**大数据介绍**

**大数据的由来**

• 大数据

– 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。

**什么是大数据**

• 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。

**大数据与Hadoop**

• 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历史起源**

• 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

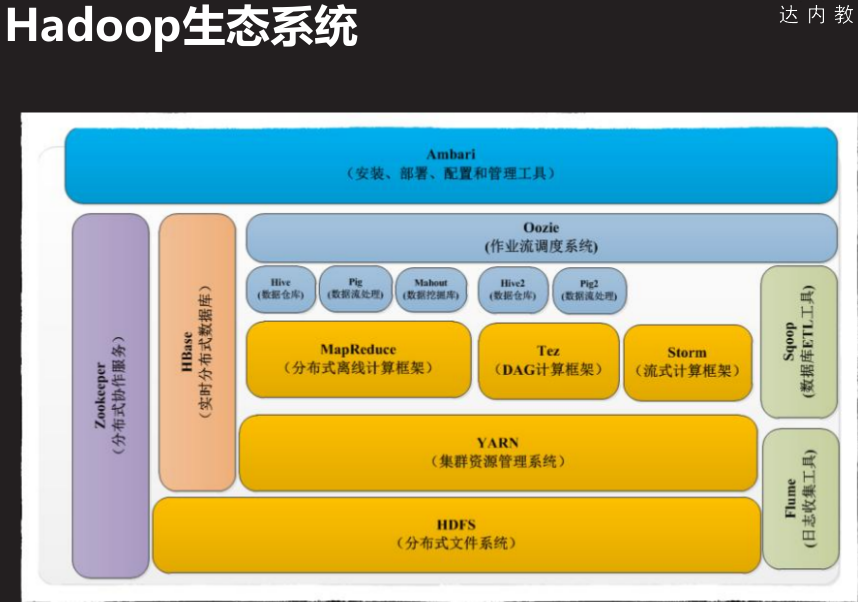
• Hive:基于Hadoop的数据仓库

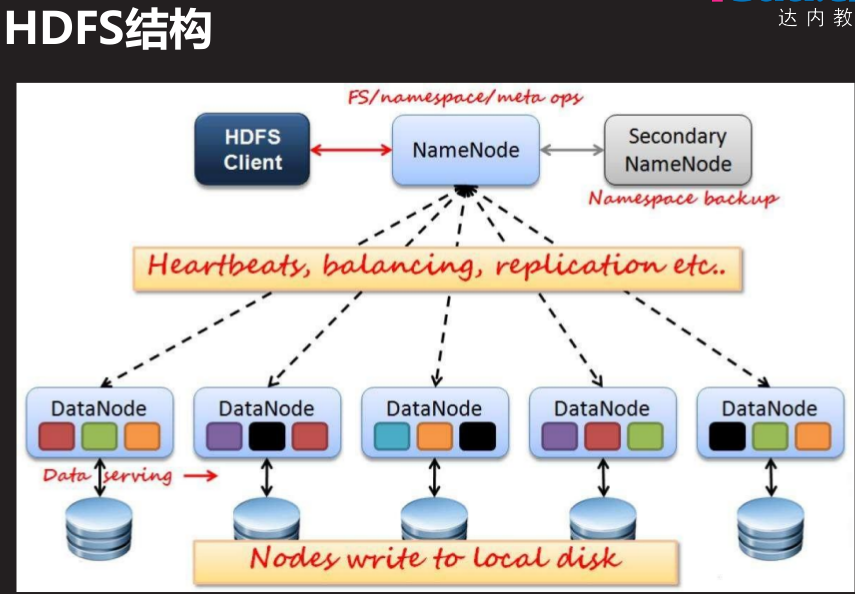
• Sqoop:数据同步工具

• 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





**数据分5片，每片备份两份**

**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（数据块映射信息）**,**推送给**NameNode

– 紧急情况下,可辅助恢复NameNode

• 但Secondary NameNode并非NameNode的热备

**fsedits数据变更日志。**

**Secondary辅助namenode处理数据块映射信息，提升效率。**

**• 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 **将整合的数据切分，平均切分，数量根据datanode的数量决定**

– 访问HDFS

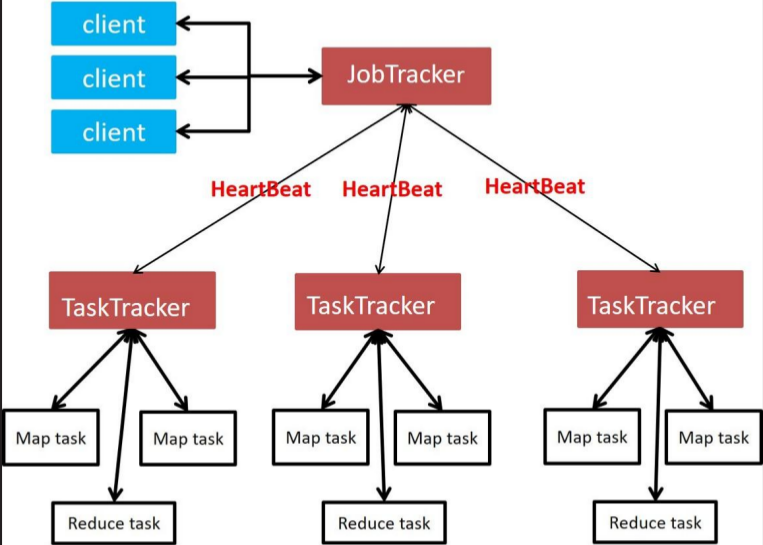
– 与NameNode交互,获取文件位置信息

– 与DataNode交互,读取和写入数据

**写入数据**

Client（数据分片）-------->namenode（fsimage）------>datanode（数据库）

**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

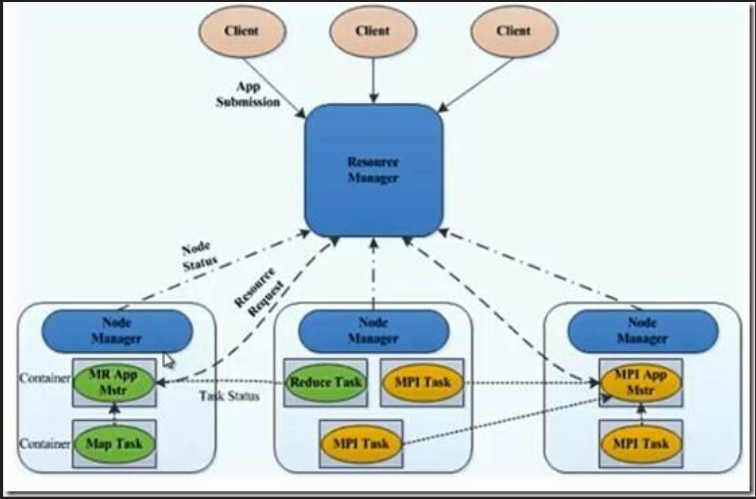
– 如果为map-only作业,直接写入HDFS

**• 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角色及概念**

• 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的创建**

1.设置主机名hadoop-nn01 ip为192.168.1.10，配置yum源

2.安装依赖包java-1.8.0-openjdk-devel

Java-version //查看系统版本

3.安装hadoop

tar -xf hadoop-2.7.6.tar.gz

mv hadoop-2.7.6 /usr/local/hadoop

4.解决报错问题

./bin/hadoop //报错，JAVA\_HOME没有找到

Error: JAVA\_HOME is not set and could not be found.

5.修改配置文件

vim /usr/local/hadoop/etc/hadoop/hadoop-env.sh

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

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

**//自己安装的路径**

6.应用实例

mkdir oo

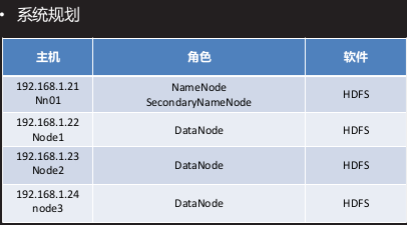
cd /usr/local/hadoop/

cp \*.txt oo

./bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.6.jar wordcount oo xx

cat xx/part-r-00000  **//显示内容**

**完全分布式**



**• 基础环境准备**

– 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**

– 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

**• 配置SSH信任关系(NameNode)**

**– 注意:不能出现要求输入yes的情况,每台机器都要能登录成功,包括本机!!!**

**– ssh\_config**

StrictHostKeyChecking no

# ssh-keygen -b 2048 -t rsa -N '' -f key

# ssh-copy-id -i ./key.pub root@ip.xx.xx.xx

**• HDFS完全分布式系统配置**

**– 环境配置文件:hadoop-env.sh**

**– 核心配置文件:core-site.xml**

**– HDFS配置文件:hdfs-site.xml**

**– 节点配置文件:slaves**

**• 环境配置文件hadoop-env.sh**

– OpenJDK的安装目录:JAVA\_HOME

– Hadoop配置文件的存放目录:HADOOP\_CONF\_DIR

**• 核心配置文件 core-site.xml**

– fs.defaultFS:文件系统配置参数

– hadoop.tmp.dir:数据目录配置参数

<property>

<name>fs.defaultFS</name>

<value>hdfs://nn01:9000</value>

</property>

<property>

<name>hadoop.tmp.dir</name>

<value>/var/hadoop</value>

</property>

**• HDFS配置文件hdfs-site.xml**

– Namenode:地址声明

dfs.namenode.http-address

– Secondarynamenode:地址声明

dfs.namenode.secondary.http-address

– 文件冗余份数

dfs.replication

**• HDFS配置文件hdfs-site.xml**

<property>

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

<value>nn01:50070</value>

</property>

<property>

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

<value>nn01:50090</value>

</property>

<property>

<name>dfs.replication</name>

<value>2</value> **//总共备份2份允许坏1份**

</property> //kibana分4份备份5份

**• 节点配置文件slaves**

– 只写DataNode节点的主机名称

vim /usr/local/hadoop/etc/hadoop/slaves

node1

node2

node3

– 同步配置

– 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

**在所有机器上安装/usr/local/hadoop**

for a in {22..24}; do scp -r /usr/local/hadoop/ root@192.168.1.$a:/usr/local/; done

**• 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

**• JPS验证角色**

**– 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

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

Configured Capacity: 51505004544 (47.97 GB)

DFS Used: 733184 (716 KB)

... ...

Missing blocks: 0

Missing blocks (with replication factor 1): 0

-------------------------------------------------

Live datanodes (3):

