



一、 课程大纲

目录

一、 课程大纲.....	1
二、 Apache Hadoop.....	3
1. Hadoop 介绍.....	3
2. Hadoop 发展简史.....	5
3. Hadoop 特性优点.....	5
4. Hadoop 国内外应用.....	6
三、 Hadoop 集群搭建.....	7
1. 发行版本.....	7
2. 集群简介.....	7
3. 服务器准备.....	8
4. 网络环境准备.....	8
5. 服务器系统设置.....	9
6. JDK 环境安装.....	10
7. Hadoop 安装包目录结构.....	10
8. Hadoop 配置文件修改.....	11
8.1. hadoop-env.sh.....	11
8.2. core-site.xml.....	11
8.3. hdfs-site.xml.....	12
8.4. mapred-site.xml.....	12
8.5. yarn-site.xml.....	13
8.6. slaves.....	13
9. Hadoop 环境变量.....	14
四、 Hadoop 集群启动、初体验.....	15
1. 启动方式.....	15
1.1. 单节点逐个启动.....	15
1.2. 脚本一键启动.....	15
2. 集群 web-ui.....	16
3. Hadoop 初体验.....	17
3.1. HDFS 使用.....	17
3.2. 运行 mapreduce 程序.....	17
五、 HDFS 入门.....	18
1. HDFS 基本概念.....	18
1.1. HDFS 介绍.....	18



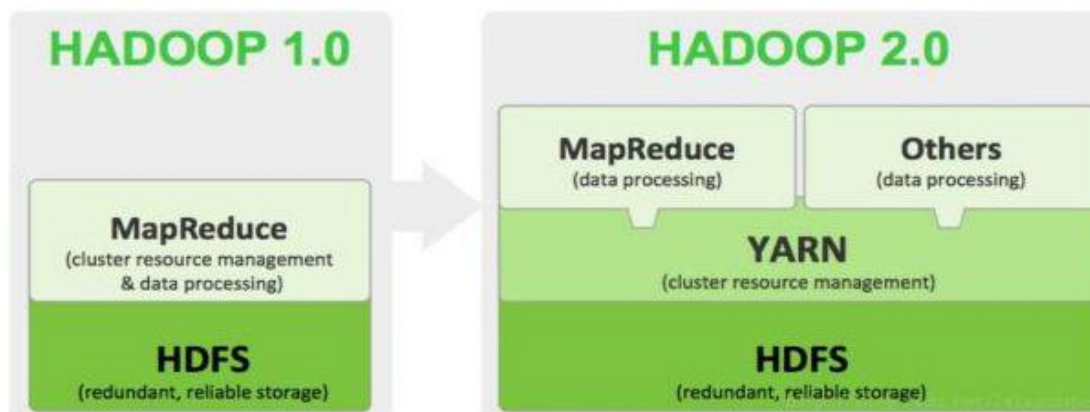
1.2.	HDFS 设计目标	18
2.	HDFS 重要特性	19
2.1.	master/slave 架构	19
2.2.	分块存储	19
2.3.	名字空间 (NameSpace)	19
2.4.	Namenode 元数据管理	19
2.5.	Datanode 数据存储	20
2.6.	副本机制	20
2.7.	一次写入，多次读出	20
3.	HDFS 基本操作	21
3.1.	Shell 命令行客户端	21
3.2.	Shell 命令选项	22
3.3.	Shell 常用命令介绍	23



二、 Apache Hadoop

1. Hadoop 介绍

Hadoop 是 Apache 旗下的一个用 java 语言实现开源软件框架，是一个开发和运行处理大规模数据的软件平台。允许使用简单的编程模型在大量计算机集群上对大型数据集进行分布式处理。



狭义上说，Hadoop 指 Apache 这款开源框架，它的核心组件有：

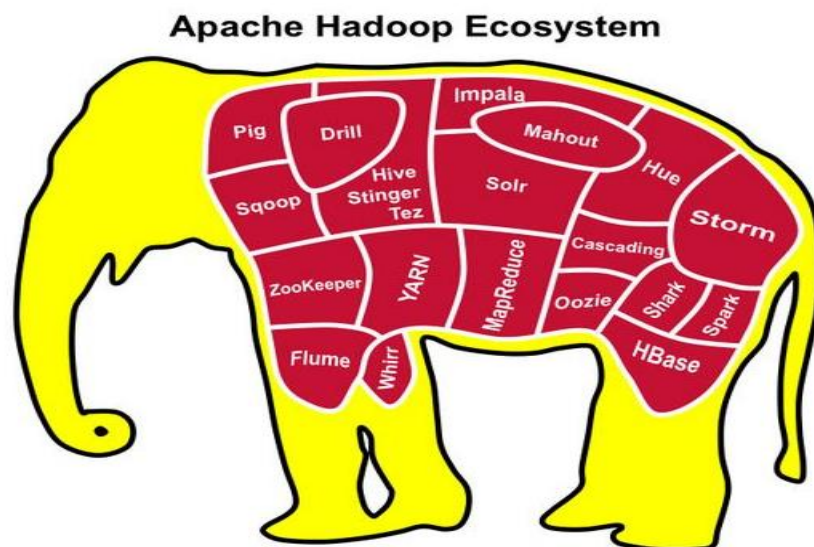
HDFS（分布式文件系统）：解决海量数据存储

YARN（作业调度和集群资源管理的框架）：解决资源任务调度

MAPREDUCE（分布式运算编程框架）：解决海量数据计算



广义上来说，Hadoop 通常是指一个更广泛的概念——Hadoop 生态圈。



当下的 Hadoop 已经成长为一个庞大的体系，随着生态系统的成长，新出现的项目越来越多，其中不乏一些非 Apache 主管的项目，这些项目对 HADOOP 是很好的补充或者更高层的抽象。比如：

HDFS：分布式文件系统

MAPREDUCE：分布式运算程序开发框架

HIVE：基于 HADOOP 的分布式数据仓库，提供基于 SQL 的查询数据操作

HBASE：基于 HADOOP 的分布式海量数据库

ZOOKEEPER：分布式协调服务基础组件

Mahout：基于 mapreduce/spark/flink 等分布式运算框架的机器学习算法库

Oozie：工作流调度框架

Sqoop：数据导入导出工具（比如用于 mysql 和 HDFS 之间）

Flume：日志数据采集框架

Impala：基于 Hadoop 的实时分析



2. Hadoop 发展简史

Hadoop 是 Apache Lucene 创始人 Doug Cutting 创建的。最早起源于 Nutch，它是 Lucene 的子项目。Nutch 的设计目标是构建一个大型的全网搜索引擎，包括网页抓取、索引、查询等功能，但随着抓取网页数量的增加，遇到了严重的可扩展性问题：如何解决数十亿网页的存储和索引问题。

2003 年 Google 发表了一篇论文为该问题提供了可行的解决方案。论文中描述的是谷歌的产品架构，该架构称为：谷歌分布式文件系统（GFS），可以解决他们在网页爬取和索引过程中产生的超大文件的存储需求。

2004 年 Google 发表论文向全世界介绍了谷歌版的 MapReduce 系统。

同时期，Nutch 的开发人员完成了相应的开源实现 HDFS 和 MAPREDUCE，并从 Nutch 中剥离成为独立项目 HADOOP，到 2008 年 1 月，HADOOP 成为 Apache 顶级项目，迎来了它的快速发展期。

2006 年 Google 发表了论文是关于 BigTable 的，这促使了后来的 Hbase 的发展。

因此，Hadoop 及其生态圈的发展离不开 Google 的贡献。

3. Hadoop 特性优点

扩容能力 (Scalable): Hadoop 是在可用的计算机集群间分配数据并完成计算任务的，这些集群可用方便的扩展到数以千计的节点中。

成本低 (Economical): Hadoop 通过普通廉价的机器组成服务器集群来分发以及处理数据，以至于成本很低。

高效率 (Efficient): 通过并发数据，Hadoop 可以在节点之间动态并行的移动数据，使得速度非常快。

可靠性 (Reliable): 能自动维护数据的多份复制，并且在任务失败后能自动地重新部署 (redploy) 计算任务。所以 Hadoop 的按位存储和处理数据的能力值得人们信赖。

4. Hadoop 国内外应用

不管是国内还是国外，Hadoop 最受青睐的行业是互联网领域，可以说互联网公司都是 Hadoop 的主要使用力量。

国外来说，Yahoo、Facebook、IBM 等公司都大量使用 Hadoop 集群来支撑业务。比如：

Yahoo 的 Hadoop 应用在支持广告系统、用户行为分析、支持 Web 搜索等。

Facebook 主要使用 Hadoop 存储内部日志与多维数据，并以此作为报告、分析和机器学习的数据源。

国内来说，BAT 领头的互联网公司是当仁不让的 Hadoop 使用者、维护者。比如 Ali 云梯（14 年国内最大 Hadoop 集群）、百度的日志分析平台、推荐引擎系统等。



国内其他非互联网领域也有不少 Hadoop 的应用，比如：

金融行业：个人征信分析

证券行业：投资模型分析

交通行业：车辆、路况监控分析

电信行业：用户上网行为分析

总之：Hadoop 并不会跟某种具体的行业或者某个具体的业务挂钩，它只是一种用来做海量数据分析处理的工具。



三、 Hadoop 集群搭建

1. 发行版本

Hadoop 发行版本分为开源**社区版**和**商业版**，社区版是指由 Apache 软件基金会维护的版本，是官方维护的版本体系。商业版 Hadoop 是指由第三方商业公司在社区版 Hadoop 基础上进行了一些修改、整合以及各个服务组件兼容性测试而发行的版本，比较著名的有 cloudera 的 CDH、mapR 等。

我们学习的是社区版：Apache Hadoop。后续如未说明都是指 Apache 版本。

Hadoop 的版本很特殊，是由多条分支并行的发展着。大的来看分为 3 个大的系列版本：1.x、2.x、3.x。

Hadoop1.0 由一个分布式文件系统 HDFS 和一个离线计算框架 MapReduce 组成。

Hadoop 2.0 则包含一个支持 NameNode 横向扩展的 HDFS，一个资源管理系统 YARN 和一个运行在 YARN 上的离线计算框架 MapReduce。相比于 Hadoop1.0，Hadoop 2.0 功能更加强大，且具有更好的扩展性、性能，并支持多种计算框架。

Hadoop 3.0 相比之前的 Hadoop 2.0 有一系列的功能增强。但目前还是个 alpha 版本，有很多 bug，且不能保证 API 的稳定和质量。

我们课程中使用的是当前 2 系列最稳定版本：**Apache Hadoop 2.7.4**。

2. 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、SecondaryNameNode

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？它其实是一个分布式运算编程框架，是应用程序开发包，由用户按照编程规范进行程序开发，后打包运行在 HDFS 集群上，并且受

到 YARN 集群的资源调度管理。

Hadoop 部署方式分三种，Standalone mode(独立模式)、Pseudo-Distributed mode(伪分布式模式)、Cluster mode(群集模式)，其中前两种都是在单机部署。

独立模式又称为单机模式，仅 1 个机器运行 1 个 java 进程，主要用于调试。

伪分布模式也是在 1 个机器上运行 HDFS 的 NameNode 和 DataNode、YARN 的 ResourceManger 和 NodeManager，但分别启动单独的 java 进程，主要用于调试。

集群模式主要用于生产环境部署。会使用 N 台主机组成一个 Hadoop 集群。这种部署模式下，主节点和从节点会分开部署在不同的机器上。

我们以 3 节点为例进行搭建，角色分配如下：

node-01	NameNode	DataNode	ResourceManager
node-02	DataNode	NodeManager	SecondaryNameNode
node-03	DataNode	NodeManager	

3. 服务器准备

本案例使用 VMware Workstation Pro 虚拟机创建虚拟服务器来搭建 HADOOP 集群，所用软件及版本如下：

VMware Workstation Pro 12.0

Centos 6.7 64bit

4. 网络环境准备

采用 NAT 方式联网。

如果创建的是桌面版的 Centos 系统，可以在安装完毕后通过图形页面进行编辑。如果是 mini 版本的，可通过编辑 ifcfg-eth*配置文件进行配置。

注意 BOOTPROTO、GATEWAY、NETMASK。



5. 服务器系统设置

同步时间

#手动同步集群各机器时间

```
date -s "2017-03-03 03:03:03"
```

```
yum install ntpdate
```

#网络同步时间

```
ntpdate cn.pool.ntp.org
```

设置主机名

```
vi /etc/sysconfig/network
```

```
NETWORKING=yes
```

```
HOSTNAME=node-1
```

配置 IP、主机名映射

```
vi /etc/hosts
```

```
192.168.33.101    node-1
```

```
192.168.33.102    node-2
```

```
192.168.33.103    node-3
```

配置 ssh 免密登陆

#生成 ssh 免登陆密钥

```
ssh-keygen -t rsa （四个回车）
```

执行完这个命令后，会生成 id_rsa（私钥）、id_rsa.pub（公钥）

将公钥拷贝到要免密登陆的目标机器上

```
ssh-copy-id node-02
```

配置防火墙

#查看防火墙状态

```
service iptables status
```



```
#关闭防火墙
service iptables stop
#查看防火墙开机启动状态
chkconfig iptables --list
#关闭防火墙开机启动
chkconfig iptables off
```

6. JDK 环境安装

```
#上传 jdk 安装包
jdk-8u65-linux-x64.tar.gz
#解压安装包
tar zxvf jdk-8u65-linux-x64.tar.gz -C /root/apps
#配置环境变量 /etc/profile
export JAVA_HOME=/root/apps/jdk1.8.0_65
export PATH=$PATH:$JAVA_HOME/bin
export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
#刷新配置
source /etc/profile
```

7. Hadoop 安装包目录结构

解压 hadoop-2.7.4-with-centos-6.7.tar.gz，目录结构如下：

bin: Hadoop 最基本的管理脚本和使用脚本的目录，这些脚本是/sbin目录下管理脚本的基础实现，用户可以直接使用这些脚本管理和使用Hadoop。

etc: Hadoop 配置文件所在的目录，包括core-site.xml、hdfs-site.xml、mapred-site.xml 等从Hadoop1.0继承而来的配置文件和yarn-site.xml等Hadoop2.0新增的配置文件。

include: 对外提供的编程库头文件（具体动态库和静态库在lib目录中），这些头文件均是用C++定义的，通常用于C++程序访问HDFS或者编写MapReduce程序。



lib: 该目录包含了 Hadoop 对外提供的编程动态库和静态库，与 include 目录中的头文件结合使用。

libexec: 各个服务对用的 shell 配置文件所在的目录，可用于配置日志输出、启动参数（比如 JVM 参数）等基本信息。

sbin: Hadoop 管理脚本所在的目录，主要包含 HDFS 和 YARN 中各类服务的启动/关闭脚本。

share: Hadoop 各个模块编译后的 jar 包所在的目录。

8. Hadoop 配置文件修改

Hadoop 安装主要就是配置文件的修改，一般在主节点进行修改，完毕后 scp 下发给其他各个从节点机器。

8.1. hadoop-env.sh

文件中设置的是 Hadoop 运行时需要的环境变量。`JAVA_HOME` 是必须设置的，即使我们当前的系统中设置了 `JAVA_HOME`，它也是不认识的，因为 Hadoop 即使是在本机上执行，它也是把当前的执行环境当成远程服务器。

```
vi hadoop-env.sh  
  
export JAVA_HOME=/root/apps/jdk1.8.0_65
```

8.2. core-site.xml

hadoop 的核心配置文件，有默认的配置项 `core-default.xml`。

`core-default.xml` 与 `core-site.xml` 的功能是一样的，如果在 `core-site.xml` 里没有配置的属性，则会自动会获取 `core-default.xml` 里的相同属性的值。

```
<!-- 用于设置 Hadoop 的文件系统，由 URI 指定 -->  
  
<property>  
  
    <name>fs.defaultFS</name>  
  
    <value>hdfs://node-1:9000</value>  
  
</property>
```



```
<!-- 配置 Hadoop 的临时目录,默认/tmp/hadoop-${user.name} -->
```

```
<property>
```

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

```
<value>/home/hadoop/hadoop-2.4.1/tmp</value>
```

```
</property>
```

8.3. hdfs-site.xml

HDFS 的核心配置文件，有默认的配置项 hdfs-default.xml。

hdfs-default.xml 与 hdfs-site.xml 的功能是一样的，如果在 hdfs-site.xml 里没有配置的属性，则会自动会获取 hdfs-default.xml 里的相同属性的值。

```
<!-- 指定 HDFS 副本的数量 -->
```

```
<property>
```

```
<name>dfs.replication</name>
```

```
<value>2</value>
```

```
</property>
```

```
<!-- secondary namenode 所在主机的 ip 和端口-->
```

```
<property>
```

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

```
<value>192.168.1.152:50090</value>
```

```
</property>
```

8.4. mapred-site.xml

MapReduce 的核心配置文件，有默认的配置项 mapred-default.xml。

mapred-default.xml 与 mapred-site.xml 的功能是一样的，如果在 mapred-site.xml 里没有配置的属性，则会自动会获取 mapred-default.xml 里的相同属性的值。

```
<!-- 指定 mr 运行时框架，这里指定在 yarn 上，默认是 local -->
```

```
<property>
```



```
<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>
```

8.5. yarn-site.xml

YARN 的核心配置文件，有默认的配置项 yarn-default.xml。

yarn-default.xml 与 yarn-site.xml 的功能是一样的，如果在 yarn-site.xml 里没有配置的属性，则会自动会获取 yarn-default.xml 里的相同属性的值。

```
<!-- 指定 YARN 的老大（ResourceManager）的地址 -->

<property>

    <name>yarn.resourcemanager.hostname</name>

    <value>node-1</value>

</property>

<!-- NodeManager 上运行的附属服务。需配置成 mapreduce_shuffle，才可运行 MapReduce
程序默认值："" -->

<property>

    <name>yarn.nodemanager.aux-services</name>

    <value>mapreduce_shuffle</value>

</property>
```

8.6. slaves

slaves 文件里面记录的是集群主机名。一般有以下两种作用：

一是：配合一键启动脚本如 start-dfs.sh、stop-yarn.sh 用来进行集群启动。这时候 slaves 文件里面的主机标记的就是从节点角色所在的机器。

二是：可以配合 hdfs-site.xml 里面 dfs.hosts 属性形成一种白名单机制。dfs.hosts 指定一个文件，其中包含允许连接到 NameNode 的主机列表。必须指定文件的完整路径名。如果值为空，则允许所有主机。例如：

```
<property>
```



```
<name> dfs.hosts </name>

<value>/root/apps/hadoop/etc/hadoop/slaves </value>

</property>
```

那么所有在 slaves 中的主机才可以加入的集群中。

9. Hadoop 环境变量

编辑环境变量的配置文件：

```
vi /etc/profile

export JAVA_HOME= /root/apps/jdk1.8.0_65

export HADOOP_HOME= /root/apps/hadoop-2.7.4

export PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
```

保存配置文件，刷新配置文件：

```
source /etc/profile
```



四、 Hadoop 集群启动、初体验

1. 启动方式

要启动 Hadoop 集群，需要启动 HDFS 和 YARN 两个集群。

注意：**首次启动 HDFS 时，必须对其进行格式化操作**。本质上是一些清理和准备工作，因为此时的 HDFS 在物理上还是不存在的。

```
hdfs namenode -format 或者 hadoop namenode -format
```

1.1. 单节点逐个启动

在主节点上使用以下命令启动 HDFS NameNode：

```
hadoop-daemon.sh start namenode
```

在每个从节点上使用以下命令启动 HDFS DataNode：

```
hadoop-daemon.sh start datanode
```

在主节点上使用以下命令启动 YARN ResourceManager：

```
yarn-daemon.sh start resourcemanager
```

在每个从节点上使用以下命令启动 YARN nodemanager：

```
yarn-daemon.sh start nodemanager
```

以上脚本位于\$HADOOP_PREFIX/sbin/目录下。如果想要停止某个节点上某个角色，只需要把命令中的 **start** 改为 **stop** 即可。

1.2. 脚本一键启动

如果配置了 etc/hadoop/slaves 和 ssh 免密登录，则可以使用程序脚本启动所有 Hadoop 两个集群的相关进程，在主节点所设定的机器上执行。

```
hdfs: $HADOOP_PREFIX/sbin/start-dfs.sh
```

```
yarn: $HADOOP_PREFIX/sbin/start-yarn.sh
```

停止集群： `stop-dfs.sh`、`stop-yarn.sh`

2. 集群 web-ui

一旦 Hadoop 集群启动并运行，可以通过 web-ui 进行集群查看，如下所述：

NameNode `http://nn_host:port/` 默认 **50070**.

ResourceManager `http://rm_host:port/` 默认 **8088**.

Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities

Overview 'node-21:9000' (active)

Started:	Thu Aug 31 15:28:35 CST 2017
Version:	2.7.4, rUnknown
Compiled:	2017-08-23T13:31Z by root from Unknown
Cluster ID:	CID-9b33f405-9a09-4541-b9dc-f9863784fc9b
Block Pool ID:	BP-923523435-192.168.30.121-1504145374347

Summary

Security is off.


Safemode is off.

15 files and directories, 4 blocks = 19 total filesystem object(s).

Heap Memory used 48.13 MB of 156.5 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 39.06 MB of 40.09 MB Committed Non Heap Memory. Max Non Heap Memory is -1 B.

Configured Capacity:	51.32 GB
DFS Used:	592 KB (1%)



All Applications

Cluster
About
Nodes
Node Labels
Applications
NEW
NEW SAVING
SUBMITTED
ACCEPTED
RUNNING
FINISHED
FAILED
KILLED
Scheduler
Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved
0	0	0	0	0	0 B	24 GB	0 B	0	24	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocated
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>

Show 20 entries

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalState
No data available in table								

Showing 0 to 0 of 0 entries

北京市昌平区建材城西路金燕龙办公楼一层 电话：400-618-9090



3. Hadoop 初体验

3.1. HDFS 使用

从 Linux 本地上传一个文本文件到 hdfs 的 /test/input 目录下

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

```
hadoop fs -put /root/somewords.txt /test/input
```

3.2. 运行 mapreduce 程序

在 Hadoop 安装包的 `hadoop-2.7.4/share/hadoop/mapreduce` 下有官方自带的 mapreduce 程序。我们可以使用如下的命令进行运行测试。

示例程序 jar:

```
hadoop-mapreduce-examples-2.7.4.jar
```

计算圆周率:

```
hadoop jar hadoop-mapreduce-examples-2.7.4.jar pi 20 50
```

关于圆周率的估算,感兴趣的可以查询资料 [Monte Carlo 方法](#) 来计算 Pi 值。



五、 HDFS 入门

1. HDFS 基本概念

1.1. HDFS 介绍

HDFS 是 Hadoop Distribute File System 的简称，意为：Hadoop 分布式文件系统。是 Hadoop 核心组件之一，作为最底层的分布式存储服务而存在。

分布式文件系统解决的问题就是大数据存储。它们是横跨在多台计算机上的存储系统。分布式文件系统在大数据时代有着广泛的应用前景，它们为存储和处理超大规模数据提供所需的扩展能力。

1.2. HDFS 设计目标

- 1) 硬件故障是常态，HDFS 将有成百上千的服务器组成，每一个组成部分都有可能出现故障。因此故障的检测和自动快速恢复是 HDFS 的核心架构目标。
- 2) HDFS 上的应用与一般的应用不同，它们主要是以流式读取数据。HDFS 被设计成适合批量处理，而不是用户交互式的。相较于数据访问的反应时间，更注重数据访问的高吞吐量。
- 3) 典型的 HDFS 文件大小是 GB 到 TB 的级别。所以，HDFS 被调整成支持大文件。它应该提供很高的聚合数据带宽，一个集群中支持数百个节点，一个集群中还应该支持千万级别的文件。
- 4) 大部分 HDFS 应用对文件要求的是 write-one-read-many 访问模型。一个文件一旦创建、写入、关闭之后就不需要修改了。这一假设简化了数据一致性问题，使高吞吐量的数据访问成为可能。
- 5) 移动计算的代价比之移动数据的代价低。一个应用请求的计算，离它操作的数据越近就越高效，这在数据达到海量级别的时候更是如此。将计算移动到数据附近，比之将数据移动到应用所在显然更好。
- 6) 在异构的硬件和软件平台上的可移植性。这将推动需要大数据集的应用更广泛地采用 HDFS 作为平台。



2. HDFS 重要特性

首先，它是一个文件系统，用于存储文件，通过统一的命名空间目录树来定位文件；

其次，它是分布式的，由很多服务器联合起来实现其功能，集群中的服务器有各自的角色。

2.1. master/slave 架构

HDFS 采用 master/slave 架构。一般一个 HDFS 集群是有一个 Namenode 和一定数目的 Datanode 组成。Namenode 是 HDFS 集群主节点，Datanode 是 HDFS 集群从节点，两种角色各司其职，共同协调完成分布式的文件存储服务。

2.2. 分块存储

HDFS 中的文件在物理上是分块存储（block）的，块的大小可以通过配置参数来规定，默认大小在 hadoop2.x 版本中是 128M。

2.3. 名字空间（NameSpace）

HDFS 支持传统的层次型文件组织结构。用户或者应用程序可以创建目录，然后将文件保存在这些目录里。文件系统名字空间的层次结构和大多数现有的文件系统类似：用户可以创建、删除、移动或重命名文件。

Namenode 负责维护文件系统的名字空间，任何对文件系统名字空间或属性的修改都将被 Namenode 记录下来。

HDFS 会给客户端提供一个统一的抽象目录树，客户端通过路径来访问文件，形如：hdfs://namenode:port/dir-a/dir-b/dir-c/file.data。

2.4. Namenode 元数据管理

我们把目录结构及文件分块位置信息叫做元数据。Namenode 负责维护整个 hdfs 文件系统的目录树结构，以及每一个文件所对应的 block 块信息（block 的 id，及所在的 datanode 服务器）。



2.5. Datanode 数据存储

文件的各个 block 的具体存储管理由 datanode 节点承担。每一个 block 都可以在多个 datanode 上。Datanode 需要定时向 Namenode 汇报自己持有的 block 信息。

存储多个副本（副本数量也可以通过参数设置 `dfs.replication`，默认是 3）。

2.6. 副本机制

为了容错，文件的所有 block 都会有副本。每个文件的 block 大小和副本系数都是可配置的。应用程序可以指定某个文件的副本数目。副本系数可以在文件创建的时候指定，也可以在之后改变。

2.7. 一次写入，多次读出

HDFS 是设计成适应一次写入，多次读出的场景，且不支持文件的修改。

正因为如此，HDFS 适合用来做大数据分析的底层存储服务，并不适合用来做网盘等应用，因为，修改不方便，延迟大，网络开销大，成本太高。



3. HDFS 基本操作

3.1. Shell 命令行客户端

Hadoop 提供了文件系统的 shell 命令行客户端，使用方法如下：

```
hadoop fs <args>
```

文件系统 shell 包括与 Hadoop 分布式文件系统（HDFS）以及 Hadoop 支持的其他文件系统（如本地 FS，HFTP FS，S3 FS 等）直接交互的各种类似 shell 的命令。所有 FS shell 命令都将路径 URI 作为参数。

URI 格式为 scheme://authority/path。对于 HDFS，该 scheme 是 hdfs，对于本地 FS，该 scheme 是 file。scheme 和 authority 是可选的。如果未指定，则使用配置中指定的默认方案。

对于 HDFS，命令示例如下：

```
hadoop fs -ls hdfs://namenode:host/parent/child
```

```
hadoop fs -ls /parent/child fs.defaultFS 中有配置
```

对于本地文件系统，命令示例如下：

```
hadoop fs -ls file:///root/
```

如果使用的文件系统是 HDFS，则使用 hdfs dfs 也是可以的，此时

```
hadoop fs <args> = hdfs dfs <args>
```



3.2. Shell 命令选项

选项名称	使用格式	含义
-ls	-ls <路径>	查看指定路径的当前目录结构
-lsr	-lsr <路径>	递归查看指定路径的目录结构
-du	-du <路径>	统计目录下个文件大小
-dus	-dus <路径>	汇总统计目录下文件(夹)大小
-count	-count [-q] <路径>	统计文件(夹)数量
-mv	-mv <源路径> <目的路径>	移动
-cp	-cp <源路径> <目的路径>	复制
-rm	-rm [-skipTrash] <路径>	删除文件/空白文件夹
-rmr	-rmr [-skipTrash] <路径>	递归删除
-put	-put <多个 linux 上的文件> <hdfs 路径>	上传文件
-copyFromLocal	-copyFromLocal <多个 linux 上的文件> <hdfs 路径>	从本地复制
-moveFromLocal	-moveFromLocal <多个 linux 上的文件> <hdfs 路径>	从本地移动
-getmerge	-getmerge <源路径> <linux 路径>	合并到本地
-cat	-cat <hdfs 路径>	查看文件内容
-text	-text <hdfs 路径>	查看文件内容
-copyToLocal	-copyToLocal [-ignoreCrc] [-crc] [hdfs 源路径] [linux 目的路径]	从本地复制
-moveToLocal	-moveToLocal [-crc] <hdfs 源路径> <li nux 目的路径>	从本地移动
-mkdir	-mkdir <hdfs 路径>	创建空白文件夹
-setrep	-setrep [-R] [-w] <副本数> <路径>	修改副本数量
-touchz	-touchz <文件路径>	创建空白文件
-stat	-stat [format] <路径>	显示文件统计信息
-tail	-tail [-f] <文件>	查看文件尾部信息
-chmod	-chmod [-R] <权限模式> [路径]	修改权限
-chown	-chown [-R] [属主][:[属组]] 路径	修改属主
-chgrp	-chgrp [-R] 属组名称 路径	修改属组
-help	-help [命令选项]	帮助



3.3. Shell 常用命令介绍

-ls

使用方法: `hadoop fs -ls [-h] [-R] <args>`

功能: 显示文件、目录信息。

示例: `hadoop fs -ls /user/hadoop/file1`

-mkdir

使用方法: `hadoop fs -mkdir [-p] <paths>`

功能: 在 hdfs 上创建目录, -p 表示会创建路径中的各级父目录。

示例: `hadoop fs -mkdir -p /user/hadoop/dir1`

-put

使用方法: `hadoop fs -put [-f] [-p] [-|<localsrc1> ..]. <dst>`

功能: 将单个 src 或多个 srcs 从本地文件系统复制到目标文件系统。

-p: 保留访问和修改时间, 所有权和权限。

-f: 覆盖目的地 (如果已经存在)

示例: `hadoop fs -put -f localfile1 localfile2 /user/hadoop/hadoopdir`

-get

使用方法: `hadoop fs -get [-ignorecrc] [-crc] [-p] [-f] <src> <localdst>`

-ignorecrc: 跳过对下载文件的 CRC 检查。

-crc: 为下载的文件写 CRC 校验和。

功能: 将文件复制到本地文件系统。

示例: `hadoop fs -get hdfs://host:port/user/hadoop/file localfile`

-appendToFile

使用方法: `hadoop fs -appendToFile <localsrc> ... <dst>`

功能: 追加一个文件到已经存在的文件末尾

示例: `hadoop fs -appendToFile localfile /hadoop/hadoopfile`



-cat

使用方法：hadoop fs -cat [-ignoreCrc] URI [URI ...]

功能：显示文件内容到 stdout

示例：hadoop fs -cat /hadoop/hadoopfile

-tail

使用方法：hadoop fs -tail [-f] URI

功能：将文件的最后一千字节内容显示到 stdout。

-f 选项将在文件增长时输出附加数据。

示例：hadoop fs -tail /hadoop/hadoopfile

-chgrp

使用方法：hadoop fs -chgrp [-R] GROUP URI [URI ...]

功能：更改文件组的关联。用户必须是文件的所有者，否则是超级用户。

-R 将使改变在目录结构下递归进行。

示例：hadoop fs -chgrp othergroup /hadoop/hadoopfile

-chmod

功能：改变文件的权限。使用-R 将使改变在目录结构下递归进行。

示例：hadoop fs -chmod 666 /hadoop/hadoopfile

-chown

功能：改变文件的拥有者。使用-R 将使改变在目录结构下递归进行。

示例：hadoop fs -chown someuser:somegrp /hadoop/hadoopfile

-copyFromLocal

使用方法：hadoop fs -copyFromLocal <localsrc> URI

功能：从本地文件系统中拷贝文件到 hdfs 路径去

示例：hadoop fs -copyFromLocal /root/1.txt /

-copyToLocal

功能：从 hdfs 拷贝到本地

示例：hadoop fs -copyToLocal /aaa/jdk.tar.gz



-cp

功能：从 hdfs 的一个路径拷贝 hdfs 的另一个路径

示例： `hadoop fs -cp /aaa/jdk.tar.gz /bbb/jdk.tar.gz.2`

-mv

功能：在 hdfs 目录中移动文件

示例： `hadoop fs -mv /aaa/jdk.tar.gz /`

-getmerge

功能：合并下载多个文件

示例：比如 hdfs 的目录 /aaa/下有多个文件:log.1, log.2, log.3,...

`hadoop fs -getmerge /aaa/log.* ./log.sum`

-rm

功能：删除指定的文件。只删除非空目录和文件。-r 递归删除。

示例： `hadoop fs -rm -r /aaa/bbb/`

-df

功能：统计文件系统的可用空间信息

示例： `hadoop fs -df -h /`

-du

功能：显示目录中所有文件大小，当只指定一个文件时，显示此文件的大小。

示例： `hadoop fs -du /user/hadoop/dir1`

-setrep

功能：改变一个文件的副本系数。-R 选项用于递归改变目录下所有文件的副本系数。

示例： `hadoop fs -setrep -w 3 -R /user/hadoop/dir1`