网络安全与大数据系列丛书

大数据技术原理与实践

主编　辛　阳　刘　治　朱洪亮　孔令爽

北京邮电大学出版社
www.buptpress.com

前　言

随着云时代的来临，大数据(Big Data)也吸引了越来越多的关注。大数据目前已成为IT领域最为流行的词汇，其实它并不是一个全新的概念。早在1980年，著名未来学家阿尔文·托夫勒便在《第三次浪潮》一书中，明确提出“数据就是财富”这一观点，并将大数据热情地赞颂为“第三次浪潮的华彩乐章”。直到现在，大数据在政府决策部门、行业企业、研究机构等地方得到了广泛的应用，并实际创造了价值。

本书较为全面地介绍了大数据相关技术和应用的现状。主要的编写思路是首先介绍概念，然后理解方法，最后结合实际。全书共7章：

第1章主要介绍了大数据的基础概念，包括大数据的定义、由来以及特点，使读者对大数据有一个感性上的认识，为之后的章节打好基础。

第2章主要介绍了面向大数据的分布式存储框架，包括Google的Bigtable和Amazon的Dynamo。从架构、实现和性能等角度进行了分析和比较，使读者了解现有的大数据存储方法与策略。

第3章在第2章的基础上介绍了面向大数据的分布式处理框架，包括Hadoop和Spark。从概况、实现和应用三个方面对两个框架进行了介绍，力求使读者对现有大数据处理框架有较为直观的认识，便于理解大数据分析的原理。

有了前3章的概念介绍，第4章开始进入实践性更强的内容。

第4章主要介绍了信息挖掘的经典算法，包括C4.5、k-means、支持向量机、Apriori、EM、PageRank等算法，结合一些生动的例子，深入浅出地介绍这些算法的工作原理，使读者在遇到实际问题时能够灵活应用。

第5章内容为数据的可视化，将数据或结果通过可视化方法呈现出来，使读者能够更加直观地传达与沟通信息。

第6章涉及大数据与人工智能的联系，主要包括深度学习中的CNN和RNN框架，以及它们在大数据下的工作方式，帮助读者了解人工智能和大数据的关系以及算法实现。

第7章主要介绍了大数据在现实生活中的实际用例，通过具体案例，向读者展示大数据在公安领域的具体应用和作用。

在本书的编写过程中，我们参考了大量相关文献资料，并且借鉴了同行专家的研究成果，听取了同行专家的宝贵意见，在此向他们表示真挚的谢意。

本书的编写和出版得到了北京邮电大学出版社的大力支持，在此表示衷心的感谢。

由于编者水平有限，加上时间仓促，书中疏漏与不妥之处在所难免，敬请有关专家和读者批评指正。

编　者

目　录

第1章 绪 论

当下人类正置身于数据的海洋当中，金融、工业、医疗、IT 等数据与各行各业的发展都息息相关，密不可分。数据和太空资源、自然资源等战略资源的地位同样重要，我们每天网上购物、聊天，使用手机通话，在商场消费，上下班打卡，机场过安检……我们的一举一动都在产生着数据，而我们的日常工作和生活甚至整个社会的向前发展都无时无刻不在受着大量数据的影响。数据潜在的巨大价值，得到了社会各界广泛关注。

这里有国际数据资讯(IDC)公司的一组监测数据：全球的数据量大致每两年翻一倍，估计在 2020 年将达到 35 ZB 的数据量，且以半结构化或非结构化的形式存在的数据将占 85% 以上。数据处理带来的巨大挑战摆在了 IT 专业人员面前。实际上，“大数据”并不是一个新鲜的名词，美国人在 20 世纪 80 年代就提了出来。“大数据”这个词在 2008 年 9 月，“*Big Data : Science in the PetaByte Era*”一文在美国《科学》杂志发表之后，开始了广泛地传播。

1.1 什么是大数据

研究机构 Gartner 给出的定义：大数据指的是只有运用新的处理模式才能具有更强的洞察发现力、决策力和流程优化能力的海量、多样化和高增长率的信息资产。

麦肯锡给出的定义：大数据是指用传统的数据库软件工具无法在一定时间内对其内容进行收集、储存、管理和分析的数据集合。

维基百科给出的定义：大数据指的是所涉的资料量规模十分庞大，以至于无法通过当前主流的软件工具，在适当时间内达到选取、管理、处理并且整理成为有助于企业经营决策的信息。

看得出来，不管在哪种定义下，大数据既不是一种新的技术也不是一种新的产品，大数据只是一种出现在数字化时代的现象，就像 21 世纪初提出的“海量数据”概念一样。但是大数据和海量数据却有着本质上的区别。从字面上讲，“大数据”和“海量数据”都来自英文的翻译，“big data”译为“大数据”，而“vast data”或者“large-scale data”则译为“海量数据”。而从组成的角度来看，大数据不仅包括海量数据所包括的半结构化和结构化的交易数据，还包括交互数据和非结构化数据。Informatica 大中国区首席产品顾问但彬更深入地指出，交易和交互数据集在内的所有数据集都包括在大数据内，它的规模和复杂程度远远超出了用常规技术按照合理的期限和成本捕获、管理并处理这些数据集的能力范围。由此可见，海量数据处理、海量交互数据、海量交易数据将会是大数据的主要技术趋势。

20世纪60年代，数据基本在文件中储存，应用程序直接对其进行管理；70年代，人们构建了关系数据模型，数据库技术为数据存储提供了一种新的手段；80年代中期，由于具有面向主题、集成性、时变性和非易失性特点，数据仓库成为数据分析和联机分析的主要平台，非关系型数据库和基于Web的数据库等技术随着网络的普及和Web 2.0网站的兴起应运而生。目前，各种类型的数据伴随着社交网络和智能手机的广泛使用呈现指数增长的态势，逐渐超出了传统关系型数据库的处理能力的范围，数据中潜在的规则和关系难以被发现，这个难题通过运用大数据技术却能够得到很好的解决，大数据技术可以在能够承受的成本范围内，在较短的时间中，将从数据仓库中采集到的数据，运用分布式技术框架对非关系型数据进行异质性处理，经过数据挖掘和分析，从海量、类别繁多的数据中提取价值，大数据技术将会成为IT业内新一代的技术和架构。

大数据是存储介质的不断扩容以及信息获取技术不断发展的必然产物。有一句名言说道：人类之前延续的是文明，现在传承的是信息。从中能够看出，数据对我们现在的生活产生了多么深刻的影响。

1.2 大数据的特征

业界将大数据的特征归纳为4个“V”：Volume(大量)、Variety(多样)、Velocity(快速)、Value(价值)。

1. 数据体量巨大(Volume)

大数据一般指10 TB(1 TB=1 024 GB)规模以上的数据量。产生如此庞大的数据量，一是因为各种仪器的使用，让我们可以感知到更多的事物，这些事物部分乃至所有的数据都被存储起来；二是因为通信工具的使用，让人们能够全天候沟通联系，交流的数据量也因为机器-机器(M2M)方式的出现而成倍增长；三是因为集成电路的成本不断降低，大量事物拥有了智能的成分。

2. 数据种类繁多(Variety)

如今，传感器的种类不断增多，智能设备、社交网络等逐渐盛行，数据的类型也变得越发复杂，不但包括传统的关系数据类型，还包括以文档、电子邮件、网页、音频、视频等形式存在的、未加工的、非结构化的和半结构化的数据。

3. 价值密度低(Value)

虽然数据量呈现指数增长的趋势，但隐藏在海量数据中有价值的信息没有对应增长，海量数据反而加大了我们获得有用信息的难度。以视频监控为例，长达数十小时的监控过程，有价值的数据可能只有几秒钟而已。

4. 流动速度快(Velocity)

一般来讲，我们所理解的速度是指数据的获取、存储以及挖掘有效信息的速度。但我们目前处理的数据已经从TB级上升到了PB级，因为“海量数据”以及“超大规模数据”同样具有规模大的特点，所以强调数据是快速动态变化的，形成流式数据则成为大数据的重要特征，数据流动的速度之快以至于很难再用传统的系统去处理。

大数据的“4 V”特征表明其数据海量，大数据分析更复杂，更追求速度，更注重实际的效益。

1.3 大数据分析的发展情况

1989年在美国底特律召开的第十一届国际人工智能联合会议专题讨论会上,“数据挖掘中的知识发现(KDD)”的概念首次被提出来。1995年召开了第一届知识发现与数据挖掘国际学术会议,KDD国际会议由于与会人员的不断增多发展为年会。1998年在美国纽约举行了第四届知识发现与数据挖掘国际学术会议,会议期间进行了学术上的讨论,有30多家软件公司展示了自己的产品。例如,SPSS股份公司展示了自己开发的基于决策树的数据挖掘软件Clementine;IBM公司展示了自己开发的用来提供数据挖掘解决方案的Intelligent Miner;Oracle公司展示了自己开发的Darwin数据挖掘套件;此外还有SGI公司的Mine Set和SAS公司的Enterprise等。

IBM、Microsoft、Google、Facebook等知名跨国公司通过大数据技术的发展具备了更强的竞争力。仅2009年一年,通过大数据业务,谷歌公司对美国经济贡献高达540亿美元;2005年以来,IBM耗资160亿美元进行了30余次和大数据相关的收购,使得业绩稳定高速增长。

2012年3月,美国政府公布“大数据研发计划”,旨在改进和提高人们从复杂、海量的数据中获取知识的能力,发展收集、储存、保留、管理、分析和共享海量数据所需的核心技术,继集成电路和互联网之后,大数据成为目前信息科技所关注的重点。

在大数据方面,国内起步稍晚于国外,而且还没有形成整体力量,企业使用数据挖掘技术也尚未形成趋势。不过值得欣慰的是,近几年我国的大数据业务也出现了朝气蓬勃的发展态势。

1993年,我国国家自然科学基金首次支持了对数据挖掘领域的研究项目。1999年,在北京召开的第三届亚太地区知识发现与数据挖掘国际会议(PAKDD)上,收到论文158篇。2011年,在深圳举办了第十五届PAKDD,会议就数据挖掘、知识发现、机器学习、人工智能等领域进行了广泛的交流,反响十分热烈。2012年6月9日,中国计算机学会常务理事会决定成立大数据专家委员会。2012年10月,成立了中国通信学会大数据专家委员会,该委员会是首家专门研究大数据应用和发展的学术咨询组织,促进了我国大数据的科研与发展。2012年11月,在以“大数据共享与开放技术”为主题的“Hadoop与大数据技术大会”上,总结了八个热点问题:数据计算的基本模式与范式、数据科学与大数据的学科边界、大数据特性与数据状态、大数据安全和隐私问题、大数据的作用力和反作用力、大数据对IT技术架构的挑战、大数据的生态环境问题以及大数据的应用及产业链。大会还成立了“大数据共享联盟”,旨在搜集大数据、展示大数据、推动大数据的研究与开发。

目前,国内主要开展的是数据挖掘相关算法、实际应用及有关理论方面的研究,涉及行业较广,包括零售、制造、电信、金融、医疗、制药等行业及科学领域,主要集中在公司、部分高等院校以及研究所,在IT等新兴领域,浪潮、华为、阿里巴巴、百度等企业也纷纷参与其中,强有力地促进了我国大数据技术的进步。

1.4 大数据的相关政策

2015 年 8 月 31 日，国务院印发了《促进大数据发展行动纲要》，首次在国家层面上提出发展大数据产业。

纲要提出，在未来 10～15 年内要逐步实现以下目标：打造精准治理、多方协作的社会治理新模式，2017 年年底前形成跨部门数据资源共享共用格局；建立运行平稳、安全高效的经济运行新机制；构建以人为本、惠及全民的民生服务新体系；开启大众创业、万众创新的创新驱动新格局；2018 年年底前建成国家政府数据统一开放平台，率先在交通、信用、金融、卫生、就业、社保、医疗、地理、教育、文化、科技、资源、农业、环境、安监、统计、质量、海洋、气象、企业登记监管等重要领域实现公共数据资源合理适度向社会开放；培育高端智能、新兴繁荣的产业发展新生态，推动大数据与物联网、云计算、移动互联网等新一代信息技术融合发展，探索大数据与传统产业协同发展的新业态、新模式，促进传统产业转型升级和新兴产业发展，培育新的经济增长点。

因此，纲要提出了加快政府数据开放共享，推动资源整合，提高治理能力；促进产业创新发展，培育新兴业态，助力经济转型；强化安全保障，提高管理水平，促进健康发展三大任务。

纲要还提出，政府数据资源共享开放工程、国家大数据资源统筹发展工程、政府治理大数据工程、公共服务大数据工程、现代农业大数据工程、工业和新兴产业大数据工程、万众创新大数据工程、大数据关键技术及产品研发与产业化工程、数据产业支撑能力提升工程 9 个专项。

其中包括建设形成国家政府数据统一开放平台、医疗、交通旅游服务大数据、工业大数据应用、服务业大数据应用、农业农村信息综合服务、构建科学大数据国家重大基础设施。

根据纲要，到 2020 年，我国将形成一批具有国际竞争力的大数据处理、分析、可视化软件和硬件支撑平台等产品；并且培育 10 家国际领先的大数据核心龙头企业，500 家大数据应用、服务和产品制造企业。

第 2 章　面向大数据的分布式存储系统

如今，信息技术迅猛发展，需要被计算机系统处理的数据量大大增加。与此同时，这些数据还需要在存储系统中有效地保存，这给数据分析和处理带来了保障与便利。分布式存储就是利用网络把数据分散在许多台单独的设备上，易扩展、高性能、高可靠性和使用方便是一个先进的分布式存储系统应具有的几个特征。本章以谷歌的 Bigtable 和亚马逊的 Dynamo 为例介绍分布式存储的最新技术。

2.1　Bigtable

谷歌设计的分布式结构化数据存储系统 Bigtable 被设计用来存储海量的数据：一般是分布在数千台服务器上的 PB(100 万 GB)级数据。适用性广泛、高性能、可扩展和高可用性是当前 Bigtable 已经达到的几个目标。谷歌的 Bigtable 技术已经使用在了 60 多个项目和产品上，其中包括 Web 索引、Google Analytics、Google Earth、Orkut、Google Finance、Personalized Search 和 Writely 等。Bigtable 基本满足了这些产品的不同需求，有的需要配置高吞吐量的批处理，有的要及时响应，及时把数据返回给用户。它们所使用的 Bigtable 集群的配置的差异也很大，有的需要上千台服务器，有的只需要几台服务器。

在许多方面，Bigtable 和数据库类似，它运用了许多数据库的实现策略。内存数据库和并行数据库已经具有高性能和可扩展性，不过 Bigtable 提供了一个与这些系统截然不同的接口。Bigtable 不支持完整的关系数据模型；与之相反，Bigtable 提供了简单的数据模型给客户，运用这个模型，客户可以对数据的分布和格式进行动态控制，并允许用户推测底层存储数据的位置相关性。数据的下标可以是任意字符串的行和列的名字。存储的数据都被 Bigtable 视为字符串，然而 Bigtable 本身并不会去解析这些字符串，客户程序一般会将各种半结构化或者结构化的数据串行化到这些字符串里。通过细心选择数据的模式，控制数据的位置，相关性可以被客户控制。最后，可以利用 Bigtable 的模式参数来决定数据是存放在硬盘上还是内存中。

2.1.1　Bigtable 构件

建立在其余的几个谷歌基础构件上的 Bigtable 使用谷歌的分布式文件系统(Google File System，2.2 节将详细介绍)存储数据文件和日志文件。Bigtable 集群一般在一个共享的机器池中运行，池中的机器还可以运行其他各种分布式应用程序，Bigtable 的进程时常要与其他应用的进程共享机器。Bigtable 依靠集群管理系统来进行任务的调度、机器上的资源管理与共享、机器的故障处理以及机器状态的监视。

Bigtable 内部是以 Google SSTable 格式存储数据文件的。SSTable 是一个排序的、持久化的、不能更改的 Map 结构，而 Map 是一个 key-value 映射的数据结构，key 和 value 的值都是随机的 Byte 串，能够对 SSTable 进行如下几个操作：查找与一个 key 值相关的 value，或遍历某个 key 值范围内全部的 key/value 对。从内部看，SSTable 是一系列的数据块（一般每个块的大小为 64KB，可以配置这个块的大小）。SSTable 利用块索引（一般存储在 SSTable 的最后）来定位数据块，在打开 SSTable 的同时索引被加载到内存。通过一次磁盘搜索可以完成一次查找：首先利用二分法在内存的索引里找到数据块的位置，接着把相应的数据块从硬盘读取出来。同样可以选择把全部 SSTable 都放在内存中，这样就不需要访问硬盘了。

为了能够实现并发控制，Bigtable 还依赖一个称为 Chubby 的高可用、序列化的分布式锁服务组件。一个 Chubby 服务包含了 5 个活动的副本，当中的一个副本被选为 Master，同时处理请求。Chubby 服务只有在大多数副本都正常运行，并且彼此之间可以互相通信的情况下才是可用的。Chubby 使用 Paxos 算法来保证当有副本失效时副本的一致性。Chubby 提供了一个包括小文件和目录的名字空间。每个目录或者文件可以当成一个锁，读写文件全部是原子性操作。Chubby 客户程序库提供对 Chubby 文件的一致性缓存。每个 Chubby 客户程序维护一个与 Chubby 服务的会话。假如客户程序无法在会话到期的时间内重新申请会话时间，那么这个会话就会过期失效。当一个会话失效时，它拥有的锁和打开的文件句柄也就失效了。Chubby 客户程序能够在文件和目录上注册回调函数，当会话过期或者文件或目录改变时，回调函数将通知客户程序。

Bigtable 利用 Chubby 完成如下几个任务：保证在任意给定的时间内最多只有一个 Master 副本在活动；存储 Bigtable 数据的自引导指令的位置；查询 Tablet 服务器，在 Tablet 服务器失效的同时进行善后；存储 Bigtable 的模式信息；存储访问控制列表。如果 Chubby 长时间不能访问，Bigtable 将会失效。

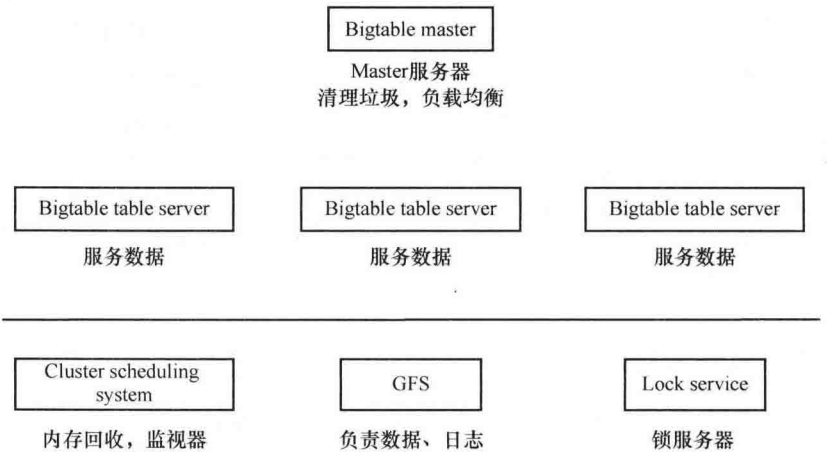

图 2.1.1 Bigtable 框架示意图

2.1.2 Bigtable 实现

如图 2.1.1 所示，Bigtable 的实现主要依赖三个组件：一个 Master 服务器、多个 Tablet 服务器和链接到客户程序中的库。在一个集群中能够动态地删除(或添加)一个 Tablet 服务器来适应工作负载的变化。

Master 主要负责如下的工作：将 Tablet 分配给 Tablet 服务器，检测刚加入的或者过期失效的 Tablet 服务器，平衡 Tablet 服务器的负载，收集 GFS(Google File System)文件中的垃圾。此外，它还可以处理模式修改操作。例如，建立表和列族。

每个 Tablet 服务器都管理一组 Tablet(一般每个 Tablet 服务器大约有数十个甚至上千个 Tablet)。Tablet 服务器对它所加载的 Tablet 的读写操作进行处理，以及对增长过大的 Tablet 进行分割。

客户程序直接与 Tablet 服务器通信来进行读写操作。每个 Bigtable 集群存储了许多表，每个表都将一组 Tablet 包括在内，而每个 Tablet 包括了某个范围内的行的所有相关数据。在初始状态下，每个表只由一个 Tablet 组成。它伴随着表中数据的增长被自动分割成多个 Tablet，在默认情况下每个 Tablet 的大小一般在 100～200 MB 范围内。

2.1.3 Tablet

Bigtable 使用一个三层，类似 B＋树的结构来存储 Tablet 的位置信息，如图 2.1.2 所示。

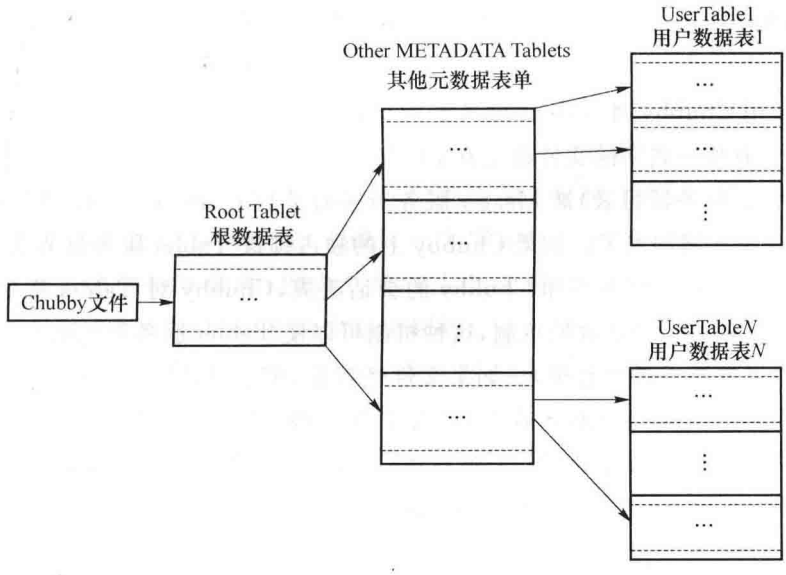

图 2.1.2 Tablet 位置层次结构

第一层是一个包含了 Root Tablet 的位置信息的存储在 Chubby 中的文件。Root Tablet 包括了一个特殊的 METADATA 表里所有的 Tablet 的位置信息。METADATA 表的每一个 Tablet 包含了一个用户 Tablet 的集合。实际上 METADATA 表的第一个 Tablet

是 Root Table，只不过对它进行了比较特殊的处理，Root Tablet 永远不可能被分割，这样就确保了 Tablet 的位置信息存储结构不可能超过三层。

在 METADATA 表里面，每一个 Tablet 的位置信息都存放在一个行关键字下面，而这个行关键字是由 Tablet 的最后一行编码和 Tablet 所在的表的标识符构成的。METADATA 的每一行都存储了将近 1KB 的内存数据。使用这种三层结构的存储模式在一个大小合适、容量限制为 128MB 的 METADATA Tablet 中，能够标识 2^{34} 个 Tablet 的地址（假设每个 Tablet 存储 128MB 数据，则总共能够存储 2^{61} 个字节数据）。

客户程序使用的库可以缓存 Tablet 的位置信息。假如客户程序没有缓存 Tablet 的地址信息，或者发现它缓存的地址信息错误，客户程序就在树状的存储结构中递归地查找 Tablet 位置信息；假如客户端缓存为空，那么寻址算法需要利用三次网络通信来寻址，其中包括一次 Chubby 读操作；假如客户端缓存的地址信息过期了，则寻址算法可能需要最多 6 次网络来回通信才可以将数据更新，因为只有在缓存中未能查到数据的时候才可以发现数据过期（假设 METADATA 的 Tablet 没有频繁的移动）。虽然 Tablet 的地址信息是存放在内存里的，不必访问 GFS 文件系统也可以对它进行操作，但是通常我们会利用预读取 Tablet 地址来进一步降低访问的开销：每次从 METADATA 表中读取一个 Tablet 的元数据的同时都会多读取几个 Tablet 的元数据，次级信息也存储在了 METADATA 表中。

任何时刻，每个 Tablet 只能分配给一个 Tablet 服务器。Master 服务器记录了当前哪些 Tablet 服务器是活跃的，哪些 Tablet 被分配给了哪些 Tablet 服务器，哪些 Tablet 仍未被分配。如果一个 Tablet 未被分配并且恰好有一个 Tablet 服务器有足够的空闲空间装载该 Tablet，那么 Tablet 服务器会收到 Master 服务器发送给它的装载请求，将 Tablet 分配给这个服务器。

BigTable 用 Chubby 将 Tablet 服务器的状态跟踪记录下来。当一个 Tablet 服务器启动时，它将一个有唯一名字的文件建立在 Chubby 的一个指定目录下，同时获得该文件的独占锁。这个目录（服务器目录）被 Master 服务器实时监控着，所以 Master 服务器可以知道有新的 Tablet 服务器加入了。如果 Chubby 上的独占锁被 Tablet 服务器弄丢了。例如，因为网络断开导致 Tablet 服务器和 Chubby 的会话丢失，Chubby 对 Tablet 提供的服务就会停止。（Chubby 提供一种高效的机制，这种机制可以使 Tablet 服务器能够在不增加网络负担的情况下知道它是否还持有锁）。如果文件还存在，那么 Tablet 服务器将会尝试重新获得对该文件的独占锁；如果文件不存在，那么 Tablet 服务器将无法继续提供服务，它将自行退出。当 Tablet 服务器终止时（例如，运行该 Tablet 服务器的主机被集群管理系统从集群中移除），它将试图释放它持有的文件锁。如此一来，Tablet 就能被 Master 服务器尽快分配到其他的 Tablet 服务器。

检查一个 Tablet 服务器是否已经不再为它的 Tablet 提供服务的任务由 Master 服务器负责，而且要尽快重新分配它加载的 Tablet。Master 服务器通过轮询 Tablet 服务器文件锁的状态的方法，来检测什么时候 Tablet 服务器不再为 Tablet 提供服务。假如一个 Tablet 服务器报告它的文件锁丢失了，或者 Master 服务器最近几次试图与它通信都没能得到响应，Master 服务器将试图获得该 Tablet 服务器文件的独占锁；如果 Master 服务器能够成

功获取独占锁，就说明 Chubby 是正常运行的，而 Tablet 服务器不是宕机了，就是无法和 Chubby 通信了，所以，Master 服务器就删除该 Tablet 服务器在 Chubby 上的服务器文件来保证终止它给 Tablet 提供的服务。

一旦在 Chubby 上的 Tablet 服务器的服务器文件被删除了，Master 服务器就把之前分配给它的全部 Tablet 放入未分配的 Tablet 集合中。为了保障 Bigtable 集群在 Master 服务器和 Chubby 之间网络出现故障时依然能够使用，Master 服务器在它的 Chubby 会话过期之后主动退出。但是无论如何，就像前面所描述的，现有 Tablet 在 Tablet 服务器上的分配状态不会因为 Master 服务器的故障而改变。

在集群管理系统启动了一个 Master 服务器以后，Master 服务器要在了解当前 Tablet 的分配状态之后才能够修改分配状态。Master 服务器在启动之时执行如下步骤：①Chubby 让 Master 服务器从它身上获得一个唯一的 Master 锁，以阻止创建别的 Master 服务器实例；②Master 服务器通过扫描 Chubby 的服务器文件锁存储目录来获得当前正在运行的服务器列表；③Master 服务器和全部的正在运行的 Tablet 表服务器通信来获得每个 Tablet 服务器上 Tablet 的分配信息；④Master 服务器通过扫描 METADATA 表获取全部的 Tablet 的集合。在扫描的过程中，如果 Master 服务器找到了一个未曾分配的 Tablet，这个 Tablet 就被 Master 服务器加入未分配的 Tablet 集合等候恰当的时机分配。

也许会遇到一种复杂的状况：在 METADATA 表的 Tablet 未被分配前它无法被扫描。所以，在扫描开始之前(步骤④)，假如在第 3 步的扫描过程中 Root Tablet 被发现未曾被分配，Root Tablet 就被 Master 服务器加入到未分配的 Tablet 集合。这个附加操作保证了 Root Tablet 一定被分配。所有 METADATA 的 Tablet 的名字都包括在 Root Tablet 内，所以 Root Tablet 被 Master 服务器扫描完之后，全部的 METADATA 表的 Tablet 的名字就都得到了。

保存现有 Tablet 的集合只在如下事件发生时才会发生：创建一个新表或者删除一个旧表，把两个 Tablet 合并在一起，把一个 Tablet 分割成两个小的 Tablet。所有这些事件都能够被 Master 服务器跟踪记录，因为除去最后一个事件外其余两个事件都是由它启动的。Tablet 分割事件需要被特殊处理，因为它是由 Tablet 服务器启动。分割操作完成以后，Tablet 服务器通过在 METADATA 表中记录新的 Tablet 的信息来提交这一操作；在分割操作提交以后，Master 服务器会收到 Tablet 服务器的通知。假如分割操作已经提交的信息未能通知到 Master 服务器(也许两个服务器之一宕机了)，已经被分割的子表被 Master 服务器要求在 Tablet 服务器装载时会发现一个新的 Tablet。Tablet 服务器通过对比 METADATA 表中 Tablet 的信息会发现 Master 服务器要求其装载的 Tablet 并不完整，因此，Tablet 服务器将再一次向 Master 服务器发送通知信息。

GFS 上保存了 Tablet 的持久化状态信息，如图 2.1.3 所示。更新操作提交到 REDO 日志中。在这些更新操作当中，最近提交的这些被存放在一个排序的缓存中，这个缓存通常被称为 Memtable；而较早的更新被存放在一系列 SSTable 中。想要恢复一个 Tablet，Tablet 服务器先要从 METADATA 表中读出它的元数据。Tablet 的元数据包括了构成这个 Tablet 的 SSTable 的列表和一系列的 Redo Point，这 Redo Point 指向已提交的有可能包含该 Tablet