Hadoop 예제 실행 매뉴얼

목차

1. 실행 가이드 개요

1.1 서비스를 사용하기 전에

- 본 매뉴얼은 사용자가 기본적인 Linux 환경에 익숙하다는 전제하에서 작성되었다.
- 본 매뉴얼은 사용자가 기본적인 Hadoop을 사용해봤다는 전제하에서 작성되었다.
- 본 매뉴얼은 Hadoop(ucloud MapReduce) 서비스가 정상적으로 실행 되었다는 가정 하에서 작성 되었다.

1.2 서비스 사용 환경

서비스의 원활한 동작을 위해서 서비스의 운용에 필요한 권장 사양은 아래와 같다.

서비스 운영 환경

구분	서비스 운영 사양
운영체제	Centos 5.4 64bit
Hadoop	Hadoop 1.0.3 stable
최소 필요한 가상머신 인스턴스 수	2 (Master (1) + Slave (1))

1.3 서비스에 대하여

ucloud MapReduce 서비스는 빅 데이터 분석을 위한 필수 플랫폼인 Hadoop을 provisioning 하는 서비스 이다. 빅 데이터 분석을 위한 맵 리듀스 프로그램을 JAR 와 Streaming 형식으로 실행 지원한다.

- 서비스의 장점: 간단한 신청만으로 Hadoop 시스템의 구축을 할 수 있다. 빅 데이터 분석 플랫폼인 Hadoop을 구축하는데 많은 시간을 소요 하지 않아도 된다.
- 서비스의 구성: Hadoop 플랫폼 기반 JAR / Streaming 실행 엔진
- 서비스 관련 용어 정의: 본 서비스에서는 사용자의 편의를 위해 일부 용어를 특정한 의미로 정의하여 사용한다. 이에 대해서는 아래 표를 참고한다.

용어	설명
MapReduce	구글에서 분산 컴퓨팅을 지원하기 위한 목적으로 제작 하여 2004년 발표한 소프트웨어 프레임워크 이다. 이 프레임 워크는 페타바이트 이상의 대용량 데이터를 신뢰할 수 없는 컴퓨터로 구성된 클러스터 환경에서 병렬 처리를 지원하기 위해서 개발되었다. 이 프레임워크는 함수형 프로그래밍에서 일반적으로 사용되는 맵 과 리듀스라는 함수 기반으로 구성 한다.

JAR	Java Archive의 준말 로써, 아카이브 파일 포맷 형식을 취한다. 많은 자바 클래스 파일들을 모아서 구성하고, 메타 데이터 및 여러 기반 리소스 파일들을 포함 한다. 맵 리듀스 프로그램을 자바 프로그램으로 구성하고 JAR 파일 형태로 만들고 실행 할 수 있다.
HDFS	Hadoop Distributed File System, Hadoop 분산 파일 시스템. 사용자의 데이터를 저장하는 용도로 분산 파일 시스템을 사용 한다.
Hadoop Streaming	하둡 스트리밍은 JAR 와 같이 맵 리듀스 모든 과정을 자바로 구구 한 것이 아니고, 사용자로 하여금 다양한 스크립트 언어로 맵 바이너리와 리듀스 바이너리로 구성하여 실행 할 수 있게 도와주는 일종의 유틸리티 이다.
Hadoop	대량의 자료를 처리할 수 있는 큰 컴퓨터 클러스터에서 동작하는 분산 응용 프로그램을 지원하는 자유 자바 소프트웨어 프레임워크 이다. 원래 너치의 분산 처리를 지원하기 위해서 개발된 것으로, 아파치 루씬의 하부 프로젝트이다. 분산 처리 시스템인 구글 파일 시스템을 대체할 수 있는 하둡 분산 파일 시스템과 맵리듀스를 구현 한 것이다.

1.4 매뉴얼에 대하여

이 매뉴얼에서는 사용자의 이해를 돕기 위해 표현 방식의 일관성을 최대한 유지한다.

- 표기 방식

이 매뉴얼에서는 다음의 표기 방식을 사용한다.

- 명령어 설명시 "< >" 안의 인자는 필수 인자 이며, "[ ]" 안의 인자는 선택 인자 이다. 예를 들면 다음과 같다.

JAR 기반 맵리듀스 프로그램 실행 명령어
hadoop@master:~$ hadoop jar <JAR 파일 이름> [mainClass 이름] [여러 인자값들..]

2. JAR 기반 하둡 실행 가이드

본 절에서는 ucloud MapReduce 서비스에서 JAR 기반 Hadoop 예제 실행 가이드를 설명한다.

2.1 JAR – WordCount

워드 카운트를 하고 싶은 입력 데이터를 ucloud MapReduce 서비스에서 제공하는 HDFS에 저장하고, 맵리듀스 실행 하는 과정을 설명한다.

- 입력 데이터 HDFS 저장하기

먼저 워드 카운트를 하고 싶은 입력 데이터를 마스터 가상 머신 인스턴스에 복사 하기 위해서 마스터 인스턴스 노드에 ssh 접속을 한다.

```
Jaeui-iMac:~ wja300$ ssh hadoop@x.x.x.x(Master IP)
```

접속한 후, 홈 디렉토리에 워드 카운트를 하려는 파일 들을 저장할 디렉토리를 생성한다.

```
hadoop@master:~$ cd ~
hadoop@master:~$ mkdir wordcount_input
```

위에서 생성한 디렉토리에 입력 데이터를 복사 해온다. (입력 데이터는 Swift , Amazon S3에서 가져올 수 있고, 임의의 파일 서버에서 가져올 수 있다. 서비스를 사용하는 사용자가 분석 하고 싶은 데이터를 복사 하면 된다. – 본 가이드에서는 마스터 인스턴스 노드 로컬 파일시스템의 특정 디렉토리 밑의 파일들을 입력 데이터로 가정 한다. 즉, 아래와 같이 특정 디렉토리 밑의 파일들을 입력 파일로 가정하고 위에서 만든 wordcount_input 디렉토리에 복사한다.)

```
hadoop@master:~$ cp /mnt/hadoop/conf/* wordcount_input/
hadoop@master:~$ ls -al wordcount_input/
total 84
drwxr-xr-x 2 hadoop hadoop 4096 2012-09-05 19:06 .
drwxr-xr-x 7 hadoop hadoop 4096 2012-09-05 19:00 ..
-rw-r--r-- 1 hadoop hadoop 7457 2012-09-05 19:06 capacity-scheduler.xml
-rw-r--r-- 1 hadoop hadoop  535 2012-09-05 19:06 configuration.xsl
-rw-r--r-- 1 hadoop hadoop  276 2012-09-05 19:06 core-site.xml
-rw-r--r-- 1 hadoop hadoop  327 2012-09-05 19:06 fair-scheduler.xml
-rw-r--r-- 1 hadoop hadoop 2282 2012-09-05 19:06 hadoop-env.sh
-rw-r--r-- 1 hadoop hadoop 1488 2012-09-05 19:06 hadoop-metrics2.properties
-rw-r--r-- 1 hadoop hadoop 4644 2012-09-05 19:06 hadoop-policy.xml
-rw-r--r-- 1 hadoop hadoop  258 2012-09-05 19:06 hdfs-site.xml
-rw-r--r-- 1 hadoop hadoop 4441 2012-09-05 19:06 log4j.properties
-rw-r--r-- 1 hadoop hadoop 2033 2012-09-05 19:06 mapred-queue-acls.xml
-rw-r--r-- 1 hadoop hadoop  271 2012-09-05 19:06 mapred-site.xml
```

```
-rw-r--r-- 1 hadoop hadoop      7 2012-09-05 19:06 masters
-rw-r--r-- 1 hadoop hadoop     14 2012-09-05 19:06 slaves
-rw-r--r-- 1 hadoop hadoop 1243 2012-09-05 19:06 ssl-client.xml.example
-rw-r--r-- 1 hadoop hadoop 1195 2012-09-05 19:06 ssl-server.xml.example
-rw-r--r-- 1 hadoop hadoop   382 2012-09-05 19:06 taskcontroller.cfg
```

위의 과정 처럼 wordcount_input 디렉토리에 입력 데이터 복사가 끝났다면, 하둡 명령어를 통해서 입력 데이터를 HDFS에 복사 한다. (입력 데이터가 클수록 복사 시간이 오래 걸리니 기다리도록 한다.)

HDFS 파일 저장하기 명령어
$hadoop fs –put <입력 데이터 디렉토리 혹은 파일> <출력 데이터 디렉토리 (HDFS 상의 디렉토리)>

HDFS 파일 리스팅 명령어
$hadoop fs –ls

```
hadoop@master:~$ /mnt/hadoop/bin/hadoop fs –put /home/hadoop/wordcount_input/ hdfs_wordcount_input

hadoop@master:~$ /mnt/hadoop/bin/hadoop fs –ls
Found 1 item
drwxr-xr-x   - hadoop supergroup          0 2012-09-05 19:19 /user/hadoop/hdfs_wordcount_input
```

- 저장된 입력 데이터에 워드 카운트 수행

위의 과정을 통해서 ucloud Server+ Hadoop 서비스의 HDFS에 입력데이터 복사 과정이 끝났으니, 저장된 입력 데이터에 워드 카운트를 수행 하자. 아래 과정은 입력 데이터 디렉토리 hdfs_wordcont_input 을 읽어서 워드 카운트를 맵리듀스로 실행 하고 결과를 hdfs_wordcount_output 디렉토리에 저장하는 과정이다.

JAR 기반 맵리듀스 프로그램 실행 명령어
hadoop@master:~$ hadoop jar <JAR 파일 이름> [mainClass 이름] [여러 인자값들..]

```
hadoop@master:~$ /mnt/hadoop/bin/hadoop jar /mnt/hadoop/hadoop-examples-1.0.3.jar wordcount hdfs_wordcount_input hdfs_wordcount_output
12/09/05 19:25:57 INFO input.FileInputFormat: Total input paths to process : 16
12/09/05 19:25:57 INFO util.NativeCodeLoader: Loaded the native-hadoop library
12/09/05 19:25:57 WARN snappy.LoadSnappy: Snappy native library not loaded
12/09/05 19:25:57 INFO mapred.JobClient: Running job: job_201208290147_0011
12/09/05 19:25:58 INFO mapred.JobClient:  map 0% reduce 0%
12/09/05 19:26:16 INFO mapred.JobClient:  map 25% reduce 0%
12/09/05 19:26:25 INFO mapred.JobClient:  map 50% reduce 0%
12/09/05 19:26:28 INFO mapred.JobClient:  map 50% reduce 12%
```

```
12/09/05 19:26:34 INFO mapred.JobClient:   map 75% reduce 12%
12/09/05 19:26:37 INFO mapred.JobClient:   map 75% reduce 16%
12/09/05 19:26:43 INFO mapred.JobClient:   map 100% reduce 25%
12/09/05 19:26:52 INFO mapred.JobClient:   map 100% reduce 100%
12/09/05 19:26:57 INFO mapred.JobClient: Job complete: job_201208290147_0011
12/09/05 19:26:57 INFO mapred.JobClient: Counters: 29
12/09/05 19:26:57 INFO mapred.JobClient:   Job Counters
12/09/05 19:26:57 INFO mapred.JobClient:     Launched reduce tasks=1
12/09/05 19:26:57 INFO mapred.JobClient:     SLOTS_MILLIS_MAPS=122961
12/09/05 19:26:57 INFO mapred.JobClient:     Total time spent by all reduces waiting after reserving slots (ms)=0
12/09/05 19:26:57 INFO mapred.JobClient:     Total time spent by all maps waiting after reserving slots (ms)=0
12/09/05 19:26:57 INFO mapred.JobClient:     Launched map tasks=16
12/09/05 19:26:57 INFO mapred.JobClient:     Data-local map tasks=16
12/09/05 19:26:57 INFO mapred.JobClient:     SLOTS_MILLIS_REDUCES=35830
12/09/05 19:26:57 INFO mapred.JobClient:   File Output Format Counters
12/09/05 19:26:57 INFO mapred.JobClient:     Bytes Written=15492
12/09/05 19:26:57 INFO mapred.JobClient:   FileSystemCounters
12/09/05 19:26:57 INFO mapred.JobClient:     FILE_BYTES_READ=22805
12/09/05 19:26:57 INFO mapred.JobClient:     HDFS_BYTES_READ=28991
12/09/05 19:26:57 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=413062
12/09/05 19:26:57 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=15492
12/09/05 19:26:57 INFO mapred.JobClient:   File Input Format Counters
12/09/05 19:26:57 INFO mapred.JobClient:     Bytes Read=26853
12/09/05 19:26:57 INFO mapred.JobClient:   Map-Reduce Framework
12/09/05 19:26:57 INFO mapred.JobClient:     Map output materialized bytes=22895
12/09/05 19:26:57 INFO mapred.JobClient:     Map input records=761
12/09/05 19:26:57 INFO mapred.JobClient:     Reduce shuffle bytes=22895
12/09/05 19:26:57 INFO mapred.JobClient:     Spilled Records=2206
12/09/05 19:26:57 INFO mapred.JobClient:     Map output bytes=35857
12/09/05 19:26:57 INFO mapred.JobClient:     CPU time spent (ms)=5860
12/09/05 19:26:57 INFO mapred.JobClient:     Total committed heap usage (bytes)=2580676608
12/09/05 19:26:57 INFO mapred.JobClient:     Combine input records=2600
12/09/05 19:26:57 INFO mapred.JobClient:     SPLIT_RAW_BYTES=2138
12/09/05 19:26:57 INFO mapred.JobClient:     Reduce input records=1103
12/09/05 19:26:57 INFO mapred.JobClient:     Reduce input groups=796
12/09/05 19:26:57 INFO mapred.JobClient:     Combine output records=1103
12/09/05 19:26:57 INFO mapred.JobClient:     Physical memory (bytes) snapshot=2796535808
12/09/05 19:26:57 INFO mapred.JobClient:     Reduce output records=796
12/09/05 19:26:57 INFO mapred.JobClient:     Virtual memory (bytes) snapshot=10167959552
12/09/05 19:26:57 INFO mapred.JobClient:     Map output records=2600
```

성공적으로 맵리듀스 실행이 되었다. 워드 카운트 결과를 확인 해보자.

- 워드 카운트 결과 확인

위에서 맵리듀스 결과를 저장한 HDFS 디렉토리인 hdfs_wordcount_output 의 내용을 아래와 같이 살펴 보면 입력 데이터의 단어들의 카운트가 제대로 된 것을 알 수 있다.

HDFS 파일 내용 읽기 명령어
$hadoop fs –cat <읽을 파일 이름(HDFS 상의 파일)>

```
hadoop@master:~$ /mnt/hadoop/bin/hadoop fs -cat hdfs_wordcount_output/*
"".             4
"*"             10
"alice,bob  10
"console"   1
"hadoop.root.logger". 1
"jks".          4
#               95
#*.sink.ganglia.dmax=jvm.metrics.threadsBlocked=70,jvm.metrics.memHeapUsedM=40   1
#*.sink.ganglia.slope=jvm.metrics.gcCount=zero,jvm.metrics.memHeapUsedM=both     1
#Default    1
#Security   1
#datanode.sink.file.filename=datanode-metrics.out       1
#datanode.sink.ganglia.servers=yourgangliahost_1:8649,yourgangliahost_2:8649             1
#jobtracker.sink.file.filename=jobtracker-metrics.out       1
#jobtracker.sink.ganglia.servers=yourgangliahost_1:8649,yourgangliahost_2:8649           1
#log4j.appender.DRFA.MaxBackupIndex=30  1
#log4j.appender.DRFA.layout.ConversionPattern=%d{ISO8601}       1
#log4j.appender.RFA.File=${hadoop.log.dir}/${hadoop.log.file}       1
#log4j.appender.RFA.MaxBackupIndex=30    1
#log4j.appender.RFA.MaxFileSize=1MB         1
#log4j.appender.RFA.layout.ConversionPattern=%d{ISO8601}         2
#log4j.appender.RFA.layout=org.apache.log4j.PatternLayout           1
#log4j.appender.RFA=org.apache.log4j.RollingFileAppender            1
#log4j.logger.org.apache.hadoop.fs.FSNamesystem=DEBUG               1
#log4j.logger.org.apache.hadoop.mapred.JobTracker=DEBUG         1
#log4j.logger.org.apache.hadoop.mapred.TaskTracker=DEBUG        1
#maptask.sink.file.filename=maptask-metrics.out             1
#maptask.sink.ganglia.servers=yourgangliahost_1:8649,yourgangliahost_2:8649             1
#namenode.sink.file.filename=namenode-metrics.out   1
#namenode.sink.ganglia.servers=yourgangliahost_1:8649,yourgangliahost_2:8649           1
#new        1
#reducetask.sink.file.filename=reducetask-metrics.out   1
#reducetask.sink.ganglia.servers=yourgangliahost_1:8649,yourgangliahost_2:8649          1
#tasktracker.sink.file.filename=tasktracker-metrics.out     1
#tasktracker.sink.ganglia.servers=yourgangliahost_1:8649,yourgangliahost_2:8649         1
$HADOOP_BALANCER_OPTS"      1
$HADOOP_DATANODE_OPTS"      1
$HADOOP_HOME/conf/slaves    1
$HADOOP_HOME/logs           1
$HADOOP_JOBTRACKER_OPTS"    1
$HADOOP_NAMENODE_OPTS"      1
$HADOOP_SECONDARYNAMENODE_OPTS" 1
```

```
.....

중략
```

- 워드 카운트 결과 출력 데이터로 저장 하기

워드 카운트 최종결과를 HDFS 에서 가져오는 예제를 살펴 보자.

HDFS 파일 가져오기 명령어
$hadoop fs –get <최종 결과 저장 디렉토리(HDFS 상의 디렉토리)> <사용자 정의 최종 결과 저장 디렉토리 (Master 노드상의 디렉토리)>

```
hadoop@master:~$ /mnt/hadoop/bin/hadoop fs -get hdfs_wordcount_output /home/hadoop/wordcount_output
hadoop@master:~$ ls -al /home/hadoop/wordcount_output/
total 28
drwxr-xr-x 3 hadoop hadoop  4096 2012-09-05 21:59 .
drwxr-xr-x 9 hadoop hadoop  4096 2012-09-05 21:59 ..
drwxr-xr-x 3 hadoop hadoop  4096 2012-09-05 21:59 _logs
-rw-r--r-- 1 hadoop hadoop 15492 2012-09-05 21:59 part-r-00000
-rw-r--r-- 1 hadoop hadoop     0 2012-09-05 21:59 _SUCCESS
```

위에서 part-r-00000 파일을 살펴보면 워드 카운트 최종 결과가 저장되어 있다.

2.2 JAR – TeraSort (CPU Bound Work)

TeraSort를 하고 싶은 입력 데이터를 ucloud MapReduce 서비스에서 제공하는 HDFS에 저장 하고, 맵리듀스 실행 하는 과정을 설명한다.

- 입력 데이터 HDFS 저장하기

먼저 TeraSort를 하고 싶은 입력 데이터를 마스터 가상 머신 인스턴스에 복사 하기 위해서 마스터 인스턴스 노드에 ssh 접속을 한다.

```
Jaeui-iMac:~ wja300$ ssh hadoop@x.x.x.x(Master IP)
```

접속한 후, 홈 디렉토리에 TeraSort를 하려는 파일 들을 저장할 디렉토리를 생성한다.

```
hadoop@master:~$ cd ~
hadoop@master:~$ mkdir terasort_input
```

위에서 생성한 디렉토리에 입력 데이터를 복사 해온다. (입력 데이터는 Swift , Amazon S3에서 가져올 수 있고, 임의의 파일 서버에서 가져올 수 있다. 서비스를 사용하는 사용자가 분석 하고

싶은 데이터를 복사 하면 된다. – 본 가이드에서는 마스터 인스턴스 노드 로컬 파일시스템의 특정 디렉토리 밑의 파일들을 입력 데이터로 가정 한다. 즉, 아래와 같이 특정 디렉토리 밑의 파일들을 입력 파일로 가정하고 위에서 만든 terasort_input 디렉토리에 복사한다.)

```
hadoop@master:~$ cp /mnt/hadoop/conf/* terasort_input/
hadoop@master:~$ ls -al terasort_input/
total 84
drwxr-xr-x 2 hadoop hadoop 4096 2012-09-05 19:06 .
drwxr-xr-x 7 hadoop hadoop 4096 2012-09-05 19:00 ..
-rw-r--r-- 1 hadoop hadoop 7457 2012-09-05 19:06 capacity-scheduler.xml
-rw-r--r-- 1 hadoop hadoop  535 2012-09-05 19:06 configuration.xsl
-rw-r--r-- 1 hadoop hadoop  276 2012-09-05 19:06 core-site.xml
-rw-r--r-- 1 hadoop hadoop  327 2012-09-05 19:06 fair-scheduler.xml
-rw-r--r-- 1 hadoop hadoop 2282 2012-09-05 19:06 hadoop-env.sh
-rw-r--r-- 1 hadoop hadoop 1488 2012-09-05 19:06 hadoop-metrics2.properties
-rw-r--r-- 1 hadoop hadoop 4644 2012-09-05 19:06 hadoop-policy.xml
-rw-r--r-- 1 hadoop hadoop  258 2012-09-05 19:06 hdfs-site.xml
-rw-r--r-- 1 hadoop hadoop 4441 2012-09-05 19:06 log4j.properties
-rw-r--r-- 1 hadoop hadoop 2033 2012-09-05 19:06 mapred-queue-acls.xml
-rw-r--r-- 1 hadoop hadoop  271 2012-09-05 19:06 mapred-site.xml
-rw-r--r-- 1 hadoop hadoop    7 2012-09-05 19:06 masters
-rw-r--r-- 1 hadoop hadoop   14 2012-09-05 19:06 slaves
-rw-r--r-- 1 hadoop hadoop 1243 2012-09-05 19:06 ssl-client.xml.example
-rw-r--r-- 1 hadoop hadoop 1195 2012-09-05 19:06 ssl-server.xml.example
-rw-r--r-- 1 hadoop hadoop  382 2012-09-05 19:06 taskcontroller.cfg
```

위의 과정 처럼 terasort_input 디렉토리에 입력 데이터 복사가 끝났다면, 하둡 명령어를 통해서 입력 데이터를 HDFS에 복사 한다. (입력 데이터가 클수록 복사 시간이 오래 걸리니 기다리도록 한다.)

HDFS 파일 쓰기 명령어
$hadoop fs –put <입력 데이터 디렉토리 혹은 파일> <출력 데이터 디렉토리 (HDFS 상의 디렉토리)>

HDFS 파일 리스팅 명령어
$hadoop fs –ls

```
hadoop@master:~$ /mnt/hadoop/bin/hadoop fs –put /home/hadoop/terasort_input/ hdfs_terasort_input

hadoop@master:~$ /mnt/hadoop/bin/hadoop fs -ls
Found 1 item
drwxr-xr-x   - hadoop supergroup          0 2012-09-05 21:32 /user/hadoop/hdfs_terasort_input
```

- 저장된 입력 데이터에 Terasort 수행

위의 과정을 통해서 ucloud MapReduce 서비스의 HDFS에 입력데이터 복사 과정이 끝났으니, 저장된 입력 데이터에 Terasort 를 수행 하자. 아래 과정은 입력 데이터 디렉토리 hdfs_terasort_input 을 읽어서 Terasort를 맵리듀스로 실행 하고 결과를 hdfs_terasort_output 디렉토리에 저장하는 과정이다.

JAR 기반 맵리듀스 프로그램 실행 명령어
hadoop@master:~$ hadoop jar <JAR 파일 이름> [mainClass 이름] [여러 인자값들..]

```
hadoop@master:~$ /mnt/hadoop/bin/hadoop jar /mnt/hadoop/hadoop-examples-1.0.3.jar terasort hdfs_terasort_input hdfs_terasort_output
12/09/05 21:39:41 INFO terasort.TeraSort: starting
12/09/05 21:39:41 INFO mapred.FileInputFormat: Total input paths to process : 16
12/09/05 21:39:41 INFO util.NativeCodeLoader: Loaded the native-hadoop library
12/09/05 21:39:41 WARN snappy.LoadSnappy: Snappy native library not loaded
12/09/05 21:39:42 INFO zlib.ZlibFactory: Successfully loaded & initialized native-zlib library
12/09/05 21:39:42 INFO compress.CodecPool: Got brand-new compressor
Making 1 from 631 records
Step size is 631.0
12/09/05 21:39:42 INFO mapred.FileInputFormat: Total input paths to process : 16
12/09/05 21:39:42 INFO mapred.JobClient: Running job: job_201208290147_0012
12/09/05 21:39:43 INFO mapred.JobClient:  map 0% reduce 0%
12/09/05 21:40:01 INFO mapred.JobClient:  map 25% reduce 0%
12/09/05 21:40:10 INFO mapred.JobClient:  map 50% reduce 0%
12/09/05 21:40:13 INFO mapred.JobClient:  map 50% reduce 16%
12/09/05 21:40:19 INFO mapred.JobClient:  map 75% reduce 16%
12/09/05 21:40:28 INFO mapred.JobClient:  map 100% reduce 25%
12/09/05 21:40:37 INFO mapred.JobClient:  map 100% reduce 100%
12/09/05 21:40:42 INFO mapred.JobClient: Job complete: job_201208290147_0012
12/09/05 21:40:42 INFO mapred.JobClient: Counters: 31
12/09/05 21:40:42 INFO mapred.JobClient:   Job Counters
12/09/05 21:40:42 INFO mapred.JobClient:     Launched reduce tasks=1
12/09/05 21:40:42 INFO mapred.JobClient:     SLOTS_MILLIS_MAPS=123530
12/09/05 21:40:42 INFO mapred.JobClient:     Total time spent by all reduces waiting after reserving slots (ms)=0
12/09/05 21:40:42 INFO mapred.JobClient:     Total time spent by all maps waiting after reserving slots (ms)=0
12/09/05 21:40:42 INFO mapred.JobClient:     Rack-local map tasks=2
12/09/05 21:40:42 INFO mapred.JobClient:     Launched map tasks=16
12/09/05 21:40:42 INFO mapred.JobClient:     Data-local map tasks=14
12/09/05 21:40:42 INFO mapred.JobClient:     SLOTS_MILLIS_REDUCES=35926
12/09/05 21:40:42 INFO mapred.JobClient:   File Input Format Counters
12/09/05 21:40:42 INFO mapred.JobClient:     Bytes Read=26853
12/09/05 21:40:42 INFO mapred.JobClient:   File Output Format Counters
12/09/05 21:40:42 INFO mapred.JobClient:     Bytes Written=27614
12/09/05 21:40:42 INFO mapred.JobClient:   FileSystemCounters
12/09/05 21:40:42 INFO mapred.JobClient:     FILE_BYTES_READ=31208
12/09/05 21:40:42 INFO mapred.JobClient:     HDFS_BYTES_READ=28783
12/09/05 21:40:42 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=437410
```

```
12/09/05 21:40:42 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=27614
12/09/05 21:40:42 INFO mapred.JobClient:   Map-Reduce Framework
12/09/05 21:40:42 INFO mapred.JobClient:     Map output materialized bytes=29234
12/09/05 21:40:42 INFO mapred.JobClient:     Map input records=761
12/09/05 21:40:42 INFO mapred.JobClient:     Reduce shuffle bytes=29234
12/09/05 21:40:42 INFO mapred.JobClient:     Spilled Records=1522
12/09/05 21:40:42 INFO mapred.JobClient:     Map output bytes=27615
12/09/05 21:40:42 INFO mapred.JobClient:     Total committed heap usage (bytes)=2580676608
12/09/05 21:40:42 INFO mapred.JobClient:     CPU time spent (ms)=6230
12/09/05 21:40:42 INFO mapred.JobClient:     Map input bytes=26853
12/09/05 21:40:42 INFO mapred.JobClient:     SPLIT_RAW_BYTES=1930
12/09/05 21:40:42 INFO mapred.JobClient:     Combine input records=0
12/09/05 21:40:42 INFO mapred.JobClient:     Reduce input records=761
12/09/05 21:40:42 INFO mapred.JobClient:     Reduce input groups=212
12/09/05 21:40:42 INFO mapred.JobClient:     Combine output records=0
12/09/05 21:40:42 INFO mapred.JobClient:     Physical memory (bytes) snapshot=2810167296
12/09/05 21:40:42 INFO mapred.JobClient:     Reduce output records=761
12/09/05 21:40:42 INFO mapred.JobClient:     Virtual memory (bytes) snapshot=10108071936
12/09/05 21:40:42 INFO mapred.JobClient:     Map output records=761
12/09/05 21:40:42 INFO terasort.TeraSort: done
```

성공적으로 맵리듀스 실행이 되었다. Terasort 결과를 확인 해보자.

- Terasort 결과 확인

위에서 맵리듀스 결과를 저장한 HDFS 디렉토리인 hdfs_terasort_output 의 내용을 아래와 같이 살펴 보면 다음과 같다.

HDFS 파일 내용 읽기 명령어
$hadoop fs –cat <읽을 파일 이름(HDFS 상의 파일)>

```
hadoop@master:~$ /mnt/hadoop/bin/hadoop fs -cat hdfs_terasort_output/*


… (공백들) : ASCII 코드 값이 작은 것부터 차례로 정렬 되므로




        Default value of -1 implies a queue can use complete capacity of the cluster.
        One important thing to note is that maximum-capacity is a percentage , so based on the cluster's capacity
        This property could be to curtail certain jobs which are long running in nature from occupying more than a
        absolute terms would increase accordingly.
        certain percentage of the cluster, which in the absence of pre-emption, could lead to capacity guarantees of
        other queues being affected.
        the max capacity would change. So if large no of nodes or racks get added to the cluster , max Capacity in
```

```
      account in scheduling decisions.
      account in scheduling decisions by default in a job queue.
      for the job queue at any given point of time by default.
      to be available for jobs in this queue.
     concurrently, by the CapacityScheduler.
    </description>
    </description>
    </description>
    </description>
    </description>
    </description>
    </description>
    </description>
    </description>
    </description>
    </description>
    </description>
    </description>
    </description>
    </description>
    </description>
    </description>
    <description>The amount of time in miliseconds which is used to poll
    <description> Each queue enforces a limit on the percentage of resources
    <description>The default maximum number of tasks, across all jobs in the
    <description>The maximum number of tasks per-user, across all the of the
    <description>The default multiple of queue-capacity which is used to
    <description>The default multipe of (maximum-system-jobs * queue-capacity)
    <description>The multipe of (maximum-system-jobs * queue-capacity) used to
    <description>Percentage of the number of slots in the cluster that are
    <description>Number of worker threads which would be used by
    <description>The multiple of the queue capacity which can be configured to
    <description>Maximum number of jobs in the system which can be initialized,
    <description>If true, priorities of jobs will be taken into
    <description>The percentage of the resources limited to a particular user
    <description>The maximum number of tasks, across all jobs in the queue,
    <description>If true, priorities of jobs will be taken into
    <description>
    <description>The default maximum number of tasks per-user, across all the of
    <description>ACL for InterTrackerProtocol, used by the tasktrackers to
    <description>ACL for ClientProtocol, which is used by user code
    <description>ACL for AdminOperationsProtocol, used by the mradmins commands
    <description>ACL for RefreshAuthorizationPolicyProtocol, used by the
    <description>ACL for ClientDatanodeProtocol, the client-to-datanode protocol

.....

중략
```

- Terasort 결과 출력 데이터로 저장 하기

Terasort 최종결과를 HDFS 에서 가져오는 예제를 살펴 보자.

HDFS 파일 가져오기 명령어
$hadoop fs –get <최종 결과 저장 디렉토리(HDFS 상의 디렉토리)> <사용자 정의 최종 결과 저장 디렉토리 (Master 노드상의 디렉토리)>

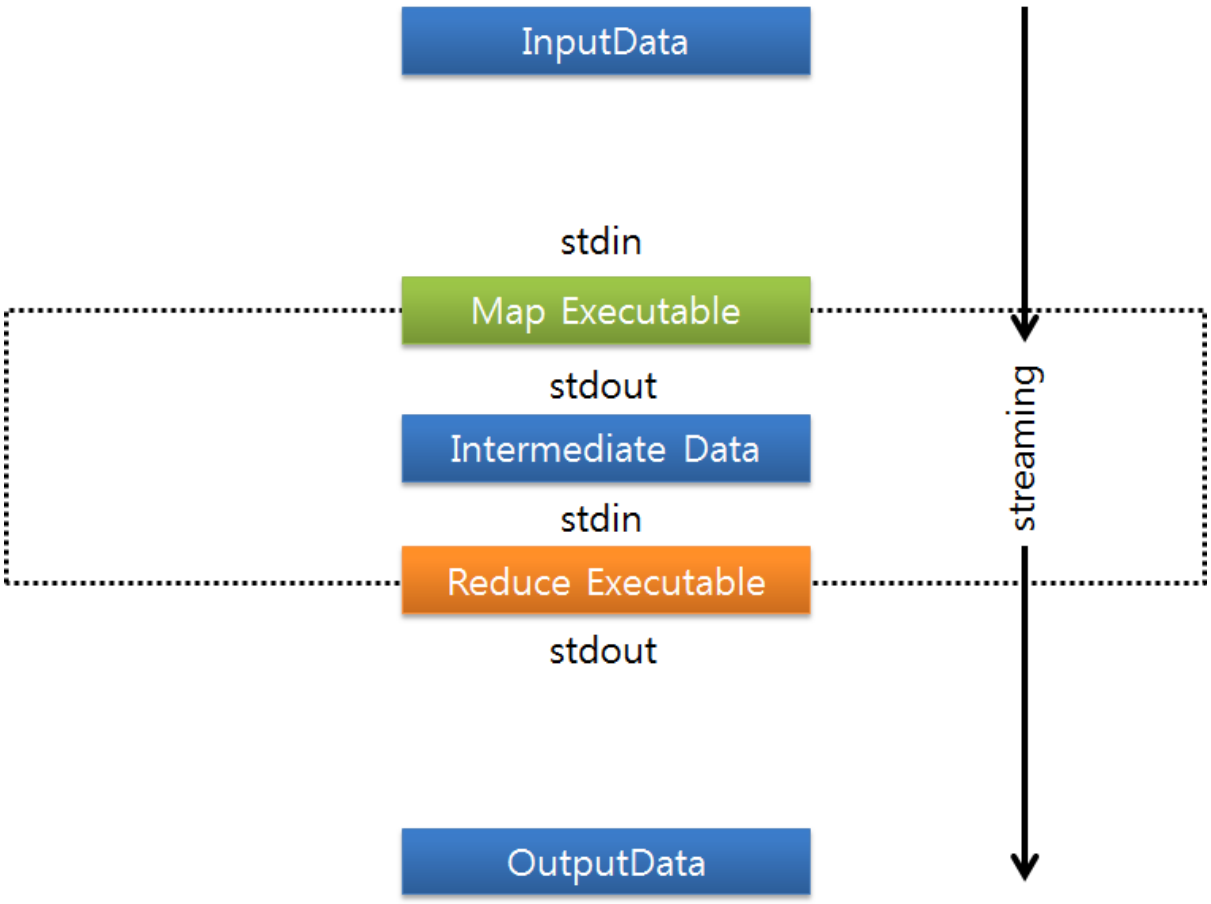
```
hadoop@master:~$ /mnt/hadoop/bin/hadoop fs -get hdfs_terasort_output /home/hadoop/terasort_output
hadoop@master:~$ ls -al /home/hadoop/terasort_output/
total 28
drwxr-xr-x 3 hadoop hadoop  4096 2012-09-05 21:59 .
drwxr-xr-x 9 hadoop hadoop  4096 2012-09-05 21:59 ..
drwxr-xr-x 3 hadoop hadoop  4096 2012-09-05 21:59 _logs
-rw-r--r-- 1 hadoop hadoop 15492 2012-09-05 21:59 part-r-00000
-rw-r--r-- 1 hadoop hadoop     0 2012-09-05 21:59 _SUCCESS
```

위에서 part-r-00000 파일을 살펴보면 Terasort 최종 결과가 저장되어 있다.

3. Streaming 기반 하둡 실행 가이드

3.1 Steaming – WordCount (Ruby)

앞 절에서 살펴본 워드 카운트 예제를 스트리밍 방식으로 ucloud MapReduce 서비스에서 제공하는 HDFS에 저장 하고, 맵리듀스 실행 하는 과정을 설명한다.



스트리밍 방식의 하둡 실행은 앞절에서 살펴본 JAR 기반 하둡 실행과는 다르다. 위의 그림을 살펴보면, JAR 기반 방식과 다르게 사용자가 작성한 Map Executable 과 Reduce Executable을 중간에 스트리밍 형식으로 맵리듀스를 실행 할 수 있다. 사용자가 작성 가능한 Map Executable 과 Reduce Executable은 다양한 프로그래밍 언어를 지원 한다. 예를 들어 Python, Ruby 등이 있다. 본 가이드에서는 워드 카운트를 Ruby 언어로 Map Executable 과 Reduce Executable로 작성하고 실행 하는 과정을 살펴 본다.

- 입력 데이터 HDFS 저장하기

먼저 워드 카운트를 하고 싶은 입력 데이터를 마스터 가상 머신 인스턴스에 복사 하기 위해서 마스터 인스턴스 노드에 ssh 접속을 한다.

```
Jaeui-iMac:~ wja300$ ssh hadoop@x.x.x.x(Master IP)
```

접속한 후, 홈 디렉토리에 워드 카운트를 하려는 파일 들을 저장할 디렉토리를 생성한다.

```
hadoop@master:~$ cd ~
hadoop@master:~$ mkdir wordcount_input_streaming
```

위에서 생성한 디렉토리에 입력 데이터를 복사 해온다. (입력 데이터는 Swift , Amazon S3에서 가져올 수 있고, 임의의 파일 서버에서 가져올 수 있다. 서비스를 사용하는 사용자가 분석 하고 싶은 데이터를 복사 하면 된다. – 본 가이드에서는 마스터 인스턴스 노드 로컬 파일시스템의 특정 디렉토리 밑의 파일들을 입력 데이터로 가정 한다. 즉, 아래와 같이 특정 디렉토리 밑의 파일들을 입력 파일로 가정하고 위에서 만든 wordcount_input_streaming 디렉토리에 복사한다.)

```
hadoop@master:~$ cp /mnt/hadoop/conf/* wordcount_input_streaming
hadoop@master:~$ ls -al wordcount_input_steaming
total 84
drwxr-xr-x 2 hadoop hadoop 4096 2012-09-05 19:06 .
drwxr-xr-x 7 hadoop hadoop 4096 2012-09-05 19:00 ..
-rw-r--r-- 1 hadoop hadoop 7457 2012-09-05 19:06 capacity-scheduler.xml
-rw-r--r-- 1 hadoop hadoop  535 2012-09-05 19:06 configuration.xsl
-rw-r--r-- 1 hadoop hadoop  276 2012-09-05 19:06 core-site.xml
-rw-r--r-- 1 hadoop hadoop  327 2012-09-05 19:06 fair-scheduler.xml
-rw-r--r-- 1 hadoop hadoop 2282 2012-09-05 19:06 hadoop-env.sh
-rw-r--r-- 1 hadoop hadoop 1488 2012-09-05 19:06 hadoop-metrics2.properties
-rw-r--r-- 1 hadoop hadoop 4644 2012-09-05 19:06 hadoop-policy.xml
-rw-r--r-- 1 hadoop hadoop  258 2012-09-05 19:06 hdfs-site.xml
-rw-r--r-- 1 hadoop hadoop 4441 2012-09-05 19:06 log4j.properties
-rw-r--r-- 1 hadoop hadoop 2033 2012-09-05 19:06 mapred-queue-acls.xml
-rw-r--r-- 1 hadoop hadoop  271 2012-09-05 19:06 mapred-site.xml
-rw-r--r-- 1 hadoop hadoop    7 2012-09-05 19:06 masters
-rw-r--r-- 1 hadoop hadoop   14 2012-09-05 19:06 slaves
-rw-r--r-- 1 hadoop hadoop 1243 2012-09-05 19:06 ssl-client.xml.example
-rw-r--r-- 1 hadoop hadoop 1195 2012-09-05 19:06 ssl-server.xml.example
-rw-r--r-- 1 hadoop hadoop  382 2012-09-05 19:06 taskcontroller.cfg
```

위의 과정 처럼 wordcount_input_streaming 디렉토리에 입력 데이터 복사가 끝났다면, 하둡 명령어를 통해서 입력 데이터를 HDFS에 복사 한다. (입력 데이터가 클수록 복사 시간이 오래 걸리니 기다리도록 한다.)

HDFS 파일 쓰기 명령어
$hadoop fs –put <입력 데이터 디렉토리 혹은 파일> <출력 데이터 디렉토리 (HDFS 상의 디렉토리)>

HDFS 파일 리스팅 명령어
$hadoop fs –ls

```
hadoop@master:~$ /mnt/hadoop/bin/hadoop fs –put /home/hadoop/wordcount_input_streaming/ hdfs_wordcount_input_streaming

hadoop@master:~$ /mnt/hadoop/bin/hadoop fs -ls
Found 1 item
drwxr-xr-x   - hadoop supergroup          0 2012-09-05 22:39 /user/hadoop/hdfs_wordcount_input_streaming
```

- 저장된 입력 데이터에 워드 카운트 Streaming 수행

위의 과정을 통해서 ucloud MapReduce 서비스의 HDFS에 입력데이터 복사 과정이 끝났으니, 저장된 입력 데이터에 워드 카운트를 수행 하자. 실제 맵 리듀스 실행을 수행 하기 전에 앞서 설명 했던, Map Executable 과 Reduce Executable을 Ruby 언어로 작성 한다.

Map.rb (워드 카운트를 위한 Map Executable)

```
#!/usr/bin/env ruby
STDIN.each_line do |line|
  line.split.each do |word|
    puts "#{word}₩t1"
  end
end
```

Reduce.rb (워드 카운트를 위한 Reduce Executable)

```
#!/usr/bin/env ruby
wordhash = {}
STDIN.each_line do |line|
  word, count = line.strip.split
  if wordhash.has_key?(word)
    wordhash[word] += count.to_i
  else
    wordhash[word] = count.to_i
  end
end
wordhash.each {|record, count| puts "#{record}₩t#{count}"}
```

위의 두 개의 Ruby Executable은 Master 인스턴스 노드의 작업 디렉토리에서 생성 하고 실행 권한을 준다.

```
hadoop@master:~$ cd ~
hadoop@master:~$ mkdir ruby_streaming_wordcount
hadoop@master:~$ cd ruby_streaming_wordcount/
hadoop@master:~/ruby_streaming_wordcount$ vi map.rb (위의 map.rb 프로그램을 작성한다.)
hadoop@master:~/ruby_streaming_wordcount$ vi reduce.rb (위의 reduce.rb 프로그램을 작성한다.)
hadoop@master:~/ruby_streaming_wordcount$ chmod +x map.rb
hadoop@master:~/ruby_streaming_wordcount$ chmod +x reduce.rb
```

아래 과정은 입력 데이터 디렉토리 hdfs_wordcount_input_streaming 을 읽어서 워드 카운트를 맵리듀스로 실행 하고 결과를 hdfs_wordcount_output_streaming 디렉토리에 저장하는 과정이다.

Streaming 기반 맵리듀스 프로그램 실행 명령어
hadoop@master:~$ hadoop jar </mnt/hadoop/contrib/streaming/hadoop-streaming-1.0.3.jar> -file <Map Executable 파일 경로 및 파일 이름> -mapper <Map Executable 파일 이름> - file <Reduce Executable 파일 경로 및 파일 이름> -reducer <Reduce Executable 파일 이름> -input <입력 데이터 디렉토리(HDFS 상의 디렉토리)> -output <출력 데이터 디렉토리(HDFS 상의 디렉토리)>

```
hadoop@master:~$ /mnt/hadoop/bin/hadoop jar /mnt/hadoop/contrib/streaming/hadoop-streaming-1.0.3.jar -file /home/hadoop/ruby_streaming_wordcount/map.rb -mapper map.rb -file /home/hadoop/ruby_streaming_wordcount/reduce.rb -reducer reduce.rb -input hdfs_wordcount_input_streaming -output hdfs_wordcount_output_streaming
packageJobJar: [/home/hadoop/ruby_streaming_wordcount/map.rb, /home/hadoop/ruby_streaming_wordcount/reduce.rb, /tmp/hadoop-hadoop/hadoop-unjar5234354400540147174/] [] /tmp/streamjob6015261705791082675.jar tmpDir=null
12/09/05 22:59:58 INFO util.NativeCodeLoader: Loaded the native-hadoop library
12/09/05 22:59:58 WARN snappy.LoadSnappy: Snappy native library not loaded
12/09/05 22:59:58 INFO mapred.FileInputFormat: Total input paths to process : 16
12/09/05 22:59:58 INFO streaming.StreamJob: getLocalDirs(): [/tmp/hadoop-hadoop/mapred/local]
12/09/05 22:59:58 INFO streaming.StreamJob: Running job: job_201208290147_0014
12/09/05 22:59:58 INFO streaming.StreamJob: To kill this job, run:
12/09/05 22:59:58 INFO streaming.StreamJob: /mnt/hadoop/libexec/../bin/hadoop job -Dmapred.job.tracker=master:9001 -kill job_201208290147_0014
12/09/05 22:59:58 INFO streaming.StreamJob: Tracking URL: http://master:50030/jobdetails.jsp?jobid=job_201208290147_0014
12/09/05 22:59:59 INFO streaming.StreamJob:  map 0%  reduce 0%
12/09/05 23:00:17 INFO streaming.StreamJob:  map 25%  reduce 0%
12/09/05 23:00:26 INFO streaming.StreamJob:  map 50%  reduce 0%
12/09/05 23:00:29 INFO streaming.StreamJob:  map 50%  reduce 17%
12/09/05 23:00:35 INFO streaming.StreamJob:  map 75%  reduce 17%
12/09/05 23:00:44 INFO streaming.StreamJob:  map 100%  reduce 17%
12/09/05 23:00:47 INFO streaming.StreamJob:  map 100%  reduce 25%
12/09/05 23:00:56 INFO streaming.StreamJob:  map 100%  reduce 100%
12/09/05 23:01:02 INFO streaming.StreamJob: Job complete: job_201208290147_0014
12/09/05 23:01:02 INFO streaming.StreamJob: Output: hdfs_wordcount_output_streaming
```

성공적으로 맵리듀스 실행이 되었다. 워드 카운트 Streaming 결과를 확인 해보자.

- 워드 카운트 Streaming 결과 확인

위에서 맵리듀스 결과를 저장한 HDFS 디렉토리인 hdfs_wordcount_output_streaming 의 내용을 아래와 같이 살펴 보면 다음과 같다.

HDFS 파일 내용 읽기 명령어
$hadoop fs –cat <읽을 파일 이름(HDFS 상의 파일)>

```
hadoop@master:~$ /mnt/hadoop/bin/hadoop fs -cat hdfs_wordcount_output_streaming/*
configure  1
(maximum-system-jobs            2
refresh     2
When        1
value.      2
includes    1
mapred.capacity-scheduler.queue.<queue-name>.property-name. 1
#Default    1
<description>Must    2
#namenode.sink.ganglia.servers=yourgangliahost_1:8649,yourgangliahost_2:8649          1
log4j.appender.DRFA.layout=org.apache.log4j.PatternLayout          1
version="1.0">          1
related     1
list        27
<value>5000</value>             1
%c{2}:      2
So          1
<value>3000</value>             1
submission,             1
"*"         10
log4j.appender.TLA.layout.ConversionPattern=%d{ISO8601}          1
default     9
where       2
<name>ssl.client.truststore.location</name>             1
policy      1
<name>mapred.capacity-scheduler.default-supports-priority</name>          1
</table>    1
log4j.appender.TLA.taskId=${hadoop.tasklog.taskid}     1
stored.     2
determine   3
allocations 1
task        1
this        19
<name>security.job.submission.protocol.acl</name>    1
nodes       2
TaskLog     1
Where       1
former      1
Sends       1
time        3
Each        1
<name>security.task.umbilical.protocol.acl</name>     1
<name>mapred.job.tracker</name>          1
```

```
configured 3
match="configuration">          1
initialize    2
.....

중략
```

- 워드 카운트 Streaming 결과 출력 데이터로 저장 하기

워드 카운트 Streaming 최종결과를 HDFS 에서 가져오는 예제를 살펴 보자.

HDFS 파일 가져오기 명령어
$hadoop fs –get <최종 결과 저장 디렉토리(HDFS 상의 디렉토리)> <사용자 정의 최종 결과 저장 디렉토리 (Master 노드상의 디렉토리)>

```
hadoop@master:~$ /mnt/hadoop/bin/hadoop fs -get hdfs_wordcount_output_streaming
/home/hadoop/wordcount_output_streaming
hadoop@master:~$ ls -al /home/hadoop/wordcount_output_streaming
total 28
drwxr-xr-x  3 hadoop hadoop  4096 2012-09-05 23:12 .
drwxr-xr-x 13 hadoop hadoop  4096 2012-09-05 23:12 ..
drwxr-xr-x  3 hadoop hadoop  4096 2012-09-05 23:12 _logs
-rw-r--r--  1 hadoop hadoop 15492 2012-09-05 23:12 part-00000
-rw-r--r--  1 hadoop hadoop     0 2012-09-05 23:12 _SUCCESS
```

위에서 part-r-00000 파일을 살펴보면 워드 카운트 Streaming 최종 결과가 저장되어 있다.