1. 自动化Shell脚本

1.1xcall

远程执行bash指令

```
#!/bin/bash

if [ $# -lt 1 ]
then
        echo not enough arguments
fi

for HOST in master slave1 slave2
do
        echo ========== $HOST ========
        CMD=""
        for I in $@
        do
                CMD="$CMD$I "
        done
        echo $CMD
        ssh $HOST "$CMD"
done
```

1.2 jpsall

查看所有节点服务器所有正在运行的Java进程

```
#!/bin/bash

for host in master slave1 slave2
do
        echo ========== $host ==========
        ssh $host "source /etc/profile;jps"
done
```

1.3 xsync

集群同步文件

```
#!/bin/bash

if [ $# -lt 1 ]
then
        echo Not Enough Argument!
        exit;
fi
# 集群所有节点的主机名
for host in master slave1 slave2
do
        echo ============== $host ==============
```

```
        for file in $@
        do
                if [ -e $file ]
                then
                        pdir=$(cd -P $(dirname $file); pwd)    # -P 防止加入软链接路
径
                        fname=$(basename $file)
                        ssh $host "mkdir -p $pdir"
                        rsync -av $pdir/$fname $host:$pdir    # 由于dest主机的目标目
录是绝对路径，所以需要确定$pdir
                else
                        echo $file dose not exists!
                fi
        done
done
```

1.4 zk

开关ZooKeeper集群

```
#!/bin/bash

case $1 in
"start"){
        for host in master slave1 slave2
        do
                echo ------------ zookeeper $host 启动 ---------------
                ssh $host "source
/etc/profile;/usr/local/src/zookeeper/bin/zkServer.sh start"
        done
}
;;
"stop"){
        for host in master slave1 slave2
        do
               echo ------------ zookeeper $host 停止 ---------------
               ssh $host "source
/etc/profile;/usr/local/src/zookeeper/bin/zkServer.sh stop"
        done
}
;;
"status"){
        for host in master slave1 slave2
        do
               echo ------------ zookeeper $host 状态 ---------------
               ssh $host "source
/etc/profile;/usr/local/src/zookeeper/bin/zkServer.sh status"
        done
}
;;
*) echo Not exist the instruction
;;
esac
```

1.5 format-ha

初始化HA集群

```
#!/bin/bash

for HOST in master slave1 slave2
do
        echo "========== delete data and logs in $HOST =========="
        ssh $HOST "rm -rf /usr/local/src/hadoop/data /usr/local/src/hadoop/logs"
done

echo "------- create journalnode --------"
xcall "hdfs --daemon start journalnode"

echo "------- format namenode -------"
ssh master "hdfs namenode -format"
ssh master "hdfs --daemon start namenode"
ssh slave1 "hdfs namenode -bootstrapStandby"
ssh slave2 "hdfs namenode -bootstrapStandby"
ssh master "/usr/local/src/hadoop/sbin/stop-dfs.sh"
zk start
hdfs zkfc -formatZK
zk stop
```

1.6 hadoop-ha

开关HA集群

```
#!/bin/bash

if [ $# -lt 1 ]
then
        echo "No Args Input ..."
        exit;
fi

if [ $# -gt 1 ]
then
        echo "Args Exceeded limit"
        exit;
fi

case $1 in
"start")
        echo "============== 启动 hadoop 集群 =============="

        echo "------------- start ZooKeeper -----------"
        ssh master "/root/bin/zk start"

        echo "------------- start Journalnode ------------"
        ssh master "/root/bin/xcall hdfs --daemon start journalnode"

        echo "------------- 启动 HDFS --------------"
        ssh master "/usr/local/src/hadoop/sbin/start-dfs.sh"

        echo "------------- 启动 yarn --------------"
        ssh slave1 "/usr/local/src/hadoop/sbin/start-yarn.sh"
```

```
        echo "------------- 启动 historyserver -------------"
        ssh master "/usr/local/src/hadoop/bin/mapred --daemon start
historyserver"
;;
"stop")
        echo "============== 关闭 hadoop 集群 ==============="

        echo "------------- 关闭 historyserver -------------"
        ssh master "/usr/local/src/hadoop/bin/mapred --daemon stop
historyserver"

        echo "------------- 关闭 yarn --------------"
        ssh slave1 "/usr/local/src/hadoop/sbin/stop-yarn.sh"

        echo "------------- 关闭 HDFS --------------"
        ssh master "/usr/local/src/hadoop/sbin/stop-dfs.sh"

        echo "------------- stop ZooKeeper -------------"
        ssh master "/r/bin/zk stop"
;;
*)
        echo "Input Args Error..."
;;
esac
```

1.7 spark.sh

开关spark ha

```
#!/bin/bash

case $1 in
"start"){
        echo ---------- spark HA 8989 启动 ------------
        ssh master "source /etc/profile && /usr/local/src/spark/sbin/start-
all.sh"
        echo ---------- spark history 18080 启动 ------------
        ssh master "source /etc/profile && /usr/local/src/spark/sbin/start-
history-server.sh"
        ssh slave1 "source /etc/profile && /usr/local/src/spark/sbin/start-
master.sh"
};;
"stop"){

        echo ---------- spark HA 8989 停止 ------------
        ssh slave1 "source /etc/profile && /usr/local/src/spark/sbin/stop-
master.sh"
        echo ---------- spark history 18080 停止 ------------
        ssh master "source /etc/profile && /usr/local/src/spark/sbin/stop-
history-server.sh"
        ssh master "source /etc/profile && /usr/local/src/spark/sbin/stop-
all.sh"

};;
esac
```

2. hadoop配置文件

2.1 core-site.xml

```
<configuration>

    <!-- 把多个 NameNode 的地址组装成一个集群 mycluster -->
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://mycluster</value>
    </property>

    <!-- 指定 hadoop 运行时产生文件的存储目录 -->
    <property>
        <name>hadoop.tmp.dir</name>
        <value>/usr/local/src/hadoop/data</value>
    </property>

    <!-- 指定zkfc要连接的zkServer地址 -->
    <property>
        <name>ha.zookeeper.quorum</name>
        <value>master:2181,slave1:2181,slave2:2181</value>
    </property>

    <!-- 配置 HDFS 网页登录使用的静态用户为 root -->
    <property>
        <name>hadoop.http.staticuser.user</name>
        <!-- hadoop用户 -->
        <value>root</value>
    </property>

    <!-- 整合 hive -->
    <property>
        <name>hadoop.proxyuser.root.hosts</name>
        <value>*</value>
    </property>

    <property>
        <name>hadoop.proxyuser.root.groups</name>
        <value>*</value>
    </property>
</configuration>
```

2.2 hdfs-site.xml

```
<configuration>

    <!-- NameNode 数据存储目录 -->
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>file://${hadoop.tmp.dir}/name</value>
    </property>

    <!-- DataNode 数据存储目录 -->
    <property>
```

```
        <name>dfs.datanode.data.dir</name>
        <value>file://${hadoop.tmp.dir}/data</value>
    </property>

    <!-- JournalNode 数据存储目录 -->
    <property>
        <name>dfs.journalnode.edits.dir</name>
        <value>${hadoop.tmp.dir}/jn</value>
    </property>

    <!-- 完全分布式集群名称 -->
    <property>
        <name>dfs.nameservices</name>
        <value>mycluster</value>
    </property>

    <!-- 集群中 NameNode 节点都有哪些 -->
    <property>
        <name>dfs.ha.namenodes.mycluster</name>
        <value>nn1,nn2,nn3</value>
    </property>

    <!-- NameNode 的 RPC 通信地址 -->
    <property>
        <name>dfs.namenode.rpc-address.mycluster.nn1</name>
        <value>master:8020</value>
    </property>
    <property>
        <name>dfs.namenode.rpc-address.mycluster.nn2</name>
        <value>slave1:8020</value>
    </property>
    <property>
        <name>dfs.namenode.rpc-address.mycluster.nn3</name>
        <value>slave2:8020</value>
    </property>

    <!-- NameNode 的 http 通信地址 -->
    <property>
        <name>dfs.namenode.http-address.mycluster.nn1</name>
        <value>master:9870</value>
    </property>
    <property>
        <name>dfs.namenode.http-address.mycluster.nn2</name>
        <value>slave1:9870</value>
    </property>
    <property>
        <name>dfs.namenode.http-address.mycluster.nn3</name>
        <value>slave2:9870</value>
    </property>

    <!-- 指定 NameNode 元数据在 JournalNode 上的存放位置 -->
    <property>
        <name>dfs.namenode.shared.edits.dir</name>
        <value>qjournal://master:8485;slave1:8485;slave2:8485/mycluster</value>
    </property>

    <!-- 访问代理类：client 用于确定哪个 NameNode 为 Active -->
    <property>
```

```
        <name>dfs.client.failover.proxy.provider.mycluster</name>

 <value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvide
r</value>
    </property>

    <!-- 配置隔离机制，即同一时刻只能有一台服务器对外响应 -->
    <property>
        <name>dfs.ha.fencing.methods</name>
        <value>sshfence</value>
    </property>

    <!-- 使用隔离机制时需要 ssh 秘钥登录-->
    <property>
        <name>dfs.ha.fencing.ssh.private-key-files</name>
        <value>/roor/.ssh/id_rsa</value>
    </property>

    <!-- 启动nn故障自动转移 -->
    <property>
        <name>dfs.ha.automatic-failover.enabled</name>
        <value>true</value>
    </property>

    <property>
        <name>dfs.webhdfs.enabled</name>
        <value>true</value>
    </property>

</configuration>
```

2.3 mapred-site.xml

```
<configuration>

    <!-- 指定 MapReduce 程序运行在 Yarn 上 -->
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>

    <!-- 历史服务器端地址 -->
    <property>
        <name>mapreduce.jobhistory.address</name>
        <value>master:10020</value>
    </property>

    <!-- 历史服务器 web 端地址 -->
    <property>
        <name>mapreduce.jobhistory.webapp.address</name>
        <value>master:19888</value>
    </property>

</configuration>
```

2.4 yarn-site.xml

```
<configuration>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>

    <!-- 启用 resourcemanager ha -->
    <property>
        <name>yarn.resourcemanager.ha.enabled</name>
        <value>true</value>
    </property>

    <!-- 声明 resourcemanager 的地址 -->
    <property>
        <name>yarn.resourcemanager.cluster-id</name>
        <value>cluster-yarn1</value>
    </property>

    <!--指定 resourcemanager 的逻辑列表-->
    <property>
        <name>yarn.resourcemanager.ha.rm-ids</name>
        <value>rm1,rm2,rm3</value>
    </property>
    <!-- ========== rm1 的配置 ========== -->
    <!-- 指定 rm1 的主机名 -->
    <property>
        <name>yarn.resourcemanager.hostname.rm1</name>
        <value>master</value>
    </property>
    <!-- 指定 rm1 的 web 端地址 -->
    <property>
        <name>yarn.resourcemanager.webapp.address.rm1</name>
        <value>master:8088</value>
    </property>
    <!-- 指定 rm1 的内部通信地址 -->
    <property>
        <name>yarn.resourcemanager.address.rm1</name>
        <value>master:8032</value>
    </property>
    <!-- 指定 AM 向 rm1 申请资源的地址 -->
    <property>
        <name>yarn.resourcemanager.scheduler.address.rm1</name>
        <value>master:8030</value>
    </property>
    <!-- 指定供 NM 连接的地址 -->
    <property>
        <name>yarn.resourcemanager.resource-tracker.address.rm1</name>
        <value>master:8031</value>
    </property>
    <!-- ========== rm2 的配置 ========== -->
    <!-- 指定 rm2 的主机名 -->
    <property>
        <name>yarn.resourcemanager.hostname.rm2</name>
        <value>slave1</value>
    </property>
    <!-- 指定 rm2 的 web 端地址 -->
    <property>
```

```
        <name>yarn.resourcemanager.webapp.address.rm2</name>
        <value>slave1:8088</value>
    </property>
    <!-- 指定 rm2 的内部通信地址 -->
    <property>
        <name>yarn.resourcemanager.address.rm2</name>
        <value>slave1:8032</value>
    </property>
    <!-- 指定 rm2 的内部通信地址 -->
    <property>
        <name>yarn.resourcemanager.scheduler.address.rm2</name>
        <value>slave1:8030</value>
    </property>
    <!-- 指定供 NM 连接的地址 -->
    <property>
        <name>yarn.resourcemanager.resource-tracker.address.rm2</name>
        <value>slave1:8031</value>
    </property>
    <!-- ========== rm3 的配置 ========== -->
    <!-- 指定 rm3 的主机名 -->
    <property>
        <name>yarn.resourcemanager.hostname.rm3</name>
        <value>slave2</value>
    </property>
    <!-- 指定 rm3 的 web 端地址 -->
    <property>
        <name>yarn.resourcemanager.webapp.address.rm3</name>
        <value>slave2:8088</value>
    </property>
    <!-- 指定 rm3 的内部通信地址 -->
    <property>
        <name>yarn.resourcemanager.address.rm3</name>
        <value>slave2:8032</value>
    </property>
    <!-- 指定 AM 向 rm3 申请资源的地址 -->
    <property>
        <name>yarn.resourcemanager.scheduler.address.rm3</name>
        <value>slave2:8030</value>
    </property>
    <!-- 指定供 NM 连接的地址 -->
    <property>
        <name>yarn.resourcemanager.resource-tracker.address.rm3</name>
        <value>slave2:8031</value>
    </property>
    <!-- 指定 zookeeper 集群的地址 -->
    <property>
        <name>yarn.resourcemanager.zk-address</name>
        <value>master:2181,slave1:2181,slave2:2181</value>
    </property>

    <!-- 启用自动恢复 -->
    <property>
        <name>yarn.resourcemanager.recovery.enabled</name>
        <value>true</value>
    </property>

    <!-- 指定 resourcemanager 的状态信息存储在 zookeeper 集群 -->
    <property>
```

```
        <name>yarn.resourcemanager.store.class</name>

 <value>org.apache.hadoop.yarn.server.resourcemanager.recovery.ZKRMStateStore</v
alue>
    </property>

    <!-- 环境变量的继承 -->
    <property>
        <name>yarn.nodemanager.env-whitelist</name>

 <value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSPATH_
PREPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_MAPRED_HOME</value>
    </property>

    <!-- 开启日志聚集功能 -->
    <property>
        <name>yarn.log-aggregation-enable</name>
        <value>true</value>
    </property>

    <!-- 设置日志聚集服务器地址 -->
    <property>
        <name>yarn.log.server.url</name>
        <value>http://master:19888/jobhistory/logs</value>
    </property>

    <!-- 设置日志保留时间为7天 -->
    <property>
        <name>yarn.log-aggregation.retain-seconds</name>
        <value>604800</value>
    </property>
</configuration>
```

3. hive配置文件

3.1 `hive-site.xml`(slave1)

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
    <!-- Hive 默认在 HDFS 的工作目录 -->
    <property>
        <name>hive.metastore.warehouse.dir</name>
        <value>/user/hive/warehouse</value>
    </property>
    <!-- jdbc 连接的 URL -->
    <property>
        <name>javax.jdo.option.ConnectionURL</name>
        <value>jdbc:mysql://master:3306/hive?
createDatabaseIfNotExist=true</value>
    </property>
    <!-- jdbc 连接的 Driver-->
    <property>
        <name>javax.jdo.option.ConnectionDriverName</name>
        <value>com.mysql.cj.jdbc.Driver</value>
    </property>
```

```
    <!-- jdbc 连接的 username-->
    <property>
        <name>javax.jdo.option.ConnectionUserName</name>
        <value>root</value>
    </property>
    <!-- jdbc 连接的 password -->
    <property>
        <name>javax.jdo.option.ConnectionPassword</name>
        <value>123456</value>
    </property>
    <property>
        <!--是否是本地模式，false-->
        <name>hive.metastore.local</name>
        <value>false</value>
    </property>
    <!-- 指定存储元数据要连接的地址 -->
    <property>
        <name>hive.metastore.uris</name>
        <value>thrift://slave1:9083</value>
    </property>
    <!-- 指定 hiveserver2 连接的 host -->
    <property>
        <name>hive.server2.thrift.bind.host</name>
        <value>slave1</value>
    </property>
    <!-- 指定 hiveserver2 连接的端口号 -->
    <property>
        <name>hive.server2.thrift.port</name>
        <value>10000</value>
    </property>
    <!-- Hive 元数据存储版本的验证 -->
    <property>
        <name>hive.metastore.schema.verification</name>
        <value>false</value>
    </property>
    <!--元数据存储授权-->
    <property>
        <name>hive.metastore.event.db.notification.api.auth</name>
        <value>false</value>
    </property>

    <!--在hive提示符中包含当前数据库-->
    <property>
        <name>hive.cli.print.current.db</name>
        <value>true</value>
    </property>
    <!--在查询输出中打印列的名称-->
    <property>
        <name>hive.cli.print.header</name>
        <value>true</value>
    </property>
</configuration>
```

3.2 `hive-site.xml` (master, slave2)

```
<?xml version="1.0"?>
```

```xml
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
    <!-- 指定存储元数据要连接的地址 -->
    <property>
        <name>hive.metastore.uris</name>
        <value>thrift://slave1:9083</value>
    </property>
    <!-- 指定 hiveserver2 连接的 host -->
    <property>
        <name>hive.server2.thrift.bind.host</name>
        <value>slave1</value>
    </property>
    <!-- 指定 hiveserver2 连接的端口号 -->
    <property>
        <name>hive.server2.thrift.port</name>
        <value>10000</value>
    </property>
    <!--在hive提示符中包含当前数据库-->
    <property>
        <name>hive.cli.print.current.db</name>
        <value>true</value>
    </property>
    <!--在查询输出中打印列的名称-->
    <property>
        <name>hive.cli.print.header</name>
        <value>true</value>
    </property>
</configuration>
```