# 大数据生态圈

# 大数据Hadoop框架

## Hadoop概述

### 1.1Hadoop历史

**Apache Lucene**

·全球第一个开源的全文检索引擎工具包

·完整的查询引擎和索引引擎

·部分文本分析引擎

·开发人员可在此基础建立起完整的全文检索引擎

**Nutch**

·开源的基于Lucene的网页搜索引擎

·加入网页抓取、解析等功能

·类似于Google等商业搜索引擎

**Google的两篇重要论文**

·Google File System

·MapReduce：Simplified Data Processing on Large Clusters

·Hadoop官网 <http://hadoop.apache.org>

**Hadoop大事记**

·2002年 Nutch项目开始运行

·2003年 谷歌发表GFS论文

·2004年 Doug根据GFS设计了NDFS

·2005年 Nutch移植到新的框架，Hadoop早期版本在20个节点上运行

·2006年1月 Doug加盟雅虎

·2006年2月 Apache Hadoop项目启动

·2008年 Hadoop赢得世界1 TB数据排序冠军

·2013年11月 Hadoop技术峰会召开，标志Hadoop进入2.0时代

### 1.2创建本地Hadoop集群

**三个问题**

·Hadoop需要的环境支持

·初学者应该选用哪个版本的Hadoop

·如何安装和配置Hadoop

**操作系统的支持**

·Linux

·Windows

—Cygwin

—hadoop-for-windows

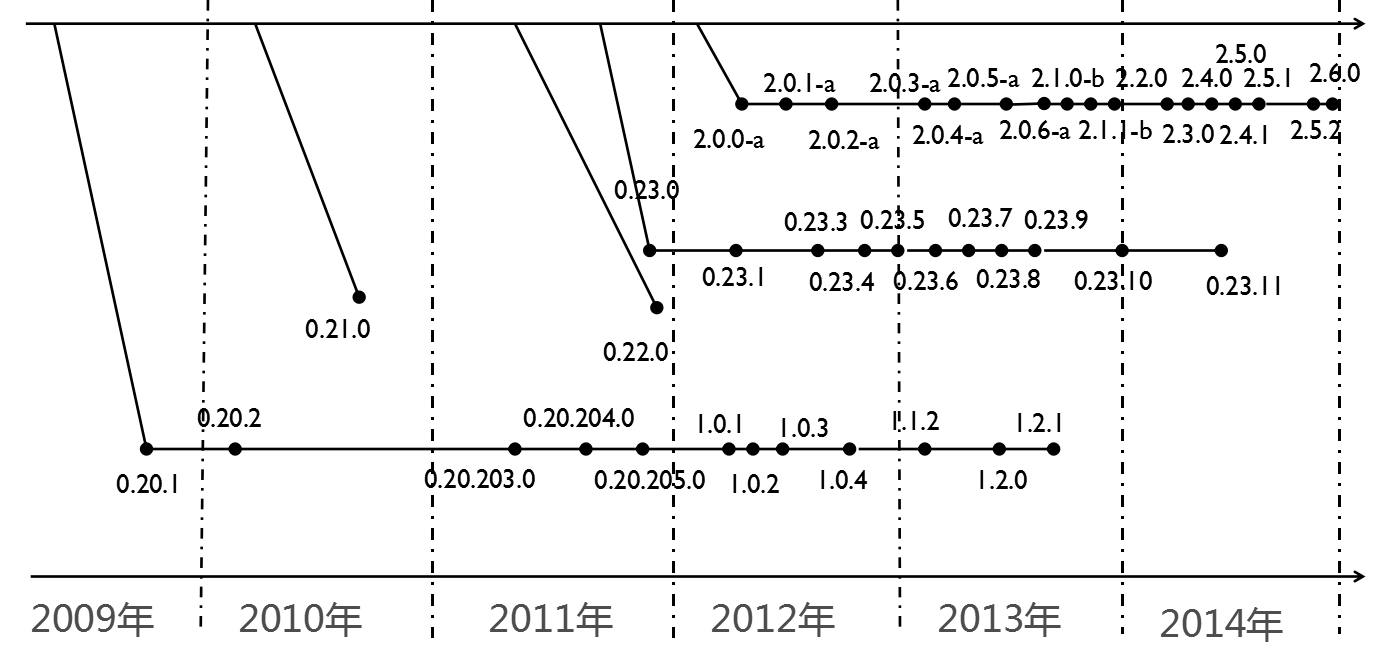
**JDK的支持**

·下载地址

http://www.oracle.com/technetwork/java/javase/downloads/index.html

·设置环境变量

**Hadoop的版本**



五个主要分支：

**0.20系列**

·.203.0 增加文件追加的特性

·.205.0 增加Security特性

·1.0.1-1.0.4 修复1.0.1的bug

·1.1.2 修复135处bug

·1.2.0 修复200处

·1.2.1 稳定版

**0.21.0、0.22.0系列**

·分割的基础库，分布式文件系统，MapReduce三个独立模块

·加入符号链接等特性

**0.23系列**

·Hadoop扩展性和通用性

·基础库，分布式文件系统，资源管理框架，MapReduce四部分

·0.23.1-0.23.11 修复系统bug

**2.0系列**

·加入NameNode热备份方案

·2.0.0-a~2.0.6-a 内部测试版

·2.1.0-b~2.1.1-b Beta测试版

·2.2.0 第一个2.0版本的稳定版

·2.2.0~2.6.0 更新到2.6稳定版

**下载安装**

·http://hadoop.apache.org/releases.html

**设置环境**

·设置环境变量

·设置用户路径

·修改配置文件

**配置SSH免密码登录**

**小结**

·Hadoop所需的环境

·Hadoop的版本演进

·Hadoop的安装步骤

·JDK的安装与配置

·Hadoop下载安装

·SSH免密码登录设置

### 1.3创建托管Hadoop集群

**Amazon Web Services——亚马逊云服务**

·Elastic Compute Cloud (EC2)

·Simple Storage Service(S3)

·Elastic MapReduce （EMR）

**Elastic Compute Cloud (EC2)**

2006年8月25日发布

·弹性云服务器

·虚拟主机

·多种操作系统

·负载均衡

·安全防护

·按使用时长付费

**Simple Storage Service(S3）**

2006年3月14日发布

·简单键值存储服务

·桶

·按存储量付费

·最大文件5TB

**Elastic MapReduce （EMR）**

·2009年4月2日发布

·支持多种接口：网页控制台、命令行、API

·支持多种开发语言

·集群配置简单

·经典模式

**操作示例**

·创建AWS账号

·注册必需的服务

·登陆AWS管理控制台

·创建桶

·创建作业流

**小结**

·AWS简介

·EC2的原理和使用

·S3的原理和使用

·EMR的原理和使用

## 第2节Hadoop架构介绍

·HDFS的体系结构

·MapReduce的体系结构

·Hadoop的运行模式

### 2.1HDFS体系结构

**设计目标**

·自动快速检测应对硬件错误

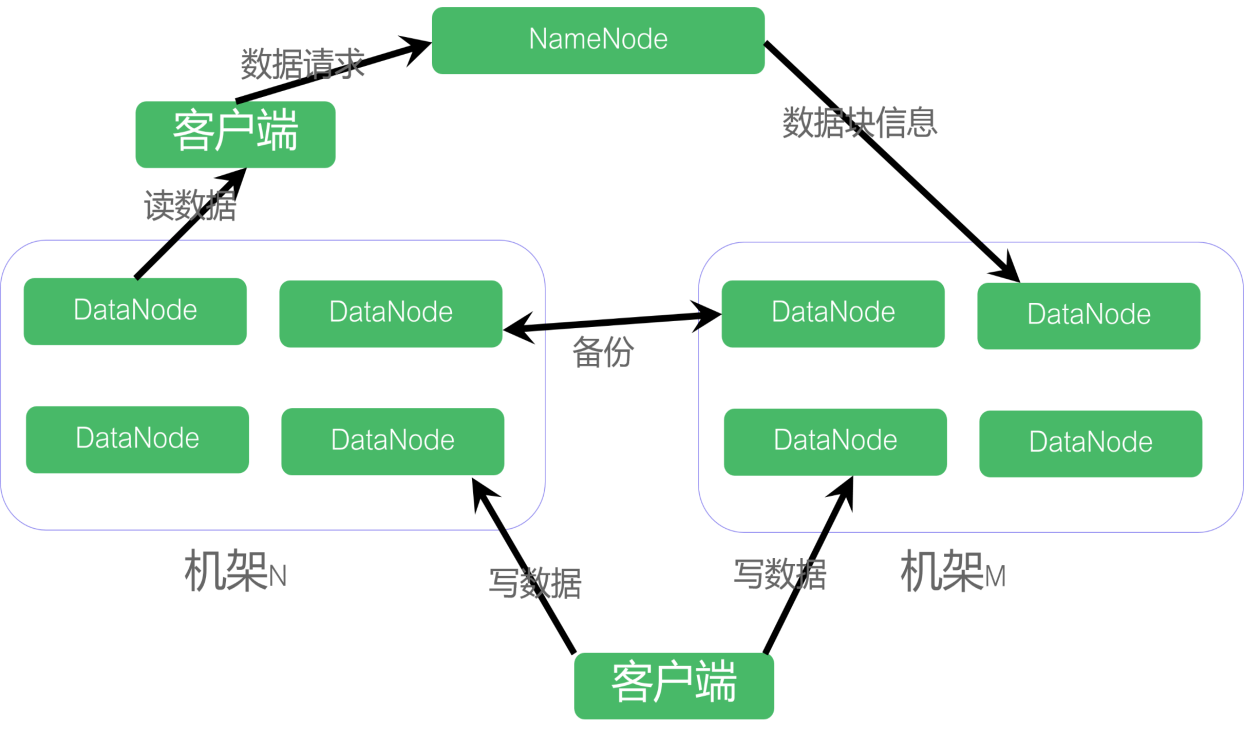
·流式访问数据

·转移计算比移动数据本身更划算

·简单一致性模型

·异构平台可移植

**HDFS的体系结构**



**基本概念**

·机架

·数据块（block）

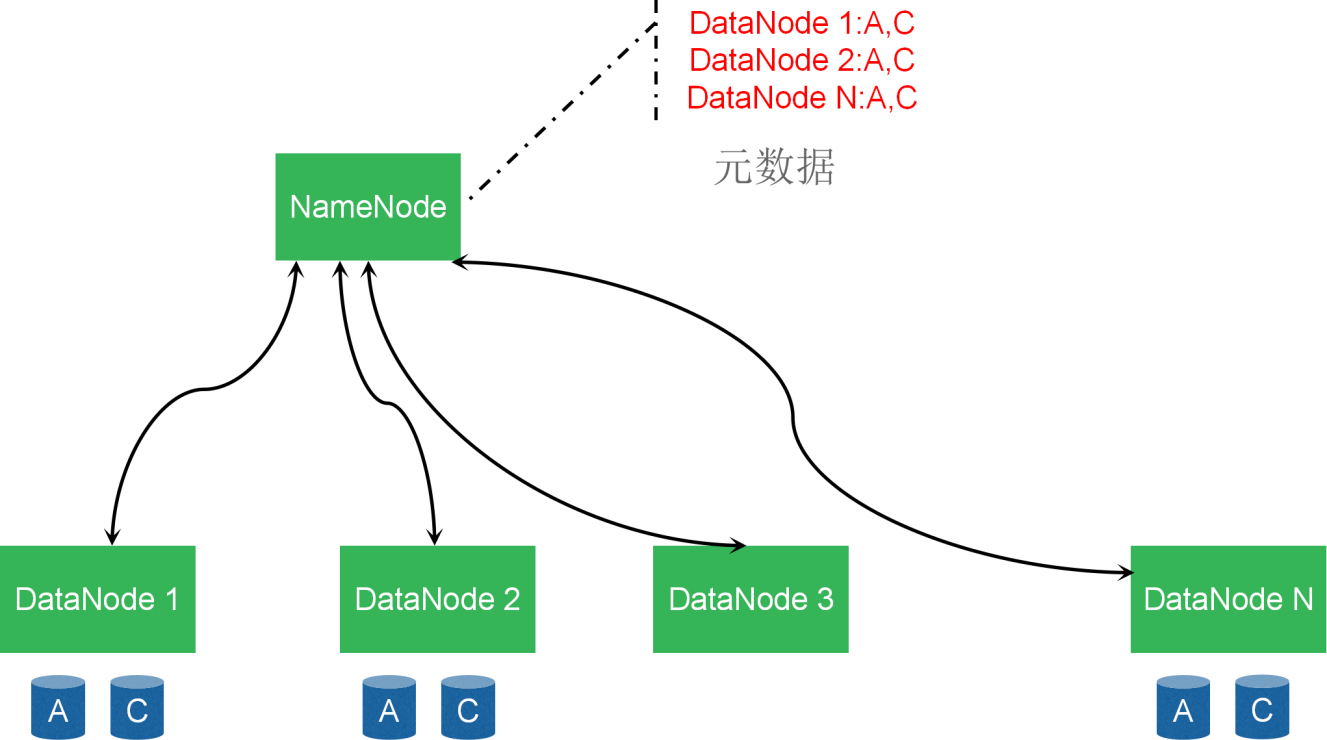
·元数据：镜像文件（Fsimage文件描述）+日志文件（EditLog文件修改)

·用户数据

**主从节点的通信**

·心跳机制（超过m次跟新失败）

·TCP连接

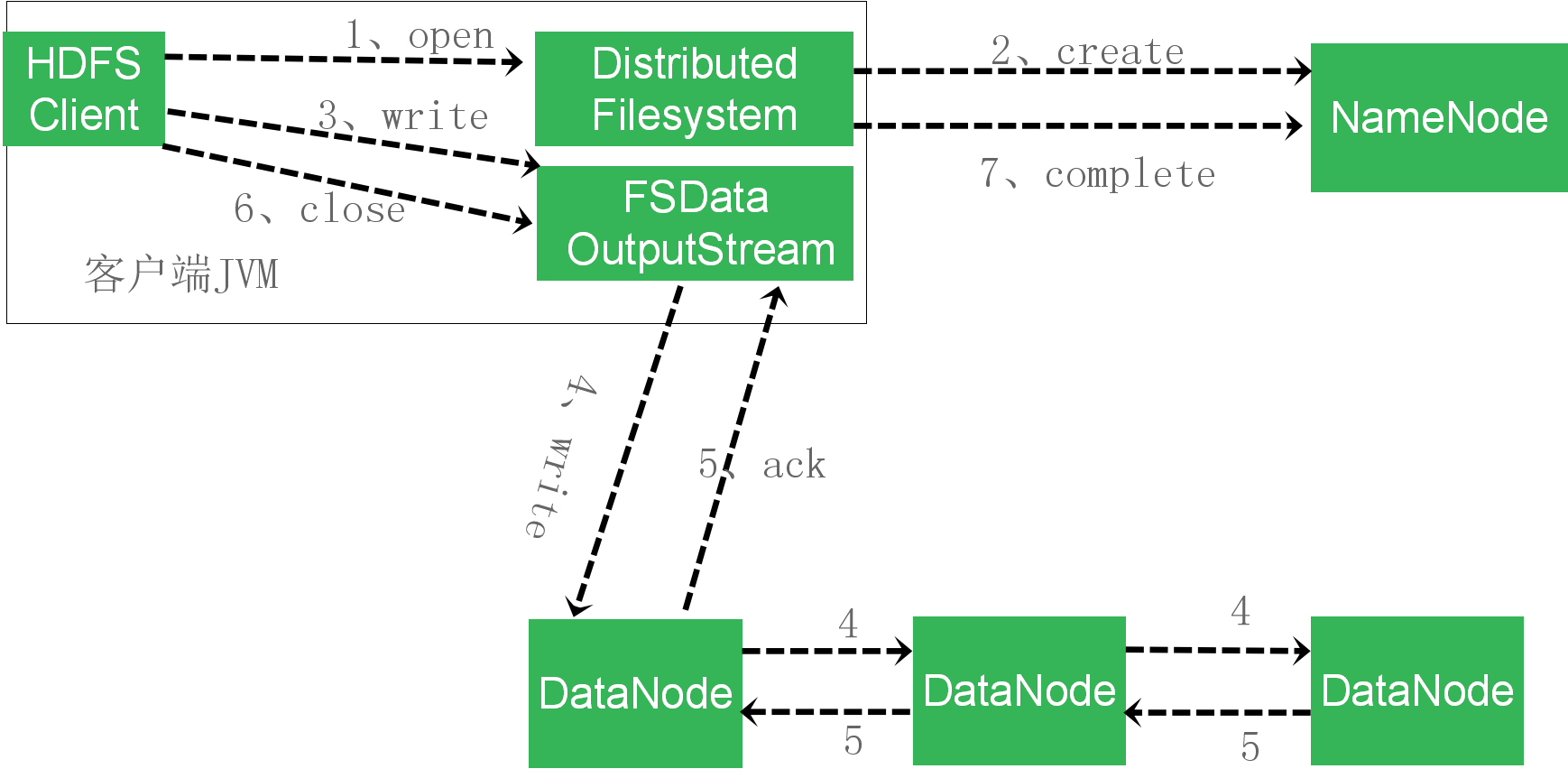


**写数据流程**

·NameNode申请

·副本写入机制

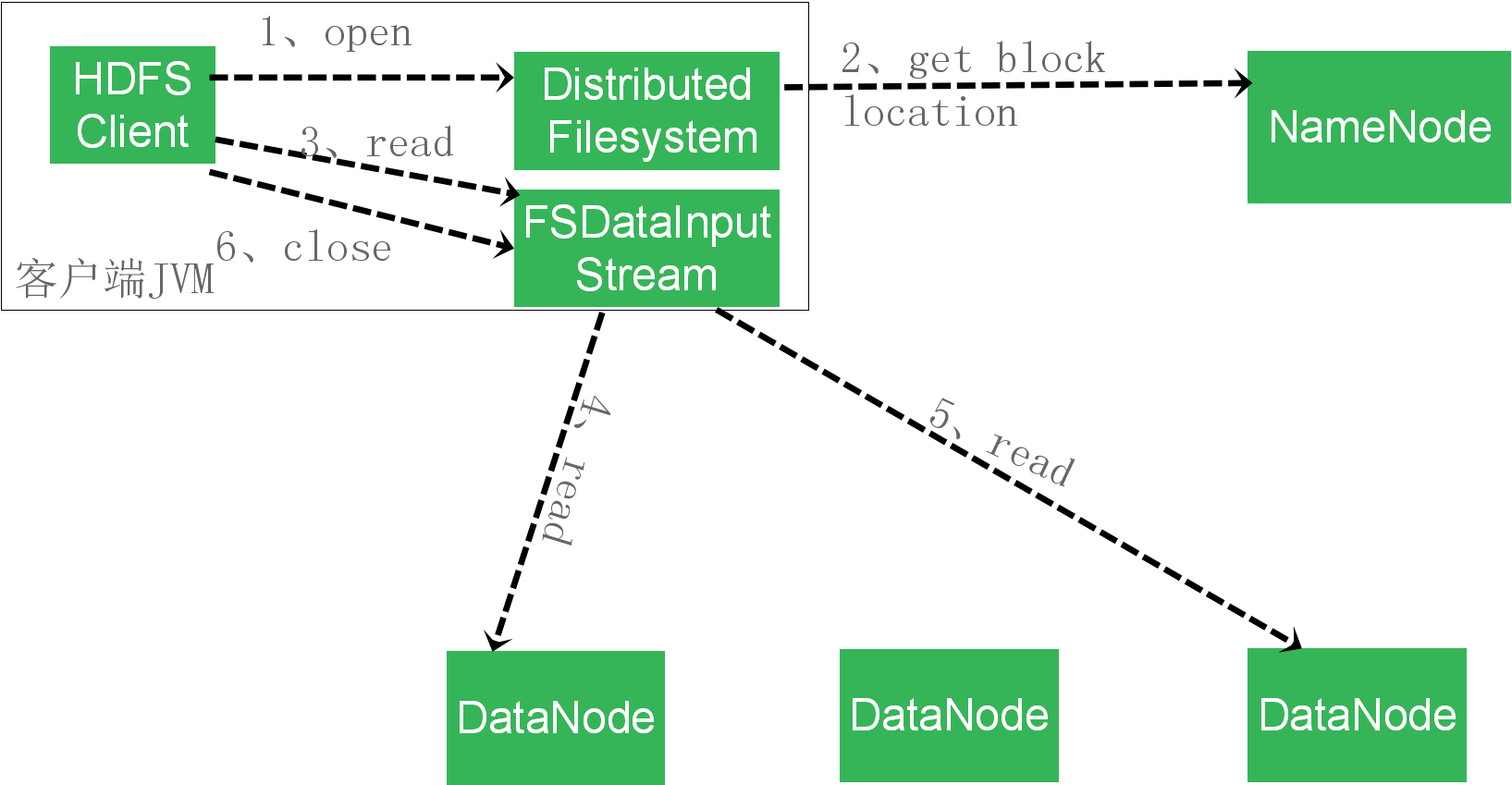
·ack队列



**读数据流程**

·NameNode申请block和location

·读写失败机制



### 2.2MapReduce的体系结构

·分布式编程架构

·以数据为中心，更看重吞吐率

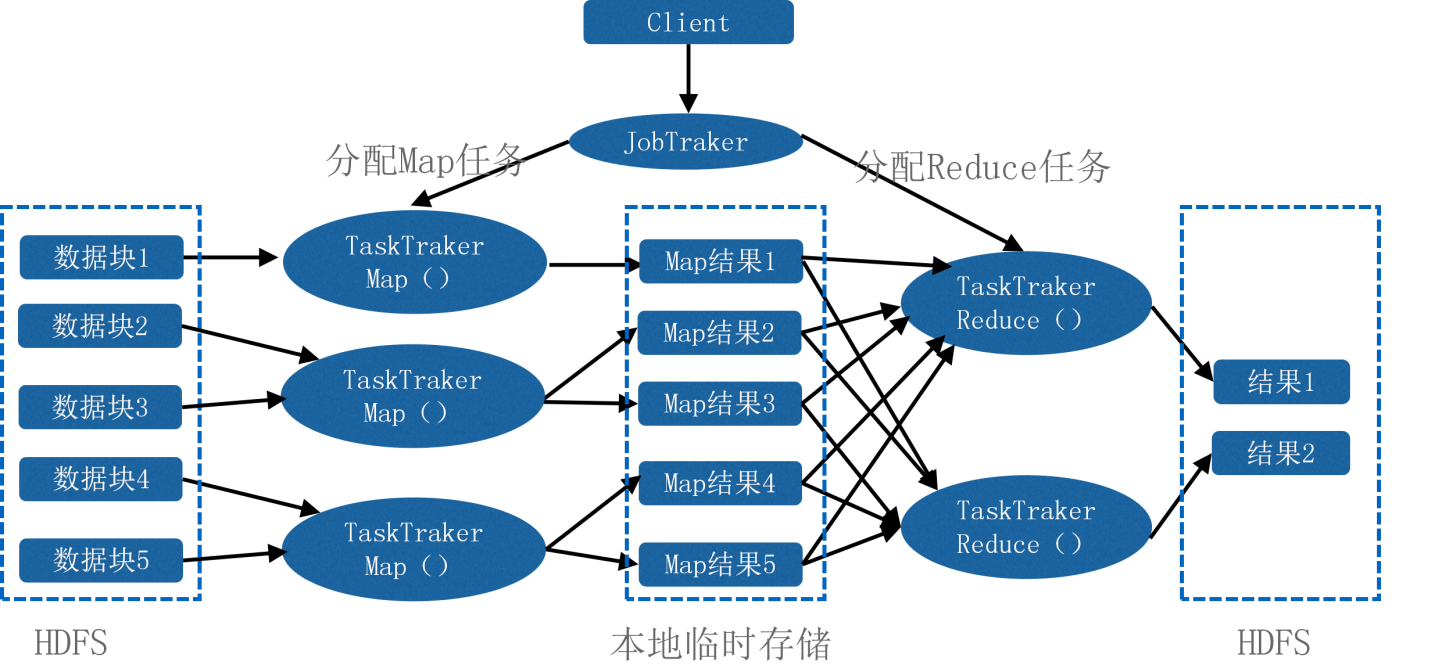
·分而治之

·Map将一个任务分解成多个子任务

·Reduce将分解后多任务分别处理，并将结果汇总为最终结果

·应用举例：清点图书馆藏书、统计单词的出现次数、混合辣椒酱的生成过程等等

**结构图**



**基本概念**

·作业（Job）

·任务（Task）

·键值对（key/value pair）

Map()、Reduce()函数的输入、输出都是<key,value>的形式

**1.提交作业**

·在作业提交之前，需要对作业进行配置

·程序代码，主要是自己编写的MapReduce程序

·配置输入输出路径、输出是否压缩

·配置完成后，通过JobClient来提交

**作业调度**

·FIFO调度器、公平调度器、容量调度器

**2.任务分配**

·TaskTracker和JobTracker之间的通信与任务的分配是通过心跳机制完成的

·TaskTracker会主动向JobTracker询问是否有作业要做，如果自己可以做，那么就会申请到作业任务，这个任务可以使Map也可能是Reduce任务

**3.任务的执行**

·拷贝代码到本地

·拷贝任务的信息到本地

·启动JVM运行任务

**4.状态更新**

·任务在运行过程中，首先会将自己的状态汇报给TaskTracker，然后由TaskTracker汇总告之JobTracker

·任务进度是通过计数器来实现的

·JobTracker是在接受到最后一个任务运行完成后，才会将作业标志为成功

·作业成功后执行删除中间结果等善后处理工作

### 2.3Hadoop的运行模式

**三种模式**

·单机模式

·伪分布式模式

·完全分布式模式

**单机模式**

·默认模式

·不对配置文件进行修改

·使用本地文件系统

·Hadoop的守护进程未启动

·用于对MapReduce程序的逻辑进行调试

**伪分布式模式**

·在一台主机模拟多主机

·每个守护进程都以Java进程的形式运行

·在单机模式之上增加了代码调试功能，允许检查内存使用情况，HDFS输入输出，以及其他的守护进程交互

·修改3个配置文件：core-site.xml、hdfs-site.xml、mapred-site.xml

·格式化文件系统

**完全分布式模式**

·多台主机

·所有主机安装JDK和Hadoop

·设置SSH免密码登陆

·修改3个配置文件： core-site.xml、hdfs-site.xml、mapred-site.xml

·格式化文件系统

**小结**

·HDFS的体系结构和数据读写流程

·MapReduce的体系结构和生命周期

·Hadoop的运行模式及示例演示

## 第3节YARN的发展背景及基本原理

·YARN产生的历史背景

·MapReduce的基本原理及劣势

·YARN的基本原理与功能组件简介

### 3.1YARN产生的历史背景

**前言：什么是Hadoop**

The Apache™ Hadoop® project develops open-source software for reliable, scalable, distributed computing.

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

Hadoop是apache基金会的一个项目，目的是开发一个开源的软件，用于可靠的，可扩展的，分布式的计算。

Hadoop不是一个软件，而是一个软件库，hadoop做为一个软件库，提供了一个框架，可以以分布式的方式，在集群的多台主机之间，使用简单的编程模型来处理大量的数据集。Hadoop被设计成可以从单主机扩展到数千台主机，并且支持本地计算和存储。

Hadoop本身被设计成可以在应用层面检测和处理底层的硬件故障，以实现高可用性，而不是由底层硬件来完成故障的检查和恢复。

**Hadoop中的主要项目**

Hadoop Common

— 为Hadoop的其它项目提供一些常用工具，主要包括系统配置工具Configuration、远程过程调用RPC、序列化机制和Hadoop抽象文件系统FieSystem.

Hadoop Distributed File System (HDFS™)

— Hadoop体系中数据存储管理的基础，实现了对应用程序大吞吐量的访问.

Hadoop YARN

— 作业调度和集群资源管理的一个框架.

Hadoop MapReduce

—基于YARN的一个系统，用于并行处理大量的数据集.

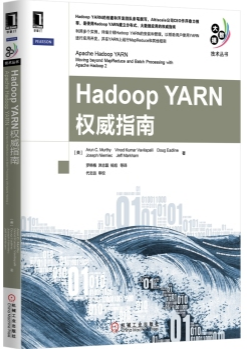
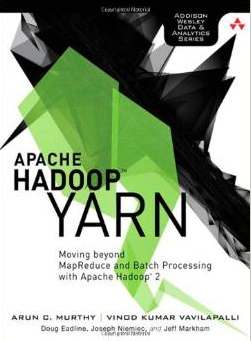
**Hadoop演进的四大阶段**

·阶段0：Ad Hoc集群时代

·阶段1：Hadoop on Demand

·阶段2：共享计算集群的黎明

·阶段3：YARN的出现



要求1：**可扩展性**

可以平滑地扩展至数万节点和并发的应用

#### 阶段0：Ad Hoc集群时代

Ad hoc集群出现之前，用户使用Hadoop与普通的桌面应用方式类似

— 几个节点上创建集群

— 载入数据至HDFS，运行MapReduce任务，得到结果

— 拆掉集群

原因：

— 没有在HDFS持久存储数据的需求

— 没有共享数据和计算结果的动机

Yahoo 实现了共享的HDFS实例，迫切要求实现共享的计算平台

#### 阶段1：Hadoop on Demand

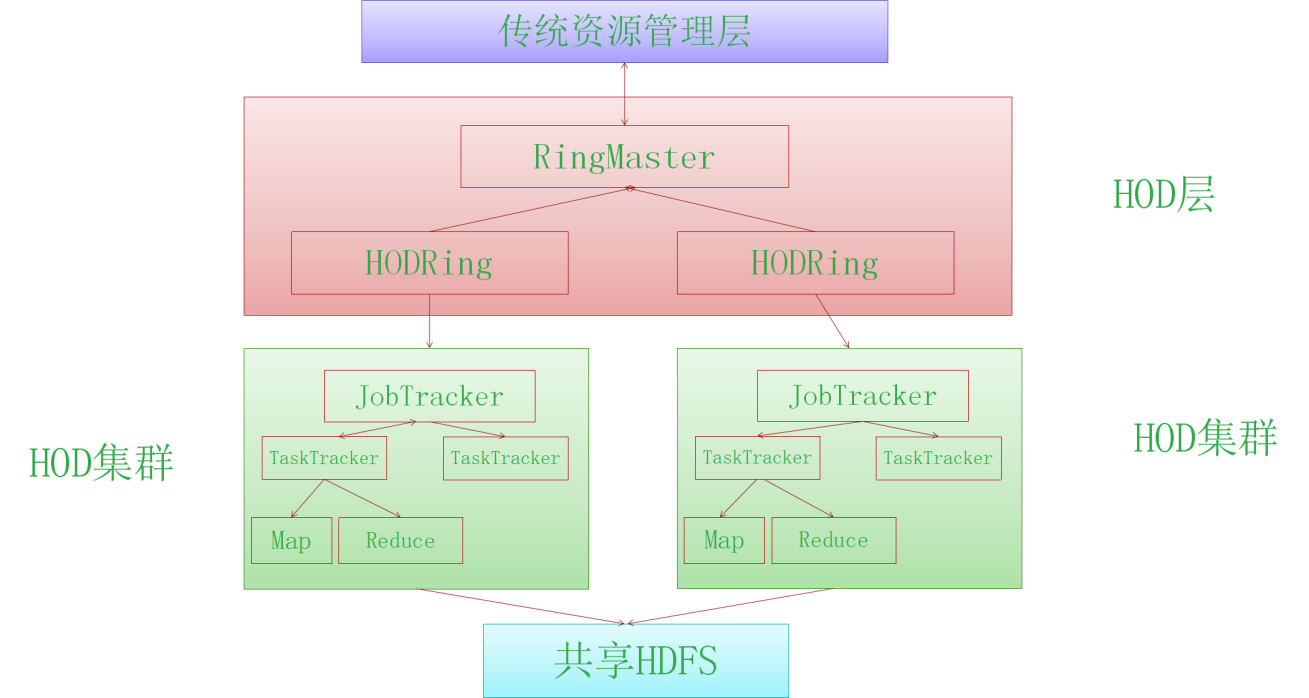
· HOD是一个能在商用硬件的共享集群上提供和管理Hadoop MapReduce和HDFS实例的系统

·使用Torque资源管理器和集群调度器(Maui)分配集群

· 并且在分配的节点上启动MapReduce和HDFS的守护进程响应用户数据和应用的请求

·典型的HOD会话：分配一个集群、在其上运行Hadoop作业、释放集群

**Hadoop on Demand的架构**



**HOD的特点及优势(1/2)**

·HOD为每一个作业建立一个新的集群，当开发人员测试新特性时，用户可以使用稳定的Hadoop版本。

·Hadoop社区每3个月发布一个修订版，为保证软件的正常升级，要求HOD具有相当的灵活性

**要求2：可维护性**

·保证集群软件的升级与用户应用程序完全解耦

**HOD的特点及优势(2/2)**

·HOD的限制：任何时刻集群的每一个节点只属于一个用户的Hadoop集群

**要求3：多租户**

·需要支持在同一个集群中多个租户并存，

·同时支持多个租户之间细粒度地共享单个节点

**HoD的缺点**

·资源管理器Torque不知道HDFS块的分布，分配集群节点时无法考虑本地化

·所以Hadoop中的某些作业，由于Torque得不到足够的信息，大多数读操作会从远程主机处获取

·虽然做了一些改进，但是收获不大，最终Yahoo停用了HOD

**要求4：位置感知**

·将计算移动到数据所在的位置

**HOD的缺点**

·HOD单个作业或者作业之间的阶段集群的规模不能调整，所以集群在容量的主要部分在大多数时间是空闲的，等待后续轻量阶段的完成

·极端但是常见的例子：运行在一个节点上的Reduce任务可能会导致数百节点组成的集群无法回收而处于空闲状态

**要求5：高集群使用率**

实现底层物理资源的高使用率

#### 阶段2：共享计算集群的黎明

·共享计算集群MapReduce

·共享计算集群MapReduce和与之协同工作的共享HDFS实例是Hadoop 1.X版本的主要架构

·共享计算架构包含两大核心组件：

— JobTracker：一个中央守护进程，负责管理集群上的所有作业，虽然和HOD中为单用户运行作业的进程一样，但是多了一些额外的功能

— TaskTracker：系统的从进程，根据JobTracker的指令来执行任务，和HOD的守护进程也是一样的，不同的是TaskTraker负责运行来自所有用户的作业。

**共享MapReduce的特征**

·中央JobTracker守护进程

·JobTracker内存管理

·已完成作业的管理

·中央调度器：Capacity调度器和Fair调度器

·恢复和升级

·单个节点上的隔离

·身份认证和访问控制【要求6】

**要求6：安全和可审计的操作**

·继续以安全的、可审计的方式使用集群资源

**共享MapReduce的问题(1/2)**

·可扩展性瓶颈

·可靠性和可用性【要求7】

·MapReduce编程模型的滥用【要求8】

**要求7：可靠性和可用性**

·具有高度可靠的用户交互，并支持高可用性

**要求8：对编程模型多样性的支持**

·支持多样化的编程模型，并需要演进为不仅仅以MapReduce为中心

**资源模型【要求9】**

**要求9：灵活的资源模型**

·支持各个节点的动态资源配置以及灵活的资源模型

#### 阶段3：YARN的出现

·以前遇到的问题迫切要求新模型的诞生

·新的架构和相关实现尽量重用现有框架的代码、相似的操作模式，并为现有的MapReduce用户提供相同的接口【要求10】

**要求10：向后兼容**

·保持现有MapReduce应用程序的向后兼容

**对YARN的需求**

·要求1：可扩展性

·要求2：可维护性

·要求3：多租户

·要求4：位置感知

·要求5：高集群使用率

·要求6：安全和可审计的操作

·要求7：可靠性和可用性

·要求8：对编程模型多样性的支持

·要求9：灵活的资源模型

·要求10：向后兼容

### 3.2MapReduce的基本原理及劣势

**MapReduce简介**

·Hadoop MapReduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.

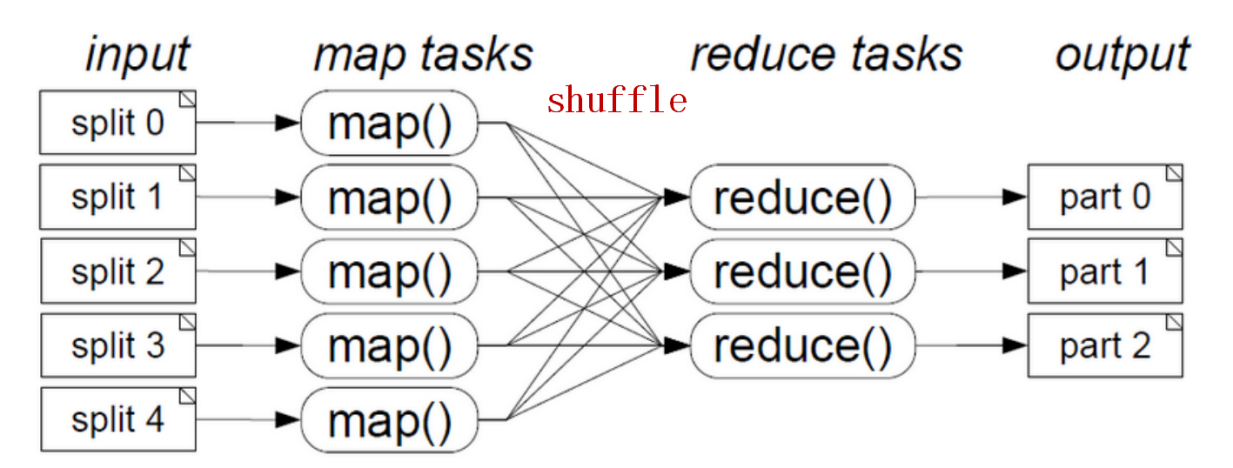
·Mapreduce是一个软件框架，可方便的编写应用程序，以并行的方式在数千商用硬件组成的集群节点中处理数TB级的数据，并且提供了可靠性和容错的能力。

**MapReduce范式**

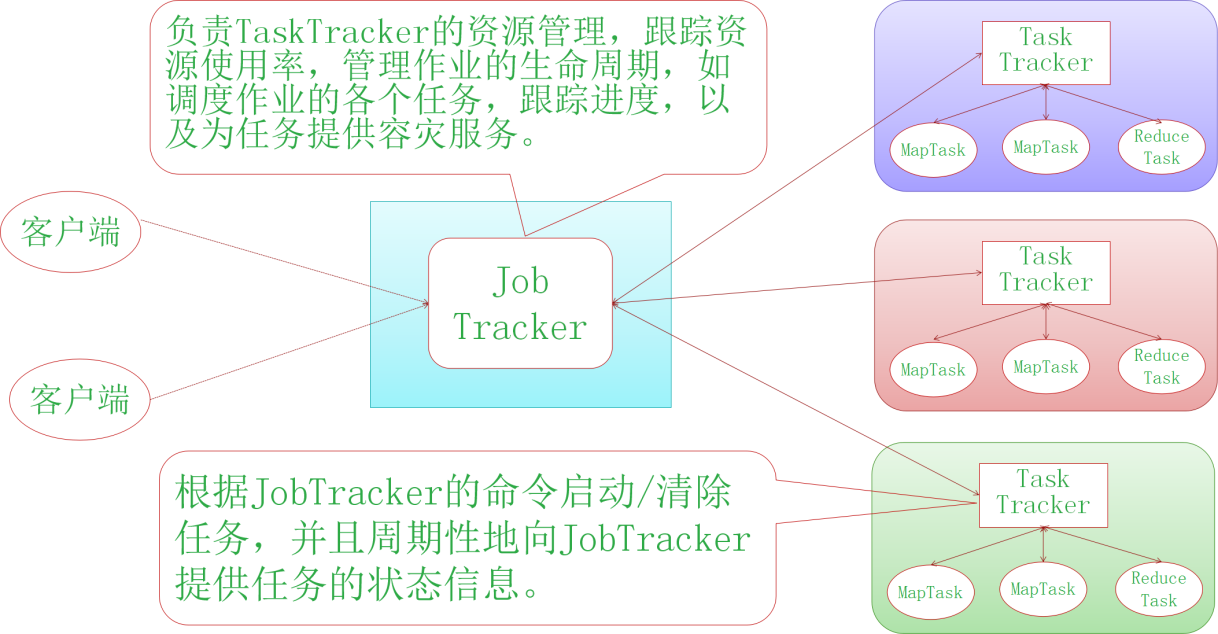
·MapReduce处理模型包括两个独立的步骤：

— 第一步是并行Map阶段，输入数据被分割成离散块以便可以单独处理

— 第二步是Reduce阶段，汇总Map阶段的输出生成预期的结果。



**MapReduce的组件及架构**



**MapReduce的缺陷**

·可扩展性

— JobTracker内存中保存用户作业的信息

— JobTracker使用的是粗粒度的锁

·可靠性和可用性

— JobTracker失效会丢失集群中所有的运行作业，用户需手动重新提交和恢复工作流

·对不同编程模型的支持

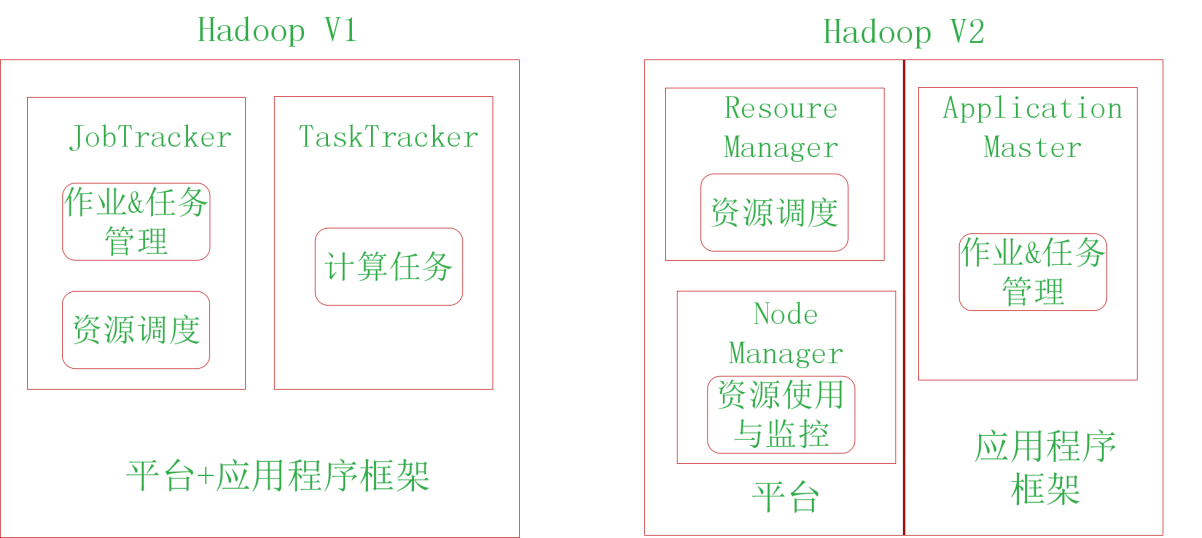
— HadoopV1以MapReduce为中心的设计虽然能支持广泛的用例，但是并不适合所有大型计算

### 3.3YARN的基本原理与功能组件简介

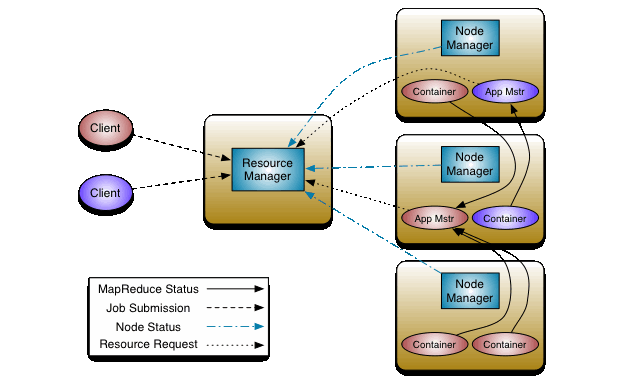
**YARN的设计思想**

·YARN（Yet Another Resource Negotiator）

·YARN的基本思想：将 JobTracker 两个主要的功能分离成单独的组件，一个全局的ResourceManager和每个应用对应的ApplicationMaster



**YARN的控制组件及架构**



**YARN的组件**

·ResourceManager

·ApplicationMaster

·资源模型

·ResourceRequest和Container

**ResourceManager**

·一个纯粹的调度器

·根据应用程序的资源请求严格限制系统的可用资源

·在保证容量、公平性及服务等级的情况下，优化集群资源利用率，让所有资源都得到充分利用

·有可插拔的调度器来应用不同的调度算法，如注重容量调度还是注意公平调度

**ApplicationMaster**

·负责与ResourceManager协商资源，并和NodeManager协同工作来执行和监控Container以及它们的资源消耗

·有责任与ResourceManager协商并获取合适的资源Container，跟踪它们的状态，以及监控其进展

·在真实环境中，每一个应用都有自己的ApplicationMaster实例，但是也可为一组应用提供一个ApplicationMaster，比如Pig或者Hive的ApplicationMaster。

**资源模型**

·YARN提供了通用的应用资源模型

·一个应用可以通过ApplicationMaster请求非常具体的资源

— 资源名称（包括主机名、机架名以及复杂的网络拓朴）

— 内存量

— CPU

— 其他资源：磁盘和网络IO等

**ResourceRequest和Container**

·一个应用程序通过ApplicationMaster请求特定的ResourceRequest来满足资源需求

·Scheduler会分配一个Container来响应资源请求

·Container是一种资源分配形式，为应用程序授予在特定主机上使用资源（如内存、CPU）的权利

·ApplicationMaster取走Container，并且交给NodeManager，NodeManager会利用相应的资源来启动Container的任务进程

**小结**

·Hadoop与YARN的作用

·YARN发展的四个阶段

·YARN应当满足的十个要求

·MapReduce的基本原理

·MapReduce的架构组件

·MapReduce的缺陷

·YARN的设计思想

·YARN的架构及主要组件的作用

### 3.4安装Hadoop YARN

·YARN安装简介

·安装YARN前的准备

·伪分布式安装

#### Hadoop的安装简介

·Hadoop的各个组件均采用XML文件进行配置，这些配置文件都在etc/hadoop子目录中

·core-site.xml文件用于配置通用属性

·hdfs-site.xml文件用于配置HDFS属性

·mapred-site.xml文件用于配置MapReduce属性

·yarn-site.xml用于配置Yarn的属性

**Hadoop的三种运行模式**

·单机模式

·伪分布式

·全分布式

**单机模式**

·单机模式是Hadoop的默认模式。

·因为不需要与其他节点交互，单机模式就不使用HDFS，也不加载任何Hadoop的守护进程。

·所有程序都在同一个JVM上执行

·该模式主要用于开发调试MapReduce程序的应用逻辑。

**伪分布模式**

·Hadoop守护进程运行在本地机器上，模拟一个小规模的集群

·Hadoop的每个守护进程都运行在单独的Java进程中

**全分布模式**

·Hadoop守护进程运行在一个集群上

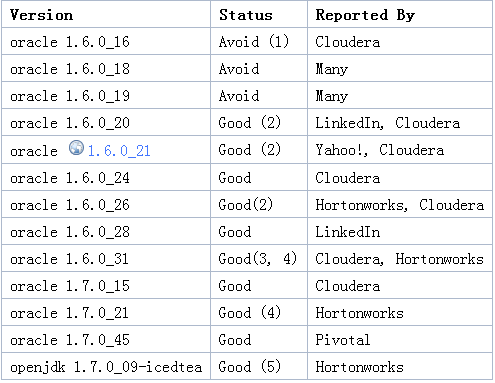
·例如：集群环境中，NameNode和ResourceManager各运行于一台单独的主机，做为master。而其他的主机做为DataNode和NodeManager，做为slave.

**Hadoop与Java版本**

·Hadoop的开发和测试是在OpenJDK和OracleJDK上进行的

·Hadoop支持java7和比较新的java6版本

·Hadoop与java版本的对应: http://wiki.apache.org/hadoop/HadoopJavaVersions



#### 安装YARN前的准备

**需要的软件**

·Ubuntu14.04.02

·Hadoop 2.6 stable

·OpenJDK 1.7

·OpenSSH

**安装JDK及SSH**

·安装OpenJDK 1.7

— apt-get install openjdk-7-jdk

— java -version

·OpenSSH

— apt-get install ssh

— apt-get install rsync

**创建用户和组**

·创建hadoop组

— addgroup hadoop

·创建用户hduser并添加到hadoop组中

— adduser --ingroup hadoop hduser

**配置SSH的无密码登录**

·检查hduser是否可以以无密码的方式登录localhost

— ssh localhost

·如果需要输入密码，进行如下操作

— ssh-keygen -t dsa

— cat ~/.ssh/id\_dsa.pub >> ~/.ssh/authorized\_keys

**解压缩及设置Hadoop环境变量**

·解压缩Hadoop

— gunzip hadoop-2.6.0.tar.gz

— tar –xvf hadoop-2.6.0.tar

·假设解压缩后的路径是在/hadoop/下在解压后的文件中，找到并编辑Hadoop环境变量文件./hadoop-2.6.0/etc/hadoop/hadoop-env.sh

— 设置Java Home: export JAVA\_HOME=/usr/lib/jvm/java-7-openjdk-amd64

— 设置hadoop prefix：HADOOP\_PREFIX=/hadoop/hadoop-2.6.0

**测试Hadoop**

·测试Hadoop:

— $ bin/hadoop

#### 伪分布式安装

**修改core-site.xml文件**

·etc/hadoop/core-site.xml

—fs.defaultFS配置NameNode的URI

|  |
| --- |
| <configuration>  <property>  <name>fs.defaultFS</name>  <value>hdfs://localhost:9000</value>  </property>  </configuration> |

**修改hdfs-site.xml文件**

·etc/hadoop/hdfs-site.xml

— dfs.replication设置块的复制数量

|  |
| --- |
| <configuration>  <property>  <name>dfs.replication</name>  <value>1</value>  </property>  </configuration> |

**修改mapred-site.xml文件**

·etc/hadoop/mapred-site.xml

— mapreduce.framework.name配置MapReduce应用使用Yarn框架

|  |
| --- |
| <configuration>  <property>  <name>mapreduce.framework.name</name>  <value>yarn</value>  </property>  </configuration> |

**修改yarn-site.xml文件**

·etc/hadoop/yarn-site.xml

— yarn.nodemanager.aux-services：为NodeManager配置MapReduce应用的Shuffle服务

|  |
| --- |
| <configuration>  <property>  <name>yarn.nodemanager.aux-services</name>  <value>mapreduce\_shuffle</value>  </property>  </configuration> |

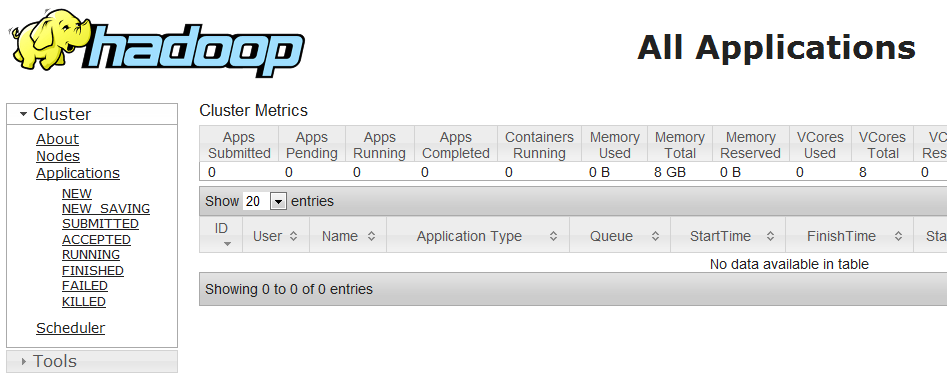
**检查Yarn是否安装成功**

·启动资源管理器(ResourceManager)和节点管理器(NodeManager)守护进

— $ sbin/start-yarn.sh

·访问资源管理器的web接口

— http://localhost:8088/



**小结**

·Hadoop配置文件的作用

·Hadoop的三种安装模式

·Hadoop与Java版本的对应关系

·如何进行伪分布式安装

# Hadoop用户行为分析项目

·Hadoop回顾

·项目整体概述

·项目工程准备

### 第1节应用概述

**Hadoop回顾**

·简介Hadoop的业务、应用场景

·用户行为分析平台搭建注意事项

**业务场景：**

·时延

·吞吐量

**应用场景：**

·MapReduce计算模型

·海量数据的离线分析

·静态数据源

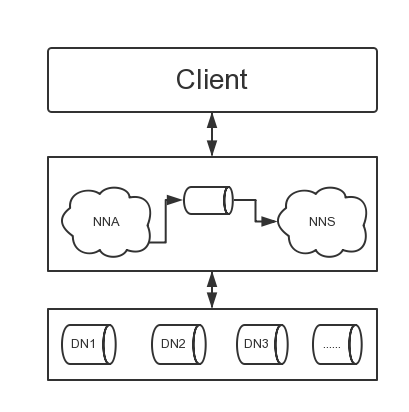
**用户行为分析平台搭建注意事项**

·高可用性

·NNA和NNS节点配置注意事项

·YARN服务配置

·启动顺序

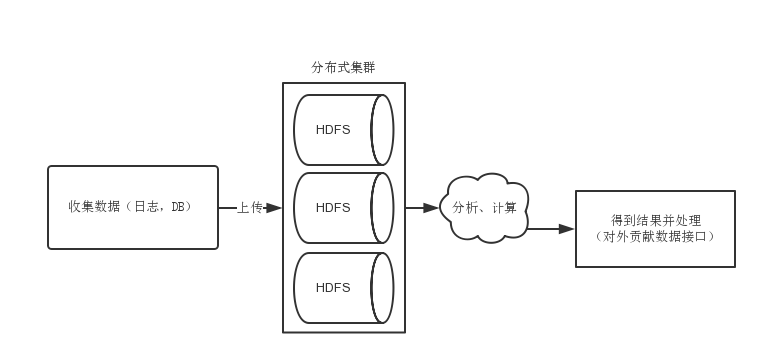


**项目整体概述**

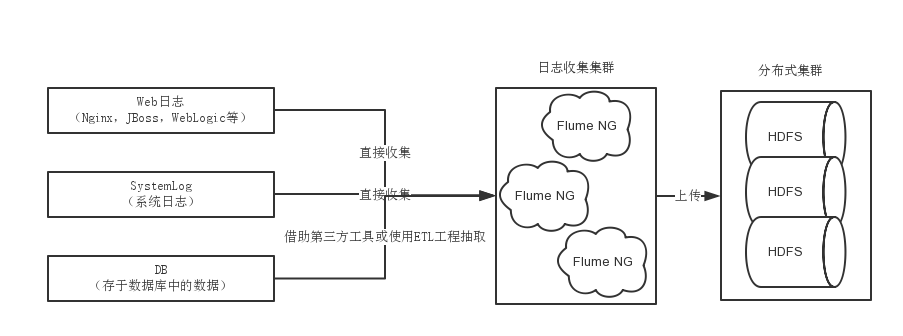
·简述项目整体流程

·介绍数据来源获取方式

**整体流程预览如下图所示：**



**数据源获取方式如下图所示：**



**项目工程准备**

·IDE和Hadoop插件的介绍与使用

·Hadoop工程创建关键步骤

**概述**

·基础软件介绍

·使用技巧

·预览、演示

**项目创建关键步骤包含以下：**

·基础环境

·工程结构选取

·预览、演示

**Hadoop的业务和应用场景**

·项目的基本流程

·开发工具和插件的使用

·Hadoop项目的创建工作