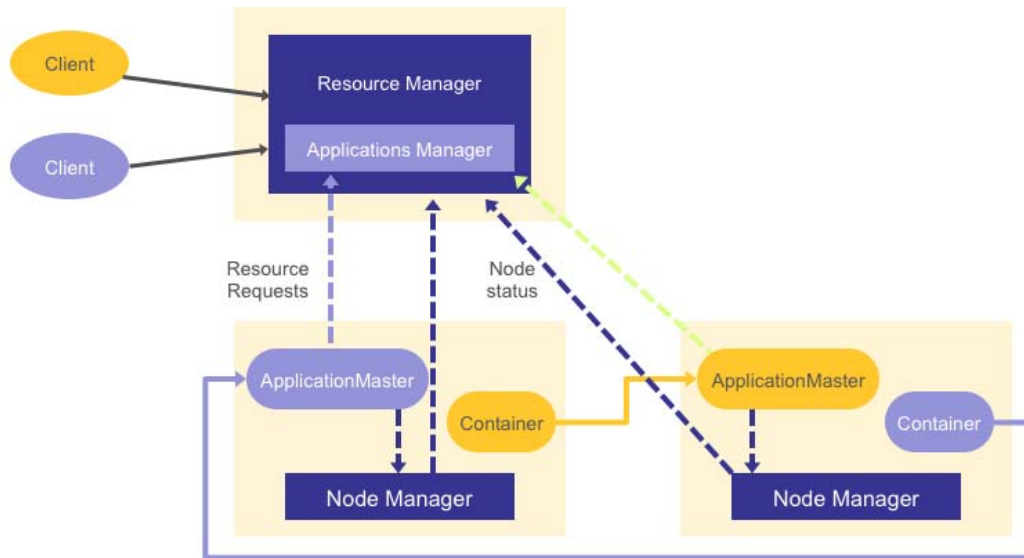
YARN 아키텍처 (1)



YARN 아키텍처 (2)

- 컨테이너 (Container)
 - 할당된 컴퓨팅 자원의 논리적인 개체 (CPU, 메모리)
- RM = Resource Manager, 자원 관리자
 - 클러스터 내의 컴퓨팅 자원 을 할당
 - 컨테이너를 생성/삭제, NM을 추적
- NM = Node Manager, 노드 관리자
 - 응용과 AM을 실행을 시작하고 상태를 보고
- AM = Application Master, 응용 마스터
 - 응용 프로그램을 위해 컨테이너를 요청하고 실행

YARN의 작업 실행 (1)




YARN의 작업 실행 (2)

- 클라이언트는 YARN의 자원 관리자(Resource Manager)에게 응용을 제출 (이때 컨테이너를 생성하기 위한 정보인 CLC(Container Launch Context)도 함께 전달)
- 자원 관리자 내부에 있는 응용 관리자(Application Manager)가 한 컨테이너(container)를 위해 협상하고, 해당 응용의 응용 마스터(Application Master) 인스턴스를 생성
- 응용 마스터는 자원 관리자에 등록되고, 노드 관리자(Node Manager)에 각 컨테이너의 CLC를 전달하고 컨테이너를 생성
- 응용 마스터는 응용의 실행을 관리하고, 실행 상황과 상태 정보 등을 모니터링
 - 클라이언트는 자원 관리자에게 질의하거나, 직접 응용 마스터와 통신하여 응용의 상태를 모니터링
- 응용 마스터는 응용의 실행이 완료되면 자원 관리자에게 알림
- 마지막으로 응용 마스터는 자원 관리자에서 등록을 해제하고, 자원 관리자는 컨테이너를 해제

YARN의 작업 모니터링 - 작업 이력 서버 (1)

□ 작업 이력 서버 (Job History Server)

- 웹 UI 상에서 실행된 작업의 요약
- 작업을 클릭하면 상세한 이력 표시



JobHistory

Logged in as: dr.who

Application: Retired Jobs

Tools: About Jobs


Show 20 entries

| Submit Time | Start Time | Finish Time | Job ID | Name | User | Queue | State | Maps Total | Maps Completed | Reduces Total | Reduces Completed |
|-------------------------|-------------------------|-------------------------|------------------------|---------|--------|---------|-----------|------------|----------------|---------------|-------------------|
| 2014.07.31 05:24:14 PDT | 2014.07.31 05:24:23 PDT | 2014.07.31 05:24:35 PDT | job_1406809109141_0002 | TeraGen | user01 | default | SUCCEEDED | 2 | 2 | 0 | 0 |
| 2014.07.31 05:23:47 PDT | 2014.07.31 05:23:59 PDT | 2014.07.31 05:23:59 PDT | job_1406809109141_0001 | TeraGen | user01 | default | FAILED | 0 | 0 | 0 | 0 |

Showing 1 to 2 of 2 entries

First Previous 1 Next Last

YARN의 작업 모니터링 - 작업 이력 서버 (2)



MapReduce Job

job_1406809109141_0002

Logged in as: dr.who

Application: Job Overview

Tools: Overview Counters Configuration Map tasks Reduce tasks

Job Name: TeraGen
User Name: user01
Queue: default
State: SUCCEEDED
Uberized: false
Submitted: Thu Jul 31 05:24:14 PDT 2014
Started: Thu Jul 31 05:24:23 PDT 2014
Finished: Thu Jul 31 05:24:35 PDT 2014
Elapsed: 11sec
Diagnostics:
Average Map Time: 8sec

| ApplicationMaster | Attempt Number | Start Time | Node | Logs |
|-------------------|----------------|------------------------------|---------------|------|
| 1 | | Thu Jul 31 05:24:19 PDT 2014 | mapr1node8042 | logs |

| Task Type | Total | Complete | |
|--------------|--------|----------|------------|
| Map | 2 | 2 | |
| Reduce | 0 | 0 | |
| Attempt Type | Failed | Killed | Successful |
| Maps | 0 | 0 | 2 |
| Reduces | 0 | 0 | 0 |

2. 맵리듀스 프로그램 작성

맵리듀스 응용 구축

맵리듀스 프로그래밍 팁

- ❑ 드라이버(driver), 매퍼(mapper), 리듀서(reducer) 클래스들의 템플릿(template)을 가지고 시작
 - 응용 로직 이외의 맵리듀스 코드의 대부분은 공통적으로 작성
 - 공통된 부분을 템플릿으로 작성
 - import 문들, 클래스 정의들, 메서드 시그니처(method signature)
- ❑ 응용의 로직 부분을 추가 수정
- ❑ 맵리듀스 프레임워크 내에서 데이터의 흐름과 변환 과정의 이해가 필수
- ❑ 키/값들 쌍에 대한 적절한 자료형(type)의 적용

맵리듀스 데이터 흐름과 변환 과정



□ 맵리듀스 시작에서 종료까지 4단계 변환 과정 이해가 필수

- 입력 파일들로부터 변환되어 매퍼에 전달되는 방법
- 매퍼에서 데이터가 변환되는 방법
- 데이터가 정렬되고 통합되어 리듀서에게 전달되는 방법
- 리듀서가 데이터를 변환하여 출력 파일들에 내보내는 방법

예제 데이터 세트

□ 각 라인 당 10개의 정보 필드들이 있는 데이터의 예

- delta 열의 최소값을 갖는 년도를 찾는 프로그램 예 소개
- 나머지 필드는 무시

| | | | | | | | | | |
|------|---------|---------|----------|---------|---------|----------|--------|--------|--------|
| 2000 | 2025191 | 1788950 | 236241 | 1544607 | 1458185 | 86422 | 480584 | 330765 | 149819 |
| 2001 | 1991082 | 1862846 | 128236 | 1483563 | 1516008 | -32445 | 507519 | 346838 | 160681 |
| 2002 | 1853136 | 2010894 | -157758 | 1337815 | 1655232 | -317417 | 515321 | 355662 | 159659 |
| 2003 | 1782314 | 2159899 | -377585 | 1258472 | 1796890 | -538418 | 523842 | 363009 | 160833 |
| 2004 | 1880114 | 2292841 | -412727 | 1345369 | 1913330 | -567961 | 534745 | 379511 | 155234 |
| 2005 | 2153611 | 2471957 | -318346 | 1576135 | 2069746 | -493611 | 577476 | 402211 | 175265 |
| 2006 | 2406869 | 2655050 | -248181 | 1798487 | 2232981 | -434494 | 608382 | 422069 | 186313 |
| 2007 | 2567985 | 2728686 | -160701 | 1932896 | 2275049 | -342153 | 635089 | 453637 | 181452 |
| 2008 | 2523991 | 2982544 | -458553 | 1865945 | 2507793 | -641848 | 658046 | 474751 | 183295 |
| 2009 | 2104989 | 3517677 | -1412688 | 1450980 | 3000661 | -1549681 | 654009 | 517016 | 136993 |
| 2010 | 2162706 | 3457079 | -1294373 | 1531019 | 2902397 | -1371378 | 631687 | 554682 | 77005 |
| 2011 | 2303466 | 3603059 | -1299593 | 1737678 | 3104453 | -1366775 | 565788 | 498606 | 67182 |
| 2012 | 2450164 | 3537127 | -1086963 | 1880663 | 3029539 | -1148876 | 569501 | 507588 | 61913 |

← year ← delta

Mapper 클래스 – 입력 (1)

□ Mapper 클래스의 입력으로 `TextInputFormat` 클래스 사용

- 디폴트 레코드 리더(record reader)의 키로 파일에서의 바이트 오프셋 (`LongWritable`)
 - `Writable`은 키와 값들을 스트림(네트워크 또는 저장장치)에 쓰기 전에 직렬화(serialization)하는 하둡의 인터페이스
 - 자바의 각 기본 타입마다 `Writable` 클래스를 가짐
 - `IntWritable`, `LongWritable`,
 - `org.apache.hadoop.io` 패키지에 타입 클래스 정의
- 디폴트 레코드 리더의 값으로 입력 파일에 읽어 들인 라인 (`Text`)

```
Input format = TextInputFormat
Key = LongWritable
Value = Text
```

} More detail later

Mapper 클래스 – 입력 (2)

- 첫 번째 레코드 예

First record:

input key input value

0 1901 588 525 63 588 525 63

- `StringTokenizer` 를 사용하여 `Text`의 문자열들의 각 필드를 분리 (공백문자로 각 필드를 구분)

```
StringTokenizer itr = new StringTokenizer(value.toString(), "\\s+");
```

Mapper 클래스 - 출력

- 한 레코드에서 1,4 번째 필드를 추출하여 키로 상수 "summary", 값으로 필드 1(년도)과 필드 4(delta)를 연결한 텍스트를 출력
 - 키 값으로 상수를 사용하여 하나의 파티션만 존재하고, 따라서 리듀서도 하나



```
context.write(new Text("summary"), new Text(year + "_" + delta));
```

First few lines of output

```
summary 1901_63
summary 1902_77
summary 1903_45
summary 1904_-43
summary 1905_-23
summary 1906_25
```

23

Mapper 클래스 - 구현

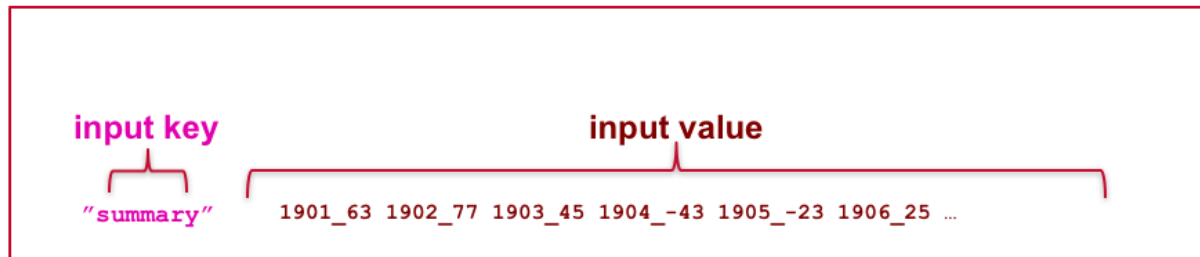
```
public class ReceiptsMapper extends Mapper
<LongWritable,Text,Text,Text> {
    public void map(LongWritable key, Text value, Context context)
    throws IOException, InterruptedException {
        StringTokenizer iterator = new
        StringTokenizer(value.toString(), "\\s+");
        String year = iterator.nextToken();
        iterator.nextToken();
        iterator.nextToken();
        String delta = iterator.nextToken();
        context.write(new Text("summary"), new Text(year + "_" +
        delta));
    }
}
```

- Mapper 클래스 인수: 입력 키 타입, 입력 값 타입, 출력 키 타입, 출력 값 타입
- map() 메서드 인수: 입력 키 타입, 입력 값 타입, 실행되는 작업의 컨텍스트
 - 컨텍스트: configuration, record reader, record writer, status reporter, input split, output committer
- 필드 구분을 위해 StringTokenizer 사용
- 키("summary")와 합성값(year_delta) 출력

Reducer 클래스 - 입력

□ 매퍼의 출력 타입과 리듀서의 입력 타입이 일치

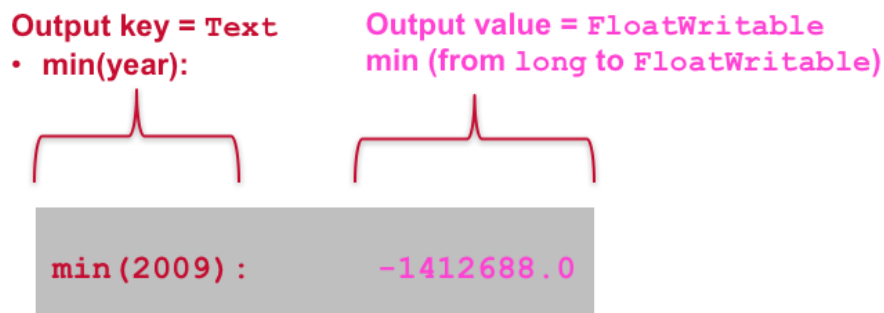
- Mapper output key = Text → Reducer input key = Text
- Mapper output value = Text → Reducer input values = Text



Reducer 클래스 - 출력

□ 리듀서의 출력

- 키: Text 타입의 min(year)
- 값: FloatWritable 타입의 delta



Reducer 클래스 - 구현 (1)

```
public class ReceiptsReducer extends Reducer
<Text,Text,Text,FloatWritable> {

    public void reduce(Text key, Iterable<Text> values, Context context)
    throws IOException, InterruptedException {
        long tempValue = 0L, min=Long.MAX_VALUE;
        Text tempYear=null, tempValue=null, minYear=null, maxYear=null;
        String compositeString;
        String[] compositeStringArray;
```

- ❑ Reducer 클래스 인수: 입력 키 타입, 입력 값 타입, 출력 키 타입, 출력 키 타입
- ❑ reduce() 메서드 인수: 입력 키 타입, 입력 값 타입, 실행되는 작업의 컨텍스트
 - 컨텍스트: configuration, record reader, output committer

Reducer 클래스 - 구현 (2)

```
for (Text value: values) {
    compositeString = value.toString();
    compositeStringArray = compositeString.split("_");
    tempYear = new Text(compositeStringArray[0]);
    tempValue = new Long(compositeStringArray[1]).longValue();
    if(tempValue < min) {
        min=tempValue;
        minYear=tempYear;
    }
}

Text keyText = new Text("min" + "(" + minYear.toString() + "): ");
context.write(keyText, new FloatWritable(min));
}
```

- ❑ 합성 값들에서 년도와 delta를 분리
- ❑ delta 값을 Long으로 형 변환 후 최소값 결정
- ❑ Text 타입의 min(year), FloatWritable 타입의 delta로 출력

Driver 클래스 - 구현 (1)

```
public class ReceiptsDriver extends Configured implements Tool {
    public int run(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.printf("usage: %s [generic options] <inputfile> <outputdir>\n", getClass().getSimpleName());
            System.exit(1);
        }
        Job job = new Job(getConf(), "my receipts");
        job.setJarByClass(ReceiptsDriver.class);
        job.setMapperClass(ReceiptsMapper.class);
        job.setReducerClass(ReceiptsReducer.class);
    }
}
```

- ❑ 명령행 인수 개수 확인: 입력 파일, 출력 디렉토리
- ❑ Job 객체 생성
- ❑ 드라이버, 매퍼, 리듀서 클래스 설정

Driver 클래스 - 구현 (2)

```
job.setInputFormatClass(TextInputFormat.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(FloatWritable.class);
job.setMapOutputValueClass(Text.class);
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
return job.waitForCompletion(true) ? 0 : 1;
}

public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    System.exit(ToolRunner.run(conf, new ReceiptsDriver(), args));
}
}
```

- ❑ 작업의 출력 키와 값의 타입을 Text와 FloatWritable로 정의
- ❑ 매퍼와 리듀서가 같은 출력 값의 타입을 사용하지 않으므로 매퍼의 출력값의 타입을 기술
 - 매퍼의 출력 값 타입은 Text, 리듀서의 출력 값 타입을 FloatWritable

Driver 클래스 - 구현 (3)

- ❑ 작업(job)은 동기식, 비동기식 2가지 방법으로 실행
- ❑ `job.waitForCompletion()` 은 작업을 동기식으로 실행
 - 작업이 완료될 때까지 드라이버 코드는 기다림(blocking)
 - 인수 `true` 는 터미널에 작업 처리 과정을 출력 (verbose output)
- ❑ `main()` 메서드는 드라이버의 엔트리 포인트
 - 작업의 configuration 객체를 생성
 - ToolRunner의 정적 메서드 `run()` 을 실행

3. Receipts.jar 코드의 빌드와 실행

예제 소스 다운로드

❑ 데이터 다운로드

- 강의 사이트에서 다운로드
 - \$ `wget http://cs.sch.ac.kr/lecture/BigData/download/receipts.txt`
- 또는, MapR 사이트에서 예제 소스 및 입력 데이터 다운로드
 - \$ `wget http://course-files.mapr.com/DEV3000/DEV300-v5.1-Lab3.zip`
 - \$ `unzip DEV300-v5.1-Lab3.zip`

❑ Receipts 예제 환경

- 소스 디렉토리: `~/MapReduce/Receipts`
- 로컬 데이터 파일: `~/MapReduce/input`

```
package Receipts;
```

```
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.LongWritable;
import java.io.IOException;
import java.util.StringTokenizer;
```

```
public class ReceiptsMapper extends Mapper <LongWritable,Text,Text,Text> {
    private final Text tempText = new Text();
    public void map(LongWritable key, Text value, Context context) throws IOException,
        InterruptedException {

        // create iterator over record assuming space-separated fields
        StringTokenizer iterator = new StringTokenizer(value.toString(), " ");

        // pull out year from record
        String year = new String(iterator.nextToken()).toString();

        // pull out 3rd field from record
        iterator.nextToken();
        iterator.nextToken();
        tempText.set(iterator.nextToken());
        context.write(new Text("summary"), new Text(year + "_" + tempText.toString()));
    }
}
```

ReceiptsMapper.java

ReceiptsReducer.java (1)

```
package Receipts;

import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.FloatWritable;
import java.io.IOException;

public class ReceiptsReducer extends Reducer<Text,Text,Text,FloatWritable> {
    private static Text tempYear=new Text();
    private static Text keyText=new Text();
    private static Text minYear=new Text();
    private static Text maxYear=new Text();
    private static String tempString;
    private static String[] keyString;

    public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
        InterruptedException {
        long tempValue = 0L;
        long max=Long.MIN_VALUE;
        long min=Long.MAX_VALUE;

```

ReceiptsReducer.java (2)

```
for (Text value: values) {
    tempString = value.toString();
    keyString = tempString.split("_");
    tempYear = new Text(keyString[0]);
    tempValue = new Long(keyString[1]).longValue();
    if(tempValue < min) {
        min=tempValue;
        minYear=tempYear;
    }
    if(tempValue > max) {
        max=tempValue;
        maxYear = tempYear;
    }
}

keyText.set("min" + "(" + minYear.toString() + "): ");
context.write(keyText, new FloatWritable(min));
keyText.set("max" + "(" + maxYear.toString() + "): ");
context.write(keyText, new FloatWritable(max));
}
}
```

ReceiptsDriver.java (1)

```
package Receipts;

import java.util.*;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;

public class ReceiptsDriver extends Configured implements Tool {

    public int run(String[] args) throws Exception {
        // check the CLI
        if (args.length != 2) {
            System.err.printf("usage: %s [generic options] <inputfile> <outputdir>\n",
                getClass().getSimpleName());
            System.exit(1);
        }
    }
}
```

ReceiptsDriver.java (2)

```
    Job job = new Job(getConf(), "my receipts");

    job.setJarByClass(ReceiptsDriver.class);
    job.setMapperClass(ReceiptsMapper.class);
    job.setReducerClass(ReceiptsReducer.class);

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

    // setup input and output paths
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    // launch job synchronously
    return job.waitForCompletion(true) ? 0 : 1;
}

public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    System.exit(ToolRunner.run(conf, new ReceiptsDriver(), args));
}
}
```

Receipts.jar 파일 빌드 과정

- ❑ 매퍼와 리듀서 자바 컴파일하여 클래스 파일 생성 후 Receipts.jar 파일 생성
 - 클래스파일들은 classes 디렉토리에 저장
- ❑ 위에서 생성된 하둡 클래스와 매퍼/리듀서 클래스의 Receipts.jar 파일을 참조하여 드라이버 컴파일한 후 Receipts.jar 파일에 추가

```
javac -d classes ReceiptsMapper.java
javac -d classes ReceiptsReducer.java
jar -cvf Receipts.jar -C classes/ .
javac -classpath $CLASSPATH:Receipts.jar -d classes ReceiptsDriver.java
jar -uvf Receipts.jar -C classes/ .
```

build.sh

```
#!/bin/bash

export HADOOP_HOME=/home/bigdata/hadoop-2.7.7
#export LD_LIBRARY_PATH=$HADOOP_HOME/lib/native/
export
CLASSPATH=$HADOOP_HOME/share/hadoop/mapreduce/*:$HADOOP_HOME/share/hadoop/mapreduce/lib/*:$HADOOP_HOME/share/hadoop/common/lib/*:$HADOOP_HOME/share/hadoop/common/*:$HADOOP_HOME/share/hadoop/tools/lib/*:$HADOOP_HOME/share/hadoop/yarn/*:$HADOOP_HOME/share/hadoop/yarn/lib/*
export HADOOP_CLASSPATH=$CLASSPATH

javac -d classes ReceiptsMapper.java
javac -d classes ReceiptsReducer.java
jar -cvf Receipts.jar -C classes/ .
javac -classpath $CLASSPATH:Receipts.jar -d classes ReceiptsDriver.java
jar -uvf Receipts.jar -C classes/ .
```

Receipts.jar 파일 빌드

```
bigdata@slave1:~/MapReduce/Receipts$ ls
build.sh input ReceiptsDriver.java ReceiptsMapper.java ReceiptsReducer.java
bigdata@slave1:~/MapReduce/Receipts$ mkdir classes
bigdata@slave1:~/MapReduce/Receipts$ ls
build.sh classes input ReceiptsDriver.java ReceiptsMapper.java ReceiptsReducer.java
bigdata@slave1:~/MapReduce/Receipts$
bigdata@slave1:~/MapReduce/Receipts$ sh build.sh
added manifest
adding: Receipts/(in = 0) (out= 0)(stored 0%)
adding: Receipts/ReceiptsReducer.class(in = 2587) (out= 1175)(deflated 54%)
adding: Receipts/ReceiptsMapper.class(in = 1839) (out= 765)(deflated 58%)
Note: ReceiptsDriver.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
adding: Receipts/(in = 0) (out= 0)(stored 0%)
adding: Receipts/ReceiptsReducer.class(in = 2587) (out= 1175)(deflated 54%)
adding: Receipts/ReceiptsMapper.class(in = 1839) (out= 765)(deflated 58%)
adding: Receipts/ReceiptsDriver.class(in = 2155) (out= 1063)(deflated 50%)
bigdata@slave1:~/MapReduce/Receipts$
bigdata@slave1:~/MapReduce/Receipts$ ls
build.sh classes input ReceiptsDriver.java Receipts.jar ReceiptsMapper.java ReceiptsReducer.java
bigdata@slave1:~/MapReduce/Receipts$
bigdata@slave1:~/MapReduce/Receipts$ ls classes
Receipts
bigdata@slave1:~/MapReduce/Receipts$ ls classes/Receipts
ReceiptsDriver.class ReceiptsMapper.class ReceiptsReducer.class
bigdata@slave1:~/MapReduce/Receipts$
```

입력 데이터 파일 복사

- 입력 데이터 로컬 파일 **receipts.txt**를 하둡 파일 시스템의 파일로 복사

- 로컬 파일: ~/MapReduce/Receipts/input/receipts.txt
- 하둡 파일: /Receipts/input/receipts.txt

\$ hadoop fs -mkdir /Receipts

\$ hadoop fs -mkdir /Receipts/input

\$ hadoop fs -put data/receipts.txt /Receipts/input

```
bigdata@slave1:~/MapReduce/Receipts$ pwd
/home/bigdata/MapReduce/Receipts
bigdata@slave1:~/MapReduce/Receipts$ ls
build.sh classes input ReceiptsDriver.java Receipts.jar ReceiptsMapper.java ReceiptsReducer.java
bigdata@slave1:~/MapReduce/Receipts$ ls input
receipts.txt
bigdata@slave1:~/MapReduce/Receipts$
bigdata@slave1:~/MapReduce/Receipts$ hadoop fs -mkdir /Receipts
bigdata@slave1:~/MapReduce/Receipts$ hadoop fs -mkdir /Receipts/input
bigdata@slave1:~/MapReduce/Receipts$ hadoop fs -put input/receipts.txt /Receipts/input
bigdata@slave1:~/MapReduce/Receipts$
bigdata@slave1:~/MapReduce/Receipts$ hadoop fs -ls /Receipts/input
Found 1 items
-rw-r--r-- 3 bigdata supergroup 7218 2019-07-12 02:54 /Receipts/input/receipts.txt
bigdata@slave1:~/MapReduce/Receipts$
```

Receipts.jar 실행

□ Receipts.jar 하둡 실행

**\$ hadoop jar Receipts.jar Receipts.ReceiptsDriver
/Receipts/input/receipts.txt /Receipts/output**

```
bigdata@slave1:~/MapReduce/Receipts$ hadoop jar Receipts.jar Receipts.ReceiptsDriver /Receipts/input/receipts.txt /Receipts/output
19/07/12 03:01:22 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
19/07/12 03:01:22 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
19/07/12 03:01:23 INFO input.FileInputFormat: Total input paths to process : 1
19/07/12 03:01:23 INFO mapreduce.JobSubmitter: number of splits:1
19/07/12 03:01:24 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local233203970_0001
19/07/12 03:01:24 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
```

```

        IV_COUNT=0
        WRONG_LENGTH=0
        WRONG_MAP=0
        WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=7218
File Output Format Counters
  Bytes Written=44
bigdata@slave1:~/MapReduce/Receipts$

```

Receipts.jar 실행 결과확인

□ Receipts.jar 실행 결과 확인

\$ hadoop fs -cat /Receipts/output/part-r-00000

```
bigdata@slave1:~/MapReduce/Receipts$ hadoop fs -ls /Receipts/output
Found 2 items
-rw-r--r--  3 bigdata supergroup          0 2019-07-12 03:01 /Receipts/output/_SUCCESS
-rw-r--r--  3 bigdata supergroup        44 2019-07-12 03:01 /Receipts/output/part-r-00000
bigdata@slave1:~/MapReduce/Receipts$
bigdata@slave1:~/MapReduce/Receipts$ hadoop fs -cat /Receipts/output/part-r-00000
min(2009):      -1412688.0
max(2000):      236241.0
bigdata@slave1:~/MapReduce/Receipts$
```

Receipts.jar 수정 1

□ 리듀서 실행을 생략하도록 코드를 수정

- ReceiptsDriver.java 파일에서 리듀서 클래스 설정을 코멘트 처리
`// job.setReducerClass(ReceiptsReducer.class);`
- 리듀서는 실행되지 않고 맵의 결과만 출력

□ 재빌드 후 실행

```
$ sh build.sh
```

```
$ hadoop jar Receipts.jar Receipts.ReceiptsDriver  
/Receipts/input/receipts.txt /Receipts/output1
```

```
$ hadoop fs -cat /Receipts/output1/part-r-00000
```

ReceiptsDriver.java

```
package Receipts;

import java.util.*;
import org.apache.hadoop.conf.Configured;
.....
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;

public class ReceiptsDriver extends Configured implements Tool {
    public int run(String[] args) throws Exception {
        .....
        Job job = new Job(getConf(), "my receipts");

        job.setJarByClass(ReceiptsDriver.class);
        job.setMapperClass(ReceiptsMapper.class);
        // job.setReducerClass(ReceiptsReducer.class);

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

        .....
    }

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        System.exit(ToolRunner.run(conf, new ReceiptsDriver(), args));
    }
}
```


Receipts.jar 수정 1 실행 결과

```
bigdata@slave1:~/MapReduce/Receipts1$ hadoop fs -cat /Receipts/output1/part-r-00000
summary 2012_-1086963
summary 2011_-1299593
summary 2010_-1294373
summary 2009_-1412688
summary 2008_-458553
summary 2007_-160701
summary 2006_-248181
summary 2005_-318346
summary 2004_-412727
summary 2003_-377585
summary 2002_-157758
summary 2001_128236
summary 2000_236241
summary 1999_125610
summary 1998_69270
summary 1997_-21884
summary 1996_-107431
summary 1995_-163952
summary 1994_-203186
summary 1993_-355054

summary 1904_-43
summary 1903_45
summary 1902_77
summary 1901_63
bigdata@slave1:~/MapReduce/Receipts1$
```

Receipts.jar 수정 2

- 리듀서에 **평균 값 계산**을 추가
 - 리듀서 매 반복 시에 누적합과 누적 카운트를 구하여 평균 값 계산
- 재빌드 후 실행

```
$ sh build.sh
$ hadoop jar Receipts.jar Receipts.ReceiptsDriver
/Receipts/input/receipts.txt /Receipts/output2
$ hadoop fs -cat /Receipts/output2/part-r-00000
```


ReceiptsReducer.java (1)

```
package Receipts;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.FloatWritable;
import java.io.IOException;

public class ReceiptsReducer extends Reducer<Text,Text,Text,FloatWritable> {
    private static Text tempYear=new Text();
    private static Text keyText=new Text();
    private static Text minYear=new Text();
    private static Text maxYear=new Text();
    private static String tempString;
    private static String[] keyString;

    public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
        InterruptedException {
        long sum = 0L;
        long count = 0L;
        long tempValue = 0L;
        long max=Long.MIN_VALUE;
        long min=Long.MAX_VALUE;

```

ReceiptsReducer.java (2)

```
    for (Text value: values) {
        tempString = value.toString();
        keyString = tempString.split("_");
        tempYear = new Text(keyString[0]);
        tempValue = new Long(keyString[1]).longValue();
        if(tempValue < min) {
            min=tempValue;
            minYear=tempYear;
        }
        if(tempValue > max) {
            max=tempValue;
            maxYear = tempYear;
        }
        sum = sum+tempValue;
        count++;
    }
    float mean = sum / count;

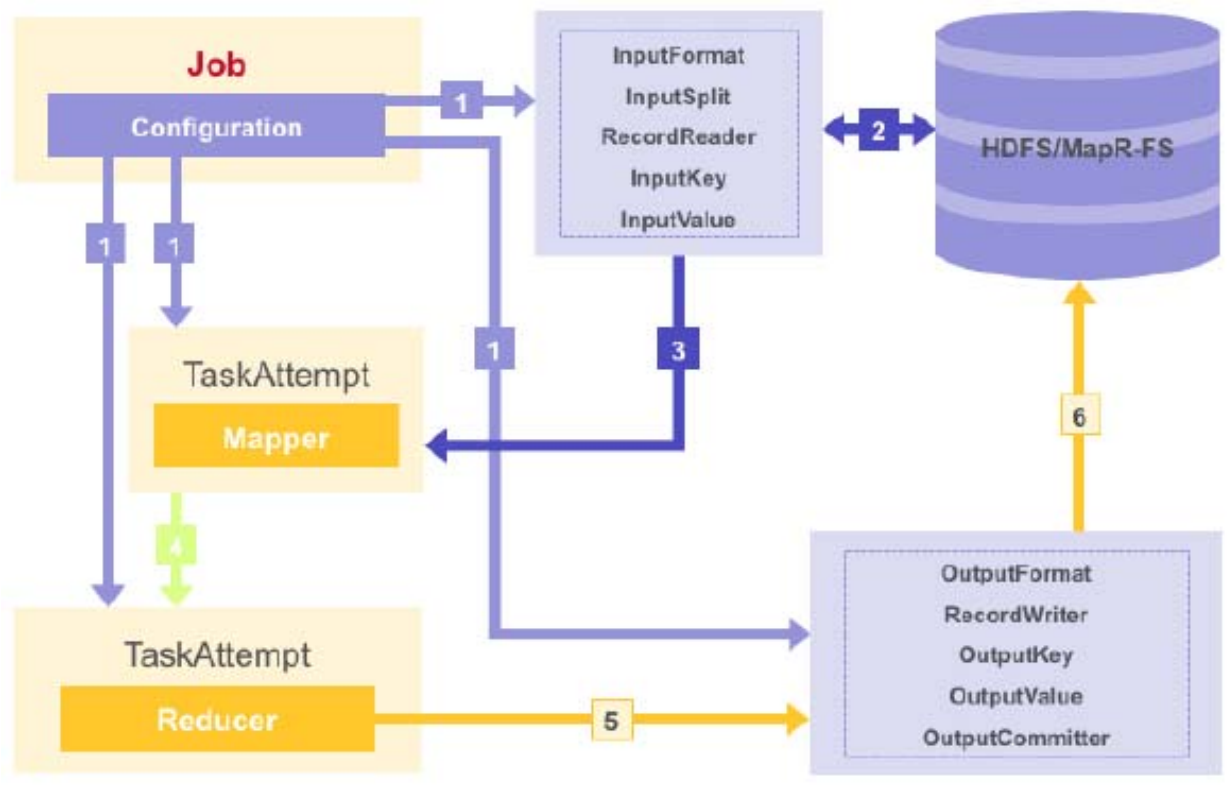
    keyText.set("min" + "(" + minYear.toString() + "): ");
    context.write(keyText, new FloatWritable(min));
    keyText.set("max" + "(" + maxYear.toString() + "): ");
    context.write(keyText, new FloatWritable(max));
    keyText.set("mean: ");
    context.write(keyText, new FloatWritable(mean));
}
}
```

Receipts.jar 수정 2 실행 결과

```
bigdata@slave1:~/MapReduce/Receipts2$ hadoop fs -cat /Receipts/output2/part-r-00000  
min(2009):      -1412688.0  
max(2009):      236241.0  
mean:          -93862.0  
bigdata@slave1:~/MapReduce/Receipts2$
```

4. 맵리듀스 코드 실행 분석

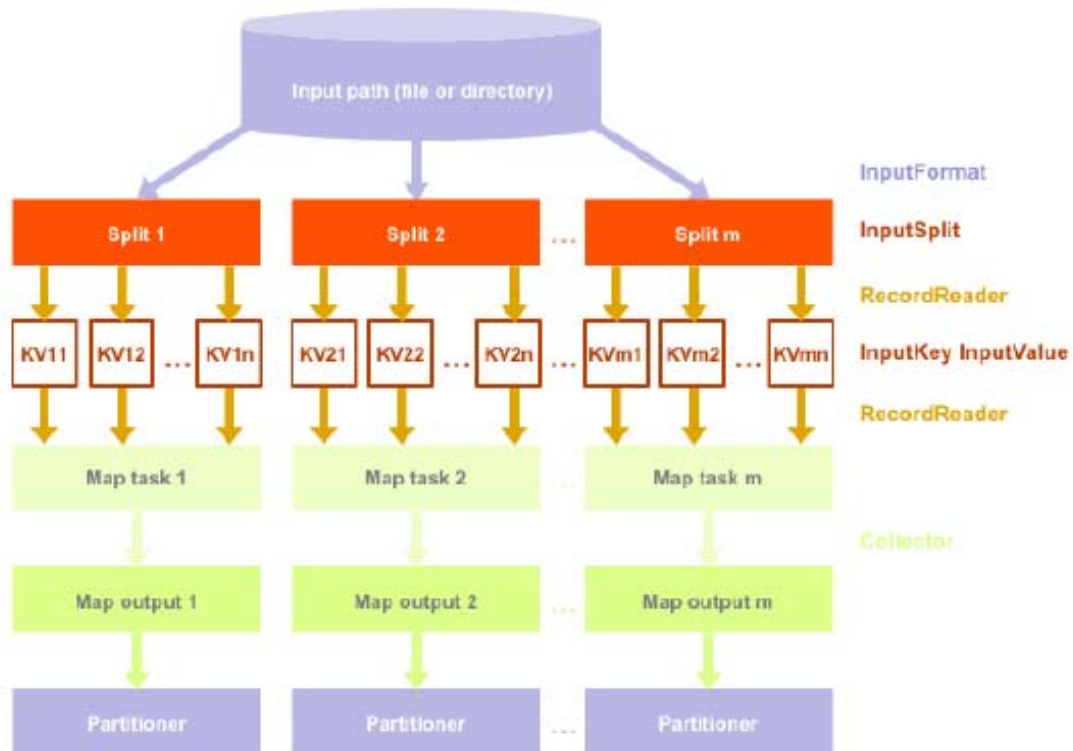
객체 상호 작용 (1)



객체 상호 작용 (2)

- 맵리듀스 작업을 실행하는 객체들 간의 관계
- **드라이버** 클래스
 - Job 과 Configuration 객체를 생성
 - set() 메서드를 사용하여 Configuration 객체의 각 요소들을 정의
- **매퍼** 클래스
 - HDFS에서 데이터를 읽고 처리
 - InputFormat, InputSplit, RecordReader, InputKey, InputValue 객체들을 사용
- **리듀서** 클래스
 - 리듀스 처리 후에 결과를 HDFS에 저장
 - OutputFormat, RecordWriter, OutputKey, OutputValue, OutputCommitter 객체들을 사용

매퍼 입력 처리 과정 (1)

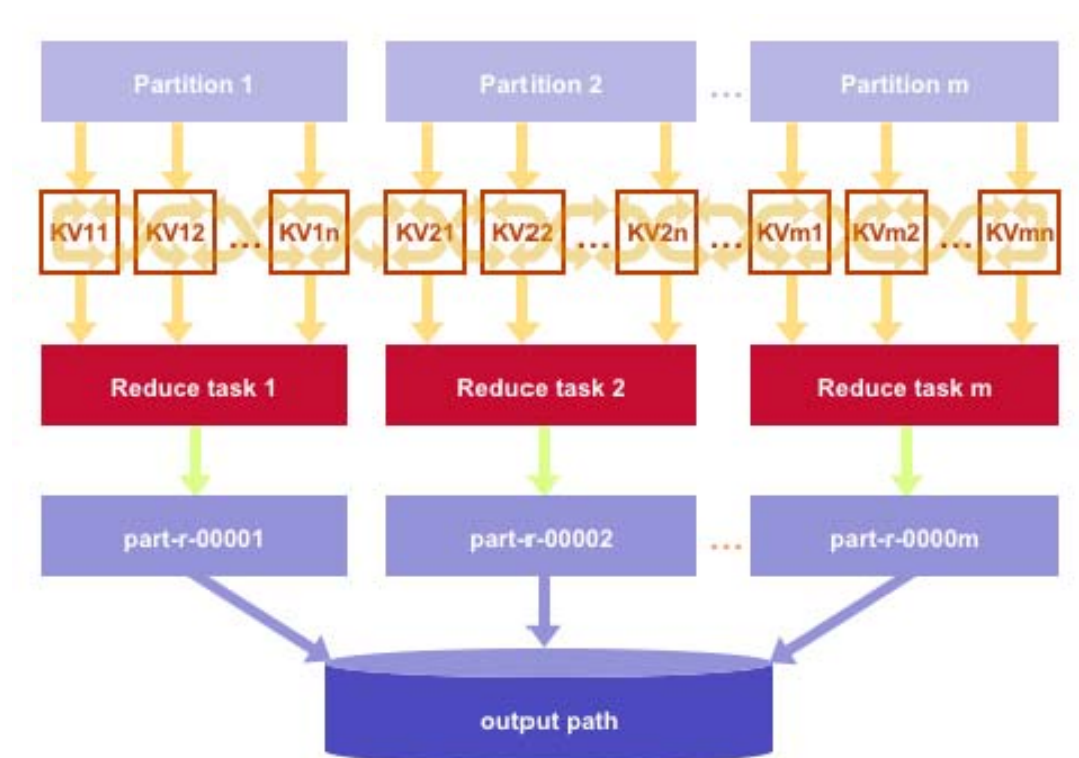


55

매퍼 입력 처리 과정 (2)

- ❑ **InputFormat** 클래스가 **입력 파일들은 분할**
- ❑ **RecordReader** 클래스가 분할된 각 **InputSplit**를 **키-값 쌍들**로 생성
 - **InputKey, InputValue** 클래스들은 **Writable** 클래스의 서브클래스
- ❑ 한 분할(split)의 모든 키-값 쌍들은 같은 매퍼로 전송되고 각 키-값 쌍들에 대해 **map() 메서드**가 호출
- ❑ 각 매퍼의 결과들은 수집되어 **파티셔너(partitioner)**로 전송되고, 매퍼의 지역 파일 시스템에 저장
- ❑ 이 후 리듀서 노드를 수행하는 하둡 프레임워크가 이들을 가져와 셔플, 정렬 단계를 수행

리듀서 출력 처리 과정 (1)



57

리듀서 출력 처리 과정 (2)

- 하둡 프레임워크는 매퍼로부터 파티션을 입력받아 셔플과 정렬한 후, 정렬된 키-값 쌍들을 각 리듀서에 단일 파티션으로 전송
 - 키는 해시 함수를 사용하여 파티션을 구동하는데 사용
 - 파티션의 수는 작업의 리듀스 태스크 수
- 리듀서는 파티션을 입력 받아 키-값 쌍들의 리스트(iterable list)를 처리하여 part-r-0000x (x는 리듀서의 번호) 출력 파일로 저장

5. WordCount 예

맵리듀스 응용 구축

WordCountMapper.java

```
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import java.io.IOException;
import java.util.StringTokenizer;

public class WordCountMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

    public void map(LongWritable key, Text input, Context context){
        try {
            StringTokenizer tokenizer = new StringTokenizer(input.toString(), " ,:;&W'W'()[]<>*/+ -=#%@!");
            while(tokenizer.hasMoreTokens()){
                Text word = new Text();
                word.set(tokenizer.nextToken());
                context.write(word, new IntWritable(1));
            }
        } catch (IOException e) {
            e.printStackTrace();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

WordCountReducer.java

```
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import java.io.IOException;

public class WordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

    public void reduce(Text key, Iterable<IntWritable> values, Context context){
        try {
            int resultNumber = 0;
            for(IntWritable value : values){
                resultNumber += value.get();
            }
            IntWritable result = new IntWritable();
            result.set(resultNumber);
            context.write(key, result);
        } catch (IOException e) {
            e.printStackTrace();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

WordCountJob.java

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

public class WordCountJob {
    public static void main(String[] args){
        try {
            Configuration conf = new Configuration();
            Job job = Job.getInstance(conf, "WordCountJob");
            job.setJarByClass(WordCountJob.class);
            job.setMapperClass(WordCountMapper.class);
            job.setCombinerClass(WordCountReducer.class);
            job.setReducerClass(WordCountReducer.class);
            job.setOutputKeyClass(Text.class);
            job.setOutputValueClass(IntWritable.class);
            FileInputFormat.addInputPath(job, new Path(args[0]));
            FileOutputFormat.setOutputPath(job, new Path(args[1]));
            System.exit(job.waitForCompletion(true) ? 0 : 1);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```
#!/bin/bash

export HADOOP_HOME=/home/bigdata/hadoop-2.7.7
#export LD_LIBRARY_PATH=$HADOOP_HOME/lib/native/
export
  CLASSPATH=$HADOOP_HOME/share/hadoop/mapreduce/*:$HADOOP_HOME/share/
  /hadoop/mapreduce/lib/*:$HADOOP_HOME/share/hadoop/common/lib/*:$HADOOP
  _HOME/share/hadoop/common/*:$HADOOP_HOME/share/hadoop/tools/lib/*:$HAD
  OOP_HOME/share/hadoop/yarn/*:$HADOOP_HOME/share/hadoop/yarn/lib/*
export HADOOP_CLASSPATH=$CLASSPATH

javac -d classes WordCountMapper.java
javac -d classes WordCountReducer.java
jar -cvf WordCount.jar -C classes/ .
javac -classpath $CLASSPATH:WordCount.jar -d classes WordCountJob.java
jar -uvf WordCount.jar -C classes/ .
```

WordCount.jar 파일 빌드

- Wordcount.jar 빌드
 - \$ mkdir classes
 - \$ sh build.sh

```
bigdata@slave1:~/MapReduce/WordCount$ mkdir classes
bigdata@slave1:~/MapReduce/WordCount$ 
bigdata@slave1:~/MapReduce/WordCount$ sh build.sh
added manifest
adding: WordCountReducer.class(in = 1756) (out= 774)(deflated 55%)
adding: WordCountMapper.class(in = 1798) (out= 786)(deflated 56%)
adding: WordCountReducer.class(in = 1756) (out= 774)(deflated 55%)
adding: WordCountMapper.class(in = 1798) (out= 786)(deflated 56%)
adding: WordCountJob.class(in = 1469) (out= 802)(deflated 45%)
bigdata@slave1:~/MapReduce/WordCount$ 
bigdata@slave1:~/MapReduce/WordCount$ ls
build.sh  classes  WordCount.jar  WordCountJob.java  WordCountMapper.java  WordCountReducer.java
bigdata@slave1:~/MapReduce/WordCount$
```


입력 데이터의 HDFS 복사

□ 입력 데이터 로컬 파일 복사

- 로컬파일: ~/hadoop-2.7.7/*.txt
- HDFS 파일 디렉토리: /WordCount/input

\$ **hadoop fs -mkdir /WordCount**

\$ **hadoop fs -mkdir /WordCount/input**

\$ **hadoop fs -copyFromLocal ~/hadoop-2.7.7/*.txt /WordCount/input**

```
bigdata@slave1:~/MapReduce/WordCount$ ls ~/hadoop-2.7.7/*.txt
/home/bigdata/hadoop-2.7.7/LICENSE.txt  /home/bigdata/hadoop-2.7.7/README.txt
/home/bigdata/hadoop-2.7.7/NOTICE.txt
```

```
bigdata@slave1:~/MapReduce/WordCount$ hadoop fs -mkdir /WordCount
```

```
bigdata@slave1:~/MapReduce/WordCount$ hadoop fs -mkdir /WordCount/input
```

```
bigdata@slave1:~/MapReduce/WordCount$ hadoop fs -copyFromLocal ~/hadoop-2.7.7/*.txt /WordCount/input
```

```
bigdata@slave1:~/MapReduce/WordCount$
```

```
bigdata@slave1:~/MapReduce/WordCount$ hadoop fs -ls /WordCount/input
```

```
Found 3 items
```

```
-rw-r--r--  3 bigdata supergroup      86424 2019-07-12 05:16 /WordCount/input/LICENSE.txt
```

```
-rw-r--r--  3 bigdata supergroup      14978 2019-07-12 05:16 /WordCount/input/NOTICE.txt
```

```
-rw-r--r--  3 bigdata supergroup       1366 2019-07-12 05:16 /WordCount/input/README.txt
```

```
bigdata@slave1:~/MapReduce/WordCount$
```

WordCount.jar 파일 실행

\$ **hadoop jar WordCount.jar WordCountJob /WordCount/input /WordCount/output**

```
bigdata@slave1:~/MapReduce/WordCount$ hadoop jar WordCount.jar WordCountJob /WordCount/input /WordCount/output
```

```
19/07/12 05:20:47 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
```

```
19/07/12 05:20:47 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
```

```
19/07/12 05:20:48 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
```

```
19/07/12 05:20:48 INFO input.FileInputFormat: Total input paths to process : 3
```

```
19/07/12 05:20:48 INFO mapreduce.JobSubmitter: number of splits:3
```

```
19/07/12 05:20:48 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1855259091_0001
```

```
19/07/12 05:20:49 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
```

```
19/07/12 05:20:49 INFO mapreduce.Job: Running job: job_local1855259091_0001
```

```
10 ERRORS=0
```

```
WRONG_LENGTH=0
```

```
WRONG_MAP=0
```

```
WRONG_REDUCE=0
```

```
File Input Format Counters
```

```
Bytes Read=102768
```

```
File Output Format Counters
```

```
Bytes Written=19069
```

```
bigdata@slave1:~/MapReduce/WordCount$
```

WordCount 실행 결과

\$ **hadoop fs -ls /WordCount/output**

\$ **hadoop fs -cat /WordCount/output/part-r-00000 | more**

```
bigdata@slave1:~/MapReduce/WordCount$ hadoop fs -ls /WordCount/output
Found 2 items
-rw-r--r--  3 bigdata supergroup          0 2019-07-12 05:20 /WordCount/output/_SUCCESS
-rw-r--r--  3 bigdata supergroup    19069 2019-07-12 05:20 /WordCount/output/part-r-00000
```

```
bigdata@slave1:~/MapReduce/WordCount$ hadoop fs -cat /WordCount/output/part-r-00000 | more
0      37
034819 1
04      1
1      96
10      6
101     2
11      4
12      5
13      3
166     1
17      1
1995    8
1996    1
1999    4
2       99
2000    1
```

| | | |
|----------------|----------------|-----------------------|
| 60 6 | ON 16 | Modifications means 1 |
| 7 7 | OR 170 | Original 1 |
| 7014 2 | ORG 1 | Patent 1 |
| 7202 4 | OTHER 16 | Source 1 |
| 740 1 | OTHERWISE 20 | You 1 |
| 8 8 | OUT 18 | for 4 |
| 9 10 | OW2 2 | means 1 |
| 94 1 | OWNER 7 | otherwise 1 |
| A 25 | Object 5 | ownership 1 |
| ACCEPTANCE 1 | Obligations 2 | rename 1 |
| ACCOMPANYING 1 | Oct 2 | the 5 |
| ACTION 5 | OneLab 1 | third 1 |
| ADVISED 13 | OpenJDK 2 | under 1 |
| AGREEMENT 4 | Oracle 3 | { 1 |
| AIX 1 | Original 37 | 'AS 1 |
| ALL 3 | Otherwise 2 | 'Contributor 1 |
| ALLOW 2 | P 1 | 'Contributor" 1 |
| ALPN 2 | PART 2 | 'Covered 1 |
| AN 10 | PARTICULAR 21 | 'Executable" 1 |
| AND 67 | PARTIES 2 | Initial 1 |
| ANY 98 | PARTY 3 | 'Larger 1 |
| APACHE 1 | PARTYS 1 | 'Licensable" 1 |
| API 6 | PERFORMANCE 2 | 'License" 1 |
| APIs 2 | PERSON 2 | Modifications" 1 |
| APPENDIX 1 | PERSONAL 2 | 'Original 1 |
| APPLICABLE 2 | POSSIBILITY 15 | 'Participant" 1 |
| APPLY 4 | POWER 1 | 'Patent 1 |
| ARE 12 | PROCUREMENT 12 | 'Source 1 |
| ARISING 18 | PROFITS 14 | 'Your" 1 |
| AS 24 | PROGRAM 4 | 'You" 2 |
| ASL2 6 | PROHIBITS 2 | 'commercial 3 |
| ASM 1 | PROVE 2 | 'control" 1 |
| ASSUME 2 | PROVIDED 21 | " 1 |
| AUTHOR 2 | --More-- | |

```
bigdata@slave1:~/MapReduce/WordCount$
```

- 강의 시간의 실습 내용을 정리하여 제출
- 임의의 맵리듀스 프로그램을 작성
 - 맵리듀스 프로그램 내용 분석
 - 실행 결과 분석

- MapR Academy, <http://learn.mapr.com/>
 - Build Hadoop MapReduce Applications
 - <https://learn.mapr.com/series/hadoop-developer-series/dev-300-build-hadoop-mapreduce-applications>
 - Lesson 2: Job Execution Framework
 - Lesson 3: Write a MapReduce Program
- WordCount 예
 - <https://hashnode.com/post/how-i-was-finally-able-to-run-the-infamous-word-count-example-on-hadoop-ciwazrq8u000vgw53qe4saran>