docker搭建hadoop完全分布式

计算机科学与技术17-1 陈巍

0. 说明

在已经构建好的单机伪分布式基础上搭建hadoop完全分布式。

> 【注意】提前创建好docker网络

```
sudo docker network create --subnet=172.24.0.0/24 will-network
```

1. 集群规划

> 这里搭建一个 5 个节点的 Hadoop 集群，其中四台主机均部署 `DataNode` 和 `NodeManager` 服务，但其中 hadoop02 上部署 `SecondaryNameNode`服务和 `NTP服务器`，hadoop01 上部署 `NameNode` 、 `ResourceManager` 和`JobHistoryServer` 服务。

> master = hadoop01 => NameNode && ResourceManager && JobHistoryServer
>
> slave = {
>
> hadoop02; => DataNode && NodeManager && SecondaryNamenode && NTP-Server
>
> hadoop03; => DataNode && NodeManager
>
> hadoop04; => DataNode && NodeManager
>
> hadoop05; => DataNode && NodeManager
>
> }

```
# 节点ip分配:
hadoop01=172.24.0.2
hadoop02=172.24.0.3
hadoop03=172.24.0.4
hadoop04=172.24.0.5
hadoop05=172.24.0.6
```

2. 创建并启动容器

> 使用一个「单机伪分布式镜像」启动多个镜像。

2.1 手动启动方式

```
# 下面会用到systemctl 管理服务（配置时间同步），需要加上参数 --privileged 来增加权并且不能使用默认的bash，换成 init（能够使用 systemctl 命令）
# hadoop01
# 50070 hdfs
# 8088 yarn
# 19888 JobHistoryServer
sudo docker run -d --name hadoop01 --hostname hadoop01 --net will-network --ip 172.24.0.2 -P -p 50070:50070 -p 8088:8088 -p 19888:19888 --privileged 2d7d22776b99 /usr/sbin/init

# hadoop02
# 50090 SecondaryNameNode
sudo docker run -d --name hadoop02 --hostname hadoop02 --net will-network --ip 172.24.0.3 -P -p 50090:50090 --privileged 2d7d22776b99  /usr/sbin/init

# hadoop03
sudo docker run -d --name hadoop03 --hostname hadoop03 --net will-network --ip 172.24.0.4 -P --privileged 2d7d22776b99 /usr/sbin/init

# hadoop04
```

```
sudo docker run -d --name hadoop04 --hostname hadoop04 --net will-network --ip 172.24.0.5 -P --privileged 2d7d22776b99 /usr/sbin/init

# hadoop05
sudo docker run -d --name hadoop05 --hostname hadoop05 --net will-network --ip 172.24.0.6 -P --privileged 2d7d22776b99 /usr/sbin/init
```

2.2 Shell脚本启动方式

> 编写脚本`hadoop-cluster.sh`，批量启动容器

```
# !/bin/bash
# description:Batch start Containers Script（批量启动容器）
# author:will

# 预判启动容器的数量：不小于1 && 提示信息
if [ $# -ne 1  ];then
        echo "You need to start several containers explicitly（批量启动容器的方法如下）."
        echo "Some like './hadoop-cluster.sh 3' or 'sh hadoop-cluster.sh 3'"
        exit 1
fi

#要启动的容器数量
NUM_CONTAINERS=$1
#自定义网络名称
NETWORK_NAME=will-network
#镜像ID
IMAGE_ID=2d7d22776b99
#前缀
PREFIX="0"
for (( i=1;i<=$NUM_CONTAINERS;i++ ))
        do
```

```
                if [ $i -eq 1  ];then
                        sudo docker run -d --name hadoop$PREFIX$i --hostname hadoop$PREFIX$i --net ${NETWORK_NAME} --ip 172.24.0.$[$i+1] -P -p 50070:50070 -p 8088:8088 -p 19888:19888 --privileged $IMAGE_ID /usr/sbin/init
                elif [ $i -eq 2 ];then
                        sudo docker run -d --name hadoop$PREFIX$i --hostname hadoop$PREFIX$i --net ${NETWORK_NAME} --ip 172.24.0.$[$i+1] -P -p 50090:50090 --privileged $IMAGE_ID /usr/sbin/init
                else
                        sudo docker run -d --name hadoop$PREFIX$i --hostname hadoop$PREFIX$i --net ${NETWORK_NAME} --ip 172.24.0.$[$i+1] -P --privileged $IMAGE_ID /usr/sbin/init
                fi
        done
echo "$NUM_CONTAINERS containers started!"
echo "====================================="
sudo docker ps | grep hadoop
echo "============IPv4=================="
sudo docker inspect $(sudo docker ps -q) | grep -i ipv4
echo "====================================="
```

使用shell脚本启动：

```
[will@master ~]$ sh ./Desktop/hadoop-cluster.sh 3
2840cee259340a22e910af7e1cbd35be060d84be92989dab5238e60dd74b6d9c
7485d39d5df39950b8f17e7dbf3f2dc489c029283925d586f0b4b1d9ea1246bf
4bd249d461cb7b3969d58d6dc0617d534c38b5d1f122edb0a89c14f98ebfe938
3 containers started!
=================================
4bd249d461cb        2d7d22776b99        "/usr/sbin/init"    4 seconds ago       Up Less than a second   0.0.0.0:32772->22/tcp
                                                            hadoop03
7485d39d5df3        2d7d22776b99        "/usr/sbin/init"    8 seconds ago       Up 4 seconds            0.0.0.0:50090->50090/tcp, 0.0.0.
0:32771->22/tcp                                             hadoop02
2840cee25934        2d7d22776b99        "/usr/sbin/init"    8 seconds ago       Up 7 seconds            0.0.0.0:8088->8088/tcp, 0.0.0.0:
19888->19888/tcp, 0.0.0.0:50070->50070/tcp, 0.0.0.0:32770->22/tcp   hadoop01
============IPv4==================
                    "IPv4Address": "172.24.0.4"
                    "IPv4Address": "172.24.0.3"
                    "IPv4Address": "172.24.0.2"
```

【受虚拟机性能所限制，只能同时启动三个容器】

3. 进入容器进行集群配置

3.1 进入容器

```
sudo docker exec -ti hadoop01 /bin/bash

sudo docker exec -ti hadoop02 /bin/bash

sudo docker exec -ti hadoop03 /bin/bash

sudo docker exec -ti hadoop04 /bin/bash

sudo docker exec -ti hadoop05 /bin/bash

# 每进入一个之后，还要进入 will用户
su will
```

3.2 配置映射

> 配置ip地址和主机名映射（所有节点都要配置）

```
172.24.02 hadoop01
172.24.03 hadoop02
172.24.04 hadoop03
172.24.05 hadoop04
172.24.06 hadoop05
```

```
# 修改
vim /etc/hosts

# 删除最后一行

# 插入
172.24.02 hadoop01
172.24.03 hadoop02
172.24.04 hadoop03
172.24.05 hadoop04
172.24.06 hadoop05
```

3.3 配置免密登录

3.3.1 生成密钥

> 在每一台主机上使用`ssh-keygen`命令生成公钥和私钥

```
ssh-keygen -t rsa -N '' -C 'will'
# -N ‘’     指定：不需要密码
# -N和-C可写，也可以不写

# 然后一路「回车」
```

3.3.2 复制公钥

> 每一个节点都要把公钥给其他节点（每个容器中都执行一遍下面的内容）

```
ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop01
ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop02
ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop03
ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop04
ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop05
```

3.3.3 验证免密登录

```
ssh hadoop01
ssh hadoop02
ssh hadoop03
ssh hadoop04
ssh hadoop05
```

3.4 同步集群时间（hadoop02）

> 思路：
>
> hadoop02开启ntp服务；其他节点同步hadoop02

3.4.1 检查ntp包是否安装

```
rpm -qa | grep ntp

# 如果没有安装的话，执行 sudo yum -y install ntp
```

> hadoop02的操作：

3.4.2 设置时间文件

```
sudo vim /etc/ntp.conf
```

```
#修改一（设置本地网络上的主机不受限制）
#restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap
restrict 172.24.0.0 mask 255.255.255.0 nomodify notrap
#修改二（添加默认的一个内部时钟数据，使用它为局域网用户提供服务）
server 127.127.1.0
fudge 127.127.1.0 stratum 10
#修改三（设置为不采用公共的服务器）
server 0.centos.pool.ntp.org iburst
server 1.centos.pool.ntp.org iburst
server 2.centos.pool.ntp.org iburst
server 3.centos.pool.ntp.org iburst
#server 0.centos.pool.ntp.org iburst
#server 1.centos.pool.ntp.org iburst
#server 2.centos.pool.ntp.org iburst
#server 3.centos.pool.ntp.org iburst
```

3.4.3 设置BIOS与系统时间同步

```
sudo vim /etc/sysconfig/ntpd

# 增加下面的内容
OPTIONS="-u ntp:ntp -p /var/run/ntpd.pid -g"
SYNC_HWCLOCK=yes
```

3.4.4 启动ntp服务并测试

```
sudo systemctl start ntpd

systemctl status ntpd

sudo systemctl enable ntpd.service

# 用来查看ntp状态（下同）
ntpstat

sudo ntpdq -p
```

3.4.5 关闭非时间服务器节点的ntpd服务

```
sudo systemctl stop ntpd
```

> 手动同步（所有节点）

```
# 手动设置时区
sudo timedatectl set-timezone Asia/Shanghai
```

> 其他节点（非hadoop02）的操作

> 定时同步

```
#非时间服务器都安装crond服务
sudo yum -y install vixie-cron crontabs
#非时间服务器节点都编写定时同步时间任务
sudo vim /etc/crontab
编写定时任务如下:
*/1 * * * * /usr/sbin/ntpdate hadoop02

#加载任务,使之生效
sudo crontab /etc/crontab
```

```
sudo ntpdate -u ntp.api.bz
```

4. 修改主节点配置文件

> hadoop01中的操作;
>
> 在hadoop目录下。

```
mkdir /opt/software/hadoop-2.7.3/tmp

mkdir -p /opt/software/hadoop-2.7.3/dfs/namenode_data

mkdir -p /opt/software/hadoop-2.7.3/dfs/datanode_data

mkdir -p /opt/software/hadoop-2.7.3/checkpoint/dfs/cname
```

4.1 hadoop-env.sh

```
#25行 export JAVA_HOME
export JAVA_HOME=/opt/moudle/jdk1.8.0_251
#33行 export HADOOP_CONF_DIR
export HADOOP_CONF_DIR=/opt/software/hadoop-2.7.3/etc/hadoop
```

4.2 core-site.xml

```
<configuration>
        <property>
            <!--用来指定hdfs的主节点，namenode的地址-->
            <name>fs.defaultFS</name>
            <value>hdfs://hadoop01:9000</value>
        </property>
        <property>
            <!--用来指定hadoop运行时产生文件的存放目录-->
            <name>hadoop.tmp.dir</name>
            <value>/opt/software/hadoop-2.7.3/tmp</value>
        </property>
        <property>
```

```
            <!--设置缓存大小，默认4kb-->
            <name>io.file.buffer.size</name>
            <value>4096</value>
        </property>
</configuration>
```

4.3 hdfs-site.xml

```
<configuration>
        <property>
           <!--数据块默认大小128M-->
           <name>dfs.block.size</name>
           <value>134217728</value>
        </property>
        <property>
            <!--副本数量，不配置的话默认为3-->
            <name>dfs.replication</name>
            <value>3</value>
        </property>
        <property>
                <!--定点检查-->
                <name>fs.checkpoint.dir</name>
                <value>/opt/software/hadoop-2.7.3/checkpoint/dfs/cname</value>
        </property>
        <property>
            <!--namenode节点数据（元数据）的存放位置-->
            <name>dfs.name.dir</name>
            <value>/opt/software/hadoop-2.7.3/dfs/namenode_data</value>
        </property>
        <property>
            <!--datanode节点数据（元数据）的存放位置-->
            <name>dfs.data.dir</name>
            <value>/opt/software/hadoop-2.7.3/dfs/datanode_data</value>
        </property>
```

```
        <property>
            <!--指定secondarynamenode的web地址-->
            <name>dfs.namenode.secondary.http-address</name>
            <value>hadoop02:50090</value>
        </property>
        <property>
            <!--hdfs文件操作权限,false为不验证-->
            <name>dfs.permissions</name>
            <value>false</value>
        </property>
</configuration>
```

4.4 mapreduce-site.xml

> 在${HADOOP_HOME}/etc/hadoop的目录下，只有一个mapred-site.xml.template文件，复制一个进行更改。

```
cp mapred-site.xml.template mapred-site.xml

<configuration>
        <property>
            <!--指定mapreduce运行在yarn上-->
            <name>mapreduce.framework.name</name>
            <value>yarn</value>
        </property>
        <property>
            <!--配置任务历史服务器地址-->
            <name>mapreduce.jobhistory.address</name>
            <value>hadoop01:10020</value>
        </property>
        <property>
            <!--配置任务历史服务器web-UI地址-->

<name>mapreduce.jobhistory.webapp.address</name>
            <value>hadoop01:19888</value>
        </property>
```

```
</configuration>
```

4.5 yarn-site.xml

```
<configuration>
        <property>
        <!--指定yarn的老大resourcemanager的地址-->
        <name>yarn.resourcemanager.hostname</name>
        <value>hadoop01</value>
        </property>
        <property>
            <name>yarn.resourcemanager.address</name>
            <value>hadoop01:8032</value>
        </property>
        <property>

<name>yarn.resourcemanager.webapp.address</name>
            <value>hadoop01:8088</value>
        </property>
        <property>

<name>yarn.resourcemanager.scheduler.address</name>
            <value>hadoop01:8030</value>
        </property>
        <property>
            <name>yarn.resourcemanager.resource-
tracker.address</name>
            <value>hadoop01:8031</value>
        </property>
        <property>

<name>yarn.resourcemanager.admin.address</name>
            <value>hadoop01:8033</value>
        </property>
        <property>
            <!--NodeManager获取数据的方式-->
            <name>yarn.nodemanager.aux-services</name>
```

```
            <value>mapreduce_shuffle</value>
        </property>
        <property>
            <!--开启日志聚集功能-->
            <name>yarn.log-aggregation-enable</name>
            <value>true</value>
        </property>
        <property>
            <!--配置日志保留7天-->
            <name>yarn.log-aggregation.retain-seconds</name>
            <value>604800</value>
        </property>
</configuration>
```

4.6 master

```
# 在 hadoop目录下
vim master
# 添加内容
hadoop01
```

4.7 slaves

```
# 在 hadoop目录下
vim slave
# 添加内容
hadoop02
hadoop03
hadoop04
hadoop05
```

4.8 分发配置文件

> hadoop01分发到其他机器上去

> 手动

```
sudo scp -r /opt/software/hadoop-2.7.3/ will@hadoop02:/opt/software/
sudo scp -r /opt/software/hadoop-2.7.3/ will@hadoop03:/opt/software/
sudo scp -r /opt/software/hadoop-2.7.3/ will@hadoop04:/opt/software/
sudo scp -r /opt/software/hadoop-2.7.3/ will@hadoop05:/opt/software/
```

> shell脚本`scp-config.sh`

```
#!/bin/bash
#description：节点间复制文件
#author：will

#首先判断参数是否存在
args=$#
if [ args -eq 0 ];then
    echo "no args"
    exit 1
fi
#获取文件名称
p1=$1
fname=$(basename $p1)
echo faname=$fname
#获取上级目录到绝对路径
pdir=$(cd $(dirname $p1);pwd -P)
echo pdir=$pdir
#获取当前用户名称
user=$(whoami)
#循环分发
#  【注意，如果节点数量更多】当前是，hadoop02 ~ hadoop05节点
for(( host=2;host<6;host++ ));do
    echo "------hadoop0$host------"
    scp -r $pdir/$fname $user@hadoop0$host:$pdir
done
echo "------All done ------"
```

```
# 使用shell脚本
sh scp-config.sh /opt/software/hadoop-2.7.3/
```

4.9 初始化

> hadoop01

```
hdfs namenode -format
```

5 启动集群

> 在hadoop01上启动集群，`haoop02`、`hadoop03`、`hadoop04`、`hadoop05`上相关服务也会被启动

```
# hadoop01上操作
start-dfs.sh

start-yarn.sh

mr-jobhistory-daemon.sh start historyserver
```

> 每一台服务器上`jps`查看即可

参考资料及笔记

> hadoop搭建完全分布式（文案）

> hadoop搭建完全分布式（视频）

> hadoop基本原理（读写 && mapreduce）

NameNode：是Master节点（主节点），可以看作是分布式文件系统中的管理者，主要负责管理文件系统的命名空间、集群配置信息和存储块的复制等（HDFS的元数据信息）。NameNode会将文件系统的Meta-data存储在内存中，这些信息主要包括了文件信息、每一个文件对应的文件块的信息和每一个文件块在DataNode的信息等（HDFS的元数据信息的持久化）。

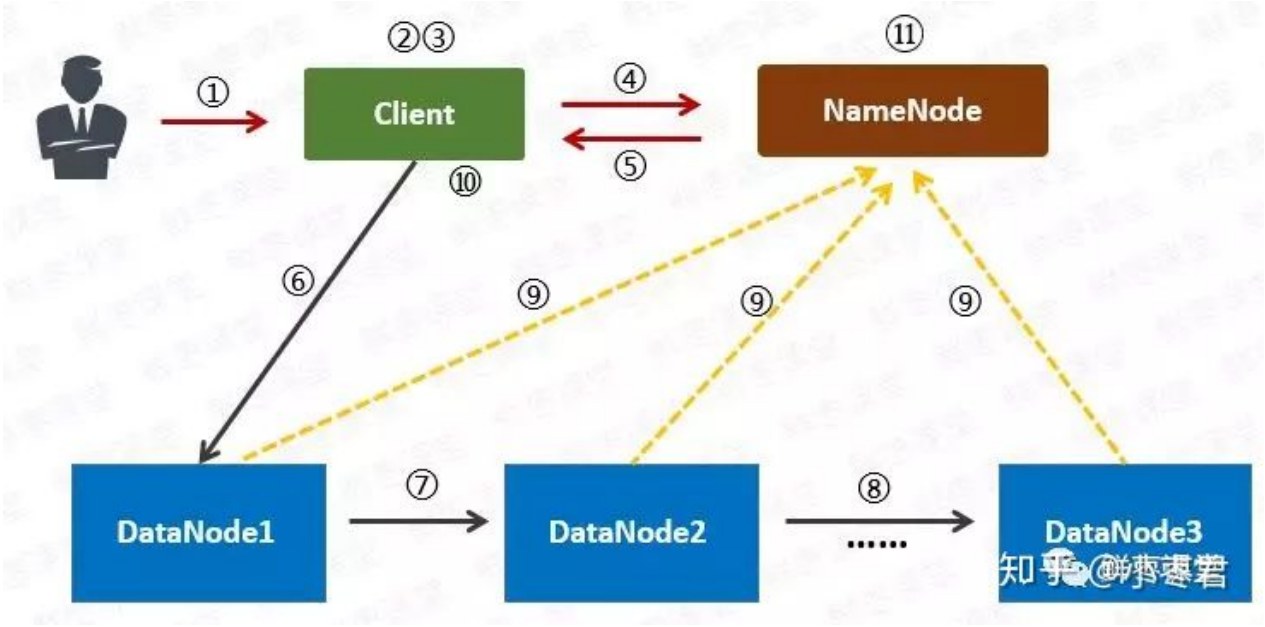
DataNode：是Slave节点（从节点），是文件存储的基本单元，它将Block存储在本地文件系统中，保存了Block的Meta-data，同时周期性地将所有存在的Block信息发送给NameNode。

Client：切分文件；访问HDFS；与NameNode交互，获得文件位置信息；与DataNode交互，读取和写入数据。

还有一个**Block（块）**的概念：Block是HDFS中的基本读写单元；HDFS中的文件都是被切割为block（块）进行存储的；这些块被复制到多个DataNode中；块的大小（通常为64MB）和复制的块数量在创建文件时由Client决定。

我们来简单看看HDFS的读写流程。

首先是**写入流程**：



1 用户向Client（客户机）提出请求。例如，需要写入200MB的数据。

2 Client制定计划：将数据按照64MB为块，进行切割；所有的块都保存三份。

3 Client将大文件切分成块（block）。

4 针对第一个块，Client告诉NameNode（主控节点），请帮助我，将64MB的块复制三份。

5 NameNode告诉Client三个DataNode（数据节点）的地址，并且将它们根据到Client的距离，进行了排序。

6 Client把数据和清单发给第一个DataNode。

7 第一个DataNode将数据复制给第二个DataNode。

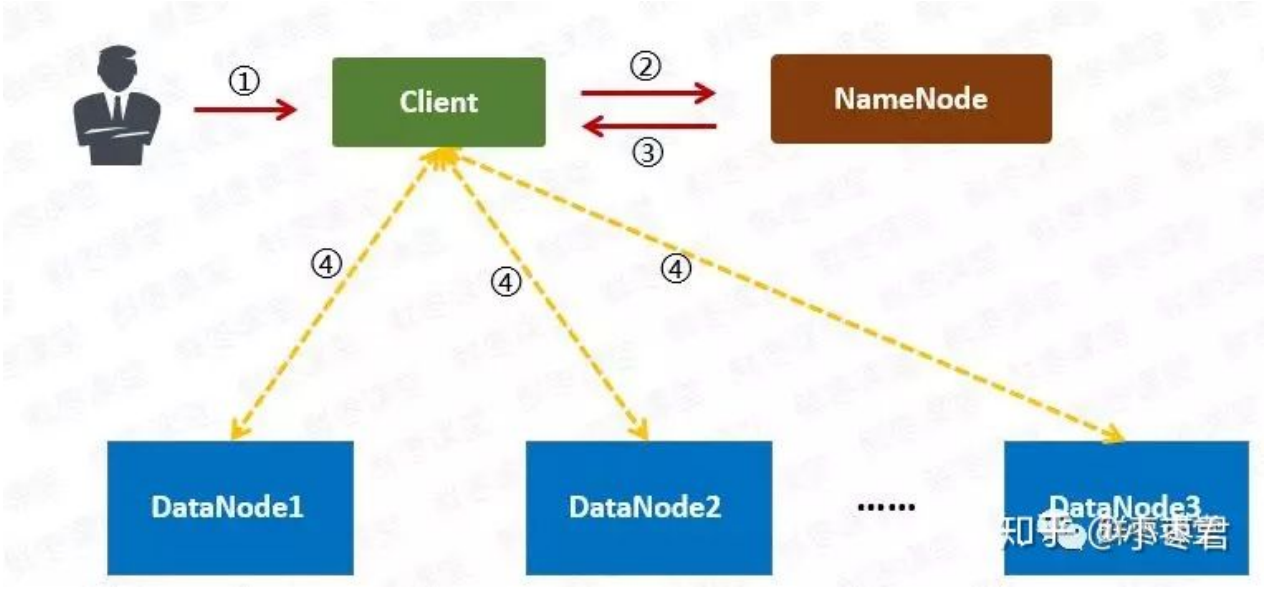
8 第二个DataNode将数据复制给第三个DataNode。

9 如果某一个块的所有数据都已写入，就会向NameNode反馈已完成。

10 对第二个Block，也进行相同的操作。

11 所有Block都完成后，关闭文件。NameNode会将数据持久化到磁盘上。

读取流程：



1 用户向Client提出读取请求。

2 Client向NameNode请求这个文件的所有信息。

3 NameNode将给Client这个文件的块列表，以及存储各个块的数据节点清单（按照和客户端的距离排序）。

4 Client从距离最近的数据节点下载所需的块。

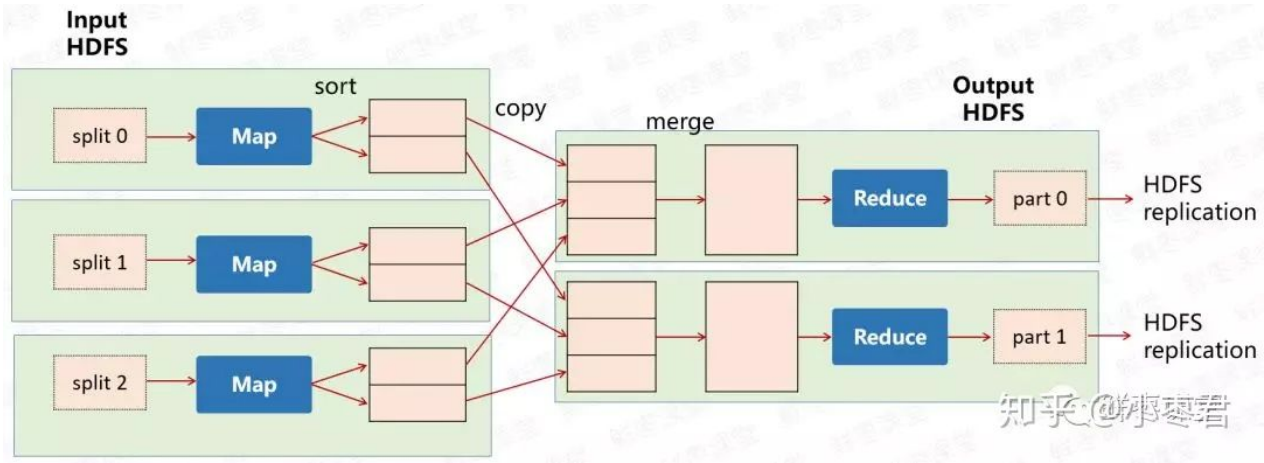
（注意：以上只是简化的描述，实际过程会更加复杂。）

再来看**MapReduce**。

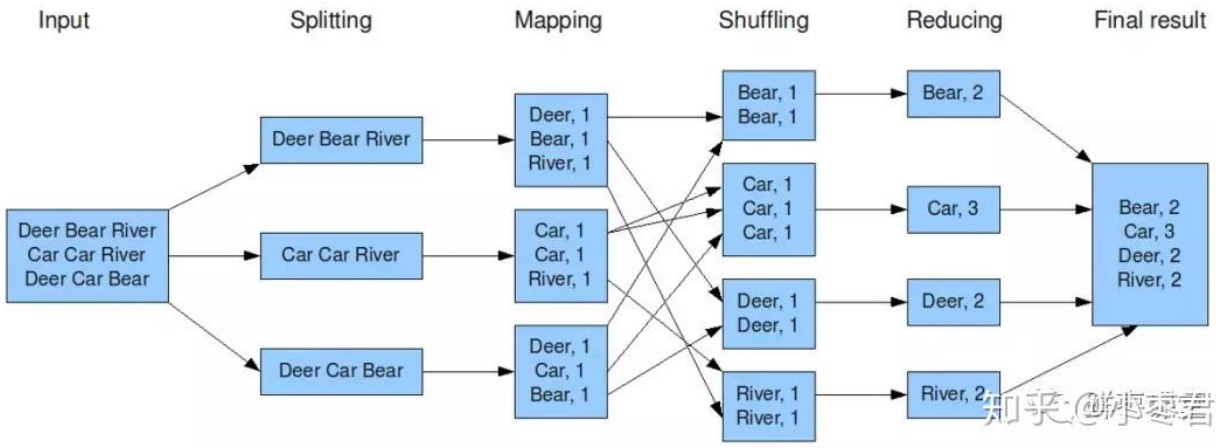
MapReduce其实是一种编程模型。这个模型的核心步骤主要分两部分：**Map（映射）**和**Reduce（归约）**。

当你向MapReduce框架提交一个计算作业时，它会首先把计算作业拆分成若干个**Map任务**，然后分配到不同的节点上去执行，每一个Map任务处理输入数据中的一部分，当Map任务完成后，它会生成一些中间文件，这些中间文件将会作为**Reduce任务**的输入数据。Reduce任务的主要目标就是

把前面若干个Map的输出汇总到一起并输出。



例子：单词统计（WordCount）



1 Hadoop将输入数据切成若干个分片，并将每个split（分割）交给一个map task（Map任务）处理。

2 Mapping之后，相当于得出这个task里面，每个词以及它出现的次数。

3 shuffle（拖移）将相同的词放在一起，并对它们进行排序，分成若干个分片。

4 根据这些分片，进行reduce（归约）。

5 统计出reduce task的结果，输出到文件。

> 节点解释

ResourceManager：资源管理，yarn集群的主节点。功能上：与客户端交流，处理客户端的请求；管理NodeManager，接收来自NodeManager的资源汇报信息，并向NodeManager下达管理指令。

JobHistoryServer：yarn查看已经完成的任务的历史日志记录的服务。功能上：查看

NodeManager：节点管理，yarn集群的从节点。功能上：任务的计算。

SecondaryNameNode：镜像备份；日志与镜像的定期合并。

NTP-Server：时间服务器。功能上：保证集群内所有主机的时间同步。