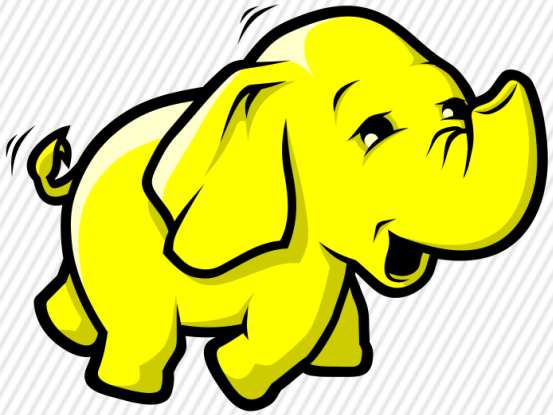
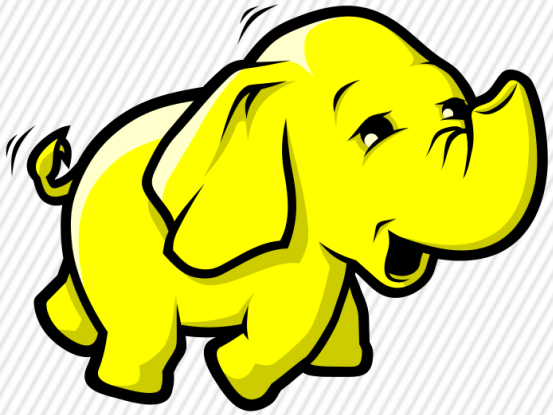


Hadoop 兴趣小组学习分享

Hadoop平台简介　新的大数据技术　学期工作汇报

组员：罗登、宋文宇、夏迎祺、张松鸣、杨世雄、周雅婷、李航、蔡秉歧、金任任
汇报人：罗登



Hadoop生态及其重要组件介绍

基于内存的计算框架和流式计算简介

小组学期工作汇报

先聊聊大数据

话说我们生活在大数据时代，数据量的激增，给我们带来了许多的机遇和挑战。

- 纽约证券交易所每天产生大约1tb的新交易数据
- Facebook拥有大约100亿张照片，占用了1pb的存储空间
- 家谱网站Ancestry.com存储了大约2.5 pb的数据
- 互联网档案存储大约2 pb字节的数据，并且以每月20 tb的速度增长
- 瑞士日内瓦附近的大型强子对撞机每年将产生约15千兆字节的数据。

——数据取自《Oreilly.Hadoop.The.Definitive.Guide.3rd.Edition》

TERABYTE	10 的 12 次方	一块 1TB 硬盘		200,000 照片或 mp3 歌曲
PETABYTE	10 的 15 次方	两个数据中心机柜		16 个 Blackblaze pod 存储单元
EXABYTE	10 的 18 次方	2,000 个机柜		占据一个街区的 4 层数据中心
ZETTABYTE	10 的 21 次方	1000 个数据中心		纽约曼哈顿的 1/5 区域
YOTTABYTE	10 的 24 次方	一百万个数据中心		特拉华州和罗德岛州

大数据的特征

- Volume：体量大
- Variety：多样化
- Velocity：变化快
- Veracity：真实性
- Value：有价值

所以，数据变多真的好吗？

几个定义

Hadoop HDFS是一个分布式文件系统

Hadoop MapReduce是一个并行计算框架

Hadoop YARN是一个资源调度框架

Hadoop是一个完整的生态系统

一些历史

Hadoop是Apache软件基金会旗下的一个开源分布式计算平台，由Java语言编写。

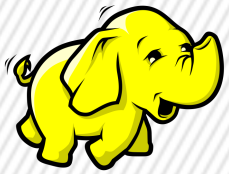
是一个为用户提供系统底层细节透明的分布式基础架构。

关于名字：Hadoop是其作者Doug Cutting儿子小时候的玩具小象的名字

- Hadoop源自始于2002年的Apache Nutch项目——一个开源的网络搜索引擎
- 2005年，Nutch开源实现了谷歌的MapReduce（著名的三篇论文之一）
- 2008年1月，Hadoop正式成为Apache顶级项目
- 2008年4月，Hadoop打破世界纪录，成为最快排序1TB数据的系统，用910个节点组成的集群排序耗时209秒
- 在2009年5月，Hadoop更是把1TB数据排序时间缩短到62秒
- 自此名声大震的Hadoop逐渐发展成为了大数据领域最具影响力的开源分布式平台，并成为了事实上的大数据处理标准

Hadoop的特性

- 高可靠性
- 高效性（至少当时而言）
- 高可扩展性（大数据需求）
- 高容错性（不怕宕机）
- 开源免费！！！
- 强大的社区支持



Hadoop HDFS是一个分布式文件系统

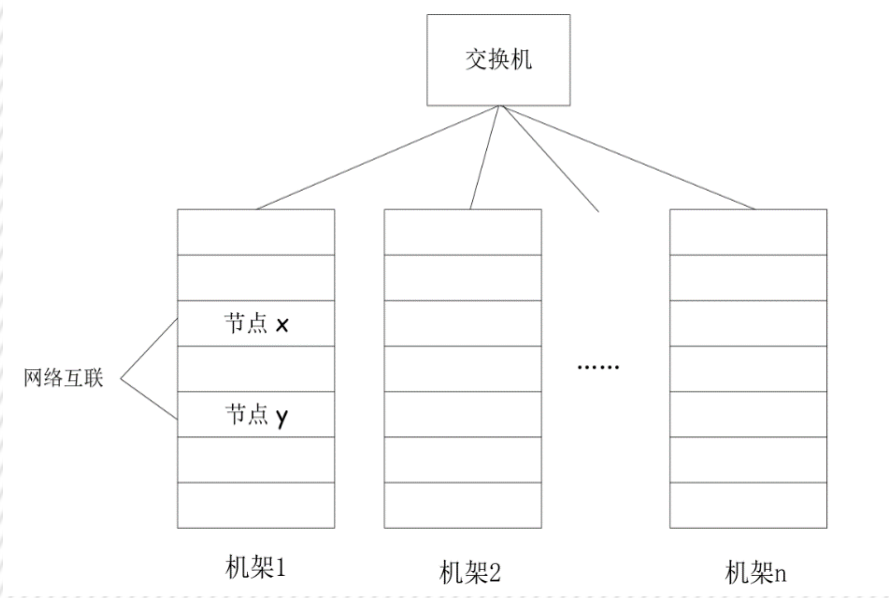
Hadoop MapReduce是一个并行计算框架

Hadoop YARN是一个资源调度框架

Hadoop是一个完整的生态系统

HDFS(Hadoop Distributed File System)

HDFS（Hadoop分布式文件系统）是一种分布式集群存储方式。

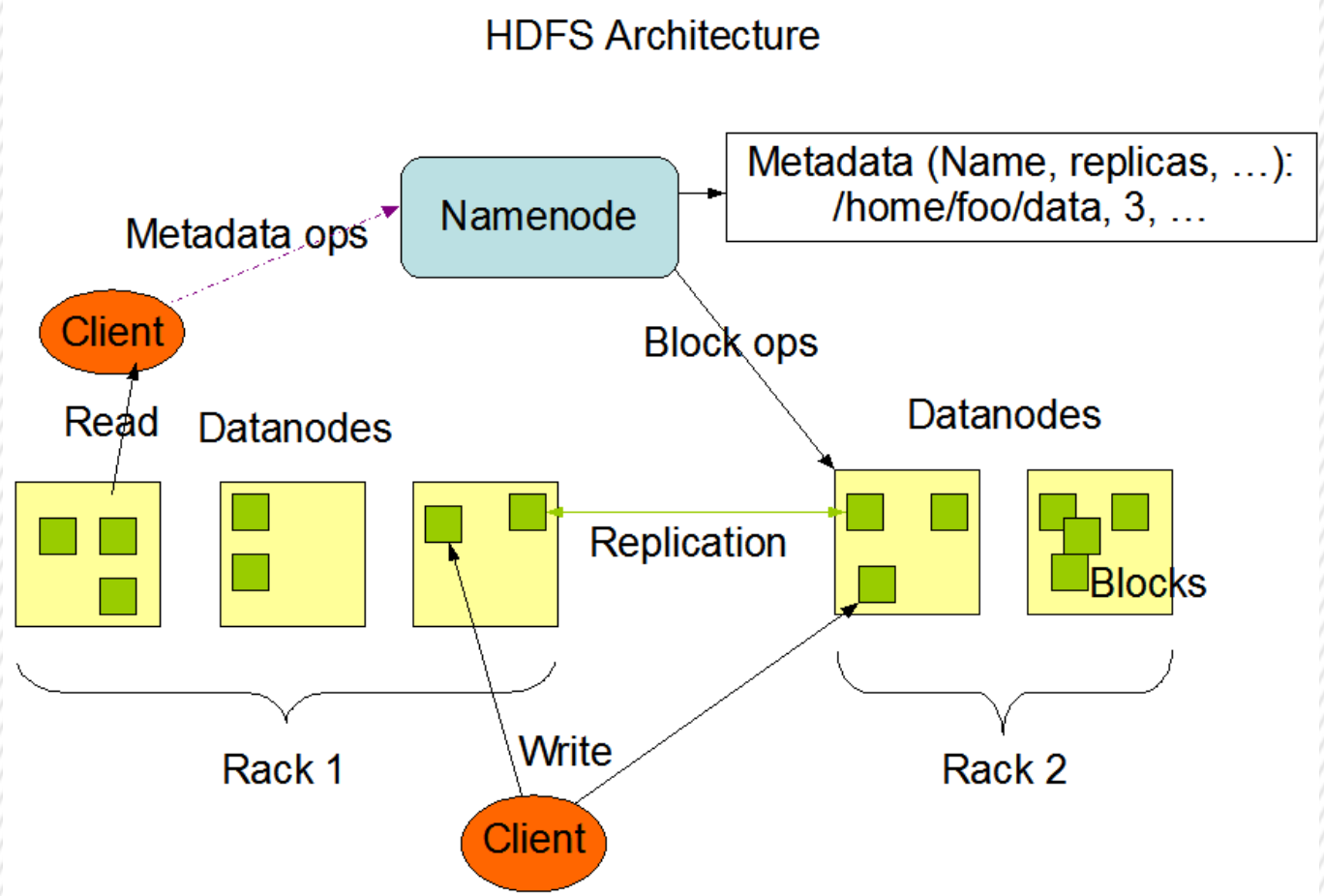


计算机集群的基本架构

将大量廉价机器通过

交换机和网络连接起来

达到增强处理能力的目的



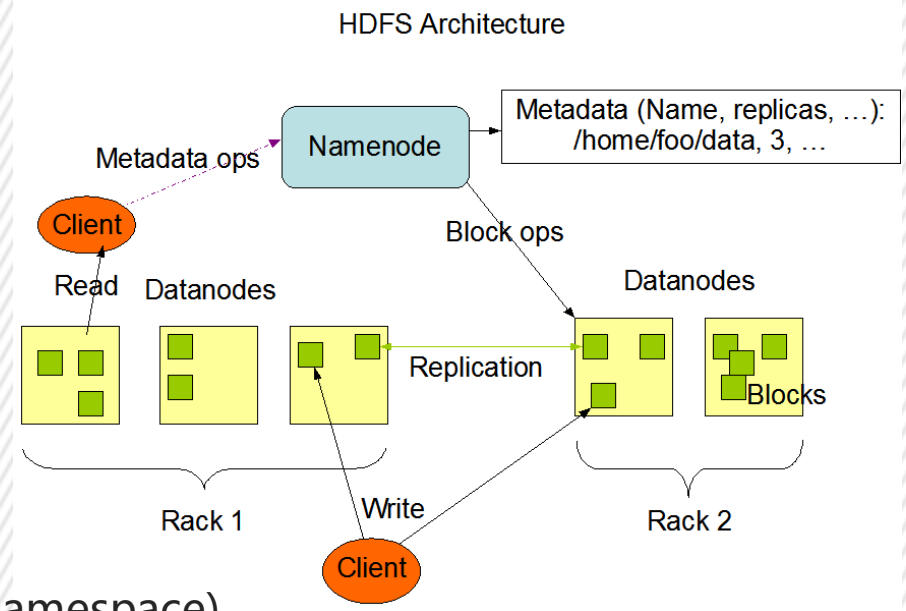
HDFS的架构(http://hadoop.apache.org/docs/r1.0.4/cn/hdfs_design.html)

Block文件分块

HDFS中文件被分成了固定大小的块（Block）。块的大小远大于普通的文件系统。

Hadoop-1.x默认是64MB，2.x默认128MB。大的分块可以减小寻址开销。

- 适合大规模文件的存储
- 便于元数据（Meta data）的管理
- 便于数据备份，提高容错性和可用性



NameNode名称节点

NameNode名称节点又称主节点。负责管理分布式文件系统的名字空间(Namespace)。

有两个重要的数据结构：FsImage和EditLog。

- 存储元数据，保存在内存中
- 元数据是描述数据的数据
- 保存文件，文件块，数据节点DataNode之间的映射关系
- 处理来着客户端的请求
- 配置副本策略

NameNode名称节点

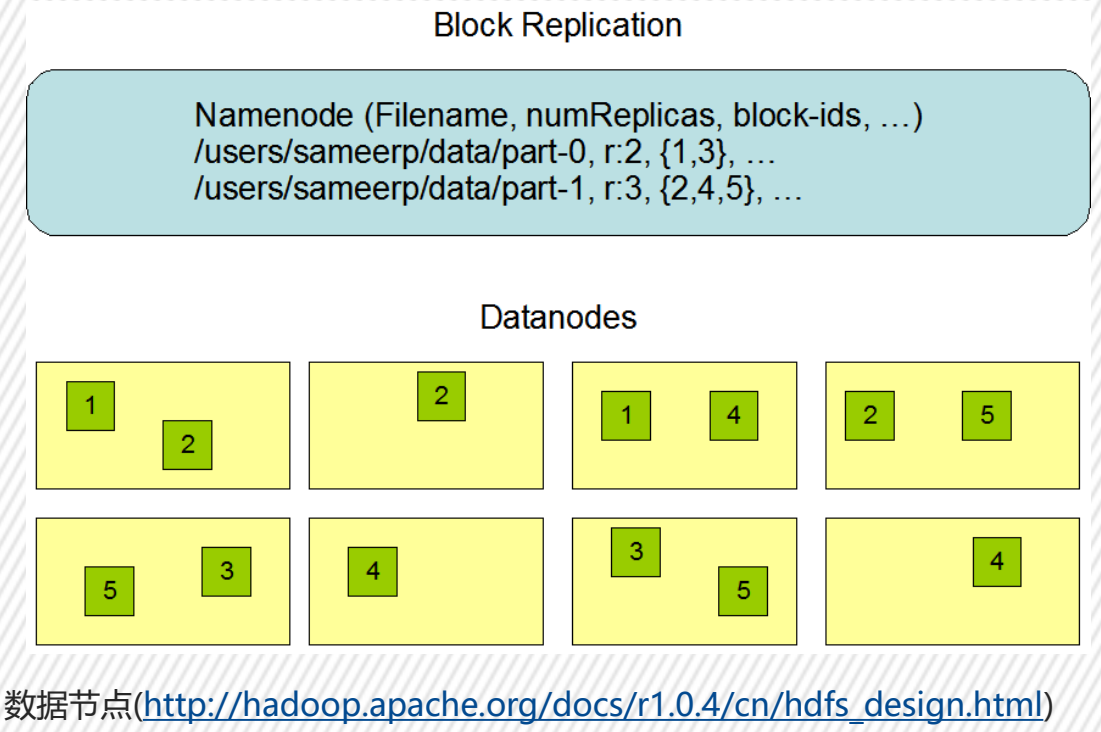
Namenode负责维护文件系统的名字空间，任何对文件系统名字空间或属性的修改都将被Namenode记录下来

- FsImage：元数据镜像文件。存储某一时段的元数据。
- EditLog：操作日志文件。记录了针对文件的创建、删除等操作。

名称节点的不会实时保证与内存中元数据的同步，而是延迟的。

当有“写请求”到来的时候，NameNode会首先修改EditLog。

然后定期的执行这种告知操作，保证块映射是最新的。



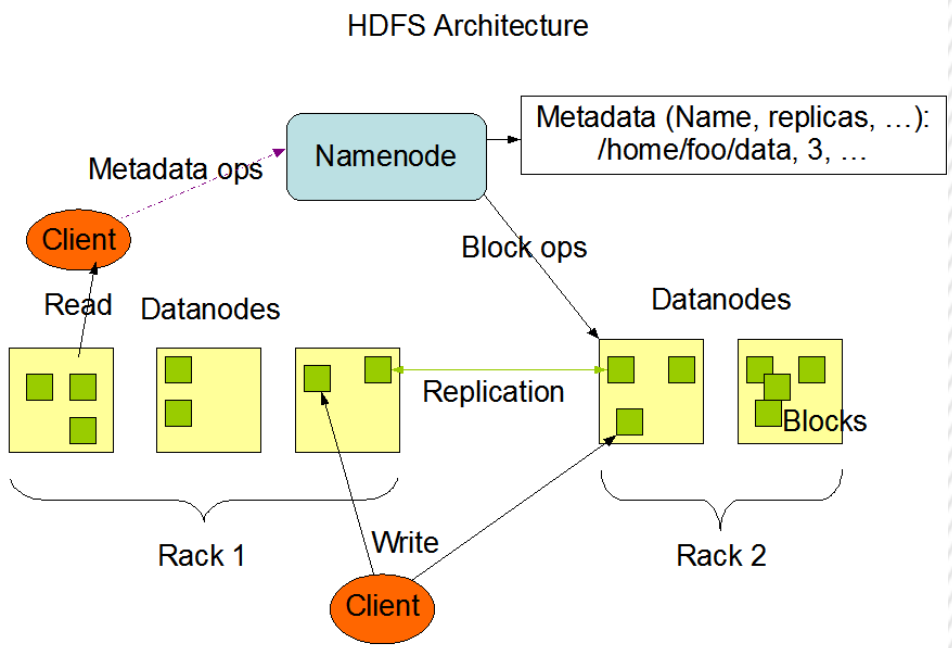
数据节点(http://hadoop.apache.org/docs/r1.0.4/cn/hdfs_design.html)

DataNode数据节点

- 存储真实数据的节点。数据被存储在机器所在的Linux文件系统上。
- 根据名称节点的调度进行存储和检索。
- 定期向名称节点发送自己的存储块列表信息。

通信协议

- 网络通信，所有的通信都是建立在TCP/IP协议上的。
- 客户端（Client）与数据节点之间通过RPC远程过程调用通信。
- NameNode和DataNode之间通过数据节点协议通信。SSH登录

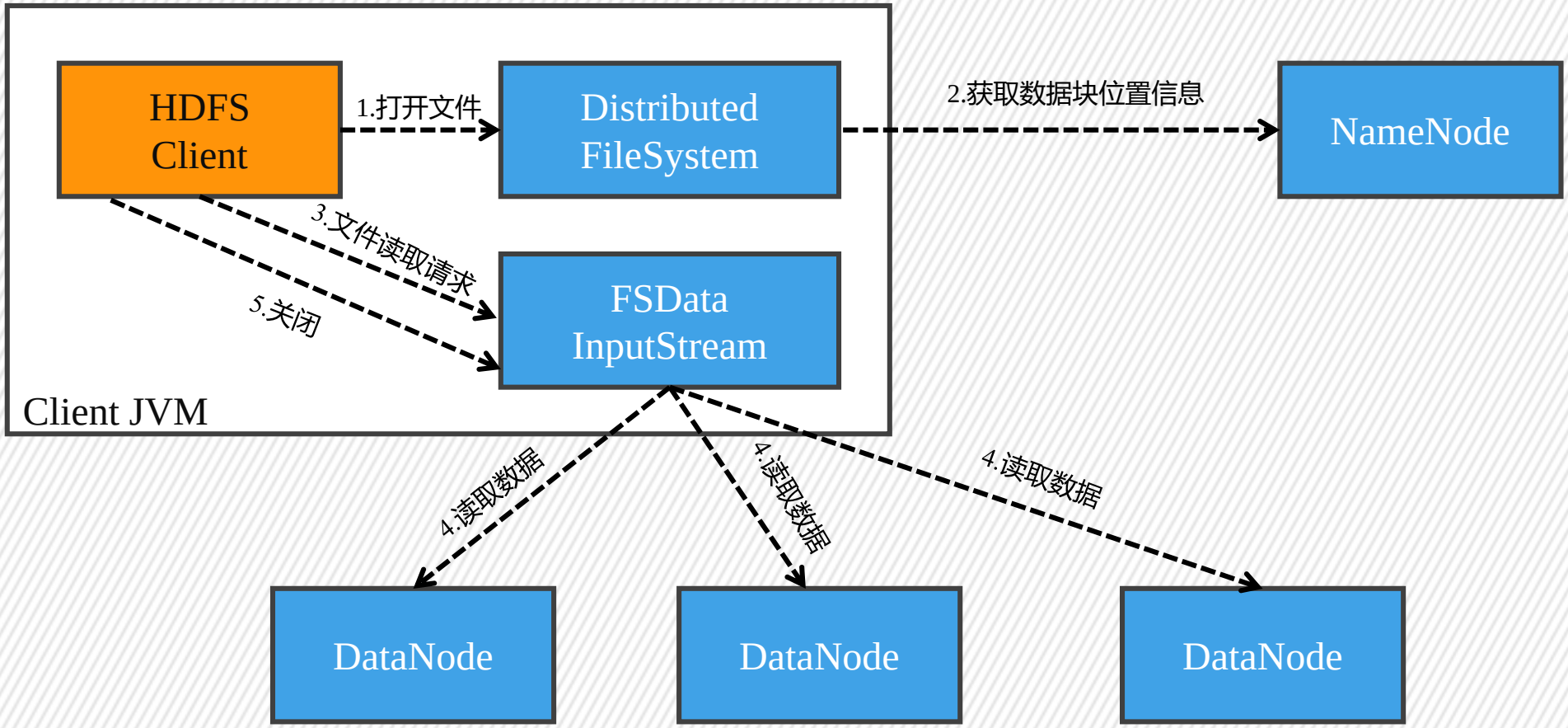


Client客户端

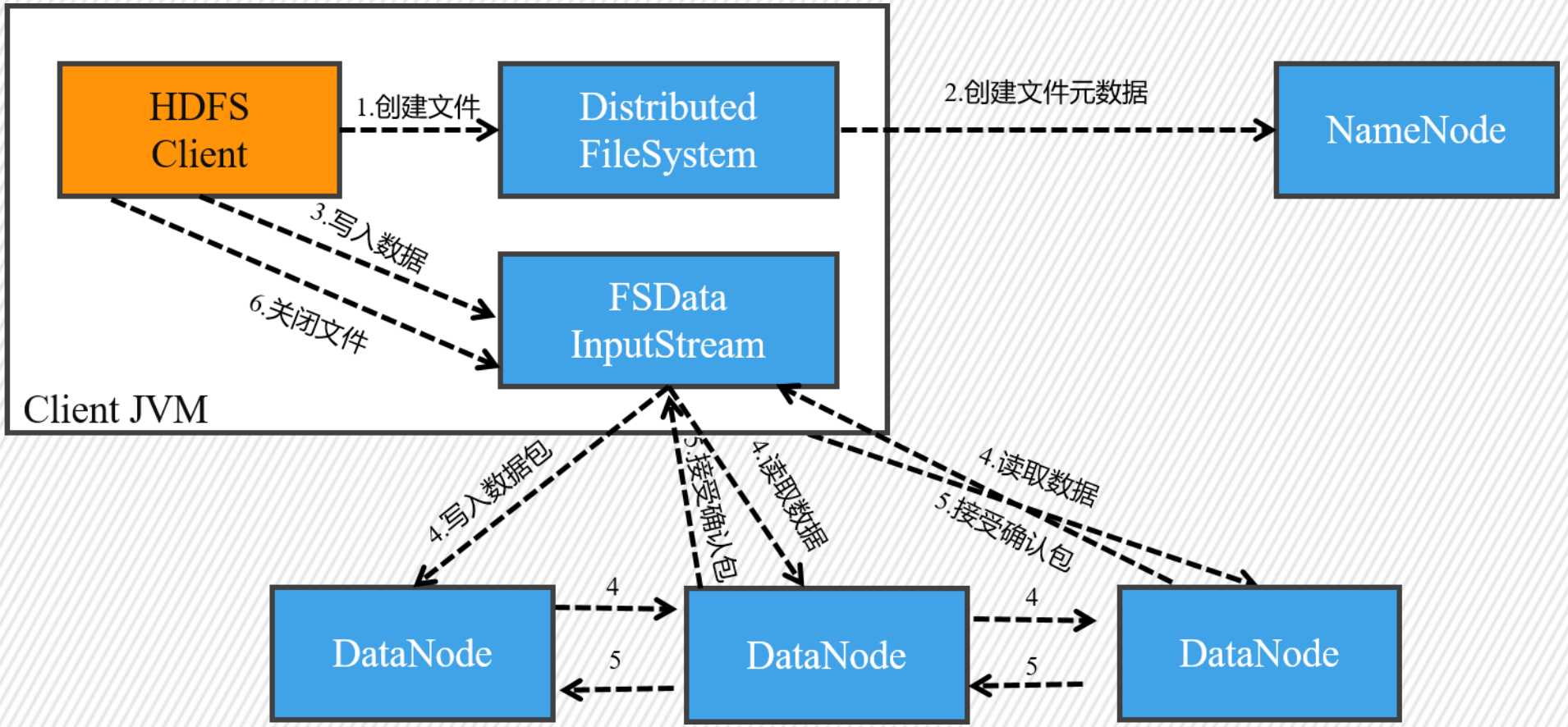
就是使用HDFS的用户或应用程序员。

- HDFS提供了应用程序接口（Java API）和Shell交互方式
- 还可以通过Web界面进行访问HDFS。http://localhost:50070

读数据的过程


HDFS Client
1.打开文件
Distributed FileSystem
2.获取数据块位置信息
NameNode
3.文件读取请求
5.关闭
FSData InputStream
Client JVM
4.读取数据
4.读取数据
4.读取数据
DataNode
DataNode
DataNode

写数据的过程

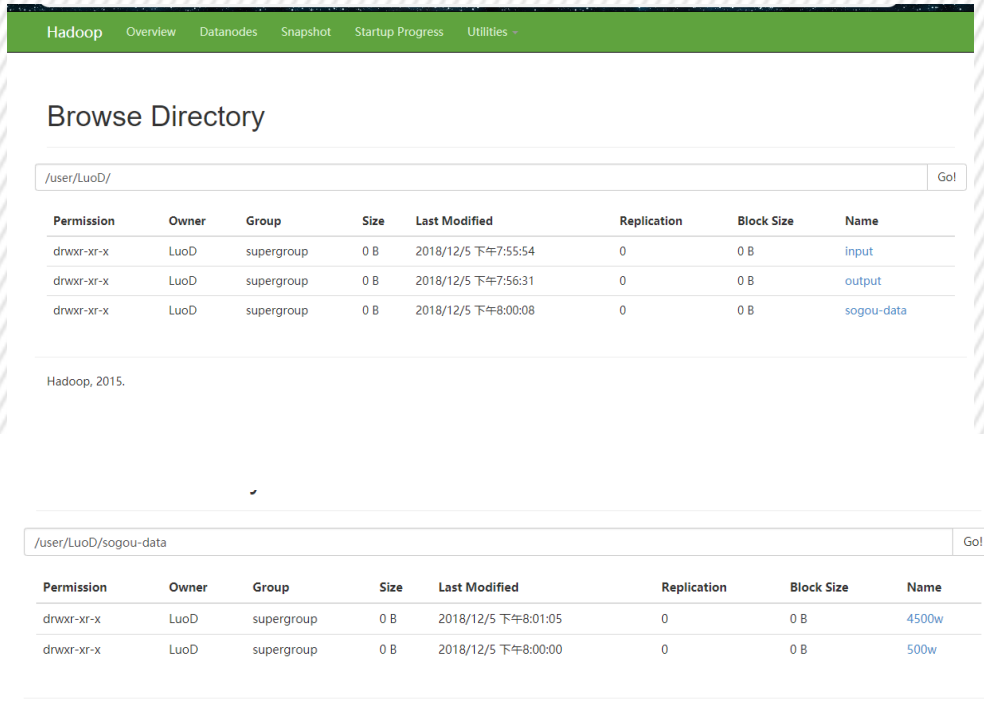

HDFS Client
1.创建文件
Distributed FileSystem
2.创建文件元数据
NameNode
3.写入数据
6.关闭文件
FSData InputStream
Client JVM
4.写入数据包
5.接受确认包
4.读取数据
4.读取数据
5.接受确认包
DataNode
DataNode
DataNode
4
5
4
5

HDFS Shell交互和Web界面

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> <linux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

HDFS提供的常用Shell操作

https://hadoop.apache.org/docs/r1.0.4/cn/hdfs_shell.html



Web界面下的HDFS

高容错性的实现

HDFS具有很高的容错性和可用性，兼容廉价的硬件资源。HDFS中，视硬件出错为一种常态。

- 名称节点出错，FsImage和EditLog被损坏，整个HDFS文件系统都会被影响。
- 数据节点出错，大量数据节点，宕机是很正常的。
- 数据出错。网络传输，磁盘写入原因导致数据出错。

HDFS的解决方案：

- SecondaryNameNode，第二名称节点。
- 心跳感知：数据节点定期向名称节点发送“心跳”信息。若没有，则视为宕机，生成新的数据副本。
- 文件被创建时，创建文件块信息摘录。读取文件时，利用该摘录文件，对数据块进行md5和sha1校验。

HDFS的特点

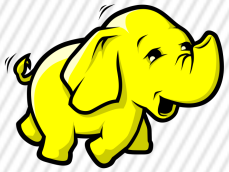
HDFS有别于其他的分布式文件系统，实现了下列目标：

- 大规模数据集
- 兼容廉价的硬件设备（设备宕机，仍然正常工作，冗余机制）
- 流式数据的读写
- 强大的平台兼容性（Linux版本多样）

但是相应的也有一些缺点：

- 不适合低延迟数据的访问
- 存储小文件不利
- 不支持多用户写入或任意修改文件
- 文件只能向后增加，不支持删除操作

Hadoop HDFS是一个分布式文件系统



Hadoop MapReduce是一个并行计算框架

Hadoop YARN是一个资源调度框架

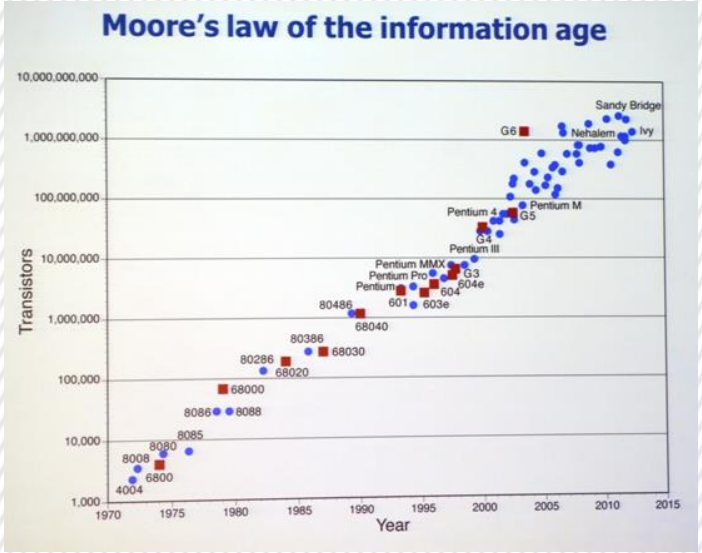
Hadoop是一个完整的生态系统

关于并行计算

请大家思考几个问题

- **什么是并行计算?**
- **并行和并发有什么区别?**
- **为什么要并行计算?**

我们已经有了像多线程并发、多核程序设计以及CUDA等编程技术，为什么还要MapReduce?



摩尔定律告诉我们，单机的计算能力有限。集群式的并行计算是超越单机计算能力的一种途径。

	传统并行计算框架	MapReduce
集群架构/容错性	共享式(共享内存/共享存储)，容错性差	非共享式，容错性好
硬件/价格/扩展性	刀片服务器、高速网、SAN，价格贵，扩展性差	普通PC机，便宜，扩展性好
编程/学习难度	what-how，难	what，简单
适用场景	实时、细粒度计算、计算密集型	批处理、非实时、数据密集型

传统并行计算框架（如OpenMP）与MapReduce的对比

关于并行计算

- **Amdal定律**

程序中必定有串行的部分。在给定可并行化的比例下，加速比不能随处理器数目增加而增加，而是受限于串行化那一部分。

$$S = \frac{1}{(1-P) + \frac{P}{N}}$$

S:加速比

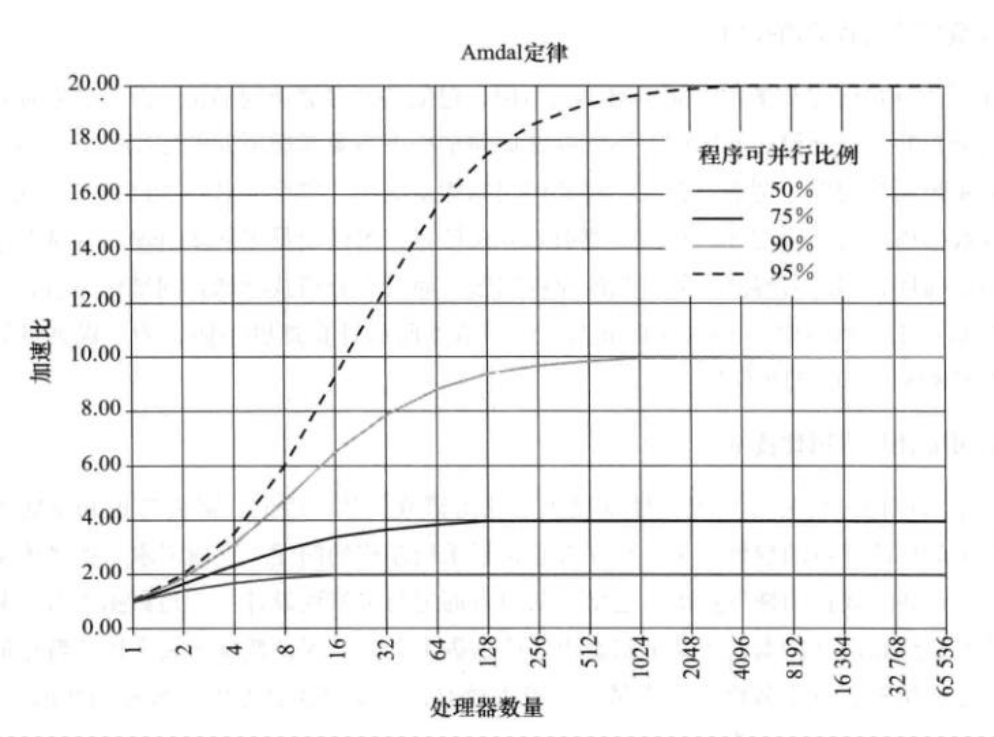
P:可并行部分比例

N:处理器数量

- **Gustafson定律**

在放大系统规模的情况下，加速比可以随处理器的数量成一定比例的线性增长，串行比例不再是加速比的瓶颈。

这恰恰很适合大数据问题场景，大量数据，简单计算。



Amdal定律的示意，截取自《深入理解大数据大数据处理与编程实践》

Hadoop MapReduce

MapReduce是一个面向大数据并行处理的计算模型、框架和平台。

- **一个模型：**

借助函数式编程语言Lisp的设计思想，提供了简洁的并行程序设计方法。

将一个并行计算任务，抽象成Map和Reduce两个基本过程，并提供了相应的接口。继承、重载…

- **一套框架：**

简单理解就是包，库。提供了一套庞大的软件集合，并实现了良好的封装。

将复杂的并行计算逻辑和系统底层细节屏蔽，给开发人员简洁易用的编程接口。

- **一个平台：**

MapReduce是基于集群的高性能并行计算平台。

它允许普通的商用服务器或廉价的个人PC机来搭建高性能的计算集群。

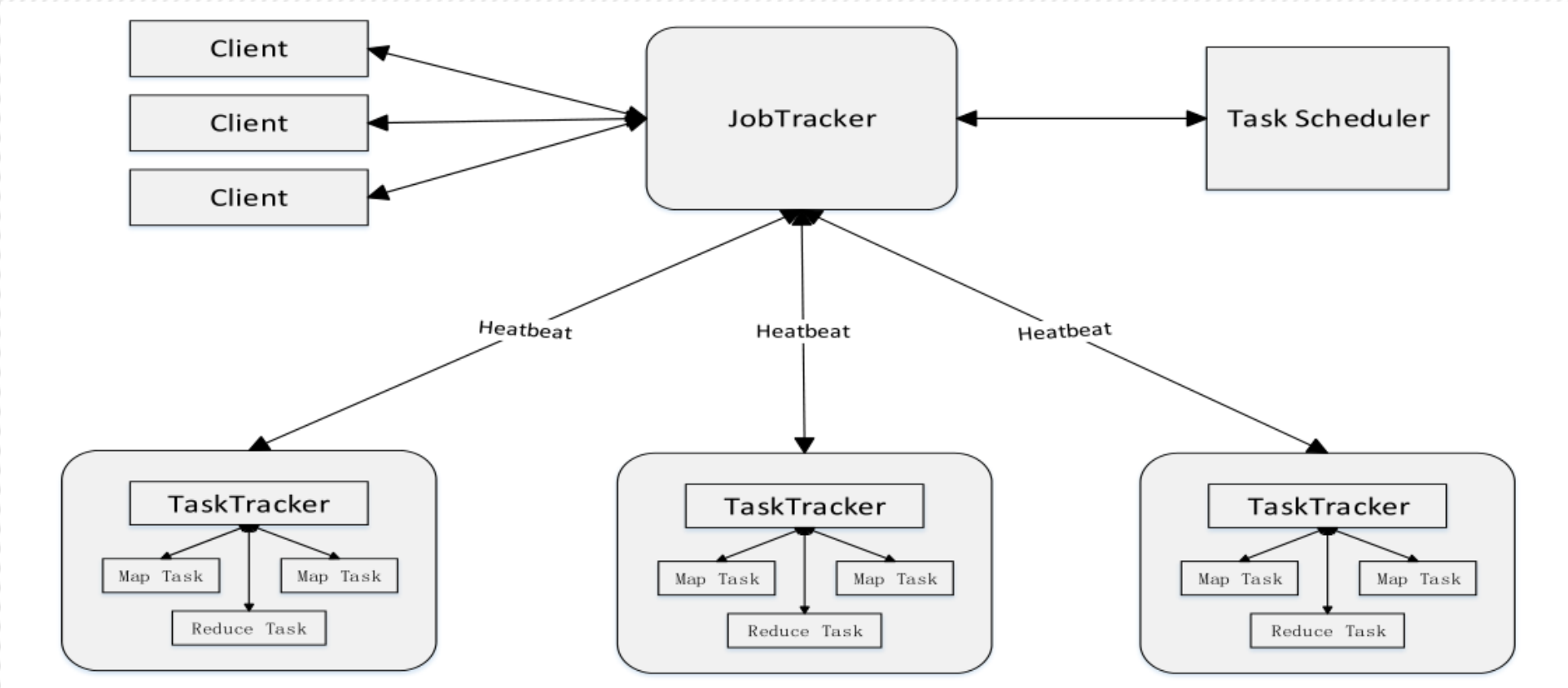
分而治之（Map and... Reduce）

- 一个存储在分布式文件系统中的大规模数据集，会被切分成许多独立的分片（split），这些分片可以被多个Map任务并行处理。
- 处理完成后通过Reduce任务进行汇总聚合，得到结果。

函数	输入	输出	说明
Map	<k_1,v_1> 如： <行号,"a b c">	List(<k_2,v_2>) 如： <"a",1> <"b",1> <"c",1>	1.将小数据集进一步解析成一批<key,value>对，输入Map函数中进行处理 2.每一个输入的<k_1,v_1>会输出一批<k_2,v_2> 。<k_2,v_2>是计算的中间结果
Reduce	<k_2,List(v_2)> 如： <"a",<1,1,1>>	<k_3,v_3> <"a",3>	输入的中间结果<k_2,List(v_2)>中的List(v_2)表示是一批属于同一个k_2的value

主从架构（Master and slave)

MapReduce体系结构主要由四个部分组成，分别是：Client、JobTracker、TaskScheduler以及TaskTracker



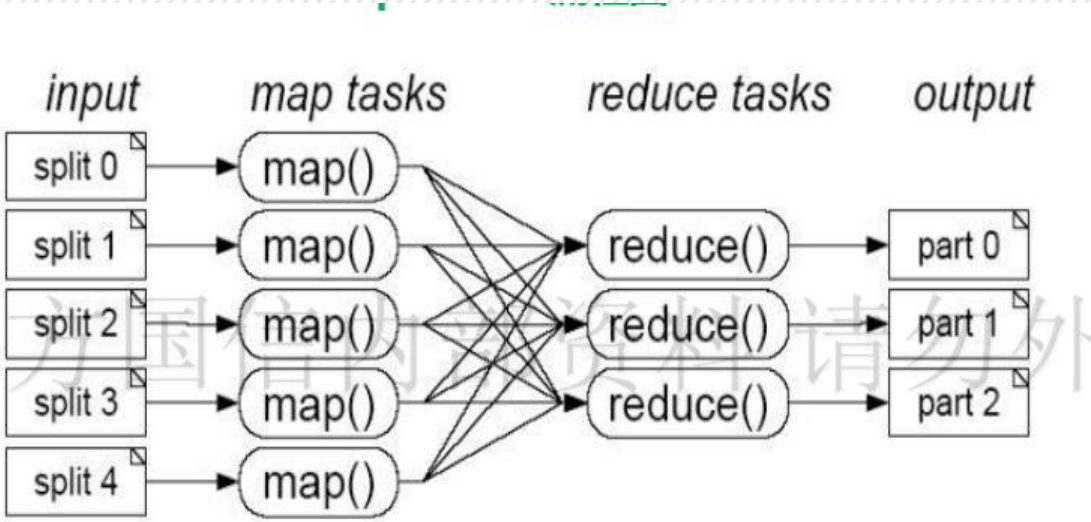
MapReduce的架构

一个JobTracker管理多个TaskTracker，通过心跳感知监督任务运行是否正常

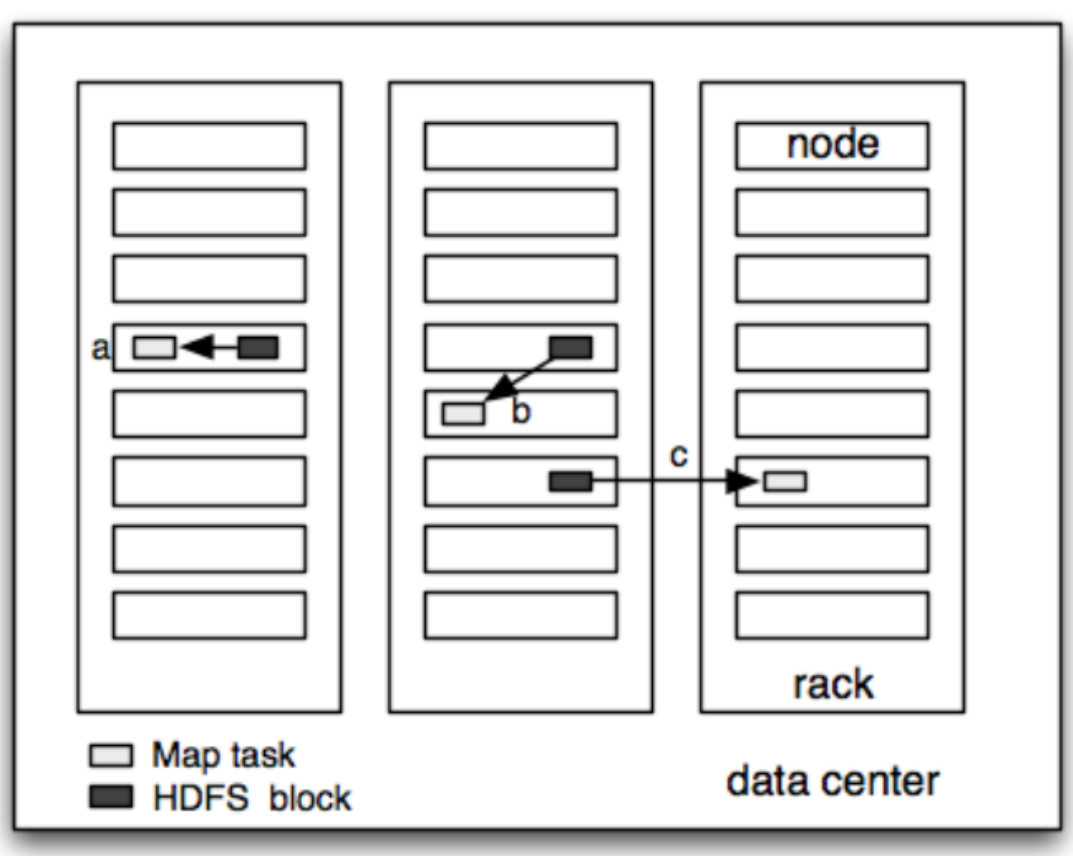
计算向数据靠拢

MapReduce的重要设计思想之一是，移动计算，就近计算，计算向数据靠拢，而非数据向计算靠拢。

因为移动数据是成本很高的。



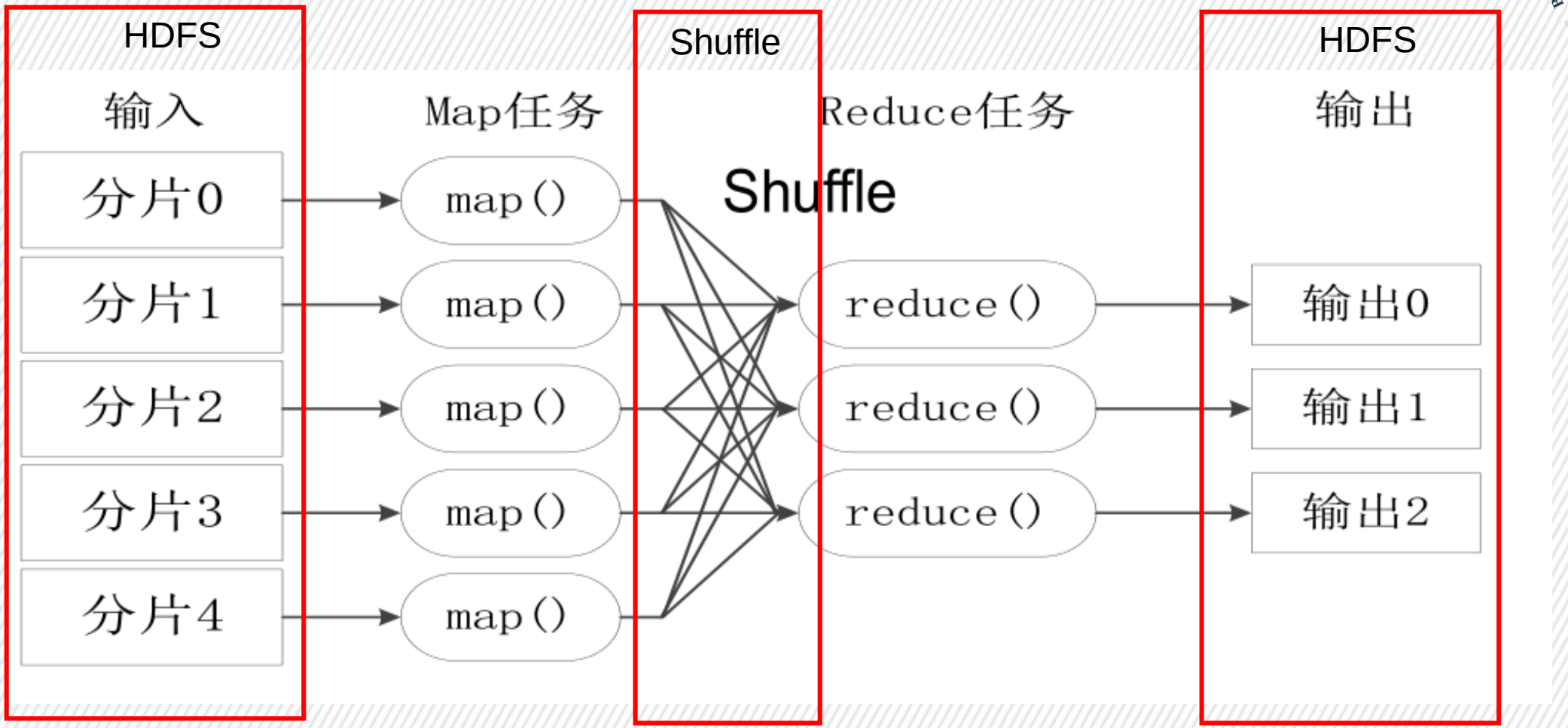
MapReduce的流程图



Hadoop尽力在输入数据驻留在HDFS中的节点上运行Map任务。

Data-local (a), rack-local (b), and off-rack (c) map tasks.

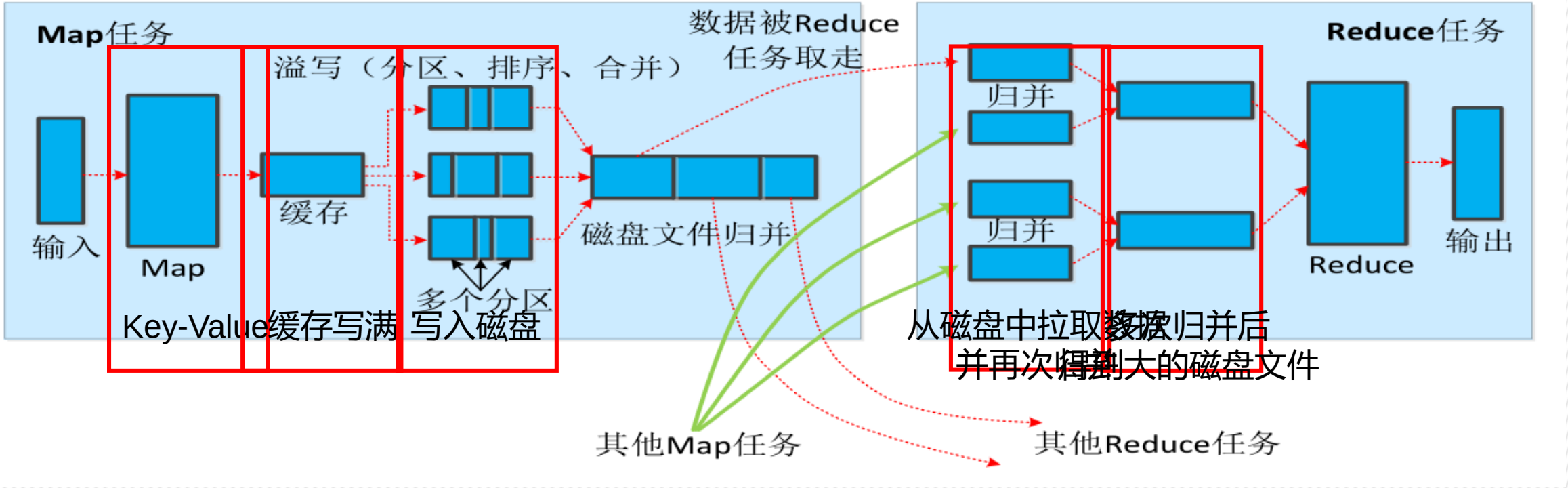
MapReduce的工作流程



MapReduce执行的流程图

注意：所有的数据流动都是MapReduce框架控制实现的，不需要用户来指定数据如何流动。

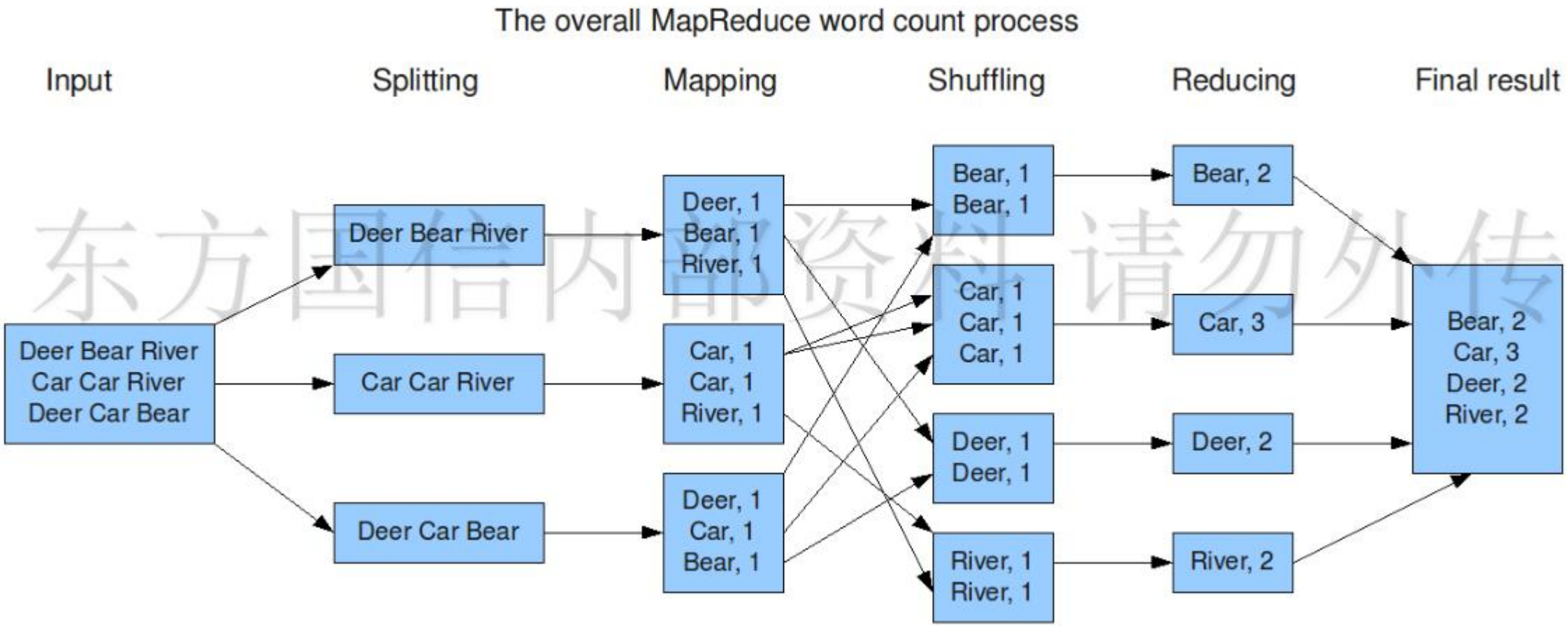
Shuffle



整体的Shuffle过程

JobTracker

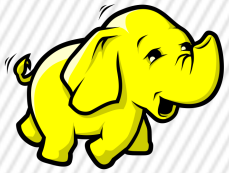
一个Word Count小例子



一个MapReduce词频统计的全流程

Hadoop HDFS是一个分布式文件系统

Hadoop MapReduce是一个并行计算框架



Hadoop YARN是一个资源调度框架

Hadoop是一个完整的生态系统

YARN: Yet Another Resource Negotiator

Hadoop2.0引入资源调度框架YARN，一种新的资源调度框架。

引入一个问题——数据倾斜

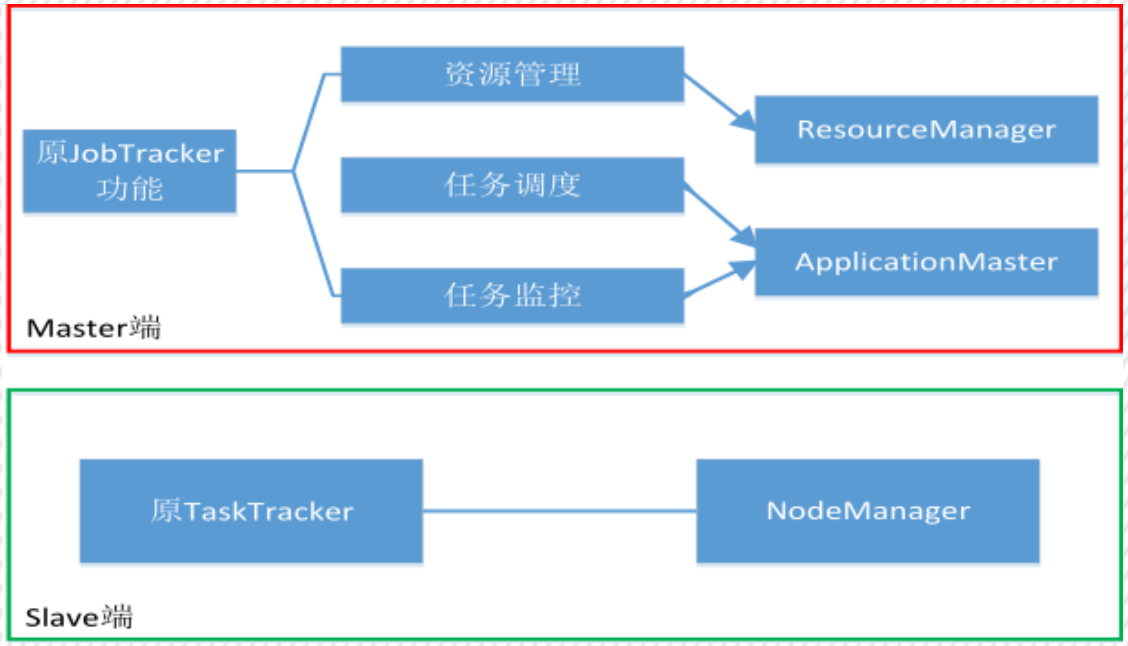
- 在实际的问题的，经常出现数据倾斜的问题，也就是数据呈现出分布不均匀的现象。
- 比如词频统计问题中，一个网页上常见词的出现频率会远多于生僻词，如果不加以调度，计算效率大大降低。
- 常见的处理方法有，设置Reducer的数量、Blancer等，以及2.0引入的YARN。

著名的短板效应

YARN的特点与解决的问题

- **1.0中存在的问题：**
  - 存在单节点故障。
  - JobTracker任务过重，全部集群中只有一个JobTracker，当任务多时内存开销很大。
  - 资源分配不合理，分配资源时只考虑MapReduce的任务数，不考虑CPU、内存的占用。

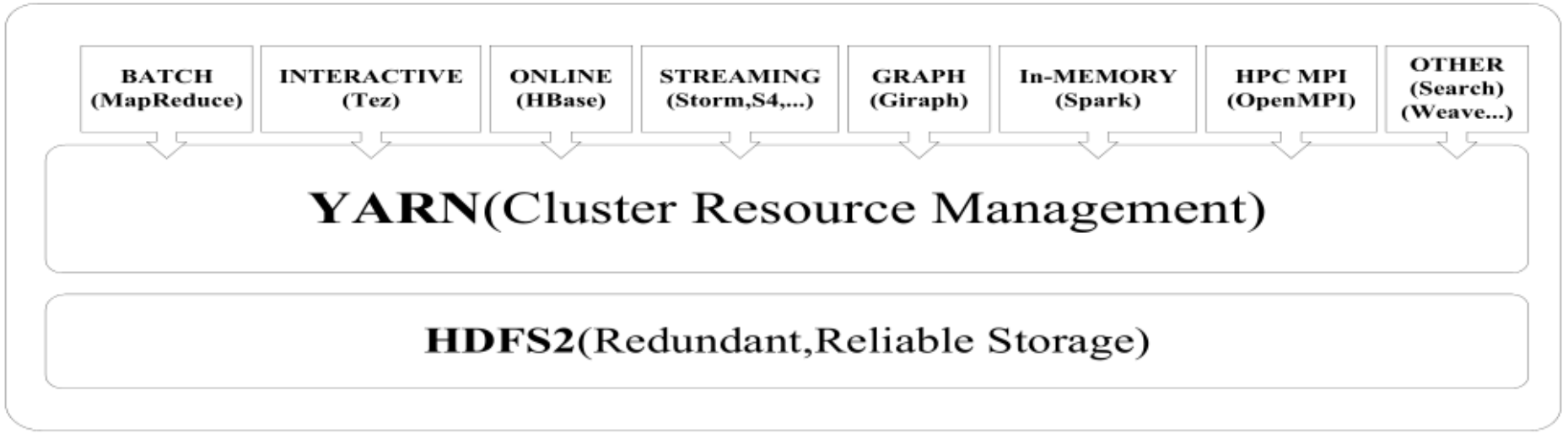
- **YARN的设计思路：**
  - 将原JobTracker的功能拆分。资源管理、任务调度、任务监控。
  - ResourceManager负责资源管理。
  - 任务监控和任务调度由ApplicationMaster负责。
  - 原来的TaskTracker成为NodeManager。



1.0到2.0的转变

目标：实现一个集群，多个框架

- **问题提出：**
- 一个企业，可能同时存在多种不同的业务应用场景，需要采用不同的计算框架来应对。
- 不同框架之间的资源调度方式不一样，情况是：一个框架，一个集群。
- 这样导致了很多问题
  - 数据无法共享
  - 资源利用率低
  - 维护代价很高
- **YARN带来了：**
- 统一的资源调度服务
- 集群上应用的负载混搭
- 各计算框架共享底层存储资源

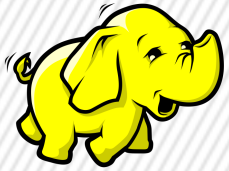


在YARN上部署不同的计算框架

Hadoop HDFS是一个分布式文件系统

Hadoop MapReduce是一个并行计算框架

Hadoop YARN是一个资源调度框架

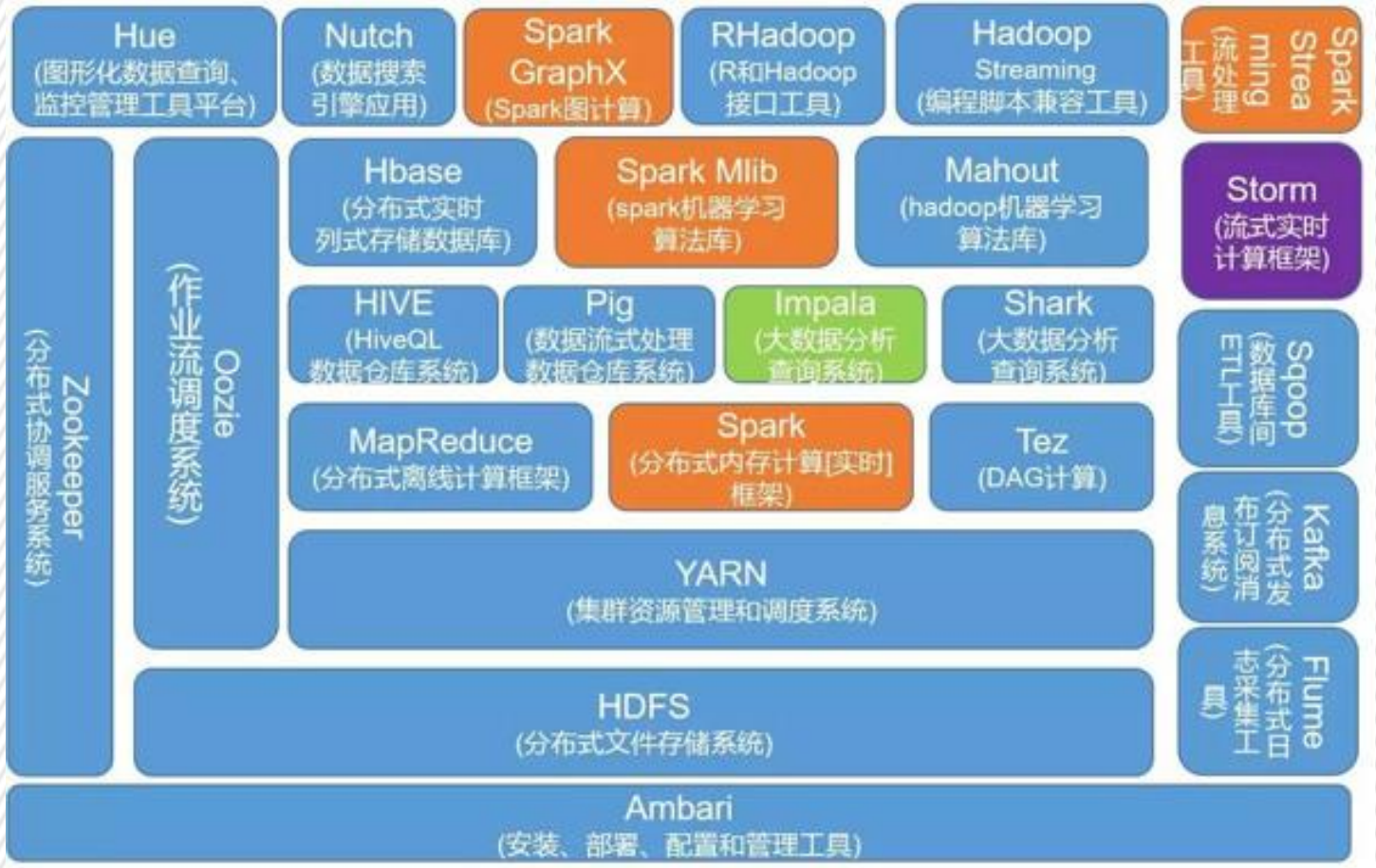


Hadoop是一个完整的生态系统

Hadoop, Not Only Hadoop

Hadoop，而又远非Hadoop。

我们平常说的Hadoop是一个完整的大数据生态系统，有着全套的技术栈。



Hadoop大数据技术栈，来自知乎

HBase面向列的实时存储数据库

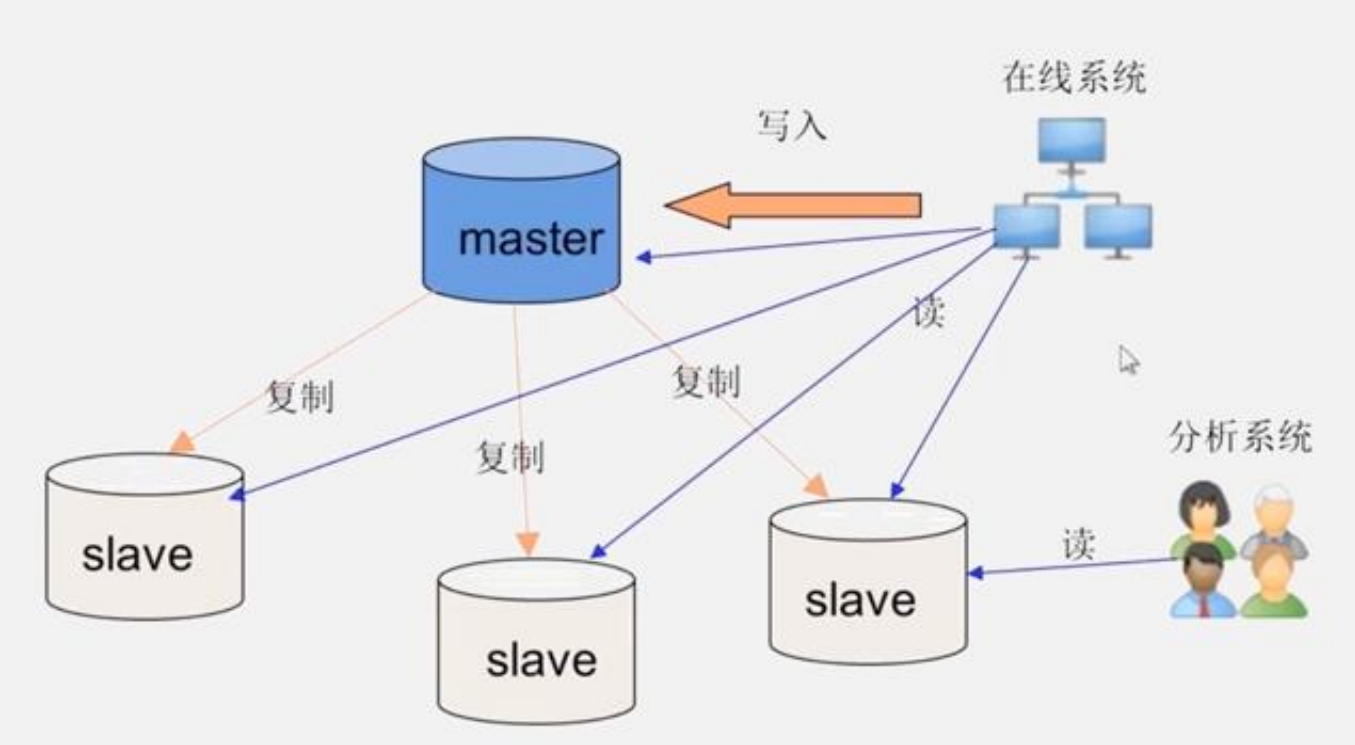
- HBase源自于谷歌公司的产品BigTable，是BigTable的一个开源实现。
- 谷歌公司用BigTable来存储大量的网页，以满足互联网搜索的需求。
- HBase具有良好的性能，以及非常好的水平扩展性。允许用上千台服务器来存储上亿行、百万列的数据。

HBase和BigTable的底层技术对应关系

	BigTable	HBase
文件存储系统	GFS	HDFS
海量数据处理	MapReduce	Hadoop MapReduce
协同管理服务	Chubby	Zookeeper

HBase解决的问题

- 传统的关系型数据库，水平扩展性不强，无法面对当前的大规模数据以及急速增长的数据存储需求。
- 关系型数据库的模式很难变更，难以面对半结构化或非结构化数据。

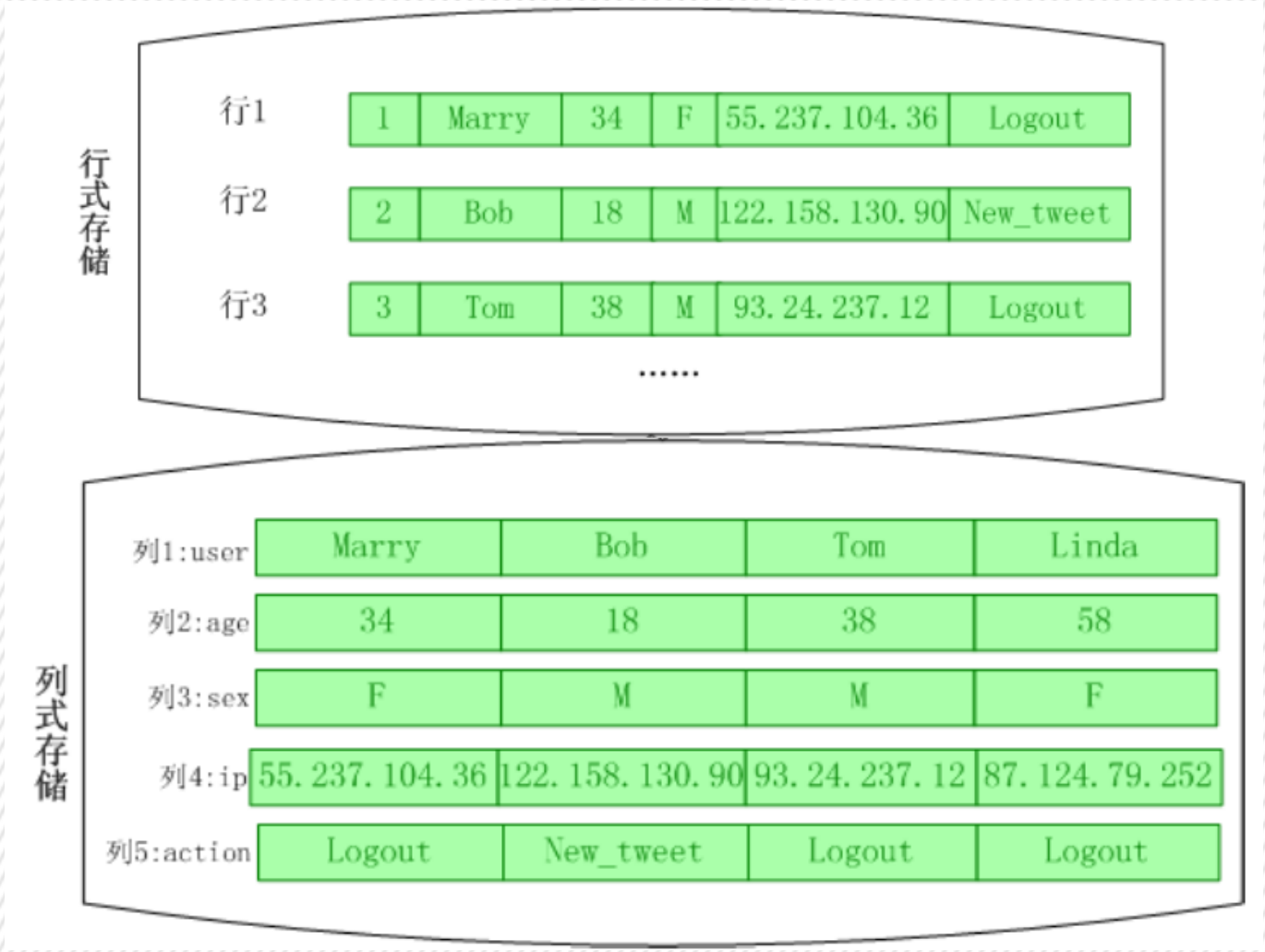


传统关系型数据库的主从复制模型

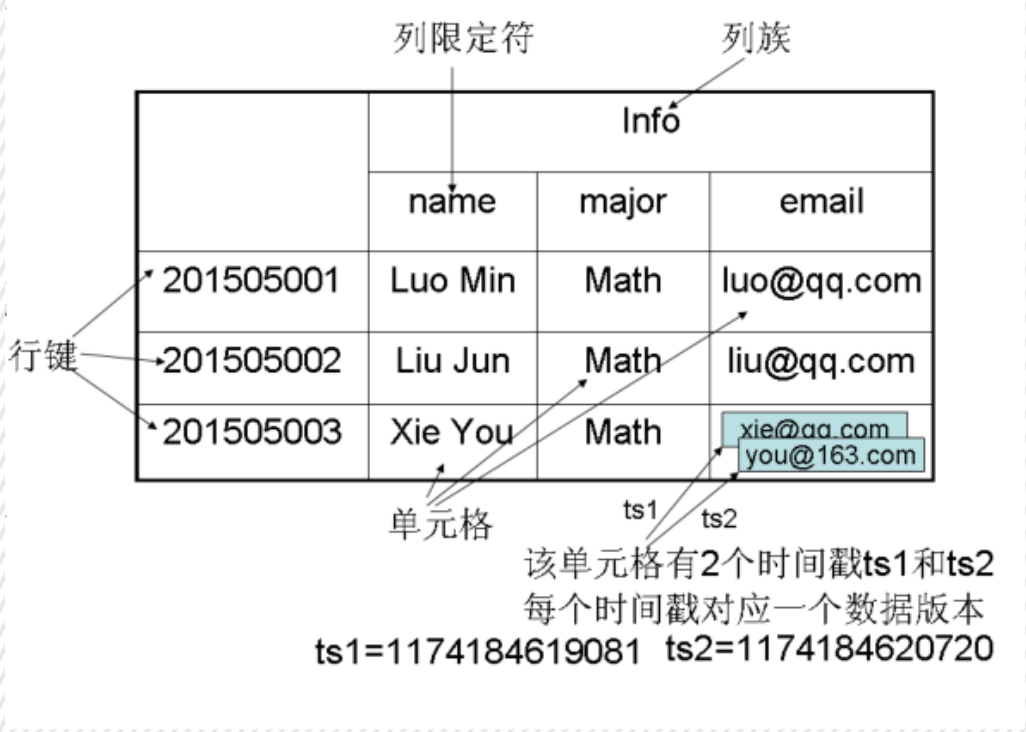
Hbase与关系型数据库的区别

- **数据类型**：传统关系型数据库，提供了大量数据类型，INT,CHAR,TEXT...而HBase只存储Java字节数组。
- **数据操作**：关系型数据库，提供了大量的数据操作，而HBase提供的操作很简单,比如join操作就被避免了。
- **存储模式**：关系型数据库，是面向行的存储，HBase是面向列的存储。这给其带了很好的扩展性。
- **数据索引**：关系型数据库，可以对不同列建立复杂的索引，而HBase只支持行键索引。
- **数据维护**：关系型数据库，做更新Update操作后，原来的数据会被覆盖掉，而HBase中不会，通过时间戳区别。
- **可伸缩性**：关系型数据库，很难实现横向也就是水平扩展，纵向扩展能力也有限。HBase就是为灵活的水平扩展而设计的，可以轻易的增加或者减少硬件的数量来实现性能的伸缩。

面向行的存储和面向列的存储



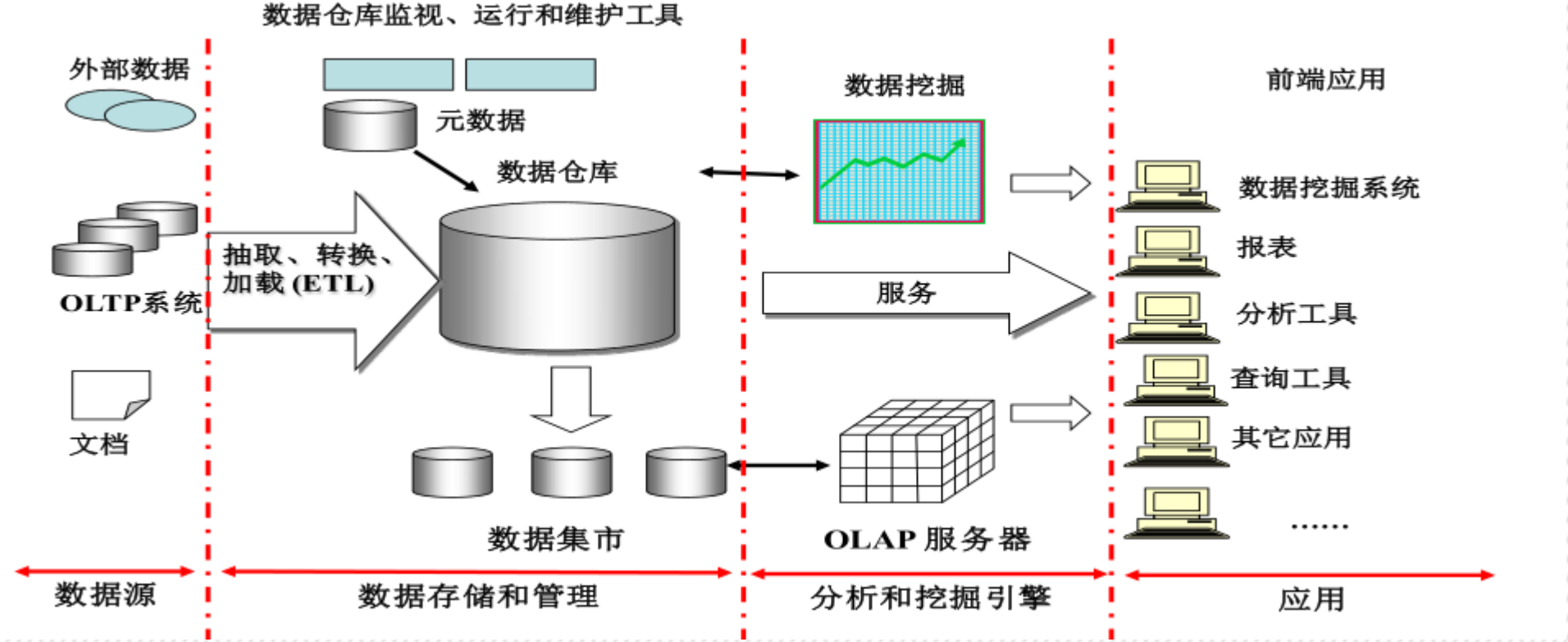
行存储与列存储的区别



HBase里的一张表

Hive数据仓库系统

什么是数据仓库?



数据仓库的体系结构

数据仓库是一个

- 面向主题的
- 集成的
- 相对稳定的
- 反映历史变化的

数据集合。

里面存储的是历史数据。存进去，就不怎么改变。

什么是Hive?

- Hive是一个构建在Hadoop顶层的数据仓库工具。
- 它支持大规模数据的存储、分析，且具有良好的可扩展性。
- 提供了类SQL语句——HiveQL，让用户用熟悉的方法对HDFS中的数据进行查询。
- HiveQL语句会被解析成MapReduce任务被执行。
- Hive本身并不存储数据，而需要依赖其他存储产品，可以视为一个用户编程接口。

从HiveQL到MapReduce

输入HiveQL语句

驱动模块接受该命令

对该命令进行编译解析

优化器对该命令进行优化

通过执行器对该命令进行执行

- 驱动模块对SQL语句进行词法语法解析，生成抽象语法树。
- 将抽象语法树转化成为查询块。
- 将查询块转化成逻辑查询计划。

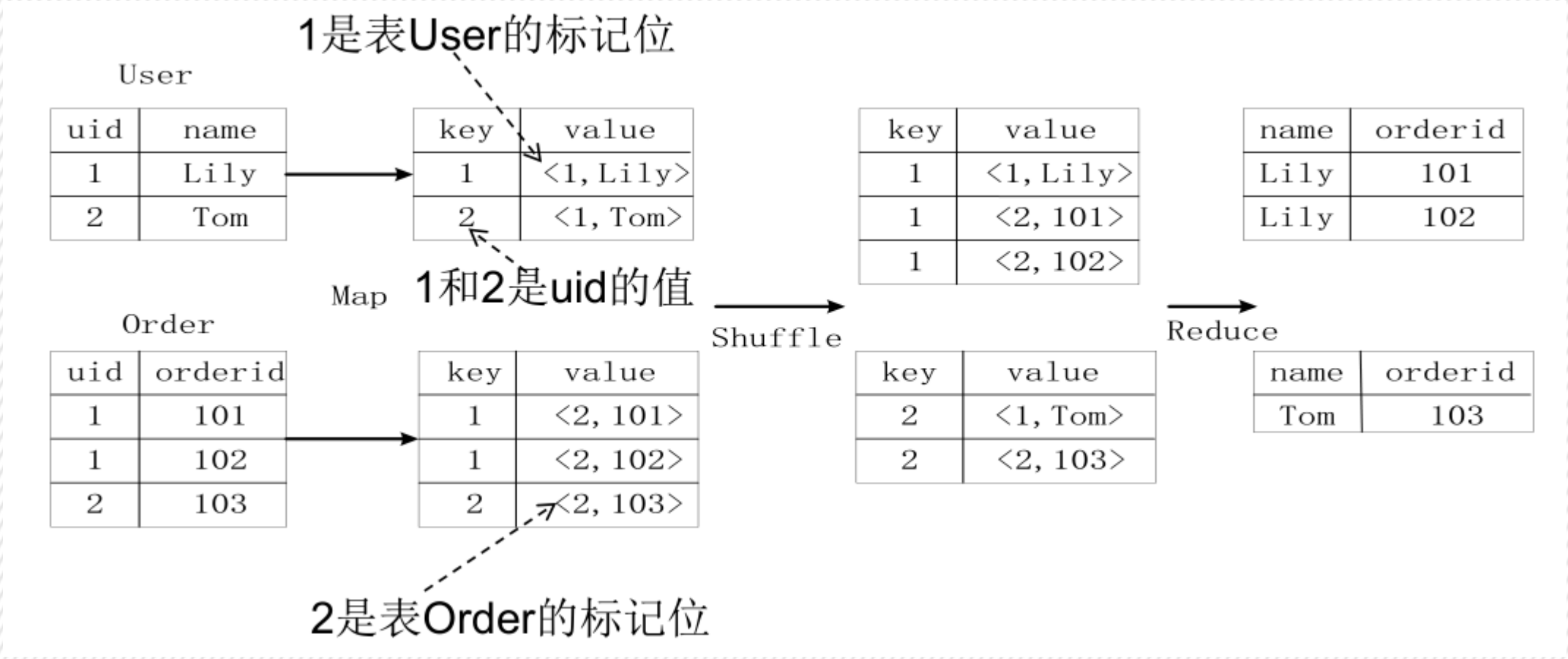
前端

- 重写逻辑查询计划，对其进行合并优化。减少MR程序的数量。
- 生成最终的MR任务。

后端

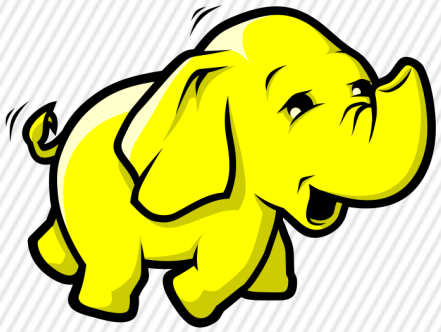
- 执行器对其进行执行。

Example of Join:

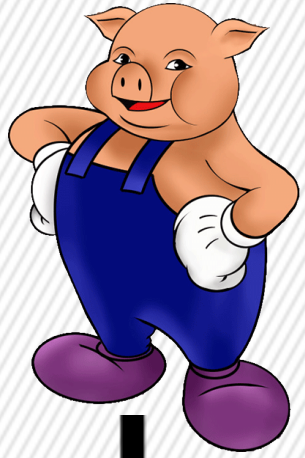


Join语句被转化成为MapReduce的执行过程

欢迎来到大数据的动物园



APACHE STORM™



Flink

APACHE HBASE

Hadoop生态及其重要组件介绍

基于内存的计算框架和流式计算简介

小组学期工作汇报

Spark的诞生和简介

Spark的特点及与Hadoop的区别

Spark RDD弹性受限分布式数据集

Hadoop的局限于不足

只支持批处理

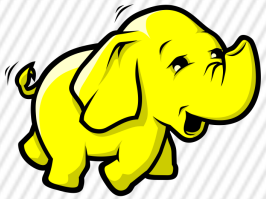
- Hadoop MapReduce只支持大规模数据的批处理，无法响应流数据以及实时性高的场景，比如交互查询。
- 一次批处理往往需要好几个小时甚至数天，对于目前多变的场景来说很不适应。

迭代计算效率低

- MapReduce在迭代计算过程中，需要将中间结果写入磁盘或HDFS，这带来了很大的开销。
- 有时Reduce任务必须要等待Map任务的结束才能开始，浪费资源。

表达能力有限

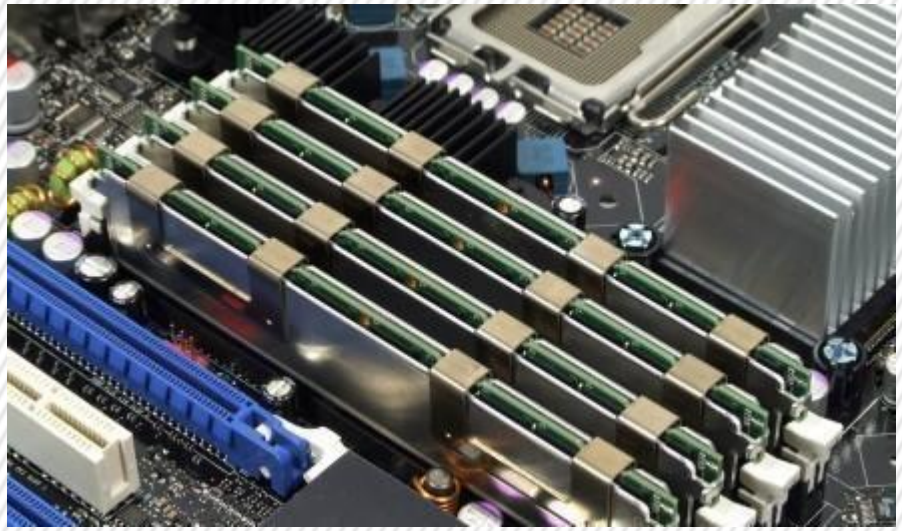
- Hadoop把所有计算问题都抽象为Map和Reduce两个过程。
- 很多问题很难或者说不能用Map和Reduce来解决。



Spark简介

什么是基于内存的计算框架?

- 随着现在半导体技术的集成度变高，内存的大小也越来越大。
- 一个正常的集群，内存总和可达1TB.
- 这时候就又可能将全部数据都加载到内存中完成计算，而不需要读写磁盘。
- 内存的读写速度，至少是磁盘的1000倍。



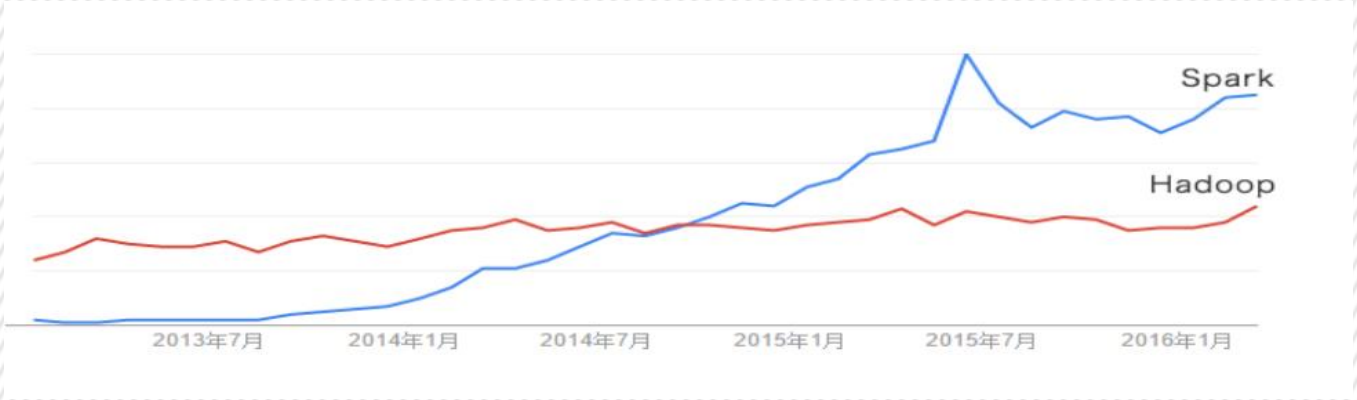
服务器中的内存条

Spark简介

- Spark由美国UC Berkeley大学AMP实验室于2009年开发。
- Spark是基于内存的大数据并行计算框架。
- Spark由Scala语言编写开发。
- Scala语言是一种现代的、多范式的编程语言，支持面向对象、函数式编程。
- 同时Spark还提供了Java,Python和R语言的编程接口。

Spark在2014年打破了Hadoop的排序记录

Spark	Hadoop
206个节点 23分钟	2000个节点 72分钟
100TB数据	



谷歌趋势——Hadoop与Spark的对比

Spark RDD

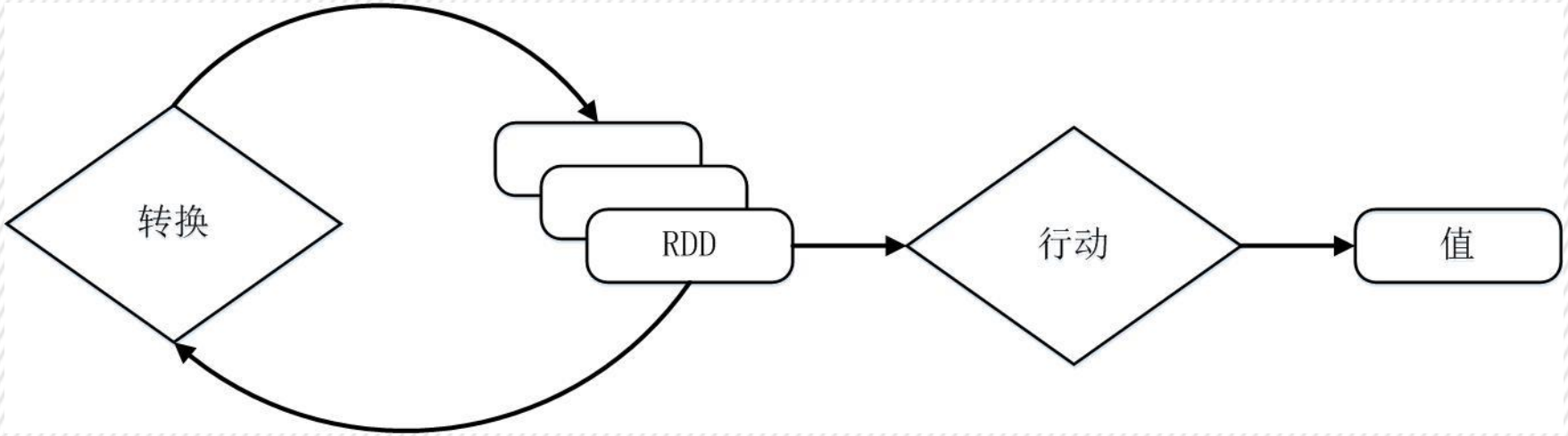
- Spark RDD(Resillient Distributed Dataset)弹性受限分布式数据集。
- RDD是分布式内存的一个概念，它提供了一种高度受限的共享内存模型。
- 它的本质上是一个**只读**的记录分区集合。

如何对RDD进行修改?

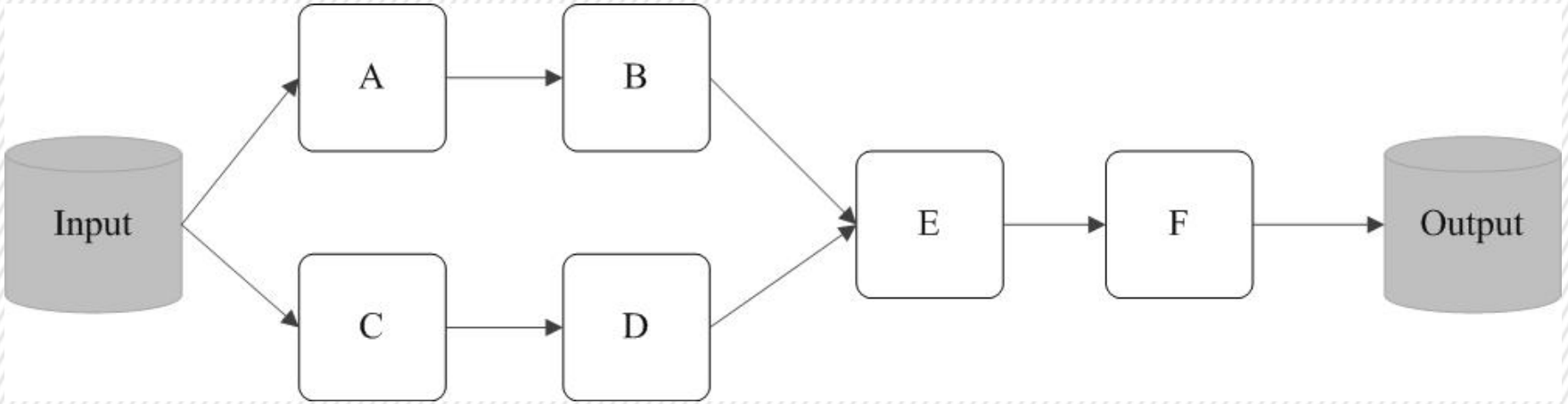
- RDD提供了一组丰富的操作并支持常见的数据运算。可以被分为***Transformation***和***Action***两种类型。

DAG有向无环图

- 每次执行一个任务时，Spark都会构建一个由RDD依赖构成的有向无环图。



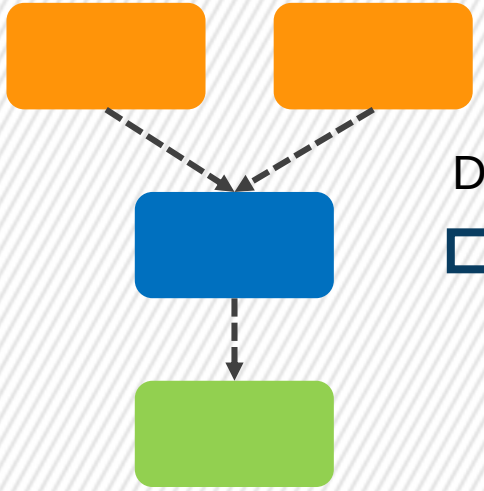
从RDD到值之间的一系列转换



Spark为RDD之间构建DAG

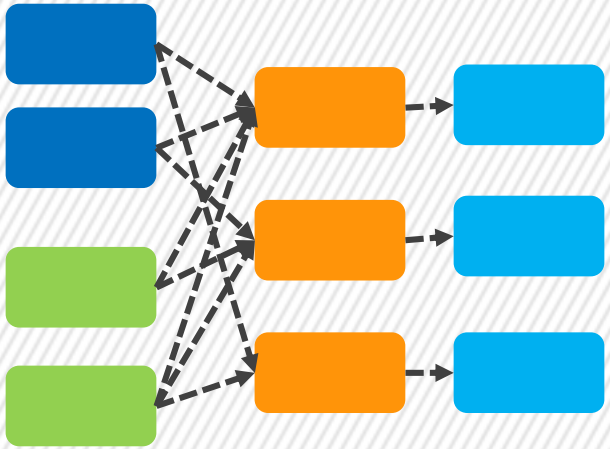
RDD运行过程

RDD对象



DAG

DAGScheduler



任务集

TaskScheduler

Cluster Manager

任务

Worker

Threads

Block Manager

rdd1.join(rdd2)
.groupyBy(…)
.filter(…)

把DAG图划分成多个阶段

Cluster Manager
进行任务调度

执行任务

RDD带来的特性

- 高效的容错性
- 现有的容错机制，多半是采用数据副本冗余以及记录日志的方式实现。
- 当发生错误处理的时候，节点间存在数据的大量拷贝复制。带来了很大的开销。
- RDD的只读特性，带来了天生的容错性。
- RDD的计算是通过相互间的依赖关系构建的DAG图。如果出现错误，只需要重新计算即可。无需数据拷贝。
- 中间结果保存在内存中，不需要写回磁盘。
- RDD中存放的数据，可以是“对象”，避免了序列化和反序列化的开销。

流式计算框架Storm

Hadoop生态及其重要组件介绍

基于内存的计算框架和流式计算简介

小组学期工作汇报

小组工作介绍

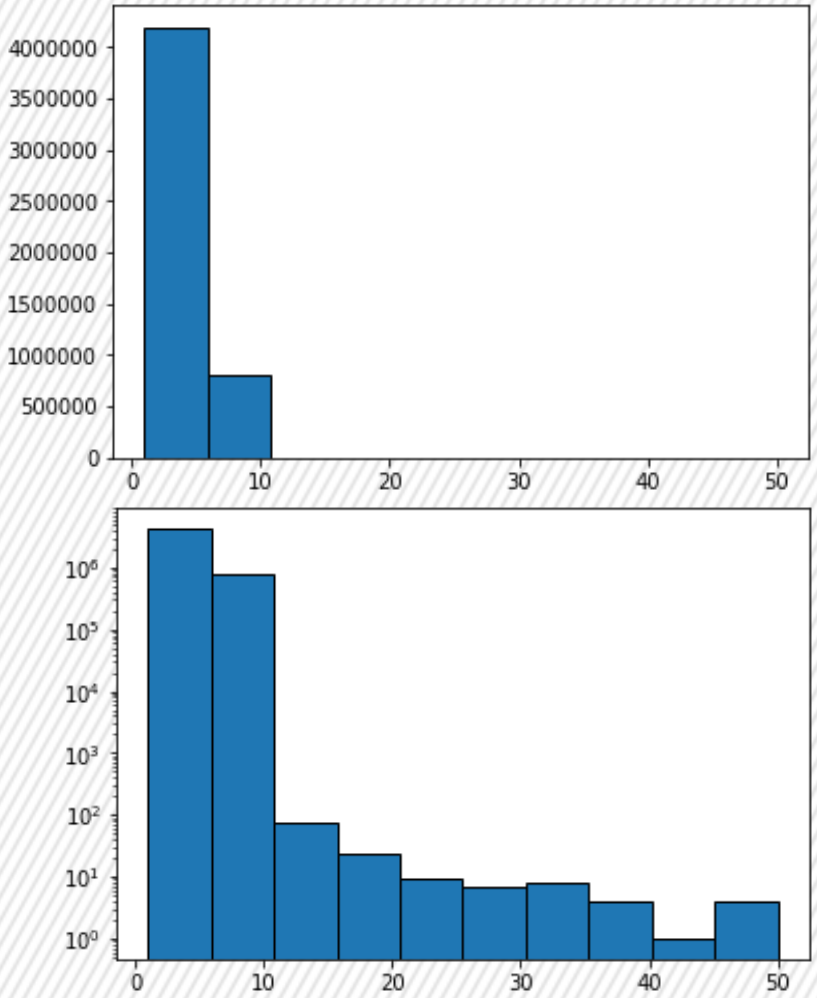
- 线上学习为主，微信群中讨论，交流学习的进度，转发大数据相关的学习资源。
- 每周进行一次线下会议，交流和分享为主。会有一到两名同学准备PPT来讲解大数据技术，或者是演示相关操作。
- 同时还维护了一个github仓库，存放学习资源，一些PDF学习资料，还有记录每周工作情况，用issue的方式呈现。

传送门在此：https://github.com/RonDen/2018--Hadoop-，欢迎访问，fork或者star....

大合照

对用户查询日志的分析

- **数据说明和字段解析：**
  - 两个数据集，4G包含4500W条记录，500M包含500W条记录
  - 第一个字段为用户访问时间，包含时序信息
  - 第二个字段为用户ID，浏览器Cookie信息自动赋值
  - 第三个字段为用户查询词
  - 第六个字段是用户最终点击的url
  - 4，5字段为数字，分别是该url在返回结果中的排名，以及点击顺序
- **统计分析：**
  - 数据共500w行，1352664不同的用户。27%的用户会在一个页面点击两次。
  - 近90%的用户会在点击页面响应的第一个url。
  - 其中还发现了一些异常点击数据，可能是网络爬虫。



个人学习的一些思考

2018年IBM收购红帽
象征其向云计算领域的扩张

关于什么是“云”

- **学习一些工具的使用：**
  - 学好Linux、用熟Vim，你会发现安装Hadoop很简单
  - Idea、PyCharm、Maven、Git这样的工具让大数据学习变得很简单。50%+的代码都能自动生成
  - 用好搜索工具...90%的问题可以通过谷歌或百度解决
- **注意版本**
- **积累、总结和交流：**
  - 各种工具，知识和属于自己的技术栈是慢慢积累起来的
  - 剩下10%谷哥和度娘解决不了的问题，通过交流和实践来解决
  - 一个人可以走的很快，一群人可以走的很远

感谢我的组员：
宋文宇、夏迎祺、张松鸣、杨世雄、
周雅婷、李航、蔡秉岐、金任任

THANK YOU

谢 谢

Don’t worry, just do IT!