搭建Hadoop平台

Hadoop的安装部署可分为三种模式: 单机模式(Local Standalone)， 伪分布模式（Pseudo-Distributed），完全分布模式（Fully-Distributed）。

单机模式有时也称为独立模式，运行在单个机器上，在该模式下，不使用分布式文件系统（HDFS），不会启动NameNode、DataNode、JobTracker等守护进程，hadoop运行单个Java 进程来完成任务。这种模式一般用于调式环境。

伪分布模式同样也只运行在单个机器上，但该模式会模拟多个主机以分布式方式来完成hadoop作务，在该模式下，会使用只有一个节点的分布式文件系统，同时作业任务会分配给多个独立的Java进程处理。

完全分布模式需要多机器来实现一个集群系统，此时将使用多节点的分布式文件系统，作业任务也会分配给多个机器进行处理，在生产环境中一般按这种模式进行部署。在完全分布式模式中使用分布式文件系统(HDFS)以提高存储性能和数据冗余能力，同时mapreduce(计算节点)一般与HDFS也部署在一起， 也就是说，计算节点和存储节点通常在一起，这样进行计算时可就近读取分布式文件系统上的数据，高效地调度任务并节约集群带宽资源。

搭建单机模式

因为hadoop是基于java开发的项目，所以要让hadoop运行起来就必须准备JDK环境。目前在Linux环境下有两种JDK可供选择，一个是Oracle的JDK环境，另一个是OpenJDK环境，在CentOS7的仓库源中已有OpenJDK软件，可以直接从仓库源进行安装，如需要安装 Oracle的JDK环境，可从Oracle官方网站下载，需要注意的是不同版本的hadoop对JDK的版本要求也是不同的。这里我们以hadoop 3.2.1版本作为部署目标，hadoop 2.x版本从2.7.x版开始支持Java7和Java8，而hadoop 3.x版本只支持java8以上版本。

关于OpenJDK

OpenJDK是Sun（Sun后来被Oracle收购）在2006年末把Java开源而形成的项目，OpenJDK采用GPL V2协议发布，而JDK则采用JRL协议发布。两个协议虽然都是开源协议，但还有是很大区别，简单来讲以GPL V2发布的内容可以在商业环境中使用，但JRL协议发布的内容仅用于个人研究，不允许商业环境使用。

OpenJDK的内容不如Oracle JDK完整，Oracle JDK还包含了一些不开放的源代码的插件。随着OpenJDK的发展，其性能与Oracle JDK已经没有太大区别。

1. **安装JDK环境**

使用yum search openjdk可发现在CentOS7中有openjkd1.6， openjdk1.7， openjdk1.8，openjkd11四个版本可供选择，这里我们选择openjdk8使用。

```
[hadoop@Master ~]# yum search openjdk
[hadoop@Master ~]# yum install java-1.8.0-openjdk.x86_64
[hadoop@Master ~]# yum install java-1.8.0-openjdk-devel.x86_64
```

2. **创建hadoop运行帐号**

基于安全性的考虑，建议创建一个普通帐号来运行软件，如果按之前的安装方式，在CentOS7安装时已经创建了hadoop帐号，则可跳过此步，否则请使用以下命令创建帐号。

```
useradd hadoop
```

3. **配置Java环境变量**

Java环境变量可以针对整个系统所有用户配置，也可只对使用的用户进行设置。对全局所有帐号配置，可将环境变量放入到/etc/profile或/etc/bashrc文件中，如对单个用户进行设置，可将环境变量放入到$HOME/.bash_profile或$HOME/.bashrc中。

```
[hadoop@Master ~]$vi  .bash_profile
```

```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi

# User specific environment and startup programs
JAVA_HOME=/usr/lib/jvm/java
JRE_HOME=/usr/lib/jvm/jre

export JAVA_HOME
export JRE_HOME

PATH=$PATH:$HOME/bin:$JAVA_HOME/bin:$JRE_HOME/bin

export PATH
alias vi=vim

```

```
[hadoop@Master ~]# yum install vim-enhanced  -y
```

4. **重新登录或手工加载环境变量**

为让环境变量生效，可退出系统然后重新登录，登录时将会自动加载新设置的环境变量，如想让环境变量不退出系统而生效，可采用手工加载方式完成

```
[hadoop@Master ~]$source  .bash_profile
```

验证环境变量是否生效:

```
[hadoop@Master ~]$ env | grep HOME
[hadoop@Master ~]$ env | grep PATH
###如未生效，请重新加载测试并验证
[hadoop@Master ~]$ source .bash_profile
[hadoop@Master ~]$ env | grep HOME
[hadoop@Master ~]$ env | grep PATH
```

5. **下载 hadoop软件包并解压**

Hadoop可从官方站进行下载: https://hadoop.apache.org/releases.html

这里我们直接下载的二进制版本进行使用:

```
[hadoop@Master ~]$ sudo yum install wget -y   #安装wget下载工具
[hadoop@Master ~]$ wget
https://mirrors.tuna.tsinghua.edu.cn/apache/hadoop/common/hadoop-
3.2.1/hadoop-3.2.1.tar.gz
```

将下载软件包解压到/opt目录

```
[hadoop@Master ~]$ sudo tar zxvf hadoop-3.2.1.tar.gz -C /opt
[hadoop@Master ~]$ sudo chown hadoop:hadoop /opt/hadoop* -R
[hadoop@Master ~]$ sudo ln -s  /opt/hadoop-3.2.1 /opt/hadoop
```

编辑hadoop-env.sh配置JAVA环境

```
export  JAVA_HOME=/usr/lib/jvm/java
```

注意:

hadoop软件包大小约300多M，解压后占用空间约900M左右，请保证相关目录有足够空间。

6. **验证hadoop：**

   默认情况下，hadoop工作为单机模式，直接解压后，如果运行环境正常，即可正常工作。使用hadoop自带测试程序可验证hadoop是否工作正常。

   在这里如果JAVA_HOME变量没有置，将会提示以下错误:

   ```
   Error: JAVA_HOME is not set and could not be found.
   ```

A.用样例程序查找字符串验证方法：

```
bin/hadoop jar \
share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.1.jar  \
grep input output 'dfs[a-z.]+'

```

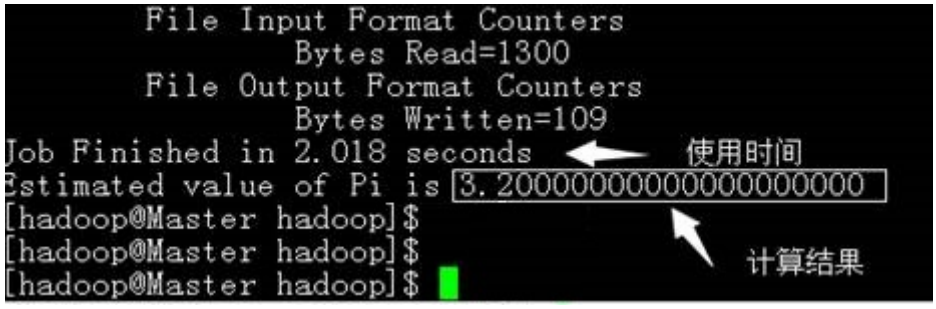
此命令是在当前目录下的子目录input的所有文件中查找dfsa开头的内容，结果将会生成以当前目录 下的output子目录 中。

B.用样例程序计算PI值进行验证：

hadoop计算Pi值

如果一切正常，在最后应该能计算出Pi值为：3.20000000000000000000，图中maps表示要运行的次数，后一个10表示取样数，这两个值作为Pi值算法使用，两者乘积越大，计算的Pi值越精确。

合理的maps数量和每个maps的取样数，可极大的提高处理速度，例如在单机模式下由于只有一个Java进程运行，如果使用1 1000和1000 1虽然都能得到相同的计算结果，但处理速度相差会很大。



检查计算结果

7.设置hadoop工作临时目录

默认情况下hadoop处理任务的临时数据会存放到/tmp/hadoop-目录下，一些操作系统在重启系统时，此目录中的内容会被清空，CentOS7系统会每天清理此目录下10天前的文件，也存在一定数据丢失的可能。如果经常运行一些较大的任务，请将计算数据指定到其它位置。修改数据目录需要编辑core-site.xml文件，增加以下内容：

```
<configuration>
  <property>
     <name>hadoop.tmp.dir</name>
     <value>/data/hdfs/tmp</value>
  </property>
</configuration>

```

hadoop临时目录设置

```
[hadoop@Master hadoop]$ sudo mkdir -p /data/hdfs/tmp
[hadoop@Master hadoop]$ sudo chown hadoop:hadoop  /data/hdfs -R
```

重新运行验证任务，查看是否在/data/tmp目录下产生文件和目录.

```
[hadoop@Master hadoop]$
[hadoop@Master hadoop]$ find /data/tmp
/data/tmp
/data/tmp/mapred
/data/tmp/mapred/staging
/data/tmp/mapred/staging/hadoop1580085120
/data/tmp/mapred/staging/hadoop1580085120/.staging
/data/tmp/mapred/local
/data/tmp/mapred/local/localRunner
/data/tmp/mapred/local/localRunner/hadoop
/data/tmp/mapred/local/localRunner/hadoop/job_local1580085120_0001
/data/tmp/mapred/local/localRunner/hadoop/jobcache
/data/tmp/mapred/local/localRunner/hadoop/jobcache/job_local1580085120_0001
/data/tmp/mapred/local/localRunner/hadoop/jobcache/job_local1580085120_0001/attempt_local1580085120_00
/data/tmp/mapred/local/localRunner/hadoop/jobcache/job_local1580085120_0001/attempt_local1580085120_00
/data/tmp/mapred/local/localRunner/hadoop/jobcache/job_local1580085120_0001/attempt_local1580085120_00
/data/tmp/mapred/local/localRunner/hadoop/jobcache/job_local1580085120_0001/attempt_local1580085120_00
/data/tmp/mapred/local/localRunner/hadoop/jobcache/job_local1580085120_0001/attempt_local1580085120_00
/data/tmp/mapred/local/localRunner/hadoop/jobcache/job_local1580085120_0001/attempt_local1580085120_00
/data/tmp/mapred/local/localRunner/hadoop/jobcache/job_local1580085120_0001/attempt_local1580085120_00
/data/tmp/mapred/local/localRunner/hadoop/jobcache/job_local1580085120_0001/attempt_local1580085120_00
/data/tmp/mapred/local/localRunner/hadoop/jobcache/job_local1580085120_0001/attempt_local1580085120_00
/data/tmp/mapred/local/localRunner/hadoop/jobcache/job_local1580085120_0001/attempt_local1580085120_00
/data/tmp/mapred/local/localRunner/hadoop/jobcache/job_local1580085120_0001/attempt_local1580085120_00
/data/tmp/mapred/local/localRunner/hadoop/jobcache/job_local1580085120_0001/attempt_local1580085120_00
[hadoop@Master hadoop-2.9.2]$
```

临时目录信息

测试练习示例：

1.使用统计方法来计算Pi值

```
[root@hadoop-01 hadoop]# bin/hadoop jar  share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.1.jar  pi   100 100
```

2.统计某个文件中单词出现次数:

```
[root@hadoop-01 hadoop]# bin/hadoop jar  share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.1.jar  wordcount    input-file.txt    output-dir

##input-file.txt为要统计的输入文件
## output-dir为一个输出目录 （不能是已输出过结果的目录）
```

3.计算某个字符串出现的次数:

```
[root@hadoop-01 hadoop]# bin/hadoop jar  share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.1.jar  grep abc.txt  /tmp/grep-out  'hel[a-z]+'
```