大数据实战应用

Presented by Ruizhi-Wu

Outline

(1)大数据平台基本框架

(2)Google

(3)Hadoop

(4)HDFS

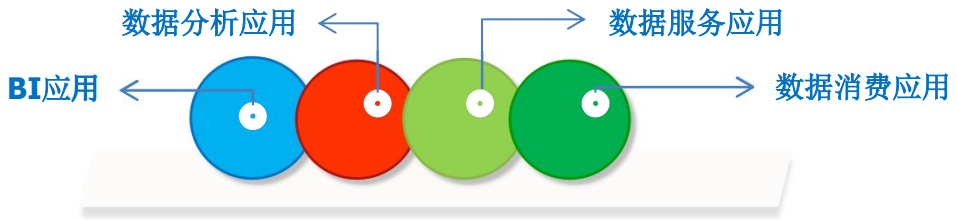
(5)Mapreduce

(6)Hadoop各个组件简介

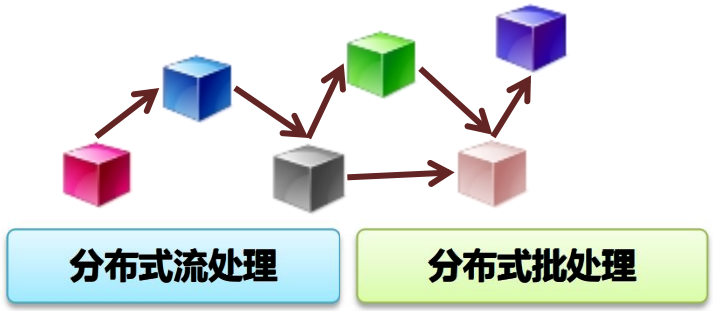
(7)Spark

大数据处理平台框架

服务层



分析层

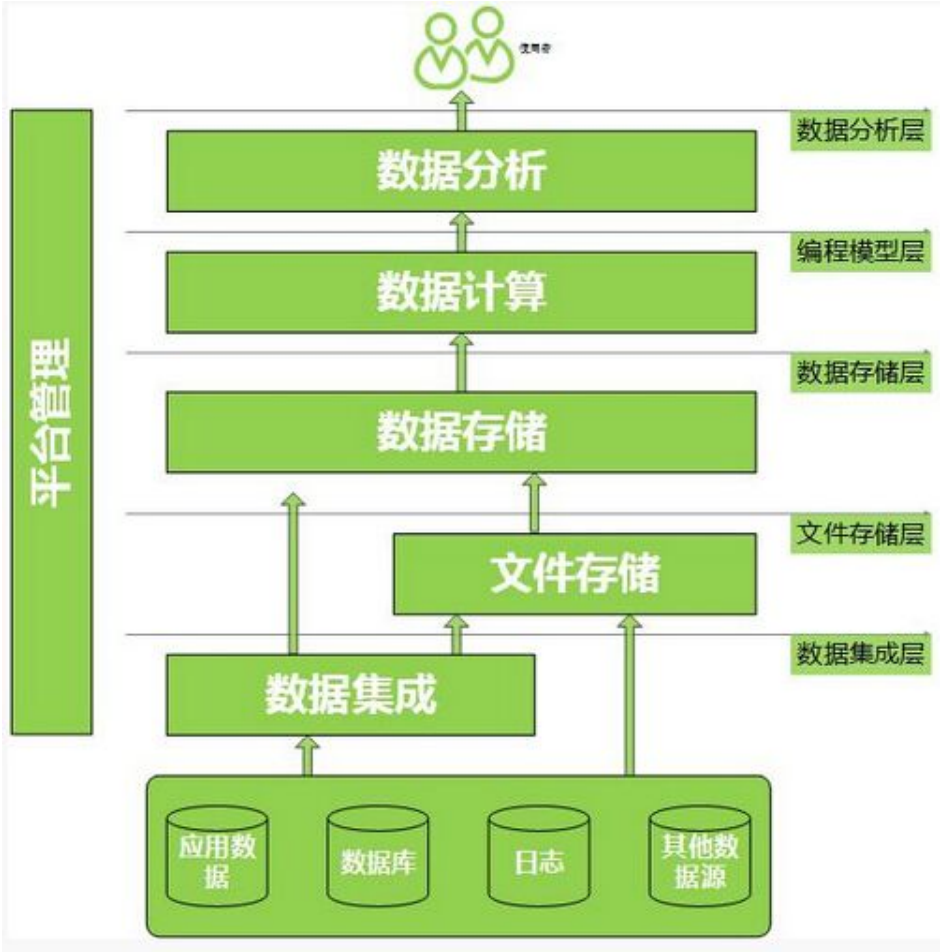


管理层

采集层

存储层

大数据平台架构


数据分析层
数据分析
编程模型层
数据计算
数据存储层
数据存储
文件存储层
文件存储
数据集成层
数据集成
应用数据
数据库
日志
其他数据源
平台管理

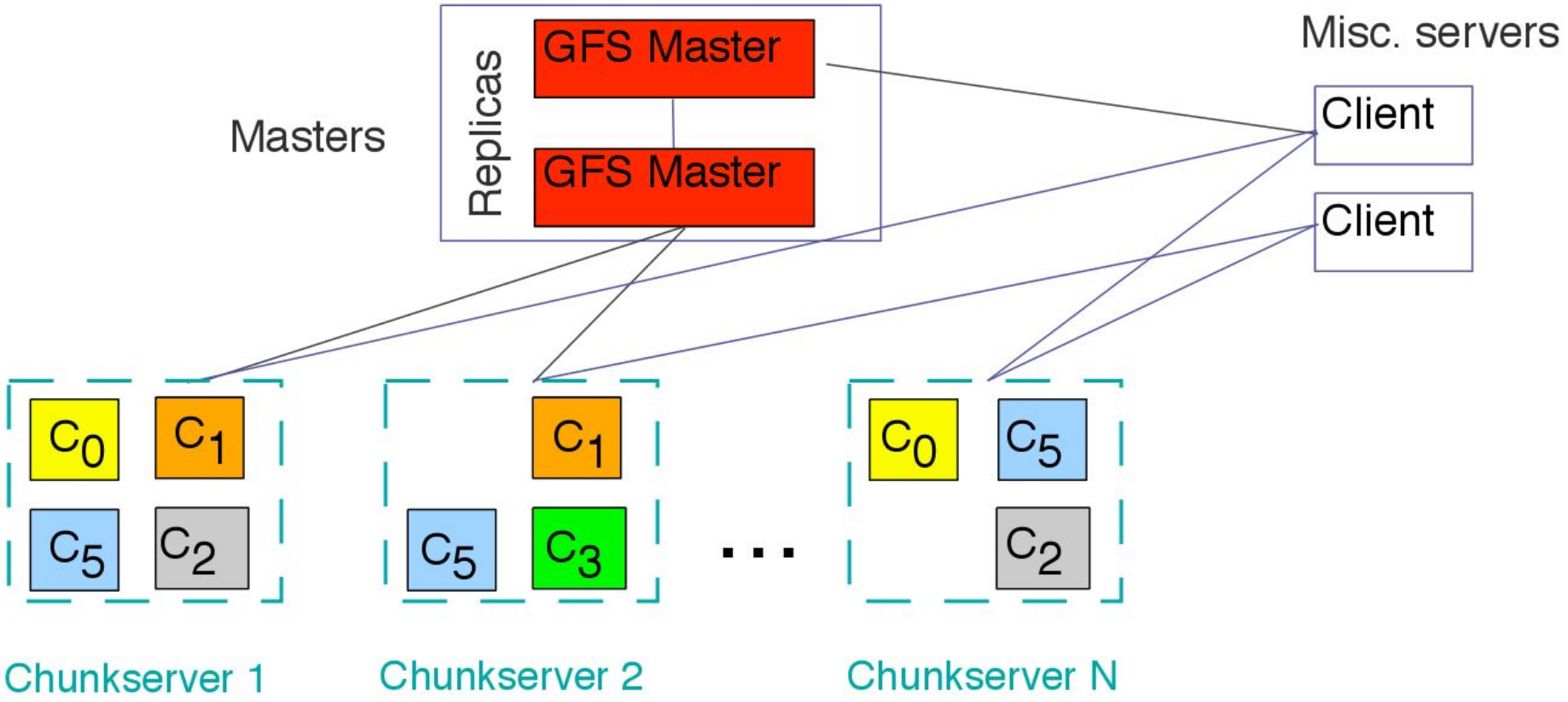
Google的核心技术

Google的十个核心技术，而且可以分为四大类：

- 分布式基础设施：**GFS**、Chubby 和 Protocol Buffer。
- 分布式大规模数据处理：**MapReduce** 和 Sawzall。
- 分布式数据库技术：BigTable 和数据库 Sharding。
- 数据中心优化技术：数据中心高温化、12V电池和服务器整合。

Google的核心技术

- GFS的架构图



Google的核心技术

- GFS的架构图

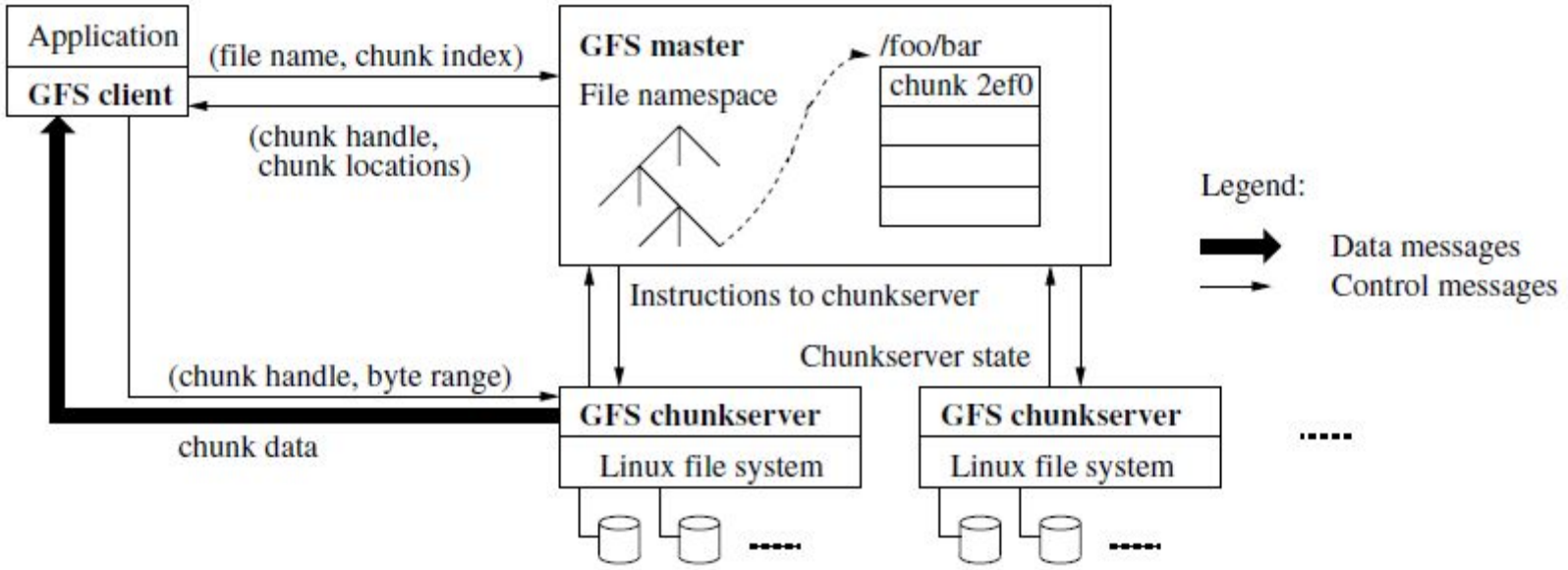


Figure 1: GFS Architecture

Google的核心技术

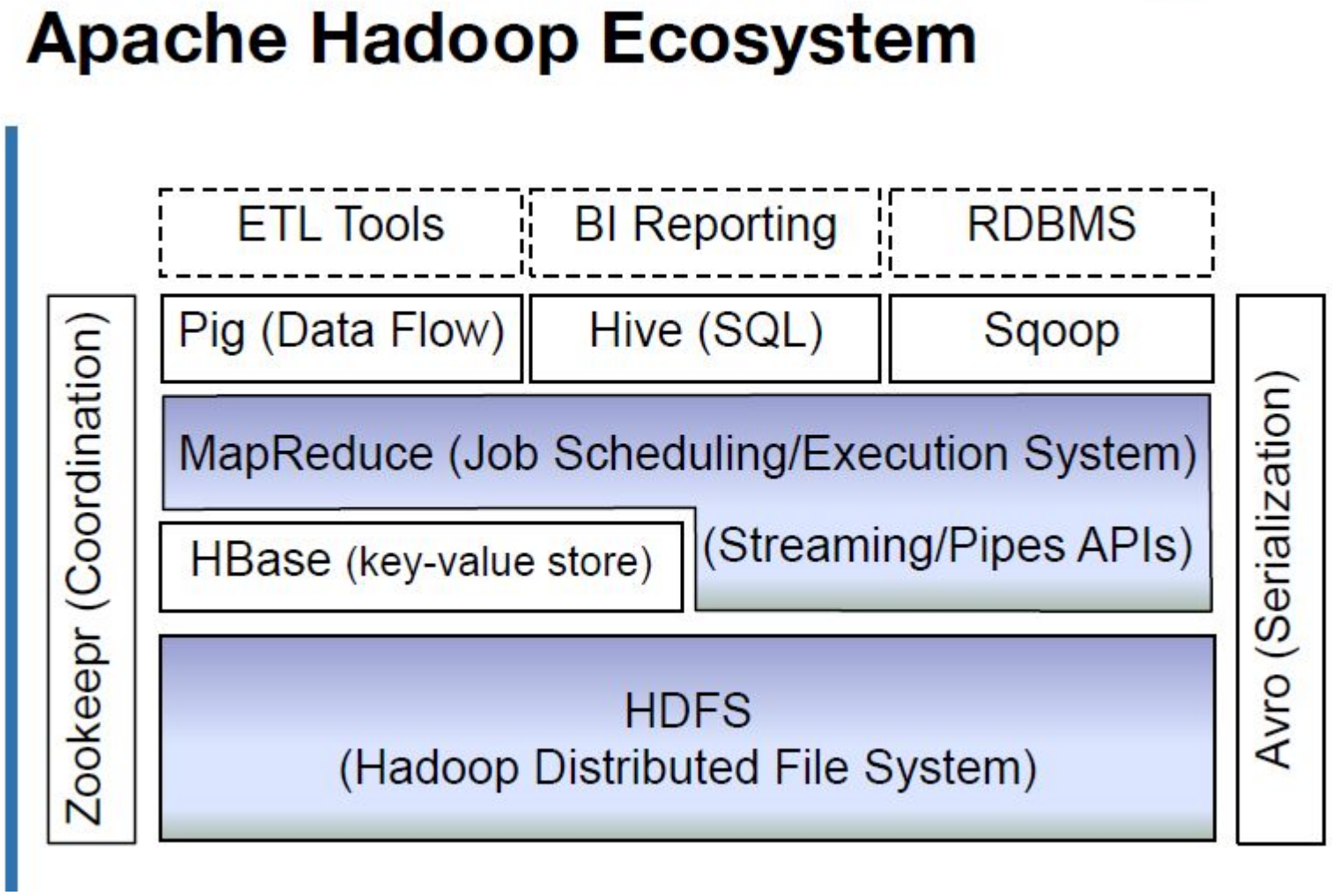
- **分布式大规模数据处理MapReduce**

在Google数据中心会有大规模数据需要处理，比如被网络爬虫（Web Crawler）抓取的大量网页等。由于这些数据很多都是PB级别，导致处理工作不得不尽可能的并行化，而Google为了解决这个问题，引入了 MapReduce这个编程模型，MapReduce是源自函数式语言，主要通过"Map（映射）"和"Reduce（化简）"这两个步骤来并行处理大规 模的数据集。

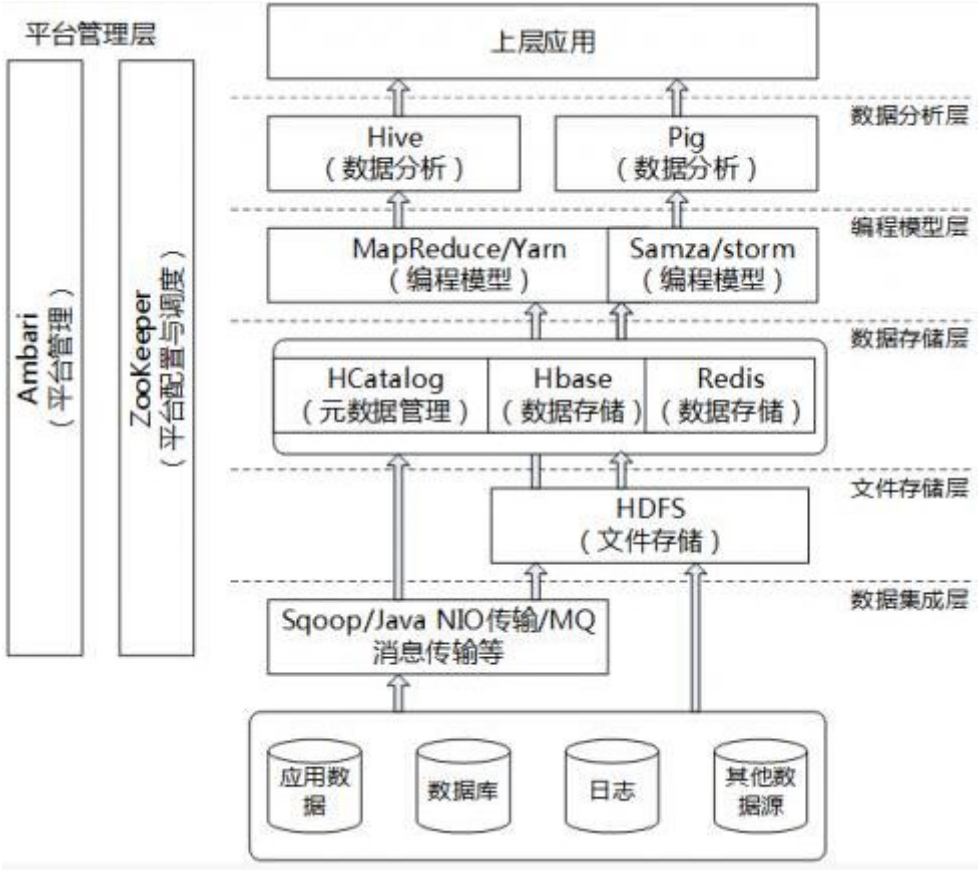
Google VS Hadoop

Google calls it:	Hadoop equivalent:
MapReduce	Hadoop MapReduce
GFS	HDFS
Sawzall	Hive, Pig
BigTable	HBase
Chubby	ZooKeeper

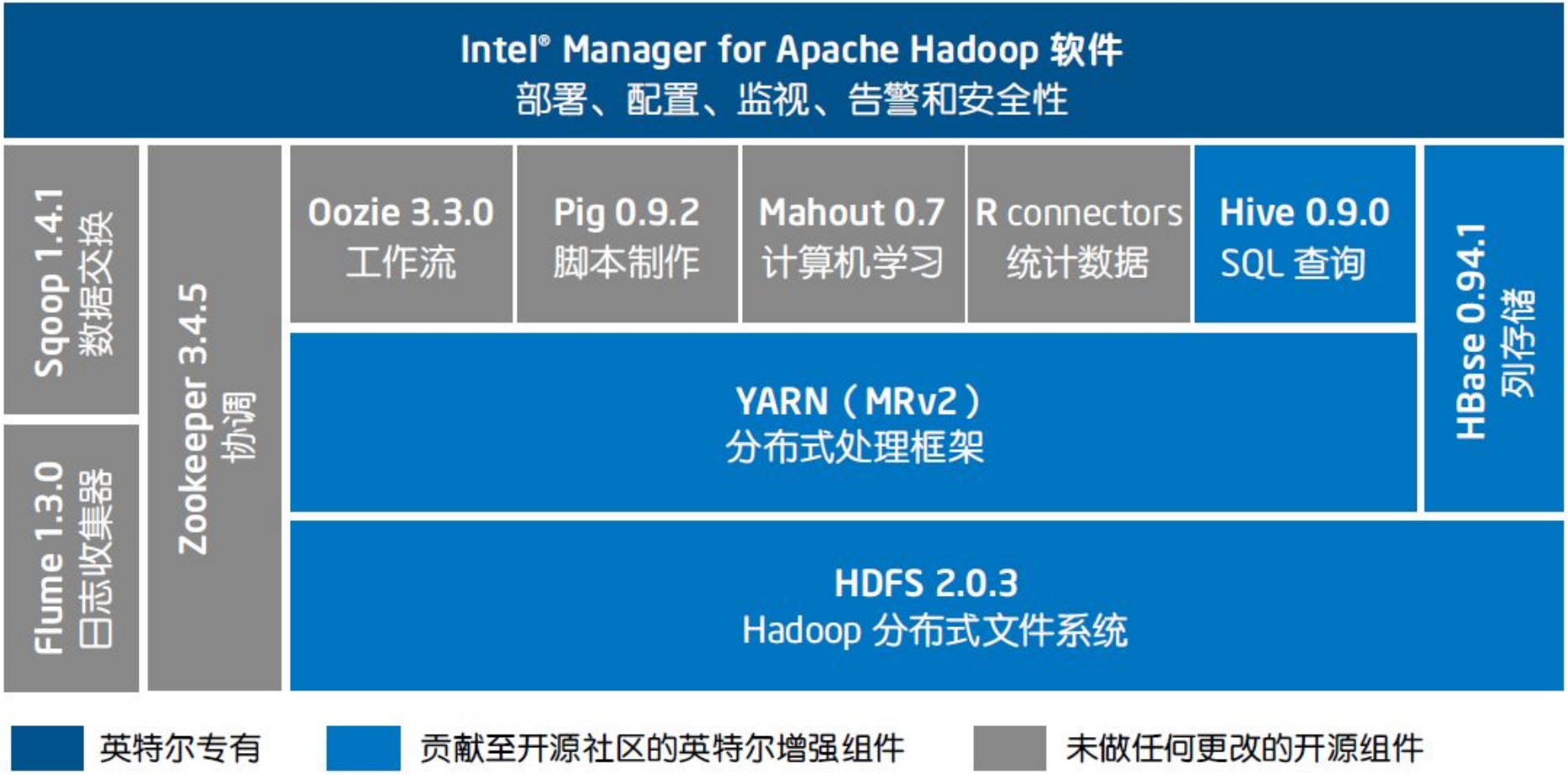
Hadoop生态系统


Apache Hadoop Ecosystem
ETL Tools
BI Reporting
RDBMS
Pig (Data Flow)
Hive (SQL)
Sqoop
MapReduce (Job Scheduling/Execution System)
HBase (key-value store)
(Streaming/Pipes APIs)
HDFS
(Hadoop Distributed File System)
Zookeepr (Coordination)
Avro (Serialization)

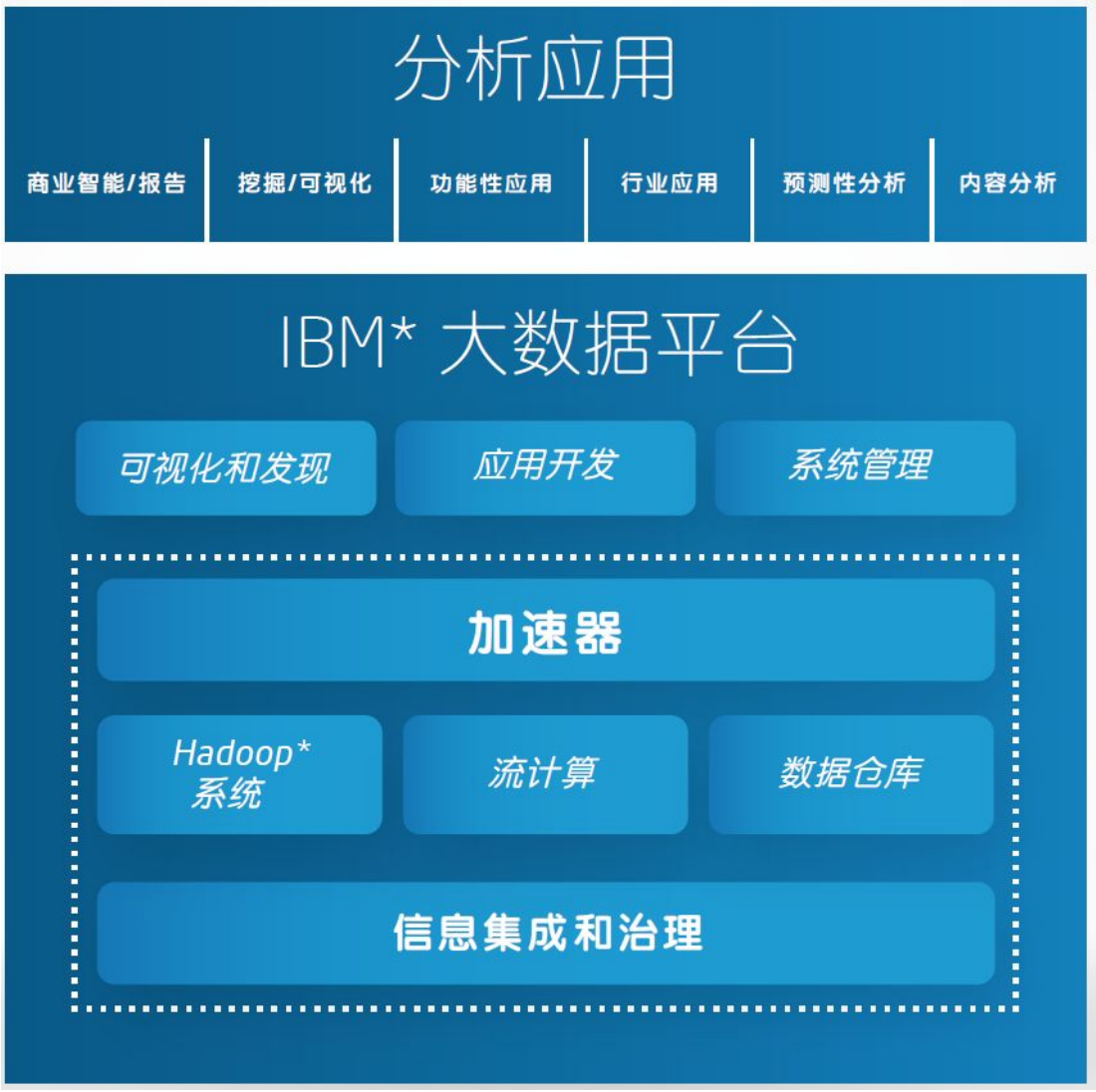
Hadoop

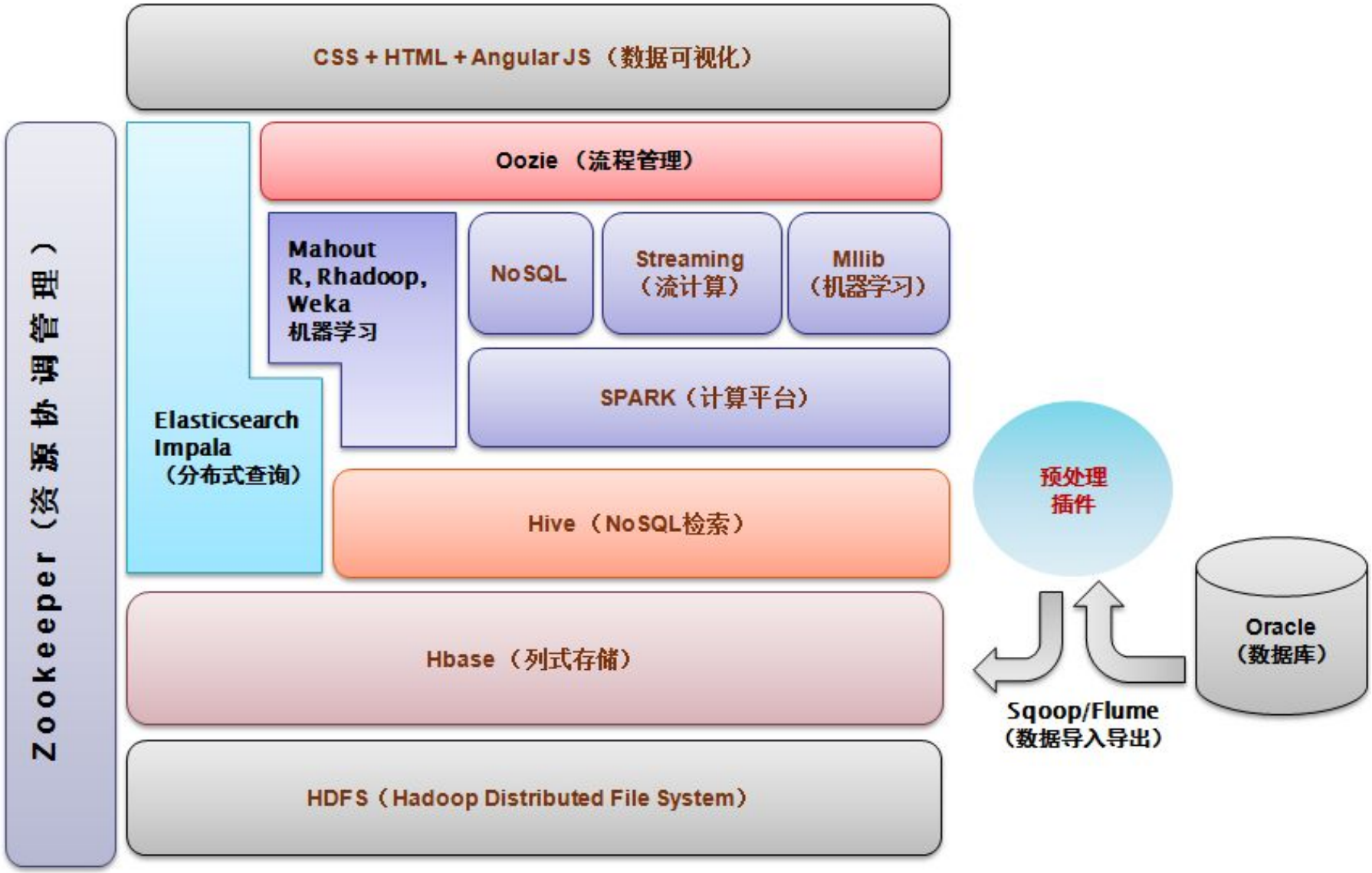

平台管理层
Ambari
（平台管理）
ZooKeeper
（平台配置与调度）
上层应用
Hive
（数据分析）
Pig
（数据分析）
数据分析层
MapReduce/Yarn
（编程模型）
Samza/storm
（编程模型）
编程模型层
HCatalog
（元数据管理）
Hbase
（数据存储）
Redis
（数据存储）
数据存储层
HDFS
（文件存储）
文件存储层
Sqoop/Java NIO传输/MQ
消息传输等
数据集成层
应用数据
数据库
日志
其他数据源

Intel大数据解决方案



IBM大数据解决方案


分析应用
商业智能/报告
挖掘/可视化
功能性应用
行业应用
预测性分析
内容分析
IBM* 大数据平台
可视化和发现
应用开发
系统管理
加速器
Hadoop* 系统
流计算
数据仓库
信息集成和治理


CSS + HTML + Angular JS （数据可视化）
Oozie （流程管理）
Mahout
R, Rhadoop,
Weka
机器学习
NoSQL
Streaming
（流计算）
Mllib
（机器学习）
SPARK（计算平台）
Elasticsearch
Impala
（分布式查询）
Hive （NoSQL检索）
Hbase （列式存储）
HDFS（Hadoop Distributed File System）
Zookeeper（资源协调管理）
预处理
插件
Oracle
（数据库）
Sqoop/Flume
（数据导入导出）

Hadoop的特点

- 扩容能力（Scalable）：能可靠地（reliably）存储和处理千兆字节（PB）数据。
- 成本低（Economical）：可以通过普通机器组成的服务器群来分发以及处理数据。这些服务器群总计可达数千个节点。
- 高效率（Efficient）：通过分发数据，hadoop可以在数据所在的节点上并行地（parallel）处理它们，这使得处理非常的快速。
- 可靠性（Reliable）：hadoop能自动地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（redeploy）计算任务。

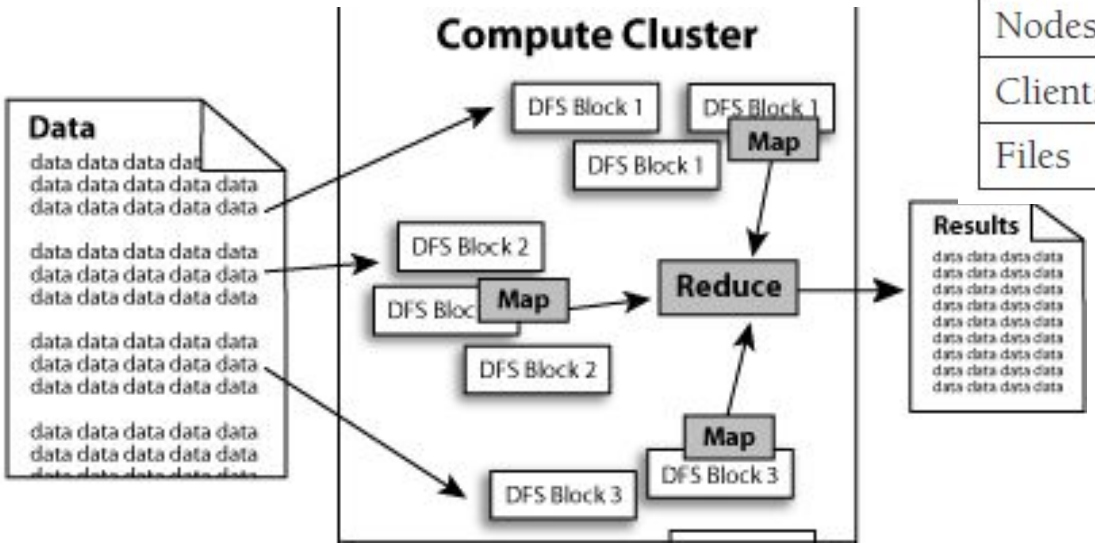
HDFS

- **Hadoop Distributed File System**
  - Hadoop Distributed File System (HDFS) is the primary storage system used by Hadoop applications. HDFS creates **multiple replicas** of data blocks and distributes them on compute nodes throughout a cluster to enable **reliable**, extremely **rapid** computations.

HDFS简介

- HDFS为了做到可靠性（reliability）创建了多份数据块（data blocks）的复制（replicas），并将它们放置在服务器群的计算节点中（compute nodes），MapReduce就可以在它们所在的节点上处理这些数据了

	Target	Deployed
Capacity	10PB	14PB
Nodes	10,000	4000
Clients	100,000	15,000
Files	100,000,000	60,000,000



HDFS能做什么？

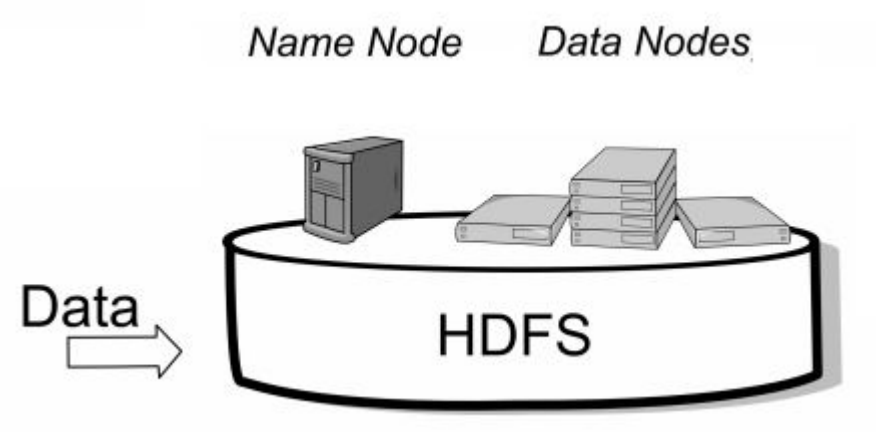
Advantage

- 存储并管理PB级数据
- 处理非结构化数据
- 注重数据处理的吞吐量（latency不敏感）
- 应用模式为：write-once-read-many存取模式

Disadvantage

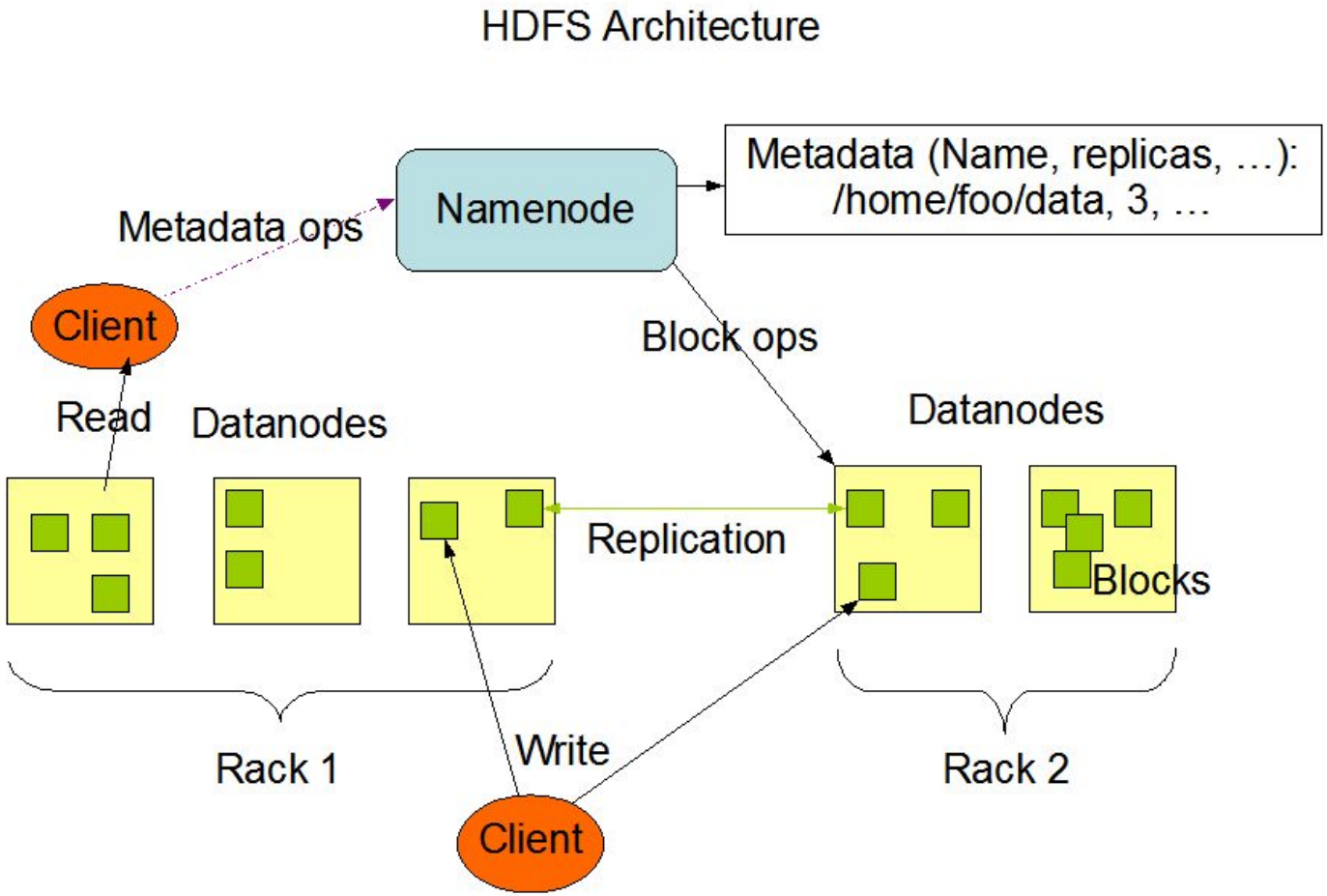
- 存储小文件 (不建议使用)
- 大量的随机读 (不建议使用)
- 需要对文件的修改 (不支持)

HDFS主要组件的功能



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

系统架构


HDFS Architecture
Metadata (Name, replicas, …):
/home/foo/data, 3, …
Namenode
Metadata ops
Client
Block ops
Read
Datanodes
Datanodes
Replication
Blocks
Rack 1
Write
Rack 2
Client

HDFS关键运行机制

--保障可靠性的措施

- 一个名字节点和多个数据节点
- 数据复制（冗余机制）

--存放的位置（机架感知策略）

- 故障检测

--数据节点

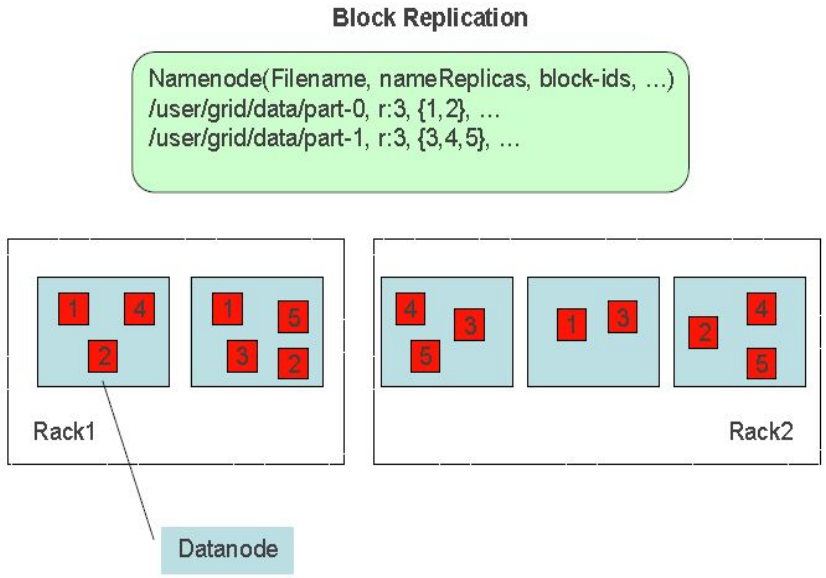
心跳包（检测是否宕机）

块报告（安全模式下检测）

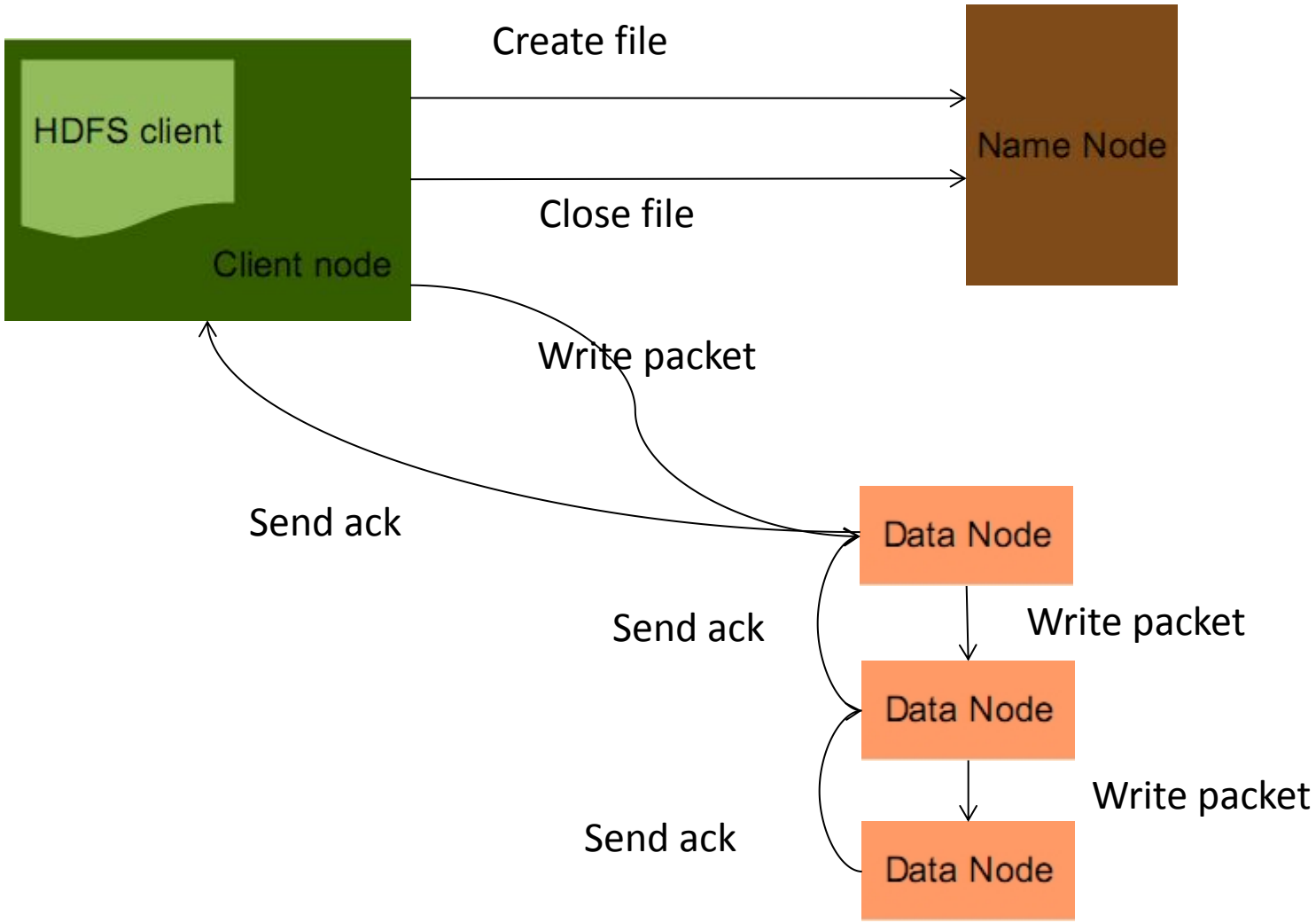
数据完整性检测（校验和比较）

--名字节点（日志文件，镜像文件）

- 空间回收机制

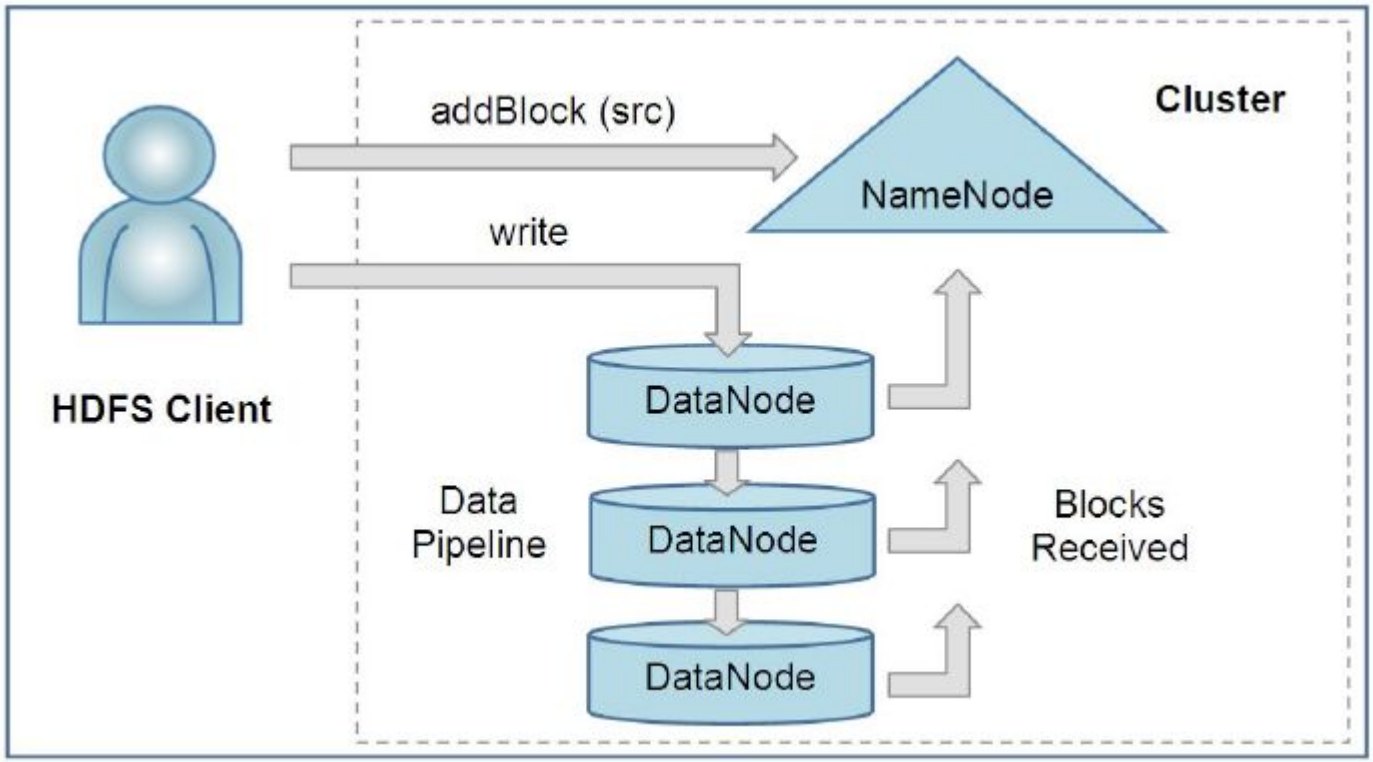


Example： HDFS如何写文件？

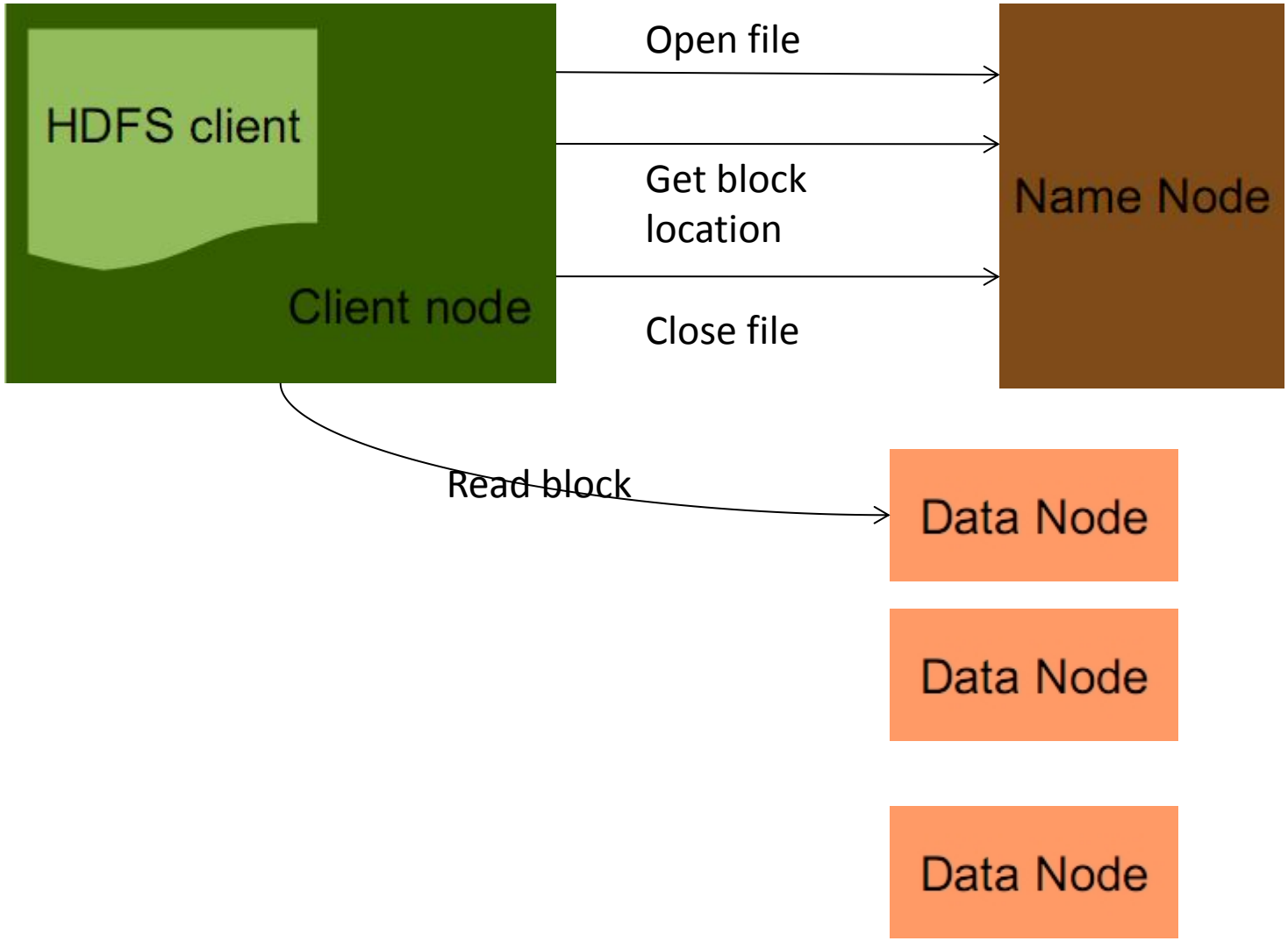


Example： HDFS如何写文件？

- 写一个数据块



Example： HDFS如何读文件？



分布式文件系统-HDFS

- 优点：
  - ➢高容错性
    数据自动保存多个副本
    副本丢失后，自动恢复
  - ➢适合批处理
    移动计算而非数据
    数据位置暴露给计算框架
  - ➢适合大数据处理
    GB、TB、甚至

- **不擅长：**
  - 低延迟与高吞吐率的数据访问（毫秒级）
  - 小文件存取
    占用NameNode大量内存
    寻道时间超过读取时间
  - 并发写入、文件随机修改
    一个文件同一个时间只能有一个写者
    仅支持append

Why MapReduce

并行处理为什么我们不能使用数据库加上更多磁盘来做大规模的批量分析？为什么我们需要MapReduce？

这个问题的答案来自于磁盘驱动器的另一个发展趋势：

寻址时间的提高速度远远慢于传输速率的提高速度。

寻址就是将磁头移动到特定位置进行读写操作的工序。

它的特点是磁盘操作有延迟，而传输速率对应于磁盘的带宽。

关系型数据库和MapReduce的比较：

	传统关系型数据库	MapReduce
数据大小	GB	PB
访问	交互型和批处理	批处理
更新	多次读写	一次写入多次读取
结构	静态模式	动态模式
集成度	高	低
伸缩性	非线性	线性

MPI vs MapReduce

大规模数据分析

MPI ：

设计前提： 输入数据一般不会多于 10TB ，计算很密集，计算相关性很强，硬件不容易坏。

特　　点： 适用于数据相关性强，迭代次数多的计算，不适合处理过大规模数据，节点数不超过百台，节点失效会影响全局。

MapReduce ：

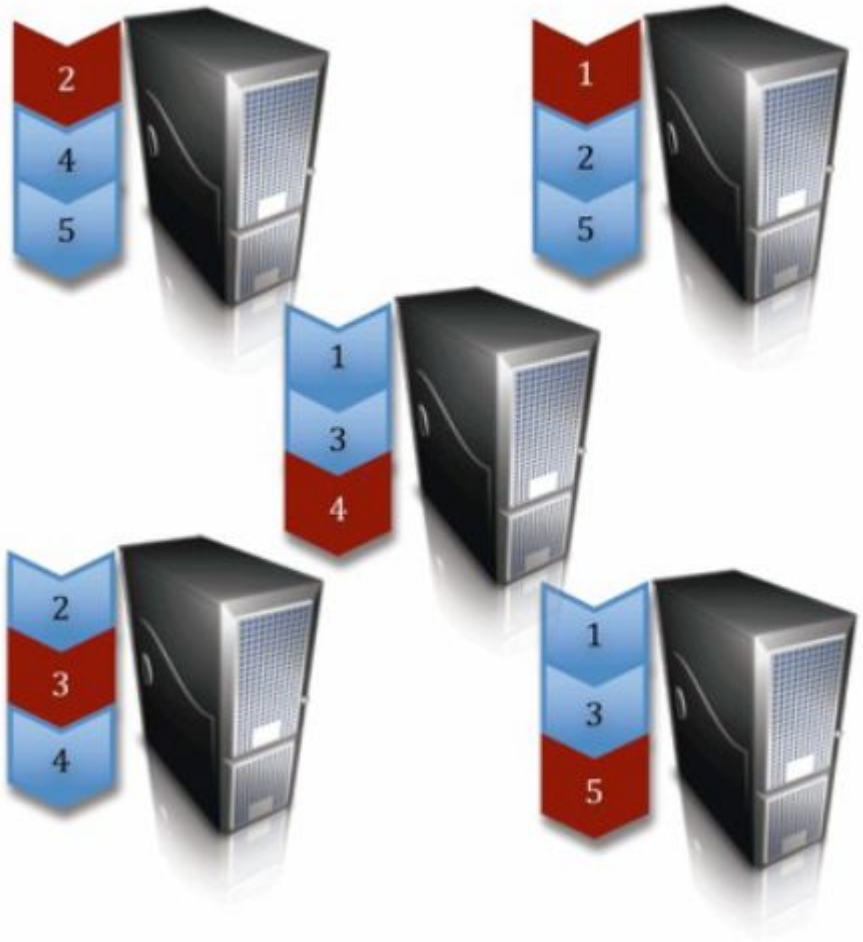
设计前提： 输入数据会超过 100TB ，数据全局相关性弱，硬件是容易坏的。

特　　点： 适用于大规模数据处理，节点规模可以达到数千台，节点失效对系统无影响。

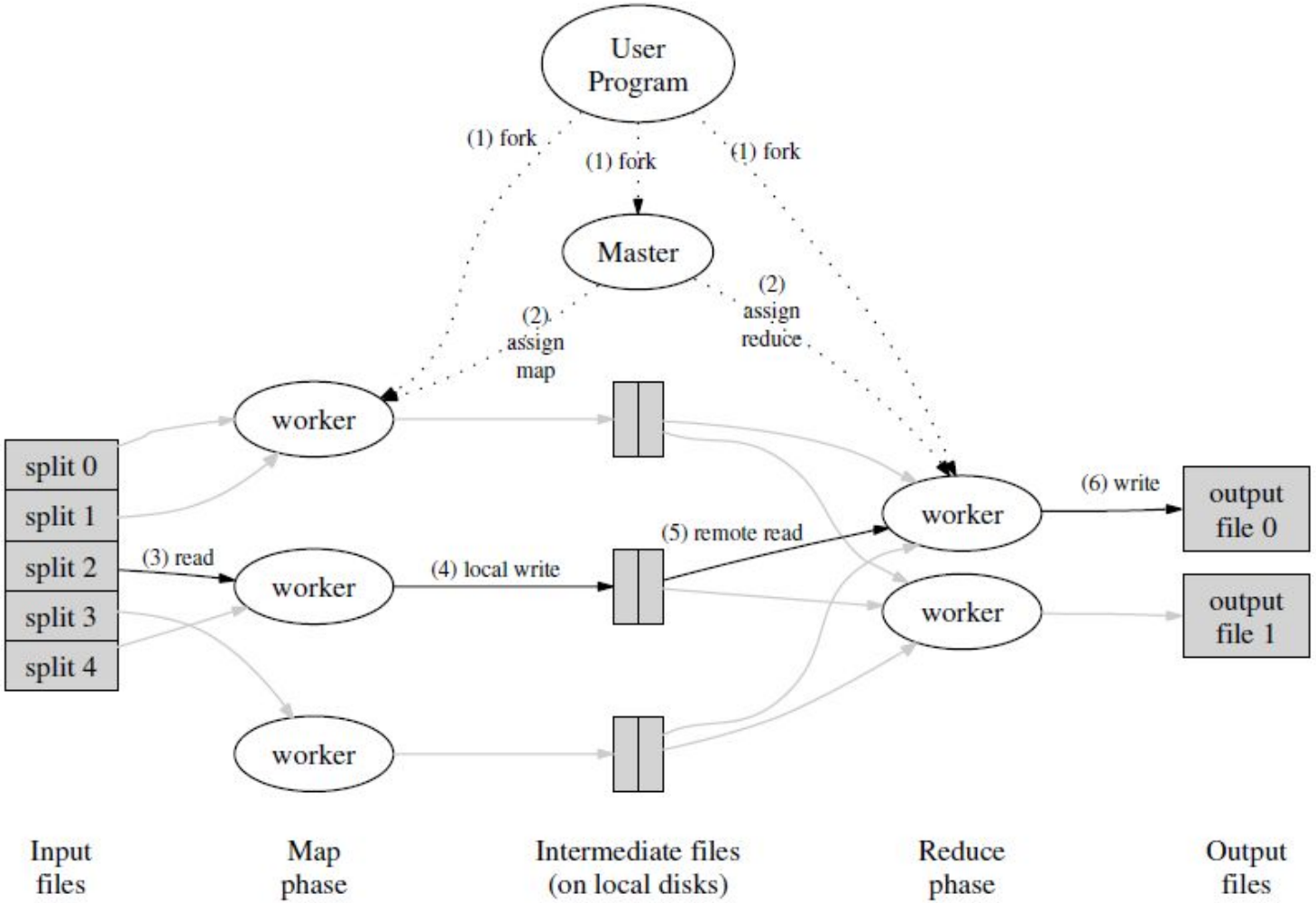
分而治之（Divide and Conquer）

MapReduce: Distributed Processing

Hadoop takes advantage of HDFS' data distribution strategy to push work out to many nodes in a cluster. This allows analyses to run in parallel and eliminates the bottlenecks imposed by monolithic storage systems.



MapReduce的运行机制：

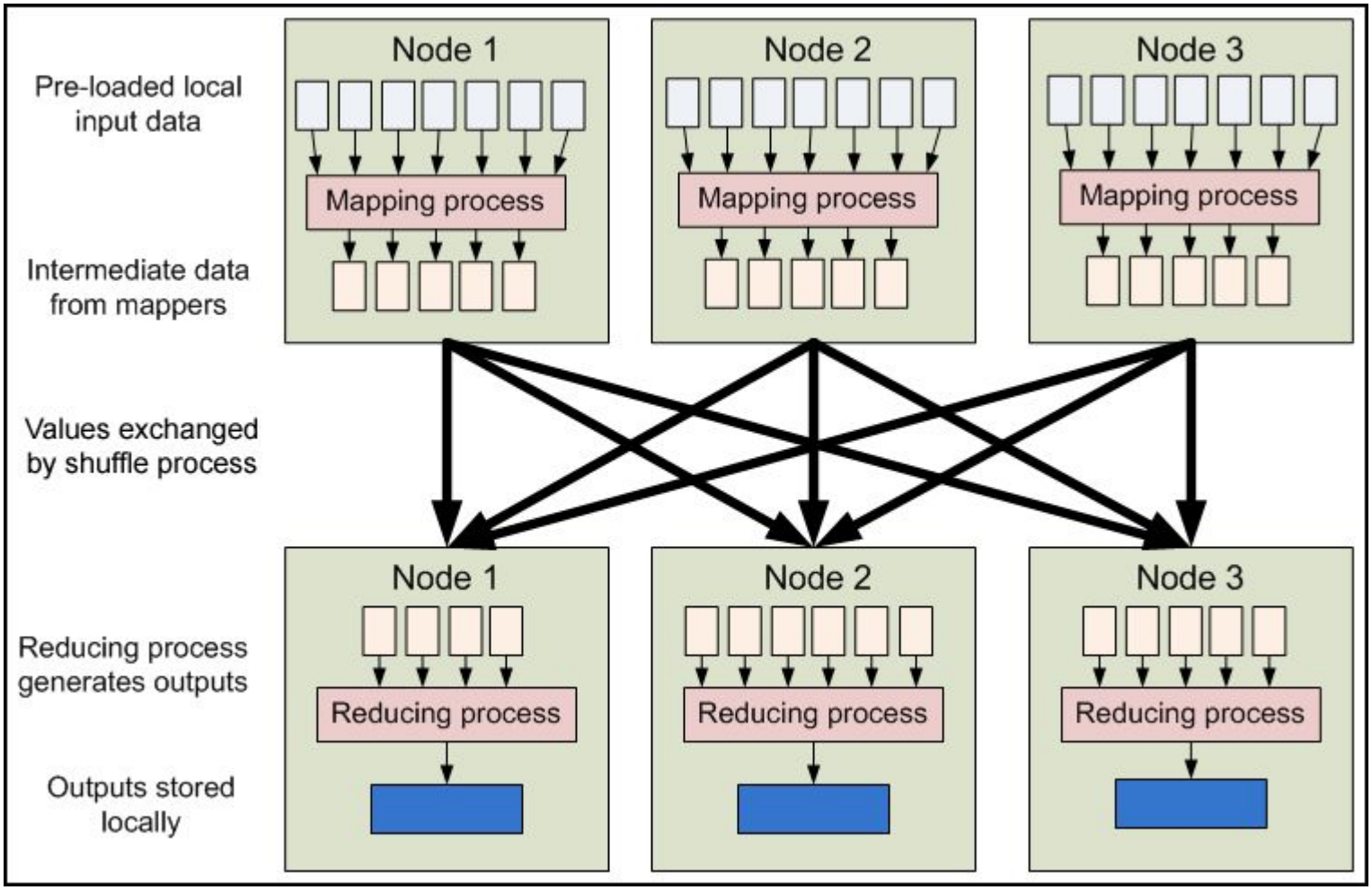


MapReduce

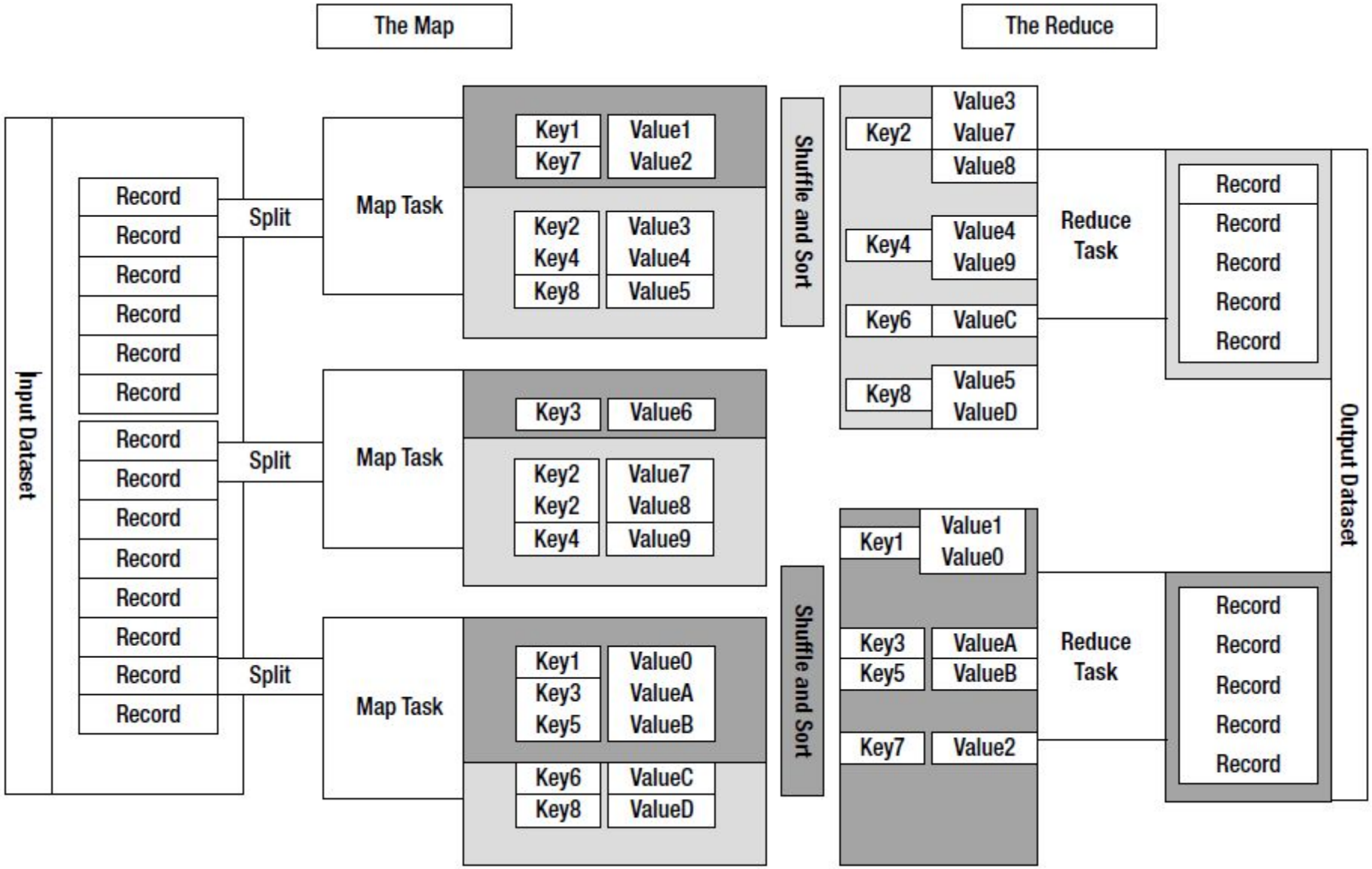
MapReduce框架

- 编程模型-Programming Model
  - 原理:利用一个输入key/value pair 集合来产生一个输出的key/value pair 集合。
  - Map函数:接受一个输入的key/value pair 值，然后产生一个中间key/value pair 值的集合。
  - Reduce函数:接受一个中间key值和相关的一个value值的集合，合并这些value 值。

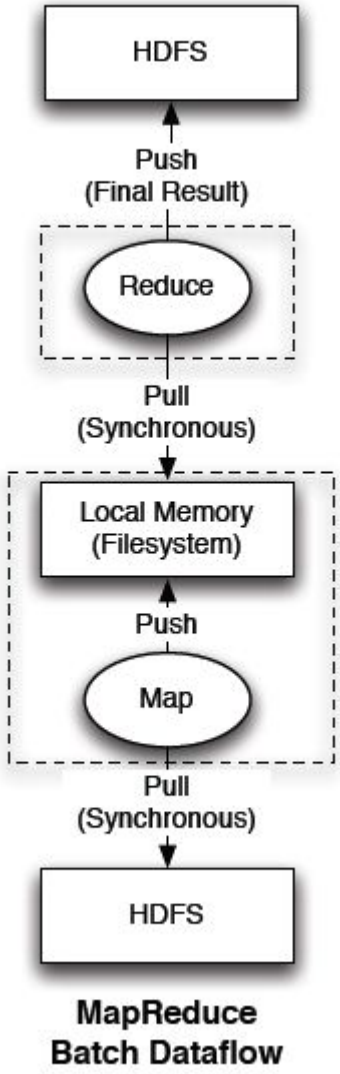
MapReduce


Pre-loaded local input data
Intermediate data from mappers
Values exchanged by shuffle process
Reducing process generates outputs
Outputs stored locally
Node 1
Node 2
Node 3
Mapping process
Mapping process
Mapping process
Node 1
Node 2
Node 3
Reducing process
Reducing process
Reducing process

MapReduce


The Map
The Reduce
Input Dataset
Record
Split
Map Task
Key1
Value1
Key7
Value2
Key2
Key4
Key8
Value3
Value4
Value5
Shuffle and Sort
Value3
Key2
Value7
Value8
Key4
Value4
Value9
Key6
ValueC
Key8
Value5
ValueD
Reduce Task
Output Dataset
Key3
Value6
Value7
Value8
Value9
Value0
ValueA
ValueB
ValueD

MapReduce



MapReduce
Batch Dataflow

MapReduce

- 你所需要做的是 实现map()和reduce()函数，剩下的框架完成。

MapReduce编程模型

- 借鉴了函数式编程方式（functional programming）
- 用户只需要实现两个函数接口：
  - map (in_key, in_value) ->
    (out_key, intermediate_value) list
  - reduce (out_key, intermediate_value list) ->
    out_value list

MapReduce示例

示例：WordCount

- 源数据
  - Page 1:
    - the weather is good
  - Page 2:
    - today is good
  - Page 3:
    - good weather is good

MapReduce示例

map 输出

- Worker 1:
  - (the 1), (weather 1), (is 1), (good 1).
- Worker 2:
  - (today 1), (is 1), (good 1).
- Worker 3:
  - (good 1), (weather 1), (is 1), (good 1).

MapReduce示例

reduce 的输入

- Worker 1:
  - (the 1)
- Worker 2:
  - (is 1), (is 1), (is 1)
- Worker 3:
  - (weather 1), (weather 1)
- Worker 4:
  - (today 1)
- Worker 5:
  - (good 1), (good 1), (good 1), (good 1)

MapReduce示例

reduce输出

- Worker 1:
  - (the 1)
- Worker 2:
  - (is 3)
- Worker 3:
  - (weather 2)
- Worker 4:
  - (today 1)
- Worker 5:
  - (good 4)

Hive

- Hive是基于Hadoop的一个数据仓库工具，可以将结构化的数据文件映射为一张数据库表，并提供完整的sql查询功能，可以将sql语句转换为MapReduce任务进行运行。其优点是学习成本低，可以通过类SQL语句快速实现简单的MapReduce统计，不必开发专门的MapReduce应用，十分适合数据仓库的统计分析。

Pig

Pig是一个基于Hadoop的大规模数据分析平台，它提供的SQL-LIKE语言叫Pig Latin，该语言的编译器会把类SQL的数据分析请求转换为一系列经过优化处理的MapReduce运算。Pig为复杂的海量数据并行计算提供了一个简单的操作和编程接口。

Mahout

- 提供一些可扩展的机器学习领域经典算法的实现，旨在帮助开发人员更加方便快捷地创建智能应用程序。Mahout包含许多实现，包括聚类、分类、推荐过滤、频繁子项挖掘。此外，通过使用 Apache Hadoop 库，Mahout 可以有效地扩展到云中。

Spark

- Spark是发源于美国加州大学伯克利分校AMPLab的集群计算平台。它立足于内存计算，从多迭代批量处理出发，兼收并蓄数据仓库、流处理和图计算等多种计算范式，是罕见的全能选手。
- 具体特点概括为“轻、快、灵和巧”。

MapReduce扩展性

- 现在的架构是简单的Master-Slave模型
  集群的规模达到4000
- Mapreduce面临的瓶颈

（1）可扩展性，内存消耗，线程模型，可靠性和性能的几个缺陷

（2）各个模块的紧耦合使得在现有设计的基础上继续改进变得举步维艰。

（3）从操作的角度，任何轻微的或修复Bug带来的巨大改动都会让Hadoop MapReduce强制进行全系统的升级。

- 下一代MapReduce

（1）新架构的主要思想是把原来JobTracker的中的资源管理和任务调度功能进行拆分，引入层次化的管理

（2）更细粒度地控制CPU，内存，磁盘，网络这些资源。

MapReduce：一个重大的[illegible]

Database column的几个数据库大牛写的，简要的介绍了MapReduce以及将其与现代数据库管理系统进行了对比，并指出了一些不足之处。

- MapReduce可能在某些特定类型的通用计算上是个不错的想法，但是对于数据库社区来说：

1. 从大规模数据应用程序模型来说是一个巨大的倒退。
2. 不是一个最优实现，因为它使用蛮力来代替索引。
3. 一点都不新奇，它只是实现了一个特定的25年前就有的众所周知的技术。
4. 失去了大部分目前数据库管理系统的特性。
5. 不能兼容所有目前数据库管理系统用户已经依赖的工具。

MapReduce：一个重大的[illegible]

MapReduce基础出发点是很易懂的。它由称为map和reduce的两部分用户程 序组成，然后利用框架在计算机集群上面根据需求运行多个程序实例来处理各个子任务，然后再对结果进行归并。

五点：

1. MapReduce是一个数据库存取的退步

- 做为一个数据处理模型，MapReduce呈现出了一个巨大的退步。数据库社区从 IBM在1968年第一次发布IMS以来的四十年中学到了以下三个经验：
  - *** 结构描述是好的。**
  - *** 将结构描述从程序中分离是好的**
  - *** 高阶的访问语言是好的**

MapReduce：一个重大的[illegible]

2. MapReduce是一个粗燥的实现

- MapReduce没有索引，理所当然的只能使用蛮力来作为处理选项。而不管索引在当前情况下是否是一个最好的访问机制。
- MapReduce同时存在很多底层的实现问题，特别是数据交换和数据斜交的情况。

3. MapReduce并不新奇

- MapReduce所使用的技术至少是20年前的。

MapReduce：一个重大的倒

4. MapReduce失去了很多特性

- 所有下面的特性都被现在的数据库管理系统提供了，而MapReduce没有：

  * **批量导入** —— 将输入数据转化成想要的格式并加载到数据库中
  * **索引** —— 如上文所述
  * **更新** —— 改变数据集中的数据
  * **事务** —— 支持并行更新以及从失败的更新中恢复
  * **完善的约束** —— 防止垃圾数据添加到数据集
  * **完善的引用** —— 类似FK，防止垃圾数据的存在
  * **视图** —— 底层逻辑数据描述可以改变但不需要重写程序

- 简单的说来，MapReduce只提供了现在数据库管理统的函数性功能。

MapReduce：一个重大的[illegible]

5. MapReduce与现有的数据库管理系统工具不兼容

- 一个现代的SQL数据库管理系统都拥有如下可用的工具：

  *** 报表** —— (比如水晶报表) 将数据友好的展示给人

  *** 商业智能工具** —— (比如Business Objects or Cognos)允许在数据仓库中进行特定查询

  *** 数据挖掘工具** —— (比如Oracle Data Mining)允许用户在大数据集中发现数据规律

  *** 复制工具** —— 允许用户在不同的数据库中进行复制传输

  *** 数据库设计工具** —— 帮助用户构建数据库

- MapReduce不能使用这些工具，同时它也没有自己的工具。直到它能与SQL兼容或者有人编写了这些工具，MapReduce仍然在端到端的任务中显 得十分困难。

Google新的索引系统

Google放弃MapReduce的原因：

- “MapReduce仅仅是一个批处理操作方式，”，“一般来说你不能启动下一阶段的命令操作，直到你完成第一项操作。”
- 它并不能为谷歌提供它所想要的索引速度，特别是随着实时检索时代的到来，谷歌需要的是在几秒内刷新索引内容，而非8小时。
- 麻省理工学院的数据库专家Mike Stonebraker认为，MapReduce的计算方法对于实时计算来说是很不合适的，是过时的。
- “MapReduce就像是游击队员而非正规军”，“如果你想基于Mapreduces建立分布式文件处理系统，如果你想实现更多的操作命令，那么必然会有错误发生。况且你并不能缩短处理的时间，这是Google选择放弃Mapreduces的原因。”

参考资料

- Book

（1）Hadoop The Definitive Guide

（2）Pro Hadoop

（3）Hadoop in action

（4）Hadoop in Practice 正在进行中

（5）Data-Intensive Text Processing with MapReduce

参考资料

- 网站

(1)http://www.cloudera.com/blog/

(2)http://developer.yahoo.com/blogs/hadoop/

(3)http://wiki.apache.org/hadoop/

(4) http://subject.csdn.net/hadoop/

(5) http://bbs.hadoopor.com/

(6) http://www.slideshare.net/

(7) http://code.google.com/intl/zh-CN/edu/parallel/index.html#_distrib_storage

(8) http://www.tbdata.org/

……

参考资料

- Paper

(1) The Google File System

(2) MapReduce: Simplied Data Processing on Large Clusters

(3) The Hadoop Distributed File System

(4) HDFS scalability: the limits to growth

(5) Case Study GFS: Evolution on Fast-forward