hadoop简介

目 录

1. 什么是Hadoop

1.1 Hadoop的特点

- Hadoop是Apache基金会下的一个开源分布式计算平台， 以 Hadoop分布式文件系统HDFS （Hadoop Distributed File System）和MapReduce分布式计算框架为核心，为用户提供了底层细节透明的分布式基础设施。
- HDFS的高容错性、高伸缩性等优点，允许用户将Hadoop部署在廉价的硬件上， 构建分布式系统。
- MapReduce分布式计算框架则允许用户在不了解分布式系统底层细节的情况下开发并行、分布式的应用程序， 充分利用大规模的计算资源， 解决传统高性能单机无法解决的大数据处理问题。
- Apache Hadoop是目前分析海量数据的首选工具。

1.2 Hadoop 的发展史

- 谈到Hadoop的历史，就不得不提到Lucene和Nutch。 Hadoop开始时是Nutch的一个子项目，而Nutch又是Apache Lucene的子项目。这3个项目都是由Doug Cutting创立，每个项目在逻辑上都是前一个项目的演进。
- Lucene是引擎开发工具包， 提供了一个纯Java的高性能全文索引， 它可以方便地嵌入各种实际应用中实现全文搜索/索引功能。
- Nutch项目开始于2002年，是以Lucene为基础实现的搜索引擎应用。Lucene为Nutch提供了文本搜索和索引的API，Nutch不光有搜索功能， 还有数据抓取的功能。
- 很快，Doug Cutting和Mike Calarella就意识到，他们的架构无法扩展以支持拥有数十亿网页的网络。
- Google在2003年的ACM SOSP会议上发表的Google分布式文件系统 （简称GFS）的论文。GFS或类似的系统可以解决他们在网络抓取和索引过程中产生的大量文件存储需求。于是，在2004年，他们开始写GFS的一个开源实现，即Nutch分布式文件系统（NDFS）。

- 2004年在OSDI会议上， Google发表了论文，向全世界介绍了MapReduce。

- 2005年初，Nutch的开发者在Nutch上有了一个可工作的MapReduce应用，到当年的年中，所有主要的 Nutch算法被迁移到 MapReduce和 NDFS上。

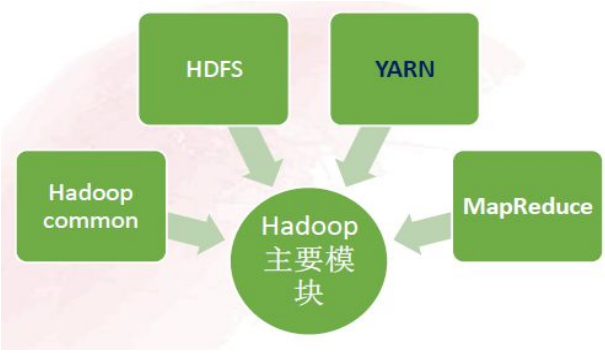
- 从Nutch0.8.0开始， Doug Cutting等人将其中实现的NDFS和MapReduce剥离出来成立了一个新的开源项目 Hadoop。同时， Nutch0.8.0在架构上完全构建在Hadoop的基础之上了。这个时候，已经是2006年2月，大约在同一时间， DougCutting加入雅虎，Yahoo投入了专门的团队和资源将H adoop发展成一个可在网络上运行的系统。

- 2008年1月，Hadoop已成为Apache顶级项目，证明它是成功的。通过这次机会，Hadoop成功地被雅虎之外的很多公司应用，如Facebook、纽约时报等。特别是纽约时报，它使用运行在亚马逊的 EC2云计算上的 Hadoop， 将4TB的报纸扫描文档压缩，转换为用于Web的PDF文档，这个过程历时不到24小时，使用100台机器运行，这成为Hadoop一个良好的宣传范例。

- 2008年2月，雅虎宣布其索引网页的生产系统采用了在10000多个核的 Linux集群上运行的Hadoop。Hadoop真正达到了万维网的规模。2008年4月，在一个900节点的Hadoop集群上，雅虎的研究人员运行1TB的Jim Gray基准排序， 只用了209秒，而到了2009年4月，在一个1400节点的集群上对500GB数据进行排序，只用了59秒，这显示了Hadoop强大的计算能力。

- 2008年开始，Hadoop迈向主流，开始了它的爆发式发展，出现了大量的相关项目，如2008年的HBase、ZooKeeper和Mahout，2009年的Pig、 Hive等。同时，还出现了像Cloudera和 Hortonworks这样的专注于Hadoop的公司。

1.3 Hadoop的核心组件



- Hadoop Common
  支持其他模块的常用工具集，主要包括系统配置工具Configuration、序列化机制、远程过程调用RPC、数据压缩与解压缩、Hadoop抽象文件系统FileSystem，为在通用硬件上搭建云计算环境提供基础服务，并为运行在该平台上的软件开发提供了所需的API。

- Hadoop Distributed File System(HDFS)
  一个高扩展、高容错、高可靠、高吞吐量，运行在低成本的通用计算机上的分布式文件系统，是GFS的开源实现。

- HadoopYARN
  一个作业调度和集群资源管理框架。

- HadoopMapReduce
  并行处理大数据集的编程框架，是Google MapReduce的开源实现。

- Hadoop Common:
  - Hadoop配置信息处理模块
    - Configuration类---处理配置信息（附配置文件）
    - Configurable接口---类的可配置性

  - Hadoop序列化与反序列化机制
    - HadoopWritable机制（附简单实例，工厂方法）
      - void write(DataOutput out)
      - void readFields(DataInput in)
    - 其他序列化框架
      - Avro数据序列化系统， 用于支持大批量数据交换的应用

- Thrift一个可伸缩、跨语言的服务开发框架，由Facebook贡
- GoogleProtocol

- 压缩框架（简单工厂/工厂方法，抽象工厂）
  - 压缩广泛应用于海量数据处理中，对数据文件进行压缩，可以有效减少存储文件所需要的空间，并加快数据在网络上或者到磁盘上的传输速度。在Hadoop中，压缩应用于文件存储的 Map阶段到Reduce阶段的数据交换（需要打开相关的选项）等情景。

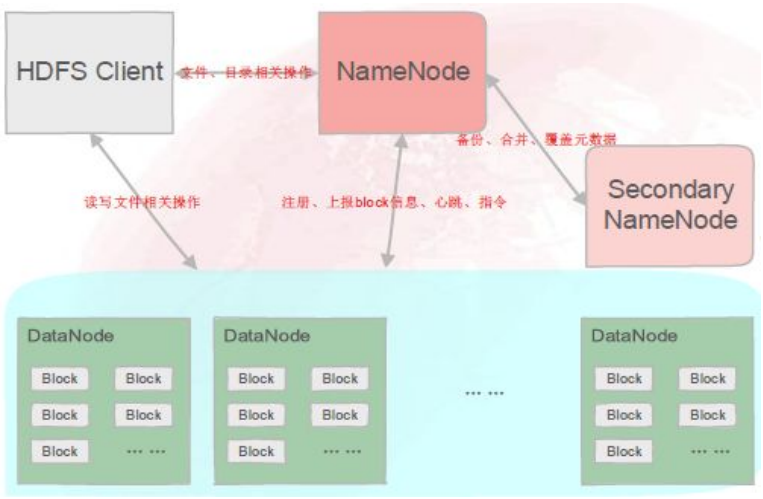
- Hadoop进程间通信IPC（代理模式）
  - 作为典型的分布式系统， hadoop中各个实体间存在着大量的交互，远程过程调用让用户可以像调用本地方法一样调用另外一个应用程序提供的服务，而不必设计开发相关的信息发送、处理和接受等具体代码，是一种重要的分布式计算技术，他提高了程序的互操作性，在Hadoop的实现中得到了广泛的应用。
  - Hadoop进程间通信机制结合数据输出流和数据输入流的 Writable序列化机制，以及可能使用到的压缩框架，形成了Hadoop自有的简洁的、 低消耗的远程过程调用机制。

- Hadoop抽象文件系统FileSystem（简单工厂，享元模式，适配器模式，多线程DCL，钩子线程）
  - 为了提供对不同数据访问的一致接口，Hadoop借鉴了 Linux虚拟文件系统的概念，引入了 Hadoop抽象文件系统，并在Hadoop抽象文件系统的基础上，提供了大量的具体文件系统的实现，满足构建于 Hadoop上应用的各种数据访问需求。
  - 抽象文件系统方法分为两部分：
    - 处理文件和目录相关的操作。
    - 读写文件数据相关操作。
  - 文件系统的缓存CACHE
    - 打开一个文件系统是一个比较耗费资源的操作，如打开HDFS，需要和NameNode节点建立IPC通信，所以，共享文件系统是一个不错的优化。
    - 但是，由于文件系统间是相互共享的，应用不小心关闭共享的文件系统将会影响其他使用者。
    - 可配置的${fs.%s.impl.disable.cache}文件共享。
    - FileSystem.newInstance()返回一个被CACHE管理且不被共享的具体文件系统。
    - FileSystem.CACHE提供的清理机制（钩子线程）。

- **HDFS: HadoopDistributed FileSystem**

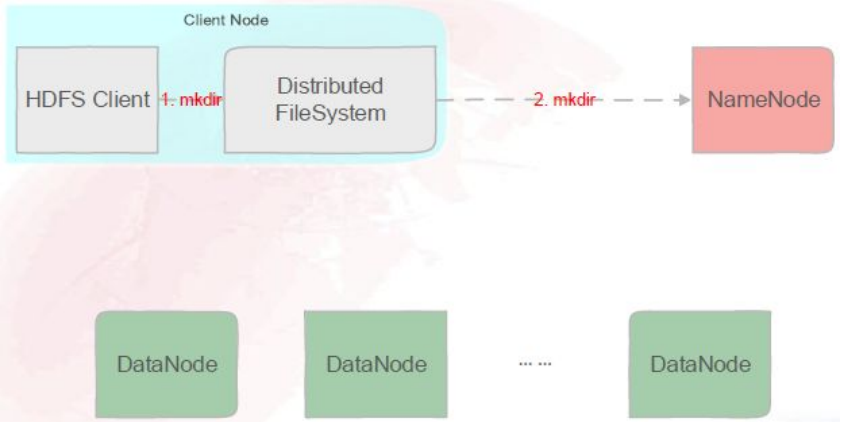
  - 一个分布式存储系统
  - GoogleGFS的开源实现
  - 数据存储采用 master/slave架构模式，主要由HDFS、Client、NameNode，SecondaryNameNode和DataNode组成

- HDFS:体系结构示意图

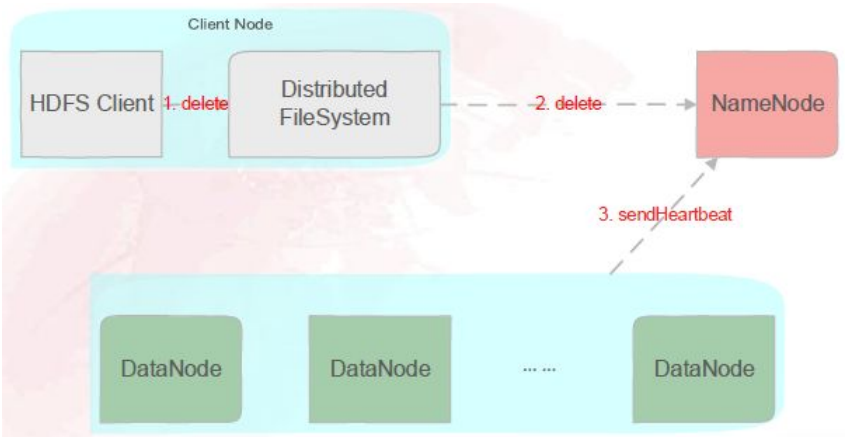


- NameNode负责存储并维的元数据信息，包括命名空间、访问控制信息、文件和Block的映射信息、以及Block的位置信息，对外提供创建、打开、删除和重命名文件 或目录的服务。
- Data Node负责存储实际文件数据，处理客户端对数据的读写请求。DataNode定期向NameNode发送心跳信息，NameNode通过响应心跳信息来给DataNode传达命令。客户端读写数据时，先与NameNode交互获取元数据信息，然后和DataNode通信读取或写入实际数据。
- SecondaryNameNode辅助NameNode完成元数据信息的合并。

- HDFS:客户端处理文件和目录相关的操作mkdir



- HDFS：客户端处理文件和目录相关的操作delete

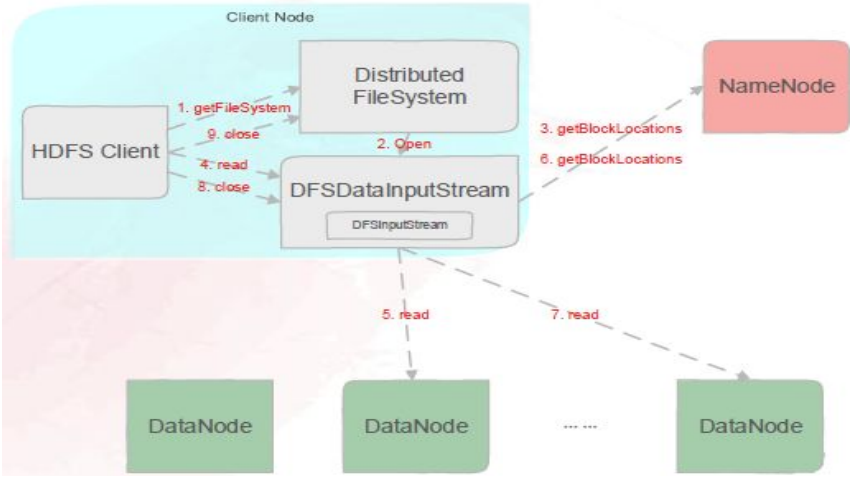


删除步骤:

- NameNode标记删除的数据块，记录delete操作持久化到日志， 不会联系DataNode进行立即删除

- 当DataNode向NameNode发送心跳时，在心跳的应答里，NameNode

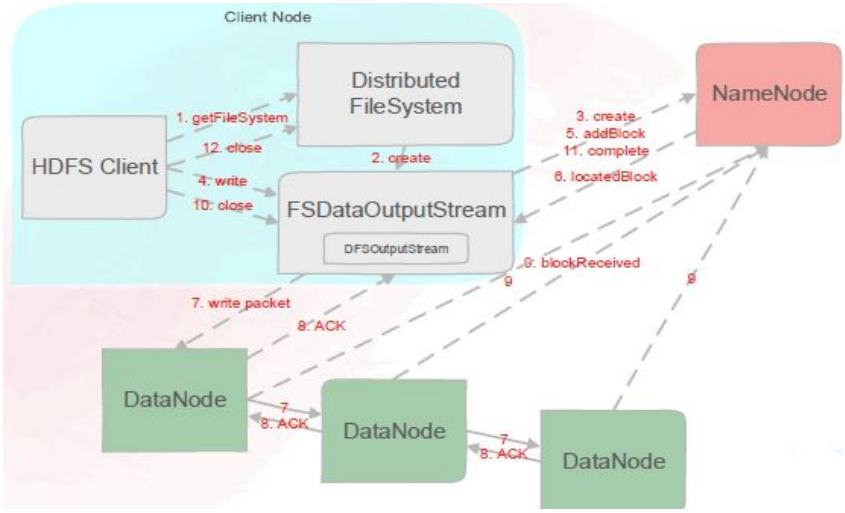
- HDFS：客户端读文件



数据的校验

- 读数据的应答包中，包含了数据的校验和
- 客户端检查数据的一致性
- 降低数据节点的负载，均衡各节点的计算能力

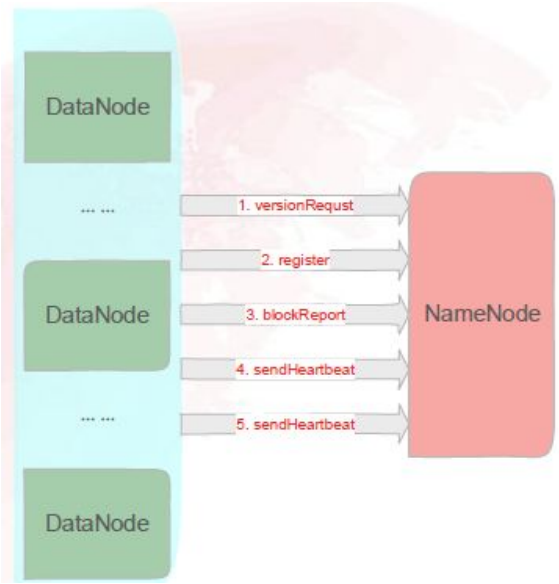
- HDFS：客户端写文件



流中包含的后台线程进行实际的写数据操作

- 数据流管道
- 发送数据Packet线程
- 确认ACK线程

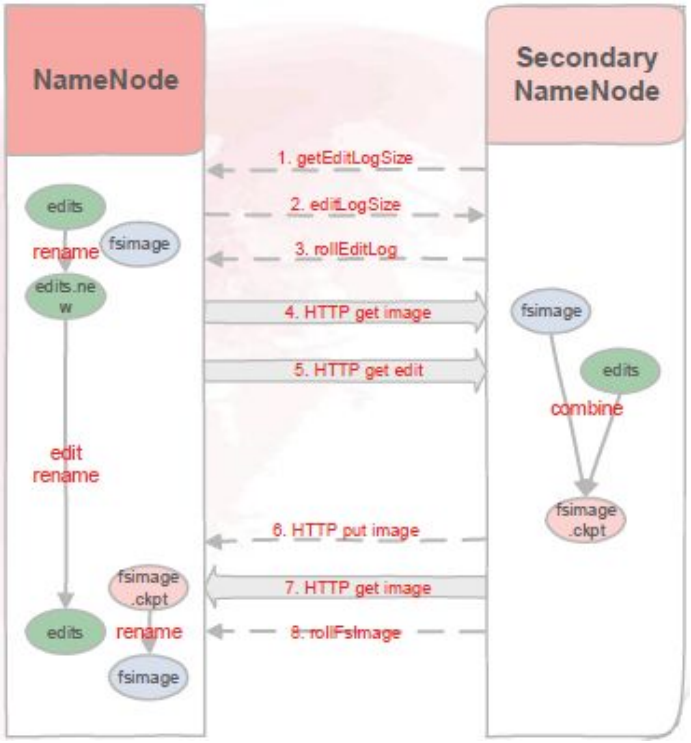
- HDFS：DataNode的启动和心跳



- DataNode与NameNode的交互过程
  - HDFS版本一致性检测
  - 成功注册到所属集群
  - 上报数据块信息
  - 间隔发送心跳

- HDFS：Secondary NameNode合并元数据

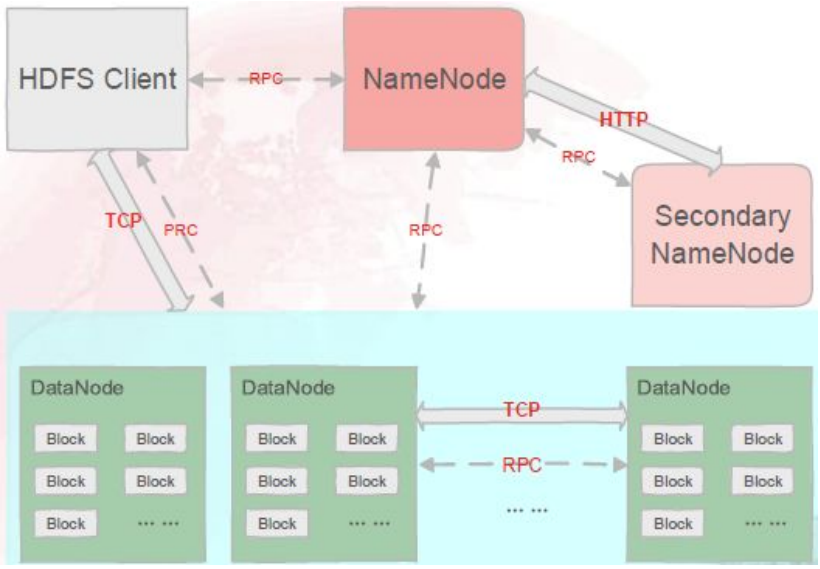
元数据机制
- 编辑日志（edit）
- 命名空间镜像（fsimage）
- 检查点机制(checkpoint)



- HDFS：各个实体间的信息交互接口

HDFS各个实体之间多种信息交互的过程：

- 简单交互Hadoop RPC实现
- 交互大量数据时，使用基于TCP或基于HTTP的流式接



- **HDFS：特性**

和普通的文件系统相比，HDFS具有如下关键特性：

- 存储容量大
  - HDFS的总数据存储容量可以达到PB级，并且容量随着集群中节点数的增加而线性增长。另外，HDFS中存储的文件也可以很大，典型的文件大小从G bytes 到T bytes。
- 分布式存储
  - HDFS中存储的大文件被框架自动分布到多个节点上存储，文件的大小可以大于集群中任意一个物理磁盘的容量。客户端在读写文件时不用关心数据的具体存储位置。

HDFS总容量随着集群中节点数的增加而线性增长。

- 高度容错
  - 快速自动的故障恢复是HDFS的核心设计目标之一。HDFS通过自动维护多个数据副本和在故障发生时自动重新部署处理逻辑来实现高可靠性。NameNode和DataNode上都部署了周密的错误检测和自动恢复机制。对大规模HDFS集群（几千个节点）来说，每天都会有一两个节点失效。集群一般能很快将失效节点上的Block副本在其他DataNode节点上重新创建出来。

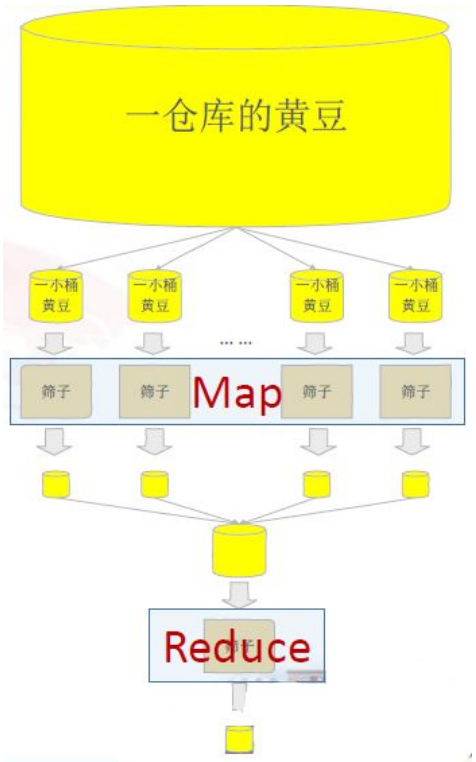
- 高吞吐量
  - 应用程序需要通过流方式访问HDFS中的数据。HDFS设计为批处理模式，而不是交互模式，它强调高吞吐量而不是低延时。HDFS是为高数据吞吐量应用优化的，而这可能会以高时间延迟为代价。
- 高可扩展性
  - HDFS无需停机动态扩容，并且系统的计算和存储能力几乎能够随着集群节点数的增加线性增长。HDFS能够实时增加服务器台数来应对一些高峰或者突发服务需求。
- 负载均衡能力
  - HDFS能够在运行时刻根据各个数据存储节点的可用存储容量变化和实际负载情况动态调整数据在多个节点上的分布，即具有一定的负载均衡能力。

- MapReduce

- 一个分布式计算框架（高性能计算对比）
- Google MapReduce的开源实现
- MapReduce能够解决的问题：
  - 任务可以被分解为多个子问题，且这些子问题相对独立，彼此之间不会有牵制，可以并行进行处理，待并行处理完这些子问题后，任务便被解决。
  - 海量数据的处理问题，如：Top K、频率统计、倒排索引构建(用于关键词搜索)等问题。

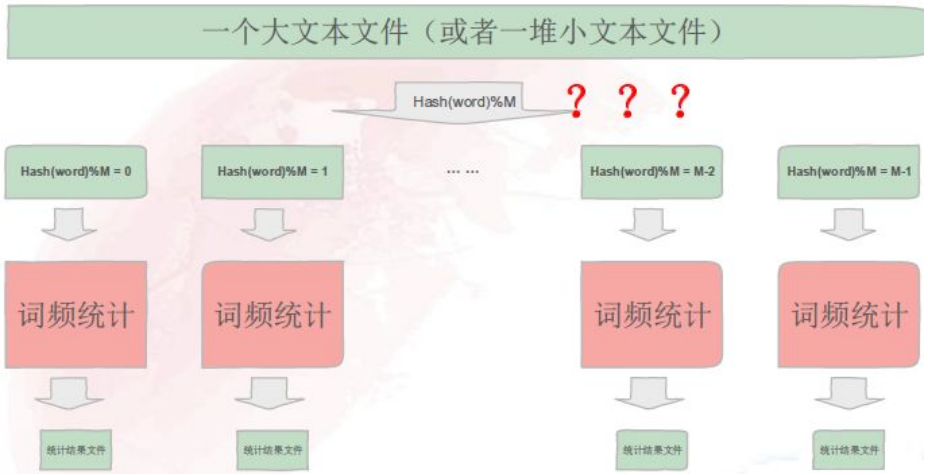
- MapReduce：（Top K）

  - 找出一仓库黄豆中最大的K个黄豆
  - 如何解决？
  - 方案1：
    - 一个很熟练的人筛黄豆，直到找出最大的K个
  - 方案2：
    - 找N个人一起筛黄豆，最后把每个人筛出的K个黄豆放在一起（总共N*K个黄豆），再交由一个人筛出N*K个黄豆里最大的K个（分布式计算）



- MapReduce：（频率统计）

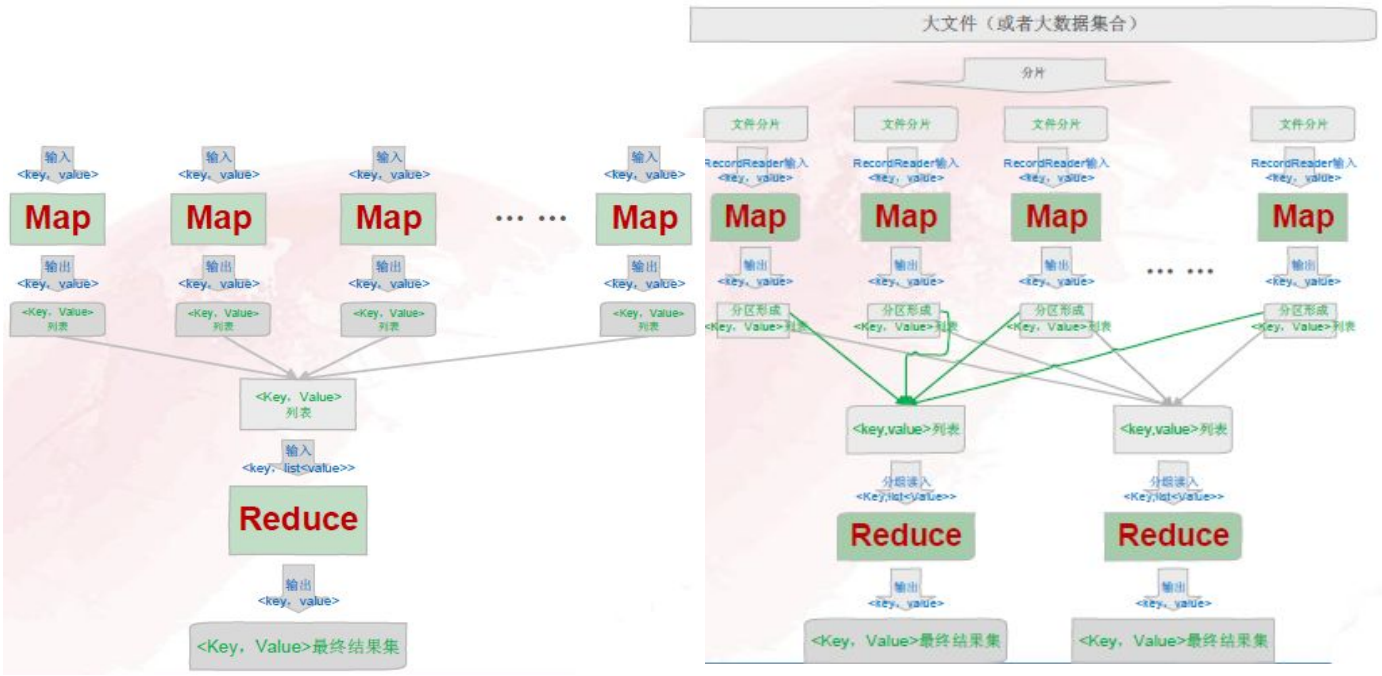
  - 统计一个大文本文件（或者一堆小文件）中的词频
  - 如何解决？
  - 方案1：
    - 一台依次扫描文件进行词频统计（高性能计算）
    - 存在问题：如果单词数很多，并假设一个单词1B，则n GB内存的计算机最多处理具有n*G个不同单词的词频统计
  - 方案2：
  - 顺序读取大文件，对于每个词x，取hash(x)%N ，然后按照该值存到N个小文件，再使用N台计算机分别扫描这N个小文件块统计文件词频，再将这N台计算机的结果汇总得到最后的词频汇总（分布式计算/分而治之）



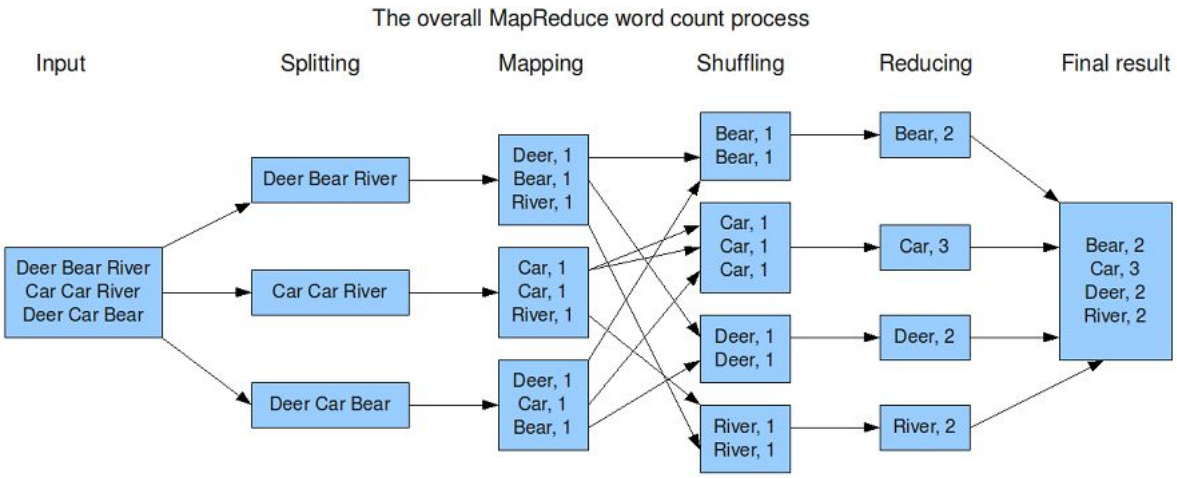
- MapReduce：（频率统计—改进）

- MapReduce：编程模型

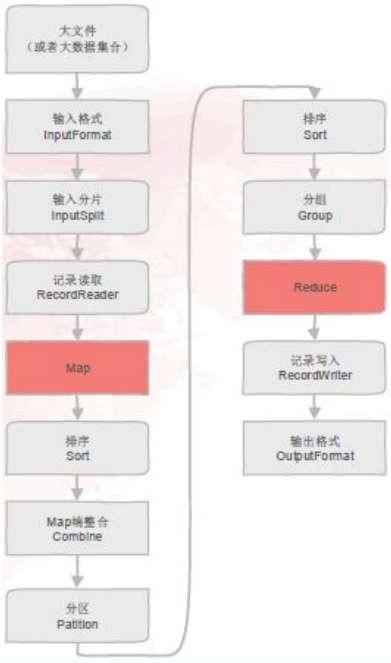
  - MapReduce由两个阶段完成：Map阶段和Reduce阶段。用户只需编写map()和reduce()两个函数，即可完成简单的分布式程序的设计。
  - Map()函数
  - 以<key，value>对作为输入，产生一系列的<key，value>对作为中间结果输出写入本地磁盘。MapReduce框架会自动将这些中间数据按照key值进行分区，且key值分区结果相同的数据会被交给同一个reduce()函数处理。
  - Reduce()函数
  - 以key以及对应的value列表（即<key，list<value>>）作为输入，经过合并key相同的value值后，产生一系列的<key，value>对作为最终结果输出。



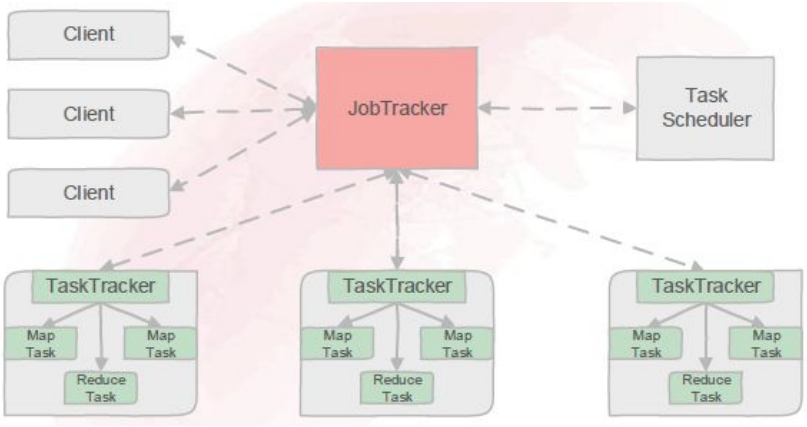
- MapReduce：编程模型之WordCount实例



- Map-Reduce工厂模型

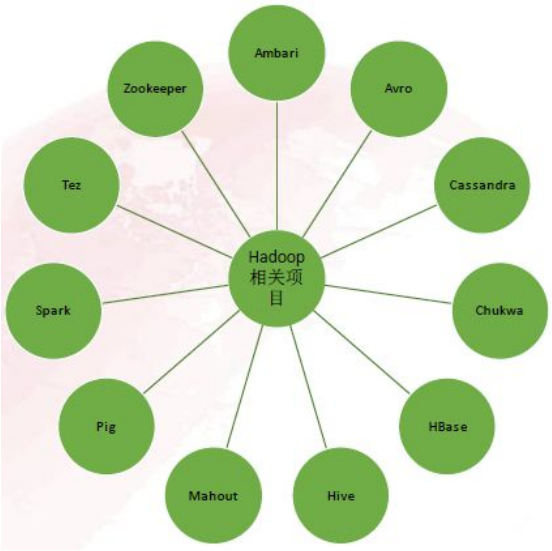


- MapReduce：架构



- YARN: Yet Another Resource Negotiator
  - 一个作业调度和集群资源管理框架
  - 可以认为是MRv2，目标已经不再局限于支持MapReduce一种计算框架，而是朝着多种框架进行统一管理的方向发展。
  - MapReduce、Spark、Storm、S4、MPI等都可以在其上运行
  - 共享集群模式，并且存在诸多优点
    - 资源利用率高
    - 运营成本低
    - 数据共享

- Related project.

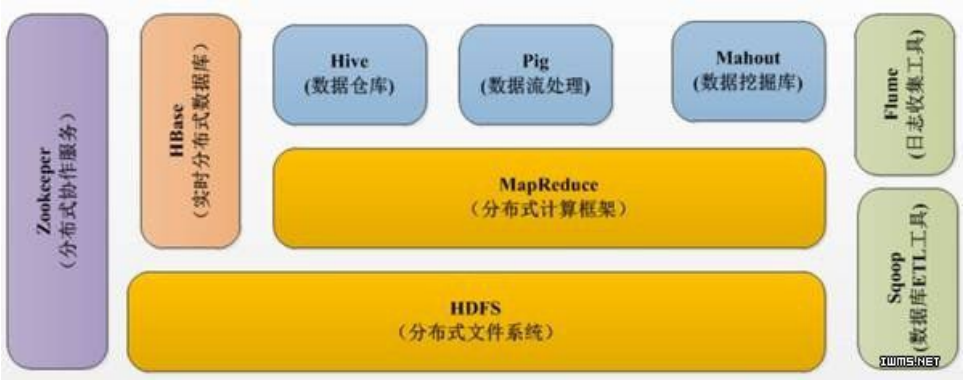


- Ambari
  - Ambari是Hortonworks公司主导的第一个开源实现的Hadoop管理平台，是一种基于Web的工具，支持Hadoop集群的供应、管理和监控。
- Avro
  - 一种支持高效、跨语言的RPC 以及永久存储数据的序列化系统。可以将数据结构或者对象转换成便于存储和传输的格式，其设计目标是用于支持数据密集型应用，适合大规模数据的存储与交换。
- Cassandra
  - 由Facebook支持的开源高可扩展分布式数据库。是Amazon低层架构Dynamo的全分布式和Google Bigtable的列式数据存储模型的有机结合。
  - 可以简单地认为，Cassandra是一个P2P的，高可靠性并具有丰富的数据模型的分布式文件系统。没有主节点，因此也没有单点故障。
- Chukwa
  - 开源的数据收集系统，用于监控大规模分布式系统（2000+以上节点，每天产生的数据量在T级别）。它构建在Hadoop的HDFS和MapReduce基础之上，提供了数据的生成、收集、排序、去重、分析和展示等一系列功能。
- HBase
  - 一个针对结构化数据的可伸缩、高可靠、高性能、分布式和面向列的动态模式数据库。和传统的关系型数据库不同，HBase采用了Google BigTable的数据模型：增强 的稀疏排序映射表（key/value），其中，键由行关键字、列关键字、和时间戳构成。HBase提供了对大规模数据的随机、实时读写访问，同时，HBase中保存的数据可以使用MapReduce来处理，它将数据存储和并行计算完美地结合在一起。
- Hive
  - 一个构建在Hadoop上的数据仓库平台，提供了类似于SQL的查询语言HiveQL以执行查询、变换数据等操作，通过解析，HiveQL语句在底层被转换为相应MapReduce操作。
- Mahout
  - 一个可扩展的机器学习和数据挖掘库，Mahout现在已经包含了聚类、分类、推荐引擎（协同过滤）和频繁集挖掘等广泛使用的数据挖掘方法。
- Pig
  - 一种高级的数据流语言和并行计算的执行框架，用以检索非常大的数据集，运行在MapReduce和HDFS的集群上，简化MapReduce任务的开发。提供了一种高层次的、面向领域的抽象语言：Pig Latin。通过Pig Latin将复杂且相互关联的数据分析任务编码成为Pig操作上的数据流脚本，通过该脚本转换为MapReduce任务链，在Hadoop上执行。
- Spark
  - Spark是一个基于内存计算的开源的集群计算系统，目的是让数据分析更加快速。Spark非常小巧玲珑，由加州伯克利大学AMP实验室的Matei为主的小团队所开发。使用的语言是Scala，项目的core部分的代码只有63个Scala文件，非常短小精悍。尽管创建Spark 是为了支持分布式数据集上的迭代作业，但是实际上它是对Hadoop的补充，可以在Hadoop文件系统中并行运行。通过名为Mesos的

第三方集群框架可以支持此行为。

- Tez
  - Apache最新开源的支持DAG作业的计算框架。它直接源于MapReduce框架，核心思想是将Map和Reduce两个操作进一步拆分，即Map被拆分成Input、Processor、Sort、Merge和Output，Reduce被拆分成Input、Shuffle、Sort、Merge、Processor和Output等，这样，这些分解后的元操作可以任意灵活组合，产生新的操作，这些操作经过一些控制程序组装后，可形成一个大的DAG作业。
- Zookeeper
  - 一个高效、可靠的协同工作系统。解决分布式系统中的一致性问题，在此基础上,还可用于处理分布式应用中经常遇到的一些数据管理问题，如统一命名服务、状态同步服务、集群管理、分布式应用项的管理等。是Google的Chubby的开源实现。

- **Hadoop生态圈**



- Hadoop不仅具有上述的主要优势，它还包括众多的子项目，共同构成了Hadoop生态圈。比如用于部署、检测、管理Hadoop的Web项目Ambari，分布式文件系统HDFS，分布式计算框架MapReduce，针对结构化数据的可伸缩、高可靠、高性能、分布式和面向列的动态模式数据库HBase，分布式协同系统Zookeeper，基于内存的分布式计算框架Spark，数据仓库平台Hive，高级的数据流语言和并行计算的执行框架Pig，可扩展的机器学习和数据挖掘库Mahout，开源的数据收集系统Chukwa，数据序列化系统Avro，开源的支持DAG作业的计算框架Tez。以上众多　的Hadoop子项目，共同使得Hadoop在学术界和工业界大受欢迎。

2. 为什么是Hadoop

2.1 Hadoop的特点

The Apache Hadoopsoftware 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.

- **优点：**
  - 高扩展性（Scalable）
    - Hadoop可以很容易地扩展，改变集群中结点的数目，稳定的存储和处理P-bytes的数据。
  - 高可靠性/高容错性：
    - 存储方面（HDFS）：数据存储的字节级别校验，并且能够自动保存数据的多个副本。
    - 计算方面（MapReduce）：能够自动将失败的计算任务重新分配。
  - 高效性
    - Hadoop可以对任务进行拆分，使得被拆分成的各个子任务可以并行的、尽可能的（数据）本地化处理。
  - 经济型（Economical）
    - 软件方面：Hadoop、Linux都是开源的。
    - 硬件方面：通过普通PC组成的服务器集群来存储、分发以及处
- **不足：**
  - 优先考虑吞吐量，很难满足低时延，尽量权衡。
  - master/slave结构，单点故障不可避免。
  - 一次写入多次读取，无法满足做到修改，只能覆盖或者追加写。
  - 网络带宽会成为系统瓶颈。
  - 采用Java实现，对于CPU密集型任务存在劣势。
  - 不适合小文件存储。
  - 安全问题。

3. 如何使用Hadoop

- **安装（linux，windows）**
- 修改和编译源码（源码编译环境配置）
- **源码阅读环境搭建（参看文档如何搭建Hadoop集群）**
  - 单机模式
  - 伪分布式模式
  - 完全分布式模式（ambari）
- **开发应用程序**
- **应用程序运行**