



What is HADÖÖP & SPÄRK

In this presentation, I will explain Hadoop&Spark and how to use it.

Jeon Dabin

[GitHub주소 : https://github.com/DabinJeon/HADOOP](https://github.com/DabinJeon/HADOOP)



< 목차 >

- **About Hadoop**
- Hadoop Setup
- Hadoop-Mapreduce
- Hadoop-Java
- **About Hive**
- Hive Setup
- Hive-HiveQL
- **About Spark**
- Spark Setup
- Spark-Scala
- Spark-Python

“ Hadoop
&
Spark ”

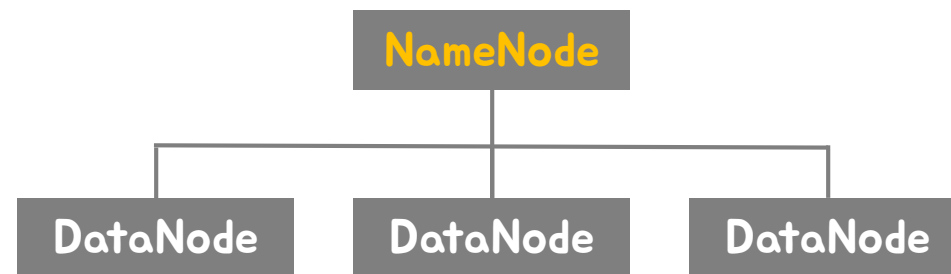
About Hadoop

The screenshot shows the Hadoop NameNode information page in a Mozilla Firefox browser. The page title is "Namenode information - Mozilla Firefox". The address bar shows "master:50070/dfshealth.html#tab-datanode". The page displays a table of Hadoop nodes with columns: Node, Http Address, Last contact, Last Block Report, Capacity, Blocks, Block pool used, and Version. The table lists four nodes: master:50010, Slave1:50010, Slave2:50010, and Slave3:50010. Each node has a green checkmark in the "Node" column. The "Capacity" column shows 13.99 GB for all nodes. The "Blocks" column shows 2 for master, 5 for Slave1, 2 for Slave2, and 1 for Slave3. The "Block pool used" column shows 216 KB (0%) for master, 272 KB (0%) for Slave1, 60 KB (0%) for Slave2, and 20 KB (0%) for Slave3. The "Version" column shows 2.9.2 for all nodes. The page also includes a search bar, a "Show 25 entries" dropdown, and pagination controls.

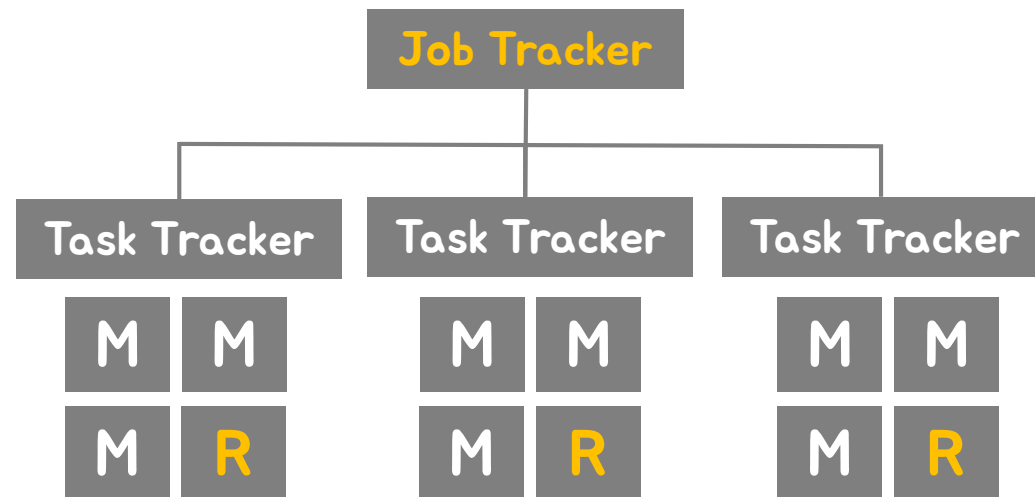
Node	Http Address	Last contact	Last Block Report	Capacity	Blocks	Block pool used	Version
✓ master:50010 (192.168.10.1:50010)	http://master:50075	0s	35m	13.99 GB	2	216 KB (0%)	2.9.2
✓ Slave1:50010 (192.168.10.2:50010)	http://Slave1:50075	514s	35m	13.99 GB	5	272 KB (0%)	2.9.2
✓ Slave2:50010 (192.168.10.3:50010)	http://Slave2:50075	1s	35m	13.99 GB	2	60 KB (0%)	2.9.2
✓ Slave3:50010 (192.168.10.4:50010)	http://Slave3:50075	505s	35m	13.99 GB	1	20 KB (0%)	2.9.2

<http://localhost:50070> -> NameNode, DataNode 확인

< HDFS (하둡분산파일시스템) >



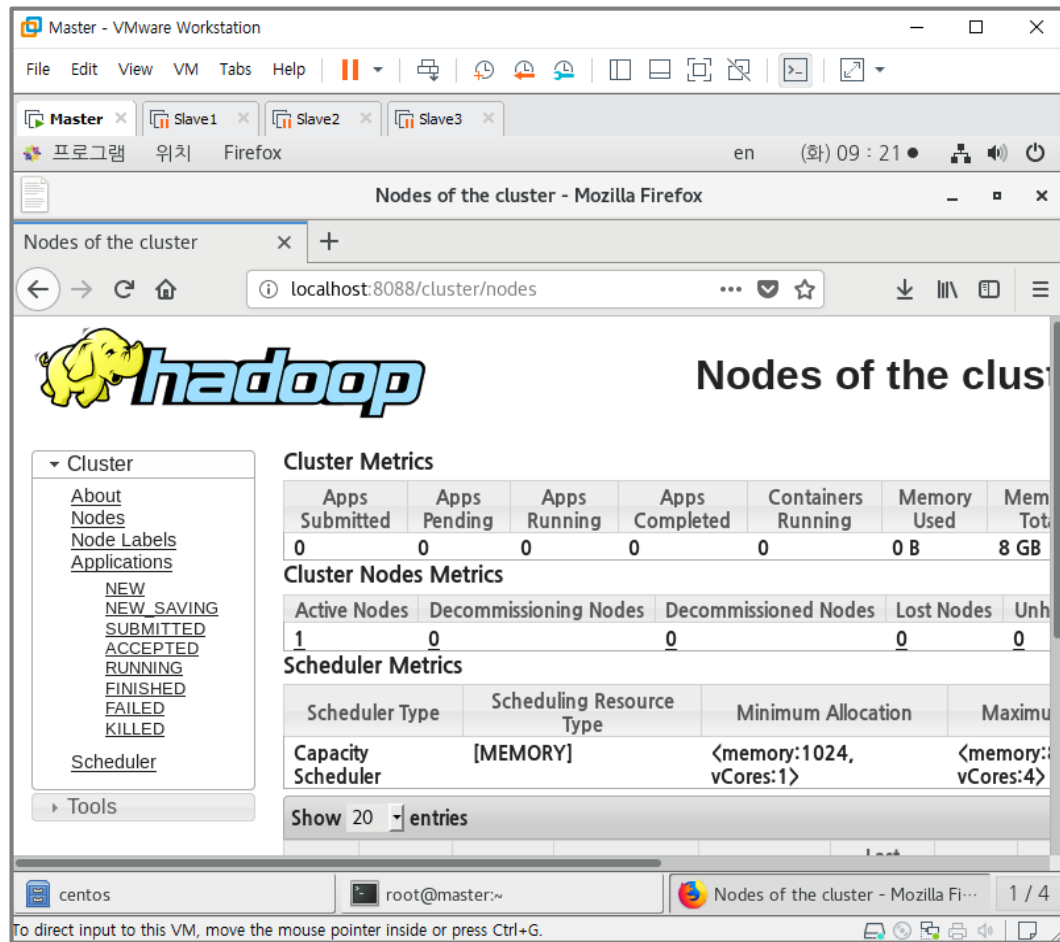
< MapReduce (병렬처리) >



(각 TaskTracker마다 맵,리듀스 슬롯을 구분하여 작업을 처리)

About Hadoop

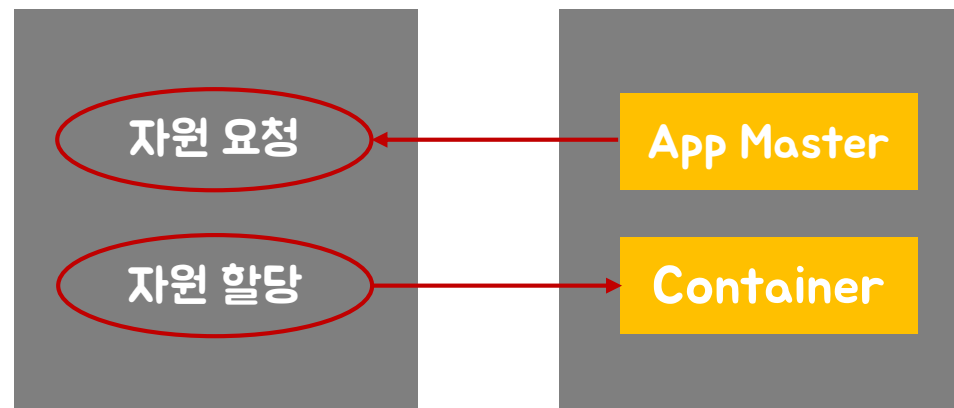
< YARN (클러스터자원관리시스템) >



<http://localhost:8088> -> YARN에 대한 인터페이스

Resource Manager

Node Manager



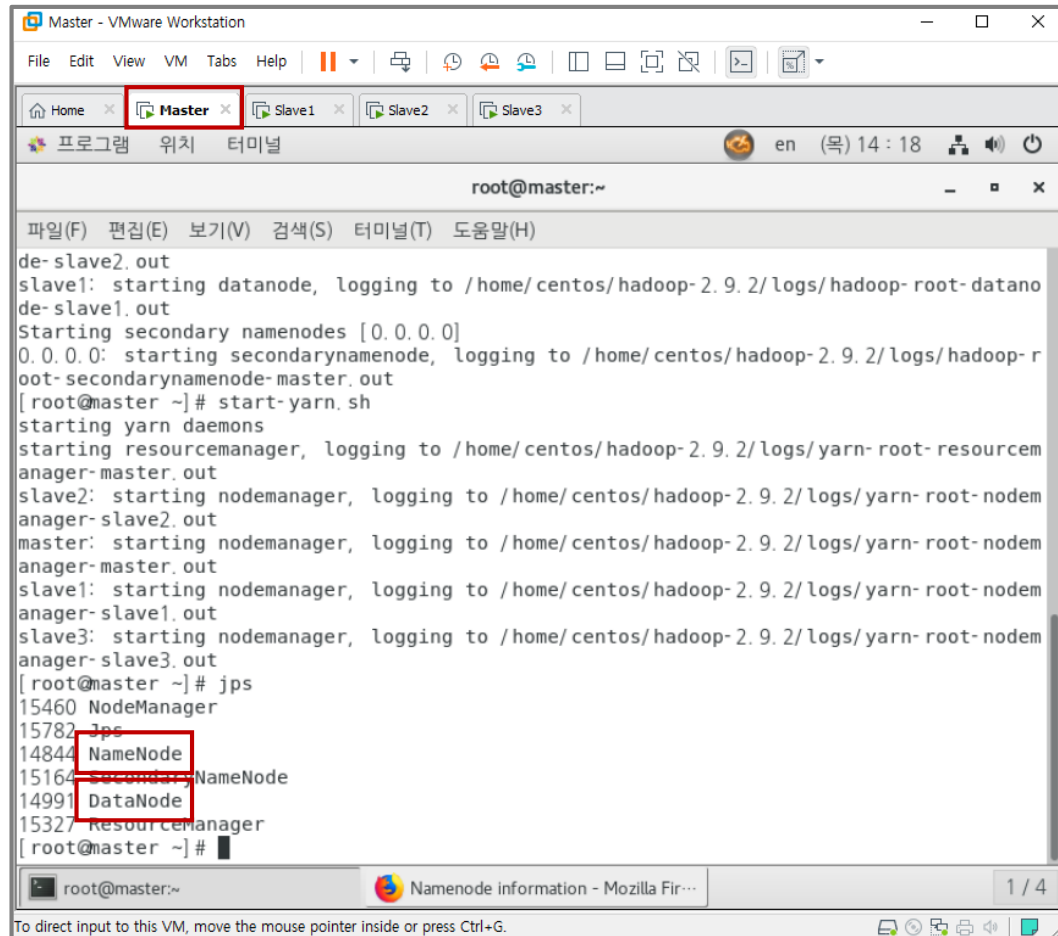
<클러스터(자원) 관리>

<작업 관리>

- Resource Manager
- Node Manager
- Application Master
- Container

1. 작업이 생기면 App Master가 생성됨
2. App Master는 Resource Manager에게 지원을 요청
3. 실제 작업을 담당하는 Container를 할당 받아서 작업처리
4. 작업이 완료되면 Container종료

Hadoop Setup

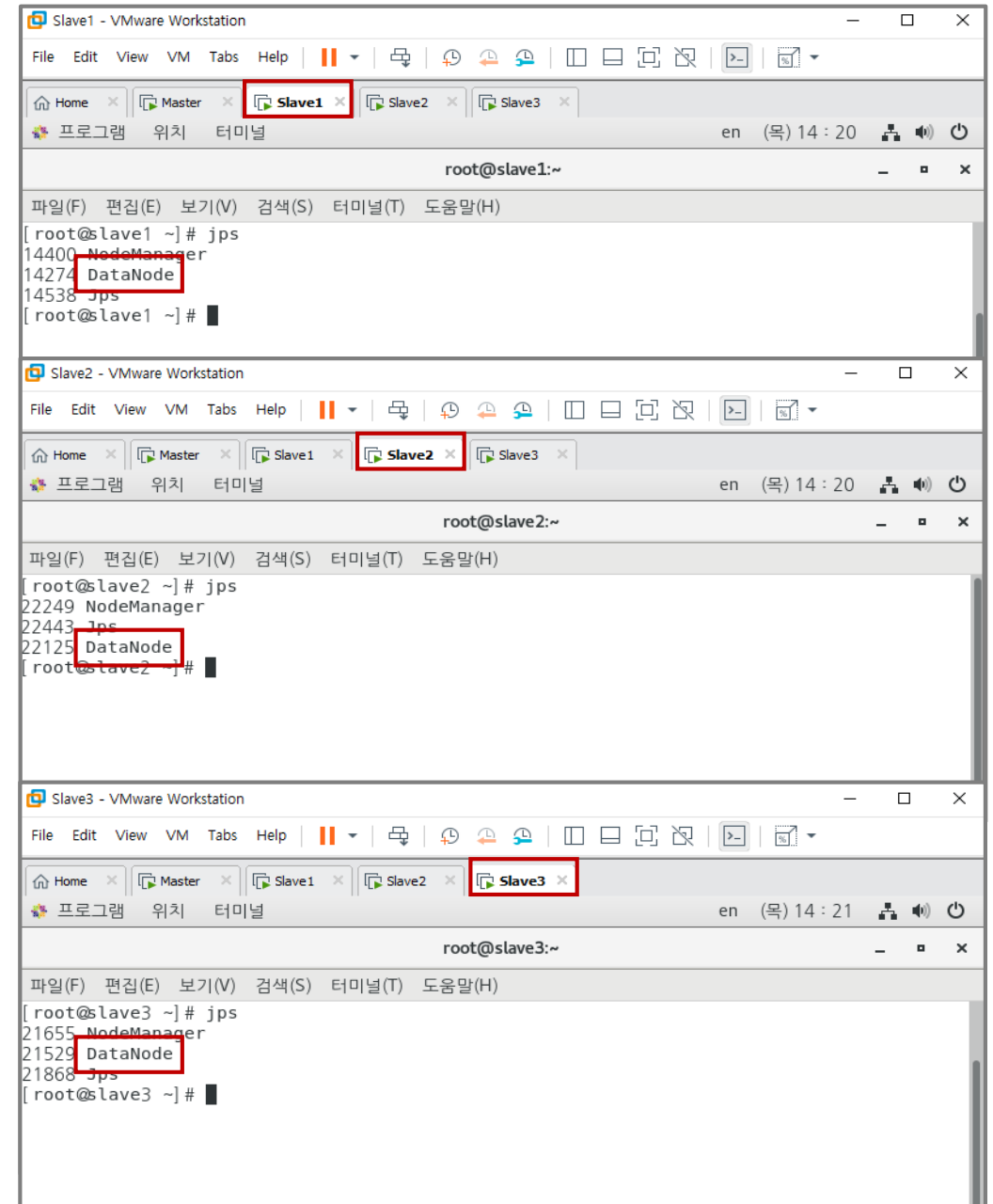


The screenshot shows the Master VM terminal with the following output:

```
de-slave2.out
slave1: starting datanode, logging to /home/centos/hadoop-2.9.2/logs/hadoop-root-datano
de-slave1.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/centos/hadoop-2.9.2/logs/hadoop-r
oot-secondarynamenode-master.out
[root@master ~]# start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /home/centos/hadoop-2.9.2/logs/yarn-root-resourcem
anager-master.out
slave2: starting nodemanager, logging to /home/centos/hadoop-2.9.2/logs/yarn-root-nodem
anager-slave2.out
master: starting nodemanager, logging to /home/centos/hadoop-2.9.2/logs/yarn-root-nodem
anager-master.out
slave1: starting nodemanager, logging to /home/centos/hadoop-2.9.2/logs/yarn-root-nodem
anager-slave1.out
slave3: starting nodemanager, logging to /home/centos/hadoop-2.9.2/logs/yarn-root-nodem
anager-slave3.out
[root@master ~]# jps
15460 NameNode
15782 jps
14844 SecondaryNameNode
15164 SecondaryNameNode
14991 DataNode
15327 ResourceManager
[root@master ~]#
```

At the bottom of the terminal window, there is a browser tab titled "Namenode information - Mozilla Fir..." and a status bar indicating "To direct input to this VM, move the mouse pointer inside or press Ctrl+G."

- NameNode 1개 : Master
- DataNode 4개 : Master, Slave1, Slave2, Slave3



The screenshots show the Slave1, Slave2, and Slave3 VM terminals with the following output:

Slave1:

```
[root@slave1 ~]# jps
14400 NameNode
14274 DataNode
14538 jps
[root@slave1 ~]#
```

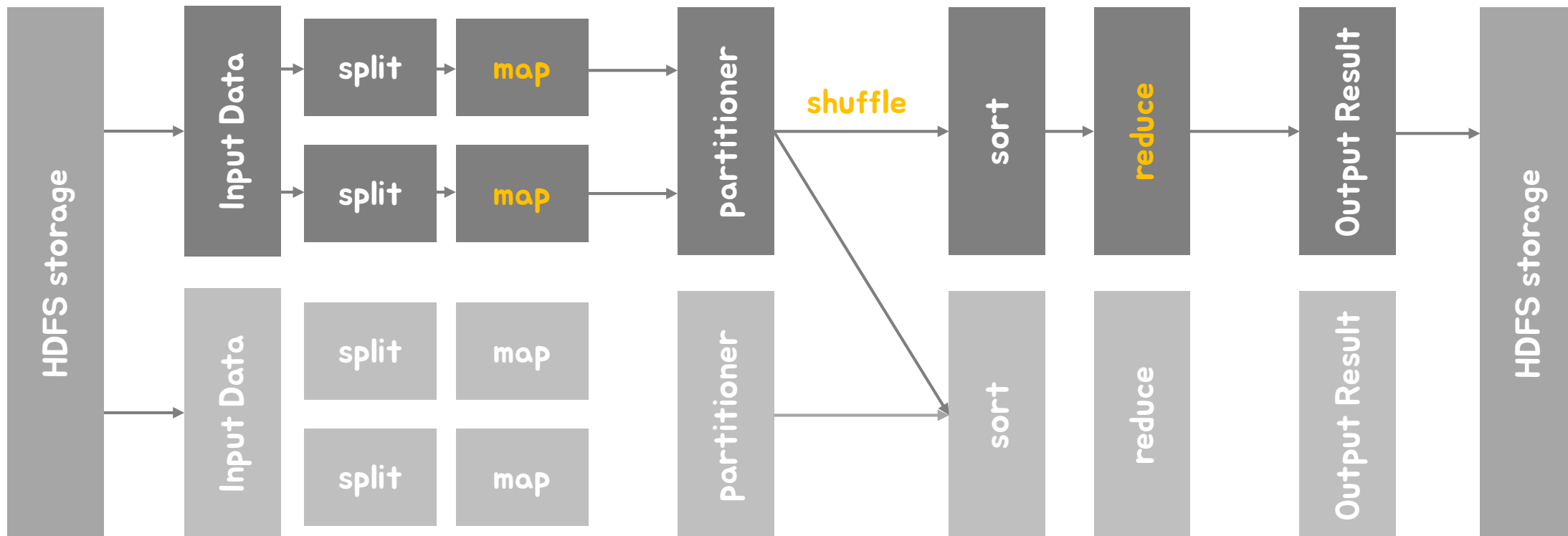
Slave2:

```
[root@slave2 ~]# jps
22249 NameNode
22443 jps
22125 DataNode
[root@slave2 ~]#
```

Slave3:

```
[root@slave3 ~]# jps
21655 NameNode
21529 DataNode
21868 jps
[root@slave3 ~]#
```

< MapReduce 처리단계 >



- **Map** : 입력을 분할하여 키별로 데이터를 처리
- **Combiner** : 맵의 결과를 정리 (설정에 따라 없을 수도 있음)
- **Partitioner** : 키값을 해쉬 처리하여 어떤 리듀서로 넘길지 결정
- **Shuffle** : 각 리듀서로 데이터 이동
- **Sort** : 리듀서로 전달된 데이터를 키값 기준으로 정렬
- **Reduce** : 리듀서로 전달된 데이터를 처리하고 결과를 저장

< MapReduce 간단 동작 테스트 >

The screenshot shows a terminal window on a Master node in a Hadoop cluster. The user runs the command `hdfs dfs -ls ~/wordcount-output`, which lists two files: `_SUCCESS` and `part-r-00000`. Then, the user runs `hdfs dfs -cat ~/wordcount-output/part-r-00000`, which outputs a list of words and their counts. The output is as follows:

```
(BIS), 1
(ECCN) 1
(TSU) 1
(see 1
5D002. C. 1, 1
740. 13) 1
<http://www.wassenaar.org/> 1
Administration 1
Apache 1
BEFORE 1
BIS 1
Bureau 1
Commerce, 1
Commodity 1
Control 1
Core 1
Department 1
ENC 1
Exception 1
```

A red box highlights the output of the `cat` command. Another red box highlights the command `hdfs dfs -ls ~/wordcount-output` in the terminal history.

단어의 개수를 count 한 결과가
키,값 형태로 출력됨

* master에서 실행 *

@ 맵리듀스 job을 실행을 위해, HDFS 디렉토리 생성

```
hdfs dfs -mkdir /user
```

```
hdfs dfs -mkdir /user/root
```

```
hdfs dfs -mkdir /user/root/conf
```

@ 테스트 파일을 HDFS에 업로드

```
hdfs dfs -mkdir /input
```

```
hdfs dfs -copyFromLocal /home/centos/hadoop-2.9.2/README.txt /input
```

```
hdfs dfs -ls /input
```

@ wordcount 프로그램 실행

(hadoop jar jar파일경로 클래스 입력파일 출력폴더)

```
hadoop jar
```

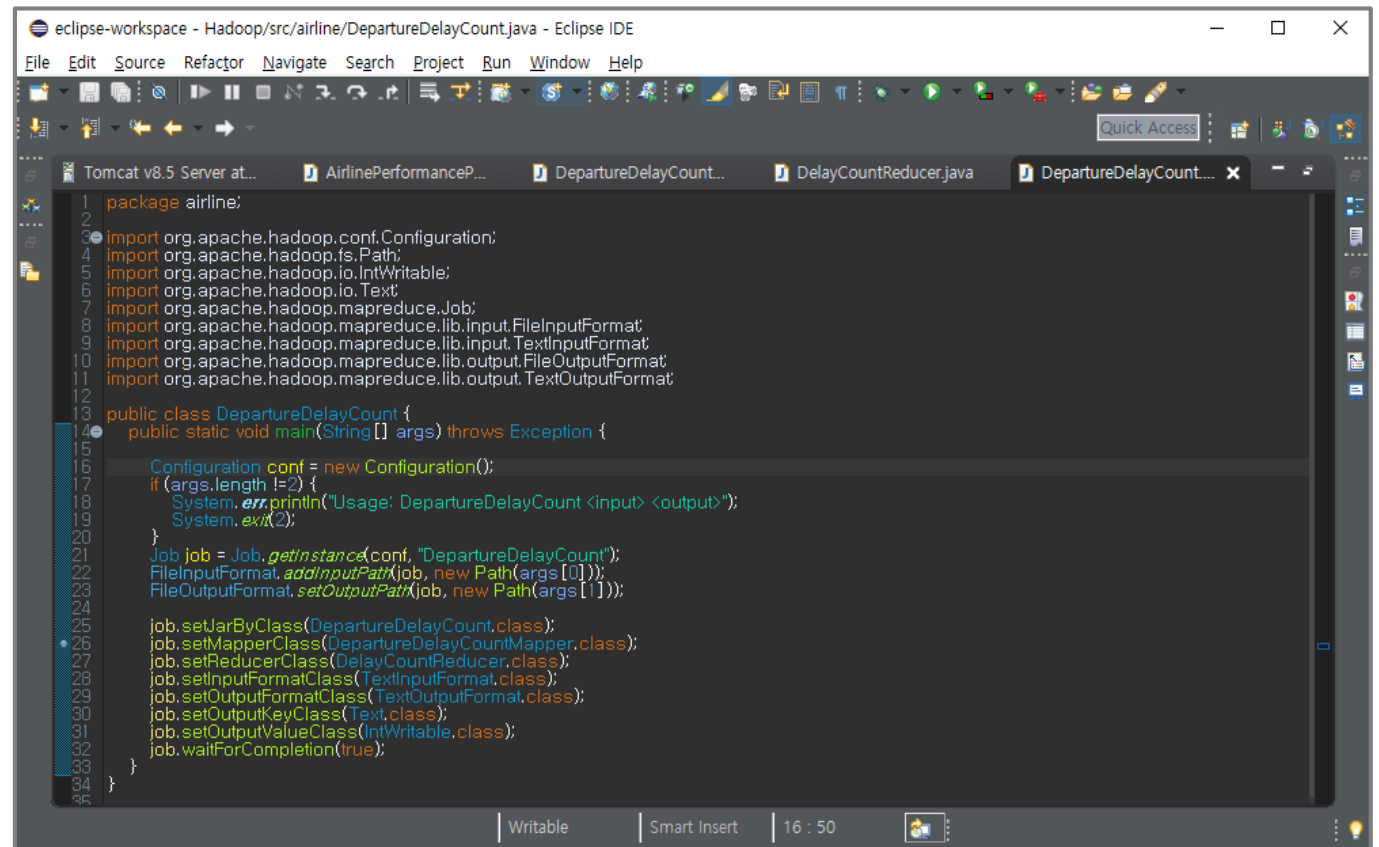
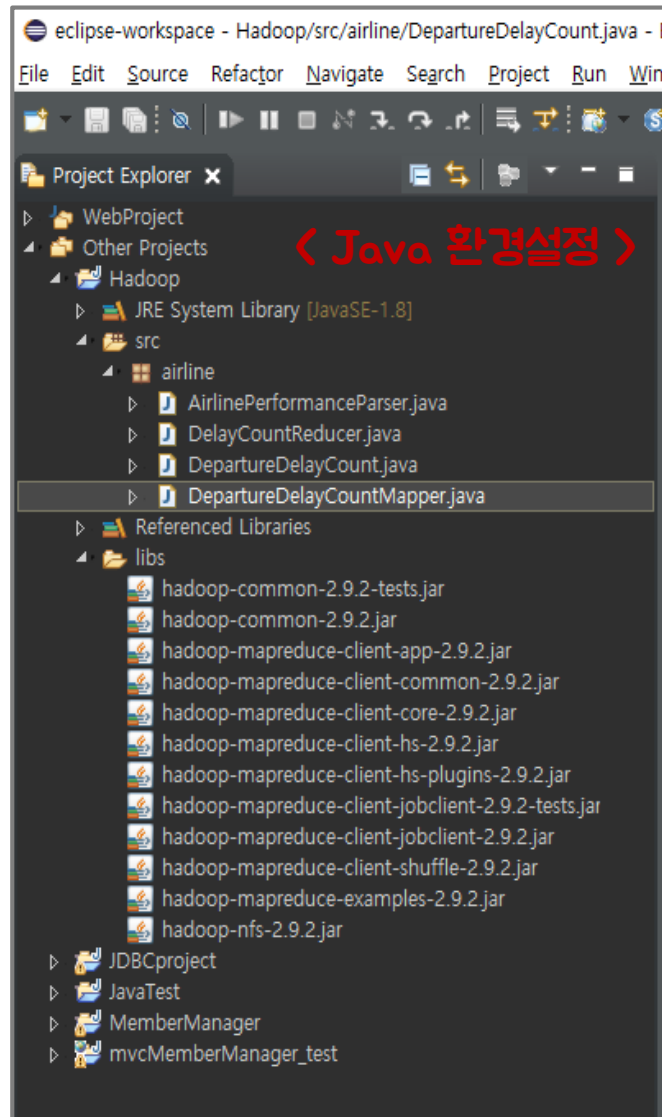
```
$HADOOP_HOME/share/hadoop/mapreduce/hadoop-  
-mapreduce-examples-2.9.2.jar wordcount  
/input/README.txt ~/wordcount-output
```

@ 작업결과 확인

```
hdfs dfs -ls ~/wordcount-output
```

```
hdfs dfs -cat ~/wordcount-output/part-r-00000
```

< Java Code 를 사용한 HDFS >



1. Hadoop project 생성
2. Libs 디렉토리 생성 후 .jar파일 추가
(/home/centos/hadoop-2.9.2/share/hadoop/common 에 있는 3개의 jar파일)
(/home/centos/hadoop-2.9.2/share/hadoop/mapreduce 에 있는 9개의 jar파일)
3. Java Build Path -> Libraries -> Add JARs

< Java Code > * Java Eclipse에서 작성 *

```
Package airline;
Import java.io.IOException;
Import org.apache.hadoop.io.IntWritable;
Import org.apache.hadoop.io.Text;
Import org.apache.hadoop.mapreduce.Reducer;

Public class DelayCountReducer extends
    Reducer<Text, IntWritable, Text, IntWritable> {

    private IntWritable result = new IntWritable();

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

        Int sum = 0;
        for (IntWritable value : values)
            sum+=value.get();

        result.set(sum);
        context.write(key,result);

    }
}
```

```
package airlines;
Import java.io.IOException;
Import org.apache.hadoop.io.IntWritable;
Import org.apache.hadoop.io.LongWritable;
Import org.apache.hadoop.io.Text;
Import org.apache.hadoop.mapreduce.Mapper;

public class DepartureDelayCountMapper extends
    Mapper<LongWritable, Textm Text, IntWritable>{

    private final static IntWritable outputValue =
        new IntWritable(1);
    private Text outputKey = new Text();

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

        AirlinePerformanceParser parser = new
            AirlinePerformanceParser(value);

        outputKey.set(parser.getYear() + '.'+parser.getMonth());

        if (parser.getDepartureDelayTime() >0) {
            context.write(outputkey, outputValue);

        }
    }
}
```

< Java Code > * Java Eclipse에서 작성 *

```
package airlines;
import org.apache.hadoop.conf.Configuration ...

public class DepartureDelayCount{
    public static void main(String[] args) throws Exception{
        Configuration conf = new Configuration();
        if (arg.lenth !=2){
            System.err.println('<input> <output>');
            System.exit(2)
        }

        Job job = Job.getInstance(conf, 'DepartureDelayCount');
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        job.setJarByClass(DepartureDelayCount.class);
        job.setMapperClass(DepartureDelayCountMapper.class);
        job.setReducerClass(DelayCountReducer.class);

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

        job.waitForCompletion(ture);
    }
}
```

```
package airlines;
import org.apache.hadoop.io.Text;
public class AirlinePerformanceParser{
    private int year;
    private int month;
    ...

    public AirlinePerformanceParser(Text text){
        try{
            String[] columns = text.toString().split(',');
            year = Integer.parseInt(columns[0]);
            month = Integer.parseInt(columns[1]);
            uniqueCarrier = columns[8];

            if (!columns[15].equals('NA')){
                departureDelayTime = Integer.parseInt(columns[15]);
            }else{
                departureDelayAvailable = false;
            }
            ...

        }catch (Exception e){
            System.out.println(e.getMessage());
        }
    }

    public int getYear() {return year;}
    public int getMonth() {return month;}
    ...
}
```

< Java Code 를 사용한 HDFS >

```

Master - VMware Workstation
File Edit View VM Tabs Help
Master x Slave1 x Slave2 x Slave3 x
프로그램 위치 터미널 en (일) 18:48
root@master:~

파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
[root@master ~]# hdfs dfs -put /home/centos/data/airline input
[root@master ~]# hadoop jar /home/centos/source/Hadoop.jar airline.DepartureDelayCount
input/airline delay_count
19/08/04 18:38:07 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
19/08/04 18:38:09 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing
not performed. Implement the Tool interface and execute your application with ToolRun
ner to remedy this.
19/08/04 18:38:09 INFO input.FileInputFormat: Total input files to process : 2
19/08/04 18:38:09 INFO mapreduce.JobSubmitter: number of splits:6
19/08/04 18:38:09 INFO Configuration.deprecation: yarn.resourcemanager.system-metrics-p
ublisher.enabled is deprecated. Instead, use yarn.system-metrics-publisher.enabled
19/08/04 18:38:10 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_156490702
0126_0008
19/08/04 18:38:10 INFO impl.YarnClientImpl: Submitted application application_156490702
0126_0008
19/08/04 18:38:10 INFO mapreduce.Job: The url to track the job: http://master:8088/prox
y/application_1564907020126_0008/
19/08/04 18:38:10 INFO mapreduce.Job: Running job: job_1564907020126_0008
19/08/04 18:38:26 INFO mapreduce.Job: Job job_1564907020126_0008 running in uber mode :
false
19/08/04 18:38:26 INFO mapreduce.Job: map 0% reduce 0%
19/08/04 18:39:33 INFO mapreduce.Job: map 1% reduce 0%
19/08/04 18:39:38 INFO mapreduce.Job: map 17% reduce 0%
19/08/04 18:39:40 INFO mapreduce.Job: map 18% reduce 0%
19/08/04 18:39:52 INFO mapreduce.Job: map 19% reduce 0%
19/08/04 18:39:58 INFO mapreduce.Job: map 21% reduce 0%
root@master:~
1 / 4
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

```

* master에서 실행 *

@ 작성한 Java Project를 jar파일로 export
이 jar파일을 /home/centos/source에 복사

@ HDFS에 데이터 업로드

(airline 폴더의 데이터 들을 hdfs input 폴더에 업로드)

```
hdfs dfs -mkdir input
```

```
hdfs dfs -put /home/centos/data/airline input
```

@ Hadoop 환경에서 Hadoop.jar파일 실행

(hadoop jar jar파일경로 클래스 입력데이터위치 출력위치)

```
hadoop jar /home/centos/source/Hadoop.jar
airline.DepartureDelayCount input/airline
delay_count
```

```

[root@master ~]# hdfs dfs -cat delay_count/part-r-00000
2006. 1 197789
2006. 10 248878
2006. 11 230224
2006. 12 274930
2006. 2 198371
2006. 3 235207
2006. 4 212412
2006. 5 218097
2006. 6 263900
2006. 7 281457
2006. 8 254405
2006. 9 209985
2007. 1 25
[root@master ~]# s

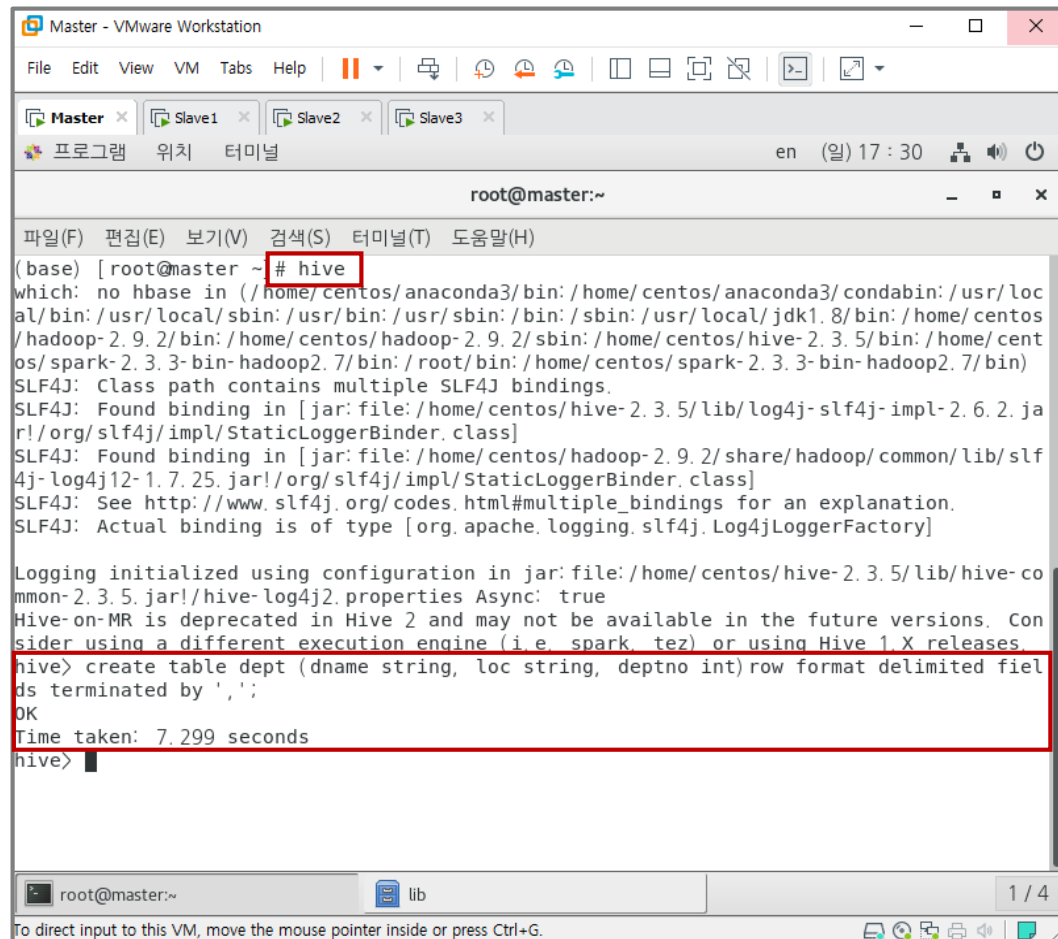
```

@ 작업결과 확인

```
hdfs dfs -cat
delay_count/
part-r-00000
```

2006년 항공 지연
횟수를 월별로 출력

Hive Setup



```
Master - VMware Workstation
File Edit View VM Tabs Help
Master x Slave1 x Slave2 x Slave3 x
프로그램 위치 터미널
en (일) 17:30
root@master:~

파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)

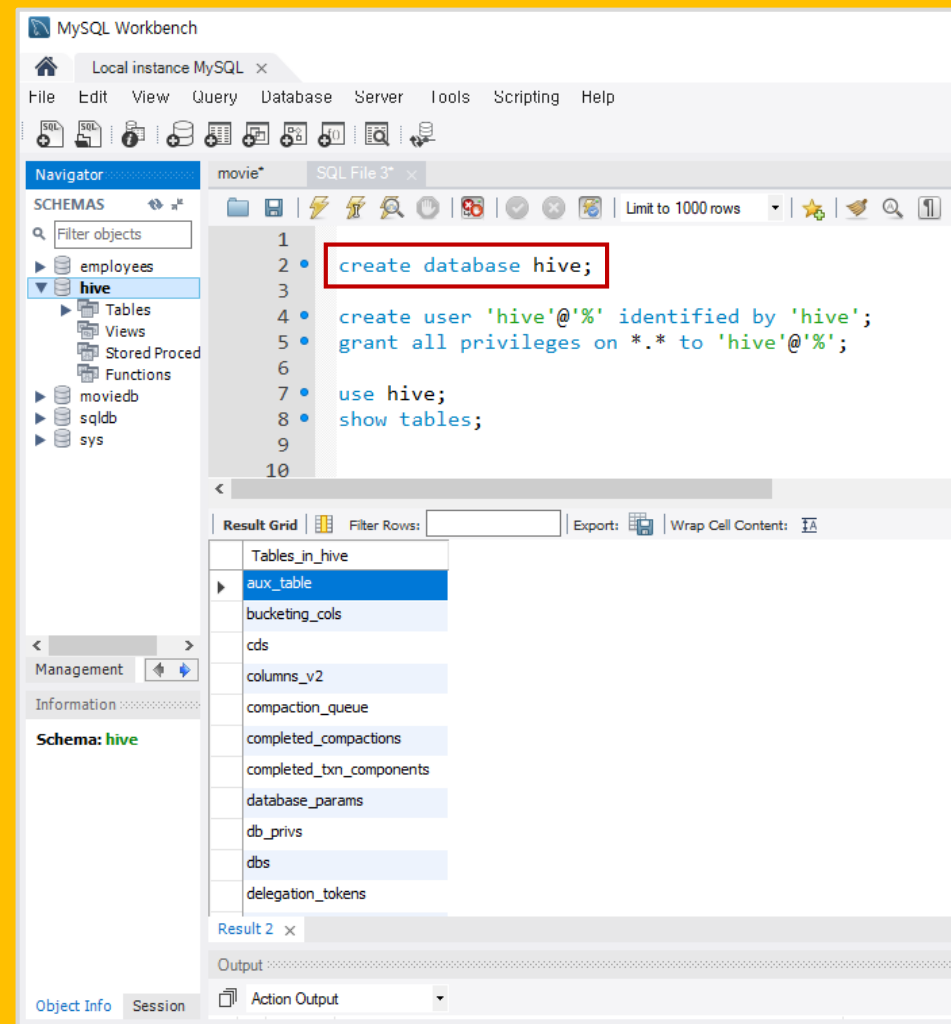
(base) [root@master ~]# hive
which: no hbase in (/home/centos/anaconda3/bin:/home/centos/anaconda3/condabin:/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin:/usr/local/jdk1.8/bin:/home/centos/hadoop-2.9.2/bin:/home/centos/hadoop-2.9.2/sbin:/home/centos/hive-2.3.5/bin:/home/centos/spark-2.3.3-bin-hadoop2.7/bin:/root/bin:/home/centos/spark-2.3.3-bin-hadoop2.7/bin)
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/centos/hive-2.3.5/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/centos/hadoop-2.9.2/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/home/centos/hive-2.3.5/lib/hive-common-2.3.5.jar!/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
hive> create table dept (dname string, loc string, deptno int)row format delimited fields terminated by ',';
OK
Time taken: 7.299 seconds
hive>
```

- 윈도우의 MySQL과 연동
- 윈도우의 MySQL을 Java에서 사용할 수 있도록 조정

About Hive

DB, Table의 형태로 HDFS에 저장된 데이터 구조를 정의
SQL과 유사한 HiveQL 쿼리를 사용



```
MySQL Workbench
Local instance MySQL x
File Edit View Query Database Server Tools Scripting Help

Navigator
SCHEMAS
Filter objects
employees
hive
  Tables
  Views
  Stored Procs
  Functions
  moviedb
  sqldb
  sys

movie* SQL File 3* x
1
2 • create database hive;
3
4 • create user 'hive'@'%' identified by 'hive';
5 • grant all privileges on *.* to 'hive'@'%';
6
7 • use hive;
8 • show tables;
9
10

Result Grid | Filter Rows: | Export: | Wrap Cell Content:
Tables_in_hive
  aux_table
  bucketing_cols
  cds
  columns_v2
  compaction_queue
  completed_compactions
  completed_txn_components
  database_params
  db_privs
  dbs
  delegation_tokens

Result 2 x
Output
Action Output
```

< Hive 간단 동작 테스트 >

```

root@master:~
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
7934 MILLER CLERK 7782 1982-01-23 1300 NULL 10
9292 JACK CLERK 7782 1982-01-23 3200 NULL 70
Time taken: 1.809 seconds, Fetched: 16 row(s)
hive> set hive.cli.print.header=true;
hive> select * from emp;
OK
emp.empno emp.ename emp.job emp.mgr emp.hiredate emp.sal emp.comm e
mp.deptno
NULL ename job NULL hiredate NULL NULL NULL
7369 SMITH CLERK 7902 1980-12-17 800 NULL 20
7499 ALLEN SALESMAN 7698 1981-02-20 1600 300 30
7521 WARD SALESMAN 7698 1981-02-22 1250 500 30
7566 JONES MANAGER 7839 1981-04-02 2975 NULL 20
7654 MARTIN SALESMAN 7698 1981-09-28 1250 1400 30
7698 BLAKE MANAGER 7839 1981-05-01 2850 NULL 30
7782 CLARK MANAGER 7839 1981-06-09 2450 NULL 10
7788 SCOTT ANALYST 7566 1987-04-19 3000 NULL 20
7839 KING PRESIDENT NULL 1981-11-17 5000 NULL 10
7844 TURNER SALESMAN 7698 1981-09-08 1500 0 30
7876 ADAMS CLERK 7788 1987-05-23 1100 NULL 20
7900 JAMES CLERK 7698 1981-12-03 950 NULL 30
7902 FORD ANALYST 7566 1981-12-03 3000 NULL 20
7934 MILLER CLERK 7782 1982-01-23 1300 NULL 10
9292 JACK CLERK 7782 1982-01-23 3200 NULL 70
Time taken: 0.363 seconds, Fetched: 16 row(s)
hive>

```

* master에서 실행 *

@hdfs 디렉토리 생성 & 권한부여

```
hdfs dfs -mkdir /user/hive/warehouse
```

```
hdfs dfs -chmod g+x /user/hive
```

```
hdfs dfs -chmod g+x /user/hive/warehouse
```

@hive 실행

```
hive
```

@테이블 만들기

```
create table dept (dname string, loc string,
                    deptno int)
```

```
row format delimited ; -> 행 포맷을 지정
```

```
fields terminated by ',' ; -> 필드 구분자는 ,(콤마)로
```

@데이터 파일불러오기

```
(inpath 경로의 .csv파일을 읽어서 dept테이블로 읽어오기)
```

```
Load data local inpath
```

```
'/home/centos/data/dept.csv' overwrite into table
dept;
```

@테이블 확인

```
select * from dept;
```

< HiveQL을 사용한 HDFS >

```

Master - VMware Workstation
File Edit View VM Tabs Help
Master x Slave1 x Slave2 x Slave3 x
프로그램 위치 터미널 en (일) 18:07
root@master:~

파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1564907020126_0004, Tracking URL = http://master:8088/proxy/application_1564907020126_0004/
Kill Command = /home/centos/hadoop-2.9.2/bin/hadoop job -kill job_1564907020126_0004
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2019-08-04 18:04:40,825 Stage-2 map = 0% reduce = 0%
2019-08-04 18:04:52,756 Stage-2 map = 100% reduce = 0% Cumulative CPU 1.43 sec
2019-08-04 18:05:00,733 Stage-2 map = 100% reduce = 100% Cumulative CPU 3.13 sec
MapReduce Total cumulative CPU time: 3 seconds 130 msec
Ended Job = job_1564907020126_0004
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 3.59 sec HDFS Read: 9559 HDFS Write: 222 SUCCESS
Stage-Stage-2: Map: 1 Reduce: 1 Cumulative CPU: 3.13 sec HDFS Read: 6215 HDFS Write: 227 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 720 msec
OK
deptno cnt avg_sal
NULL 1 NULL
10 3 2916.6666666666665
20 5 2175.0
30 6 1566.6666666666667
70 1 3200.0
Time taken: 84.226 seconds, Fetched: 5 row(s)
hive>

select deptno, count(*) cnt,
avg(sal) avg_sal from emp
group by deptno order by
deptno; => 의 결과

```

* master에서 실행 *

@ 테이블 조인

```
select d.deptno, d.dname, e.ename, e.sal from emp e,
dept d where e.deptno=d.deptno;
```

@ 그룹화

```
select deptno, count(*) cnt, avg(sal) avg_sal from
emp group by deptno order by deptno;
```

@ 실행결과 저장 (Table에)

```
create table airport_copy(airport string ... ..)
```

```
insert overwrite table airport_copy select * from
airport_code
```

```
select * from airport_copy
```

@ 실행결과 저장 (HDFS에)

```
insert overwrite directory '/tmp/airport_result'
select * from airport_code;
```

```
hdfs dfs -ls /tmp/airport_result
```

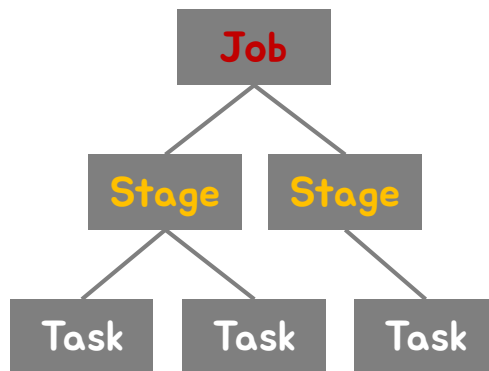
```
hdfs dfs -cat /tmp/airport_result/000000_0
```


About Spark



- 빠른 처리 : 인메모리 기반
- 다양한 언어지원 : Java, Scala, R, Python
- 다양한 컴포넌트 지원 : SQL, Streaming, 머신러닝 등
- 다양한 클러스터에서 동작 : Yarn, Mesos, Kubernetes
- 다양한 파일포맷 지원 : Hdfs, Casandra, HBase

인메모리 : 작업의 중간결과를 메모리에 저장하여 반복작업의 처리효율을 높일 수 있다.



'스파크 컴포넌트 구성'

Spark Library

- Spark SQL : SQL을 이용하여 작업을 생성하고 처리
- Spark Streaming : 실시간 데이터 스트림을 처리 (스트림 데이터를 작은 사이즈로 쪼개어 RDD처럼 처리)
- MLlib : 스파크 기반의 머신러닝 기능을 제공하는 컴포넌트
- GraphX : 분산형 그래프 프로세싱이 가능하게 해줌

Spark Core

- 메인 컴포넌트로 작업의 스케줄링, 메모리관리, 장애복구와 같은 기본적인 기능을 제공
- RDD, DataSet, DataFrame을 이용한 연산처리 수행

Cluster Manager

- 스파크 작업을 운영하는 클러스터 관리자
- 스파크에서 제공하는 Standalone 관리자 이용가능 (Mesos, Yarn, Kubernetes 등의 관리자도 지원)

About Spark

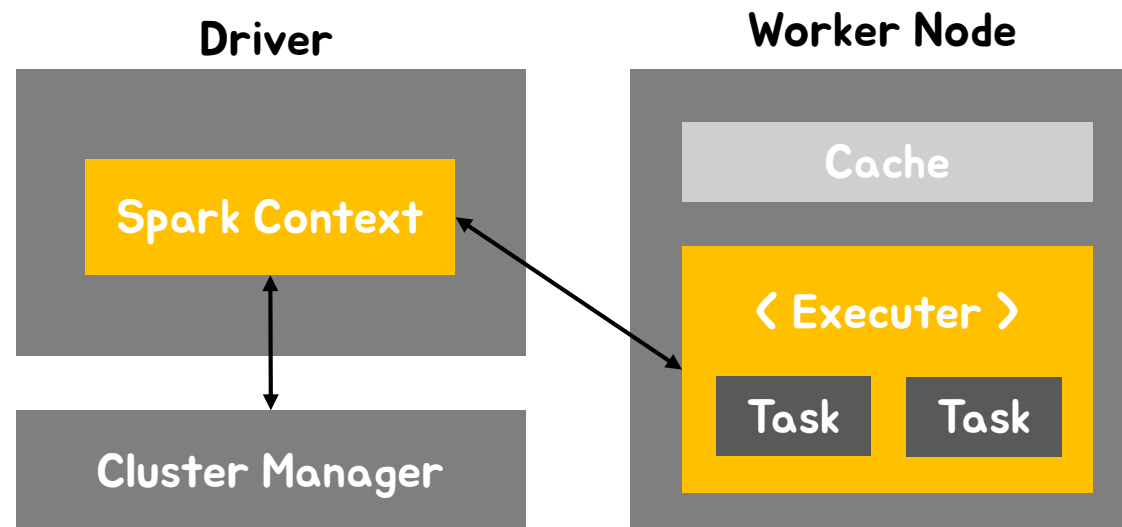
< Spark Job >

- Job : 스파크 애플리케이션으로 제출된 작업
- Stage : Job을 작업단위에 따라 구분한 것
- Task : 익스큐터에서 실행한 실제 작업

< Spark Application >

- 스파크 실행 프로그램
- Driver와 Executor로 실행됨
- Cluster Manager가 Spark Application의 리소스를 효율적으로 배분

< Spark 구조 >

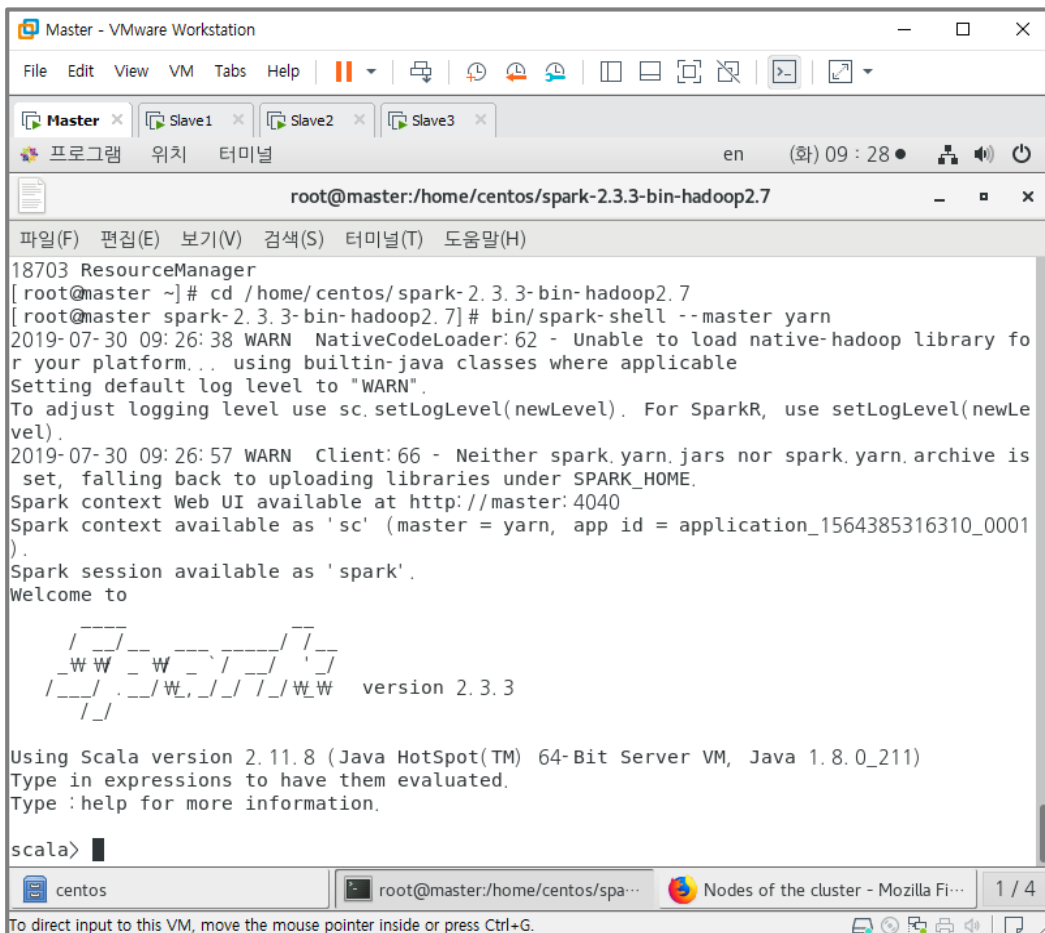


1. Driver는 Spark Context 객체를 생성하여 Cluster Manager와 통신하면서 클러스터 자원관리를 지원한다.
2. Spark Context는 WorkerNode에서 Executor를 실행하여 실제 작업인 Task를 처리하고, Application의 리사이클을 관리한다.

Driver : Spark Application을 실행하는 프로세스

Executor : 실제 작업을 진행하는 프로세서

Spark Setup



```
root@master:/home/centos/spark-2.3.3-bin-hadoop2.7
18703 ResourceManager
[root@master ~]# cd /home/centos/spark-2.3.3-bin-hadoop2.7
[root@master spark-2.3.3-bin-hadoop2.7]# bin/spark-shell --master yarn
2019-07-30 09:26:38 WARN NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
2019-07-30 09:26:57 WARN Client:66 - Neither spark.yarn.jars nor spark.yarn.archive is set, falling back to uploading libraries under SPARK_HOME.
Spark context Web UI available at http://master:4040
Spark context available as 'sc' (master = yarn, app id = application_1564385316310_0001).
Spark session available as 'spark'.
Welcome to

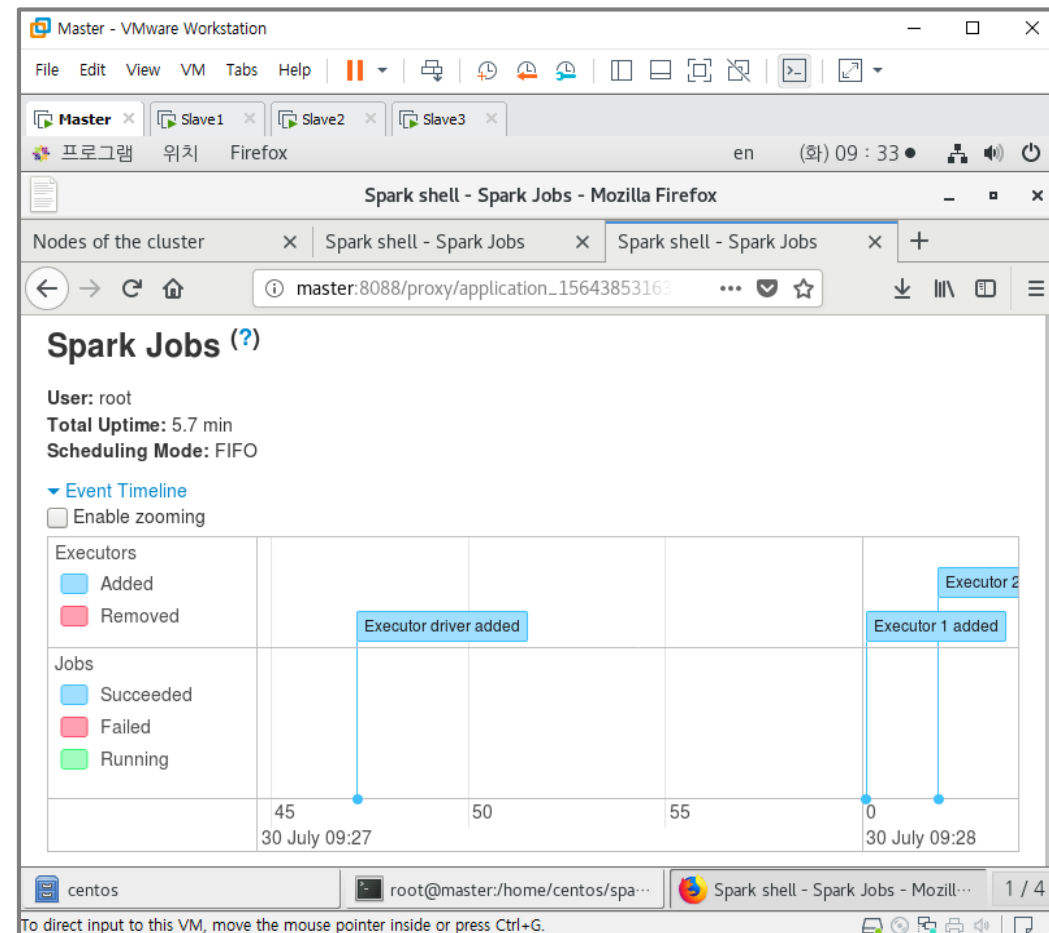
  ____      _
 / ___|  _ \| | | |
 \___ \_||_| | |_| |
  ___) | | | | |_| |
 |____|_|_|_|_|_|_|

 version 2.3.3

Using Scala version 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_211)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

bin/spark-shell --master yarn
-> yarn 환경에서 spark 실행



http://master:4040
-> spark의 정상작동 확인

< Spark 간단 동작 테스트 >

```

Master - VMware Workstation
File Edit View VM Tabs Help
Master x Slave1 x Slave2 x Slave3 x
프로그램 위치 터미널 en (화) 09 : 37
root@master:/home/centos/spark-2.3.3-bin-hadoop2.7

파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
Spark session available as 'spark'.
Welcome to

  _ _ _ _ _
 _W W _ _ _ W _ _ _ _ _
|_ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ |
|_ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ | _ _ _ |
  _ _ _ _ _ version 2.3.3

Using Scala version 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_211)
Type in expressions to have them evaluated.
Type :help for more information.

scala> val hFile = sc.textFile("hdfs://master:9000/input/README.txt")
hFile: org.apache.spark.rdd.RDD[String] = hdfs://master:9000/input/README.txt MapPartit
ionsRDD[1] at textFile at <console>:24

scala> val wc = hFile.flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey
ey(_ + _)
wc: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey at <console
>:25

scala> wc.take(5)
res0: Array[(String, Int)] = Array((under,1), (this,3), (distribution,2), (Technology,1
), (country,1))

scala>
  
```

* master에서 실행 *

@ 테스트 파일을 HDFS에 업로드

```
hdfs dfs -mkdir /input
```

```
hdfs dfs -copyFromLocal /home/centos/hadoop-2.9.2/README.txt /input
```

@ 프로그램 실행하기

```
Start-dfs.sh
```

```
Start-yarn.sh
```

```
Spark-shell --master yarn
```

@ 스파크에서 작업

```
spark> val hFile =
```

```
sc.textFile("hdfs://master:9000/input/README.txt")
```

```
spark> val wc = hFile.flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey(_ + _)
```

```
spark> wc.take(5)
```

@정상출력 확인 -> 다음과 같이 출력되면 정상 실행된 것

```
Array[(String, Int)] = Array((under,1), (this,3), (distribution,2), (Technology,1), (country,1))
```

< Python을 사용한 Spark >

```

Master - VMware Workstation
File Edit View VM Tabs Help
Master x Slave1 x Slave2 x Slave3 x
프로그램 위치 터미널 en (화) 09:43
root@master:/home/centos/spark-2.3.3-bin-hadoop2.7
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
5 tasks are for partitions Vector(0))
2019-07-30 09:41:06 INFO TaskSchedulerImpl: 54 - Adding task set 1.0 with 1 tasks
2019-07-30 09:41:06 INFO TaskSetManager: 54 - Starting task 0.0 in stage 1.0 (TID 1, lo
calhost, executor driver, partition 0, PROCESS_LOCAL, 7882 bytes)
2019-07-30 09:41:06 INFO Executor: 54 - Running task 0.0 in stage 1.0 (TID 1)
2019-07-30 09:41:06 INFO BlockManager: 54 - Found block rdd_1_0 locally
2019-07-30 09:41:06 INFO PythonRunner: 54 - Times: total = 74, boot = -142, init = 215,
finish = 1
2019-07-30 09:41:06 INFO Executor: 54 - Finished task 0.0 in stage 1.0 (TID 1). 1504 by
tes result sent to driver
2019-07-30 09:41:06 INFO TaskSetManager: 54 - Finished task 0.0 in stage 1.0 (TID 1) in
95 ms on localhost (executor driver) (1/1)
2019-07-30 09:41:06 INFO TaskSchedulerImpl: 54 - Removed TaskSet 1.0, whose tasks have
all completed, from pool
2019-07-30 09:41:06 INFO DAGScheduler: 54 - ResultStage 1 (count at /home/centos/spark-
2.3.3-bin-hadoop2.7/test.py:9) finished in 0.107 s
2019-07-30 09:41:06 INFO DAGScheduler: 54 - Job 1 finished: count at /home/centos/spark
-2.3.3-bin-hadoop2.7/test.py:9, took 0.114446 s
lines with a: 61, lines with b: 30
2019-07-30 09:41:06 INFO SparkContext: 54 - Invoking stop() from shutdown hook
2019-07-30 09:41:06 INFO AbstractConnector: 318 - Stopped Spark@e2035f1{HTTP/1.1,[http
/1.1]}{0.0.0.0:4040}
2019-07-30 09:41:06 INFO SparkUI: 54 - Stopped Spark web UI at http://master:4040
2019-07-30 09:41:06 INFO ContextCleaner: 54 - Cleaned accumulator 42
2019-07-30 09:41:06 INFO MapOutputTrackerMasterEndpoint: 54 - MapOutputTrackerMasterEnd
point stopped!
centos root@master:/home/centos/spa... Spark shell - Spark Jobs - Mozill... 1 / 4
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

```

* master에서 실행 *

@ python code 작성

\$ vim test.py

```
from pyspark import SparkContext
```

```
logFile = "/sample/README.md"
//hdfs에 올려둔 파일사용
(hdfs://master:9000/sample/README.md)
```

```
sc = SparkContext("local","Simple App")
logData = sc.textFile(logFile).cache()
```

```
numA = logData.filter(lambda s: 'a' in s).count()
numB = logData.filter(lambda s: 'b' in s).count()
```

```
print("lines with a: %i, lines with
b:%i" %(numA,numB))
```

@ 작성된 python code 를 실행

\$ bin/spark-submit --master local[4] test.py



< 감사합니다 >

Hadoop, Hsive, Spark 프로그램의 설치방법은
저의 GitHub에 올려두었으니 필요하신 분은 참고하시길 바랍니다.

GitHub주소 : <https://github.com/DabinJeon/HADOOP>

