文章编号:1671-4598(2014)07-2259-03　　中图分类号:TP312　　文献标识码:A

基于云环境下的海量大数据存储系统设计

费贤举，王树锋
（常州工学院，江苏 常州　213002）

摘要：云计算是目前国际和国内新兴的一项热门技术，正在给人们的生产生活方式带来深刻的变革；在分布式文件系统中，为了实现数据的可用性和持久性，数据通常被分割成大小相同的文件块，通过多副本的形式分布存储在不同地理位置；大型的云计算服务提供商拥有自己的数据中心，这些数据中心往往分布在世界的不同角落，数据以副本的形式分布在不同的地理位置可以实现更好的容错机制，从而提供高可用的持久化的数据存储；文章主要研究了海量大数据存储系统设计，系统设计以 Hadoop 为平台，提出了一种新的基于云计算环境的海量大数据存储设计方法，主要给出了文件存储方案设计以及副本方案设计等，为云计算海量数据存储与管理提供了一种可行的关键技术方案。

关键词：云计算；海量数据；存储系统；Hadoop 平台

Storage and Management of Massive Data in Cloud Computing

Fei Xianju, Wang Shufeng
(Changzhou Institute of Technology, Changzhou　213002, China)

Abstract: The explosive growth of enterprise and scientific data poses a huge challenge to the storage and management of massive data. In recent years, as a data intensive computing platform, Hadoop has attracted much attention from both industry and academy. In the distributed file system, in order to availability and persistent data, the data are usually divided into files of the same size block, through the form of distributed memory multi copies are stored in different geographic locations. Large cloud computing service providers have their own data center, different parts of these data centers are often distributed in the world, the data in different geographic locations can achieve fault tolerance mechanism better as a copy distribution, thus providing the persistence of high availability of data storage. Based on the Hadoop platform, this paper proposes a framework of storage and management of massive data, and analyzes key techniques that we need to implement the proposed framework.

Keywords: cloud computing; massive data; storage; Hadoop

0　引言

云计算是将计算、存储、软件等服务从传统的桌面电脑和服务器通过互联网搬到数据中心的计算模式。这些数据中心包含大量的 IT 基础设施，其所有者往往是大型的 IT 企业，如谷歌、亚马逊、微软、雅虎等。中小企业可以通过网络租用这些服务，而不必搭建自己的服务器系统。数据存储和管理对企业来说非常重要，然而构建安全可靠的企业数据库系统将耗费大量的资金。对初创企业来说，这种按需付费的云计算模式不但可以节约大量的购买相应的软件和硬件的资金，还不用担心软/硬件的维护问题。

在云计算环境下，数据的可用性和持久性对存储服务提供商来说是至关重要的，因为数据的丢失或不可用不但会对客户造成巨大的损失，还会严重影响企业的声誉。在分布式文件系统中，为了实现数据的可用性和持久性，数据通常被分割成大小相同的文件块，通过多副本的形式分布存储在不同地理位置。大型的云计算服务提供商拥有自己的数据中心，这些数据中心往往分布在世界的不同角落，数据以副本的形式分布在不同的地理位置可以实现更好的容错机制，从而提供高可用的，持久化的数据存储。

谷歌公司提出的 MapReduce[1]，作为一种新的编程模型，能有效地并行处理海量的数据，其采用的文件系统和数据管理模式分别是 GFS[2] 和 BigTable[3]。近些年，作为 MapReduce 的开源实现，Hadoop[4] 得到了企业和研究机构的广泛关注。本文基于 Hadoop 平台，提出了一种海量大数据存储系统设计，并分析了实现应用系统需要实现的关键技术。

收稿日期:2014-03-11；　**修回日期**:2014-04-11。

作者简介:费贤举(1975-),男,安徽合肥人,硕士,讲师,主要从事 Web 数据挖掘、图像处理与数据可视化方向的研究。

1　Hadoop 系统环境

云计算是一种商业计算模式，其来源于分布式计算，并行计算和网格计算。云计算将网络中的计算节点看作一个大的资源池。她将网络中的各种资源进行整合，并且应用特定的软件进行资源的自动管理。该方法使系统内的各种资源协同工作，从而能处理海量的数据。Hadoop 作为云计算的核心技术，目前在工业界得到了广泛的应用。

Hadoop 是 Apache 开源组织按照 MapReduce 的工作原理设计的一种开源的分布式处理框架，也是云计算环境下的最著名的开源软件。在 Hadoop 系统中，应用程序可以并行运行在由大规模廉价硬件构成的分布式系统中。Hadoop 在内部实现了容错和扩展机制，可以构建成高可靠性和高扩展性的分布式系统。

Hadoop 主要有三部分组成：HDFS（Hadoop Distributed File System），MapReduce 分布式计算模型，和 Hbase（Hadoop Database），其结构如图 1 所示。

HDFS[5] 与谷歌的 GFS 相对应，部署在廉价的硬件设备上，是 Hadoop 的最底层。HDFS 可以存储 TB 级（甚至 PB 级）的海量数据，并为应用程序提供高吞吐率的数据访问。

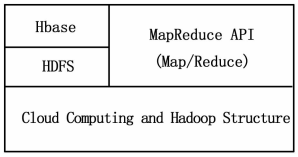


图 1 Hadoop 结构图

HDFS 的数据访问是顺序的，适用于数据密集型的应用。

MapReduce[1] 是一种海量数据处理的分布式计算模型。在集群中运行分布式应用程序时，MapReduce 编程模型简单易用。Hadoop 提供的 MapReduce 编程模型是谷歌 MapReduce 的开源实现。在 MapReduce 编程模型中，开发者只需要编写 Map 和 Reduce 函数，而任务调度、容错等机制由底层实现。因此，即使开发者没有分布式系统的经验也能编写高效的分布式应用程序。

Hbase[6] 是 BigTable 的开源实现。Hbase 构建在 HDFS 之上，提供分布式数据库服务。Hbase 提供一种按列存储的模型，用户可以实时地读写，也可以在大规模数据集上进行随机的访问。

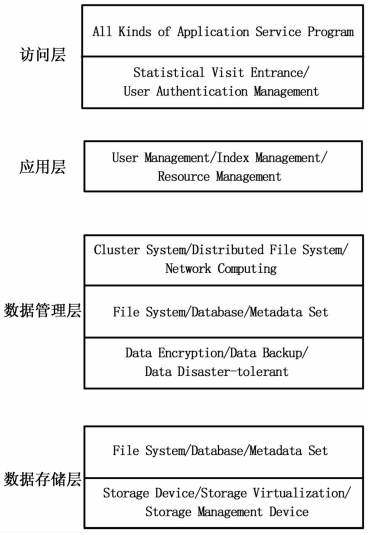


图 2 海量数据存储和管理框架图

2 海量数据存储系统设计

2.1 云存储文件系统的设计

把节点主要分为两种部分，一种是数据节点，另外一种是非数据结点，其中系统中的主要成分都是数据节点（如下图所示的 DataNode 节点），非数据节点主要指管理节点和监控节点在这里我们统一由 Master 节点表示。如图 3 所示。

Client 节点：这个节点主要是指访问的客户，可以是我们虚拟银行的 Web 应用服务器，也可以是其他公司通过我们的系统访问接口进行访问的服务器。

Node 节点：作为系统的主要构成部分，Node 节点负责了系统正常运行的大部分任务，其中包括：数据存储、提供查询、事务处理，并且在必要的时候根据系统的需求提供计算能力。其中所有 Node 节点之间的关系也不完全是相同的，我们根据地域划分，使得同一地域内的节点都是临近几点，基于这样设计有几点考虑，因为系统可能会发展到很大，如果只有一层关系管理节点，将会变得很困难，并且在实际使用中，同一

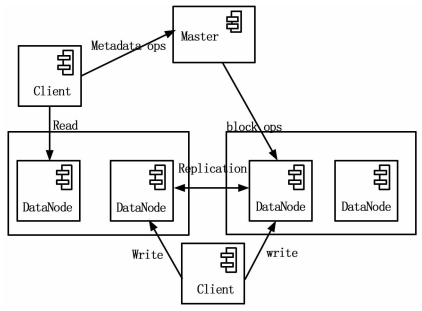


图 3 云存储系统的结构示意图

地域的节点之间的通信单价和质量都是比较好的，所以我们让系统的管理分为 3 层，一个 Master 以每个组的关系看待节点，而节点自己有能够区分是邻居节点（同一组）还是远程节点（不同组的）。

Master 节点：master 主要负责系统的整体状态的监控。其中包括：整个系统的节点状态、提供局部数据节点的查询、保持文件块的地址信息等等。这里需要注意的是，根据系统负载能力的需求我们的 master 节点本身不一定是单个 PC 机器，也可能有几台机器组成一个集群共同提供服务。这样我们才能保证系统不会因为管理节点的瓶颈而受到限制。

2.2 文件块存储策略及副本策略设计

在文件块存储时，我们规定每个文件块都用一个主副本，即每次每个事务处理我们本文件块的所有副本的更新都由主副本控制。基于这种设计我们设计每个文件块除了本身包含的信息之外必须有以下控制信息块：

主副本所在节点编号：每个节点在加入系统时都从 master 那里得到自己的唯一编号并且和自己的地址组成一个节点编号。

副本个数：副本个数包括主副本和其他副本，如果为 1 说明没有其他副本，如果为 0 说明此文件块不存在。

副本所在节点编号列表：保存所有节点编号，在必要的时候可以根据这里的节点编号找到保存了副本的节点的地址和系统编号以进行访问。

在 master 里面有一个根据系统的客户信息生成的一个客户编号的快照，并且有此快照构成系统文件块保存的地址信息的索引，在进行全局查询的时候，master 就是根据这个快照表的信息进行客户信息定位的。然后根据算法把相应的文件块的地址返回到应用服务器，让它自行直接去访问相应的节点。master 快照表结构如表 1 所示。

表 1 Master 客户编号快照表

客户编号	文件块编号
客户编号 2	文件块 a
客户编号 3	文件块 d
客户编号 4	文件块 a
客户编号 5	文件块 e

这里需要说明的是有可能多个客户的信息保存在同一个文件块中所以，文件块出现重复是完全正常的。

除了客户快照表之外，master 还保存了另外一个重要的表，文件块副本表，这个表借用了 google 的 bigtable 的思想，主要包括文件块编号表项和节点信息表项。

表 2　文件块副本信息表

文件块编号	节点信息
文件块编号 d	节点 2：：主副本：：可用 节点 5：：普通副本：可用 节点 8：：普通副本：：可用
文件块 e	节点 6：：主副本：：可用
文件块 a	节点 1：：主副本：：可用 节点 2：：普通副本：：可用 节点 3 ：：普通副本：：不可用

2.3　文件块更新算法设计

采用 Chubby 提供进行文件块更新的锁控制服务，在进行事务处理的时候我们经常会遇到如下一个问题：同一个事务中需要更新的信息不在一个文件块中也不在一个节点中，在这个时候为了保证我们的事务顺利的完成我们需要在多个设计到信息更新的节点中选择一个作为我们的协调节点，由他负责整个事务的更新流程和，决定事务最后的成败，即决定事务最后是成功提交还是失败回滚。

传统的分布式的处理方式是使用比较著名的 paxos 算法，但是我们采用的是 google 的 Chubby 服务机制。之所以采用这种机制有以下几点原因：

（1）大部分开发人员在开始开发 service 的时候都不会考虑到这种一致性的问题，所以一开始都不会使用 consensus protocol。只有当 service 慢慢成熟以后，才开始认真对待这个问题。采用 lock service 可以使得在保持原有的程序架构和通信机制的情况下，通过添加简单的语句就可以解决一致性问题。

（2）很多时候并不仅仅是选举出一个 master，还需要将这个 master 的地址告诉其它人或者保存某个信息，这种时候，使用 Chubby 中的文件，不仅仅是提供锁功能，还能在文件中记录下有用的信息（比如 master 的地址）。所以，很多的开发人员通过使用 Chubby 来保存 metadata 和 configuration。

（3）一个基于锁的开发接口更容易被开发人员所熟悉。并不是所有的开发人员都了解 consensus protocol 的，但大部分人应该都用过锁。

（4）一个 consensus protocol 一般来说需要使用到好几台副本来保证 HA（详见 Paxos 算法），而使用 Chubby，就算只有一个 client 也能用。

Chubby 的结构图如图 4 所示。

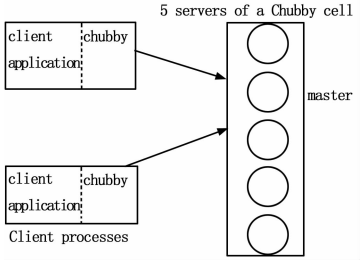


图 4　Chubby 系统结构图

Chubby 一般由 5 台机器组成就足以提供上万台机器的锁服务，而他自己本身的五台机器都是采用完全冗余策略来保证的，在 Chubby 内部采用 Consensusprotocol 协议保证系统的一致性，在每次 5 台机器内部通过此协议选出 master 并且在一定时间后更新 master，在每次数据更新的时候 5 台机器在 master 的控制下同步更新。

Client 和 Chubby 之间采用 event 进行通信，并且为了降低通信频率，client 在本地会保存一个和自己相关的 Chubby 文件的 cache，cache 有两个状态一个有效一个无效，当文件在 chubby 端发生更新的时候 chubby 通知 client 文件无效，然后 client 自己去更新文件。

2.4　事务故障恢复系统设计

云存储系统事务的恢复：对于云存储系统事务来说，因为处于网络环境中，其恢复过程远远要不集中式数据库复杂的多，在云存储系统事务恢复中，本地事务的恢复类同集中式事务的恢复。而整个云存储系统事务的恢复由云存储系统管理器与本地事务管理器协同完成。图 5 是本系统的全局事务恢复模型。

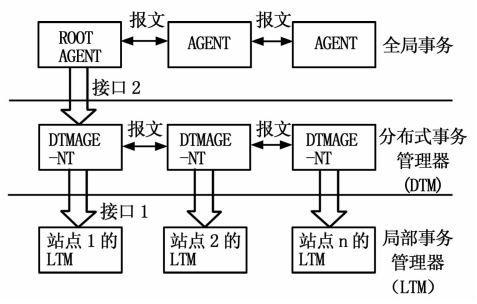


图 5　云存储系统事务恢复模型

3　目录存储与负载均衡设计

系统目录存储和管理是将系统的目录分为若干个组，即前面所提到的按地域分组的方式，在每个组内我们会由 master 指定一个节点专门提供目录服务，它本身也是一个普通节点，只是根据系统的设置成为一个为系统提供目录服务的组服务器，master 同时会将所有存储目录的节点的的地址和编号信息通知所有节点。每个节点根据自己的信息和 master 提供的信息对有目录的节点进行代价排序。然后在目录查询的时候各个节点就能够根据已知目录节点信息进行查询，虽然这个节点的其他查询任务也能执行，但是我们在设计系统负载平衡算法的时候，尽量减轻提供目录查询的节点的其他查询任务。

在地区之间我们把来自客户的访问进行按地区分类，并且根据客户的 IP 信息通过 DNS 进行分流。而在同一地区我们的负载进行轮转法分流本地区的各个访问到本地区的不同的数据节点上，我们保持 master 和 DNS 之间的通信，master 根据节点自身反应的信息对 DNS 中的各个节点的权值进行调整，这样能够及时调整整个系统的负载平衡。

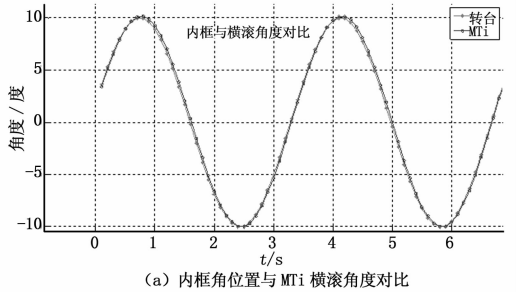
4　结束语

本文首先分析了 Hadoop 云计算平台，在文件系统架构层次中本系统尽量保持云计算平台的各种优势，提出了一种基于 Hadoop 的海量大数据存储系统设计方案。对某些地方比如文件块的可读性特性方面进行了改进。并分析了实现应用系统所需要的关键技术，为云计算海量数据存储于管理提供了一种可行的关键技术方案。

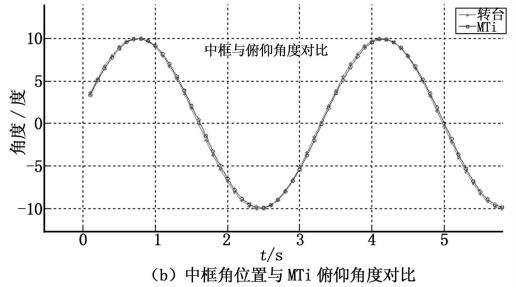
（下转第 2273 页）

（8）采集结束，点击“停止采集”和“关闭”按钮关闭系统。

测试结束后，观察各路串口采集次数，分别为：6 157，6 157，6 157，6 156。这表明：采集时长约为 615.7 s，采集到的转台数据比 MTi 数据多 1 组。可能的原因是：串口初始化和启动监测线程是按串行通道 1、2、3、4 的顺序依次进行的，当 1、2、3 串口初始化和启动监测后，第 4 路串口数据恰好到达，此时应用程序还未完成串行通道 4 的初始化和启动监测线程工作，那么 1、2、3 路数据能够正常接收，但第 4 路数据就会丢失。对保存下来的原始数据，需舍去 1、2、3 串口采集多出来的第一组数据。利用 MTLAB 软件，对调整后的数据文件进行处理和曲线图绘制，得到如图 4（a）、（b）所示结果。



（a）内框角位置与 MTi 横滚角度对比



（b）中框角位置与 MTi 俯仰角度对比

图 4　转台角位置与 MTi 姿态角对比结果

由图 4（a）、（b）可知，MTi 输出角度曲线与转台输出角度曲线具有较高的相似度，同时刻，MTi 输出角度略滞后转台输出角度，显示出 MTi 系统测量的动态响应特性。测试数据验证了本文所提测试方法的有效性，证明该应用程序运行稳定、可靠。

5　结论

实现 SGT320E 型三轴多功能转台和待测负载设备测试数据同步采集的关键问题包括两个方面：（1）两者能够在同一时刻输出测试数据；（2）多路串口测试数据能够由应用程序并行采集和对应保存。本文提出了基于脉冲同步机制的转台与待测负载数据同步输出的方法，用于解决第一个问题；基于改进后多线程串口通讯 CSerialPort 类，开发了多串口数据采集程序，较好地解决了第二个问题。试验测试结果表明，本文所提出的多串口数据同步采集方法适用于 SGT320E 型三轴多功能转台和待测负载设备的测试。

参考文献：

[1] 谭志斌，赵祚喜，张　霖，等. MEMS 惯性传感器的三轴转台实验研究［J］. 电子测量技术，2012，35（4）：110－115.

[2] SGT320E 型三轴多功能转台使用维护说明书［Z］. 中国航空工业集团公司北京航空精密机械研究所，2011.

[3] Remon Spekreijse. A communication class for serial port［EB/OL］. http：//www. codeguru. com/Cpp/I － N/network/serialcommunications/article. php/c2483，2000.

[4] 张　源，卞鸿巍. 组合导航计算机高效多串口通讯技术实现［J］. 计算机测量与控制，2009，17（6）：1163－1165.

[5] 张鸿超，张宏林，等. Windows API 函数参考手册［M］. 北京：人民邮电出版社，2002.

[6] 王艳伟，程　放，周玉成. 多线程 CSerialPort 类的多串口通信实现［J］. 木材加工机械，2012，2：1－4.

[7] MTi and MTx User Manual and Technical Documentation［Z］. Xsens Technologies B. V. 2009.

（上接第 2261 页）

参考文献：

[1] Jeffrey D，Ghemawat S. MapReduce：simplified data processing on large clusters［J］. Communications of the ACM，2008，51（1）：107－113.

[2] Ghemawat，Sanjay，Gobioff H，Leung S T. The Google file system［J］. ACM SIGOPS Operating Systems Review. ACM，2003，37（5）.

[3] Chang，Fay，et al. Bigtable：A distributed storage system for structured data［J］. ACM Transactions on Computer Systems（TOCS），2008，26（2）：4.

[4] 武海平，余宏亮，郑纬民，等. 联网审计系统中海量数据的存储与管理策略［J］. 计算机学报，2006，29（4）：618－624.

[5] 崔　杰，李陶深，兰红星. 基于 Hadoop 的海量数据存储平台设计与开发［J］. 计算机研究与发展，2012，49（s1）：12－18.

[6] 刘树仁，宋亚奇，朱永利，等. 基于 Hadoop 的智能电网状态监测数据存储研究［J］. 计算机科学，2013，40（1）：81－84.

[7] Zeng W Y，et al. Research on cloud storage architecture and key technologies［A］. Proceedings of the 2nd International Conference on Interaction Sciences：Information Technology，Culture and Human［C］，ACM，2009.

[8] Grossman，Robert L，et al. Compute and storage clouds using wide area high performance networks［J］. Future Generation Computer Systems 25. 2（2009）：179－183.

[9] Mark W. Storer Kevin Greenan Darrell D. E. Long Ethan L. Miller. Secure Data Deduplication［A］. StorageSS’08［C］，October 31，2008，Fairfax，Virginia，USA. 2008，1－10.

[10] 童晓渝，张云勇，房秉毅，等. 大数据时代电信运营商的机遇［J］. Science，2013，3：4.

[11] 汤　羽，王英杰，范爱华，等. 基于 HDFS 开源架构与多级索引表的海量数据检索 mDHT 算法［J］. 计算机科学，2013，40（2）：195－199.