ADDB

Hadoop, Spark, Redis Setting

연세대학교 컴퓨터과학과 박상현

2019년 11월

과제명: IoT 환경을 위한 고성능 플래시 메모리 스토리지 기반 인메모리 분산 DBMS 연구개발

과제번호: 2017-0-00477

과학기술정보통신부
Ministry of Science and ICT

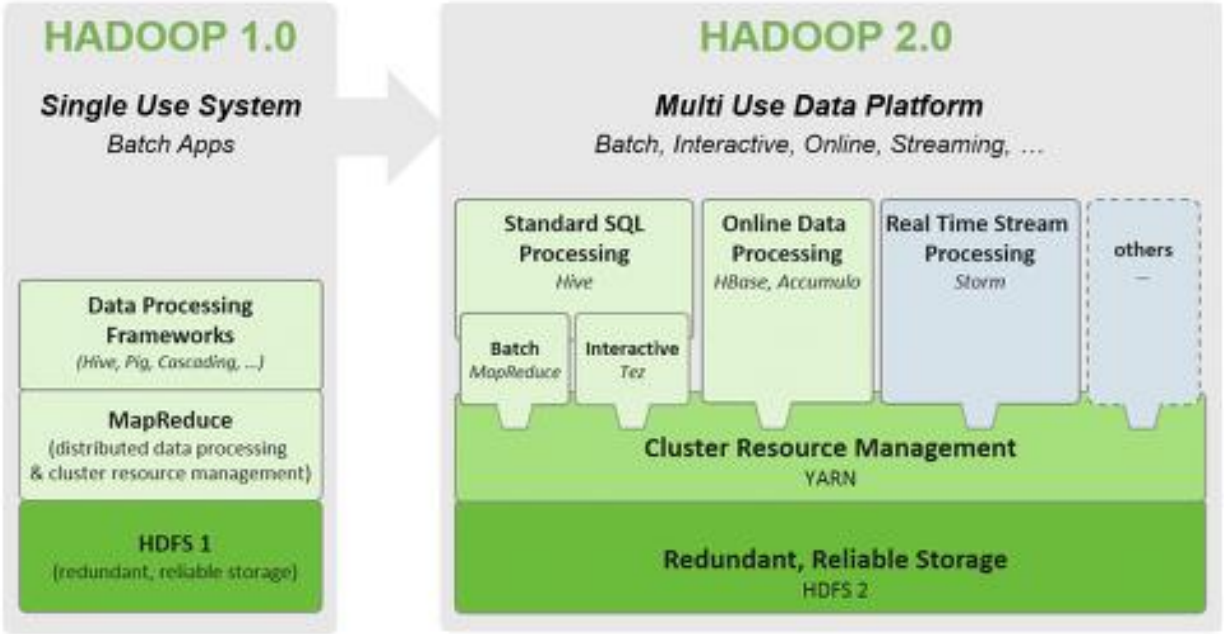
연세대학교
YONSEI UNIVERSITY

정보통신기술진흥센터
Institute for Information & communications Technology Promotion

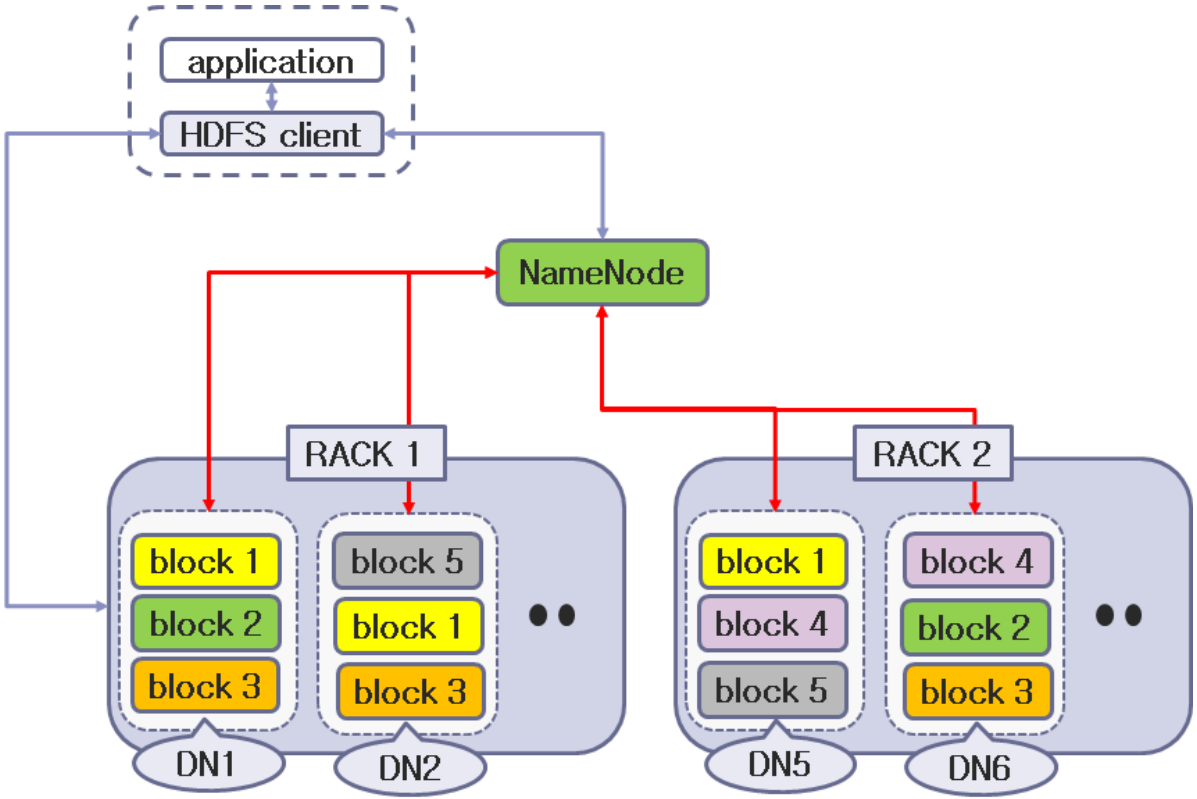
Contents

- Review
- ADDB overview
- Apache Hadoop (HDFS + YARN)
- Apache Spark, Thrift server (JDBC)
- Redis, RocksDB

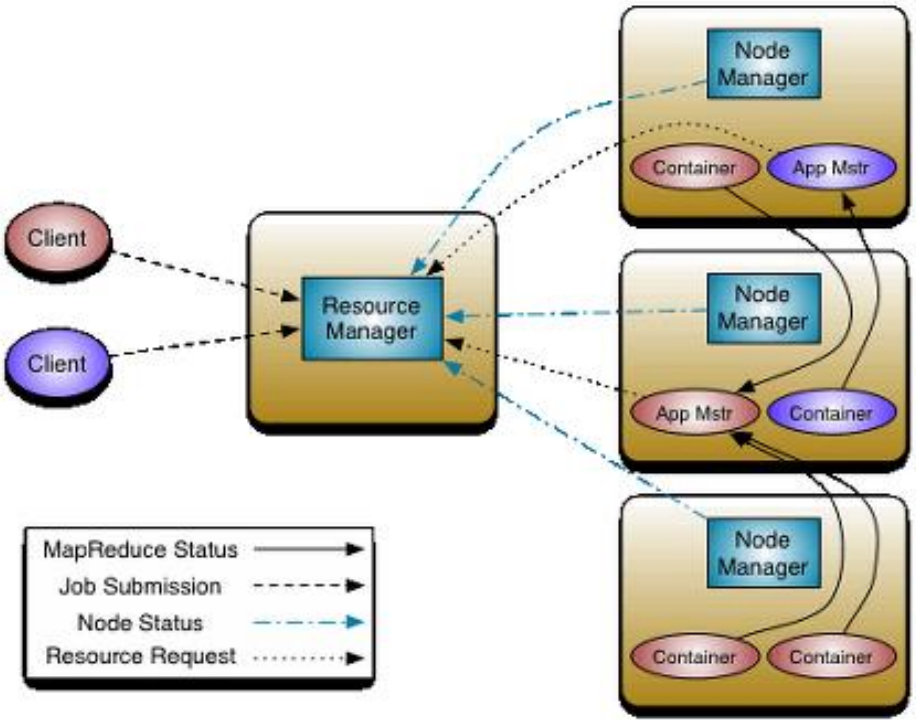
Review - Apache Hadoop



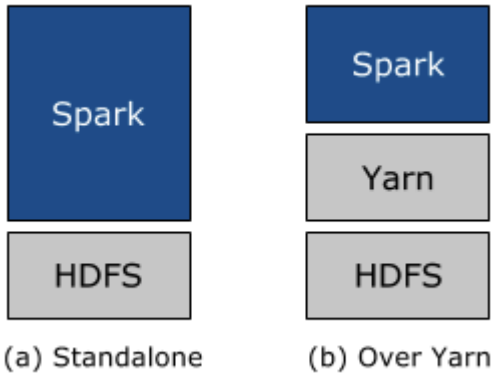
Review - HDFS

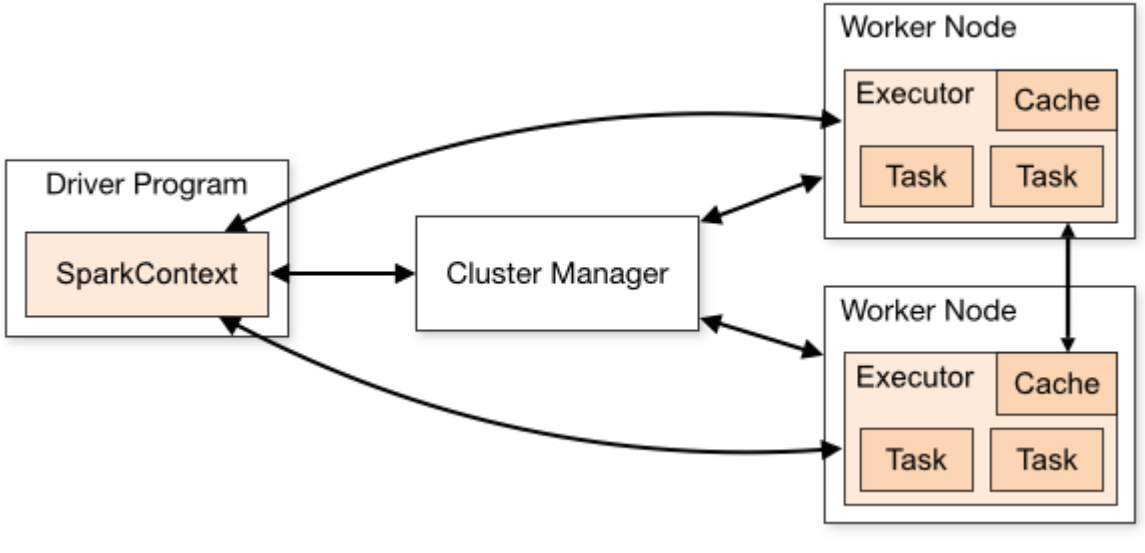

application
HDFS client
NameNode
RACK 1
RACK 2
block 1
block 2
block 3
block 5
block 1
block 3
block 1
block 4
block 5
block 4
block 2
block 3
DN1
DN2
DN5
DN6

Review - YARN

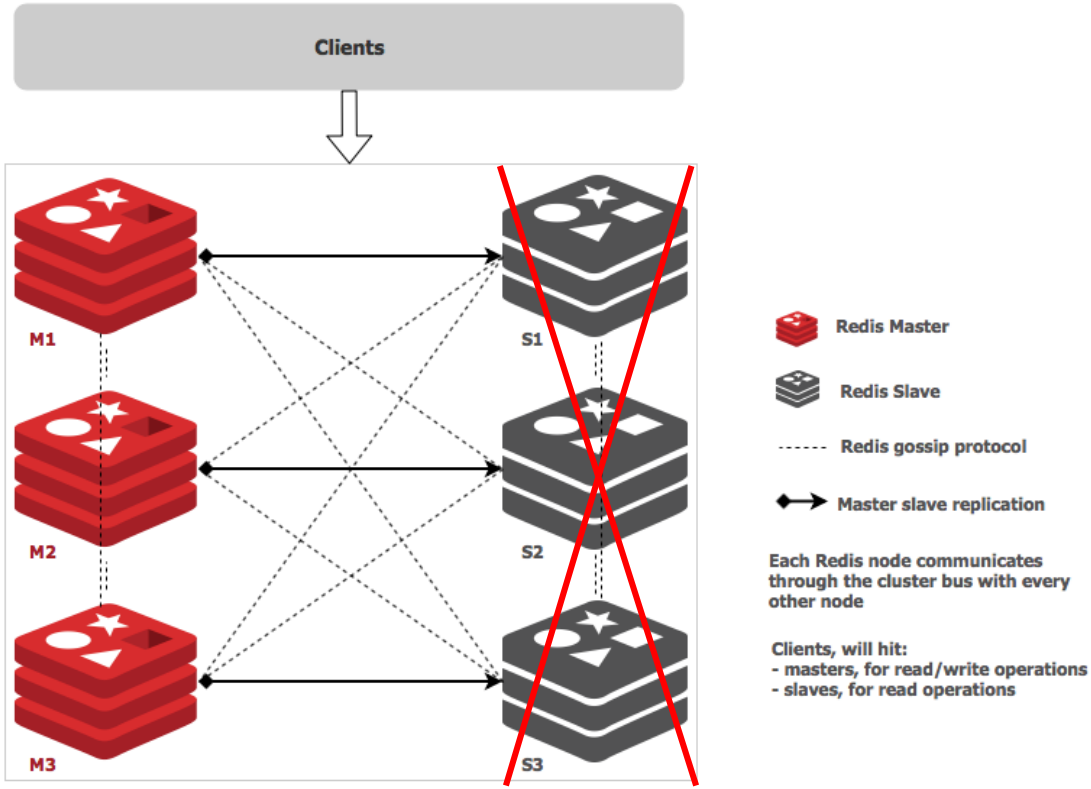

Node Manager
Container
App Mstr
Client
Client
Resource Manager
Node Manager
App Mstr
Container
Node Manager
Container
Container
MapReduce Status
Job Submission
Node Status
Resource Request

Review - Apache Spark

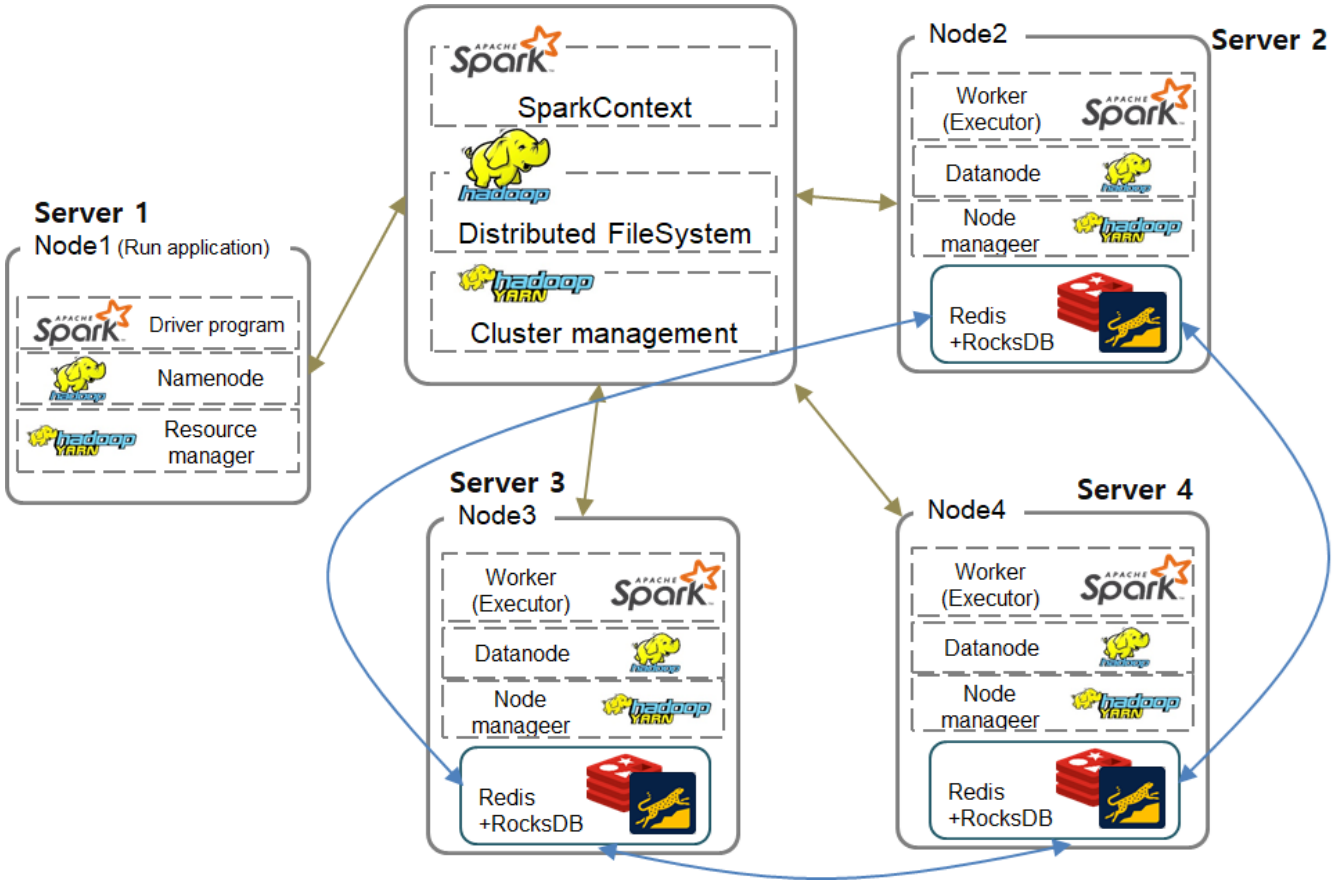




Review - Redis cluster



ADDB overview



ADDB overview - cluster spec.

- Server 총 8대 := 1대[Master] - Namenode, Resource manager, Driver program

  7대[Slaves] - Datanode, Node manager, Executor, Redis-RocksDB

- CPU - 8 core (4 core + hyper threading)
- Memory - 64GB := Spark executor 약 20~25GB(3 executors * 6~8GB),

  Redis cluster 30GB(6 instance * 5GB)
- RocksDB에 대한 resource 할당, configuration은 불분명한 상태.

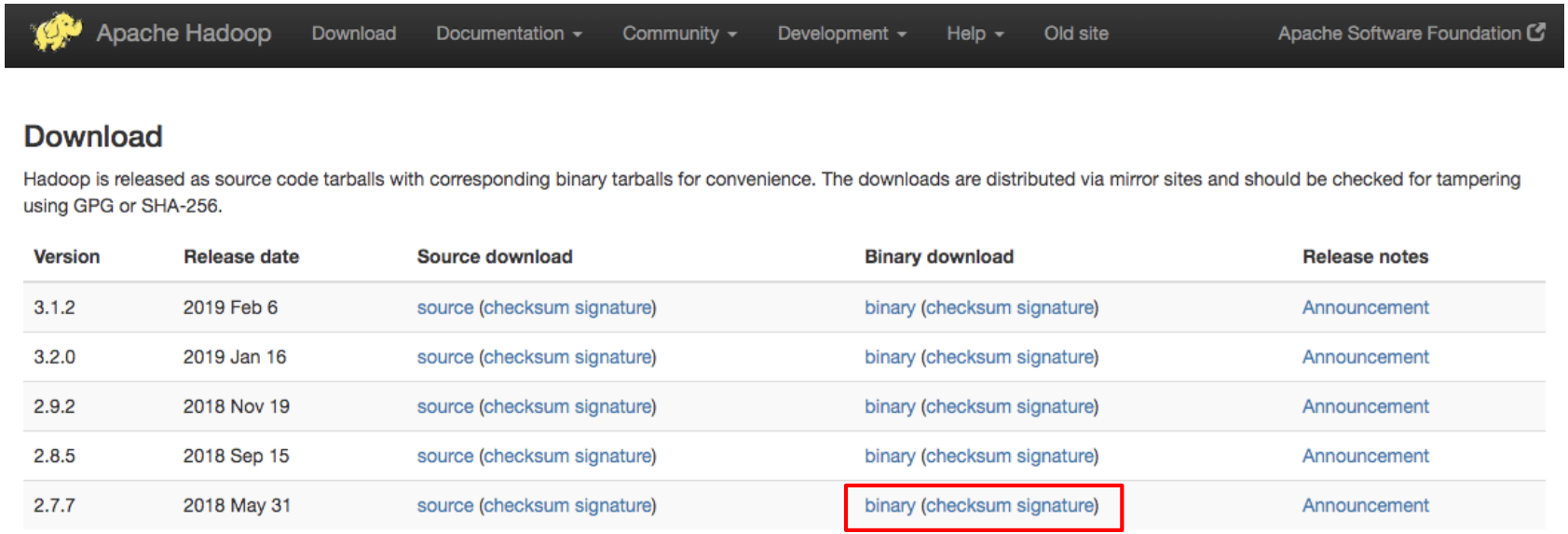
Apache Hadoop

1. Prerequisite
2. Configuration file 설정
3. 방화벽 설정
4. SSH 접속 설정
5. 실행 및 데이터 import

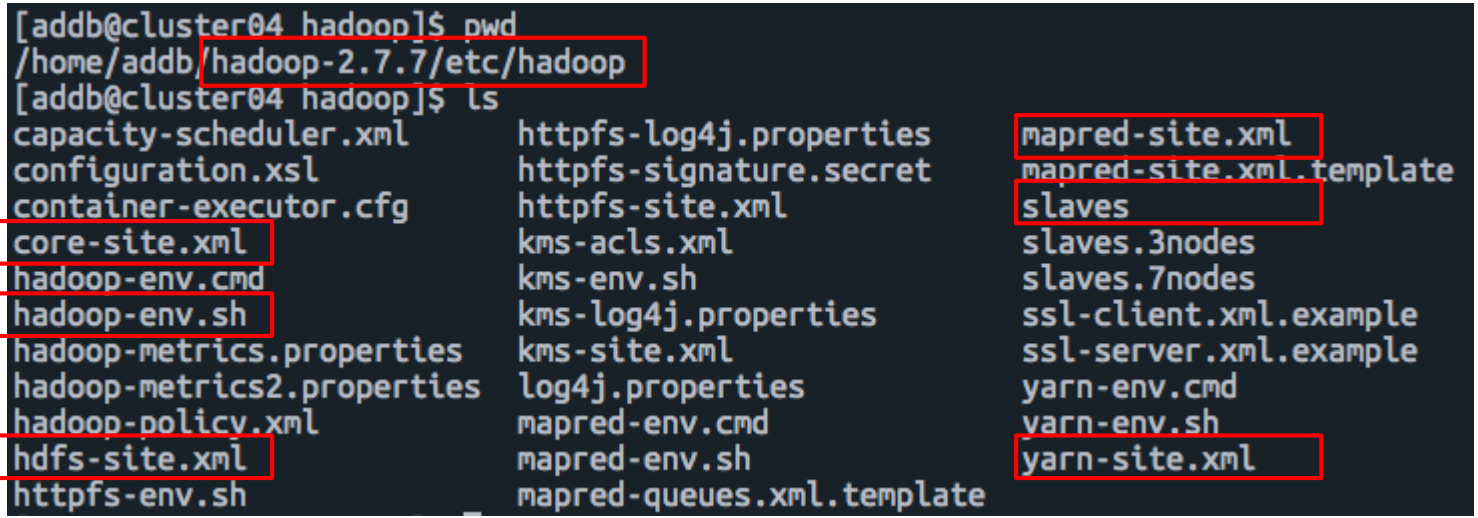
Apache Hadoop - 1. Prerequisite

Java

Apache Hadoop binary



Apache Hadoop - 2. Configuration file 설정



Apache Hadoop - 2. Configuration file 설정

1. core-site.xml - namenode에 대한 host 설정
2. hadoop-env.sh - JAVA_HOME 설정
3. hdfs-site.xml - namenode log, datanode path, replication 수 등 hdfs 설정
4. mapred-site.xml - map-reduce 사용에 대한 설정
5. yarn-site.xml - resource manager, node manager, shuffling 설정
6. slaves - slaves 설정

Apache Hadoop - 2. Configuration file 설정

1. core-site.xml - namenode에 대한 host 설정

```
<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>fs.defaultFS</name>
    <!--<value>hdfs://192.168.1.4:9007</value>-->
    <value>hdfs://cluster04:9007</value>
  </property>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/home/addb/hadoopData/tmp</value>
  </property>
</configuration>
```

Apache Hadoop - 2. Configuration file 설정

2. hadoop-env.sh - JAVA_HOME 설정

```
# Set Hadoop-specific environment variables here.

# The only required environment variable is JAVA_HOME.  All others are
# optional.  When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.

# The java implementation to use.
export JAVA_HOME=${JAVA_HOME}
```

Apache Hadoop - 2. Configuration file 설정

3. hdfs-site.xml - namenode log, datanode path, replication 수 등 hdfs 설정

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>3</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>/home/addb/hadoopData/name</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>/home/addb/hadoopData/data,/home/addb/hadoopData/data1,/home/addb/hadoopData/data2</value>
  </property>
  <property>
    <name>dfs.namenode.datanode.registration.ip-hostname-check</name>
    <value>false</value>
  </property>
  <property>
    <name>dfs.namenode.http-address</name>
<!--     <value>192.168.1.4:55070</value> -->
    <value>cluster04:55070</value>
  </property>
  <property>
    <name>dfs.namenode.fs-limits.min-block-size</name>
    <value>1</value>
  </property>
</configuration>
```

Apache Hadoop - 2. Configuration file 설정

4. mapred-site.xml - map-reduce 사용에 대한 환경설정

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
  <property>
    <name>mapreduce.shuffle.port</name>
    <value>13563</value>
  </property>
</configuration>
```

https://hadoop.apache.org/docs/r2.7.2/hadoop-mapreduce-client/hadoop-mapreduce-client-core/mapred-default.xml

Apache Hadoop - 2. Configuration file 설정

5. yarn-site.xml

resource manager,

node manager,

shuffling 설정

```
<configuration>

<!-- Site specific YARN configuration properties -->
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <!-- <value>192.168.1.4</value> -->
    <value>cluster04</value>
  </property>
  <property>
    <name>yarn.log-aggregation-enable</name>
    <value>false</value>
  </property>
  <property>
    <name>yarn.resourcemanager.scheduler.address</name>
    <!-- <value>192.168.1.4:8880</value> -->
    <value>cluster04:8880</value>
  </property>
  <property>
    <name>yarn.nodemanager.resource.memory-mb</name>
    <!--<value>40960</value>-->
    <value>30720</value>
  </property>
  <property>
          <name>yarn.nodemanager.resource.cpu-vcores</name>
          <value>6</value>
  </property>
  <property>
          <name>yarn.scheduler.minimum-allocation-mb</name>
          <value>512</value>
  </property>
  <property>
          <name>yarn.scheduler.maximum-allocation-mb</name>
          <!-- <value>8192</value> -->
          <!-- <value>40960</value> -->
          <value>30720</value>
  </property>
  <property>
          <name>yarn.scheduler.increment-allocation-mb</name>
          <value>512</value>
  </property>
  <property>
          <name>yarn.scheduler.minimum-allocation-vcores</name>
          <value>1</value>
  </property>
  <property>
          <name>yarn.scheduler.maximum-allocation-vcores</name>
          <value>4</value>
  </property>
  <property>
          <name>yarn.nodemanager.log.retain-seconds</name>
          <value>259200</value>
```

```
<property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle,spark_shuffle</value>
</property>
<property>
  <name>yarn.nodemanager.aux-services.spark_shuffle.class</name>
  <value>org.apache.spark.network.yarn.YarnShuffleService</value>
</property>
<property>
        <name>yarn.nodemanager.vmem-check-enable</name>
        <value>false</value>
</property>
<property>
        <name>yarn.nodemanager.vmem-pmem-ratio</name>
        <value>3</value>
</property>
```

https://blrunner.tistory.com/103

Apache Hadoop - 2. Configuration file 설정

Configure external shuffle service (Spark with YARN)

```
[addb@cluster04 yarn]$ pwd
/usr/local/spark/spark-2.0.2/yarn
[addb@cluster04 yarn]$ ls
spark-2.0.2-yarn-shuffle.jar
[addb@cluster04 yarn]$
```

```
[addb@cluster04 common]$ pwd
/home/addb/hadoop-2.7.7/share/hadoop/common
[addb@cluster04 common]$ ll
합계 5540
-rw-r--r--. 1 addb addb 1974977  7월 19  2018 hadoop-common-2.7.7-tests.jar
-rw-r--r--. 1 addb addb 3499158  7월 19  2018 hadoop-common-2.7.7.jar
-rw-r--r--. 1 addb addb  183155  7월 19  2018 hadoop-nfs-2.7.7.jar
drwxr-xr-x. 2 addb addb    4096  7월 19  2018 jdiff
drwxr-xr-x. 2 addb addb    4096  7월 19  2018 lib
drwxr-xr-x. 2 addb addb      89  7월 19  2018 sources
lrwxrwxrwx. 1 addb addb      62  9월 11  2018 spark-2.0.2-yarn-shuffle.jar -> /usr/local/spark/spark-2.0.2/yarn/spark-2.0.2-yarn-shuffl
e.jar
drwxr-xr-x. 2 addb addb      27  7월 19  2018 templates
[addb@cluster04 common]$
```

https://spark.apache.org/docs/latest/running-on-yarn.html

Apache Hadoop - 2. Configuration file 설정

6. slaves

```
cluster01
cluster05
cluster06
cluster07
cluster09
cluster13
cluster16
```

```
#Cluster Hosts
xxx.xxx.xxx.xx   cluster01
xxx.xxx.xxx.xx   cluster05
xxx.xxx.xxx.xx   cluster06
xxx.xxx.xxx.xx   cluster07
⋮
```

/etc/hosts

Apache Hadoop - 3. 방화벽 설정

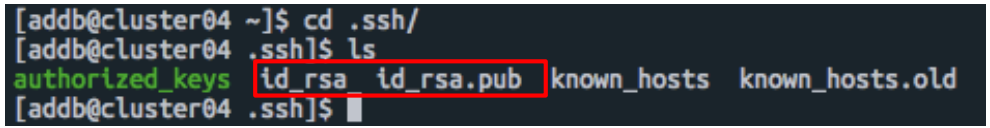
- 방화벽이 올바르게 설정되지 않으면, HDFS/YARN 가 켜지지 않음

```
$ sudo firewall-cmd --state
$ sudo firewall-cmd --zone=public --add-port=8005/tcp --permanent
$ sudo firewall-cmd --reload
$ sudo firewall-cmd --stop
```

Apache Hadoop - 4. SSH 접속 설정

1. ssh-keygen

   $ ssh-keygen -t rsa



2. public key를 remote server에 등록

   $ ssh-copy-id -i ~/.ssh/id_rsa.pub ${REMOTE_USER}@${REMOTE_HOST}

3. remote서버의 .ssh directory, authorized_keys 권한 변경

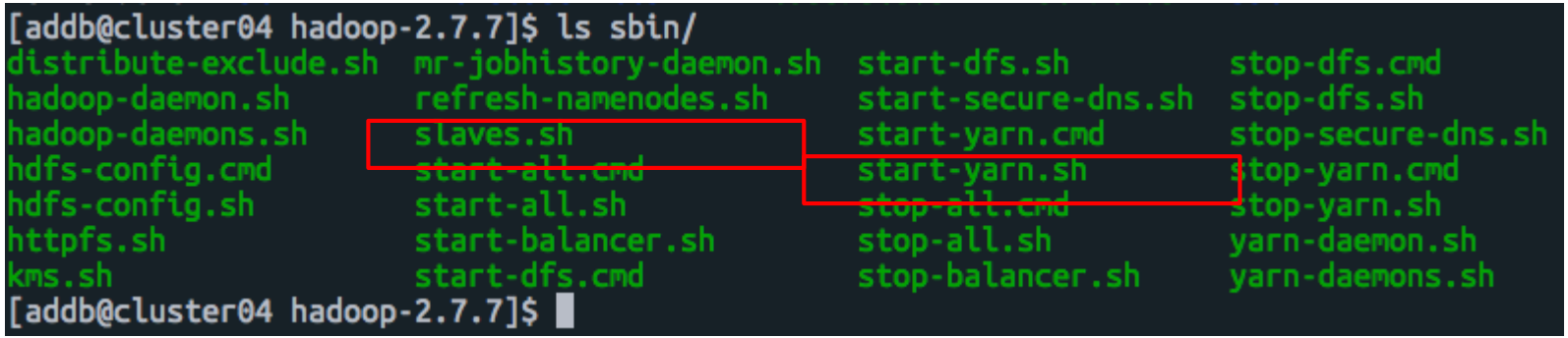
   .ssh -> 0700

   authorized_keys -> 0600

Apache Hadoop - 5. 실행 및 데이터 import

${HADOOP_HOME}/sbin

HDFS, YARN 실행 / 중지 - start-all.sh / stop-all.sh



Apache Hadoop - 5. 실행 및 데이터 import

1. HDFS, YARN 실행 확인 => jps 명령어로 올바르게 실행되었는지 확인
2. HDFS에 directory 생성
3. Directory에 tbl 파일 import

Apache Spark

1. Prerequisite
2. Configuration file 설정 (+Hadoop, YARN)
3. jar 파일 build
4. Thrift server
5. Beeline 사용법 (JDBC)
6. 실행 방법

Apache Spark - 1. Prerequisite

Java

Scala

Maven (Java & Scala build)

Apache Spark source

ADDB-jedis, ADDB Spark-Redis Connector

Apache Spark - 2. Configuration file 설정

1. spark-defaults.conf - 개별 application 단위의 설정
2. spark-env.sh - Cluster의 각 서버 단위의 설정
3. log4j.properties - log 설정

https://www.slideshare.net/JunyoungPark22/spark-config

Apache Spark - 2. Configuration file 설정

1. spark-defaults.conf

```
# Example:
# spark.master                     spark://master:7077
# spark.eventLog.enabled           true
# spark.eventLog.dir               hdfs://namenode:8021/directory
# spark.serializer                 org.apache.spark.serializer.KryoSerializer
# spark.driver.memory              5g
# spark.executor.extraJavaOptions  -XX:+PrintGCDetails -Dkey=value -Dnumbers="one two three"
#spark.sql.hive.metastore.version 1.1.2
#spark.sql.hive.metastore.jars maven
```

Apache Spark - 2. Configuration file 설정

Spark Application properties 설정 3가지 방법

1. SparkConf()
2. spark-submit 실행 매개변수
3. spark-defaults.conf 파일 로드

한 가지 옵션에 대한 설정이 다중으로 되었을 시, 위와 같은 순위로 우선시 됨

Apache Spark - 2. Configuration file 설정

Spark Application properties - 1. SparkConf()

```
val conf = new SparkConf().setMaster("yarn-client").setAp
val sc = new SparkContext(conf)
```

Apache Spark - 2. Configuration file 설정

Spark Application properties - 2. spark-submit 실행 매개변수

```
./bin/spark-submit \
  --class <main-class> \
  --master <master-url> \
  --deploy-mode <deploy-mode> \
  --conf <key>=<value> \
  ... # other options
  <application-jar> \
  [application-arguments]
```

```
start-thriftserver.sh --master yarn --conf spark.default.parallelism=20 --conf spark.sql.crossJoin.enabled=true --conf spark.network.timeout=800 --conf spark.rpc.askTimeout=700 --conf spark.sql.broadcastTimeout=800 --conf spark.shuffle.service.enabled=true --driver-memory 16g --num-executors 21 --executor-cores 2 --executor-memory 6g  --jars=$ADDB_SRC_DIR/target/addb-srconnector-0.0.1-jar-with-dependencies.jar
```

Apache Spark - 2. Configuration file 설정

Spark Application properties - 3. spark-defaults.conf 파일 로드

```
# Example:
# spark.master                     spark://master:7077
# spark.eventLog.enabled           true
# spark.eventLog.dir               hdfs://namenode:8021/directory
# spark.serializer                 org.apache.spark.serializer.KryoSerializer
# spark.driver.memory              5g
# spark.executor.extraJavaOptions  -XX:+PrintGCDetails -Dkey=value -Dnumbers="one two three"
#spark.sql.hive.metastore.version 1.1.2
#spark.sql.hive.metastore.jars maven
```

https://spark.apache.org/docs/latest/configuration.html

Apache Spark - 2. Configuration file 설정

2. spark-env.sh

```
# This file is sourced when running various Spark programs.
# Copy it as spark-env.sh and edit that to configure Spark for your site.
export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_121
export HADOOP_HOME=/home/addb/hadoop-2.7.7
export SPARK_HOME=/usr/local/spark/spark-2.0.2

export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export YARN_CONF_DIR=$HADOOP_HOME/etc/hadoop

#export SPARK_MASTER_PORT=40400
# Options read when launching programs locally with
# ./bin/run-example or ./bin/spark-submit
# - HADOOP_CONF_DIR, to point Spark towards Hadoop configuration files
# - SPARK_LOCAL_IP, to set the IP address Spark binds to on this node
# - SPARK_PUBLIC_DNS, to set the public dns name of the driver program
# - SPARK_CLASSPATH, default classpath entries to append

# Options read by executors and drivers running inside the cluster
# - SPARK_LOCAL_IP, to set the IP address Spark binds to on this node
# - SPARK_PUBLIC_DNS, to set the public DNS name of the driver program
# - SPARK_CLASSPATH, default classpath entries to append
# - SPARK_LOCAL_DIRS, storage directories to use on this node for shuffle and RDD data
# - MESOS_NATIVE_JAVA_LIBRARY, to point to your libmesos.so if you use Mesos

# Options read in YARN client mode
# - HADOOP_CONF_DIR, to point Spark towards Hadoop configuration files
# - SPARK_EXECUTOR_INSTANCES, Number of executors to start (Default: 2)
# - SPARK_EXECUTOR_CORES, Number of cores for the executors (Default: 1).
# - SPARK_EXECUTOR_MEMORY, Memory per Executor (e.g. 1000M, 2G) (Default: 1G)
# - SPARK_DRIVER_MEMORY, Memory for Driver (e.g. 1000M, 2G) (Default: 1G)
```

Apache Spark - 2. Configuration file 설정

3. log4j.properties

```
# Set everything to be logged to the console
log4j.rootCategory=INFO, console
#log4j.rootCategory=WARN, console
log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.target=System.err
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.ConversionPattern=%d{yy/MM/dd HH:mm:ss} %p %c{1}: %m%n

# Set the default spark-shell log level to WARN. When running the spark-shell, the
# log level for this class is used to overwrite the root logger's log level, so that
# the user can have different defaults for the shell and regular Spark apps.
log4j.logger.org.apache.spark.repl.Main=WARN

# Settings to quiet third party logs that are too verbose
log4j.logger.org.spark_project.jetty=WARN
log4j.logger.org.spark_project.jetty.util.component.AbstractLifeCycle=ERROR
log4j.logger.org.apache.spark.repl.SparkIMain$exprTyper=INFO
log4j.logger.org.apache.spark.repl.SparkILoop$SparkILoopInterpreter=INFO
log4j.logger.org.apache.parquet=ERROR
log4j.logger.parquet=ERROR

# SPARK-9183: Settings to avoid annoying messages when looking up nonexistent UDFs in SparkSQL with Hive support
log4j.logger.org.apache.hadoop.hive.metastore.RetryingHMSHandler=FATAL
log4j.logger.org.apache.hadoop.hive.ql.exec.FunctionRegistry=ERROR

# ADDB
log4j.logger.org.apache.spark.sql.hive.thriftserver=INFO
log4j.logger.org.apache.hive.service.server=INFO

# Disabling Logging
#log4j.logger.org=OFF
~
```

Apache Spark - 3. jar 파일 build

1. Maven parent로 한 번에 build

2. ADDB-jedis build 후, ADDB-SRConnector build

Apache Spark - 3. jar 파일 build

1. Maven parent로 한 번에 build

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>kr.ac.yonsei.delab</groupId>
  <artifactId>addb-parent</artifactId>
  <version>0.0.1</version>
  <packaging>pom</packaging>

  <name>ADDB Parent</name>
<!--
  <url>http://delab.yonsei.ac.kr</url>
-->
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  <description> ADDB Parent
Including modules:= ADDB-SparkRedisConnector, ADDB-Jedis
</description>

  <modules>
  <module>addb-jedis</module>
  <module>addb-SRConnector</module>
  </modules>
```

$ cd ${ADDB_SPARK_HOME}

$ mvn clean install

Apache Spark - 3. jar 파일 build

2. ADDB-jedis build 후, ADDB-SRConnector build

$ cd ${ADDB_SPARK_HOME}/addb-jedis

$ mvn clean install // addb-jedis를 maven local repository에 저장하기 위해

$ cd ../addb-SRConnector

$ mvn clean package // 더이상 maven dependency를 찾지 않고
target directory 아래의 jar 파일만을 사용

Apache Spark - 4. Thrift server

spark에서 기본으로 제공해주는 thrift server 사용 (JDBC server)

```
[addb@cluster04 sbin]$ pwd
/usr/local/spark/spark-2.0.2/sbin
[addb@cluster04 sbin]$ ls
slaves.sh          start-history-server.sh          start-slave.sh          stop-master.sh                stop-slaves.sh
spark-config.sh    start-master.sh                  start-slaves.sh         stop-mesos-dispatcher.sh      stop-thriftserver.sh
spark-daemon.sh    start-mesos-dispatcher.sh        start-thriftserver.sh   stop-mesos-shuffle-service.sh
spark-daemons.sh   start-mesos-shuffle-service.sh   stop-all.sh             stop-shuffle-service.sh
start-all.sh       start-shuffle-service.sh         stop-history-server.sh  stop-slave.sh
[addb@cluster04 sbin]$
```

thrift server를 실행할 때, SRConnector jar 파일을 옵션으로 넣음

```
    start-thriftserver.sh --master yarn --conf spark.default.parallelism=20 --conf spark.sql.crossJoin.enabled=true --conf spark.netw
ork.timeout=800 --conf spark.rpc.askTimeout=700 --conf spark.sql.broadcastTimeout=800 --conf spark.shuffle.service.enabled=true --dri
ver-memory 16g --num-executors 21 --executor-cores 2 --executor-memory 6g  --jars=$ADDB_SRC_DIR/target/addb-srconnector-0.0.1-jar-wit
h-dependencies.jar
```

```
CREATE TABLE customer
(c_custkey INTEGER , c_name VARCHAR(25) , c_address VARCHAR(40) , c_nationkey INTEGER ,
c_phone CHAR(15) , c_acctbal DECIMAL(15,2) , c_mktsegment CHAR(10) , c_comment VARCHAR(117) )
USING kr.ac.yonsei.delab.addb_srconnector
OPTIONS (host "                ", port "8000", table "6", partitions "c_nationkey");
```

Apache Spark - 4. Thrift server

thrift server 작동 확인

```
[addb@cluster04 addb-spark]$ jps
31873 Jps
5542 NameNode
5835 SecondaryNameNode
6031 ResourceManager
[addb@cluster04 addb-spark]$ addb_spark -start

## ADDB Spark - Start thrift server with custom resource setting

starting org.apache.spark.sql.hive.thriftserver.HiveThriftServer2, logging to /usr/local/spark/spark-2.0.2/logs/spark-addb-org.apache
.spark.sql.hive.thriftserver.HiveThriftServer2-1-cluster04.out

## Check Whether running SparkSubmit (jps)
31922 SparkSubmit
5542 NameNode
32023 Jps
5835 SecondaryNameNode
6031 ResourceManager
[addb@cluster04 addb-spark]$
```

thrift server log path => /usr/local/spark/spark-2.0.2/logs

Hadoop에 비해 Thrift-server는 완전히 켜지는데에 시간이 더욱 필요함

Apache Spark - 5. Beeline 사용법 (JDBC client)

1. Thrift server 실행
2. Beeline 접속(spark 기본 제공): JDBC connection 링크, ID, PW 필요
3. SQL command line처럼 접속

Apache Spark - 5. Beeline 사용법 (JDBC client)

```
[addb@cluster04 addb-spark]$ addb_spark -connect

## ADDB Spark - Connect JDBC beeline
Please enter this:
!connect jdbc:hive2://cluster04:10000
Beeline version 1.2.1.spark2 by Apache Hive
beeline> !connect jdbc:hive2://cluster04:10000
Connecting to jdbc:hive2://cluster04:10000
Enter username for jdbc:hive2://cluster04:10000: addb
Enter password for jdbc:hive2://cluster04:10000: ****
19/05/28 10:17:28 INFO Utils: Supplied authorities: cluster04:10000
19/05/28 10:17:28 INFO Utils: Resolved authority: cluster04:10000
19/05/28 10:17:28 INFO HiveConnection: Will try to open client transport with JDBC Uri: jdbc:hive2://cluster04:10000
Connected to: Spark SQL (version 2.0.2)
Driver: Hive JDBC (version 1.2.1.spark2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://cluster04:10000> show databases;
+---------------+--+
| databaseName  |
+---------------+--+
| default       |
| tpch100g      |
| tpch10g       |
+---------------+--+
3 rows selected (0.268 seconds)
0: jdbc:hive2://cluster04:10000>
```

beeline 주요 command := !connect !q

Apache Spark - 6. 실행 방법

1. HDFS, YARN실행 (Namenode/Datanode, ResourceManager/NodeManager)
2. connector jar파일 준비
3. Spark cluster의 resource 설정
4. Thrift server 실행
5. beeline 접속

Redis, RocksDB

1. Prerequisite
2. Configuration file 설정
3. Cluster 설정
4. LIBRARY PATH 설정
5. 실행 방법

Redis, RocksDB - 1. Prerequisite

1. Ruby gem
2. ~~RocksDB prerequisite~~

Redis, RocksDB - 2. Configuration file 설정

```
[addb@cluster05 conf]$ pwd
/home/addb/addb-RR/conf
[addb@cluster05 conf]$ ls
1.conf  5.conf       nodes3.conf     redis8001.conf  redis8005.conf  redis8009.conf           redis_tiering_8003.conf
2.conf  nodes0.conf  nodes4.conf     redis8002.conf  redis8006.conf  redis_tiering_8000.conf  redis_tiering_8004.conf
3.conf  nodes1.conf  nodes5.conf     redis8003.conf  redis8007.conf  redis_tiering_8001.conf  redis_tiering_8005.conf
4.conf  nodes2.conf  redis8000.conf  redis8004.conf  redis8008.conf  redis_tiering_8002.conf
[addb@cluster05 conf]$
```

redis_tiering_8000.conf

```
cluster-config-file /home/addb/addb-RR/conf/nodes0.conf
pidfile /home/addb/addb-RR/dir/run/redis_8000.pid
logfile /home/addb/addb-RR/dir/log/redis_8000.log
dir /home/addb/addb-RR/dir/ssd1/8000
cluster-enabled yes
tiering_enabled yes
#bind 127.0.0.1
```

conf파일 자동 수정 스크립트 구현목록 :

```
# Configuration
[ -AOF <yes | no> ]
[ -CV  <columnvector_size> ]
[ -offCV ]
[ -IP ]
[ -loglevel <notice | verbose | debug | warning> ]
[ -memory <maxmemory> ]
[ -rewrite <yes | no>]
[ -RG <rowgroup_size>]
[ -ziplist <hash-max-ziplist-entries>]
```

Redis, RocksDB - 2. Configuration file 설정

redis_tiering_8000.conf

```
dir /home/addb/addb-RR/dir/ssd1/8000
```

```
[addb@cluster05 dir]$ pwd
/home/addb/addb-RR/dir
[addb@cluster05 dir]$ ls
log  run  ssd1  ssd2
[addb@cluster05 dir]$ lsblk
NAME        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda           8:0    0 931.5G  0 disk
├─sda1        8:1    0     1G  0 part /boot
└─sda2        8:2    0 930.5G  0 part
  ├─cl-root 253:0    0    50G  0 lvm  /
  ├─cl-swap 253:1    0  30.3G  0 lvm  [SWAP]
  └─cl-home 253:2    0 850.2G  0 lvm  /home
sdb           8:16   0   2.7T  0 disk
sdc           8:32   0   2.7T  0 disk /home/skt_test/hadoopFile/data1
sdd           8:48   0 238.5G  0 disk /home/addb/addb-RR/dir/ssd1
sde           8:64   0 238.5G  0 disk /home/addb/addb-RR/dir/ssd2
sdf           8:80   0   2.7T  0 disk /home/skt_test/hadoopFile/data2
nvme0n1     259:0    0 953.9G  0 disk /home/skt_test/hadoopFile/data
[addb@cluster05 dir]$
```

/etc/fstab

```
# /etc/fstab
# Created by anaconda on Thu Jan  5 17:16:22 2017
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
/dev/mapper/cl-root     /                       xfs     defaults        0 0
UUID=b13302af-fd33-40a9-80b0-44b6a78eba10 /boot                   xfs     defaults        0 0
/dev/mapper/cl-home     /home                   xfs     defaults        0 0
/dev/mapper/cl-swap     swap                    swap    defaults        0 0
/dev/sdd    /home/addb/addb-RR/dir/ssd1   ext4  defaults  0 0
/dev/sde    /home/addb/addb-RR/dir/ssd2   ext4  defaults  0 0
```

Redis, RocksDB - 3. Cluster 설정

Redis 명령어를 통해서 cluster를 생성할 수는 있으나, 그 과정이 복잡하여 ruby로 script화 시켜놓아진 파일이 있음 ⇒ src/redis-trib.rb

1. cluster 상태 초기화

redis-cli >> cluster info

redis-cli >> cluster reset hard

2. cluster 구축 (Host 역할을 할 하나의 서버에서만 실행)

$ ./src/redis-trib.rb create ${HOST_1}:${PORT_1} ${HOST_1}:${PORT_2} ...

Redis, RocksDB - 4. LD_LIBRARY_PATH 설정

Redis build시 일부 라이브러리에 의한 error 발생 ⇒ RocksDB 관련 library일 확률이 높음

```
# ADDB shard library load
export ADDB_HOME=/home/addb/addb-RR
export LD_LIBRARY_PATH=$ADDB_HOME/deps/rocksdb:$ADDB_HOME/deps/jemalloc/lib:/usr/local/lib
```

$ vim ~/.bashrc

$ source ~/.bashrc

이미 rocksdb directory에서 필요 library들을 가지고 있는 것으로 보임

⇒ 따라서 기존의 RocksDB를 설치하기 위해 필요했던 library들은 굳이 설치하지 않아도 작동함

Redis, RocksDB - 5. 실행 방법

1. directory와 conf 파일 모두 설정
2. 각 redis instance 실행 (다른 remote server 포함)
3. cluster 생성 (slot 할당 등 통신에 시간 소요)
4. Redis dbsize command와 RocksDB의 데이터 디렉토리의 변경 확인

설정 시 발생할 수 있는 문제점

1. Namenode, Datanode, tmp directory, Redis log/run/data directory 등 모든 directory 및 기본 파일이 준비되어있어야하고, device mount에 따른 permission 수정이 필요
2. Hadoop

(1) Namenode/Datanode가 잘 안뜨면 시스템 재시작에 의한 방화벽 설정 의심

(2) 기본적으로 cluster에 대한 ID값을 시스템 /tmp 디렉토리에 저장하는데, Hadoop을 실행하고 오랜시간이 지나면 이 ID값이 소멸될 수 있음 => 직접 process kill 후 재실행

(3) Hadoop 혹은 thrift-server를 실행할 때, 각 서버의 시스템 시간에 오차가 클 경우 실행되지 않을 수 있음 => 시스템 시간 설정 필요

(4) 현재 tbl파일을 load할 때, 각 디렉토리당 하나의 tbl을 로드하도록 했음

설정 시 발생할 수 있는 문제점

3. Spark

(1) thrift-server를 실행했는데도 Spark-Submit이 뜨지 않는다면 반드시 log 파일부터 확인할 것. 여러가지 이유가 얽혀있을 가능성이 큼(+HDFS,YARN)

(2) thrift-server가 켜졌는데 beeline에 접속이 안될 때가 있음 => thrift-server가 활성화되는데까지 시간이 필요하여, 길게는 10~20초 후 beeline에 재접속하면 됨

(3) 현재 ADDB에 데이터를 삽입할 때, HDFS에 저장한 데이터를 INSERT (SELECT) 문으로 실행함. 따라서 HDFS 테이블 먼저 setup할 것 (tbl파일을 바로 load하고싶으나, 현재 그러한 SQL interface를 제공해주지 않음)

설정 시 발생할 수 있는 문제점

4. Redis - RocksDB

(1) Redis cluster 생성은 /etc/hosts에 명명한 호스트 닉네임이 아닌 IP를 직접 입력해야함

(2) Redis는 필요한 파일/디렉토리만 미리 setup해두면 실행하는 것에 큰 문제가 없음

(3) Redis build시에 RocksDB library 관련 오류가 나면, LD_LIBRARY_PATH를 설정해주고 source 명령어를 수행했는지 확인

(4) Cluster 생성이나 node 추가 시, 이미 redis instance에 데이터가 있으면 수행되지 않을 수 있음 (redis slot 조정에 따른 충돌 방지)

설정 시 발생할 수 있는 문제점

5. 전체 실험 시

(1) nmon으로 CPU, Memory, Disk 점유를 확인하되, Memory가 swap-space를 사용하는지 확인해야함 => 사용하게 될 시 Redis나 Spark executor 메모리 설정을 낮춤

(2) Spark-YARN의 Dynamic allocation은 켜지 말 것. => Redis의 메모리 영역을 침범

(3) 매 측정 당 CPU cache는 비워 줄 것