**大数据介绍**

**– 随着计算机技术的发展,互联网的普及,信息的积累已经到了一个非常庞大的地步,信息的增长也在不断的加快,随着互联网、物联网建设的加快,信息更是爆炸是增长,收集、检索、统计这些信息越发困难,必须使用新的技术来解决这些问题**

**大数据的定义**

**– 大数据指无法在一定时间范围内用常规软件工具进行捕捉、管理和处理的数据集合,需要新处理模式才能具有更强的决策力、洞察发现力和流程优化能力的海量、高增长率和多样化的信息资产**

**-- 是指从各种各样类型的数据中,快速获得有价值的信息**

**• 大数据能做什么**

**– 企业组织利用相关数据分析帮助他们降低成本、提高效率、开发新产品、做出更明智的业务决策等**

**– 把数据集合并后进行分析得出的信息和数据关系性,用来察觉商业趋势、判定研究质量、避免疾病扩散、打击犯罪或测定即时交通路况等**

**– 大规模并行处理数据库,数据挖掘电网,分布式文件系统或数据库,云计算平和可扩展的存储系统等**

**大数据特性**

****

**• 大数据的5V特性是什么?**

**– (V)olume (大体量)**

**可从数百TB到数十数百PB、甚至EB的规模**

**– (V)ariety(多样性)**

**大数据包括各种格式和形态的数据**

**– (V)elocity(时效性)**

**很多大数据需要在一定的时间限度下得到及时处理**

**– (V)eracity(准确性)**

**处理的结果要保证一定的准确性**

**– (V)alue(大价值)**

**大数据包含很多深度的价值,大数据分析挖掘和利用将带来巨大的商业价值**

**• Hadoop是什么**

**– Hadoop是一种分析和处理海量数据的软件平台**

**– Hadoop是一款开源软件,使用JAVA开发**

**– Hadoop可以提供一个分布式基础架构**

**• Hadoop特点**

**– 高可靠性、高扩展性、高效性、高容错性、低成本**

**Hadoop起源**

**• 2003年开始Google陆续发表了3篇论文**

**– GFS,MapReduce,BigTable**

**• GFS**

**– GFS是一个可扩展的分布式文件系统,用于大型的、分布式的、对大量数据进行访问的应用**

**– 可以运行于廉价的普通硬件上,提供容错功能**

**• MapReduce**

**– MapReduce是针对分布式并行计算的一套编程模型,由Map和Reduce组成,Map是映射,把指令分发到多个worker上,Reduce是规约,把worker计算出的结果合并**

**BigTable**

**– BigTable是存储结构化数据**

**– BigTable建立在GFS,Scheduler,Lock Service和MapReduce之上**

**– 每个Table都是一个多维的稀疏图**

**• GFS、MapReduce和BigTable三大技术被称为Google的三驾马车,虽然没有公布源码,但发布了这三个产品的详细设计论**

**• Yahoo资助的Hadoop,是按照这三篇论文的开源Java实现的,但在性能上Hadoop比Google要差很多**

**– GFS - - -> HDFS**

**– MapReduce - - -> MapReduce**

**– BigTable - - -> Hbase**

**Hadoop常用组件**

**• HDFS:Hadoop分布式文件系统(核心组件)**

**• MapReduce:分布式计算框架(核心组件)**

**• Yarn:集群资源管理系统(核心组件)**

**• Zookeeper:分布式协作服务**

**• Hbase:分布式列存数据库**

**• Hive:基于Hadoop的数据仓库**

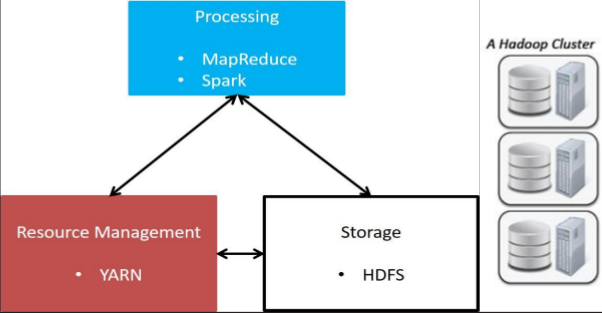
**• Sqoop:数据同步工具**

**• Pig:基于Hadoop的数据流系统**

**• Mahout:数据挖掘算法库**

**• Flume:日志收集工具**

**Hadoop核心组件**

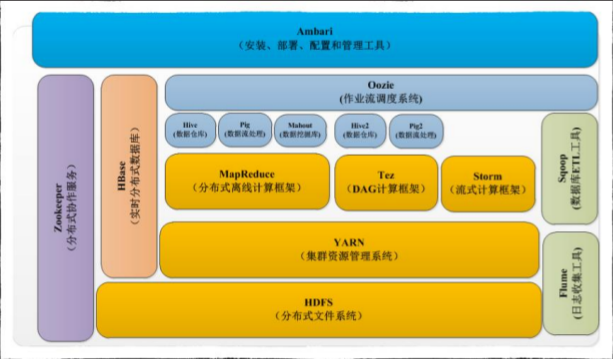
****

**布式文件系统**

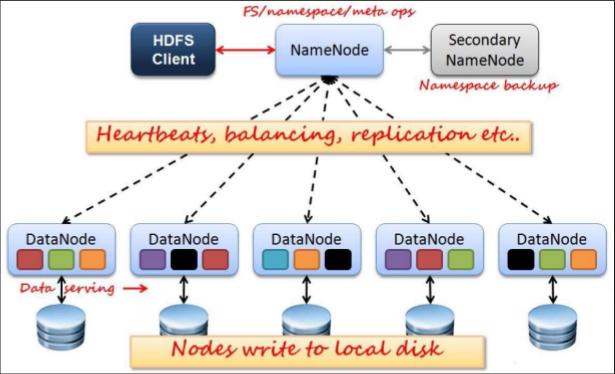
**集群资源管理系统**

**:分布式计算框架**

**Hadoop生态系统**

****

**HDFS结构**

****

**HDFS角色及概念**

**• Hadoop体系中数据存储管理的基础,是一个高度容错的系统,用于在低成本的通用硬件上运行**

**• 角色和概念**

**– Client**

**– Namenode**

**– Secondarynode**

**– Datanode**

**• NameNode**

**– Master节点,管理HDFS的名称空间和数据块映射信息,配置副本策略,处理所有客户端请求**

**• Secondary NameNode**

**– 定期合并fsimage名称空间 和fsedits更改变更日志,推送给NameNode**

**– 紧急情况下,可辅助恢复NameNode**

**• 但Secondary NameNode并非NameNode的热备**

**• DataNode**

**– 数据存储节点,存储实际的数据**

**– 汇报存储信息给NameNode**

**• Client**

**– 切分文件 原则：每块128M**

**– 访问HDFS**

**– 与NameNode交互,获取文件位置信息**

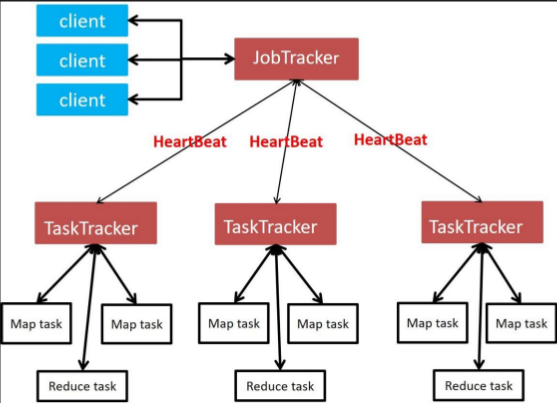
**– 与DataNode交互,读取和写入数据**

**• Block**

**– 每块缺省128MB大小**

**– 每块可以多个副本**

**MapReduce结构**

****

**MapReduce角色及概念**

**• 源自于Google的MapReduce论文,JAVA实现的分布式计算框架**

**• 角色和概念**

**– JobTracker**

**– TaskTracker**

**– Map Task**

**– Reducer Task**

**• JobTracker**

**– Master节点只有一个**

**– 管理所有作业/任务的监控、错误处理等**

**– 将任务分解成一系列任务,并分派给TaskTracker**

**• TaskTracker**

**– Slave节点,一般是多台**

**– 运行Map Task和Reduce Task**

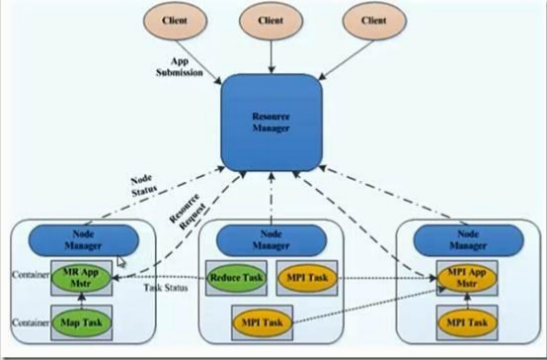
**– 并与JobTracker交互,汇报任务状态**

**• Map Task:解析每条数据记录,传递给用户编写的map()并执行,将输出结果写入本地磁盘**

**– 如果为map-only作业,直接写入HDFS**

**• Reducer Task:从Map Task的执行结果中,远程读取输入数据,对数据进行排序,将数据按照分组传递给用户编写的reduce函数执行**

**Yarn结构**

****

**• Yarn是Hadoop的一个通用的资源管理系统**

**• Yarn角色**

**– Resourcemanager**

**– Nodemanager**

**– ApplicationMaster**

**– Container**

**– Client**

**• ResourceManager**

**– 处理客户端请求**

**– 启动/监控ApplicationMaster**

**– 监控NodeManager**

**– 资源分配与调度**

**• NodeManager**

**– 单个节点上的资源管理**

**– 处理来自ResourceManager的命令**

**– 处理来自ApplicationMaster的命令**

**• Container**

**– 对任务运行行环境的抽象,封装了CPU 、内存等**

**– 多维资源以及环境变量、启动命令等任务运行相关的信息资源分配与调度**

**• ApplicationMaster**

**– 数据切分**

**– 为应用程序申请资源,并分配给内部任务**

**– 任务监控与容错**

**• Client**

**– 用户与Yarn交互的客户端程序**

**– 提交应用程序、监控应用程序状态,杀死应用程序等**

**• Yarn的核心思想**

**• 将JobTracker和TaskTacker进行分离,它由下面几大构成组件**

**– ResourceManager一个全局的资源管理器**

**– NodeManager每个节点(RM)代理**

**– ApplicationMaster表示每个应用**

**– 每一个ApplicationMaster有多个Container在NodeManager上运行**

**Hadoop安装与配置**

**Hadoop模式**

**• Hadoop的部署模式有三种**

**– 单机 – 伪分布式 – 完全分布式**

**• Hadoop的单机模式安装非常简单**

**安装配置Java环境,安装jps工具**

**[root@hadoop-nn01 ~]# yum -y install java-1.8.0-openjdk-devel(jps的包) java-openjdk java-openjdk-devel**

**[root@hadoop-nn01 hadoop]# pwd**

**/usr/local/hadoop/etc/hadoop**

**[root@hadoop-nn01 hadoop]# vim hadoop-env.sh**

**export JAVA\_HOME="/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.131-11.b12.el7.x86\_64/jre/"**

**export HADOOP\_CONF\_DIR="/usr/local/hadoop/etc/hadoop"**

**[root@hadoop-nn01 hadoop]# rpm -ql java-1.8.0-openjdk 查看安装路径**

**[root@hadoop-nn01 hadoop]# cd /usr/local/hadoop/**

**[root@hadoop-nn01 hadoop]# ./bin/hadoop version**

**Hadoop 2.7.6**

**Subversion https://shv@git-wip-us.apache.org/repos/asf/hadoop.git -r 085099c66cf28be31604560c376fa282e69282b8**

**Compiled by kshvachk on 2018-04-18T01:33Z**

**Compiled with protoc 2.5.0**

**From source with checksum 71e2695531cb3360ab74598755d036**

**This command was run using /usr/local/hadoop/share/hadoop/common/hadoop-common-2.7.6.jar**

**– 测试 --- 统计词频**

**[root@hadoop-nn01 hadoop]# mkdir oo**

**[root@hadoop-nn01 hadoop]# cp \*.txt oo**

**[root@hadoop-nn01 oo]# ls**

**LICENSE.txt NOTICE.txt README.txt**

**[root@hadoop-nn01~hadoop]#./bin/hadoop (参数jar包)jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.6.jar wordcount oo xx 统计oo的 结果放在xx下**

**[root@hadoop-nn01 hadoop]# cat xx/\* 查看结果**

**完全分布式**

**Hadoop配置文件及格式**

**• 文件格式**

**– Hadoop-env.sh**

**JAVA\_HOME**

**HADOOP\_CONF\_DIR**

**– xml文件配置格式**

**<property>**

**<name>关键字</name>**

**<value>变量值</value>**

**<description> 描述 </description>**

**</property>**

**系统规划**

|  |  |  |
| --- | --- | --- |
| **主机** | **角色** | **软件** |
| **192.168.1.101**  **Nn01** | **NameNode**  **SecondaryNameNode** | **HDFS** |
| **192.168.1.102**  **Node1** | **DataNode** | **HDFS** |
| **192.168.1.103**  **Node2** | **DataNode** | **HDFS** |
| **192.168.1.104**  **node3** | **DataNode** | **HDFS** |

**新开启3台虚拟机**

**– 禁用 selinux**

**SELINUX=disabled**

**– 禁用 firewalld**

**# systemctl stop firewalld**

**# systemctl mask firewalld**

**– 安装 java-1.8.0-openjdk-devel**

**# yum install -y java-1.8.0-openjdk-devel**

**基础环境准备**

**– 在3台机器上配置/etc/hosts**

**[root@hadoop-nn01 ~]# cat /etc/hosts**

**192.168.1.101 hadoop-nn01**

**192.168.1.102 node1**

**192.168.1.103 node2**

**192.168.1.104 node3**

**– 注意:所有主机都能ping通namenode的主机名,namenode能ping通所有节点**

**– java -version 验证java安装**

**– jps 验证角色**

**布置SSH信任关系**

**[ root@nn01 ~] # vim /etc/ssh/ssh\_config /第一次登陆不需要输入y es**

**Host \***

**GSSAPIAuthentication y es**

**StrictHostKey Checking no 添加**

**[root@hadoop-nn01 ~]# ssh-keygen -t rsa -b 2048 -N '' -f key**

**[root@hadoop-nn01 ~]# ssh- copy - id 192.168.1.$i;**

**https://hadoop.apache.org/docs/r2.7.6/**

**[root@hadoop-nn01 ~]# cd /usr/local/hadoop/etc/hadoop/**

**[root@hadoop-nn01 hadoop]# vim core-site.xml**

**<configuration>**

**<property>**

**<name>fs.defaultFS</name> 文件系统配置参数**

**<value>hdfs://hadoop-nn01:9000</value>**

**</property>**

**<property>**

**<name>hadoop.tmp.dir</name> 数据目录配置参数**

**<value>/var/hadoop</value>**

**</property>**

**</configuration>**

**/tmp/hadoop-${user.name} hadoop的根目录**

**[root@hadoop-nn01 hadoop]# vim hdfs-site.xml**

**<configuration>**

**<property>**

**<name>dfs.namenode.http.address</name> 地址声明**

**<value>hadoop-nn01:50070</value>**

**</property>**

**<property>**

**<name>dfs.namenode.secondary.http-address</name>**

**<value>hadoop-nn01:50090</value>**

**</property>**

**<property>**

**<name>dfs.replication</name>**

**<value>2</value> 指定几份数据 冗余性 文件冗余份数**

**</property>**

**</configuration>**

**[root@hadoop-nn01 hadoop]# vim slaves 节点配置文件**

**node1 只写DataNode节点的主机名称**

**node2**

**node3**

**Hadoop所有节点的配置参数完全一样,在一台配置好后,把配置文件同步到其它所有主机上**

**rsync -aSH --delete /usr/local/hadoop 192.168.1.102:/usr/local/ &**

**-H 保存硬连接 -a归档模式 -H对稀疏文件进行特殊处理以节省DST的空间**

**HDFS完全分布式配置**

**– 在所有机器上创建/var/hadoop文件夹**

**# mkdir /var/hadoop**

**– 在namenode上执行格式化操作**

**# ./bin/hdfs namenode -format**

**– 启动集群**

**# ./sbin/start-dfs.sh**

**NameNode验证**

**[root@nn01 hadoop]# jps**

**29826 SecondaryNameNode**

**31237 Jps**

**29643 NameNode**

**DataNode验证**

**[root@node1 ~]# jps**

**24472 Jps**

**24027 DataNode**

**节点验证**

**– NameNode上**

**– bin/hdfs dfsadmin -report**

**2018-11-27 18-25-25 的屏幕截图**

**课程知识点总结**

**• 大数据的5V特性 • Hadoop是用什么语言开发的**

**– (V)olume (大体量) – JAVA**

**– (V)ariety(多样性) • Hadoop的三大核心组件**

**– (V)elocity(时效性) – Hdfs**

**– (V)eracity(准确性) – MapReduce**

**– (V)alue(大价值) – Yarn**

**完全分布式**

**• Hadoop三大核心组件**

**– 分布式文件系统**

**– HDFS已经部署完毕**

**– 分布式计算框架**

**– MapReduce**

**– 集群资源管理**

**– yarn**

**系统规划**

|  |  |  |
| --- | --- | --- |
| **主机** | **角色 消耗** | **软件** |
| **192.168.1.101**  **Nn01** | **NameNode内存和CPU**  **SecondaryNameNode**  **ResourceManager** | **HDFS**  **YARN** |
| **192.168.1.102**  **Node1** | **DataNode**  **NodeManager** | **HDFS**  **YARN** |
| **192.168.1.103**  **Node2** | **DataNode**  **NodeManager** | **HDFS**  **YARN** |
| **192.168.1.104**  **node3** | **DataNode**  **NodeManager** | **HDFS**  **YARN** |

**Datanode占用磁盘I/O 存储的数据 nodemanager 处理数据占用的CPU**

**mapred部署**

**[root@hadoop-nn01 hadoop]# vim mapred-site.xml**

**<configuration>**

**<property>**

**<name>mapreduce.framework.name</name> 资源管理**

**<value>yarn</value> 集群使用yarn 单机使用local**

**</property>**

**</configuration>**

**[root@hadoop-nn01 hadoop]# vim yarn-site.xml**

**<configuration>**

**<!-- Site specific YARN configuration properties -->**

**<property>**

**<name>yarn.resourcemanager.hostname</name> 地址res..ager**

**<value>hadoop-nn01</value>**

**</property>**

**<property>**

**<name>yarn.nodemanager.aux-services</name>使用哪个计算框架**

**<value>mapreduce\_shuffle</value> 计算框架的名称**

**</property>**

**</configuration>**

**[root@hadoop-nn01 ~]# for i in node{1..3} 将etc同步到其它机器**

**> do**

**> rsync -asH --delete /usr/local/hadoop/etc ${i}:/usr/local/hadoop/ -e 'ssh'**

**> done**

**[root@hadoop-nn01 ~]# cd /usr/local/hadoop/**

**[root@hadoop-nn01 hadoop]# ./sbin/start-yarn.sh**

**[root@hadoop-nn01 hadoop]# jps**

**2880 Jps**

**983 NameNode**

**1177 SecondaryNameNode**

**2623 ResourceManager**

**使用Web访问Hadoop**

**– namenode web页面(nn01)**

**# http://192.168.1.101:50070/**

**– secondory namenode web 页面(nn01)**

**# http://192.168.1.101:50090/**

**– datanode web 页面(node1,node2,node3)**

**# http://192.168.1.102:50075/**

**• 使用Web访问Hadoop**

**– resourcemanager web页面(nn01)**

**# http://192.168.1.101:8088/**

**– nodemanager web页面(node1,node2,node3)**

**# http://192.168.1.102:8042/**

**查看根下文件 空**

**[root@hadoop-nn01 hadoop]# ./bin/hadoop fs -ls /**

**创件文件夹abc**

**[root@hadoop-nn01 hadoop]# ./bin/hadoop fs -mkdir /abc**

**创建文件**

**[root@hadoop-nn01 hadoop]# ./bin/hadoop fs -touchz /readme**

**[root@hadoop-nn01 hadoop]# ./bin/hadoop fs -ls /**

**Found 2 items**

**drwxr-xr-x - root supergroup 0 2018-11-28 11:11 /abc**

**-rw-r--r-- 2 root supergroup 0 2018-11-28 11:14 /readme**

**上传文件**

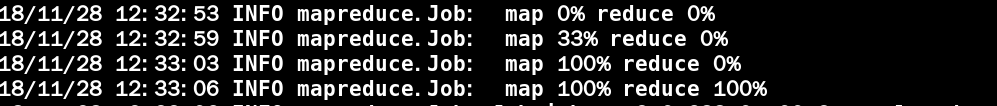
**[root@hadoop-nn01 hadoop]# ./bin/hadoop fs -put \*.txt /abc/**

**下载文件**

**[root@hadoop-nn01 shabi]# /usr/local/hadoop/bin/hadoop fs -get /abc/\*.txt**

**测试分析文件 注意时间同步**

**[root@hadoop-nn01 hadoop]# ./bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.6.jar wordcount /abcc /outbut**

****

**HDFS节点管理**

**• HDFS增加结点**

**– 启动一个新的系统,禁用Selinux、禁用firewalld**

**– 设置SSH免密码登录**

**[root@hadoop-nn01 hadoop]# ssh-copy-id root@192.168.1.105**

**– 在所有节点修改/etc/hosts,增加新节点的主机信息**

**[root@hadoop-nn01 hadoop]# ansibel all -m shell -a “echo newnode >> /etc/hosts”**

**– 安装java运行环境(java-1.8.0-openjdk-devel)**

**– 拷贝NamNode的/usr/local/hadoop到本机**

**[root@hadoop-nn01]# rsync -aSH --delete /usr/local/hadoop root@192.168.1.105:/usr/local/**

**– 修改NameNode的slaves文件增加该节点**

**[root@hadoop-nn01 hadoop]# ansibel all -m shell “echo newnode >>/usr/local/hadoop/etc/hadoop/slaves”**

**– 在该节点启动DataNode**

**[root@newnode hadoop]# ./sbin/hadoop-daemon.sh start datanode**

**HDFS节点管理**

**– 设置同步带宽,并同步数据**

**[root@hadoop-nn01 hadoop]#./bin/hdfs dfsadmin -setBalancerBandwidth 60000000**

**[root@hadoop-nn01 hadoop]#./sbin/start-balancer.sh**

**查看集群状态**

**[root@hadoop-nn01 hadoop]# ./bin/hdfs dfsadmin -report**

**• HDFS修复节点**

**– 修复节点比较简单,与增加节点基本一致**

**– 注意:新节点的ip和主机名要与损坏节点的一致**

**– 启动服务**

**# ./sbin/hadoop-daemon.sh start datanode**

**– 数据恢复是自动的**

**– 上线以后会自动恢复数据,如果数据量非常巨大,可能需要一定的时间**

**HDFS删除节点状态**

**– 配置NameNode的hdfs-site.xml**

**– 增加dfs.hosts.exclude配置**

**<property>**

**<name>dfs.hosts.exclude</name>**

**<value>/usr/local/hadoop/etc/hadoop/exclude</value>**

**</property**

**– 增加exclude配置文件,写入要删除的节点主机名 newnode**

**[root@nn01hadoop]#touch/usr/local/hadoop/etc/hadoop/exclude**

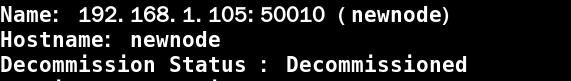
**– 更新数据**

**# ./bin/hdfs dfsadmin -refreshNodes**

**2018-11-28 15-29-48 的屏幕截图**

**– 查看状态**

**# ./bin/hdfs dfsadmin -report**

****

**– Normal:正常状态**

**– Decommissioned in Program:数据正在迁移**

**– Decommissioned:数据迁移完成**

**– 注意:仅当状态变成Decommissioned才能down机下线**

**yarn节点**

**– 由于Hadoop在2.x引入了yarn框架,对于计算节点的**

**操作已经变得非常简单**

**– 增加节点**

**# sbin/yarn-daemon.sh start nodemanager**

**– 删除节点**

**# sbin/yarn-daemon.sh stop nodemanager**

**– 查看节点 (ResourceManager)**

**# ./bin/yarn node -list**

**yarn的系统环境配置与HDFS的基础环境配置是相同的**

**由于yarn不包含数据,所以在增加删除修复节点的时候比较简单,HDFS要注意数据安全**

**NFS网关**

**• NFS 网关用途**

**– 用户可以通过操作系统兼容的本地NFSv3客户端来浏览HDFS文件系统**

**– 用户可以从HDFS文件系统下载文档到本地文件系统**

**– 用户可以通过挂载点直接流化数据,支持文件附加,但是不支持随机写**

**– NFS网关支持NFSv3和允许HDFS作为客户端文件系统的一部分被挂载**

**• 特性**

**– HDFS超级用户是与NameNode进程本身具有相同标识的用户,超级用户可以执行任何操作,因为权限检查永远不会认为超级用户失败**

**• 注意事项**

**– 在非安全模式下,运行网关进程的用户是代理用户**

**– 在安全模式下,Kerberos keytab中的用户是代理用户**

**调试**

**– 在配置NFS网关过程中经常会碰到各种各样的错误,如果出现错误,打开调试日志是一个不错的选择 /usr/local/hadoop/logs**

**• 日志排错(log4j.property)**

**– log4j.logger.org.apache.hadoop.hdfs.nfs=DEBUG**

**– log4j.logger.org.apache.hadoop.oncrpc=DEBUG**

**1 修改/etc/hosts同步给所有主机**

**2 在naemmode nfsgw添加用户**

**[root@nfsgw ~]# groupadd -g 200 nfsuser**

**[root@nfsgw ~]# useradd -u 200 -g 200 -r nfsuser**

**3 配置nfs主机**

**核心配置core-site.xml**

**hadoop.proxyuser.{代理用户}.groups**

**hadoop.proxyuser.{代理用户}.hosts**

**– 这里的{代理用户}是主机上真实运行的nfs3的用户**

**– 在非安全模式下,运行nfs网关的用户为代理用户**

**– groups为挂载点用户所使用的组**

**– hosts为挂载点主机地址**

**<property>**

**<name>hadoop.proxyuser.nsd1802.groups</name>**

**<value>\*</value>**

**</property>**

**<property>**

**<name>hadoop.proxyuser.nsd1802.hosts</name>**

**<value>\*</value>**

**</property>**

**• 配置步骤**

**– 停止集群所有服务**

**# ./sbin/stop-all.sh**

**– 同步配置文件到所有主机**

**– 启动 hdfs**

**# ./sbin/start-dfs.sh**

**[root@nfsgw hadoop]#rsync -aSH --delete hadoop-nn01:/usr/local/hadoop /usr/local/ 同步namemode配置hadoop**

**卸载本机的 rpcbind nfs-utils**

**配置/etc/hosts,添加所有NameNode和DataNode的主机名与ip对应关系**

**配置hdfs-site.xml**

**– 默认情况下,export可以被任何客户端挂载。为了更好的控制访问,可以设置属性。值和字符串对应机器名和访问策略,通过空格来分割。机器名的格式可以是单一的主机、Java的正则表达式或者IPv4地址**

**– 使用rw或ro可以指定导出目录的读写或只读权限。如果访问策略没被提供,默认为只读。每个条目使用";"来分割**

**nfs.dump.dir**

**– 用户需要更新文件转储目录参数。NFS客户端经常重新安排写操作,顺序的写操作会随机到达NFS网关。这个目录常用于临时存储无序的写操作。对于每个文件,无序的写操作会在他们积累在内存中超过一定阈值(如,1M)时被转储。需要确保有足够的空间的目录**

**– 如:应用上传10个100M,那么这个转储目录推荐1GB左右的空间,以便每个文件都发生最坏的情况。只有NFS网关需要在设置该属性后重启**

**<property>**

**<name>nfs.exports.allowed.hosts</name>**

**<value>\* rw</value>**

**</property>**

**<property>**

**<name>nfs.dump.dir</name>**

**<value>/var/nfstmp</value>**

**</property>**

**配置完该属性后要创建/var/nfstmp文件夹 并且把该文件夹的属组改成代理用户**

**chown 200:200 /var/nfstmp**

**设置/usr/local/hadoop/logs权限,为代理用户赋予读写执行的权限**

**setfacl -m user:nfsuser:rwx /usr/local/hadoop/logs/**

**使用root用户启动portmap服务**

**[root@nfsgw hadoop]#./sbin/hadoop-daemon.sh --script ./bin/hdfs start portmap**

**使用代理用户启动nfs3**

**[root@nfsgw hadoop]# su -l nfsuser**

**Bash$4.2# ./sbin/hadoop-daemon.sh --script ./bin/hdfs start nfs3**

**警告**

**– 启动portmap需要使用root用户**

**– 启动nfs3需要使用core-site里面设置的代理用户**

**– 必须先启动portmap之后再启动nfs3**

**– 如果portmap重启了,在重启之后nfs3也需要重启**

**启动与挂载**

**– 目前NFS只能使用v3版本**

**vers=3**

**– 仅使用TCP作为传输协议**

**proto=tcp**

**– 不支持NLM**

**nolock**

**– 禁用access time的时间更新**

**noatime**

**启动一台机器并安装nfs-utils**

**[root@es1 ~]#mount -t nfs -o vers=3,proto=tcp,nolock,noatime,sync 192.168.1.105:/ /mnt/**

**2018-11-28 19-21-12 的屏幕截图**