

1. 集群搭建

1.1 HADOOP 集群搭建

1.1.1 集群简介

HADOOP 集群具体来说包含两个集群：HDFS 集群和 YARN 集群，两者逻辑上分离，但物理上常在一起

HDFS 集群：

负责海量数据的存储，集群中的角色主要有 NameNode / DataNode

YARN 集群：

负责海量数据运算时的资源调度，集群中的角色主要有 ResourceManager / NodeManager
(那 mapreduce 是什么呢？它其实是一个应用程序开发包)

本集群搭建案例，以 3 节点为例进行搭建，角色分配如下：

1.1.2 服务器准备

本案例使用虚拟机服务器来搭建 HADOOP 集群，所用软件及版本：

- ✓ Vmware 11.0
- ✓ Centos 6.5 64bit

1.1.3 网络环境准备

- ✓ 采用 NAT 方式联网
- ✓ 网关地址：192.168.33.1
- ✓ 3 个服务器节点 IP 地址：192.168.33.101、192.168.33.102、192.168.33.103
- ✓ 子网掩码：255.255.255.0

1.1.4 服务器系统设置

- ✓ 添加 HADOOP 用户
- ✓ 为 HADOOP 用户分配 sudoer 权限

- ✓ 同步时间
- ✓ 设置主机名
 - hdp-node-01
 - hdp-node-02
 - hdp-node-03
- ✓ 配置内网域名映射:

■ 192.168.33.101	hdp-node-01
■ 192.168.33.102	hdp-node-02
■ 192.168.33.103	hdp-node-03
- ✓ 配置 ssh 免密登陆
- ✓ 配置防火墙

1.1.5 Jdk 环境安装

- ✓ 上传 jdk 安装包
- ✓ 规划安装目录 /home/hadoop/apps/jdk_1.7.65
- ✓ 解压安装包
- ✓ 配置环境变量 /etc/profile

1.1.6 HADOOP 安装部署

- ✓ 上传 HADOOP 安装包
- ✓ 规划安装目录 /home/hadoop/apps/hadoop-2.6.1
- ✓ 解压安装包
- ✓ 修改配置文件 \$HADOOP_HOME/etc/hadoop/

最简化配置如下:

vi hadoop-env.sh

```
# The java implementation to use.
export JAVA_HOME=/home/hadoop/apps/jdk1.7.0_51
```

vi core-site.xml

```
<configuration>
<property>
<name>fs.defaultFS</name>
<value>hdfs://hdp-node-01:9000</value>
</property>
<property>
<name>hadoop.tmp.dir</name>
<value>/home/HADOOP/apps/hadoop-2.6.1/tmp</value>
</property>
```

```
</configuration>
```

```
vi  hdfs-site.xml
```

```
<configuration>
<property>
<name>dfs.namenode.name.dir</name>
<value>/home/hadoop/data/name</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>/home/hadoop/data/data</value>
</property>

<property>
<name>dfs.replication</name>
<value>3</value>
</property>

<property>
<name>dfs.secondary.http.address</name>
<value>hdp-node-01:50090</value>
</property>
</configuration>
```

```
vi  mapred-site.xml
```

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

```
vi  yarn-site.xml
```

```
<configuration>
<property>
<name>yarn.resourcemanager.hostname</name>
<value>hadoop01</value>
</property>

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

```
</configuration>
```

```
vi salves
```

```
hdp-node-01  
hdp-node-02  
hdp-node-03
```

1.1.7 启动集群

初始化 HDFS

```
bin/hadoop namenode -format
```

启动 HDFS

```
sbin/start-dfs.sh
```

启动 YARN

```
sbin/start-yarn.sh
```

1.1.8 测试

1、上传文件到 HDFS

从本地上传一个文本文件到 hdfs 的 /wordcount/input 目录下

```
[HADOOP@hdp-node-01 ~]$ HADOOP fs -mkdir -p /wordcount/input  
[HADOOP@hdp-node-01 ~]$ HADOOP fs -put /home/HADOOP/somewords.txt  
/wordcount/input
```

2、运行一个 mapreduce 程序

在 HADOOP 安装目录下，运行一个示例 mr 程序

```
cd $HADOOP_HOME/share/hadoop/mapreduce/  
hadoop jar mapredcue-example-2.6.1.jar wordcount /wordcount/input /wordcount/output
```

2 集群使用初步

2.1 HDFS 使用

1、查看集群状态

命令：`hdfs dfsadmin -report`

```
[hadoop@hdp-node-01 ~]$ hdfs dfsadmin -report
Safe mode is ON
Configured Capacity: 63392870400 (59.04 GB)
Present Capacity: 59549425664 (55.46 GB)
DFS Remaining: 56611115008 (52.72 GB)
DFS Used: 2938310656 (2.74 GB)
DFS Used%: 4.93%
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0

-----
Live datanodes (3):

Name: 192.168.33.154:50010 (hdp-node-03)
Hostname: hdp-node-03
Decommission Status : Normal
Configured Capacity: 21130956800 (19.68 GB)
DFS Used: 388399104 (370.41 MB)
Non DFS Used: 1264472064 (1.18 GB)
DFS Remaining: 19478085632 (18.14 GB)
DFS Used%: 1.84%
DFS Remaining%: 92.18%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Mon Dec 28 09:47:36 CST 2015

Name: 192.168.33.153:50010 (hdp-node-02)
Hostname: hdp-node-02
Decommission Status : Normal
Configured Capacity: 21130956800 (19.68 GB)
DFS Used: 389345280 (371.31 MB)
```

可以看出，集群共有 3 个 datanode 可用

也可打开 web 控制台查看 HDFS 集群信息，在浏览器打开 <http://hdp-node-01:50070/>

Hadoop Overview Datanodes Snapshot Startup Progress Utilities	
Overview 'hdp-node-01:9000' (active)	
Started:	Mon Dec 28 09:46:48 CST 2015
Version:	2.6.1, rb4d876d837b830405ccdb6af94742f99d49f9c04
Compiled:	2015-09-16T21:07Z by jenkins from (detached from b4d876d)
Cluster ID:	CID-b80db5b3-77df-459d-9d0e-707a6c47607b
Block Pool ID:	BP-1308957366-192.168.33.152-1447899502943

2、上传文件到 HDFS

✧ 查看 HDFS 中的目录信息

命令: `hadoop fs -ls /`

```
[hadoop@hdp-node-01 ~]$ hadoop fs -ls /
Found 13 items
drwxr-xr-x - hadoop supergroup 0 2015-12-06 15:39 /hbase
drwxr-xr-x - hadoop supergroup 0 2015-11-24 09:25 /hivebak
drwxr-xr-x - hadoop supergroup 0 2015-11-25 11:15 /mllib
drwxr-xr-x - hadoop supergroup 0 2015-12-07 17:27 /mydata
drwxr-xr-x - hadoop supergroup 0 2015-11-30 22:42 /savelist
drwxr-xr-x - hadoop supergroup 0 2015-12-08 14:48 /staytime
-rw-r--r-- 1 hadoop supergroup 27 2015-12-02 23:14 /stu.data
-rw-r--r-- 1 hadoop supergroup 68 2015-11-24 15:38 /test.file
drwx----- hadoop supergroup 0 2015-11-23 09:08 /tmp
drwxr-xr-x - hadoop supergroup 0 2015-11-23 10:13 /user
drwxr-xr-x - hadoop supergroup 0 2015-12-08 11:17 /weblog
drwxr-xr-x - hadoop supergroup 0 2015-11-30 15:01 /wordcount
-rw-r--r-- 1 hadoop supergroup 1330 2015-12-08 09:29 /yarn-site.xml
```

✧ 上传文件

命令: `hadoop fs -put ./scala-2.10.6.tgz to /`

```
[hadoop@hdp-node-01 ~]$ hadoop fs -ls /
Found 14 items
drwxr-xr-x - hadoop supergroup 0 2015-12-06 15:39 /hbase
drwxr-xr-x - hadoop supergroup 0 2015-11-24 09:25 /hivebak
drwxr-xr-x - hadoop supergroup 0 2015-11-25 11:15 /mllib
drwxr-xr-x - hadoop supergroup 0 2015-12-07 17:27 /mydata
drwxr-xr-x - hadoop supergroup 0 2015-11-30 22:42 /savelist
-rw-r--r-- 1 hadoop supergroup 29928531 2015-12-28 09:53 /scala-2.10.6.tgz
drwxr-xr-x - hadoop supergroup 0 2015-12-08 14:48 /staytime
-rw-r--r-- 1 hadoop supergroup 27 2015-12-02 23:14 /stu.data
-rw-r--r-- 1 hadoop supergroup 68 2015-11-24 15:38 /test.file
drwx----- hadoop supergroup 0 2015-11-23 09:08 /tmp
drwxr-xr-x - hadoop supergroup 0 2015-11-23 10:13 /user
drwxr-xr-x - hadoop supergroup 0 2015-12-08 11:17 /weblog
drwxr-xr-x - hadoop supergroup 0 2015-11-30 15:01 /wordcount
-rw-r--r-- 1 hadoop supergroup 1330 2015-12-08 09:29 /yarn-site.xml
```

✧ 从 HDFS 下载文件

命令: `hadoop fs -get /yarn-site.xml`

```
[hadoop@hdp-node-01 ~]$ hadoop fs -get /yarn-site.xml
[hadoop@hdp-node-01 ~]$ ll
total 498812
-rw-rw-r-- 1 hadoop hadoop 61084192 Nov 5 17:11 access_2013_05_30.log
-rw-rw-r-- 1 hadoop hadoop 3025757 Oct 3 23:07 access.log.10
-rw-rw-r-- 1 hadoop hadoop 170 Dec 7 21:32 accumulate2.txt
-rw-rw-r-- 1 hadoop hadoop 102 Dec 7 20:56 accumulate.txt
-rw-rw-r-- 1 hadoop hadoop 35777 Dec 8 14:45 weblog.jar
drwxrwxr-x 5 hadoop hadoop 4096 Nov 23 16:25 wf-oozie
-rw-rw-r-- 1 hadoop hadoop 1330 Dec 28 09:55 yarn-site.xml
-rw-rw-r-- 1 hadoop hadoop 23111 Dec 6 22:55 zookeeper.out
```

2.2 MAPREDUCE 使用

mapreduce 是 hadoop 中的分布式运算编程框架，只要按照其编程规范，只需要编写少量的业务逻辑代码即可实现一个强大的海量数据并发处理程序

2.2.1 Demo 开发——wordcount

1、需求

从大量（比如 T 级别）文本文件中，统计出每一个单词出现的总次数

2、mapreduce 实现思路

Map 阶段：

- a) 从 HDFS 的源数据文件中逐行读取数据
- b) 将每一行数据切分出单词
- c) 为每一个单词构造一个键值对(单词, 1)
- d) 将键值对发送给 reduce

Reduce 阶段：

- a) 接收 map 阶段输出的单词键值对
- b) 将相同单词的键值对汇聚成一组
- c) 对每一组，遍历组中的所有“值”，累加求和，即得到每一个单词的总次数
- d) 将(单词, 总次数)输出到 HDFS 的文件中

1、具体编码实现

(1)定义一个 mapper 类

```
//首先要定义四个泛型的类型
//keyin: LongWritable    valuein: Text
//keyout: Text           valueout: IntWritable

public class WordCountMapper extends Mapper<LongWritable, Text, Text, IntWritable>{
    //map 方法的生命周期： 框架每传一行数据就被调用一次
    //key: 这一行的起始点在文件中的偏移量
    //value: 这一行的内容
    @Override
    protected void map(LongWritable key, Text value, Context context) throws IOException,
    InterruptedException {
        //拿到一行数据转换为 string
        String line = value.toString();
        //将这一行切分出各个单词
        String[] words = line.split(" ");
        //遍历数组，输出<单词, 1>
        for(String word:words){
```

```

        context.write(new Text(word), new IntWritable(1));
    }
}
}

```

(2)定义一个 reducer 类

```

//生命周期：框架每传递进来一个 kv 组，reduce 方法被调用一次
@Override
protected void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException,
InterruptedException {
    //定义一个计数器
    int count = 0;
    //遍历这一组 kv 的所有 v，累加到 count 中
    for(IntWritable value:values){
        count += value.get();
    }
    context.write(key, new IntWritable(count));
}
}

```

(3)定义一个主类，用来描述 job 并提交 job

```

public class WordCountRunner {
    //把业务逻辑相关的信息（哪个是 mapper，哪个是 reducer，要处理的数据在哪里，输出的结果放哪
    里。。。。。）描述成一个 job 对象
    //把这个描述好的 job 提交给集群去运行
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job wcjob = Job.getInstance(conf);
        //指定我这个 job 所在的 jar 包
        //
        wcjob.setJar("/home/hadoop/wordcount.jar");
        wcjob.setJarByClass(WordCountRunner.class);

        wcjob.setMapperClass(WordCountMapper.class);
        wcjob.setReducerClass(WordCountReducer.class);
        //设置我们的业务逻辑 Mapper 类的输出 key 和 value 的数据类型
        wcjob.setMapOutputKeyClass(Text.class);
        wcjob.setMapOutputValueClass(IntWritable.class);
        //设置我们的业务逻辑 Reducer 类的输出 key 和 value 的数据类型
        wcjob.setOutputKeyClass(Text.class);
        wcjob.setOutputValueClass(IntWritable.class);

        //指定要处理的数据所在的位置
        FileInputFormat.setInputPaths(wcjob, "hdfs://hdp-server01:9000/wordcount/data/big.txt");
        //指定处理完成之后的结果所保存的位置
    }
}

```



```

        FileOutputFormat.setOutputPath(wcjob, new Path("hdfs://hdp-server01:9000/wordcount/output/"));

        //向 yarn 集群提交这个 job
        boolean res = wcjob.waitForCompletion(true);
        System.exit(res?0:1);
    }

```

2.2.2 程序打包运行

1. 将程序打包
2. 准备输入数据

vi /home/hadoop/test.txt

```

Hello tom
Hello jim
Hello ketty
Hello world
Ketty tom

```

在 hdfs 上创建输入数据文件夹:

```
hadoop fs mkdir -p /wordcount/input
```

将 words.txt 上传到 hdfs 上

```
hadoop fs -put /home/hadoop/words.txt /wordcount/input
```

```

[hadoop@hdp-node-01 ~]$ hadoop fs -ls /wordcount/input
Found 1 items
-rw-r--r-- 1 hadoop supergroup 68 2015-11-30 14:38 /wordcount/input/test.txt
[hadoop@hdp-node-01 ~]$ hadoop fs -cat /wordcount/input/test.txt
hello world
very good
good day
very nice
nice world
hello say a day
[hadoop@hdp-node-01 ~]$

```

3. 将程序 jar 包上传到集群的任意一台服务器上

4. 使用命令启动执行 wordcount 程序 jar 包

```
$ hadoop jar wordcount.jar cn.zju.bigdata.mrsimple.WordCountDriver /wordcount/input /wordcount/out
```

```

drwxr-xr-x 2 hadoop hadoop 4096 Sep 17 05:18 sources
[hadoop@hdp-node-01 mapreduce]$ hadoop jar hadoop-mapreduce-ex
15/12/28 11:53:48 INFO client.RMProxy: Connecting to ResourceM
15/12/28 11:53:49 INFO input.FileInputFormat: Total input path
15/12/28 11:53:49 INFO mapreduce.JobSubmitter: number of split
15/12/28 11:53:50 INFO mapreduce.JobSubmitter: Submitting toke
15/12/28 11:53:51 INFO impl.YarnClientImpl: Submitted applicat
15/12/28 11:53:51 INFO mapreduce.Job: The url to track the job
15/12/28 11:53:51 INFO mapreduce.Job: Running job: job_1451274
15/12/28 11:54:02 INFO mapreduce.Job: Job job_1451274797559_00
15/12/28 11:54:02 INFO mapreduce.Job: map 0% reduce 0%
15/12/28 11:54:13 INFO mapreduce.Job: map 100% reduce 0%

```

5. 查看执行结果

\$ `hadoop fs -cat /wordcount/out/part-r-00000`

```
[hadoop@hdp-node-01 mapreduce]$ hadoop fs -ls /wordcount/out
Found 2 items
-rw-r--r-- 1 hadoop supergroup 0 2015-12-28 11:54 /wordcount/out/_SUCCESS
-rw-r--r-- 1 hadoop supergroup 53 2015-12-28 11:54 /wordcount/out/part-r-00000
[hadoop@hdp-node-01 mapreduce]$ hadoop fs -cat /wordcount/out/part-r-00000
a 1
day 2
good 2
hello 2
nice 2
say 1
very 2
world 2
```