# 第1章 初识Hadoop

## 1.1 数据！数据！

## 1.2 数据的存储与分析

要对多个硬盘中的数据并行进行读/写，有两个问题需要解决。

第一个需要解决的是**硬件故障问题**。一旦开始使用多个硬件，其中个别硬件就很有可能发生故障。为了避免数据丢失，最常见的做法是复制，如Hadoop的文件系统（HDFS）。

第二个问题是**大多数分析任务需要以某种方式结合大部分数据来共同完成分析**，即从一个硬盘读取的数据可能需要与从另外99个硬盘中读取的数据结合使用。MapReduce提出一个编程模型，该模型抽象出这些硬盘读/写问题并将其转换为对一个数据集（由键-值对组成）的计算。这样的计算由**map**和**reduce**两部分组成，而且只有这两部分提供对外的接口。

## 1.3 查询所有数据

MapReduce是一个批量查询处理器，能够在合理的时间范围内处理针对整个数据集的动态查询。

## 1.4 不仅仅是批处理

从MapReduce的所有长处来看，它基本上是一个**批处理系统**，并不适合交互式分析。你不可能执行一条查询并在几秒内或更短的时间内得到结果。因此，**MapReduce更适合那种没有用户在现场等待结果的离线使用场景**。

## 1.5 相较于其他系统的优势

### 1.5.1 关系型数据库管理系统

* 为什么不能用配有大量硬盘的数据库来进行大规模数据分析？为什么要Hadoop？

寻址（将磁头移动到特定硬盘位置进行读/写操作的过程，是导致磁盘操作延迟的主要原因）时间的提升远不如传输速率（取决于硬盘的带宽）的提升。

**如果数据访问模式中包含大量的硬盘寻址，那么读取大量数据集就必然会花更长的时间**。另一方面，如果数据库系统只更新一小部分记录，那么传统的B树就更有优势。但**数据库系统如果有大量数据更新时，B树的效率就明显落后于MapReduce，因为需要使用“排序/合并”来重建数据库**。

MapReduce适合解决需要以**批处理**方式分析整个数据集的问题。RDBMS适用于索引后数据集的**点查询和更新**，建立索引的数据库系统能够提供对小规模数据的低延迟数据检索和快速更新。MapReduce适合**一次写入、多次读取**数据的应用，关系型数据库则更适合**持续更新**的数据集。

**结构化数据**：具有既定格式的实体化数据，如XML文档和满足特定预定义格式的数据库表。

**半结构化数据**：比较松散，有格式，但常被忽略，作为对数据结构的一般性指导，如电子表格，它在结构上是由单元格组成的网格，但是每个单元格内可以保存任何形式的数据。

**非结构化数据**：无内部结构，如纯文本和图像数据。

Hadoop对非结构化或半结构化数据非常有效，因为它是**在处理数据时才对数据进行解释**（“**读时模式**”）。这种模式在提供灵活性的同时避免了RDBMS数据加载阶段带来的高开销，因为在Hadoop中仅仅是一个文件拷贝操作。

关系型数据往往是规范的，以保持其数据的完整性且不含冗余。规范给Hadoop处理带来了问题，因为它使记录读取成为非本地操作，而Hadoop的核心假设之一偏偏就是可以进行高速的流读/写操作。Web服务器日志是典型的非规范化数据记录（每次都需要记录客户端主机全名，这会导致同一客户端的全名可能多次出现）。Hadoop能够做join操作。

MapReduce以及Hadoop中其他的处理模型是可以**随着数据规模线性伸缩的**。对数据分区后，函数原语（map和reduce）能够在各个分区上并行工作。如果输入的数据量是原来的两倍，那么作业的运行时间也需要两倍。如果集群规模扩展为原来的两倍，那么作业的运行速度仍然与原来一样快。SQL查询没有该特性。

### 1.5.2 网格计算

高性能计算和网格计算主要使用类似于消息传递接口MPI（Message Passing Interface）的API。高性能计算将作业分散到集群的各台机器上，这些机器访问存储区域网络所组成的共享文件系统。这适合于计算密集型作业，但**如果节点需要访问的数据量很庞大，很多计算节点会因为网络带宽的瓶颈闲下来等待数据**。

**数据本地化**：Hadoop在计算节点上存储数据，以实现数据的本地快速访问。网络带宽是数据中心环境最珍贵的资源（到处复制数据很容易耗尽网络带宽），因此Hadoop通过显式网络拓扑结构来保留网络带宽。

MPI需要程序员显式处理数据流机制，包括用C语言构造底层的功能模块和高层的数据分析算法。而Hadoop在更高层次上执行任务，程序员仅从数据模型（MapReduce的键-值对）的角度考虑任务的执行，数据流仍是隐形的。

在大规模分布式计算环境下，协调各个进程的执行时一个很大的挑战。最困难的是合理处理系统的部分失效问题（在不知道一个远程进程是否挂了的情况下）同时还需要继续完成整个计算。MapReduce分布式处理框架使得程序员不必担心系统失效的问题，因为框架**能够检测到失败的任务并重新在正常的机器上执行**。正因为采用的是无共享框架，MapReduce才能够呈现出这种特性，这意味着各个任务之间是彼此独立的。

### 1.5.3 志愿计算

**志愿计算**项目将问题分成很多块，每一块称为一个**工作单元**，发到世界各地的计算机上进行分析。

志愿计算项目时CPU高度密集的，志愿者贡献的是CPU周期，而不是网络带宽。

MapReduce的三大设计目标：

（1）**为只需要短短几分钟或几个小时就可以完成的作业提供服务**。

（2）**运行于同一个内部有高速网络连接的数据中心内**。

（3）**数据中心内的计算机都是可靠的、专门的硬件**。

## 1.6 Apache Hadoop发展简史

## 1.7 本书包含的内容

# 第2章 关于MapReduce

MapReduce是一种用于数据处理的编程模型，本质上是并行运行的，可以将大规模的数据分析任务分发给任何一个拥有足够多机器的数据中心。

## 2.1 气象数据集

## 2.2 使用Unix工具来分析数据

## 2.3 使用Hadoop来分析数据

### 2.3.1 map和reduce