大数据分析的硬件与系统支持综述

孙仕亮，陈俊宇
（华东师范大学 计算机科学技术系 上海 200241）
(slsun@cs.ecnu.edu.cn)

A review of hardware and system supports for big data analysis

Sun Shiliang and Chen Junyu
(Department of Computer Science and Technology, East China Normal University, Shanghai 200241)

Abstract The big data era has come, which brings new challenges and opportunities to the information science community. In this paper, we illustrate the basic concept of big data, “four Vs” features and the big data lifecycle, as well as review several data processing models and data structures. Then, we not only summarize hardware support but also introduce a variety of typical system supports for big data analysis, such as Hadoop, Spark and Storm. Finally, we present several typical applications for big data analysis in the field of government, education, transportation and medical treatment.

Keywords big data analysis; data processing technology; hardware support; distributed system

摘要 近几年来，大数据的浪潮席卷全球，这给信息科学带来了新的挑战和机遇。本文首先阐述了大数据的基本概念、4V 特征，介绍了大数据的生命周期，并列举了几种数据处理模式与数据结构。然后，文中归纳总结了目前针对大数据分析的硬件支持，介绍了几种典型的大数据分析的系统技术，例如 Hadoop、Spark、Storm 等。最后简述了大数据分析在政府、教育、交通、医疗中的典型应用。

关键词 大数据分析；数据处理技术；硬件支持；分布式系统

在大数据时代来临之前，各行各业就早已产生了大量的数据。但由于当时技术上的限制，电脑尚未普及，因此数据难以通过电子信息设备进行存储、管理与分析。如今，我们生活在一个互联网的时代，大量数据可以通过数字信号的形式存储在电子设备中。在 2013 年国际数据公司(International Data Corporation, IDC)评估出了数据宇宙的容量大概在 4.4ZB，并预测在 2020 年将达到 44ZB[1]。人们生活中的一些行为每时每刻都能产生大量的数据，例如在纽约股票交易所，每日将产生约 4TB 到 5TB 的数据。在社交网站 Facebook，每月有超过 7PB 的照片发布[2]。随着这几年互联网行业的发展，大数据处理技术的出现也正逐渐改变着人们的生活方式。

大数据最早源自商业智能的发展。1989 年，Howard Dresner 最早提出了商业智能的定义。商业智能就是针对企业运营过程产生的大量数据，采用数学统计建模的方法，挖掘出有用的信息，从而基于事实来做出一些更好的商业决策，获得更高的利润。其实质就是一个从数据到信息再到知识的智能分析处理。为了构建一个完整的商业智能系统，企业需要采用数据仓库技术，数据挖掘技术，联机分析处理(OLAP)等等[3]。然而，随着技术的发展，所收集数据的复杂度越来越高，传统的工具逐渐难以满足目前企业的需求。相比传统的企业数据源（例如客户信息，财务，

收稿日期：0000-00-00 修回日期：0000-00-00

人力资源等），目前数据的来源也趋向于多样化（例如社交网络，电商平台，物联网等），因此不少企业开始寻求大规模数据的分析与存储方法，随之催生了一个新的名词——“大数据”。全球知名咨询公司麦肯锡称：“数据，已经渗透到当今每一个行业和业务职能领域，成为重要的生产因素。人们对于海量数据的挖掘和运用，预示着新一波生产率增长和消费者盈余浪潮的到来[4]。”近年来，大数据已成为互联网技术行业的一个热门词汇。与此同时，大数据的产生促进了云计算的发展。云计算使大数据有了运行的轨道，有了应用的价值，是大数据的信息技术基础。企业用户可根据自身需求将大量的数据存储到云端，降低了自身搭建平台的成本。

虽然大数据的出现给予了企业大量有用的信息，但与此同时企业也意识到这种附带价值所带来的许多挑战。例如需要花费昂贵的价格购买数据分析软件，购置大量的基础计算设备以满足计算密集型应用的需要，需要雇佣一个数据分析团队来帮助企业分析、整合这些数据[5]。然而，在大部分情况下这些数据具有时效性，只能在某一段特定的时间内有效，并且要花费昂贵的代价来计算与分析。尽管如此，企业还是乐此不疲，因为企业可以通过对数据的分析，更好地了解客户的行为与潜在需求，可以为将来的新产品做一个良好的铺垫，做出更加合理的决策。

大数据分析主要涉及两个不同的领域：一是如何将海量的数据存储起来，二是如何在短时间内处理大量不同类型的数据。简而言之，就是要致力于解决大数据存储与大数据处理等问题。

1. 大数据概览

1.1 大数据的基本概念

大数据从字面意义上看，是指规模庞大的数据，然而大数据的意义远不止这些。不过，对于“大数据”的概念目前并没有一个明确的定义。麦肯锡公司给出的大数据定义是：“数据大小超出常规的数据库工具获取、存储、管理和分析能力的数据集[4]。”但她同时强调，并不是一定要超过特定容量的数据才能算是大数据。IDC 将大数据技术定义为：“为更经济地从高频率的、大容量的、不同结构和类型的数据中获取价值而设计的新一代架构和技术[6]。”虽然描述不尽相同，但许多公司、组织机构都存在一种共识，就是大数据的关键在于种类繁多，数量庞大，使用传统的数据分析工具无法在可容忍的时间内处理相应的数据。

1.2 大数据的主要特征

目前，人们普遍认为，大数据具有 4V 特征：大量化(Volume)、多样性(Variety)、快速化(Velocity)、价值密度低(Value)[7][8][9]。

大量化(Volume)指数据的数量巨大。这些年来，日新月异的信息存储技术使得存储大量数据的成本越来越低，特别是分布式存储技术的日益成熟，逐渐使得存储 PB、EB 甚至 ZB 级别的数据成为可能。随着大数据存储技术的发展，数据量的规模已不再是一个挑战与难题。

多样性(Variety)指数据的种类繁多。人们只需要连上互联网，就可以随时随地查看并获取想要的数据，但与此同时也面临了一系列的挑战。互联网上的数据虽多，但大部分数据的呈现形式为非结构化或半结构化的。如何将不同的数据结构归结到统一的结构中是一个重要的问题。表 1 列举了几种不同的数据结构类型。

表 1 数据结构类型

Table 1 Type of Data Structure

数据结构类型	描述	实例
结构化 (Structured Data)	有相同的表结构并以相同的模式呈现	Database
非结构化 (Unstructured Data)	没有固定的表现模式	文本，音频，图片
半结构化 (Semi-Structured Data)	没有严格的结构形式	XML，HTML

快速化(Velocity)是指目前大数据时代，数据越来越实时化，数据的产生与处理速度逐渐能够满足人们的需求。例如，微博能通过最近人们所发的博文快速地反馈出微博热点，外出就餐能快速地查询到附近评价较好的餐厅，网上购物能迅速地反馈货物信息等等，这些无疑都是人们与机器进行互动的过程。

价值密度低(Value)是大数据中最为关键的一点，虽然真实世界中的数据量极大，但真正有价值的内容却较少。以监控视频为例，虽然监控视频的内容极其之大，但实际有价值的部分可能不过几分钟。如何利用云计算等技术从大量的数据中提取出最为关键、最有价值的部分，并将信息转换成知识无疑是值得研究的内容。

1.3 大数据的生命周期

如图 1 所示[10]，在传统的数据分析处理流程中，数据的来源是多种多样的，既可以从企业的数据库中获取，也可以从社交媒体，例如微博，Facebook

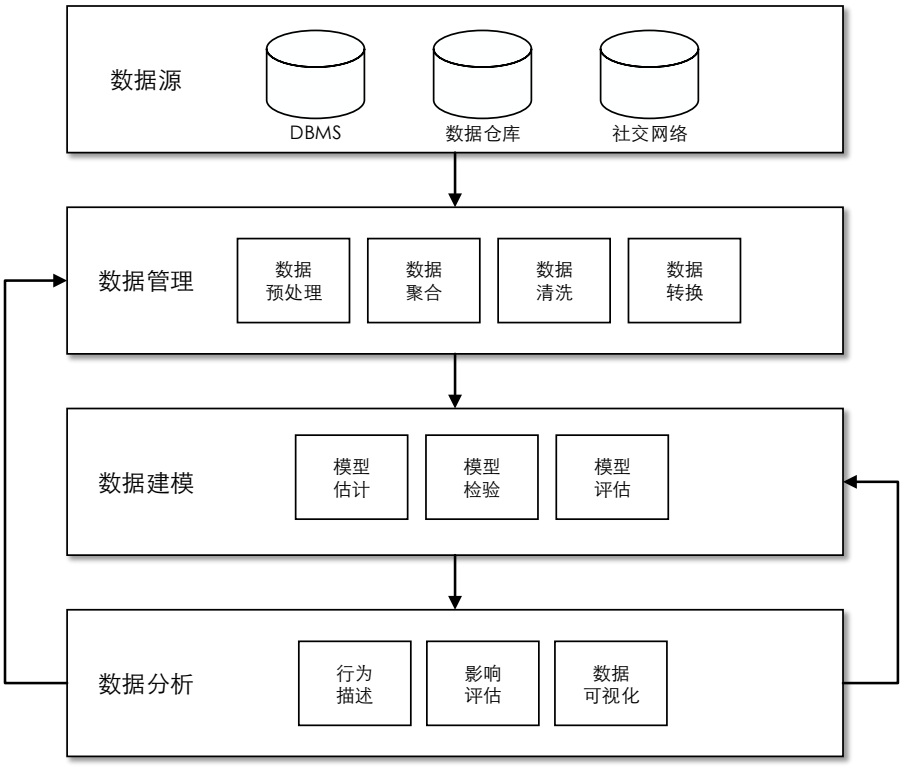

图 1 大数据的处理流程

Fig. 1 Big data processing

中获取。尽管数据的来源是多种多样的，但由于大数据价值密度低的特点，通常获取到的数据并不能直接使用进行分析，还需要进行一系列预处理。例如，将无用或者重复的数据过滤并去除，将大量的数据分类并进行管理，根据业务需要对相同类型的数据进行聚合，将非结构化或半结构化的数据结构化并存储到数据库中，或者将原结构化的数据从原有表现形式统一成另一种表示形式，从而使数据井井有条以便于数据分析工作的开展。

一旦数据整合完成，就可以使用统计建模方法建立模型，用数据集进行训练，估计出模型参数。模型建立完成后，在模型投入使用之前还需要对其进行数据检验。最后，训练好的模型要接受新数据的检验，这一阶段也叫模型评估，它可以用于决策、推荐，也可以通过新收集的数据重新估计参数更新模型。

通过数据建模进行统计分析是极其有意义的。用户行为数据是大数据中一种较为常见的类型。通过大数据技术可以对用户行为数据(例如商品购买记录，网页访问记录等等)进行分析，从而挖掘出用户与商品之间的关联性，并以此推荐出用户喜爱的商品。

1.4 大数据的计算模式

大数据的计算模式主要有三种：批处理(Batch Process)、流处理(Stream Process)和实时处理(Real-Time Process)。表 2 列举了这三种计算模式的区别。其中，批处理是对静态数据的计算，也称离线计算。在计算中，首先要将数据存储集中起来，才能进行相关的计算工作。Hadoop 的 MapReduce 就是批处理计算的典型例子，通过 Hadoop 分布式文件系统(Hadoop Distributed File System, HDFS)存储数据，大量的数据通过分块作多次冗余的方式进行分布式存储，然后通过 MapReduce 切分任务并分配给各个计算节点进行并行计算。批处理对容错的要求很高，适合于数据精度要求严格、实时性要求不高的场景。目前关于批处理的相关技术研究已经相对成熟，Apache 开源的 Hadoop 计算系统以其稳定、高效、容错率高等特点已广泛用于各类应用场景中。相比而言，流处理是对动态数据的计算，它无法确切地获知下一时刻到来的数据，也无法将所有的数据存储起来。所以，流处理不会存储数据到磁盘中去，而是将数据直接放置到内存里实时计算。Storm（一款由 Twitter 开发的分布式的、容错的实时计算系统）与 Yahoo! S4 (Simple Scalable Streaming System，一款由 Yahoo 公司开发的分布式流处理引擎)就是流处理计算范式的典型实现。对于容错率要求不高，但实时性要求高的业务场景，流处理就有着明显的优势。实际上，在大多数应用场景下，两者并不冲突，可以将批处理与流处理结

合起来使用，充分发挥各自的优势。而实时处理就意味着要在较低的时延下满足客户端的请求。对于大数据分析，实时处理也是一个极为常见的计算模式，例如淘宝网上的订单数量极其庞大，而且增长速率极为迅速，查询订单时要满足较低的时延，提交订单时对并发性又有较高的要求。使用传统的数据库难以承担这一重任。而 Hadoop 生态圈下的 HBase 就非常适合这一类应用场景。HBase 是一款分布式、面向列存储的数据库系统，主要用于存储非结构化或半结构化的松散数据。

表 2 数据处理类型

Table 2 Type of Data Processing

数据处理类型	描述	实例
批处理 (Batch Process)	系统能处理某个区间内时间的数据	银行流水账单对账
流处理 (Stream Process)	系统能满足源源不断的数据的计算需求	视频监控、社交网络分析
实时处理 (Real-Time Process)	系统能在较低的延迟内回馈操作的请求	数据库交互式操作

2. 大数据分析的硬件支持

在目前数据飞速增长的背景下，大数据的计算和存储对硬件平台提出了新的要求。企业的存储容量面临着巨大的压力，而应用的多元化增加了企业系统资源管理的复杂性；为了满足海量数据处理的需求，企业的运营成本常常居高不下。因此，为了满足大数据时代的需求，企业需要大容量、高性能、高可靠性、高性价比的硬件来满足大数据存储、保护和处理等方面的需求。

2.1 GPU

GPU，亦称图形处理单元，它在执行数学和几何计算方面具有天生的优势，而这些计算都是图形渲染所必需的。近几年，随着 GPU 运算速度的大幅提升和可编程性的发展，GPU 的调用也越来越方便，它的应用也已经不限于图像渲染任务。人们逐渐采用 GPU 作为通用计算加速器，以满足快速运算的功能需求。目前，人们开始趋向于计算的异构化，Intel 公司自 Sandy Bridge 架构开始，就开始将 CPU 与 GPU 集成到一个芯片中，AMD 公司的 APU (Accelerated Processing Unit, 加速处理器)也采用了 CPU 和 GPU 的异构融合，对两边的计算进行协调[11]。

GPU 和 CPU 的差异很大。CPU 中的功能模块很多，大部分晶体管主要用作控制电路和 Cache，仅有少部分的晶体管用于完成单纯的计算，因此比较适合流程复杂的运算环境。相比 CPU 计算，GPU 虽然有着巨大的数学运算能力，但是它执行分支控制指令的性能较为低下。这也是由于在 GPU 中流处理器与显存控制器占据了大部分晶体管，而对 Cache 的需求较小，因此对单纯的运算密集型应用具有天生的优势。

GPU 最初的设计思路是实现图形加速。以 3D 图形的渲染为例，GPU 在渲染中需要完成如下几个操作：顶点生成，顶点处理，光栅化计算，纹理贴图，像素处理等。GPU 具备并行处理的能力，能够合成高分辨率的图像。它面临的数据类型较为单一，通常为浮点数运算，而且数据量大，并且相比 CPU，GPU 的技术进步已超过摩尔定律，每六个月性能就能翻一倍。因此，GPU 计算也具有更高的性价比与功耗比。

2.2 内存与外存

2008 年的单服务器主流配置中，系统内存通常只有 8GB，硬盘容量只有 1TB，而如今主流的内存最高可达 192GB，硬盘可挂载 48TB，内存和硬盘的容量逐渐增加[12]。计算过程中越来越多的内容可以被直接缓存，而不需要存储到本地硬盘中。这实现了大数据“内存运算”的伟大愿景。以 Apache Spark (一款由 UC Berkeley AMP lab 所开发的基于 MapReduce 的通用并行框架)为代表，与 Hadoop 不同，Spark 将 MapReduce 的中间结果存储到内存中，而不是暂时存储到硬盘中，从而不需要读写 HDFS，使得 Spark 能更好地适用于数据挖掘与机器学习等这些需要迭代的 MapReduce 算法，极大地提高了迭代效率。其实现归功于如今内存成本的降低和内存容量的增大。

内存是随机存取存储器(RAM)，作为与 CPU 直接交换数据的设备，虽然它的传输速度很快，但是这种存储器在机器断电时将丢失其存储内容。在传统的可靠性研究中，可以通过多机备份或将数据写入硬盘以保证数据的可靠性，但这又要以牺牲系统吞吐量为代价。因此，如何设计出高效的数据可靠性保障机制是一个关键问题。这项研究也已经取得了很多进展，例如 Redis 使用快照(RDB)和写操作存储(AOF)来支持容错[13]，Spark 使用 checkpoint 和 lineage 来支持容错[14]等等。

另一方面，大规模数据的存储也是值得关注的问题。文献[15]较早地提出了 Ceph 分布式存储系统，在硬件性能有所局限的条件下，通过分布式技术对存储空间进行扩展。Ceph 系统尽可能地将元数据与数据分离，使其具有可扩展性。这兼容了原有的操作系统接口，同时加入了容错与冗余复制的功能。随着存储技术的迅速发展，对存储容量的提升已不再是最主要的需求，存储的速率更为重要。固态硬盘(Solid State

Drives, SSD)作为一种新型的存储设备也逐渐普及。相比传统的机械硬盘，固态硬盘读写速度更快。由于不采用磁头，因此几乎没有寻道时间，持续写入的时间十分惊人。而且 SSD 的成本相对低廉，同等容量下内存的成本要比 SSD 贵三十倍。SSD 在大数据运算场景下的使用越来越多，并能与内存计算相得益彰。

2.3 通信

目前计算的趋势越来越趋向于分布式内存计算。这种技术首先要减少跨节点数据的传输，使得不同节点间的数据传输量尽可能少，以花费较多的时间在计算任务上。如果不可避免，那就需要较为高速的传输速率，充分发挥内存计算的优势。大数据量高性能运算的需求，催生了远程直接数据存储技术(Remote Direct Memory Access, RDMA)的产生。

RDMA 的出现是为了解决网络传输中的延迟问题。在传统的 TCP/IP 协议中，将一些异构的机器连接起来，组成一个大的计算机网络。虽然这种协议有很好的兼容性和通用性，但是却是以牺牲效率作为代价的。例如在数据传输过程中要在操作系统的用户空间与内核空间之间进行多次重复的拷贝，若能减少这种拷贝的次数将能极大地提升传输的效率。RDMA 协议通过网络直接将数据传输至某台机器一块特定的存储空间，而不对内核空间造成影响。从而减少了不必要的频繁数据拷贝，减少了数据传输的延迟，增大了数据的吞吐量，使得传输的开销与带宽最小化[16]。

此外，也可以通过采用有弹性的网络拓扑结构，在需要的时候灵活调整网络吞吐量，从而更好地适应大数据分析应用的需求。例如，[17]提出了软件定义网络(Software Defined Network, SDN)的概念。在 SDN 架构下，经过改造后的网络设备的控制面与数据面将被分离开来，并由集中的控制器管理。这样，底层设备被抽象出来，降低了不同设备之间的差异性。一切规则都可以以编程的方式进行设定与管理。如今，各网络硬件厂商也陆续推出了各自的 SDN 解决方案，这使得大数据分析平台拥有更加灵活多变的网络拓扑结构，以适应更稳定、高效的网络需求。

3. 大数据分析的系统支持

数据的急速增长使得传统的单机服务器难以面临如此巨大的挑战。在大规模数据处理系统中，通常有两个方面需要保证，一是要保证数据的完备性、可靠性，二是要保证数据处理能在可容忍的时间内完成。因此传统的单机作业，例如 CPU 多线程编程，难以处理大规模的数据，单台服务器的外存也并不能满足大规模数据快速增长的需求。因此，在大规模数据系统中，通常采用分布式处理，通过硬件横向扩展的方式来保证满足上述的要求。

3.1 Hadoop

Apache Hadoop 是分布式数据处理中最著名的一款软件框架。这款框架主要实现了一种编程范式：MapReduce。运用这种编程范式，开发人员可以容易地开发分布式应用[18]。Hadoop 平台包括 MapReduce、Hadoop 分布式存储(HDFS)和其他一些相关的项目，例如 Hive，HBase 等等。

3.2 MapReduce

MapReduce 是一种编程模型，被广泛应用于大规模数据的处理中。2004 年，Google 发表了一篇论文，介绍了 MapReduce 编程框架。通过这款软件框架，开发人员可以快速地编写分布式应用程序。现今，该框架已被广泛地应用到日志分析、海量数据排序等任务中。MapReduce 编程模式实际上是基于一种传统的算法——分治法而实现的。分治法将复杂问题分成多个类似的子问题，直到子问题的规模小到能直接得出结果，再聚合中间数据所得到的最终结果就是原问题的解。在 Hadoop 中，分(Divide) 与合(Conquer)的步骤分别在 Map 和 Reduce 中实现。

Hadoop MapReduce 采用了主从式结构，集群中存在两种类型的节点，主节点(Master Node)与工作节点(Worker Node)，其作业运行机制如图 2 所示。Master 是整个集群中唯一的管理者，主要工作有任务调度，状态监控等。Worker 则负责计算工作与任务状态回复。在 Map 阶段，主节点将输入数据分割，并将原问题分成多个类似的子问题，然后将分割好的数据与任务交给工作节点进行计算。在 Reduce 阶段，主节点将工作节点计算好的中间结果收集起来并聚合成最终的结果。

在 MapReduce 计算模型下，两个函数是以键值对的形式进行分割与汇总的：Map(k, v)→<k′, v′>，Reduce(k′, v′) →<k″, v″>。开发人员只需要实现这两个抽象函数就可实现分布式运算。以官方的例子单词统计为例，这个例子用于统计一系列文章中相同单词出现的次数。在 Map 阶段，Hadoop 集群会并行地将文章分割成块，产生一组键值对(key, value)，其中 key 是单词，value 指这个文本块中出现的次数，并随之产生大量的中间数据。在 Reduce 阶段，集群会将相同 key 的键值对聚集起来，也就是将不同块中相同的单词统计的中间结果聚集起来，最后逐个相加即可得到最终结果。

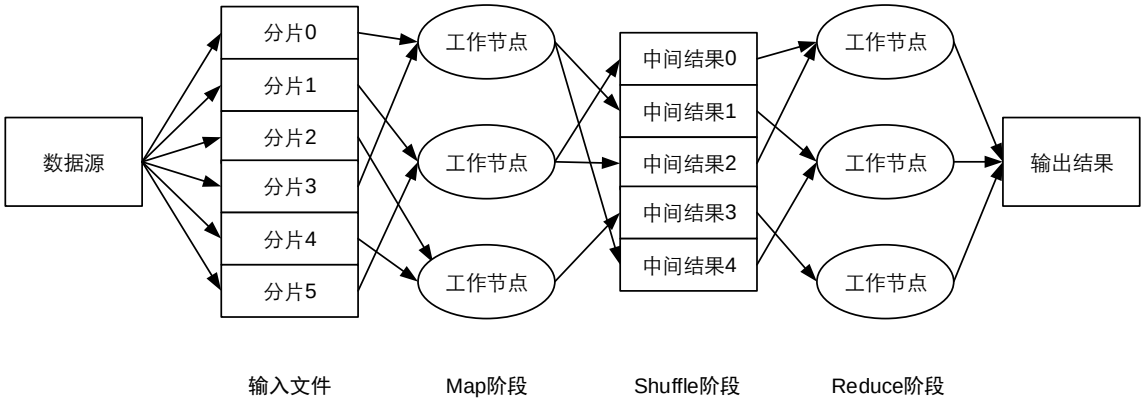

图 2 MapReduce 作业运行机制

Fig. 2 MapReduce

在 Hadoop MapReduce 中，每一个应用程序都被表示为一个作业。Master 节点中会运行 JobTracker，以负责调度任务与分配任务给不同的 Worker 节点。JobTracker 是一个后台服务进程，每个集群中仅有一个。系统启动后，集群当中的 JobTracker 会一直监听每个 Worker 节点中的 TaskTracker 发送的状态信息，从而监控当前的任务运行状态。一旦发现某个工作节点没有工作，JobTracker 会重新请求数据并使相应的工作节点重新执行任务。

3.3 HDFS

HDFS 是 Hadoop 项目的一个子项目，是 Hadoop 应用下的一个分布式文件系统，它提供了文件系统实现的各类接口，使文件系统易于操作。它是基于流式的数据访问模式和处理超大文件的需求而开发，可以运行在廉价的机器上。总的来说，它的主要特点有如下几个[19]：

第一，处理超大文件。这里的超大文件通常是 GB 级甚至是 TB 级的文件。目前在实际应用中，HDFS 已经可以用来管理 PB 级的数据了。

第二，流式地访问数据。HDFS 的设计理念是“一次写入、多次读取”，这意味着当一个数据集生成之后，就会被切分成小文件块，并复制多份分发到不同的存储节点中，然后响应各种数据分析任务请求。在大多数情况下，对 HDFS 来说，读取整个数据集要比读取其中一条记录的单位访问速率快得多。

第三，能运行在低廉的机器集群上。Hadoop 对机器的硬件需求并不高，但廉价的机器通常节点故障的发生概率非常高，这意味着 Hadoop 在设计时要充分考虑数据的可靠性和安全性。

虽然 HDFS 有许多优势，但也存在下面几个缺点：

第一，高延迟的数据访问。这意味着 HDFS 在处理过程中，会有一定程度上的延迟，导致响应缓慢，用户体验较差。主要原因是 HDFS 是为高吞吐量而设计的，因此需要一些高延迟作为补偿代价。

第二，无法高效处理小文件。在 Hadoop 中，需要用到名节点(NameNode)来管理文件系统的元数据(描述数据的数据)，以响应客户端的请求并返回文件位置等信息，因此文件数量的多少，决定着名节点存储的多少。这意味着有许多小文件存储时，名节点的工作压力就会很大，检索元数据的处理时间会显得过长。

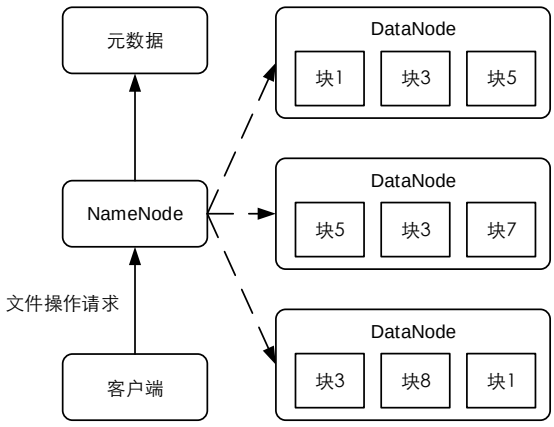

图 3 HDFS 系统架构

Fig. 3 HDFS Architecture

HDFS 采用的是主从式架构。一个集群中存在一个名节点(NameNode)，和多个数据节点(DataNode)，如图 3 所示。名节点作为一个中心服务器，负责管理整个集群文件的读写等操作，存储着各个数据节点的

信息，同时也负责处理用户的请求，进行数据节点的调度。作为整个集群的管理者，名节点的主要任务是对元数据的管理，而不作为存储数据的节点，这样减少了自身的负载。这种结构极大地简化了系统架构，但与此同时也带来了单点故障等问题。

3.4 HBase

Apache HBase 是 Hadoop 项目下的子项目。它是一款基于 HDFS 列存储的可扩展的分布式数据库系统，也就是说，HBase 可以通过 MapReduce 以 Key-Value 的方式实时查询，也可以利用 MapReduce 进行数据的离线处理，从中得到综合报告[20]。传统数据库以满足数据一致性(ACID)为目标，并没有考虑对存储海量数据的扩展性，以及系统故障的容错性。HBase 是用于存储海量数据的，使得数据能够被高效地并发访问。

图4给出了HBase的系统架构图，从中可以看出，HBase 主要处理两种类型的文件：预写式日志(Write-Ahead Log)文件与实际存储的内容文件。这些文件主要由 HRegionServer 进行管理，那些实际存储的内容文件会根据配置的大小分割成小块存储在 HDFS 中。其中 ZooKeeper 是一款分布式的协调系统，通常在另一个集群中部署，以协调管理 HBase 集群中的节点。由于 HBase 是一款数据库系统，对一致性和系统健壮性有着很高的要求，而且通常只有一个 HMaster，所以会存在单点故障问题，然而 ZooKeeper 的出现解决了这一系统的架构问题。

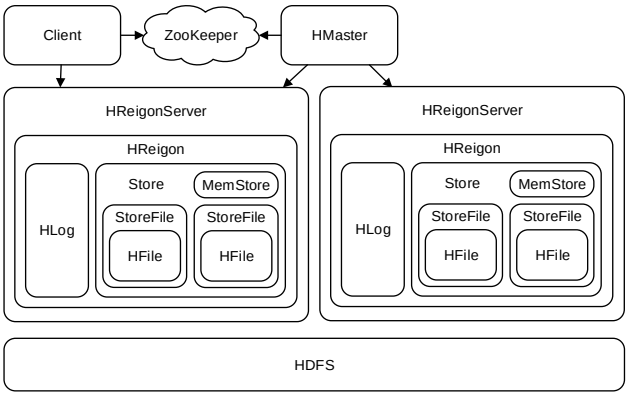

图 4 HBase 系统架构

Fig. 4 HBase Architecture

在 HBase 中，每一个 Table 都会被分割成多个 Region 并存储在不同的 HRegionServer 中，而 Region 是分布式存储和负载均衡的最小单元。在每个 HRegionServer 中都有一个 HLog，用于实现日志的存储。每次在用户进行数据库操作时，会将操作写入内存中，也就是 HBase 中的 MemStore，并定期将日志存储到硬盘中，实现持久化操作。StoreFile 是存储实际数据的位置，通过 HDFS 得以可靠地存储[21]。

相比传统的数据库管理系统，HBase 有以下几个显著优势[22]：

第一，高扩展性。HBase 是真正意义上的线性水平扩展。当数据量累计到一定程度时，HBase 系统会自动对数据进行水平切分，并分配不同的服务器来管理这些数据。这些数据可以被扩散到上千个普通服务器上。这样，一方面可以由大量普通服务器组成大规模集群，来存放海量数据。另一方面，当数据峰值接近系统设定容量时，可以简单通过增加服务器的方式来扩大容量。这个动态扩容过程无需停机，HBase 系统可以照常运行并提供读写服务，完全实现动态无缝无宕机扩容。

第二，高性能。HBase 的设计目的之一是支持高并发用户数的高速读写访问。这是通过两方面来实现的。首先，数据行被水平切分并分布到多台服务器上，在大量用户访问时，访问请求也被分散到了不同的服务器上。虽然每个服务器的服务能力有限，但是数千台服务器汇总后可以提供极高的访问能力。其次，HBase 设计了高效的缓存机制，有效提高了访问的命中率，提高了访问性能。

第三，高可用性。HBase 是建立在 HDFS 上的应用，HBase 的日志和数据都存放在 HDFS 中，这意味着 HBase 具有自动复制与容错的能力。若是出现单个节点设备损坏或者网络故障等问题，数据也不会丢失，可以迅速地从其他节点中将冗余数据取出。HBase 系统会通过一些算法自动分配其他节点接管这些数据。因此，一旦数据成功写入，这些数据就保证被冗余复制。整个系统的高可用性也得以保证。

3.5 Hive

随着商业智能的快速发展，企业数据的规模日益增大，传统的数据仓库解决方案逐渐难以满足高效的性能要求。Hadoop 作为一款开源的存储与处理大数据的系统框架，能够解决传统数据仓库所带来的问题。然而 Hadoop MapReduce 作为一种编程模型是极为底层的，开发者难以有效地开发与维护[23]。Apache Hive 作为一款建立在 Hadoop 上的数据仓库，就是为了解决上述问题而诞生的。它提供了一种用户编程接口，用于查询与管理分布式存储的数据。Hive 依赖于 HDFS 和 MapReduce，并提供了一种类似于 SQL 的查询语言，称为 HiveQL，它允许那些传统的数据库管理人员通过所熟悉的类 SQL 语法来查询数据，而

不需要专门开发相应的 MapReduce 应用[24]。目前，HiveQL 可以通过 DDL (Data Definition Language) 语句完成表结构的创建，数据序列化规则的定义，分区与桶的分配等操作，也可以通过 DML (Data Manipulation Language) 语句导入和新增数据。

Hive 中的数据主要由以下几部分来组织[23]：

(1) 表(Tables)。Hive 中的表与传统关系型数据库中的表类似，每一张表都会对应 HDFS 中的一个目录。表中的数据将被序列化并存储到对应的目录中。

(2) 分区(Partitions)。每一张表都可以拥有一个或者多个分区，每个分区将以子目录的形式存在表目录下。在定义表的时候，主要以列值作为分区块。例如表 T 以列 A 和列 B 进行分区，假如存在 A 的值为 100，B 的值为 200 的数据，则这些数据将被存储在目录 **/T/A=100/B=200** 之下。与单纯地存储在目录**/T** 下的方式相比，分区的做法能提高查询效率。

(3) 桶(Buckets)。每个分区还可以继续划分，划分成多个桶。相比分区，桶对数据的划分颗粒度更细。Hive 通过计算列的哈希值取模可以得到该列数据属于哪个桶，从而使得在小范围数据的查询中效率更高。

Hive 有如下的几个特点：

第一，面向海量数据。Hive 通过将类似 SQL 的查询语句翻译成 MapReduce 程序，并在其基础上做了大量的优化，从而保证了 Hive 可以高效地对海量数据进行处理。

第二，灵活的可扩展性。除了 HiveQL 自身提供的功能外，开发人员还可以根据实际的业务需求，采用任何语言编写自定义的 Mapper 与 Reducer 脚本。因此开发人员可以根据实际情况做出优化，也可以利用这种可扩展性实现复杂的查询。

第三，高度容错性。由于 Hive 底层是基于 HDFS 的，那么相应地，Hive 也有着 HDFS 的容错性。

从图 5 可以看出，Hive 是 Hadoop 顶层的数据仓库，在 Hive 系统架构中，存在以下几个组成部分。用户接口主要有三个，CLI（命令行界面），JDBC（Java 数据库连接），WebGUI。它们提供了访问数据仓库的接口。Thrift 是一款由 Facebook 开发的软件框架，可用于跨语言的开发。该框架允许不同的编程语言调用 Hive 接口。Driver 是 Hive 中较为核心的组件，该组件包括编译器、解释器和优化器，主要用于解析开发人员编写的 HiveQL，并编译优化以调用底层的 MapReduce 框架。MetaStore 组件主要用于存储 Hive 的元数据，这些元数据通常存储在关系型数据库里（如 MySQL）。同时，MetaStore 组件也可以从 Hive 中分离出来，在远程服务器中部署，从而保证了安全性与健壮性。

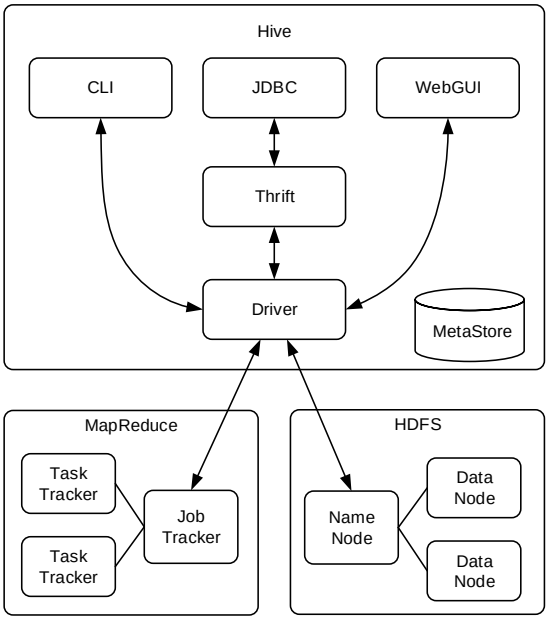

图 5 Hive 系统架构

Fig. 5 Hive Architecture

3.6 Spark

Apache Spark 是一款针对大数据处理的集群计算框架。它与 Hadoop MapReduce 计算框架有许多相似之处，能很好地与 Hadoop 相结合。例如它能运行在 YARN (Yet Another Resource Negotiator，一种 Hadoop 通用资源管理系统，可为上层应用提供统一的资源管理和调度)中，能处理像 HDFS 这种分布式存储格式的文件[14]。

Spark 最广为人知的特点在于它能在不同的作业(Jobs)之间操作一大块内存空间，这使得它比从硬盘中交换数据的 Hadoop 工作方式好。由于集群的数据交换是以内存作为介质而不是从硬盘中读取，Spark 非常适用于迭代算法的并行处理和交互式分析，因为这两种应用通常需要较高的数据交换速率。

弹性分布式数据集(Resilient Distributed Datasets, RDD) 是 Spark 中一项富有特色的核心技术。它允许开发人员能在较大的集群中实现分布式内容计算，同时又有很好的并行性和容错性。以内存为介质记录中间结果能够有效地减少读写操作，从而解决机器学习算法在分布式计算中的瓶颈问题[25]。RDD 是一个容错的、并行的数据结构，任何数据在 Spark 中都被表示为 RDD。它提供了一种共享内存模型，在内存中只读，开发人员只能对其进行转换操作。并且 Spark 内核会在数据处理过程中采用懒加载的处理方式。在这种方式下，RDD 的一系列转换操作不会立即执行，

而是生成一张关于计算路径的有向无环图(Directed Acyclic Graph, DAG)。当需要得到输出或进行 Action 操作时才会进行计算。也就是说，Spark 的内核在不需要输出结果时，仅记录了 RDD 的生成和依赖关系。

Spark 同时也提供了许多内置的数据分析工具，例如机器学习库(MLLib)，图计算库(GraphX)，流处理(Spark Streaming)和 Spark SQL 等。

3.7 Storm

在大数据处理中，Hadoop 以吞吐量大、容错率高等优点在海量数据处理中有着极其广泛的应用。但是，Hadoop 主要用于批处理而不适用于实时处理。于是，Storm 这款实时计算系统就随之产生了。Storm 是一款最早由 BackType 公司 (现已被 Twitter 公司收购)开发的分布式实时计算系统。Storm 为分布式实时计算提供了一组通用原语，其用法与 Hadoop 极其类似，也被称为实时计算版的 Hadoop。它也可被用于“流处理”中，实现实时处理消息并更新数据库。同时 Storm 可以采用任意编程语言编写[26]。

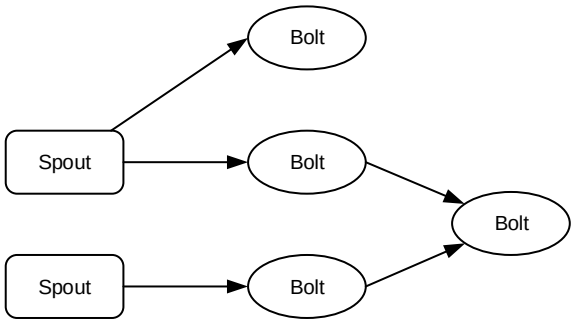

图 6 任务拓扑

Fig. 6 Topology

在 Storm 中，任务拓扑 Topology 是 Storm 的逻辑单元，如图 6 所示。一个任务拓扑是由一系列 Spout 和 Bolt 构成的有向无环图。Spout 是消息的生产者，负责从外部数据源中源源不断地读取数据，并通过使用元组的数据结构将中间结果传递给 Bolt。Bolt 是数据的处理者，负责对到来的中间数据进行转换，如过滤，聚合等，也可以发送到外部的数据流中。Storm 的编程模型十分简单，开发人员只需要编写 Spout 与 Bolt 的任务就可以实现对流数据的处理。

如图 7 所示，Storm 采用主从系统架构，在 Storm 中存在两类节点，一个主节点(Nimbus)与多个从节点(Supervisor)。与 Hadoop 类似，主节点的主要功能是分配计算资源，任务调度，系统状态监测等等。主节点会接收来自客户端的请求，通过 ZooKeeper，将任务信息写入系统，然后将任务分配到多个从节点中。此外，主节点还需要通过 ZooKeeper 对系统当前的运行状态进行监测，对失效的节点进行修复或重启等操作。ZooKeeper 作为一款分布式协调系统，简化了 Storm 的设计，保证了集群运行的可靠性，也便于系统的管理。

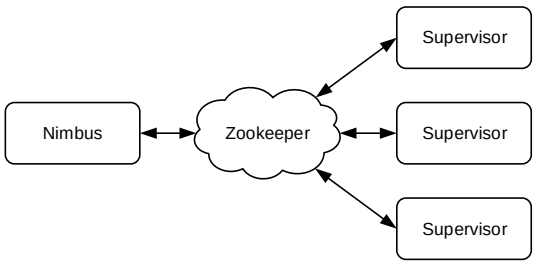

图 7 Storm 系统架构

Fig. 7 Storm Architecture

3.8 Petuum

在人工智能领域，机器学习已经成为一种从数据中提取信息的主要方法。机器学习算法与其他类型的算法相比，具有如下几个特点[27]：(1)容错性。在参数学习过程中，即使出现了某些节点计算产生的偏差，也不会影响某些模型的最终收敛结果。(2)动态结构依赖性。在模型的学习过程中，各个节点的计算并不独立，参数之间的相关性对并行效率影响巨大。(3)参数收敛不均匀性。在模型的学习过程中，有些参数收敛速度较快，而有些参数收敛较慢。这些特点导致了机器学习算法在分布式环境下的实现与其他算法有很大的不同。

许多大数据处理平台，例如 Hadoop 提供的 MapReduce 范式，能让开发人员更为方便地编写分布式程序。但对于机器学习算法来说，Hadoop 没有对数据的迭代计算作出有效的优化。Spark 作为新一代的 MapReduce 并行计算平台，虽然采用了 RDD 技术使计算更加高效，但它并没有提供一个较好的节点调度机制。此外，由于 Spark 是基于数据流的并行计算，在执行机器学习算法时，会耗费较多时间用于节点之间的通信。

Petuum 是一个专门针对机器学习的分布式平台，它的系统设计充分考虑机器学习算法的特点。它的出现主要解决两类机器学习在规模上面临的问题：大数据(大量的数据样本)和大模型(大量的模型参数)。换句话说，Petuum 主要提供了处理这两类问题的机器学习并行化算法：数据并行算法与模型并行算法。数据并行算法指的是在大数据的情况下，数据集可以被切分成多个小块并分配到工作节点中进行处理，这类

算法需要学习的参数不多。而模型并行算法需要学习的参数较多，这种情况下，需要将不同参数的学习分配到多个工作节点中，而且不同参数集的学习具有一定的独立性。

Petuum 包含两个主要模块: (1)基于受限异步一致性协议的分布式键值存储系统 Bösen；(2)动态参数更新调度器 Strads。Bösen 模块采用受限异步一致性协议(Stale Synchronous Parallel, SSP)进行并行数据处理。SSP[28]的基本思想是允许各机器以不同步调对模型进行更新，同时限制最快机器和最慢机器之间的进度不要相差太大，从而缓解了慢机器拖慢整体进度的状况。Strads 模块作为调度器模块，提供的编程接口主要包含三个操作：(1)Schedule: 调度节点根据模型参数的相互依赖性和收敛不均匀性，自动选择一个待更新的参数子集；(2)Push: 调度节点令计算节点并行地为选好的参数计算更新；(3)Pull: 调度节点从计算节点收集信息，并更新参数。

目前 Petuum 还推出了基于深度学习框架 Caffe 的分布式 GPU 深度学习框架 Poseidon。这个框架维持了与 Caffe 类似的程序接口以方便开发人员快速开发，并通过分布式架构使性能得到了较为满意的提升。

3.9 对比分析

在表3中，我们对上述的一些系统进行了对比，其中由于MapReduce与HDFS是底层框架结构，主要用于支撑顶层应用，因此不列入比较范围之内。

表3 大数据系统对比

Table 3 Comparison of Big Data Systems

	HBase	Hive	Spark	Storm	Petuum
主要用途	分布式存储	分布式存储	分布式计算	分布式计算	分布式计算
主要处理模式	实时处理	批处理	实时处理、流处理	实时处理、流处理	批处理
开发语言	Java	Java	Scala	Java	C++
典型应用	联机实时分析	用户行为日志分析	机器学习	社交网络分析	机器学习

从表中我们可以看到，HBase 与 Hive 虽然均用于分布式存储，但是适合的应用场景不尽相同。HBase 是基于 HDFS 列存储的可扩展的分布式数据库系统，主要特点是数据可靠，健壮性好，并发量大。HBase 非常适用于高并发的顺序读写场景，对数据的实时处理较为友好。在这类场景下，系统的时延要求高，数据的更新速度快。而 Hive 作为一款分布式的数据仓库软件，同时也作为一款行之有效的 ETL(Extract Transform Load)数据挖掘工具，它的主要处理模式是批处理模式。这类场景下的数据更新不频繁，实时性要求不高，吞吐量大。用户行为日志分析十分符合这类特点，因此 Hive 适用于这类应用。Spark 与 Storm 均是基于分布式计算对数据进行分析处理的系统。Spark 采用了弹性数据集，在集群中能够实现内存计算，从而使得需要多次迭代的机器学习算法能够在分布式环境下进行。而 Storm 的特点在于十分擅长处理流数据。对于源源不断的数据（例如在社交网络中不断产生的大量数据）能够进行有效的处理。但与此同时也牺牲了一部分容错性。Petuum 作为后起之秀，是一款针对机器学习算法而设计的分布式系统。比起 Spark、Hadoop 等其他分布式系统，Petuum 的性能更为优异，算法收敛速度更快。

4. 大数据处理技术的典型应用

为更好地挖掘大数据中蕴含的宝贵价值，大数据处理技术已被逐渐应用于各行各业。下面列举几个典型应用。

4.1 政府

大数据在政府事务处理上的运用极大地降低了运作成本，提高了生产效率。美国从 2009 年起开放了 40 万联邦政府原始数据集，并宣布采用新“开源政府平台”管理数据。奥巴马政府认为，大数据堪比金矿并称其为“未来的新石油”，也是一种国家核心能力。2012 年，奥巴马竞选连任成功，其中他的数据分析团队无疑是一支重要的力量，它对奥巴马在获取有效选民、投放广告、募集资金等方面起到了一定作用[12]。

此外，美国纽约能源研究和发展管理局运用一系列大数据处理技术来评估气候变化对纽约州的影响，并为各个领域提供了应对气候变化的策略。这一应用也被引入美国疾病控制中心。

4.2 教育

随着人工智能、大数据、云计算等信息处理技术的快速发展，计算教育学作为一个新兴的交叉学科研究方向呼之欲出。最近，华东师范大学孙仕亮教授对计算教育学的定义与研究范畴等进行了前瞻性的归纳和总结[29]。计算教育学定义为运用人工智能等信息处理技术（理论、算法、软件）对过去与现在的教育数据进行定量分析，以发现和揭示教育中的规律，更好地为教育服务。计算教育学的已有研究相对缺乏，还需在实践中不断摸索前进。

计算教育学作为一个新兴学科，有众多的研究内容。下面举几个例子。(1) 智能教学设计。备课是教师教学中一个十分值得关注的内容。传统的教学设计需要教师自行查询书籍或使用互联网查阅信息，这些过程需要大量的人工筛选，耗费了不必要的时间。智能教学设计通过大数据处理技术，例如根据课程的知识点从大数据中检索出相关资料，并尽可能地构建教学环节。教师只需做检查和修改便可形成上课内容，从而使得教学设计自动化，既减轻了教师的负担又丰富了教学内容。(2) 主观题自动评判。在考试评卷中，客观题有固定的答案，因此可以轻易地通过计算机实现自动评阅。然而主观题的自动评阅难以实现。主观题的自动评阅需要依靠大数据分析与自然语言处理技术，以给出相对客观的评分，提高评卷效率。(3) 平安校园建设。校园“霸凌”行为时有发生，这对学生的身心发育和学习都会造成不良影响。通过对大量视频监控数据的处理，对霸凌行为进行有效检测和预警，可以更好地维护校园秩序以及保障学生的健康成长。

4.3 交通

在大城市中，交通拥挤、交通事故层出不穷，智能交通系统成为了改善交通的关键所在。人们希望准确地获取交通数据并构建合理的处理模型。其中，车辆的检测与追踪是一类较为复杂的流处理问题。它通过分析路面摄像头拍摄的数据，分析出车辆的各项交通流量信息，如车速，车流密度等，从而减少人工监测的工作[30]。以丹东某交警项目为例，该项目的卡口电警设备每天产生将近 300 万条过车记录，整个系统累积过车图片数目超过 2 亿条[31]，若不是依托大数据平台的发展，如此大量的数据将难以存储和处理。

4.4 医疗

医疗行业的信息化发展也离不开大数据技术的支持。人们在医疗过程中产生了大量的数据，例如病人保健，治疗记录等。以美国健康系统为例，2011 年美国健康数据达 150EB。这些数字化的医学数据有着极其广泛的用途，例如临床决策支持、疾病监测、人口健康管理等。人们已经开始利用大数据挖掘和分析技术来减少医疗浪费并改善医疗效果。例如，大数据分析每年可以使美国节约大约 3 千亿美元的保健费用。在医疗数据上大规模分析还可以减少浪费，改善医疗效率[32]。

相比传统医疗信息系统，大数据处理技术提升了工作效率，解决了传统方法难以分析处理医院日常产生的海量健康数据的问题。但是目前医学数据的搜集、分析还不够便捷，对数据的处理还需要一定程度的人工干预，例如，不同区域卫生机构之间的数据存在异构问题。如何获取各医院的数据，如何整合不同医院的数据都是需要值得关注的问题。

5. 结束语

随着云计算、物联网等信息技术的发展，大数据分析也被逐渐应用于各行各业中，在政府、教育、交通、医疗等方面都有望取得显著成效。尽管大数据的出现带来了诸多效益，但同时也带来了数据管理和处理的挑战。在本文中，我们阐述了大数据的基本概念与特征，介绍了大数据的生命周期与计算模式，然后介绍了针对大数据可用的硬件支持，以及几种典型的大数据处理与管理关键技术。最后介绍了大数据处理技术的几种典型应用场景。

总的来说，大数据的发展不是单一学科分支的发展，而将是多种学科的交叉式发展。随着相关技术的不断进步，大数据处理技术也会日益成熟，人们也能从规模和复杂度更大的数据中挖掘到有用的信息，从而为人类社会的发展提供更好的服务。大数据时代已经来临，我们要解决的问题还有很多，希望本文能给读者提供一定的参考。

参考文献

[1] Zwolenski M., Weatherill L. The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things [J]. Australian Journal of Telecommunications & the Digital Economy, 2014, 2(3)

[2] While T. Hadoop the Definitive Guide 4th Edition [M]. O'Reilly Media, Inc., 2015

[3] 余长慧, 潘和平. 商业智能及其核心技术[J]. 《计算机应用研究》, 2002, 19(9): 14-16

[4] Manyika J., Chui M., Brown B., et al. Big Data: The Next Frontier for Innovation, Competition, and Productivity [R]. McKinsey & Company, 2011

[5] Sun X., Gao B., Zhang Y., Towards Delivering Analytical Solutions in Cloud: Business Models and Technical Challenges [C]. The 8th IEEE International Conference on e-Business Engineering, 2011: 347-351

[6] Gantz J., Reinsel D. Extracting value from chaos [R]. IDC, 2011

[7] 大数据的四个典型特征[N]. 中国电子报, 电子信息产业网, 2012, http://cyyw.cena.com.cn/a/2012-12-04/135458292978407.shtml

[8] Barwick H.IIIS: The “four Vs” of Big Data, 2011, http://www.computerworld.com.au/article/396198/iiis_four_vs_big_data

[9] 大数据成信息技术领域热门概念[N]. 人民日报, 2013,

http://news.xinhuanet.com/info/2013-02/23/c_132186670.htm

[10] Assun ção M. D., Calheiros R. N., Bianchi S., el al. Big Data computing and clouds: Trends and future directions [J]. Journal of Parallel and Distributed Computing, 2014, 75:3-15

[11] 吴恩华,柳有权. 基于图形处理器(GPU)的通用计算[J]. 计算机辅助设计与图形学学报, 2004, 16(05):601-612

[12] 中国计算机学会大数据专家委员会,中关村大数据产业联盟. 2014 中国大数据技术与产业发展报告[M]. 机械工业出版社, 2015

[13] Redis. http://redis.io/documentation

[14] Spark. http://spark.apache.org/

[15] Weil S., Brandt S., Miller E., Long D.. Ceph: A scalable, high-performance distributed file system [C]. Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation, 2006: 307-320

[16] 余胜生, 初莹莹, 周敬利等. 基于 RDMA 协议的零拷贝技术研究[J]. 计算机工程与应用, 2004, 40(3):126-128

[17] ON Foundation. Software-defined networking: The new norm for networks. ONF White Paper, 2012

[18] Apache Hadoop. http://hadoop.apache.org/

[19] Hadoop Apache Project. HDFS architecture guide. http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

[20] Apache HBase. https://hbase.apache.org/

[21] George L. HBase the Definitive Guide [M]. O'Reilly Media, Inc., 2011

[22] 英特尔 Hadoop 大数据解决方案白皮书, 2012 http://image.tianjimedia.com/uploadImages/2012/241/57L9S496152H.pdf

[23] Thusoo A., Sarma J., Jain N., el al. Hive - A Warehousing Solution Over a Map-Reduce Framework [J]. Proceedings of the VLDB Endowment, 2009, 2(2):1626-1629

[24] Apache Hive. https://hive.apache.org/

[25] Zaharia M., Chowdhury M., Das T., et al. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing [C]. Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation 2012, 70(2): 141-146

[26] Storm. http://storm.incubator.apache.org/

[27] Xing E. P., Ho Q., Dai W., et al. Petuum: A New Platform for Distributed Machine Learning on Big Data [C]. ACM SIGKDD international conference on Knowledge discovery and data mining, 2015, 1(2): 49-67

[28] Ho Q, Cipar J, Cui H, et al. More Effective Distributed ML via a Stale Synchronous Parallel Parameter Server [C]. Advances in Neural Information Processing Systems, 2013: 1223-1231

[29] 孙仕亮. 计算教育学与十大研究主题[J]. 中国人工智能学会通讯, 2015, 5(9): 15-16

[30] 王圣男,郁梅,蒋刚毅. 智能交通系统中基于视频图像处理的车辆检测与跟踪方法综述[J].计算机应用研究, 2005, 22(9):9-14

[31] 周春梅. 大数据在智能交通中的应用与发展[J]. 中国安防, 2014,(6): 33-36

[32] Transforming Health Care Through Big Data [R]. New York: Institute for Health Technology Transformation, 2013