第 38 卷 第 4 期
Vol.38 No.4

计 算 机 工 程
Computer Engineering

2012 年 2 月
February 2012

·开发研究与设计技术· 文章编号：1000—3428(2012)04—0260—03 文献标识码：A 中图分类号：TP391

分布式系统数据一致性和并发性优化研究

蒋卫寅，李 斌，凌 力

(复旦大学通信科学与工程系，上海 200433)

摘 要：为满足云存储的高数据读写吞吐量需求，提出一种用于分布式系统的数据锁和快速定位数据锁方法。该锁结构通过细化数据覆盖的颗粒度，在确保数据一致性的前提下允许数据共享。以吞吐量和冲突率为指标进行仿真实验，结果证明，该数据锁定位方法可以向上层提供高性能的随机数据存取服务。

关键词：分布式系统；云计算；哈希；B+树索引；数据锁；并发性

Research on Data Consistency and Concurrency Optimization of Distributed System

JIANG Wei-yin, LI Bin, LING Li

(Department of Communication Science and Engineering, Fudan University, Shanghai 200433, China)

【Abstract】A novel lock scheme applied in distributed systems is proposed in order to fulfill the requirement of huge data throughput. The lock structure refines the granularity of data coverage and allows data sharing in the premise of data consistency while the lock localization algorithm decreases the time to look up lock objects. These methodologies are evaluated in terms of throughput benchmark and lock collision ratio, and simulation result demonstrates that with these optimizations the storage layer can provide high-performance random access to data retrieval requests from upper layer.

【Key words】distributed system; cloud computing; Hash; B+ tree index; data lock; concurrency

DOI: 10.3969/j.issn.1000-3428.2012.04.085

1 概述

随着互联网信息的爆炸式发展，数据存储量每年以指数级的速度增长。中小型企业投入了大量财力购置服务器，带动了信息存储领域的发展。服务器数量的增长，不可避免地引起了诸多方面的困扰，如电力耗费、碳排放量增长、水污染。然而，这些花巨资购买的服务器实际并未得到高效的利用，很多服务器在夜间处于长期闲置状态。云计算[1-2]在这个背景下应运而生，旨在为企业集中统筹规划数据中心，将服务器更高效地利用起来，减少庞大的设备开支。如 EMC 和 IBM 等设备商以流量或租赁时间计费，提供数据存储服务，降低了中小企业和政府部门的信息设备开支。要提供商业化的集群式(SDC)数据服务，对数据存储技术有苛刻的要求：(1)数据完整性。确保服务器之间的数据同步，用户事务之间不相互影响。这是最基本的服务指标。(2)系统稳定性。当服务器集群某个节点出现硬件故障，整个系统不能受影响，数据应该可以及时恢复。(3)接入安全性[3]。客户的数据是保密的，不能被窃取。

本文针对数据完整性和并发性，提出一种用于分布式系统的数据锁和快速定位数据锁的方法，优化数据锁结构设计，提高查询速度，解决了文献[4-6]方法维护代价高等问题。

2 分布式存储架构

图 1 是分布式服务器网络拓扑图。网络存储的信息均衡分摊给每个服务器节点。接入控制服务器受上层应用控制，向集群发送访问请求。索引服务器从集群存储器中提取建立索引信息，提供快速内容定位。索引可以是 B+树索引(B+ Tree Index)、哈希索引(Hash Index)或倒排索引(Inverted Index)。数据复制服务器(Replication Server)提供数据迁移转储功能。备份服务器(Backup Server)为存储集群做数据和日志备份。

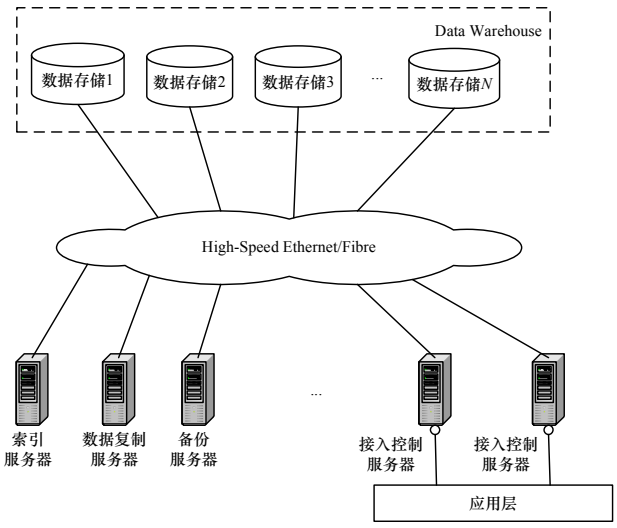

图 1 分布式存储网络拓扑

3 数据锁结构

3.1 锁状态兼容

表 1 为数据请求状态兼容表。以覆盖数据块面积即颗粒度划分，可以将锁分为主机、磁盘逻辑卷和 I-Node 节点锁；以类型划分，可分为互斥锁和共享锁。读数据请求共享锁，写数据请求互斥锁。

作者简介：蒋卫寅(1986－)，男，硕士研究生，主研方向：无线传感网络；李 斌，硕士研究生；凌 力，副教授

收稿日期：2011-07-27 **E-mail：**082021078@fudan.edu.cn

表 1　锁状态兼容情况

现有锁类型	请求锁类型					
	EX_HOST	SH_HOST	EX_VOL	SH_VOL	EX_INODE	SH_INODE
EX_HOST	×	×	×	×	×	×
SH_HOST	×	√	×	√	×	√
EX_VOL	×	×	×	×	×	×
SH_VOL	×	√	×	√	×	√
EX_INODE	×	×	×	×	×	×
SH_INODE	×	√	×	√	×	√

3.2　锁逻辑结构

图 2 是锁的结构，服务器 ID 做 Hash 映射 Bucket *K*= HASH(Hi)，得到桶地址，Hash 表的大小为大于存储服务器节点数量的任意质数。每个哈希入口挂载了一条拉链，拉链上的节点对应于该服务器上某块数据的锁对象(LockObj)，每个节点都有磁盘逻辑卷(VID)和 I-Node(NID)标识。锁对象是优先级队列的首部，每个队列等待项(WaitItem)又挂载了一个双向链表。第 1 个等待项挂载了 2 个锁请求记录(LockRec)，因为多个查询操作间是共享的，可以挂载于同一个队列项。而第 2 个等待项只挂载了 1 个锁请求记录，因为添加、删除、更新操作与查询操作之间是互斥的。

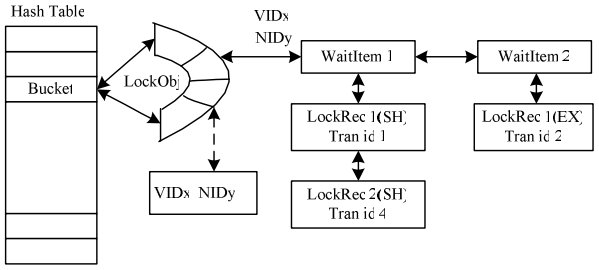

图 2　数据锁结构

请求锁的操作等效于根据数据所在服务器映射到桶(Bucket)，遍历桶上拉链，查询有无对应的锁对象。如果没有，则创建空的锁对象，将锁请求记录挂载于第 1 个等待项；如果已有锁对象，那么根据锁类型的兼容性，在相应位置的等待项挂载锁请求记录。能否得到锁取决于该锁请求记录是否挂载于第 1 个等待项。释放锁操作等效于从数据块对应锁对象删除锁请求记录。

该锁机制的设计同时满足了事务的共享和互斥，且申请和释放锁的算法复杂度为 $O(1)+O(K)+O(P)$。K 为平均每个拉链上锁对象的数量，P 为平均每台服务器上运行的互斥事务数量。第 1 项为 Hash 映射复杂度，第 2 项为遍历拉链复杂度，第 3 项为删除增加锁请求记录的复杂度。

当存储服务器哈希表映射的桶上拉链太长，锁对象数量大于阈值时，可以增大锁的覆盖面积(Lock Upgrade)，降低锁的颗粒度。

3.3　全局锁和本地锁

接入控制服务器和存储服务器处都保存锁信息。前者称为全局锁，后者是本地锁。全局锁完整地记录了存储服务器节点集群的锁状态；本地锁仅保存当前存储节点的锁状态，是全局锁的一根拉链，见图 3。这 2 张表需要同步，通常本地锁能更好地反映当前节点实时的锁状态。接入服务器根据全局锁的松散程度，可以切分控制应用层的访问请求，为读写繁忙的服务器减轻负载。

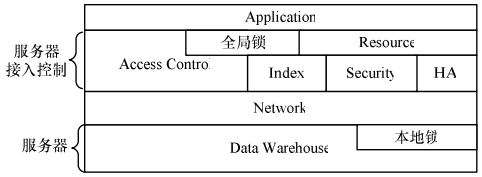

图 3　模块抽象层次

4　基于 B+树的全局锁分布

存储节点的数量随信息量的爆发而急剧增长，这种增长的趋势是不收敛的。存储节点的增加导致全局锁表(GLT)的规模增长，将全局锁切割为 GLT1, GLT2,…, GLT*K* 并分摊于多个接入控制服务器。考虑到负载过重和服务器可能出现的瘫痪，将全局锁表分布地存储比集中式管理更安全稳定。控制服务器的数量不可避免地也会增加，有个潜在的不稳定因素，即如何高效地定位目标存储节点的锁对象。遍历是一种最直观的方法，其复杂度为线性，但随着控制服务器的增多，终会成为系统的瓶颈。B+树是更好的解决方案，它是平衡树，从根到任意树叶的距离是相同的，复杂度为 $O(\log_n N)$，控制服务器数量的增长对于检索时间影响甚微。

图 4 示意了全局锁的分布以及在索引服务器中维护的构建在全局锁上的 B+树锁索引。

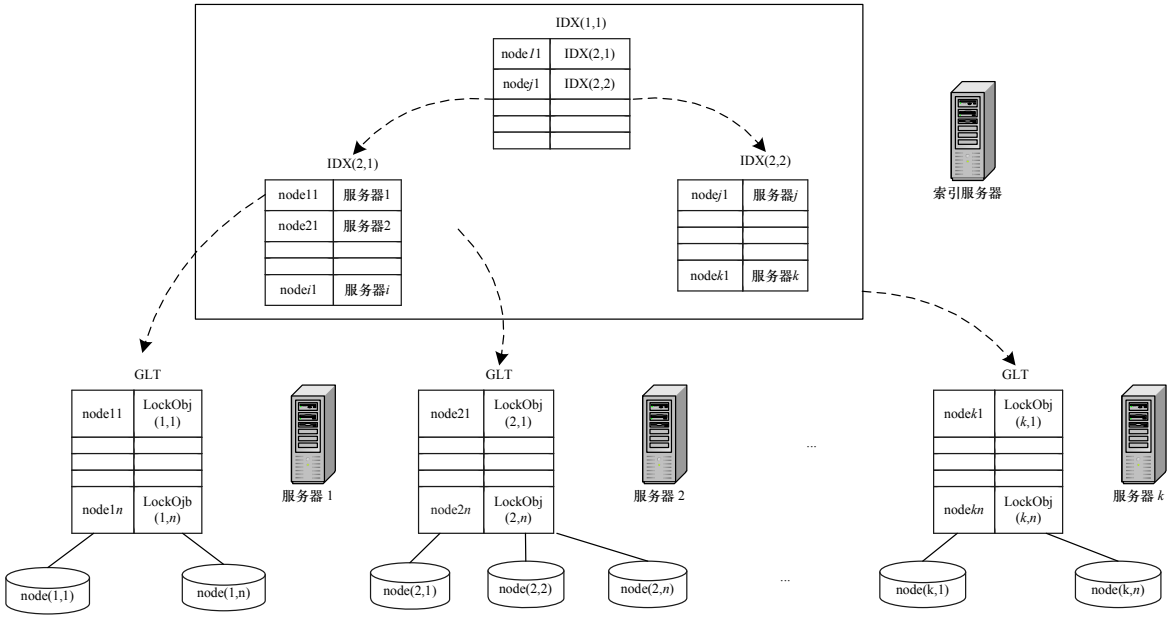

图 4　数据锁分布和锁索引

图 4 中，存储服务器是树的叶子节点；GLT 位于非叶节点的底层；非叶节层以上的节点称为索引节点。父节点表项来源于子节点第 1 个行，并指向子节点。查找定位目标 GLT，需要从根节点往下遍历。存储服务器 ID 是按序的，比较目标 ID 与表项关系，找到更低层的索引节点。以此类推，最终可以找到该存储服务器对应 GLT 所在的接入控制服务器。

5 仿真结果

5.1 锁请求分布和并发性测试

图 5 是对某存储节点进行随机的压力请求测试获得的结果，锁的颗粒度为 I-Node 级别。节点上的 2 048 个锁对象共承受了 13 000 次锁请求。每个锁对象有 6.35 个锁请求，而最多的锁请求是 62 个。从数据可以发现，多数锁对象(>90%)上等待的锁请求数量控制在 10 个以内。锁请求分布越松散，碰撞冲突的概率就越低。

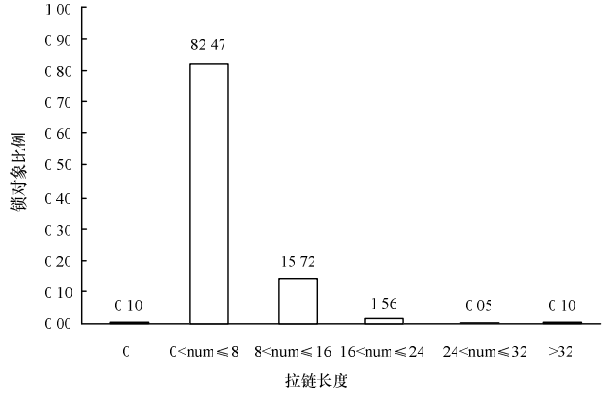

图 5 锁对象负载情况分布

用同样的请求压力对吞吐量性能做进一步分析。分 2 个阶段将锁的颗粒度定为逻辑卷级别和 I-Node 级别，测试结果见表 2。粗颗粒度虽然可以减少锁对象数量，但过多的锁等待有限的资源，导致死锁和发生冲突的概率非常高。当颗粒度细化为 I-Node 级别，冲突率急剧降低，系统的并发性得到明显改善。

表 2 冲突检测实验结果

方法	授予	等待	死锁	冲突率/(%)
逻辑卷共享锁	9 401	532	4	5.36
逻辑卷互斥锁	2 432	898	11	26.97
I-Node 共享锁	9 906	27	2	0.27
I-Node 互斥锁	3 288	42	0	1.26

5.2 锁对象检索比较

仿真比较了遍历和用 B+树定位查找锁对象的时间花费。存储节点数量不断增长导致需要更多的接入控制服务器，从而全局锁表更为分散。从图 6 的查询速度比较可以知道，遍历定位的查询定位时间是随全局锁表的分散而线性增长的。而借助 B+树索引可以将查询开销时间控制在常数时间。

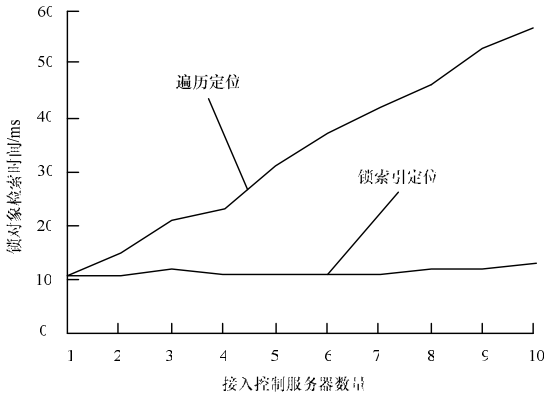

图 6 锁对象查找速度比较

6 结束语

本文研究了当今分布式存储的发展现状和主流的架构，在此基础上提出一种用于分布式系统的数据锁和快速定位数据锁的方法，满足了系统数据一致性和并发性的要求。此外，改进了全局锁定位方法，使得能够适应不断膨胀的分布式网络规模。在实验仿真阶段，对系统进行压力测试，分析了影响并发性能的一系列因素。今后将进一步优化该方案。

参考文献

[1] Buyya R, Yeo C S, Venugopal S. Market-oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities[C]//Proc. of the 10th IEEE Int'l Conf. on High Performance Computing and Communication. Dalian, China, [s. n.], 2008.

[2] Sotomayor B, Montero R S, Lorente I M, et al. Virual Infrastructure Management in Private and Hybrid Clouds[J]. IEEE Internet Computing, 2009, 13(5): 14-22.

[3] 麻 浩，王晓明. 外包数据库的安全访问控制机制[J]. 计算机工程, 2011, 37(9): 173-175.

[4] Burns R C, Rees R M, Darrel D E L. Semi-preemptible Locks for a Distributed File System[C]//Proc. of International Performance Computing and Communication Conference. Phoenix, USA: [s. n.], 2000, 397-404.

[5] Jaechun N. Data Consistency Protocol for Distributed File Systems[C]//Proc. of Conf. on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications. Rende, Italy: [s. n.], 2009.

[6] Choi Sung-Chune, Choi Min-Seuk, Lee Chun-Kyeong, et al. Distributed Lock Manager for Distributed File System in Shared-disk Environment[C]//Proc. of the 10th Int'l Conference on Computer and Information Technology. Bradford, UK: [s. n.], 2010.

编辑 张正兴

(上接第 259 页)

[5] Chew W C. Reconstruction of Two-dimensional Permittivity Distribution Using the Distorted Born Iterative Method[J]. IEEE Transactions on Medical Imaging, 1990, 9(2): 218-225.

[6] Hansen P C. Numerical Tools for Analysis and Solution of Fredholm Integral Equations of the First Kind[J]. Inverse Problems, 1992, 8(6): 849-872.

[7] Alifanov O M. Methods of Solving Ill-posed Inverse Problems[J]. Journal of Engineering Physics and Thermophysics, 1983, 45(5): 1237-1245.

[8] Hansen P C, Jensen T K. An Adaptive Pruning Algorithm for the Discrete L-curve Criterion[J]. Journal of Computational and Applied Mathematics, 2007, 198(2): 483-492.

编辑 张正兴