Hadoop Eco System with IRIS

Mobigen, November 20, 2015

목차

1 지원 정보

1.1 문서 버전

0.1

1.2 IRIS 버전

1.5.1

1.3 사용 문의 및 기술 정보

(주) 모비젠

본사

- 135-280 서울 강남구 대치동 967-3번지 KM빌딩 2층, (주) 모비젠
- T : 02 - 538 - 9360
- F : 02 - 538 - 9369

기술 연구소

- 135-280 서울 강남구 대치동 967-3번지 KM빌딩 5층, (주) 모비젠
- T : 02 - 538 - 9364
- F : 02 - 538 - 9368

모비젠 IRIS 기술 지원

- T : 02 - 538 - 9364
- M : iris@mobigen.com

2 IRIS with Hadoop Eco System

본 항목에서는 Hadoop Eco System 에서 어떻게 IRIS의 데이터를 이용할 수 있는지에 대해서 설명합니다.

2.1 Supported Hadoop Eco System

IRIS에서는 아래 플랫폼과의 연결을 지원하고 있습니다.

- Hadoop MapReduce
- Hive
- Spark
- Tajo

Tajo를 제외한 세가지 플랫폼은 IRIS에서 제공하는 API를 이용하여 쉽게 접근이 가능하며 Tajo는 Tajo 소스코드를 수정하여 제공하고 있습니다.

2.2 Hadoop MapReduce with IRIS

본 항목에서는 Hadoop MapReduce 에서 IRIS 데이터를 불러오기 위한 방법을 설명합니다.

2.2.1 Hadoop MapReduce의 한계점

Hadoop MapReduce와 IRIS 연결에는 다음과 같은 한계점이 있습니다.

- MapReduce 로 가공된 데이터를 IRIS에 넣지 못합니다.
- IRIS와 Hadoop은 동일한 계정으로 설치되어야 합니다.

2.2.2 Hadoop의 설치

본 항목에서는 Hadoop 설치에 대해서 다룹니다. 먼저 IRIS는 설치 되어 있다는 가정으로 진행합니다. IRIS 설치에 관한 자세한 내용은 IRIS 설치 관련 문서를 참조하세요.

```
# IRIS의 설치 위치
/home/iris/IRIS

# Hadoop이 설치될 위치
/home/iris/tools/hadoop
```

먼저 IRIS는 iris 계정 아래의 IRIS 폴더에 설치되어 있는 것을 확인할 수 있습니다. Hadoop은 iris 계정 아래의 tools 폴더 아래 hadoop 폴더에 설치 될 것입니다.

2.2.2.1 JDK 설치 Hadoop을 동작 시키기 위해서는 JDK 설치가 필요합니다. 본 문서에서는 oracle JDK1.7을 이용하기로 하였습니다. 오라클 사이트로 이동하여 해당 파일을 다운로드 합니다. IRIS에서는 Redhat 기반의 운영체제를 사용하니 rpm 파일을 다운로드 받아 설치를 진행합니다.

```
[iris@all-node ~]$ rpm -Uvh jdk-[JAVA_VERSION]-linux-64.rpm
.
.
.
[iris@all-node ~]$
```

JDK는 모든 노드에서 설치를 진행합니다. 설치가 정상적으로 되었는지 확인을 위해 다음 명령어를 사용합니다.

```
[iris@all-node ~]$ java -version
java version "[JAVA_VERSION]"
Java(TM) SE Runtime Environment (build [JAVA_VERSION]-b15)
Java HotSpot(TM) 64-Bit Server VM (build 24.79-b02, mixed mode)
[iris@all-node ~]$
```

2.2.2.2 Hadoop 설치하기 JDK 설치 후에 Hadoop 설치 하기 위해 Hadoop 을 다운로드 합니다. 본 문서에서는 Hadoop 2.7.1을 기준으로 합니다. 먼저 Hadoop을 다운 받기 위해 아래와 같은 명령어를 사용합니다.

```
[iris@all-node ~]$ mkdir tools
[iris@all-node ~]$ cd tools
[iris@all-node tools]$ wget http://mirror.apache-kr.org/hadoop/common/hadoop-2.7.1/hadoop-2.7.1.tar.gz
.
.
.
[iris@all-node tools]$ ls
hadoop-2.7.1.tar.gz
[iris@all-node tools]$ tar xzvf hadoop-2.7.1.tar.gz
.
.
.
[iris@all-node tools]$ ls
hadoop-2.7.1  hadoop-2.7.1.tar.gz
[iris@all-node tools]$ ln -s hadoop-2.7.1 hadoop
[iris@all-node tools]$ rm hadoop-2.7.1.tar.gz
[iris@all-node tools]$ mkdir -p data/namenode
[iris@all-node tools]$ mkdir -p data/tmp
[iris@all-node tools]$ mkdir -p data/datanode
hadoop  hadoop-2.7.1  data
[iris@all-node tools]$
```

위의 과정을 거치면 hadoop이 /home/iris/tools/hadoop 에 위치하게 됩니다.

2.2.2.3 Hadoop 설정하기 hadoop 설정을 위해서는 몇개의 설정 파일을 수정해야 합니다.

```
* 주의 사항
아래와 설정이 동일할 필요는 없습니다. [] 안에 있는 내용은 환경에 맞게 입력해야 합니다.
```

아래 예제를 따라 Hadoop을 설정합니다. 먼저 root 권한이 필요한 파일입니다.

```
#### /etc/hosts
# 위에 부분에 존재하는 localhost 관련된 항목은 삭제합니다.
[MASTER_IP_ADDRESS]     [MASTER_HOST_NAME]
[SLAVE_1_IP_ADDRESS]        [SLAVE_1_HOST_NAME]
[SLAVE_2_IP_ADDRESS]        [SLAVE_2_HOST_NAME]
...
```

다음은 iris 계정으로 편집합니다.

```
#### /home/iris/IRIS/env.sh
... # 최하단에 작성
export HADOOP_HOME=/home/iris/tools/hadoop
export JAVA_HOME=/usr/java/jdk_[JAVA_VERSION]
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH

#### /home/iris/tools/hadoop/etc/hadoop/hadoop-env.sh
...
export JAVA_HOME=/usr/java/jdk_[JAVA_VERSION]
export HADOOP_HOME=/home/iris/tools/hadoop
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export YARN_CONF_DIR=$HADOOP_HOME/etc/hadoop    ...

#### /home/iris/tools/hadoop/etc/hadoop/yarn-env.sh
```

```
...
if [ "$JAVA_HOME" != "" ]; then
#echo "run java in $JAVA_HOME"
  export JAVA_HOME=/usr/java/jdk_[JAVA_VERSION]
fi
export HADOOP_HOME=/home/iris/tools/hadoop
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export YARN_CONF_DIR=$HADOOP_HOME/etc/hadoop
...

#### /home/iris/tools/hadoop/etc/hadoop/core-site.xml
...
<configuration>
        <property>
                <name>fs.defaultFS</name>
                <value>hdfs://[MASTER_NODE_HOST_NAME]:9000</value>
        </property>
        <property>
                <name>hadoop.tmp.dir</name>
                <value>/home/iris/tools/data/tmp</value>
        </property>
</configuration>
...

#### /home/iris/tools/hadoop/etc/hadoop/hdfs-site.xml
...
<configuration>
        <property>
                <name>dfs.namenode.name.dir</name>
                <value>/home/iris/tools/data/namenode</value>
        </property>
        <property>
                <name>dfs.datanode.data.dir</name>
                <value>/home/iris/tools/data/datanode</value>
        </property>
</configuration>
...
```

```
#### /home/iris/tools/hadoop/etc/hadoop/mapred-site.xml
<configuration>
        <property>
                <name>mapreduce.framework.name</name>
                <value>yarn</value>
        </property>
</configuration>

#### /home/iris/tools/hadoop/etc/hadoop/yarn-site.xml
<configuration>
        <property>
                <name>yarn.nodemanager.aux-services</name>
                <value>mapreduce_shuffle</value>
        </property>
        <property>
                <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
                <value>org.apache.hadoop.mapred.ShuffleHandler</value>
        </property>
        <property>
                <name>yarn.resourcemanager.resource-tracker.address</name>
                <value>[MASTER_NODE_HOST_NAME]:8025</value>
        </property>
        <property>
                <name>yarn.resourcemanager.scheduler.address</name>
                <value>[MASTER_NODE_HOST_NAME]:8030</value>
        </property>
        <property>
                <name>yarn.resourcemanager.address</name>
                <value>[MASTER_NODE_HOST_NAME]:8040</value>
        </property>
</configuration>

#### /home/iris/tools/hadoop/etc/hadoop/slaves
[슬레이브 노드 호스트 네임]
[슬레이브 노드 호스트 네임]
[슬레이브 노드 호스트 네임]
...
```

2.2.2.4 SSH Key 분배하기 Hadoop을 쉽게 이용하기 위해서는 ssh key를 미리 분배하여 상호 접속시 비밀번호 입력을 생략할 수 있도록 하는것이 좋습니다. 먼저 각각의 노드에서 ssh 키를 생성합니다. 아래 명령어를 입력하고 입력 창이 나오면 모두 엔터를 입력합니다.

```
[iris@all-node ~]$ ssh-keygen -t rsa
...
[iris@all-node ~]$
```

위의 명령어를 모든 노드에 실행 후 키가 다 만들어 졌다면 아래 과정을 마스터에서 실행합니다. 마지막에 Key 폴더는 삭제합니다.

```
[iris@master ~]$ mkdir keys
[iris@master ~]$ cd keys
[iris@master keys]$ scp [master_ip_address]:~/.ssh/id_rsa.pub ./master
[iris@master keys]$ scp [slave_1_ip_address]:~/.ssh/id_rsa.pub ./slave_1
#### 모든 슬레이브 노드에서 받아올 때 까지 반복
...
[iris@master keys]$ ls
master  slave_1  slave_2  slave_3 ......
[iris@master keys]$ cat * > ~/.ssh/authorized_keys
[iris@master keys]$ cd ~/.ssh
[iris@master .ssh]$ scp [slave_1_ip_address]:`pwd`
#### 모든 슬레이브 노드에 배포할 때 까지 반복
...
[iris@master .ssh]$ cd
[iris@master ~]$ rm -Rf keys
```

다음 명령어를 모든 노드에서 실행합니다.

```
[iris@all-node ~]$ chmod 600 ~/.ssh/authorized_keys
```

위의 명령을 실행한 후 부터는 모든 노드 간에 iris 계정 로그인은 패스워드 없이 진행됩니다.

2.2.2.5 Hadoop 처음 실행하기 설정을 모두 완료 하였다면 iris 계정을 로그 아웃했다가 다시 로그인합니다. 이후 hadoop을 동작 시키기 위해서 먼저 HDFS 파일 시스템을 포맷하는 과정을 수행합니다.

```
[iris@master ~]$ hadoop namenode -format
...
```

다음은 hadoop을 동작 시킬 차례입니다.

```
[iris@master ~]$ start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
...
[iris@master ~]$
```

2.2.2.6 Hadoop 정상 동작 확인하기 간단한 예제를 통해서 Hadoop이 정상 동작하는지 확인을 하도록 하겠습니다. 먼저 예제 프로그램을 만들 폴더를 생성합니다.

```
[iris@master ~]$ mkdir -p example/wordcount
[iris@master ~]$ cd example/wordcount
[iris@master wordcount]$
```

해당 디렉토리에 아래 파일을 WordCount.java 로 생성합니다. 아래 파일은 Hadoop 문서에서 Tutorial에 나와 있는 코드입니다.

```
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;

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 {
      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();
    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);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

해당 파일을 생성한 후에 다음과 같은 명령어를 입력하여 컴파일 합니다.

```
[iris@master wordcount]$ export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar
[iris@master wordcount]$ hadoop com.sun.tools.javac.Main WordCount.java
[iris@master wordcount]$ jar cf wc.jar WordCount*.class
[iris@master wordcount]$ ls *.jar
wc.jar
[iris@master wordcount]$
```

컴파일 완료 후 wc.jar 파일이 생성된 것을 확인할 수 있습니다. 다음은 예제 파일 생성입니다. 다음 내용을 파일 이름 example.txt 로 저장합니다.

```
hello world
hello hadoop
hello iris
iris with hadoop eco system
good job
```

해당 파일을 HDFS로 올리기 위해 HDFS에 테스트 폴더를 생성하고 업로드 합니다.

```
[iris@master wordcount]$ hadoop dfs -mkdir /test
[iris@master wordcount]$ hadoop dfs -mkdir /test/input
[iris@master wordcount]$ hadoop dfs -copyFromLocal example.txt /test/input
[iris@master wordcount]$ hadoop dfs -ls /test/input
-rw-r--r--   3 iris supergroup         73 ****-**-** **:** /test/input/example.txt
[iris@master wordcount]$
```

데이터가 업로드 된 것을 확인하고 다음과 같이 실행합니다.

```
[iris@master wordcount]$ hadoop jar wc.jar WordCount /test/input /test/output
15/07/09 08:25:47 INFO client.RMProxy: Connecting to ResourceManager at HMaster1/192...
15/07/09 08:25:48 WARN mapreduce.JobResourceUploader: Hadoop command-line option ...
interface and execute your application with ToolRunner to remedy this.
15/07/09 08:25:49 INFO input.FileInputFormat: Total input paths to process : 1
15/07/09 08:25:49 INFO mapreduce.JobSubmitter: number of splits:1
15/07/09 08:25:50 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_...
15/07/09 08:25:51 INFO impl.YarnClientImpl: Submitted application application_...
15/07/09 08:25:51 INFO mapreduce.Job: The url to track the job: http://HMaster1:8088/...
15/07/09 08:25:51 INFO mapreduce.Job: Running job: job_1436430120941_0001
15/07/09 08:26:06 INFO mapreduce.Job: Job job_1436430120941_0001 running in ...
15/07/09 08:26:06 INFO mapreduce.Job:  map 0% reduce 0%
15/07/09 08:26:18 INFO mapreduce.Job:  map 100% reduce 0%
15/07/09 08:26:33 INFO mapreduce.Job:  map 100% reduce 100%
...
File Output Format Counters
    Bytes Written=67
[iris@master wordcount]$
```

MapReduce 잡이 종료한 후에 결과 파일이 생성되었는지 확인합니다.

```
[iris@master wordcount]$ hadoop dfs -ls /test/output    Found 2 items
-rw-r--r--   3 iris supergroup          0 ****-**-** **:** /test/output/_SUCCESS
-rw-r--r--   3 iris supergroup         67 ****-**-** **:** /test/output/part-r-00000
[iris@master wordcount]$ hadoop dfs -cat /test/output/part-r-00000
eco 1
good    1
```

```
hadoop  2
hello   3
iris    2
job 1
system  1
with    1
world   1
[iris@master wordcount]$
```

정상적으로 결과가 출력된것을 확인할 수 있습니다. 이로서 Hadoop 설치가 종료 되었습니다.

2.2.3 Hadoop-IRIS API 설치

Hadoop의 설치가 정상적으로 종료되고 나면 IRIS-Hadoop간의 연결을 위한 API가 필요합니다. 해당 API는 IRIS 기술 지원을 통해서 요청 할 수 있습니다.

```
mailto : iris@mobigen.com
subject : [Hadoop-IRIS API] Hadoop-IRIS API 요청
```

위의 형태로 메일을 보내주시면 아래 형식에 해당하는 파일을 보내드립니다.

```
hadoop-iris-[VERSION].jar
sqlite3-jdbc-[VERSION].jar
```

VERSION 부분에는 해당 API 버전을 표기합니다. 해당 파일을 다운로드 받아서 해당 패스에 복사를 합니다.

```
[iris@all-node ~]$ cp *.jar ~/tools/hadoop/share/hadoop/mapreduce/
```

그 후 hadoop을 재시작합니다.

```
[iris@master ~]$ stop-all.sh
...
[iris@master ~]$ start-all.sh
...
```

2.2.4 IRIS Record WordCount

본 항목에서는 Hadoop WordCount 프로그램에서 IRIS 데이터를 쓸 수 있는 프로그램을 테스트용 프로그램을 작성합니다. 이 과정은 Hadoop-IRIS API가 설치되어 있는 상태에서 동작합니다.

2.2.4.1 테스트용 테이블 생성 및 데이터 입력 먼저 Hadoop MapReduce 프로그램을 작성하기 전에 IRIS에 테이블을 생성하고 데이터를 입력합니다. 데이터의 스키마는 다음과 같습니다.

```
create table HADOOP_TEST_TABLE (
    NAME TEXT,
    UPDATE_TIME TEXT,
    TALK TEXT
)
datascope LOCAL
ramexpire 30
diskexpire 28880
partitionkey NAME
partitiondate UPDATE_TIME
partitionrange 10
;
```

위의 쿼리를 iplus를 통해서 실행합니다.

```
[iris@master ~]$ iplus test
Password : test
```

```
Connecting to IRIS(test@192.168.111.10:5050).
Connected to IRIS.
Enter ".help" for instructions
iplus> create table HADOOP_TEST_TABLE (
    >   NAME TEXT,
    >   UPDATE_TIME TEXT,
    >   TALK TEXT
    > )
    > datascope LOCAL
    > ramexpire 30
    > diskexpire 28880
    > partitionkey NAME
    > partitiondate UPDATE_TIME
    > partitionrange 10
    > ;
Ret : +OK Create table

0.4828 sec
iplus>
```

테이블을 생성한 후에 아래 쿼리를 이용하여 데이터를 입력합니다.

```
INSERT INTO HADOOP_TEST_TABLE (NAME, UPDATE_TIME, TALK)
VALUES ('TOM',
    '20150710103000',
    'Woud you mind taking my car down to the garage and changing the oil?');
INSERT INTO HADOOP_TEST_TABLE (NAME, UPDATE_TIME, TALK)
VALUES ('SAM',
        '20150710103200',
        'You are the boss');
```

아래는 실제 실행한 결과 입니다.

```
iplus> INSERT INTO HADOOP_TEST_TABLE (NAME, UPDATE_TIME, TALK)
    > VALUES ('TOM',
    >   '20150710103000',
    >   'Woud you mind taking my car down to the garage and changing the oil?');
```

```
Ret : +OK

0.2683 sec

iplus> INSERT INTO HADOOP_TEST_TABLE (NAME, UPDATE_TIME, TALK)
    > VALUES ('SAM',
    >       '20150710103200',
    >       'You are the boss');
Ret : +OK

0.1351 sec

iplus>
```

데이터가 정상적으로 입력 되었는지 확인합니다.

```
iplus> SELECT * FROM HADOOP_TEST_TABLE;
Ret : +OK Success

 NAME          UPDATE_TIME   TALK
=========================================================
 TOM           20150710103000 Woud you mind taking my car down to the garage and ...
 SAM           20150710103200 You are the boss
=========================================================
2 row in set
0.1002 sec

iplus>
```

2.2.4.2 Hadoop MapReduce 프로그램 작성

데이터 입력이 완료되었다면 프로그램을 작성할 차례입니다. 다음은 기본적인 틀입니다.

```
import java.io.IOException;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
```

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

import com.mobigen.iris.hadoop.IRISConf;
import com.mobigen.iris.hadoop.IRISRow;
import com.mobigen.iris.hadoop.impl.mr.IRISCountMapper;
import com.mobigen.iris.hadoop.impl.mr.IRISCountReducer;
import com.mobigen.iris.hadoop.impl.mr.IRISMapper;
import com.mobigen.iris.hadoop.impl.mr.IRISOutputCollector;
import com.mobigen.iris.hadoop.impl.mr.IRISReduceOutputCollector;
import com.mobigen.iris.hadoop.impl.mr.IRISReducer;
import com.mobigen.iris.hadoop.mapred.IRISInputFormat;


public class IRISTest extends Configured implements Tool {

    public class IRISTestMapper extends
        IRISMapper<Text, IRISRow, Text, LongWritable> {

        @Override
        protected void setup(Configuration conf) {
                }

        @Override
        protected void map(Text key, IRISRow value,
            IRISOutputCollector<Text, LongWritable> output)
            throws IOException {
            /*
            Map 구현 부분입니다.
            */
        }

        @Override
```

```
        public void cleanup(Configuration conf) {
        }
    }
    public class IRISTestReducer extends
        IRISReducer<Text, LongWritable, Text, LongWritable> {

        @Override
        protected void setup(Configuration conf) {
        }

        @Override
        protected void reduce(Text key, Iterable<LongWritable> values,
            IRISReduceOutputCollector<Text, LongWritable> output)
            throws IOException {
            /*
            Reduce 구현 부분입니다.
            */
                }

        @Override
        public void cleanup(Configuration conf) {
        }
    }

    private static final Log LOG = LogFactory.getLog(IRISTest.class);

    public int run(String[] args) throws Exception {
        if (args.length != 1) {
            System.err.println("Usage: ${command} <out>");
            System.exit(2);
        }
        Configuration conf = getConf();
        LOG.info("configuration");
        try {
            LOG.info("-base query : " + conf.get(IRISConf.BASE_QUERY));
        } catch (Throwable e) {
            LOG.error("Unsufficient arguments");
            LOG.error(e);
            System.exit(2);
        }
        LOG.info("starting");
```

```
        JobConf job = (JobConf) conf;
        job.setJobName("IRIS Test Code");
        job.setJarByClass(IRISTest.class);
        job.setMapperClass(IRISTestMapper.class);
        job.setReducerClass(IRISTestReducer.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(LongWritable.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(LongWritable.class);
        job.setInputFormat(IRISInputFormat.class);
        FileOutputFormat.setOutputPath(job, new Path(args[0]));
        JobClient.runJob(job);
        LOG.info("done");
        return 0;
    }

    public static void main(String[] args) throws Exception {
        int res = ToolRunner.run(new JobConf(), new IRISTest(), args);
        System.exit(res);
    }
}
```

간단한 구조입니다. 구현해야 하는 부분은 3부분입니다. Mapper, Reducer 그리고 Main 부분을 구현하면 됩니다.

2.2.4.2.1 Mapper 구현

먼저 구현할 부분은 Mapper 부분입니다.

```
public class IRISTestMapper extends
    IRISMapper<Text, IRISRow, Text, LongWritable> {
    // 1을 계속해서 써야 하므로 미리 선언해둡니다.
    private LongWritable one = new LongWritable(1);

    @Override
    protected void setup(Configuration conf) {
            }

    @Override
```

```
    protected void map(Text key, IRISRow value,
        IRISOutputCollector<Text, LongWritable> output)
        throws IOException {

        // IRIS 데이터를 읽어오는 부분입니다.
        // row 하나를 가져와서 문자열 배열에 입력합니다.
        String [] data = value.getValues();

        // 읽어온 데이터 배열에서 문자열 하나하나를 가공합니다.
        for(int i=0;i<data.length;i++){

            // 하나의 컬럼에 들어 있는 문자열을 다시 스페이스를 구분자로 쪼갭니다.
            String [] temp_data = data[i].split(" ");

            // 쪼개진 문자열을 map 결과물로 output 합니다.
            for(int j=0;j<temp_data.length;j++){
                Text newKey = new Text(temp_data[j]);
                output.write(newKey, one);
            }
        }
    }

    @Override
    public void cleanup(Configuration conf) {
    }
}
```

가장먼저 보이는 부분은

```
private LongWritable one = new LongWritable(1);
```

입니다. WordCount 프로그램에서 Map 하나의 단어를 형태로 계속 값을 생성하므로 미리 1을 출력할 값을 선언해두었습니다. 그 다음은 map 함수 내부 입니다. IRIS에서 데이터를 읽어오면 하나의 row로 처리를 하게 됩니다. 즉, IRISRow는 하나의 레코드를 의미합니다.

```
String [] data = value.getValues();
```

위의 코드가 사용되었습니다. 이 코드의 경우 하나의 레코드에 있는 모든 필드를 문자열 배열로 리턴합니다. 만약 특정 컬럼을 가져오고 싶다면 아래와 같이 사용하시면 됩니다.

```
String specific_column = value.getValue([int index])
```

index는 0부터 시작하는 정수로 입력이 가능합니다. 문자열 배열 내부의 문자열은 하나의 컬럼입니다. 이를 반복적으로 for문으로 처리합니다.

```
for(int i=0;i<data.length;i++){
...
}
```

즉, 모든 컬럼의 데이터를 반복적으로 처리합니다. 내부에는 아래와 같은 코드가 들어 있습니다.

```
String [] temp_data = data[i].split(" ");
for(int j=0;j<temp_data.length;j++){
    Text newKey = new Text(temp_data[j]);
    output.write(newKey, one);
}
```

하나의 컬럼을 읽어서 이 내용을 다시 단어로 분리합니다. 분리된 결과를 Text 형태로 변경한 후에 output으로 임시 데이터에 쓰게 됩니다.

2.2.4.2.2 Reducer 구현 Mapper 구현이 완료 된 이후 Reducer 부분의 구현을 진행합니다. 아래는 Reducer 코드의 전문입니다.

```
public class IRISTestReducer extends
    IRISReducer<Text, LongWritable, Text, LongWritable> {
```

```
    @Override
    protected void setup(Configuration conf) {
    }

    @Override
    protected void reduce(Text key, Iterable<LongWritable> values,
        IRISReduceOutputCollector<Text, LongWritable> output)
        throws IOException {
        long sum = 0;
        for (LongWritable item:values){
            sum += item.get();
        }
        output.write(key, new LongWritable(sum));
    }

    @Override
    public void cleanup(Configuration conf) {
    }
}
```

Mapper 보다 조금 더 단순한 구조를 가지고 있습니다. 먼저 Mapper에서 임시로 쓰여진 파일은 아래와 같은 형태로 저장됩니다.

```
<단어_1, 1>
<단어_1, 1>
<단어_1, 1>
<단어_2, 1>
<단어_3, 1>
```

위의 데이터는 Reducer로 들어오기전에 가공되어 아래와 같은 형태로 들어오게 됩니다.

```
<단어_1, (1,1,1)>
<단어_2, 1>
<단어_3, 1>
```

즉, Key, ValueList 형태로 Reducer로 들어오게 됩니다. WordCount의 경우 단순히 단어의 수를 카운팅하므로 카운팅할 변수를 하나 선언합니다.

```
long sum = 0
```

다음은 특정 단어에 다수의 값을 더하는 작업입니다.

```
for (LongWritable item:values){
    sum += item.get();
}
```

마지막으로 결과 파일에 값을 쓰면 종료 됩니다.

```
output.write(key, new LongWritable(sum));
```

여기서 key 값은 변하는 것이 없으므로 입력받은 그대로 사용하고 sum값은 LongWritable로 변경하여 기록합니다.

2.2.4.2.3 run 부분 구현하기 run 부분은 설정을 통해 실질적으로 실행될 MapReduce를 정의하는 부분입니다.

```
public int run(String[] args) throws Exception {
    if (args.length != 1) {
        System.err.println("Usage: ${command} <out>");
        System.exit(2);
    }
    Configuration conf = getConf();
    LOG.info("configuration");
    try {
        LOG.info("-base query : " + conf.get(IRISConf.BASE_QUERY));
    } catch (Throwable e) {
        LOG.error("Unsufficient arguments");
```

```
        LOG.error(e);
        System.exit(2);
    }
    LOG.info("starting");
    JobConf job = (JobConf) conf;
    job.setJobName("IRIS Test Code");
    job.setJarByClass(IRISTest.class);
    job.setMapperClass(IRISTestMapper.class);
    job.setReducerClass(IRISTestReducer.class);
    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(LongWritable.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(LongWritable.class);
    job.setInputFormat(IRISInputFormat.class);
    FileOutputFormat.setOutputPath(job, new Path(args[0]));
    JobClient.runJob(job);
    LOG.info("done");
    return 0;
}
```

위의 코드에서 LOG는 전부 LOG를 기록하는 부분입니다. LOG 부분을 제외하면 아래 부분이 남습니다. 각각의 코드에 대한 주석을 참조하시면 됩니다.

```
## 새로운 Config 생성
JobConf job = (JobConf) conf;
## 작업 이름 설정
job.setJobName("IRIS Test Code");
## 작업에 사용될 클래스 설정
job.setJarByClass(IRISTest.class);
## 작업에 사용될 Map, Reduce 클래스 설정
job.setMapperClass(IRISTestMapper.class);
job.setReducerClass(IRISTestReducer.class);
## Map에서 출력되는 데이터 타입
job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(LongWritable.class);
## 최종적으로 출력되는 데이터 타입
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(LongWritable.class);
## 데이터를 읽어올때 사용할 클래스 설정
```

```
job.setInputFormat(IRISInputFormat.class);
## 최종 기록될 파일의 위치
FileOutputFormat.setOutputPath(job, new Path(args[0]));
```

2.2.4.3 완성된 코드 컴파일하기 위에서 완성된 코드를 이용하여 다음과 같이 컴파일 수행합니다.

```
[iris@master ~]$ mkdir -p example/iris-test
[iris@master ~]$ cd example/iris-test
[iris@master iris-test]$ vim IRISTest.java
========== 완성된 코드 입력 ==========
...
[iris@master iris-test]$ export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar
[iris@master iris-test]$ hadoop com.sun.tools.javac.Main IRISTest.java
[iris@master iris-test]$ jar cf IRISTest.jar IRISTest*.class
[iris@master iris-test]$ ls *.jar
IRISTest.jar
[iris@master iris-test]$
```

2.2.4.4 IRIS 설정 파일 생성하기 IRIS와 Hadoop을 연결하기 위한 설정들은 따로 설정파일로 관리 됩니다. 아래는 그 예제입니다.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
    <!--
    Configuration for IRIS
    -->
    <property>
        <name>iris.master.ip</name>
        <value>127.0.0.1</value>
    </property>
    <property>
        <name>iris.port.nsd</name>
        <value>5000</value>
```

```
    </property>
    <property>
        <name>iris.port.dsd</name>
        <value>5110</value>
    </property>
    <property>
        <name>iris.port.cmdd</name>
        <value>5800</value>
    </property>
    <property>
        <name>iris.port.ehd</name>
        <value>5678</value>
    </property>
</configuration>
```

설정 파일은 xml 형태로 제작되며 위의 설정값은 모두 포함되어야 합니다. 대부분의 경우 위의 기본 설정을 사용하시면 동작 가능합니다.

2.2.4.5 실행하기

기본적으로는 Hadoop MapReduce의 실행과 동일합니다.

```
[iris@master iris-test]$ hadoop jar IRISTest.jar IRISTest \
> --conf iris.conf \
> -Diris.base.query="SELECT * FROM HADOOP_TEST_TABLE;" \
> /test/iris_test_ouput
15/07/13 01:22:49 INFO IRISTest: configuration
15/07/13 01:22:49 INFO IRISTest: -base query : SELECT * FROM HADOOP_TEST_TABLE;
15/07/13 01:22:49 INFO IRISTest: starting
...
        Shuffle Errors
        BAD_ID=0
        CONNECTION=0
        IO_ERROR=0
        WRONG_LENGTH=0
        WRONG_MAP=0
        WRONG_REDUCE=0
    File Input Format Counters
        Bytes Read=0
```

```
    File Output Format Counters
        Bytes Written=156
15/07/13 01:23:35 INFO IRISTest: done
[iris@master iris-test]$
```

옵션 값을 살펴보면 다음과 같습니다.

```
hadoop jar [JAR_FILE_NAME] [CLASS_NAME] \
--conf [IRIS_CONFIGURATION_FILE_NAME]
-Diris.base.query="SELECT *FROM HADOOP_TEST_TABLE;" \
/test/iris_test_ouput
```

conf 값 까지는 이전에 만든 파일들을 대상으로 합니다. JAR_FILE는 IRISTest.jar 를 의미하고 내부 클래스 이름을 IRISTest, 마지막으로 설정파일은 iris.conf를 이용한다는 의미입니다. 다음에 나오는 옵션은 -Diris.base.query 입니다.

```
-Diris.base.query="SELECT *FROM HADOOP_TEST_TABLE;" \
```

이 의미는 IRIS에 위의 쿼리를 통해 데이터를 가져 오겠다는 의미입니다. 마지막 옵션은 최종 저장되는 데이터의 HDFS 위치 입니다. 프로그램이 정상적으로 동작한 후에 확인을 위해 아래와 같이 실행합니다.

```
[iris@HMaster1 iris-test]$ hadoop dfs -ls /test/iris_test_ouput
Found 2 items
-rw-r--r--   3 iris supergroup          0 2015-07-13 01:23 /test/iris_test_ouput/_SUCCESS
-rw-r--r--   3 iris supergroup        156 2015-07-13 01:23 /test/iris_test_ouput/part-00000
[iris@HMaster1 iris-test]$ hadoop dfs -cat /test/iris_test_ouput/part-00000
20150710103000  1
20150710103200  1
SAM 1
TOM 1
Woud    1
```

```
You 1
and 1
are 1
boss    1
car 1
changing    1
down    1
garage  1
mind    1
my  1
oil?    1
taking  1
the 3
to  1
you 1
[iris@HMaster1 iris-test]$
```

2.3 Spark with IRIS

본 항목에서는 Spark 에서 IRIS 데이터를 불러오기 위한 방법을 설명합니다.

2.3.1 Hadoop MapReduce의 한계점

Hadoop MapReduce와 IRIS 연결에는 다음과 같은 한계점이 있습니다.

- HDFS 가 설치되어 있어야 합니다. 앞장의 Hadoop 설치를 참조하세요.
- 가공한 데이터를 IRIS에 넣지 못합니다.
- IRIS와 Spark 은 동일한 계정으로 설치되어야 합니다.

2.3.2 Spark 설치

HDFS 를 사용하기 위해서, 우선 Hadoop 을 설치합니다. 앞장의 Hadoop 설치를 참조하세요.

2.3.2.1 Spark 설치파일 다운받기

Spark 패키지를 다운받습니다. IRIS는 Spark 1.4 를 지원합니다. 설치된 Hadoop 버전에 맞는 pre-built 패키지를 받습니다.

sqlite jdbc 드라이버 파일(.jar 파일)을 다운받습니다. 아래의 사이트에서 최신버전 jdbc 드라이버를 다운받습니다.

```
https://bitbucket.org/xerial/sqlite-jdbc/downloads
```

iris spark 드라이버 파일(.jar 파일)을 다운받습니다. 회사에 문의주시면 메일로 보내드립니다.

- 파일명 : `iris-spark-{버전정보}.jar`

iris pyspark 모듈(.py 파일)을 다운받습니다. 회사에 문의주시면 메일로 보내드립니다.

- 파일명 : `sparkiris.py`

2.3.2.2 Spark 설치하기

모든 노드에서 작업합니다.

Spark 패키지파일 압축을 해제합니다. Hadoop 과 동일하게, iris 계정의 ~/tools/ 에 압축을 해제합니다. 생성된 spark 디렉토리를 spark 로 링크를 생성합니다.

```
Spark 1.4 기준 설치화면

[iris@all-node tools]$ cd ~/tools
[iris@all-node tools]$ tar xzf spark-1.4.0-bin-hadoop2.6.tgz
[iris@all-node tools]$ ln -s spark-1.4.0-bin-hadoop2.6 spark
```

Spark 압축을 푼 디렉토리 속에 extlib 디렉토리를 생성합니다.

```
[iris@all-node ~]$ mkdir ~/tools/spark/extlib
```

extlib 디렉토리 속에 위에서 받은 추가파일들을 넣습니다. 파일은 총 세개로, sqlite jdbc 드라이버 파일, iris spark 드라이버 파일, iris pyspark 모듈 파일입니다.

```
[iris@all-node ~]$ ls ~/tools/spark/extlib/
iris-spark-1.5.0.1.jar
sparkiris.py
sqlite-jdbc-3.8.10.1.jar
```

2.3.2.3 Spark 설정하기

모든 노드에서 작업합니다.

설정파일은 모두 ~/tools/spark/conf 디렉토리 안에 있습니다.

spark-env.sh 파일 설정 : 추가라이브러리 파일 등록 및 마스터노드를 설정합니다.

```
[iris@all-node ~]$ vi ~/tools/spark/conf/spark-env.sh

SPARK_CLASSPATH=/home/iris/tools/spark/ext_lib/iris-spark-1.5.0.1.jar:\
/home/iris/tools/spark/ext_lib/sqlite-jdbc-3.8.10.1.jar
SPARK_MASTER_IP=master
```

slaves 파일 설정 : 슬레이브 노드 목록을 입력합니다. IP 또는 hostname 을 입력합니다.

```
[iris@all-node ~]$ vi ~/tools/spark/conf/slaves

slave01
slave02
slave03
```

2.3.3 Spark 실행하기

Spark 마스터노드에서 ~/tools/spark/sbin/start-all.sh 를 실행합니다.

2.3.4 Spark 정상 동작 확인하기

웹브라우저로 마스터노드의 8080 포트로 접속하면 클러스터 정보를 출력합니다. Workers 에 Slave 노드가 전부 표시되는 것을 확인할 수 있습니다.

```
![IRIS Directory Architecture](./images/01.sparkinfo.png)
```

예제를 실행해서 실제로 잘 동작하는지 확인할 수 있습니다.

```
[iris@master ~]$ cd ~/tools/spark/bin/
[iris@master bin]$ ./pyspark
.
.
.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 1.4.0
      /_/

Using Python version 2.7.5 (default, Apr 24 2015 02:38:59)
SparkContext available as sc, HiveContext available as sqlContext.

>>> from sparkiris import sparkiris
>>> sc = sparkiris(sc)

>>> iris_master_ip = {IRIS 마스터노드 IP}
>>> iris_id = 'test'
>>> iris_pwd = 'test'

>>> sc.set_iris_info(iris_master_ip, iris_id, iris_pwd)
```

```
>>> data = sc.irisDB("/*+ LOCATION (PARTITION >= '20150710000000') */ \
SELECT NODE_ID, UPDATETIME, USED FROM SYS_CPU_INFO;")
.
.
.
>>> for row in data.take(5):
        print row
.
.
.
(0, u'2,20150710040002,22.8915662651')
(0, u'2,20150710040102,2.60521042084')
(0, u'2,20150710040203,21.4428857715')
(0, u'2,20150710040303,23.2931726908')
(0, u'2,20150710040404,3.01810865191')
>>>
```

2.4 Tajo with IRIS

본 항목에서는 Tajo 에서 IRIS 데이터를 불러오기 위한 방법을 설명합니다.

2.4.1 Hadoop MapReduce의 한계점

Tajo 와 IRIS 연결에는 다음과 같은 한계점이 있습니다.

- HDFS 가 설치되어 있어야 합니다.
  - Tajo Worker 가 여러 노드에 분산실행하는경우, 노드간 데이터공유를 위해 필요합니다.
  - 앞장의 Hadoop 설치를 참조하세요.
- 가공한 데이터를 IRIS 에 넣지 못합니다.

2.4.2 Tajo 설치

HDFS 를 사용하기 위해서, 우선 Hadoop 을 설치합니다. 앞장의 Hadoop 설치를 참조하세요.

2.4.2.1 Tajo 설치파일 다운받기 Tajo-IRIS 패키지를 다운받습니다. - 회사에 문의주시면 메일로 보내드립니다. - 현재 0.9.0 버전 Tajo 패키지를 제공합니다. - tajo-iris-0.9.0.tar.gz - Hadoop 2.4.1 버전을 지원합니다.

2.4.2.2 Tajo 설치하기 모든 노드에서 작업합니다.

Tajo 패키지파일 압축을 해제합니다. Hadoop 과 동일하게, iris 계정의 ~/tools/ 에 압축을 해제합니다. 생성된 tajo-0.9.0 디렉토리를 tajo 로 링크를 생성합니다.

```
tajo-iris-0.9.0.tar.gz 기준 설치화면

[iris@all-node tools]$ cd ~/tools
[iris@all-node tools]$ tar xzf tajo-iris-0.9.0.tar.gz
[iris@all-node tools]$ ln -s tajo-0.9.0 tajo
```

2.4.2.3 Tajo 설정하기 모든 노드에서 작업합니다.

설정파일은 모두 ~/tools/tajo/conf 디렉토리 안에 있습니다.

tajo-env.sh 파일 설정 : 추가라이브러리 파일 등록 및 마스터노드를 설정합니다.

```
[iris@all-node ~]$ vi ~/tools/tajo/conf/-env.sh

export HADOOP_HOME=/home/iris/tools/hadoop
export JAVA_HOME=/usr/java/jdk1.7.0_79
```

tajo-site.xml 파일 설정 : conf 디렉토리 안에 tajo-site.xml.template 파일을 복사해서 기본파일을 만들고 아래의 내용을 넣습니다.

```
[iris@all-node ~]$ cp ~/tools/tajo/conf/tajo-site.xml.template \
~/tools/tajo/conf/tajo-site.xml
[iris@all-node ~]$ vi ~/tools/tajo/conf/tajo-site.xml

<configuration>
        <property>
                <name>tajo.rootdir</name>
                <value>hdfs://{MASTER_NODE_HOST_NAME}:9000/tajo-0.9.0-src</value>
        </property>
        <property>
                <name>tajo.master.umbilical-rpc.address</name>
                <value>{MASTER_NODE_HOST_NAME}:26001</value>
        </property>

        <property>
                <name>tajo.master.client-rpc.address</name>
                <value>{MASTER_NODE_HOST_NAME}:26002</value>
        </property>

        <property>
                <name>tajo.resource-tracker.rpc.address</name>
                <value>{MASTER_NODE_HOST_NAME}:26003</value>
        </property>

        <property>
                <name>tajo.catalog.client-rpc.address</name>
                <value>{MASTER_NODE_HOST_NAME}:26005</value>
        </property>
</configuration>
```

workers 파일 설정 : 슬레이브 노드 목록을 입력합니다. hostname 을 입력합니다.

```
[iris@all-node ~]$ vi ~/tools/tajo/conf/workers
```

```
{SLAVE_NODE_1_HOST_NAME}
{SLAVE_NODE_2_HOST_NAME}
{SLAVE_NODE_3_HOST_NAME}
```

2.4.3 Tajo 실행하기

Tajo 마스터노드에서 ~/tools/tajo/bin/start-tajo.sh 를 실행합니다.

2.4.4 Tajo 정상 동작 확인하기

tsql CLI 에서 쿼리를 실행해서 실제로 잘 동작하는지 확인할 수 있습니다. 테이블 등록 쿼리 (CREATE EXTERNAL TABLE) 과 SELECT 쿼리를 차례대로 실행합니다.

```
[iris@master ~]$ cd ~/tools/tajo/bin/
[iris@master bin]$ ./tsql
Java HotSpot(TM) 64-Bit Server VM warning: You have ...
It's highly recommended that you fix the library with ...
Try \? for help.

default> CREATE EXTERNAL TABLE SYS_CPU_INFO (NODE_ID text, \
CPU_CLOCK text, CPU_CORE text, USED text, LOAD_AVG text, \
IOWAIT text, UPDATETIME text) using iris with ('recordsep'='\n', \
'fieldsep'=',', 'irishome'='/home/iris/IRIS') partition by column \
(key text, partition text) location \
'iris://{마스터_hostname}:5210/SYS_CPU_INFO';

OK

default> SELECT NODE_ID, UPDATETIME, USED FROM SYS_CPU_INFO \
where partition >= '20150721000000' limit 5;

Progress: 0%, response time: 0.896 sec
Progress: 0%, response time: 0.899 sec
Progress: 6%, response time: 1.301 sec
```

```
Progress: 50%, response time: 2.103 sec
Progress: 100%, response time: 2.289 sec
node_id,  updatetime,  used
-------------------------------
0,  20150721081244,  4.03760408353
0,  20150721081300,  17.5869120654
0,  20150721081400,  2.05761316872
0,  20150721081501,  99.8007968127
0,  20150721081602,  2.05338809035
(5 rows, 2.289 sec, 155 B selected)
default>
```

2.4.5 Tajo with IRIS 사용하기

2.4.5.1 IRIS 테이블을 Tajo 에 등록하기 Tajo 와 IRIS 는 Catalog 정보 (DDL정보) 가 동기화되지 않습니다. 그래서 사용자가 CREATE EXTERNAL TABLE 쿼리를 사용해서 기존 IRIS 테이블을 Tajo Catalog 에 직접 등록합니다. 컬럼정보는 IRIS 와 동일하게 설정하며, 컬럼타입은 text 만 지원하고 있으므로, IRIS 테이블에서 integer 타입 등의 컬럼을 Tajo 에서는 text 로 설정합니다.

```
CREATE EXTERNAL TABLE {테이블이름}
(
    {컬럼이름} {컬럼타입} [, {컬럼이름} {컬럼타입} ]*
)
using iris with
    ('recordsep'='\n', 'fieldsep'=',', 'irishome'='/home/iris/IRIS')
partition by column
    (key text, partition text)
location 'iris://{마스터-hostname}:5210/{테이블이름}';


예시)
CREATE EXTERNAL TABLE SYS_CPU_INFO
(
    NODE_ID text,
```

```
    CPU_CLOCK text,
    CPU_CORE text,
    USED text,
    LOAD_AVG text,
    IOWAIT text,
    UPDATETIME text
)
using iris with
    ('recordsep'='\n', 'fieldsep'=',', 'irishome'='/home/iris/IRIS')
partition by column
    (key text, partition text)
location 'iris://master01:5210/SYS_CPU_INFO';
```

2.4.5.2 힌트 검색을 사용하기 Tajo 는 HINT 문법을 지원하지 않는 대신 WHERE 절에서 key, partition 컬럼을 사용해서 힌트를 설정합니다. Tajo 에서는 컬럼 대소문자를 구분하므로 key, partition 컬럼을 모두 소문자로 사용하세요.

```
- 기존 IRIS 쿼리

/*+ LOCATION (PARTITION >= '20150721000000') */
SELECT
    NODE_ID, UPDATETIME, USED
FROM
    SYS_CPU_INFO
WHERE
    and NODE_ID = '1'
limit 5;

- 위와 같은 의미의 Tajo 쿼리

SELECT
    NODE_ID, UPDATETIME, USED
FROM
    SYS_CPU_INFO
WHERE
    partition >= '20150721000000' and NODE_ID = '1'
```

```
limit 5;
```

2.5 Hive with IRIS

본 항목에서는 Hive에서 IRIS에 접근하기 위한 방법에 대해서 설명합니다.

2.5.1 Hive with IRIS 의 한계점

Hive 와 IRIS를 연동하여 사용하는 것에는 다음과 같은 한계점을 가집니다.

- Hadoop은 설치되어 있어야 한다.
- IRIS, Hadoop, Hive는 동일한 계정에 설치되어야 한다.
- Hive 버젼은 1.0.0 버전을 기준으로 한다.

2.5.2 Hive 설치

우선 Hive 설치에 앞서서 iris 계정에 Hadoop, IRIS는 이미 설치되어 있다는 가정하에 설명을 진행합니다.

```
[Hadoop MapReduce with IRIS 항목 참조]
```

Hive의 설치 디렉토리는 Hadoop과 동일하게 tools 밑으로 설치합니다.

2.5.2.1 Hive 다운로드 및 Hive 압축 해제 먼저 Hive 1.0.0 을 다운로드 합니다.

```
[iris@master ~]$ cd tools
[iris@master tools]$ wget https://archive.apache.org/dist/hive/hive-1.0.0/apache-hive-1.0.0-bin.tar.gz
...
[iris@master tools]$ tar xzvf apache-hive-1.0.0-bin.tar.gz
...
[iris@master tools]$ ln -s apache-hive-1.0.0-bin hive
[iris@master tools]$ ls
apache-hive-1.0.0-bin  apache-hive-1.0.0-bin.tar.gz
hadoop  hadoop-2.7.1  hadoop-2.7.1.tar.gz  hive
[iris@master tools]$
```

apache-hive-1.0.0-bin 폴더를 hive 폴더로 링크를 생성하여 준비를 합니다.

2.5.2.2 Hive 환경변수 설정 압축을 해제한 후 먼저 환경 변수를 선언합니다. 아래는 Hadoop까지 설치된 IRIS의 환경 설정 파일입니다.

```
[iris@master ~]$ vim ~/IRIS/env.sh
...
export HADOOP_HOME=/home/iris/tools/hadoop
export JAVA_HOME=/usr/java/jdk1.7.0_79
export PATH=$JAVA_HOME/bin:$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH
```

위의 부분을 다음과 같이 수정합니다.

```
[iris@master ~]$ vim ~/IRIS/env.sh
...
export HADOOP_HOME=/home/iris/tools/hadoop
export JAVA_HOME=/usr/java/jdk1.7.0_79
# 추가
export HIVE_HOME=/home/iris/tools/hive
# 수정
export PATH=$JAVA_HOME/bin:$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$HIVE_HOME/bin:$PATH
```

위의 설정 파일을 수정한 후 다음 명령어를 실행합니다.

```
[iris@master ~]$ source ~/IRIS/env.sh
```

2.5.2.3 Hive-IRIS API 설치 Hive에서 IRIS와 연결을 위해서 Hive-IRIS API를 제공합니다. 아래와 같이 메일을 보내주시면 해당 파일을 보내드립니다.

```
mailto : iris@mobigen.com
subject : [Hive-IRIS API] Hive-IRIS API 요청
```

요청 후 답변 메일을 받으시면 첨부로 아래와 같은 파일이 첨부됩니다.

```
hive-iris-[VERSION].jar
```

위의 파일을 다운로드 한 후에 아래에 설명한 디렉토리에 복사합니다.

```
[iris@master ~]$ cp hive-iris-[VERSION].jar ~/tools/hive/lib
```

2.5.2.4 Hive 설정 변경하기 Hive 설정을 변경하기 위해서 아래와 같은 파일을 생성해야합니다.

```
[iris@master ~]$ vim ~/tools/hive/conf/hive-site.xml
```

아래는 hive-site.xml의 내용입니다.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
    <property>
        <name>hive.metastore.rawstore.impl</name>
        <value>com.mobigen.iris.hive.metastore.IRISObjectStore</value>
    </property>
    <property>
        <name>fs.iris.impl</name>
        <value>com.mobigen.iris.FileSystem.IRISFileSystem</value>
    </property>
</configuration>
```

2.5.3 Hive 처음 실행하기

Hive를 실행하기 위해서는 아래 명령어를 입력하면 됩니다.

```
[iris@masster ~]$ hive
...
hive>
```

위와 같이 hive 콘솔이 뜬다면 정상적으로 성공한 것입니다.

2.5.4 IRIS에 테이블 생성하고 데이터 준비하기

Hive에서 데이터를 불러오기 전, IRIS에 테이블이 생성되어 있어야 합니다. iplus를 이용하여 아래 쿼리를 실행하여 테이블을 생성합니다.

```
[iris@master ~]$ iplus test
Password:
Connecting to IRIS(test@192.168.111.10:5050).
Connected to IRIS.
Enter ".help" for instructions
iplus> CREATE TABLE HIVE_TEST_TABLE (
     > NAME TEXT,
     > UPDATE_TIME TEXT,
     > MESSAGE TEXT
     > )
     > datascope LOCAL
     > ramexpire 30
     > diskexpire 60000
     > partitionkey NAME
     > partitiondate UPDATE_TIME
     > partitionrange 5;
Ret : +OK Create table

0.5576 sec
iplus>
```

위의 테이블을 생성한 후 간단한 데이터를 몇개 입력합니다.

```
iplus> INSERT INTO HIVE_TEST_TABLE (NAME, UPDATE_TIME, MESSAGE)
    > VALUES ( 'TOM', '20150713090000', 'hello word' );
Ret : +OK

0.2713 sec

iplus> INSERT INTO HIVE_TEST_TABLE (NAME, UPDATE_TIME, MESSAGE)
    > VALUES ( 'SAM', '20150713091000', 'not word, world' );
Ret : +OK

0.1382 sec

iplus> INSERT INTO HIVE_TEST_TABLE (NAME, UPDATE_TIME, MESSAGE)
    > VALUES ( 'MARRY', '20150713091400', 'word? world?' );
Ret : +OK

0.1333 sec

iplus>
```

2.5.5 Hive에서 IRIS 테이블 등록하기

Hive를 처음 실행하였다면 아직 테이블이 등록되지 않은 상태입니다. create external table 을 사용하여 IRIS 테이블을 등록합니다.

```
CREATE EXTERNAL TABLE [TABLE_NAME] (
    [COLUMN_INFO]
    ...
)
PARTITIONED BY (
    key STRING,
    partition STRING
)
STORED BY
    'com.mobigen.iris.hive.IRISStorageHandler'
```

```
TBLPROPERTIES (
    'iris.master.ip' = '[MASTER_IP_ADDRESS]',
    'iris.daemon.dld.port' = '5210',
    'iris.daemon.dsd.port' = '5110',
    'iris.user.name' = '[IRIS_USER_NAME]',
    'iris.user.passwd' = '[IRIS_USER_PASSWD]'
)
;
```

위의 쿼리는 Hive에 IRIS 테이블을 등록하기 위한 쿼리입니다. 위의 쿼리를 사용하기 전에 IRIS에는 해당 테이블이 이미 생성된 상태여야 합니다. 이전 항목에서 생성한 HIVE_TEST_TABLE을 등록하기 위해서 아래와 같은 쿼리를 입력합니다.

```
CREATE EXTERNAL TABLE HIVE_TEST_TABLE (
    NAME String,
    UPDATE_TIME String,
    MESSAGE String
)
PARTITIONED BY (
    key STRING,
    partition STRING
)
STORED BY
    'com.mobigen.iris.hive.IRISStorageHandler'
TBLPROPERTIES (
    'iris.master.ip' = '192.168.111.10',
    'iris.daemon.dld.port' = '5210',
    'iris.daemon.dsd.port' = '5110',
    'iris.user.name' = 'test',
    'iris.user.passwd' = 'test'
)
;
```

위의 쿼리를 실행한 결과는 아래와 같습니다.

```
hive> CREATE EXTERNAL TABLE HIVE_TEST_TABLE (
    >         NAME String,
    >         UPDATE_TIME String,
    >         MESSAGE String
    > )
    > PARTITIONED BY (
    >         key STRING,
    >         partition STRING
    > )
    > STORED BY
    >         'com.mobigen.iris.hive.IRISStorageHandler'
    > TBLPROPERTIES (
    >         'iris.master.ip' = '192.168.111.10',
    >         'iris.daemon.dld.port' = '5210',
    >         'iris.daemon.dsd.port' = '5110',
    >         'iris.user.name' = 'test',
    >         'iris.user.passwd' = 'test'
    >         'table.type' = 'iris'
    > )
    > ;
OK
Time taken: 0.988 seconds
hive> show tables;
OK
hive_test_table
Time taken: 0.074 seconds, Fetched: 1 row(s)
hive>
```

테이블이 생성된 것을 확인하였다면 셀렉트 쿼리를 전송하여 쿼리가 정상 실행되는지 확인해야 합니다. 먼저 iplus에서 조회를 진행합니다.

```
iplus> select * from HIVE_TEST_TABLE;
Ret : +OK Success

 NAME          UPDATE_TIME   MESSAGE
=======================================================
 SAM           20150713091000 not word, world
 TOM           20150713090000 hello word
```

```
 MARRY         20150713091400 word? world?
=======================================================
3 row in set
0.1492 sec

iplus>
```

조회 결과 3개의 데이터가 존재하는 것을 확인할 수 있습니다. 다음은 hive 에서 실행한 결과 입니다.

```
hive> select * from hive_test_table;
OK
MARRY   20150713091400  word? world?    MARRY   20150713091000
SAM 20150713091000  not word    SAM 20150713091000
TOM 20150713090000  hello word  TOM 20150713090000
Time taken: 0.939 seconds, Fetched: 3 row(s)
hive>
```

출력 형태에 차이가 있지만 정상적으로 3개의 데이터를 조회 할 수 있음을 확인할 수 있습니다.