

基于事物内存的分布式编程环境中缓存一致性维护机制

余林琛^{1,2}, 廖小飞²

(1 中国地质大学(武汉) 计算机学院, 湖北 武汉 430074; 2 华中科技大学 计算机学院, 湖北 武汉 430074)

摘 要: 本文提出了基于事务内存的分布式编程框架, 能够有效地减少由事务内存带来的网络带宽消耗, 并且降低事务内存中事务操作的通信延迟. 其中, 重点设计实现了一个新的事物内存缓存数据一致性维护机制. 测试结果表明, 使用该新型基于事物内存的分布式编程框架实现的分布式计算任务具有简单的设计模型和较高的效率, 有效的减少了系统网络带宽需求, 在大规模集群中相对于现有的分布式事务内存系统有 30% 的性能提升.

关键词: 分布式; 编程框架; 事物内存; 缓存一致性

中图分类号: TP393

文献标识码: A

文章编号: 1000-7180(2013)03-0059-04

Cache Coherency Scheme in Transactional Memory Based Distributed Programming Environment

YU Lin-chen^{1,2}, LIAO Xiao-fei²

(1 School of Computer Science, China University of Geosciences, Wuhan 430074, China;

2 School of Computer Science, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract: In this paper, we propose a generalized and scalable distributed programming environment based on transactional memory, which consumes low cost of bandwidth and reduce the latency of transactional operations. In the scheme, a cache coherency scheme has been proposed. Experimentations show that the new scheme has good performance, and has 30% improvement, compared with traditional transactional memory system.

Key words: distributed; programming environment; transactional memory; cache coherency

1 引言

事务内存做为并行程序设计中的一种工具, 在最近几年里, 已经得到了广泛的研究, 其提供的简易的编程模型能够有效地降低并行程序设计的难度. 然而, 目前现有的分布式事务内存系统都会因大量的消耗网络带宽而带来过高的通信延迟, 这使得它们往往难以融入到实际应用中去.

在分布式环境下, 分布式锁的复杂度和开销都远高于普通的锁, 而由分布式锁引发的优先级反转、死锁、护航等问题更难以发现和解决. 将事务内存技术融入到分布式共享内存中, 为分布式共享内存提供访问控制机制, 可以避免分布式锁的使用, 从而降低分布式程序的设计和开发的难度. 近年来, 关于分

布式环境下的事务内存的研究越来越多.

Maurice Herlihy 等人提出了一种弹道缓存一致性协议^[1], 这种协议假设集群节点是层次型的, 当事务需要访问内存时, 将通过类似 DNS 的在层次中进行上溯的方式进行查找, 直到找到位置为止. 在这个协议中, 事务的提交过程很短, 因为在事务提交的时候不会有网络通信过程, 但是, 每当事务需要访问远程数据时, 都会产生额外的网络通信, 因此, 给事务普通的读写操作带来了更大的时延. K. Manassiev 等人实现了在分布式共享内存系统 (Distributed Shared Memory, DSM) 中基于内存页的多版本控制协议^[2]. 使用这种对共享数据的版本控制协议, 能实现多线程访问的同步, 此实现还支持 MySQL 的数据库事务操作. 集群中的每个节点任何时候都维持

收稿日期: 2012-06-08; 修回日期: 2012-07-15

基金项目: 中央高校基本科研业务费专项资金(XUFL110232)

着一份全部共享数据的物理拷贝. 对于只读事务来说,所有的操作都可以在本地完成,因此具有较高的效率.但是,该协议的缓存模型也是基于缓存全局一致性的,不能支持大规模集群,支持的处理器个数仅 10 个左右.由 Christos Kotselidis 等人提出的 TCC 协议^[3]中,事务在提交时使用了懒惰的有效性检测机制.每个准备提交的事务会进入仲裁阶段,在仲裁阶段中,事务将广播它的读写集.其它的并发事务在接收到广播之后,将自身的读写集与之进行比较,发生冲突的事务将被撤销,以保证其他事务的有效性.但是由于标签机制的存在,给 TCC 协议带来了一个瓶颈.每当一个事务需要提交时,都需要和主节点通信去获得一个标签,这样也给事务的提交增加了通信延时. Cluster-STM 系统^[4]是基于 GasNet 通信库上开发的支持大规模集群的软件事务内存系统.它使用 GasNet 来实现远程过程调用,支持全局统一地址和弱的缓存一致性.但是此系统只是一个尝试开发原型系统的项目,例如它支持的任务个数不能超过处理器的数目,不能支持动态创建线程,不支持容错等. GTM 系统^[5]是基于大规模集群开发的分布式事务内存系统,它同时支持 SPMD 和线程级别的并行,并且提供了异步的远程访问接口和异步的远程过程调用.

目前,在分布式事务内存领域的研究大多都建立在现有的分布式共享内存(DSM)系统^[6]或者 UPC 这类基于 PGAS 模型^[7]的分布式编程框架之上.由于 DSM 等分布式共享内存系统或编程框架本身都有一套分布式共享数据一致性的维护机制,而建立在它们之上的分布式事务内存系统也提供了一套更高层次的数据一致性维护机制和并发控制机制,其中就会产生冗余的通信开销,而共享数据的分布策略也很难完全适应事务内存本身的需求.

2 缓存一致性维护机制

本文设计实现了一种高效且简易的基于事务内存的分布式编程框架,它提供了分布式共享数据对象的实现接口、分布式计算任务的实现接口和分布式事务内存的访问接口,这些接口都拥有良好的透明性.其体系结构如图 1 所示,系统由计算调度节点和计算载体节点组成,其中计算调度节点也可以同时做为计算载体节点使用.

本系统的最关键技术之一就是共享数据缓存一致性.对于任何一种分布式系统,对共享数据的缓存是必不可少的,一套好的缓存机制可以显著地减少

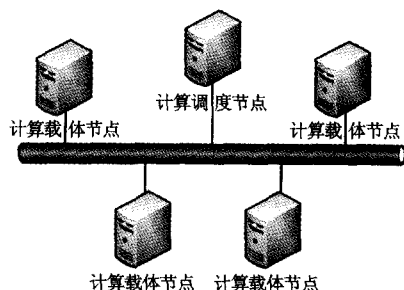


图1 本系统的基本结构

网络通信的次数,提高分布式系统中访存的时延.本课题提出了一套高效的分布式共享数据缓存一致性协议.

当节点对共享数据的远程读操作时,数据的一个副本将被发送到该节点上,如图 2 所示,并在该共享数据宿主节点上元数据的读列表中加入该副本的记录,远程节点也将会在本地的数据缓存中暂时保存此共享数据的副本.在向读列表中加入远程节点上的副本记录时,也将生成一种特殊全局序列号,其中序列号的高 4 位为此远程节点编号,而低 4 位(即节点内部的序列号)设置为零,并将此全局序列号做为远程节点上的副本记录加入到读列表中.当一个节点的缓存中存在有效的数据副本时,针对该数据的读操作均可在该节点本地完成,而不需执行远程的读操作.在这种机制下,共享数据在宿主节点上的读列表中,只可能存在本地的事务访问记录和远程节点上的副本记录,而不会存在远程节点上的事务访问记录;而共享数据在非宿主节点上的读列表中,只可能存在本地的事务访问记录.从而,在系统中,共享数据的读记录实际上是分布在各个节点之上的,而不是集中在其宿主节点上.

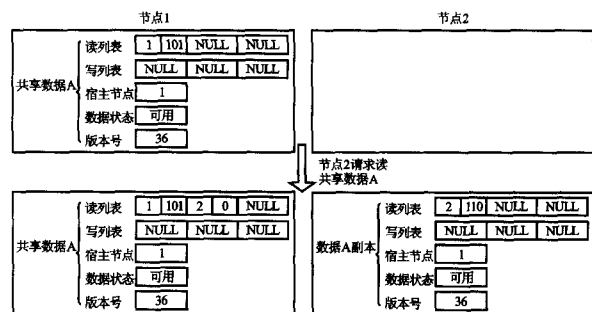


图2 共享数据副本生成机制

在事务内存中,对共享数据的写操作会触发更多的对共享数据原始数据的访问(如提交数据、更新数据等操作),即存在一定的数据局部性.结合事务内存的这一特性,并针对分布式环境下远程数据访问的开销远大于本地数据访问开销的特点,在本课

题的设计中,远程写操作被定义为共享数据迁移的触发条件,即当某个节点执行对共享数据的写操作时,该共享数据将迁移到该节点上。

共享数据的迁移过程不仅仅包括数据本身内容的迁移,还包括其元数据的迁移,即在共享数据迁移的同时,还需要将源节点上的元数据(包括宿主节点、读列表、写列表、版本号等)一同迁移到目标节点,并更新此次迁移操作源节点和目标节点上的元数据。如图3所示,共享数据迁移的具体操作为:首先在源节点上将共享数据元数据中的宿主节点修改为本次迁移的目标节点,并将共享数据以及元数据发送到目标节点;然后清理该共享数据在源节点上的元数据中的读写列表,即删除读写列表中非本地事务访问的记录;在目标节点收到共享数据和元数据之后,使用收到的元数据更新本地的元数据的宿主节点和版本号,并将收到的元数据中的读写列表分别合并到本地的读写列表中;其中,在合并后的读写列表中,可能存在源节点上的事务访问记录,如果存在,则删除这些访问记录,并在读写列表中加入源节点上的数据副本记录。

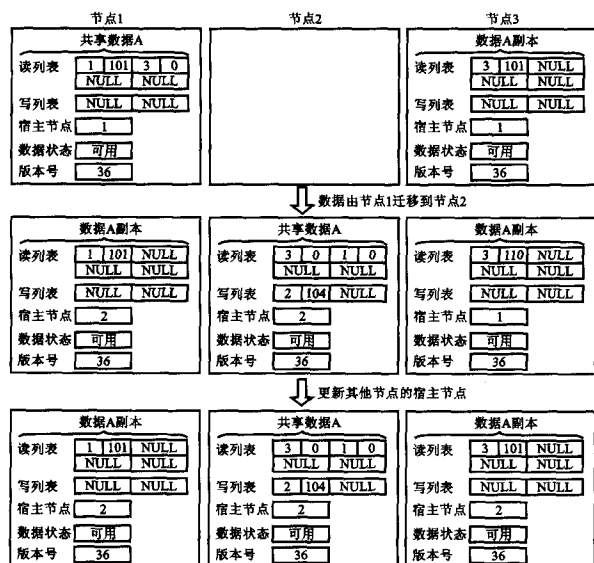


图3 共享数据迁移机制

在一个共享数据迁移之后,还需要向该共享数据的原始节点和存有其副本的节点(即该共享数据元数据读列表中所记录的节点)发送一个消息,该消息中附带本次迁移的目标节点。原始节点和存有其副本的节点在收到这个消息后,将本地该共享对象元数据的宿主节点更新为目标节点,如图3所示,因此在一个共享数据的原始节点和存有其副本的节点上的元数据往往保存的是共享数据当前所在的宿主节点的消息。

当一个共享数据被更新后,则会触发副本失效机制,如图4所示,向所有保存有其副本的节点发送数据失效消息,并且清空其对应的读列表中的记录。当这些节点接收到失效消息后,将会删除该数据在本地的副本信息,并将该数据副本置为失效状态,进而在特定的阶段删除该共享数据副本。

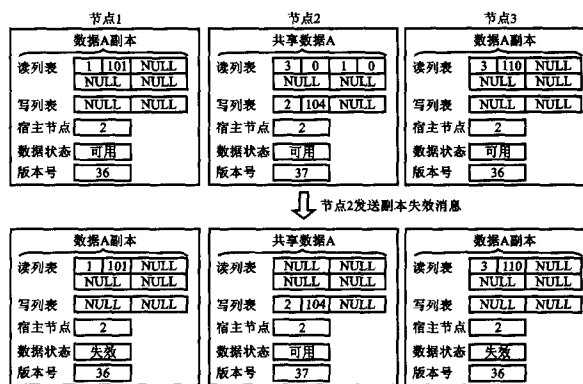


图4 共享数据副本失效机制

3 系统实现与测试分析

为评测所提出了缓存一致性协议的效率,在这里将建立相应的数学模型分析其 cache 命中率。为了方便数学模型的建立,首先需要做出如下几点假设:(1)初始化时,系统中所有的共享数据随机且均匀分布;(2)系统对共享数据的访问操作是完全随机的;(3)系统中的读写操作是按固定比例且随机分布的,并且各个节点上执行的操作数是相等的;(4)各个节点上一次读写操作的时间是相同的,即各个节点在同一时间段内完成的读写操作数目相等。(5)写操作会导致其他节点上的数据副本失效。

设系统中总的共享数据数为 M ,节点数为 N ,写操作占有所有数据操作的比例为 α 。这里将针对系统中的一个节点进行分析,设该节点执行第 x 个读写操作时,cache 命中率为 $f(x)$ 。由于初始化是系统的所有数据均匀分布的,所以有: $f(1) = \frac{1}{N}$ 。根据

上述缓存一致性的原理,当一个节点执行第 x ($x \geq 2$) 个读写操作前,分布在该节点上的共享数据以及副本的总和为 $f(x) \times M$,而在第 x 个操作执行完成之后,由于其它节点的写操作而使得本节点上的部分共享数据和数据副本失效, $(N-1) \times \alpha$ 即为在这次操作的过程中,其他节点上执行的写操作数,而根据数据在系统中均匀分布的假设,本节点上失效的数据数为 $(N-1) \times \alpha \times f(x)$, (由于 M 远大于 N , 故在此处不考虑单个数据多写的情况),而由于本节点执行的操作,本节点上将会增加的有效共享数据

和数据副本的数目为 $1-f(x)$ 。因此可以得出如下递推式:

$$f(x) = \frac{M - (N-1) \times \alpha - 1}{M} \times f(x-1) + \frac{1}{M} (x \geq 2)$$

令 $k = \frac{M - (N-1) \times \alpha - 1}{M}$, $b = \frac{1}{M}$, 上述公式可转化为: $f(x) = k \times f(x-1) + b (x \geq 2)$, 由此递推可得 $f(x) = k^{x-1} \times f(1) + b \times \frac{1-k^{x-1}}{1-k} (x \geq 1)$ 。

而当系统中读写操作数足够大时, 系统的 cache 命中率将趋于稳定。故对 $f(x)$ 取极限, 并将 k 和 b 的值代入式子中, 可得: $\lim_{x \rightarrow \infty} f(x) = \frac{1}{(N-1) \times \alpha + 1}$ 。由上可知, 当系统趋于稳定之后, 系统的 Cache 命中率与系统中的总数据量无关, 而与系统的节点数和写操作数所占比例成反比。使用该数学模型模拟的结果如图 5 所示。当节点数为 1, 或者写操作数所占百分比为 0 时, Cache 命中率将保持为 1; 而当写操作数所占百分比为 1 时, 系统的 Cache 命中率退化为 $1/N$, 即与无 Cache 的情况相同。由此可知, 该缓存命中率达到设计初衷。

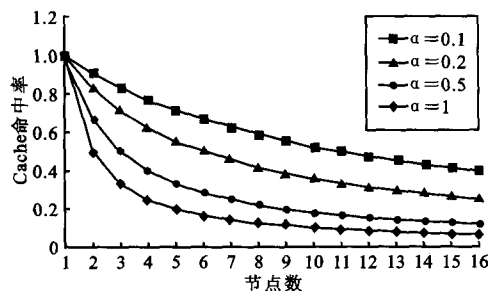


图 5 缓存 Cache 命中率

4 结束语

本文提出了基于事物内存的分布式编程环境中的缓存一致性维护机制。该机制还提高了缓存命中率。未来, 还可继续研究其在实现不同类型的应用

时的效率提升问题。

参考文献:

- [1] Herlihy M, Sun Y, Distributed transactional memory for metric-space networks[J]. Distributed Computing, 2007, 20(3):195-208.
- [2] Manassiev K, Mihailescu M, Amza P. Exploiting distributed version concurrency in a transactional memory cluster[C]// PPOPP'06. New York, NY, USA, ACM 2006:198-208.
- [3] Kotselidis C, Ansari M, Jarvis K, et al. Dstm: a software transactional memory framework for clusters[C]//ICPP'08. USA:Orgeon, 2008:51-58.
- [4] Bocchino R L, Adve V S, Chamberlain B L, Software transactional memory for large scale clusters[C]// PPOPP'08. New York, NY, USA, ACM, 2008:247-258.
- [5] Sridharan S, Vetter J S, Kogge P M. Scalable software transactional memory for global address space architectures[C]// SC'09. Portland; Oregon, 2009.
- [6] Romano P, Rodrigues L, Carvalho N, et al. Cloud-TM: harnessing the cloud with distributed transactional memories[C]// LADIS' 2009. Big Sky Resort, Big Sky, MT, USA, 2009.
- [7] Saha B, Adl-Tabatabai P, Hudson L R, et al. McRT-STM: a high performance software transactional memory system for a multi-core puntime[C]// Proceedings of 11th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPOPP 2006). New York, NY, USA, ACM, 2006: 187-197.

作者简介:

余林琛 女, (1977-), 讲师、博士研究生。研究方向为对等计算。

廖小飞 男, (1978-), 教授、博士生导师。