



# 第一讲-2-Hadoop 和HDFS

鄂海红 计算机学院 副教授

[ehaihong@bupt.edu.cn](mailto:ehaihong@bupt.edu.cn) 微信/QQ : 87837001

2017.10.28

## 提纲

- Hadoop生态系统
- Hadoop平台
- HDFS文件存储
- MapReduce模型

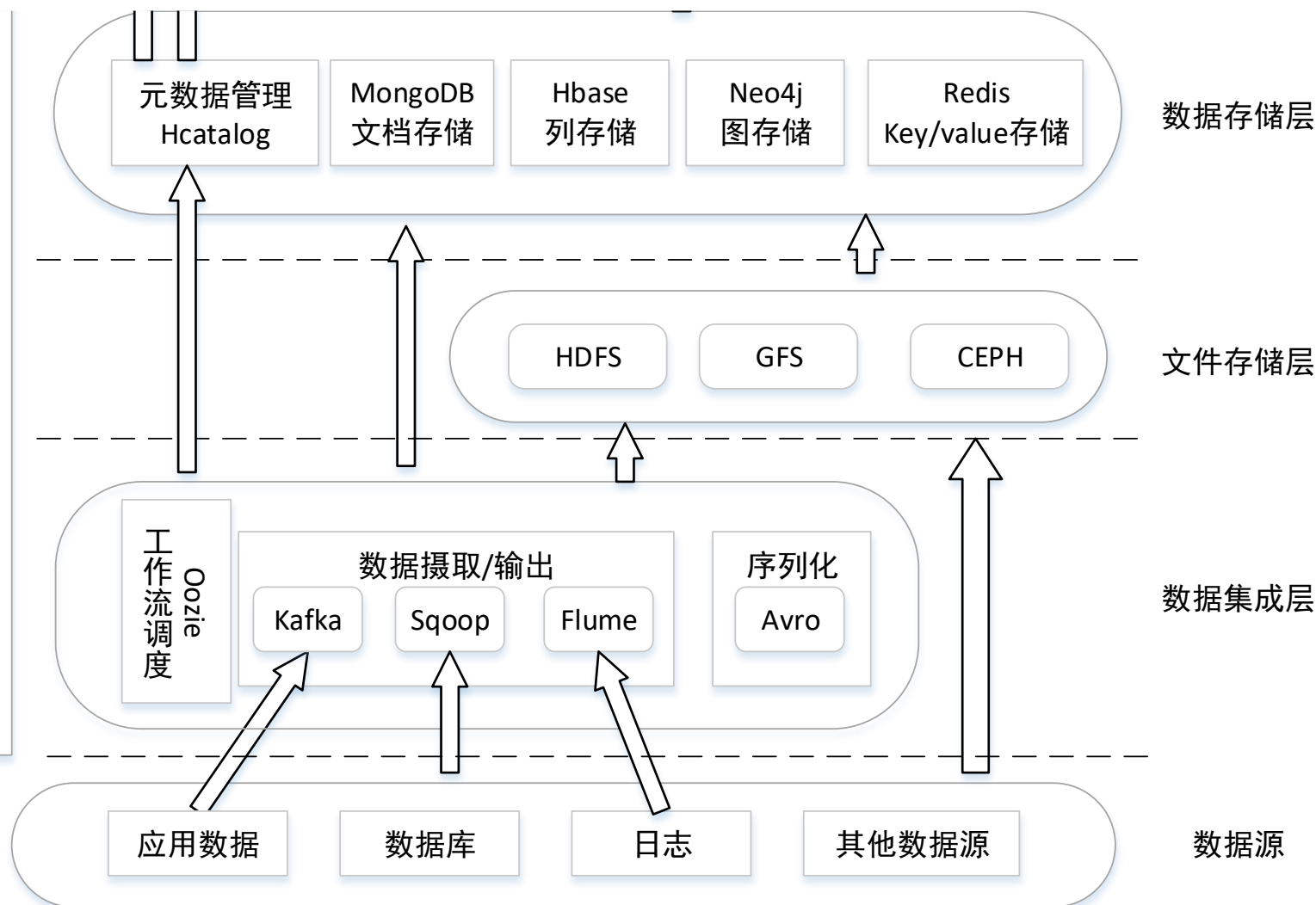
1

# 大数据生态系统



# 大数据生态系统 (1/2)

Amb

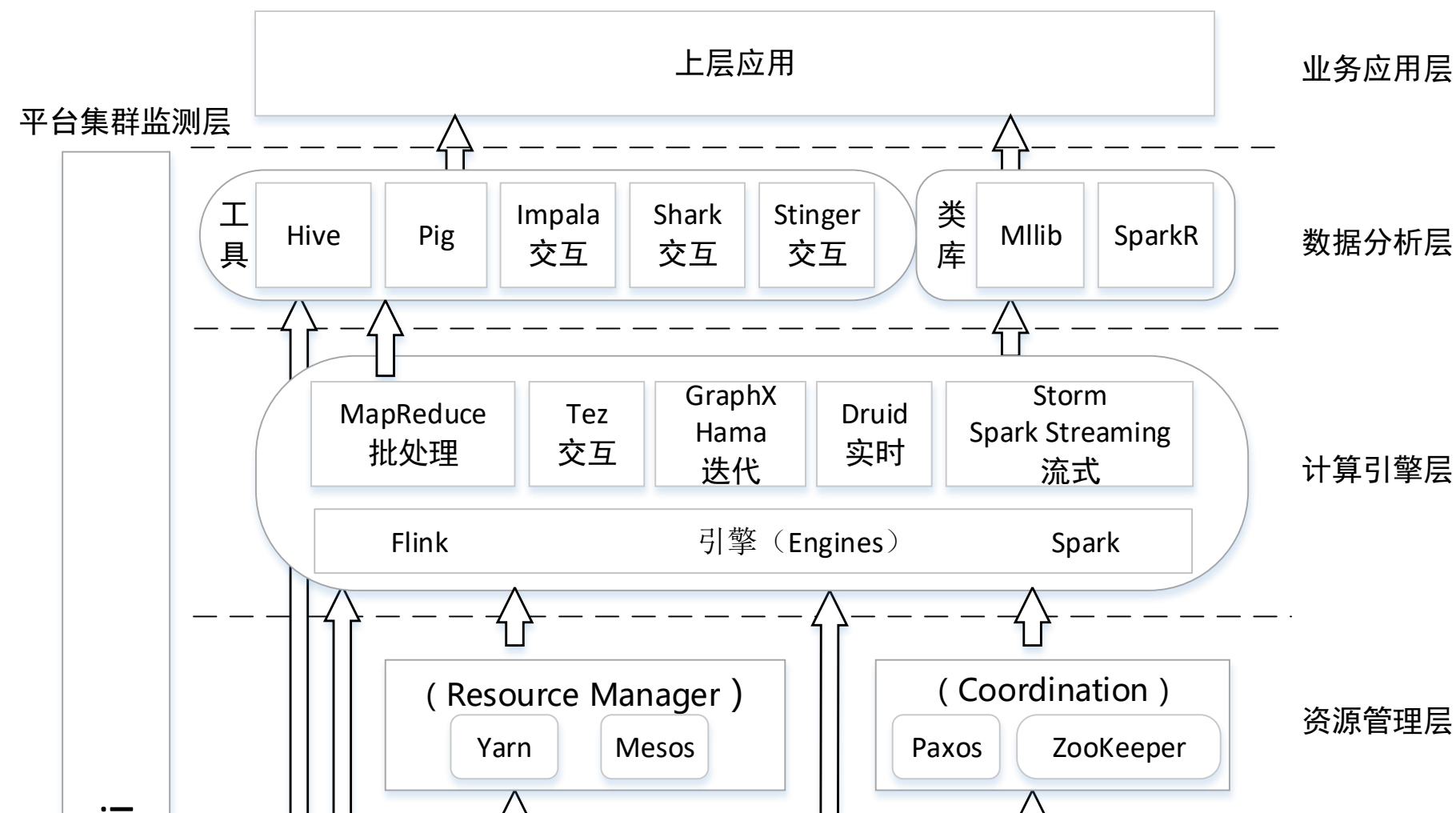


- 数据存储层
  - 提供分布式、可扩展的大量数据表的存储和管理；支持在大规模数据量下完成快速数据读写操作，且可随数据量快速增长通过简单硬件扩展实现存储能力的线性增长
- 文件存储层
  - 利用分布式文件系统技术，通过统一的接口向上层应用提供对象级文件访问能力；为上层应用屏蔽了存储设备类型、型号、接口协议、分布位置等技术细节，并提供了数据备份、故障容忍、状态监测、安全机制等多种保障可靠的文件访问服务的管理性功能
- 数据集成层
  - 协助高效地摄取和输出大数据系统之间的数据；采用元数据框架，实施数据在整个生命周期的管理和治理

# 大数据处理流程-采集、存储、处理

- (1) 大数据采集的框架
- 数据来源：企业自己的生产系统产生的数据，以及系统运维产生的用户行为数据、日志式的活动数据、事件信息等。数据采集Sqoop用于在Hadoop(Hive)与传统的数据库(mysql)间进行数据的传递，可以将一个关系型数据库中的数据导进到Hadoop的HDFS中，也可以将HDFS的数据导进到关系型数据库中。Apache Kafka被设计成能够高效地处理大量实时数据，其特点是快速的、可扩展的、分布式的，分区的和可复制的。
- (2) 大数据存储的框架
- HDFS是Hadoop**分布式文件处理系统**。分布式文件系统在**大规模集群**上运行，集群构建是使用廉价的普通机器，整个文件系统采用的是**元数据集中管理**与**数据块分散存储**相结合的模式，并通过**数据的复制**来实现高度容错，分布式文件处理系统在通用的服务器、操作系统或虚拟机上架构。
- (3) 大数据处理的框架
- MapReduce一个**分布式并行计算**软件框架，基于Map和Reduce的**Java**函数，基于MapReduce写出来的应用程序能够在由上千个普通机器组成的大型集群上运行，并以一种可靠容错的方式并行处理TB级别以上的数据集。可以用多种语言来编写Mapper和Reducer的主代码。

# 大数据生态系统 ( 1/2 )



- 平台管理层
  - 提供了包括配置管理、运行监控、故障管理、性能优化、安全管理等功能
- 数据分析层
  - 提供数据分析工具、算法库，建立数据分析模型，共享（复用）数据分析模型
- 计算引擎层（编程模型层）
  - 为大规模数据处理提供一个抽象的并行计算编程模型，以及为此模型可实施的编程环境和不同种类计算的运行(runtime)环境
- 资源管理层
  - 大数据作业调度管理、集群的资源管理（CPU/内存等）

# Hadoop生态圈



# Hadoop生态的项目结构

组件	功能
HDFS	分布式文件系统
MapReduce	分布式并行编程模型
YARN	资源管理和调度器
Tez	运行在YARN之上的下一代Hadoop查询处理框架
Hive	Hadoop上的数据仓库
HBase	Hadoop上的非关系型的分布式数据库
Pig	一个基于Hadoop的大规模数据分析平台，提供类似SQL的查询语言Pig Latin
Sqoop	用于在Hadoop与传统数据库之间进行数据传递
Oozie	Hadoop上的工作流管理系统
Zookeeper	提供分布式协调一致性服务
Storm	流计算框架
Flume	一个高可用的，高可靠的，分布式的海量日志采集、聚合和传输的系统
Ambari	Hadoop快速部署工具，支持Apache Hadoop集群的供应、管理和监控
Kafka	一种高吞吐量的分布式发布订阅消息系统，可以处理消费者规模的网站中的所有动作流数据
Spark	类MR的通用并行框架，不同的是Job中间输出结果可以保存在内存中，从而不再需要读写HDFS



2

# Hadoop平台





# 什么是Hadoop

Hadoop是什么呢，Hadoop是Apache Software Foundation开发的一个分布式系统基础构架，是对Google的MapReduce核心技术的开源实现，应用JAVA语言进行开发。

Hadoop能够在大规模计算机集群中对海量数据进行分布式计算，它的核心设计为分布式文件系统（Hadoop Distributed File System，HDFS）和分布式计算框架MapReduce两个模块。



# Hadoop产生背景

# Hadoop之前无法解决的问题

单机无法  
计算海量  
数据

3个500G的文件  
中找出重复出现  
或不重复的行

关系型数  
据库无法  
统计大量  
数据

100亿条信息中  
统计TOP10热点  
新闻

Nutch索  
引和存储  
无法实现

Nutch是搜索引  
擎，需要存储海  
量数据，查询需  
要建立大量索引  
以提高速度

数据太大  
性能与效  
率低下

有100T的网站日  
志，计算PV，  
UV，IP



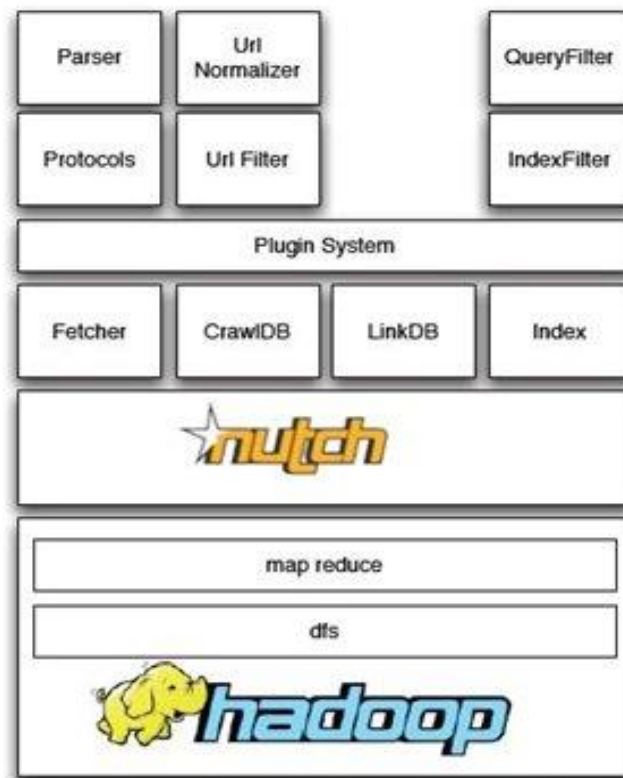
# Hadoop的前身：Nutch搜索引擎

- Hadoop最早起源于Nutch
- Nutch是一个开源的网络搜索引擎，由Doug Cutting于2002年创建

# Nutch

Nutch 是一个开源Java实现的搜索引擎。它提供了我们运行自己的搜索引擎所需的全部工具。包括全文搜索和Web爬虫。

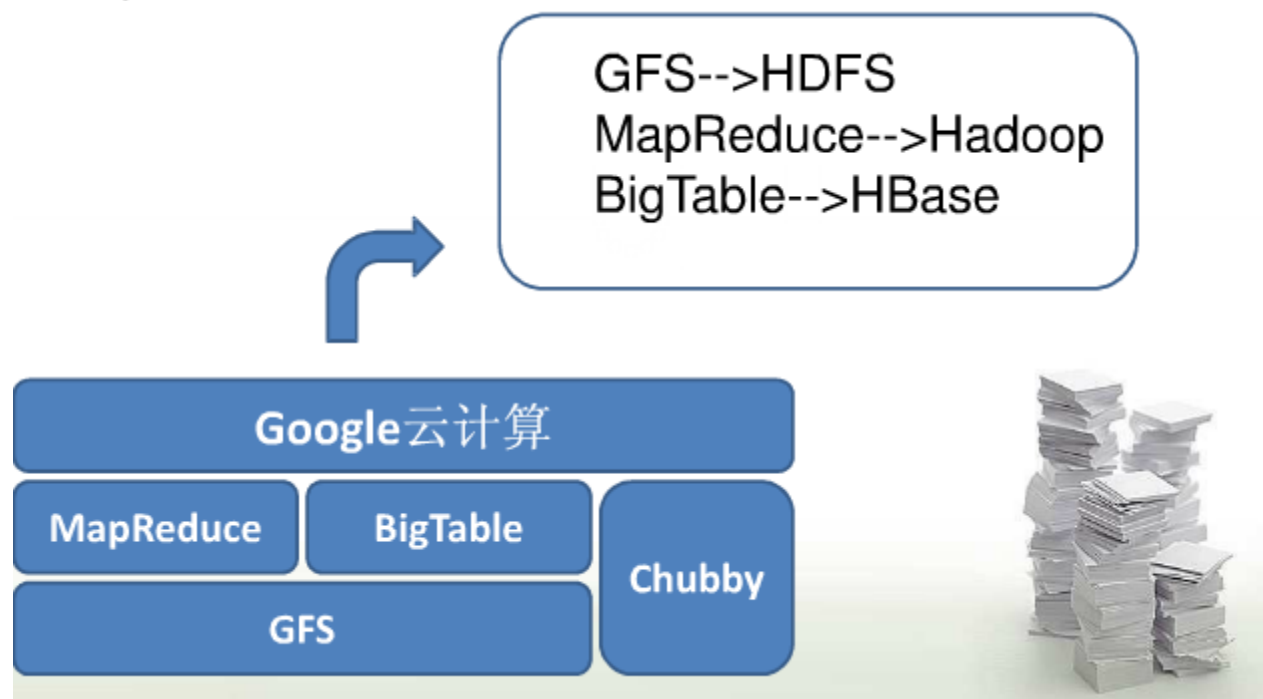
## Architecture



# Nutch存在的问题

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，即不能解决数十亿网页的存储和索引问题。

# Hadoop的由来





# Google论文

- 2003年，Google发布Google File System论文，这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。

Google File System 中文版 1.0 版

## Google File System 中文版<sup>1</sup>

### 摘要

我们设计并实现了 Google GFS 文件系统，一个面向大规模数据密集型应用的、可伸缩的分布式文件系统。GFS 虽然运行在廉价的普遍硬件设备上，但是它依然了提供灾难冗余的能力，为大量客户机提供了高性能的服务。

虽然 GFS 的设计目标与许多传统的分布式文件系统有很多相同之处，但是，我们的设计还是以我们对我们的应用的负载情况和技术环境的分析为基础的，不管现在还是将来，GFS 和早期的分布式文件系统的设想都有明显的不同。所以我们重新审视了传统文件系统在设计上的折衷选择，衍生出了完全不同的设计思路。

GFS 完全满足了对存储的需求。GFS 作为存储平台已经被广泛的部署在 Google 内部，存储我们的服务产生和处理的数据，同时还用于那些需要大规模数据集的研究和开发工作。目前为止，最大的一个集群利用数千台机器的数千个硬盘，提供了数百 TB 的存储空间，同时为数百个客户机服务。

在本论文中，我们展示了能够支持分布式应用的文件系统接口的扩展，讨论我们设计的许多方面，最后列出了小规模性能测试以及真实生产系统中性能相关数据。



- 2004年公布的 MapReduce论文，论文描述了大数据的分布式计算方式，主要思想是将任务分解然后在多台处理能力较弱的计算节点中同时处理，然后将结果合并从而完成大数据处理。

Google MapReduce 中文版 1.0 版

## Google MapReduce 中文版<sup>1</sup>

### 摘要

MapReduce 是一个编程模型，也是一个处理和生成超大数据集的计算模型的相关实现。用户首先创建一个 Map 函数处理一个基于 key/value pair 的数据集合，输出中间的基于 key/value pair 的数据集合；然后再创建一个 Reduce 函数用来合并所有的具有相同中间 key 值的中间 value 值。现实世界中有很多满足上述处理模型例子，本论文将详细描述这个模型。

MapReduce 架构的程序能够在大量的普通配置的计算机上实现并行化处理。这个系统在运行时只关心：如何分割输入数据，在大量计算机组成的集群上的调度，集群中计算机的错误处理，管理集群中计算机之间必要的通信。采用 MapReduce 架构可以使那些没有并行计算和分布式处理系统开发经验的程序员有效利用分布式系统的丰富资源。

我们的 MapReduce 实现运行在规模可以灵活调整的由普通机器组成的集群上：一个典型的 MapReduce 计算往往由几千台机器组成、处理以 TB 计算的数据。程序员发现这个系统非常好用：已经实现了数以百计的 MapReduce 程序，在 Google 的集群上，每天都有 1000 多个 MapReduce 程序在执行。



# Hadoop发展

由于谷歌未开源代码，Nutch的开发人员完成了一个开源实现。由于NDFS和MapReduce不仅适用于搜索领域，2006年年初，开发人员便将其移出Nutch，成为Lucene的一个子项目，称为Hadoop。大约同一时间，Doug Cutting加入雅虎公司，且公司同意组织一个专门的团队继续发展Hadoop。同年2月，Apache Hadoop项目正式启动以支持MapReduce和HDFS的独立发展。

2008年1月，Hadoop成为Apache顶级项目

# Hadoop发展



- 2004年-- 最初的版本(现在称为HDFS和MapReduce)由Doug Cutting和Mike Cafarella 开始实施。
- 2005年12月-- Nutch移植到新的框架，Hadoop在20个节点上稳定运行。
- 2006年1月-- Doug Cutting加入雅虎。
- 2006年2月-- Apache Hadoop项目正式启动以支持MapReduce和HDFS的独立发展。
- 2006年2月-- 雅虎的网格计算团队采用Hadoop。
- 2006年4月-- 标准排序(10 GB的数据进行排序)在188个节点上运行47.9个小时。
- 2006年5月-- 雅虎建立了一个300个节点的Hadoop研究集群。
- 2006年5月-- 标准排序在500个节点上运行42个小时(硬件配置比4月的更好)。
- 2006年11月-- 研究集群增加到600个节点。



# Hadoop发展

- 06年12月-- 标准排序在20个节点上运行1.8个小时，100个节点3.3小时，500个节点5.2小时，900个节点7.8个小时。
- 07年4月-- 研究集群达到两个1000个节点的集群。
- 08年4月-- 赢得世界最快1 TB数据排序在900个节点上用时209秒。
- 09年4月-- 赢得每分钟排序，59秒内排序500 GB(在1400个节点上)和173分钟内排序100 TB数据(在3400个节点 )

# Hadoop的现在



[Top](#) [Wiki](#)

▼ About

▫ Welcome

**Releases**

▫ Download

Release Notes

▫ Release Versioning

▫ Mailing Lists

▫ Issue Tracking

▫ Who We Are?

▫ Who Uses Hadoop?

▫ Buy Stuff

▫ Sponsorship

▫ Thanks

▫ Privacy Policy

▫ Bylaws

▫ Committer criteria

▫ License

▶ Documentation

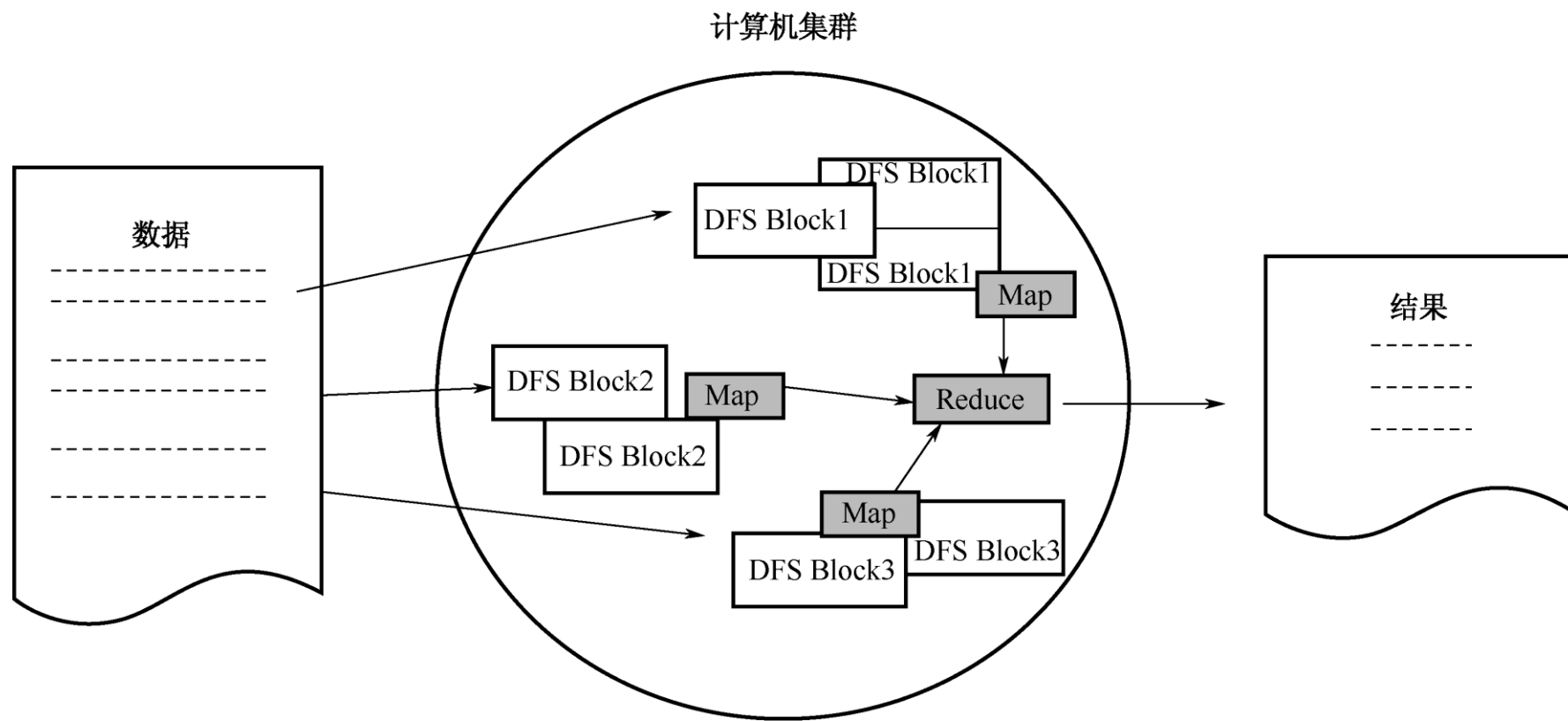
## Apache Hadoop Releases

### Download

Hadoop is released as source code tarballs with corresponding binary tarballs for convenience.

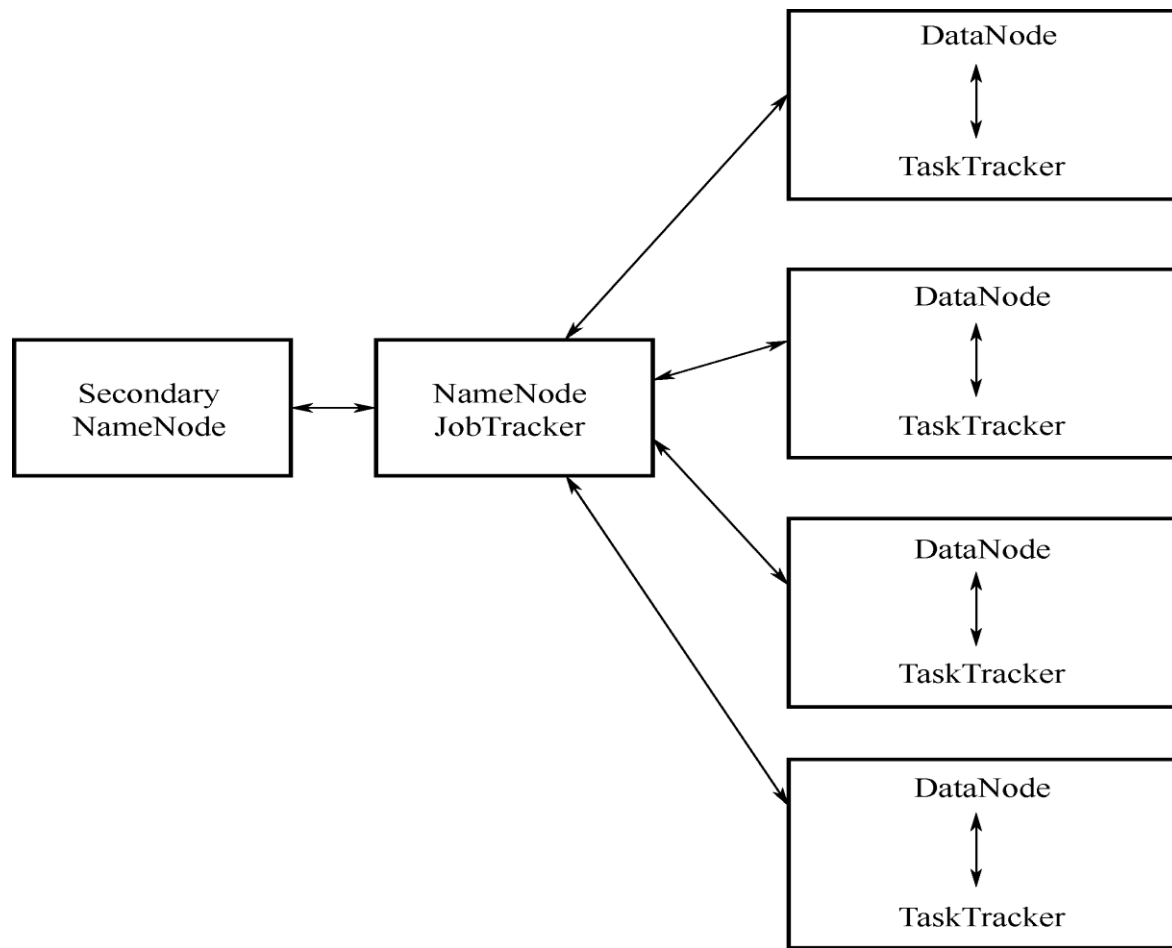
Version	Release Date
<a href="#">3.0.0-beta1</a>	03 October, 2017
<a href="#">2.8.1</a>	08 June, 2017
<a href="#">2.7.4</a>	04 August, 2017
<a href="#">2.6.5</a>	08 October, 2016

数据在Hadoop中处理的流程，可以简单表示为下图：



Hadoop中的数据处理过程

在分布式存储和计算方面，Hadoop采用的是主/从(Master/Slave)架构，集群主要由 NameNode，DataNode，Secondary NameNode，JobTracker，TaskTracker组成：



Hadoop集群



# Hadoop的优势



1. 高容错性。这主要表现在Hadoop可以自动的重新分配失败的作业，还可以自动把数据的多个副本保存。

2. 高扩展性。这是Hadoop的一个重要特点，主要是由于Hadoop完成作业是在可用的计算机集的簇间分配数据，而这些集簇能够扩展到海量的节点中。

3. 高可靠性。Hadoop按位存储和处理数据的效果满足人们的需求，值得人们信赖。

4. 高效性。Hadoop的处理速度是很快的，这是由于Hadoop是在节点间动态的移动数据并使每个节点达到动态平衡。

5. 低成本。Hadoop是开源的，其软件成本不高，并且对硬件的要求也不高，所以它的成本比较低。

3

# HDFS-Hadoop文件存储系统

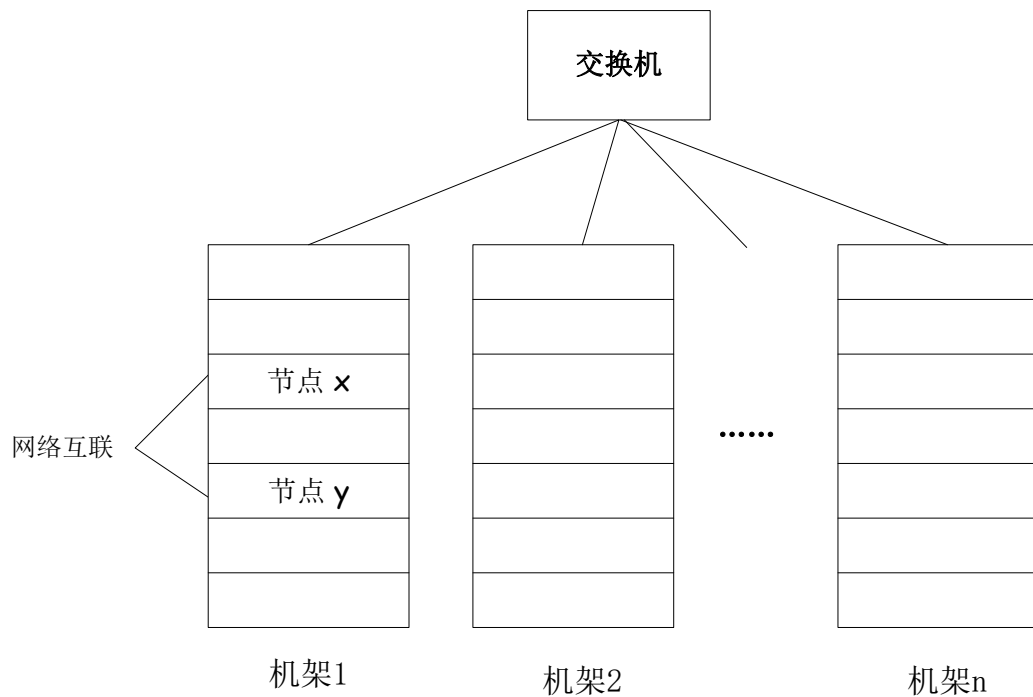


# HDFS

Hadoop主要由HDFS和MapReduce两部分组成，最底部是HDFS。HDFS是Hadoop distributed File System的简称，也就是分布式文件管理系统。HDFS与现在已有的分布式文件系统相似，能够运行在通用的机器设备上，但是也存在一些不同。HDFS以流的方式访问文件数据，用高吞吐量进行应用程序的数据访问，这使得它能够应对海量数据集的应用程序。

# 计算机集群结构

- ✓ 分布式文件系统把文件分布存储到多个计算机节点上，成千上万的计算机节点构成计算机集群
- ✓ 与之前使用多个处理器和专用高级硬件的并行化处理装置不同的是，目前的分布式文件系统所采用的计算机集群，都是由普通硬件构成的，这就大大降低了硬件上的开销



计算机集群的基本架构

# 分布式文件系统的结构

分布式文件系统在物理结构上是由计算机集群中的多个节点构成的，这些节点分为两类，一类叫“主节点” (Master Node) 或者也被称为“名称结点” (NameNode)，另一类叫“从节点” (Slave Node) 或者也被称为“数据节点” (DataNode)

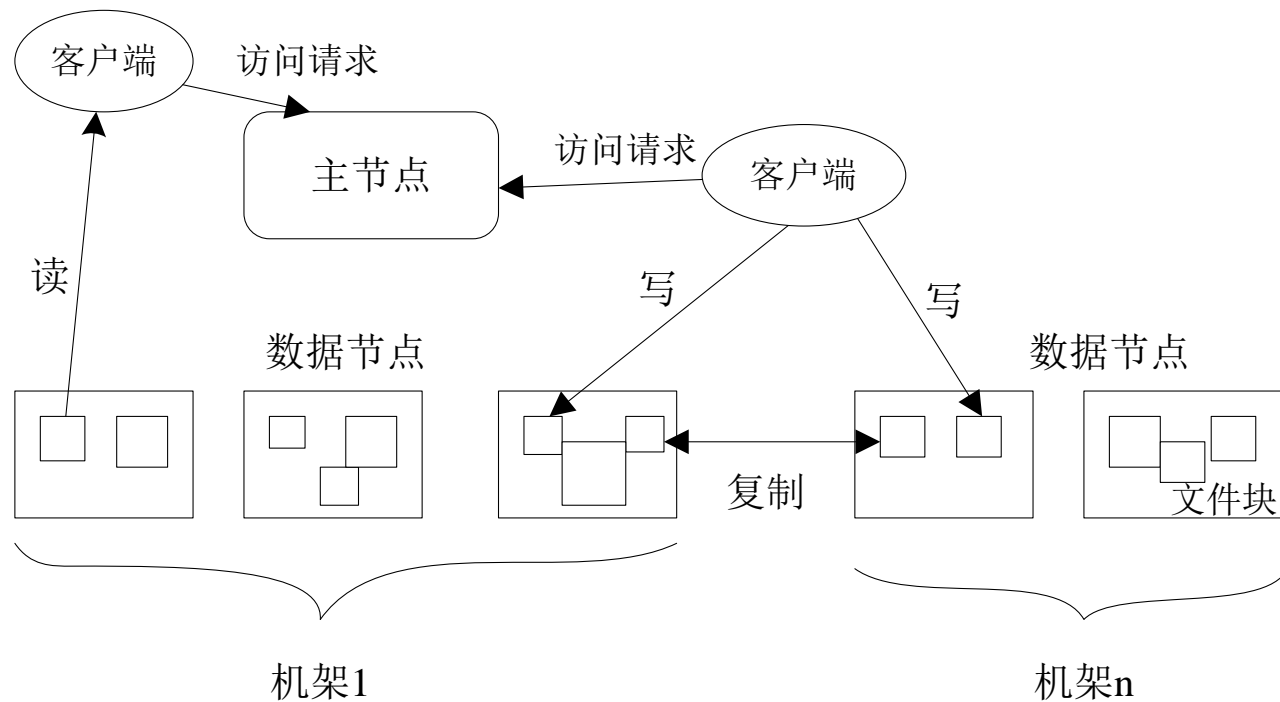


图3-2 大规模文件系统的整体结构



# 分布式文件系统的设计需求

分布式文件系统的设计目标主要包括透明性、并发控制、可伸缩性、容错以及安全需求等。

## ✓ 透明性

含义：具备访问透明性、位置透明性和伸缩透明性

HDFS实现情况：只能提供一定程度的访问透明性，完全支持位置透明性和伸缩透明性

## ✓ 并发控制

含义：客户端对于文件的读写不应该影响其他客户端对同一个文件的读写

HDFS实现情况：机制非常简单，任何时间都只允许有一个程序在写入某个文件

# 分布式文件系统的设计需求

- ✓ 文件复制

含义：一个文件可以拥有在不同位置的多个副本

HDFS实现情况：HDFS采用了多副本机制

- ✓ 硬件和操作系统的异构性

含义：可以在不同的操作系统和计算机上实现同样的客户端和服务端程序

HDFS实现情况：采用Java语言开发，具有很好的跨平台能力



# 分布式文件系统的设计需求

## ✓ 容错

含义：保证文件服务在客户端或者服务端出现问题的时候能正常使用

HDFS实现情况：具有多副本机制和故障自动检测、恢复机制

## ✓ 可伸缩性

含义：支持节点的动态加入或退出

HDFS实现情况：建立在大规模廉价机器上的分布式文件系统集群，具有很好的可伸缩性

## ✓ 安全

含义：保障系统的安全性

HDFS实现情况：安全性较弱





# HDFS简介

总体而言，HDFS要实现以下目标：

- ✓ 兼容廉价的硬件设备
- ✓ 流数据读写
- ✓ 大数据集
- ✓ 简单的文件模型
- ✓ 强大的跨平台兼容性

HDFS特殊的设计，在实现上述优良特性的同时，也使得自身具有一些应用局限性，主要包括以下几个方面：

- ✓ 不适合低延迟数据访问
- ✓ 无法高效存储大量小文件
- ✓ 不支持多用户写入及任意修改文件

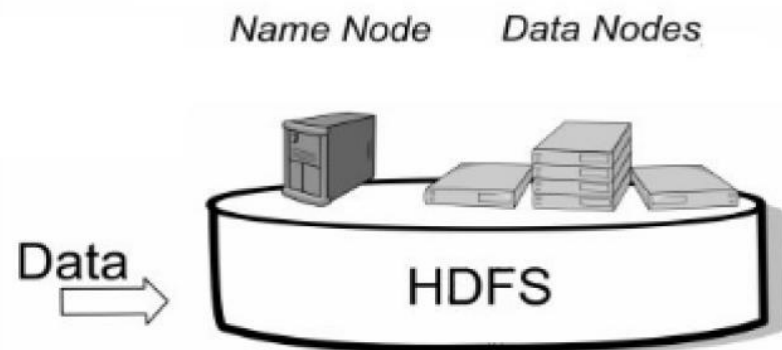
# 块

HDFS采用抽象的块概念可以带来以下几个明显的好处：

- ✓ **支持大规模文件存储**：文件以块为单位进行存储，一个大规模文件可以被分成若干个文件块，不同的文件块可以被分发到不同的节点上，因此，一个文件的大小不会受到单个节点的存储容量的限制，可以远远大于网络中任意节点的存储容量
- ✓ **简化系统设计**：首先，大大简化了存储管理，因为文件块大小是固定的，这样就可以很容易计算出一个节点可以存储多少文件块；其次，方便了元数据的管理，元数据不需要和文件块一起存储，可以由其他系统负责管理元数据
- ✓ **适合数据备份**：每个文件块都可以冗余存储到多个节点上，大大提高了系统的容错性和可用性

# 名称节点和数据节点

## HDFS主要组件的功能



### metadata

File.txt=  
Blk A:  
DN1, DN5, DN6  
  
Blk B:  
DN7, DN1, DN2  
  
Blk C:  
DN5, DN8, DN9

NameNode	DataNode
• 存储元数据	• 存储文件内容
• 元数据保存在内存中	• 文件内容保存在磁盘
• 保存文件,block , datanode 之间的映射关系	• 维护了block id到datanode本地文件的映射关系




# HDFS的设计目标

HDFS可以运行在低廉的设备组成的集群上，并且是通过流式数据访问的方式进行超大文件的存储。下面对一些相关概念进行介绍。

- (1) 超大文件
- (2) 流式数据访问
- (3) 商用硬件
- (4) 集群规模动态扩展
- (5) 移动计算而不移动数据

HDFS不适用的场景有：

1. 低延迟的数据访问。
2. 存在的小文件较多的文件系统中。



# HDFS文件系统的原型GFS

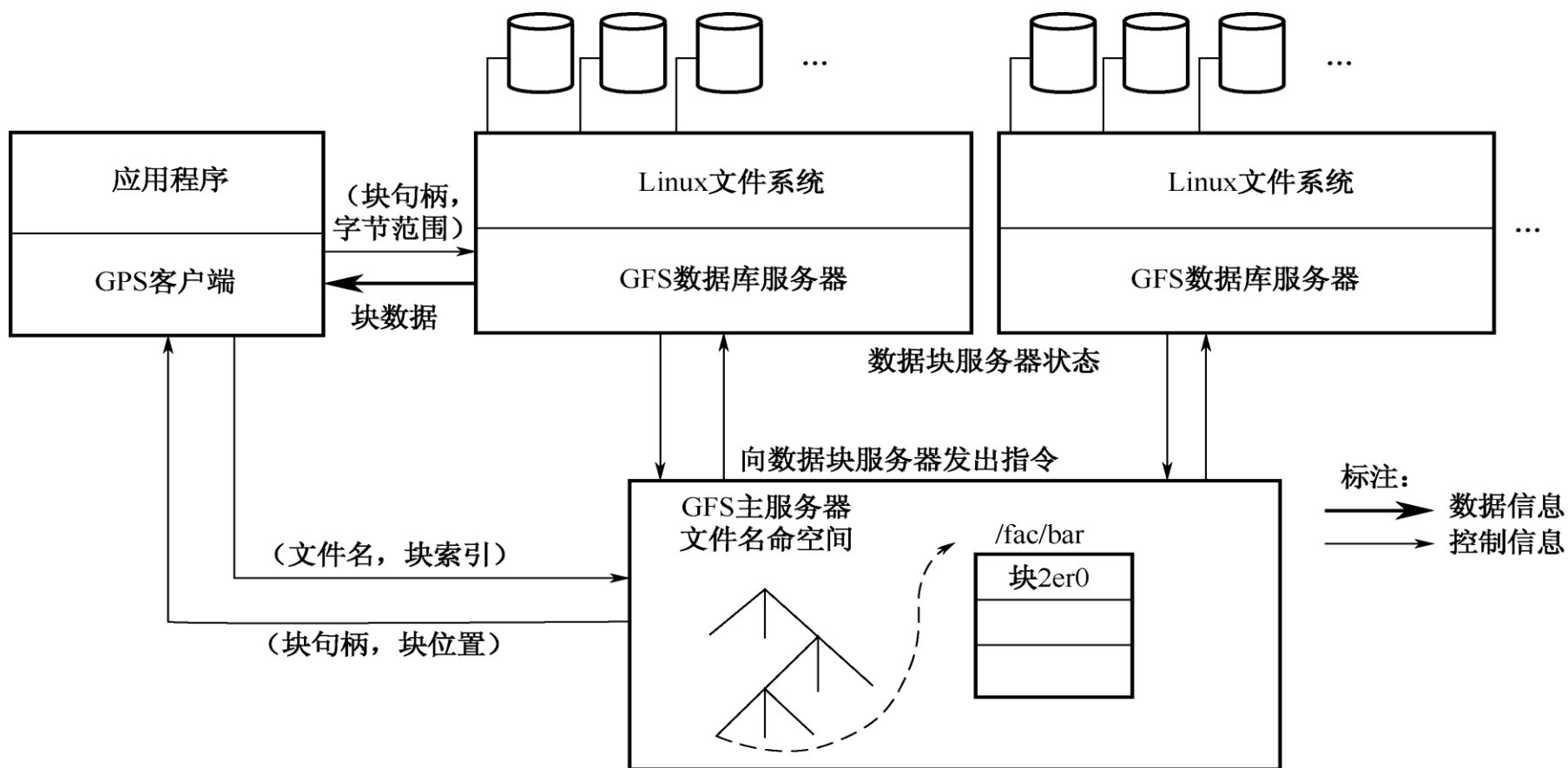
GFS被用来访问海量数据的大型分布式应用，是一个可扩展的分布式文件系统。

GFS把服务器故障视为正常的现象，在保证其系统的可用性和可靠性的情况下使用软件的方式进行自动容错，这就使得系统的成本在很大程度上得以降低。

GFS是Google云存储的基础，存在的**其他存储系统都是直接或者间接的构建在它的基础上**，比如Google Bigtable，Google Megastore，Google Percolator，GFS与Chubby、MapReduce以及Bigtable等技术联系很紧密，为核心技术的底层。

# GFS的系统架构

在整个系统中节点有三种类型分别为客户端（Client）、主服务器（Master）和数据块服务器（Chunk Server）。



GFS的系统架构



# HDFS文件的基本结构

- ✓ HDFS采用主从（ master/slave ）体系结构，由于分布式存储的性质，在HDFS集群由一个NameNode和若干DataNode构成。
  - ✓ NameNode（ 名称节点 ）是用来管理文件系统的命名空间，又叫元数据节点。
  - ✓ DataNode（ 数据节点 ）是用来存储实际的数据。
- ✓ 这两类节点分别承担Master和Worker具体任务的执行节点。



## HDFS的架构图如图所示

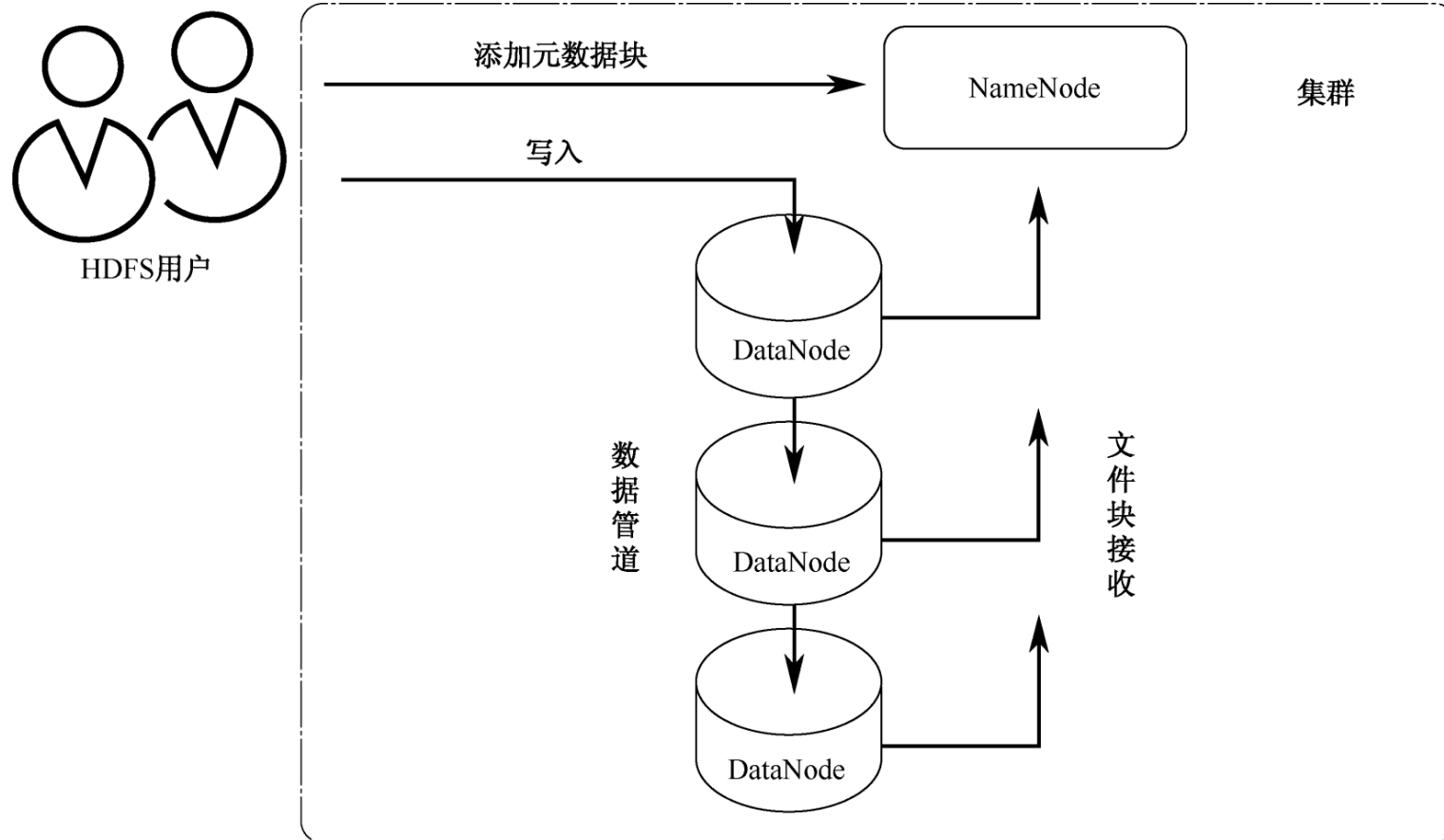


图 HDFS的架构图

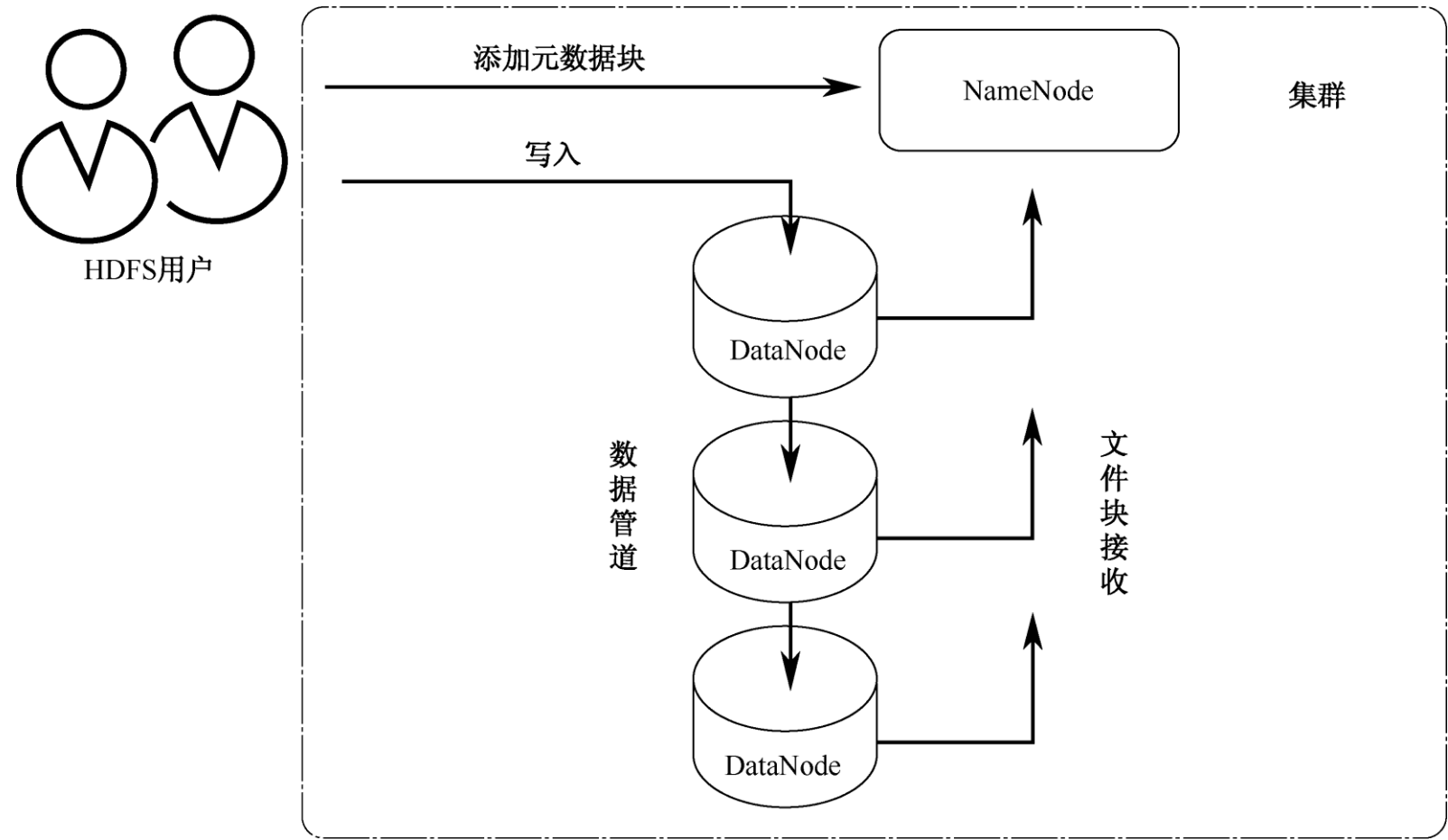
NameNode：主服务器

**作用是：**对整个文件系统的命名空间进行管理，对外界的文件访问请求进行处理。将所有文件和文件夹的元数据保存在一个文件系统树中，NameNode也负责向DataNode分配数据块并建立数据块和DataNode的对应关系。






# HDFS的架构图



HDFS的架构图

DataNode是文件系统的工作节点，同时也是文件存储的基本单元。

DataNode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。

- 
1. HDFS通常的部署是NameNode程序单独运行于一台服务器节点上（主服务器），其余的服务器节点每台运行一个DataNode程序。
  2. HDFS在进行基本文件访问时，首先由用户的应用程序经过HDFS的客户端将文件发送到NameNode。
  3. HDFS所有的通讯协议都是基于TCP/IP协议。
  4. HDFS为了提高NameNode的可靠性，引入了Secondary NameNode节点，辅助NameNode对映像文件和事物日志进行处理。



# HDFS的文件读写操作

操作系统中都存在文件块，什么是文件块呢？简单来说就是文件以块的形式存在磁盘中而块的大小是在进行系统设计时的一个关键的参数。

一般的文件块都要比HDFS中的数据块小，在HDFS中的数据块默认值为64M，是可以修改设定的。

与一般的文件系统不同，在HDFS中如果数据块比文件的尺寸大，不会占用数据块整个的空间。

1. **HDFS的文件的读取**，对于文件的读取最主要的是进行三步，Client向NameNode 发起文件读取的请求，NameNode 返回文件存储的DataNode 的信息，Client读取文件信息。

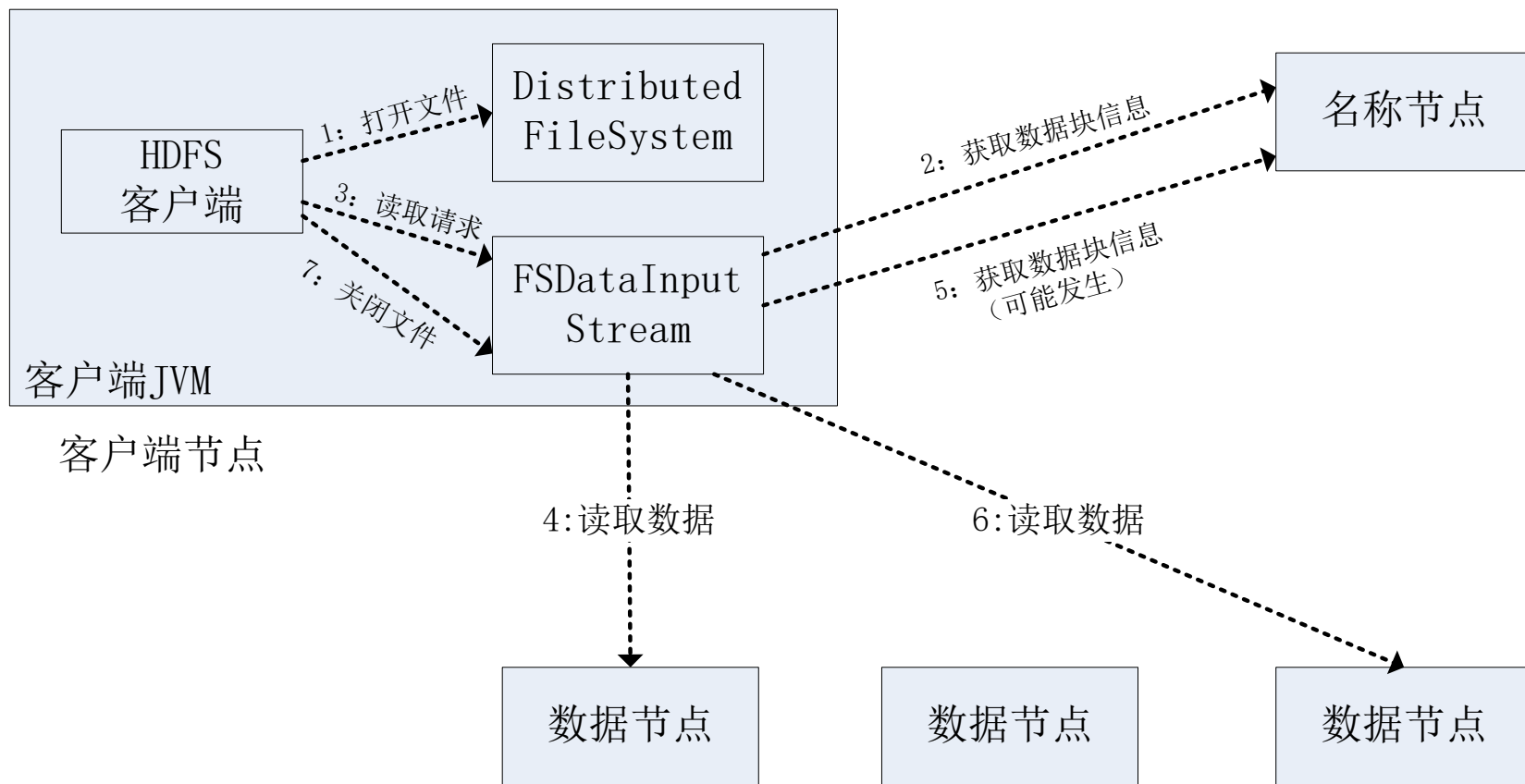
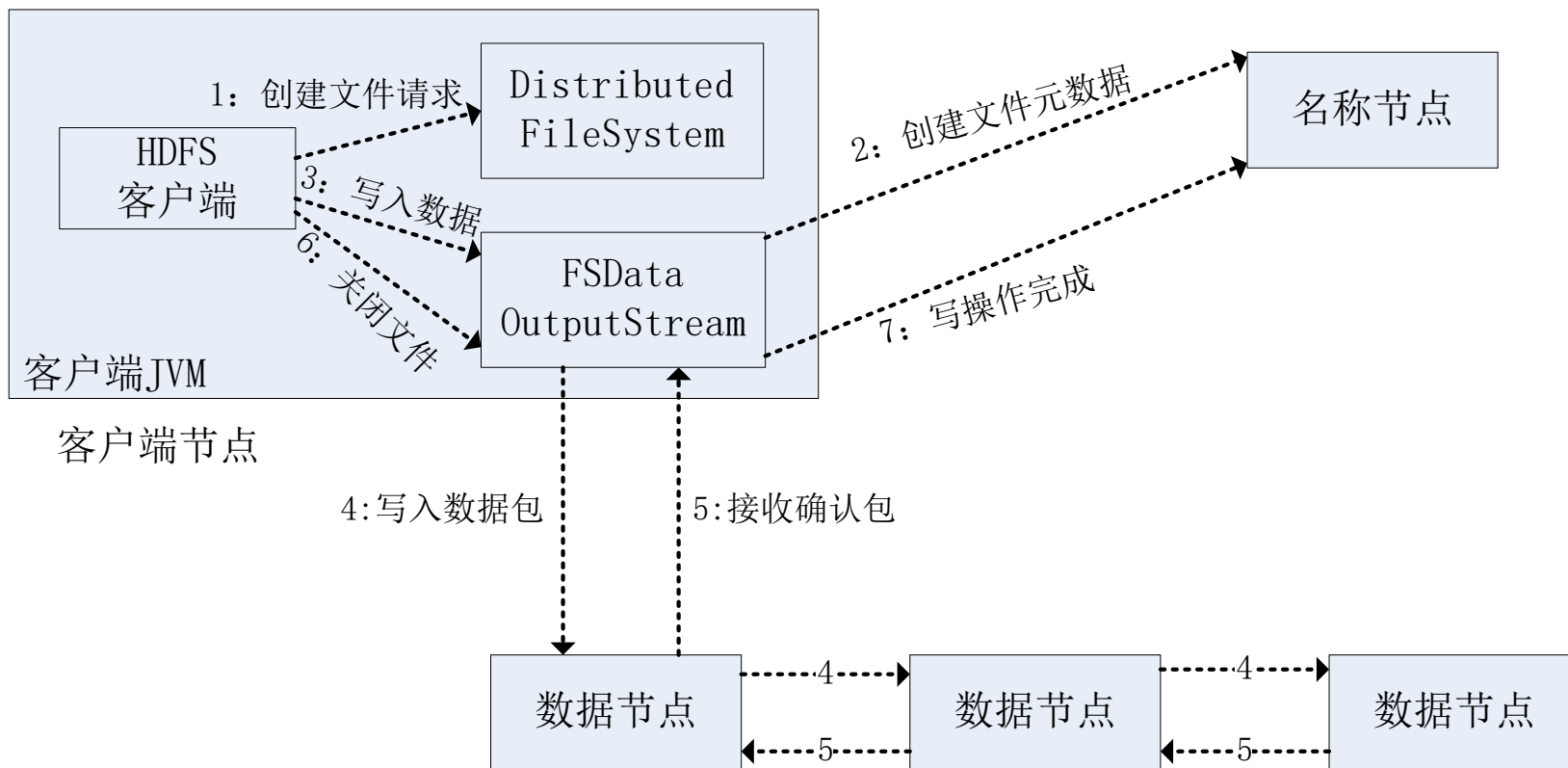


图 文件读取过程

2. HDFS中对文件的写入主要的流程是**创建、写数据、写文件包、完成**四个步骤，也可以简化为核心的三步：Client向NameNode发起文件写入的请求，NameNode根据文件大小和文件块配置情况，返回给Client它所管理部分DataNode的信息，Client将文件划分为多个block，根据DataNode的地址信息，按顺序写入到每一个DataNode块中。具体的流程介绍：



客户端在HDFS中写入数据具体流程介绍：

- (1). 创建文件
- (2). 验证创建文件权限
- (3). 写入数据
- (4). 写入文件包并返回确定信息

客户端在HDFS中写入数据



# HDFS的存储过程

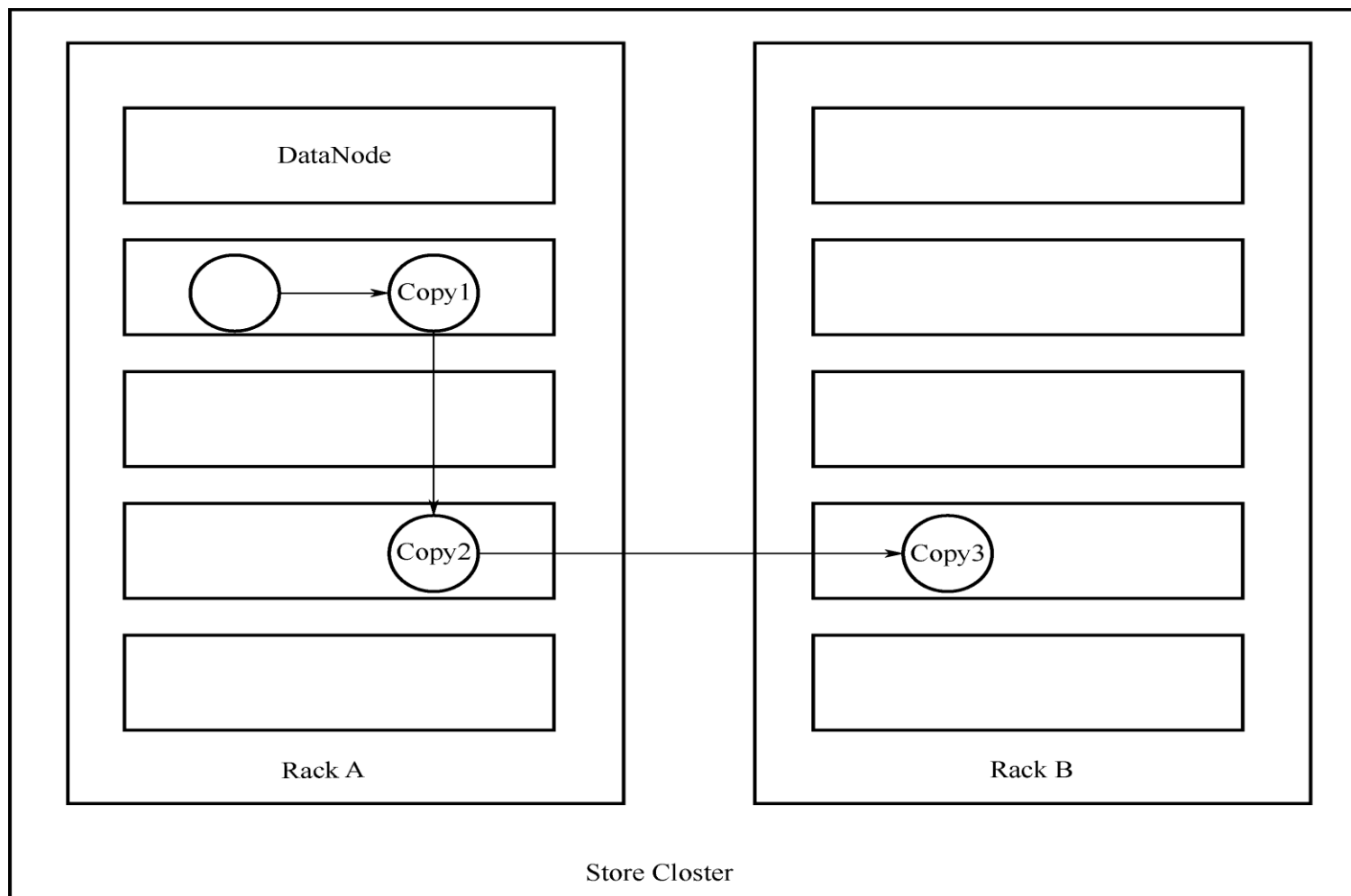
在HDFS中进行数据存储主要涉及到数据的复制，容错等性质，下面对其性质以及存储的优缺点进行了详细介绍。

## 1. 数据复制

### (1) 数据的副本放置

目前HDFS的数据放置策略：数据块的第一个副本先放置在本地机架上的一个节点上，然后第二个副本放置在同一个机架的另一节点上，之后第三个副本放在另外的机架上的节点上。

副本系数为 3 时的数据放置策略：



HDFS默认数据放置策略




## (2) 副本选择

HDFS会优先让客户端对与它最近的数据副本进行读取，这样可以把整体的读取延迟和带宽消耗降低。

如果HDFS集群是跨越了多个的数据中心，那么它会优先选择本地数据中心的副本；如果客户端所在的本地机架上存在数据副本，那么它将优先选择本地副本。





## 2. HDFS的容错机制

在HDFS中比较常见的故障有两种，分别是节点故障和数据损坏。

### (1) 节点故障

通常情况下，节点故障时发生在数据节点上的，发生原因有两点，可能是数据节点本身的故障，也可能是网络割裂。

### (2) 数据损坏

这里的数据损坏是指在数据节点读取数据时数据时损坏的，数据节点存储设备，软件bug，网络错误都可能会导致数据损坏。

### 3. HDFS的平衡

所谓的平衡其实是指系统中的文件块能够在集群中的每个节点得到较好的分配。文件系统的平衡在进行大规模的数据复制时，是必须要考虑的关键因素。

为了保证集群能够保持这个平衡，HDFS提供了balancer工具。这个工具的作用是用来调整文件块存储的平衡，它会重新配置文件块的位置，使得集群能够相对平衡。

所谓的相对平衡是说让每个 DataNode的磁盘使用率和整个集群的资源使用率之间的差值比指定的阈值要小。



## 4. HDFS的优缺点

### ( 1 ) HDFS的优点

处理超大文件

流式的访问数据

运行于廉价的商用机器集群上

### ( 2 ) HDFS的缺点

低延迟数据访问受限

无法高效存储大量小文件

不支持多用户写入及任意修改文件



## 5. HDFS常用命令

HDFS有很多shell命令，其中，fs命令可以说是HDFS最常用的命令。利用该命令可以查看HDFS文件系统的目录结构、上传和下载数据、创建文件等。该命令的用法为：

```
hadoop fs [genericOptions] [commandOptions]
```

备注：Hadoop中有三种Shell命令方式：

hadoop fs适用于任何不同的文件系统，比如本地文件系统和HDFS文件系统

hadoop dfs只能适用于HDFS文件系统

hdfs dfs跟hadoop dfs的命令作用一样，也只能适用于HDFS文件系统

## 5. HDFS常用命令

实例：

`hadoop fs -ls <path>`:显示<path>指定的文件的详细信息

`hadoop fs -mkdir <path>`:创建<path>指定的文件夹

```
administrator@ubuntu:~/hadoop/hadoop-1.2.1/bin$ ./hadoop fs -mkdir hdfs://127.0.0.1:9000/tempDir

administrator@ubuntu:~/hadoop/hadoop-1.2.1/bin$ ./hadoop fs -ls hdfs://127.0.0.1:9000/
Found 4 items
drwxr-xr-x   - administrator supergroup          0 2015-04-26 16:30 /hbase
drwxr-xr-x   - administrator supergroup          0 2015-04-26 15:44 /home
drwxr-xr-x   - administrator supergroup          0 2015-04-26 16:46 /tempDir
drwxr-xr-x   - administrator supergroup          0 2015-04-26 15:55 /user
```

## 5. HDFS常用命令

实例：

`hadoop fs -cat <path>`:将<path>指定的文件的内容输出到标准输出（ stdout ）

`hadoop fs -copyFromLocal <localsrc> <dst>`:将本地源文件<localsrc>复制到路径<dst>指定的文件或文件夹中

```
administrator@ubuntu:~/hadoop/hadoop-1.2.1/bin$ ./hadoop fs -copyFromLocal /home/administrator/tempfile/* hdfs://127.0.0.1:9000/tempDir
administrator@ubuntu:~/hadoop/hadoop-1.2.1/bin$ ./hadoop fs -ls hdfs://127.0.0.1:9000/tempDir/
Found 8 items
-rw-r--r-- 1 administrator supergroup 18 2015-04-26 16:48 /tempDir/file1.txt
-rw-r--r-- 1 administrator supergroup 14 2015-04-26 16:48 /tempDir/file1.txt~
-rw-r--r-- 1 administrator supergroup 18 2015-04-26 16:48 /tempDir/file2.txt
-rw-r--r-- 1 administrator supergroup 18 2015-04-26 16:48 /tempDir/file3.txt
-rw-r--r-- 1 administrator supergroup 18 2015-04-26 16:48 /tempDir/file4.abc
-rw-r--r-- 1 administrator supergroup 18 2015-04-26 16:48 /tempDir/file5.abc
-rw-r--r-- 1 administrator supergroup 17 2015-04-26 16:48 /tempDir/testFile
-rw-r--r-- 1 administrator supergroup 0 2015-04-26 16:48 /tempDir/testFile~
administrator@ubuntu:~/hadoop/hadoop-1.2.1/bin$ ./hadoop fs -cat hdfs://127.0.0.1:9000/tempDir/*
this is file1.txt
this is file1
this is file2.txt
this is file3.txt
this is file4.abc
this is file5.abc
welcome to DBLab
```

## 6. HDFS的Web界面

在配置好Hadoop集群之后，可以通过浏览器登录  
“http://[NameNode/P]:50070”访问HDFS文件系统

localhost:50070/dfshealth.jsp

### NameNode 'localhost:8020'

Started: Thu Jan 15 15:30:34 CST 2015  
Version: 1.2.1, r1503152  
Compiled: Mon Jul 22 15:23:09 PDT 2013 by mattf  
Upgrades: There are no upgrades in progress.

[Browse the filesystem](#)  
[Namenode Logs](#)

#### Cluster Summary

Safe mode is ON. The reported blocks 7 has reached the threshold 0.9990 of total blocks 7. Safe mode will be turned off automatically in 29 seconds.  
15 files and directories, 7 blocks = 22 total. Heap Size is 61.5 MB / 889 MB (6%)

Configured Capacity	:	17.27 GB
DFS Used	:	88 KB
Non DFS Used	:	7.79 GB
DFS Remaining	:	9.48 GB
DFS Used%	:	0 %
DFS Remaining%	:	54.9 %
<a href="#">Live Nodes</a>	:	1
<a href="#">Dead Nodes</a>	:	0
<a href="#">Decommissioning Nodes</a>	:	0
Number of Under-Replicated Blocks	:	0

#### NameNode Storage:

Storage Directory	Type	State
/home/administrator/hadoop_temp/dfs/name	IMAGE_AND_EDITS	Active

localhost:50075/browseBlock.jsp?blockId=4572935344242694132&blockSize=18&genstamp=102

File: [/home/administrator/tempfile/file1.txt](#)

Goto :

[Go back to dir listing](#)  
[Advanced view/download options](#)

this is file1.txt





# 谢谢聆听 批评指正

[ehaihong@bupt.edu.cn](mailto:ehaihong@bupt.edu.cn)



北京邮电大学