Hadoop HA 集群搭建

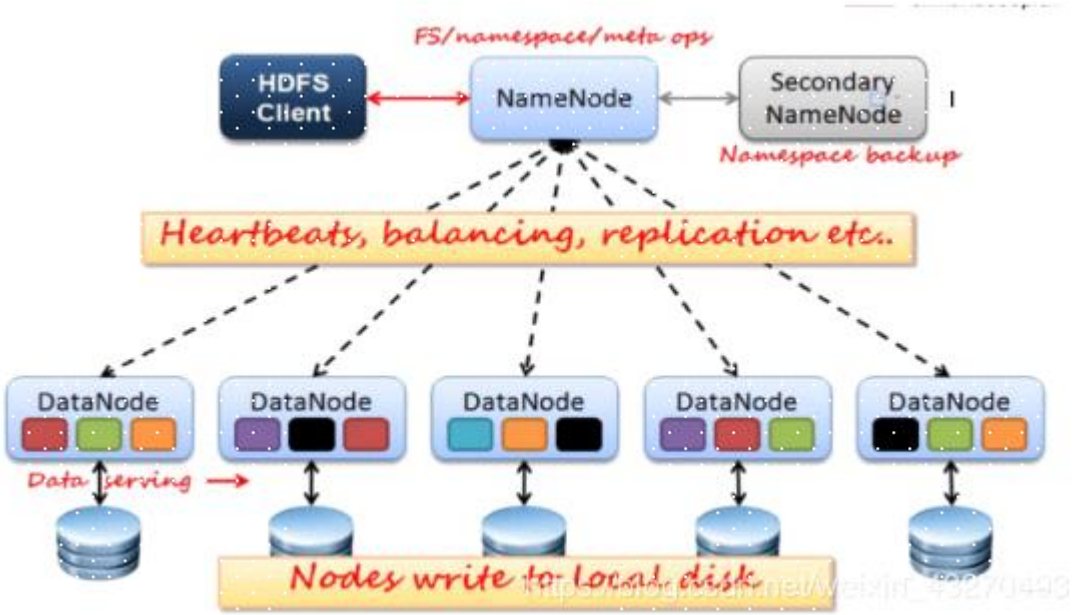
2020年3月17日 星期二
12:16

Hadoop的HA搭建及Yarn简单理解

一，Hadoop2.0产生背景

之间我们搭建了 hadoop 1.0 版本 完全分布式，详情见博客–>完全分布式搭建HDFS

回顾1.0版本中搭建的架构图，如下：



存在以下问题：

– Hadoop 1.0 中 HDFS和MapReduce在高可用，扩展性方面存在问题。

HDFS存在的问题:

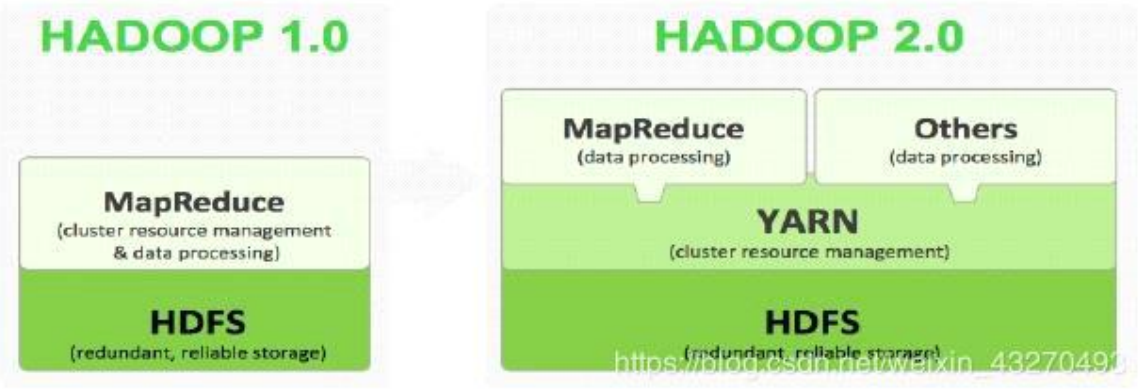
NameNode单点故障问题，NameNode一旦挂掉，就失去了作业能力，难以应用在生产环境

NameNode压力过大，且内存受限，影响了系统的扩展性

MapReduce存在的问题:

JobTracker访问压力过大，影响系统扩展性难以支持除MapReduce之外的其他计算框架，如 spark storm等

以下是hadoop1.0 和 hadoop 2.0 的区别分析图:



hadoop2.0以后 由 HDFS MapReduce Yarn 三个分支架构

HDFS: 分布式文件存储系统

Yarn : 分布式资源管理系统

MapReduce : 运行在 Yarn上的 MR

二，Hadoop 2.x

2.1 解决单点故障
通过主备NameNode解决，如果主NameNode发生故障，则切换到备用的NameNode上

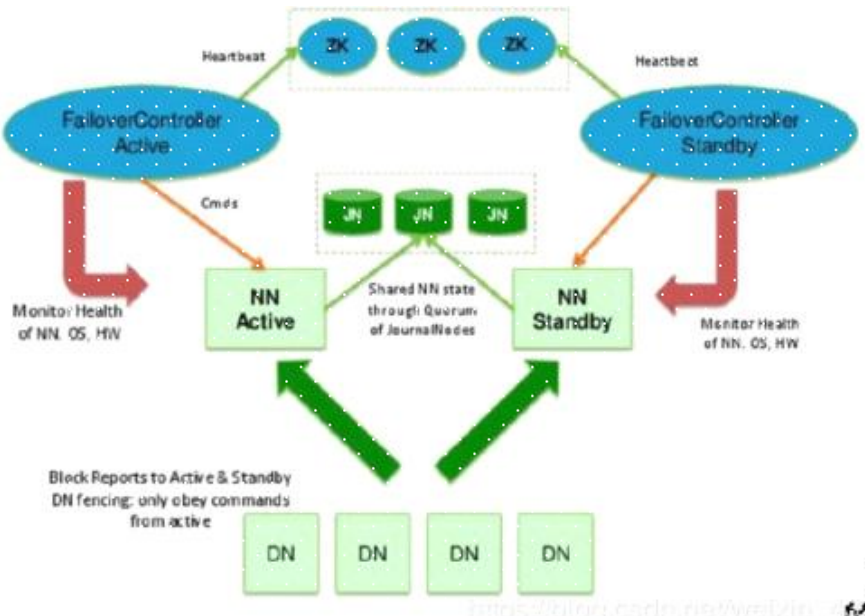
2.2 解决内存受限问题
Fsderation(联邦机制)，HA

2.x 支持2个NN节点的HA ,3.x实现了NN一主多从

2.x仅仅是架构上发生了改变，使用方式不变

对HDDS使用者透明 hdfs1.x中的命令和 API依旧可以使用

三，HDFS2.0 HA高可用
3.1 2.x的 HA
HDFS的高可用架构图：



可以看到，HA架构中，存在两个NameNode(一个是Active，一个是Standby)

为什么两个的状态不能同时是Active?

如果两个NameNode的状态都是Active,同时工作，同时对客户端进行读写服务，这样会造成 brain boom DataNode向两个NameNode汇报文件的存储信息，保证数据的一致性很重要

工作中的NameNode会实时的产生edits.log文件，那是怎么实现两个NameNode的一致性呢？

这里hadoop2.0采用了 journalNode集群来存放NameNode的元信息数据，这样处于standby的NameNode可以动态的去到journalNode中去获取最新的数据元信息，从而保持数据的一致性？

在生产环境中，Active的NameNode挂掉，Standby的NameNode怎么做到立刻切换状态呢？

采用了zookeeper分布式服务协调来操作，为每个NameNode的节点服务器上，有监听NameNode状态的zkfc，一旦其中一个挂掉，zkfc立马汇报给zookeeper集群，集群通知standbyNameNode，这时候standbyNameNode不会立刻去切换成为Active状态，(防止脑裂 brain-spilt) 会发送ssh指定(杀死对方nameNode)，长时间无反应，它也会执行预先定义好的 shell脚本，根据执行的结果判断是否启动

3.2 Federation 联邦

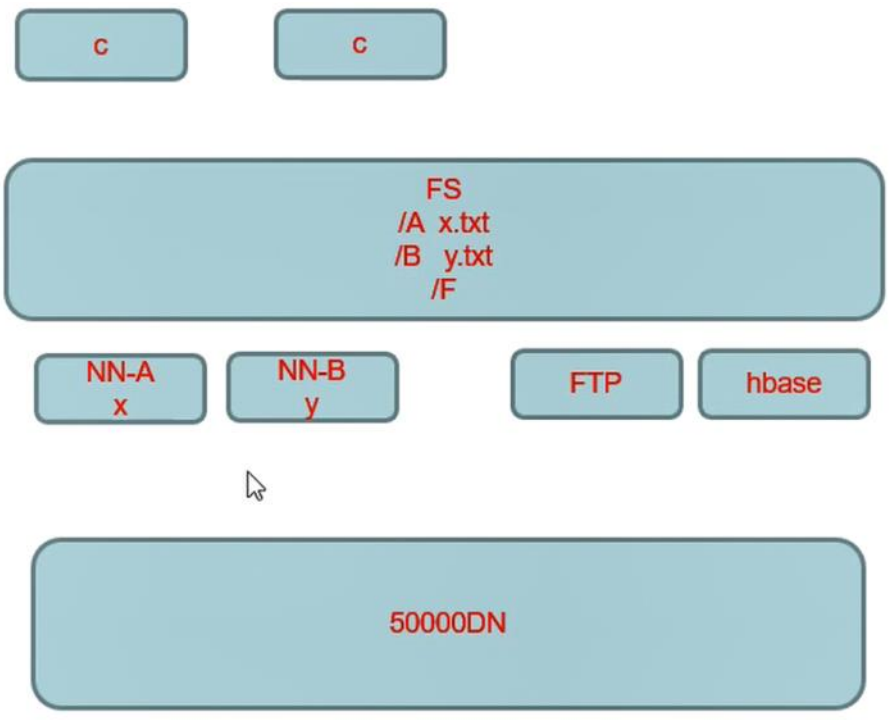
通过多个 namenode/namespace把元数据的存储和管理分散到多个节点中，使到namenode

/namespace 可以通过增加机器来进行水平扩展。

能把单个namenode的负载分散到多个节点中，在HDFS数据规模较大的时候不会也降低

HSFS的性能，可以通过多个namespace来隔离不同类型的应用，把不同类型应用的HDFS

元数据的存储和管理分派到不同的namenode中。



四，Yarn介绍

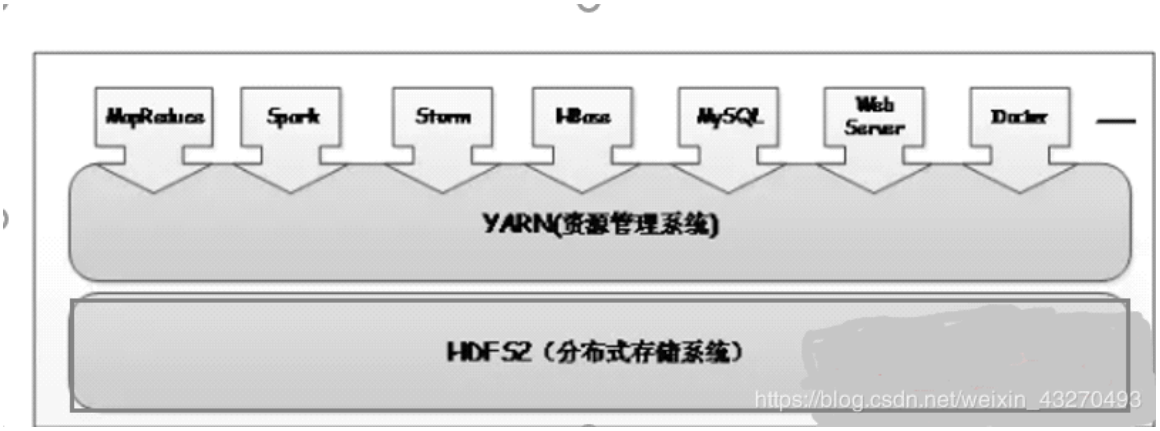
hadoop 2.0 新引入的资源管理系统 ，直接从MRv1演化而来的。

核心思想： 将MRv1中的 JobTracker的资源管理和任务调度两个功能分开，分别是

ResourceManager 和 ApplicationMaster进程实现。

ResourceManager ： 负责整个集群的资源管理和调度。

ApplicationMaster ： 负责应用程序相关的事务，比如集群的任务调度，任务监控，容错等。

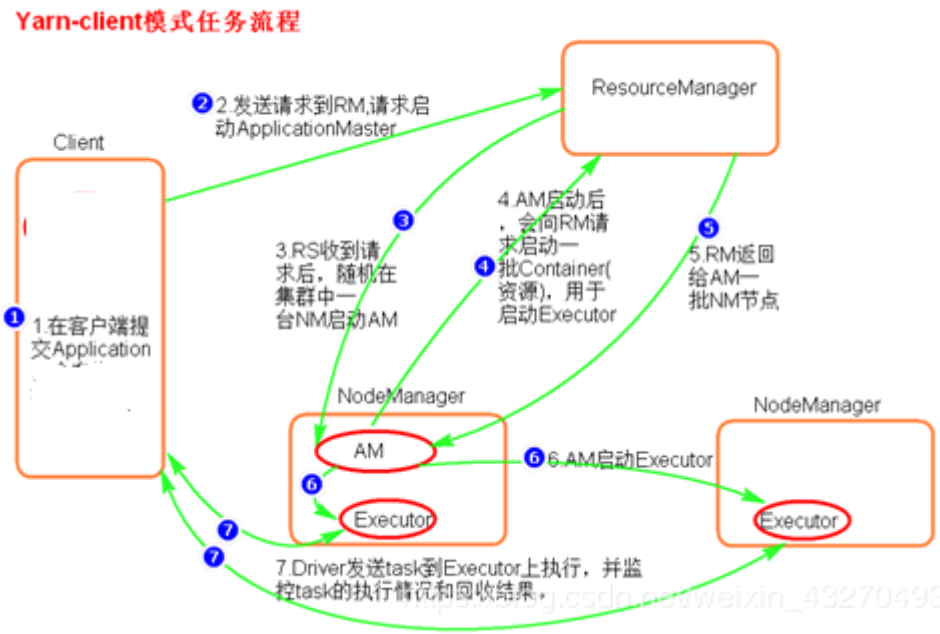


Yarn的引入使得多个计算框架可运行在一个集群当中

每个应用程序对应一个 ApplicationMaster(应用程序的主人)

目前多个计算框架可以运行在Yarn上，如 MapReduce Spark Storm等…

Yarn资源管理任务调度流程;



1. 客户端提交Application(应用程序)
2. 客户端发送请求给ResourceManager，请求启动ApplicationMaster
3. ResourceManager收到客户端请求后，会随机的在集群节点中为其分配一个NodeManager(其实就是DataNode节点，也就是所谓的计算向数据靠拢)启动ApplicationMaster
4. ApplicationMaster启动后，会根据任务向ResourceManager申请启动一批资源(Container),用于启

动Executor
5. ResourceManager返回给ApplicationMaster一批NodeManager节点，ApplicationMaster去响应的NodeManager上开启一个Container，用于执行Executor
6. ApplicationMaster启动Executor执行应用程序
7. 程序发送task到Executor上执行，并监控tasj的执行情况和回收结果，在此过程中，,ApplicationMaster时刻关注Executor的任务进度，同时它也会向ResourceManager汇报任务的进度
8. 任务执行完毕，Executor向ApplicationMaster汇报，ApplicationMaster向ResourceManager汇报，回收相关资源

五，MapResuce On Yarn
Yarn ： 负责资源管理和调度

MRApplicationMaster ： 负责任务切分，任务调度，任务的监控和容错

MapTask/ReduceTask ： 任务驱动引擎，和MRv1一样

每个MapReduce作业对应一个MRAppMaster任务调度

Yarn将资源分配给MRAppMaster

MRAppMaster进一步将资源分配给内部的任务

MRAppMaster容错

失败后，由Yarn重新启动

任务失败后，MRAppMaster重新申请资源

六，Hadoop 2.x 集群搭建
节点分布如下图：
6.1 上传解压缩
这里省略去步骤，详情见博客–> 完全分布式搭建HDFS

6.2 配置Hadoop-env.sh
这里配置JAVA_HOME的位置信息

6.3 配置core-site.xml

```
<configuration>
<property>
  <name>fs.defaultFS</name>
  <value>hdfs://wcb</value>
</property>
<property>
   <name>ha.zookeeper.quorum</name>
   <value>node01:2181,node02:2181,node03:2181</value>
</property>
<property>
  <name>hadoop.tmp.dir</name>
  <value>/opt/hadoop</value>
</property>
</configuration>
```

https://blog.csdn.net/weixin_43270493

配置解释：

<configuration>

```
    <!-HA模式下对应的 NameNode工作组 可以自定义名称 -->
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://hacluster</value>
     </property>
    <!--配置zookeeper集群相关节点  有多少配置多少  逗号隔开即可-->
     <property>
        <name>hadoop.tmp.dir</name>
        <value>/opt/hadoop</value>
     </property>
    <!--hadooop 生成文件的路径  启动后可在此下查看相关内容-->
     <property>
        <name>ha.zookeeper.quorum</name>
        <value>node1:2181,node2:2181,node3:2181</value>
     </property>
</configuration>
```

6.4 hdfs-site.xml 配置

这个图太长了，我就不截图了，直接上配置解释

```
<configuration>

 <!--HA配置 -->
 <property>
     <name>dfs.nameservices</name>
     <value>hacluster</value>
 </property>
 <property>
     <name>dfs.ha.namenodes.hacluster</name>
     <value>nn1,nn2</value>
 </property>

 <!--namenode1 RPC端口 -->
 <property>
     <name>dfs.namenode.rpc-address.hacluster.nn1</name>
     <value>node1:9000</value>
 </property>

 <!--namenode1 HTTP端口 -->
 <property>
     <name>dfs.namenode.http-address.hacluster.nn1</name>
     <value>node1:50070</value>
 </property>

  <!--namenode2 RPC端口 -->
  <property>
      <name>dfs.namenode.rpc-address.hacluster.nn2</name>
      <value>node2:9000</value>
  </property>

  <!--namenode1 HTTP端口 -->
  <property>
      <name>dfs.namenode.http-address.hacluster.nn2</name>
      <value>node2:50070</value>
  </property>

  <!-- journalnode 配置 -->
```

```
  <property>
      <name>dfs.namenode.shared.edits.dir</name>
      <value>qjournal://node2:8485;node3:8485;node4:8485/hacluster</value>
  </property>

  <!-- 指定HDFS客户端连接active namenode的java类 -->
  <property>
      <name>dfs.client.failover.proxy.provider.hacluster</name>
      <value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>
  </property>

  <!--发生failover时，Standby的节点要执行一系列方法把原来那个Active节点中不健康的NameNode服务给杀掉，这个叫做fence过程。sshfence会通过ssh远程调用fuser命令去找到Active节点的NameNode服务并杀死它-->
 <property>
     <name>dfs.ha.fencing.methods</name>
     <value>shell(/bin/true)</value>
  </property>

  <!--SSH私钥 -->
  <property>
      <name>dfs.ha.fencing.ssh.private-key-files</name>
      <value>/root/.ssh/id_rsa</value>
  </property>

  <!-- 指定journalnode日志文件存储的路径 -->
  <property>
      <name>dfs.journalnode.edits.dir</name>
      <value>/opt/hadoop/data</value>
  </property>

  <!-- 开启自动故障转移 -->
  <property>
      <name>dfs.ha.automatic-failover.enabled</name>
      <value>true</value>
  </property>

</configuration>
```

6.5 配置slaves

```
node02
node03
node04
```

不要有空格，写完一个，换行

6.6 zookeeper准备

6.6.1 上传解压zookeeper

6.6.2 修改zoo.cfg配置文件

拷贝zoosimple.cfg到所在目录，命名为zoo.cfg

修改dataDir配置文件

```
# sending a request and getting an acknowledgement syncLimit=5
# the directory where the snapshot is stored.
# do not use /tmp for storage, /tmp here is just
# example sakes.

//数据存放路径，不建议在 /tmp下 因为是临时文件目录
dataDir=/opt/zookeeper

# the port at which the clients will connect
clientPort=2181
# the maximum number of client connections.
# increase this if you need to handle more clients
#maxClientCnxns=60
#
# Be sure to read the maintenance section of the
# administrator guide before turning on autopurge.
#
# http://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc_maintenance
#
# The number of snapshots to retain in dataDir
#autopurge.snapRetainCount=3
# Purge task interval in hours
# Set to "0" to disable auto purge feature
#autopurge.purgeInterval=1
//server.1 代表 服务器的节点标识   node01 代表对应的ip
//2888代表zookeeper监听端口
//3888代表leader挂掉后，leader的选举端口
server.1=node01:2888:3888
server.2=node02:2888:3888
server.3=node03:2888:3888
```

6.6.3 在DataDir目录下新建 myid文件，文件内容分别为1,2,3
在此之前，你应该拷贝解压配置好的Hadoop和Zookeeper去另外两个节点,例如
scp -r hadoop node02:`pwd`

6.7 配置hadoop环境变量和zookeeper环境变量
如图:

```
# User specific environment and startup programs

PATH=$PATH:$HOME/bin
JAVA_HOME=/usr/java/jdk1.8.0_191-amd64

PATH=$PATH:$JAVA_HOME/bin
HADOOP_HOME=/home/hadoop-2.6.5

PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin

ZOOKEEPER_HOME=/home/zookeeper-3.4.6

PATH=$PATH:$ZOOKEEPER_HOME/bin

export PATH
~
```

https://blog.csdn.net/weixin_43270493

所有节点都得配置，配置完别忘记 source .bash_profile

我这边配置的局部环境变量，你也可以配置全局的环境变量，在/etc/profile中配置

6.8 启动三个 zookeeper

启动前，先关闭防火墙— systemctl stop iptables.service

然后三个节点执行

zkServer.sh start

启动完成通过jps查看是否启动成功，jps显示有不一定代表显示成功，但是jps都不显示

说明你肯定没启动成功

QuorumPeerMain—>对应的就是zookeeper服务

6.9 启动三个journalNode

journalNode集群是负责存放editslog的，是保证NameNode之间数据一致性的关键

hadoop-daemon.sh start journalnode

6.10 在其中一个namenode上格式化

仅第一次启动前需要格式化nameNode

hdfs namenode -format

6.11 把刚刚格式化之后的元数据拷贝到另一个namenode上

启动刚刚格式化完的 namenode
hadoop-daemon.sh start namenode

在没有格式化的namenode上执行
hdfs namenode -bootstrapStandby

启动第二个NameNode
hadoop-daemon.sh start namenode

6.12 在其中一个NameNode上初始化zkfc

仅第一次需要
hdfs zkfc -formatZK

6.13 停止以上节点
stop-dfs.sh

6.14 全面启动
start-dfs.sh

6.15 启动Yarn
yarn-daemon.sh start resourcemanager

6.16 易错点

1， 确认每台机器防火墙均关掉

2， 确认每台机器的时间是一致的

3， 确认配置文件无误，并且确认每台机器上面的配置文件一样

4， 如果还有问题想重新格式化，那么先把所有节点的进程关掉

5， 删除之前格式化的数据目录hadoop.tmp.dir属性对应的目录，所有节点同步都删掉，别单删掉之前的一个，删掉三台JN节点中dfs.journalnode.edits.dir属性所对应的目录

6， 接上面的第6步又可以重新格式化已经启动了

7， 最终Active Namenode停掉的时候，StandBy可以自动接管！

七，关于brain-split(脑裂)
脑裂是指在主备切换的时候，由于切换不彻底或者其他原因，导致客户端和slave误以为出现两个

active状态的NameNode,最终使得整个集群处于混乱状态。

解决脑裂，通常采用隔离(Fencing机制),包括三个方面

共享存储fencing: 确保只有一个Master在共享存储中写数据

客户端fencing: 确保只有一个Master可以响应客户端的请求

slave fencing: 确保这有一个Master可以向slave下发命令

Hadoop公共库中对外提供了两种fencing实现:

sshFence: 通过ssh登陆到目标Master节点上，使用命令杀死进程

shellFence(缺省实现):执行一个用户事先定义的shell脚本，完成隔离