一种改进的内存数据库封锁机制①

周游弋　(复旦大学 计算机科学技术学院 上海 200433)

摘　要： 事务之间的封锁协议一直是数据库领域中的研究重点。在典型的内存数据库系统中，由于事务通常都比较短小，执行速度快，因此事务之间通常采用类似单写多读这样较为简单的封锁控制协议，将不同事务近似串行地分别执行。在事务并发量较高但是相互之间数据交集不大的情况下，可以通过引入粗粒度意向锁和锁粒度变换的方式来进一步提高事务之间的并行程度，加速事务的整体执行，提高事务吞吐量和系统响应速度。

关键词： 内存数据库；并发控制；封锁机制；粗粒度锁；锁粒度变换

Improved Locking Protocol in Main Memory Database

ZHOU You-Yi (School of Computer Science and Technology, Fudan University, Shanghai 200433, China)

Abstract: Locking protocols between transactions have long been an important field in database research. In a classic main memory database system, transactions tend to be short and small so that the system usually adopts some simple locking protocols like single-write-multiple-read. Transactions are executed in an almost serialized sequence. In a system of heavy transaction concurrencies but relatively small overlapping data sets between transactions, locking protocol could be improved by introducing coarse-grained intent share lock and lock granularity transformation in order to boost concurrencies between transactions, raise transaction throughput and improve overall system performance.

Keywords: main memory database; concurrency management; locking protocol; coarse-grained lock; lock granularity transform

在数据库系统中，事务可以一个个地串行执行，即每个时刻只有一个事务运行，其他事务必须等到这个事务结束之后方能运行。为了充分利用系统资源，发挥数据库共享资源的特点，应该允许多个事务并行地执行。当多个用户并发地存取数据库时就会产生多个事务同时存取同一个数据的情况。若对并发操作不加控制，就可能会存取不正确的数据，破坏数据库的一致性。所以数据库管理系统必须提供并发控制机制。对于内存数据库系统而言，数据加锁开销与处理开销相当，而且由于内存的存取速度比磁盘块的多，事务的执行时间相对磁盘数据库则大大缩短；相应的，锁的占有时间也就大大缩短，故细粒度锁的基本优点与必要性也随之丧失。因此，在内存数据库中一般采用较大粒度锁，如关系级或数据库级，这无疑降低了并发机制的复杂度，减轻了系统负担，从而使系统总体性能提高 [1]。目前在内存数据库系统中普遍采用的是“单写多读”并发控制策略，利用内存数据库事务执行时间短的特点，通过牺牲一定并发度来提高事务执行效率。这种方式在应对大量并发的短小事务时效率并不十分理想，本文通过引入粗粒度意向锁和事务自有锁粒度变换两种方式来改进这种并发控制方式，提高内存数据库在应对高并发短小事务时的整体性能。

1　封锁机制

内存数据库使用封锁机制对并发事务进行控制。确切的控制由封锁的类型决定。基本的封锁类型有两

① 基金项目:上海市科委科研计划项目(08511500902,08511501903)
收稿时间:2010-03-22;收到修改稿时间:2010-05-10

种：排他锁(Exclusive Locks，简称 X 锁)和共享锁(Share Locks，简称 S 锁)。封锁对象的大小称为锁粒度(granularity)。封锁对象可以是逻辑单元，也可以是物理单元。封锁粒度与系统的并发度和并发控制的开销密切相关。选择封锁粒度时应该同时考虑封锁开销和并发度两个因素，适当选择封锁粒度以求得最优的效果 2。在此基础上，为了能够进一步提高并发度和减少锁开销，我们引入了两种意向锁：读意向锁(IS 锁)和写意向锁(IX 锁)。意向锁的含义是如果对一个节点加意向锁，则说明该节点的下层节点正在被加锁；对任意节点加锁时，必须先对它的上层节点加意向锁。引入意向锁之后，事务 T 要对一个数据对象加锁，必须先对它的上层节点加意向锁。申请封锁时候应该按自上而下的次序进行；释放封锁时则应该按自下而上的次序进行。表 1 给出了这些锁的相容矩阵。Y=Yes，表示相容的请求，N=No，表示不相容的请求。

表 1 锁相容矩阵

	S	X	IS	IX	-
S	Y	N	Y	N	Y
X	N	N	N	N	Y
IS	Y	N	Y	Y	Y
IX	N	N	Y	Y	Y
-	Y	Y	Y	Y	Y

2 引入粗粒度意向锁的锁粒度变换

在内存数据库中，由于数据存储在内存中，事务执行时间较短，持锁时间也较短，可以采用粗粒度锁和锁粒度变换的方法 3。除了数据库级读写锁和表级读写锁两种粒度的锁，我们引入额外的一种粗粒度意向锁——数据库级读写意向锁，该锁用于表示事务在数据库级别上的操作意向(读取或者写入)，可以作为其他事务操作数据库上锁之前的重要参考，事务可以根据需要在操作时加不同类型的锁。在此基础上，为了进一步提高并发度，引入了事务自有锁粒度变换的机制，即数据库级锁和表级锁可以根据事务对关系共享程度的需求进行锁粒度的转换。因为数据库级锁比表级锁开销小，因此当不需要较细的锁粒度时使用数据库级锁。如果一个或者多个事务在访问数据库时被其他事务的锁阻塞，数据库级锁会被分解为表级锁，并且再次尝试加锁。当不需要细粒度锁时，表级锁被转化为数据库级锁。如图 1 所示，无论对于读事务还是写事务来说，在事务开始操作数据之前，系统会提前分析事务中涉及到的关系，以便检查相关锁信息。对于读事务来说，事务首先对数据库请求数据库级读意向锁，然后根据数据库的写意向锁信息来判断数据库中是否有写事务正在进行操作，如果存在写意向锁，则说明数据库中有尚未完成的写事务正在修改数据库，此时需要检查前面分析得到的相关数据关系上是否有写锁，若存在一个关系上有写锁，则本次读事务需要等待写事务的完成；若相关数据关系上不存在写锁，则读事务可以锁住相关数据关系进行操作。对于写事务来说，请求锁的过程类似读事务，但是写事务在检测到数据库上有写意向锁的时候，就会阻塞等待，并且只有确定相关数据关系没有读锁的情况下才会对相关关系上锁，也就是说，系统中不会存在同时有两个写事务向数据库写入数据的情况。

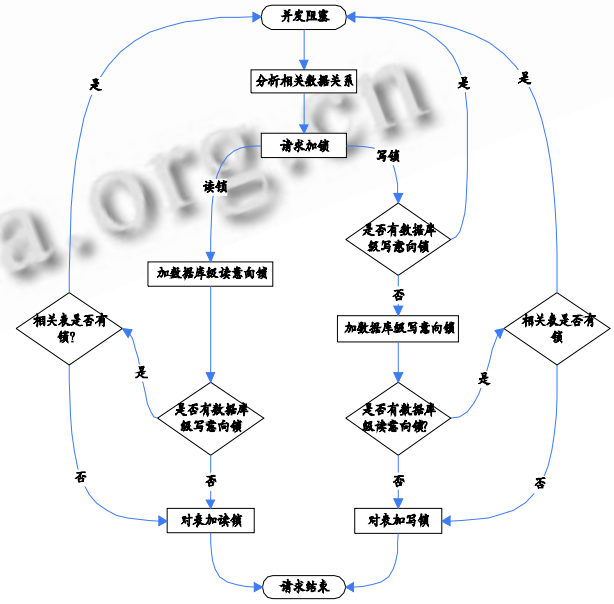

图 1 事务自有锁转换示意图

3 并发控制实现

整个并发控制的过程可以分为加锁和解锁两个阶段。加锁过程由 BeginTransaction()过程实现，解锁

过程由 EndTransaction()过程实现。

加锁过程伪码：

```
int BeginTransaction() {
分析所有需要使用的表;
if (申请读锁) {
if (没有申请过数据库级读意向锁) {
对数据库加读意向锁;
}
if (有数据库级写意向锁) {
if (需要使用的表有锁) {
形成阻塞;
} else {
对相关表加读锁;
}
} else {
对相关表加读锁;
}
} else if (申请写锁) {
if (有数据库级写意向锁) {
形成阻塞;
} else {
if (没有申请过数据库写意向锁) {
对数据库加写意向锁;
}
if (有数据库级读意向锁) {
if (相关表有锁) {
形成阻塞;
} else {
对相关表加写锁;
}
} else {
对相关表加写锁;
}
}
}
}
```

释放锁时，由于采用了一次性封锁机制，释放锁只需要把当前事务锁持有的锁释放掉，然后通知其他事务进入执行即可。

解锁过程伪码：

```
int EndTransaction()
{
释放相关表上的锁;
释放数据库上的意向锁;
通知其他事务进入执行;
}
```

由此可以看出，该协议是一个适用于短小事务的内存数据库系统并发控制封锁协议，它基于单写多读(Single-Write-Multiple-Read)并发策略进行改进，允许多个读事务与单个写事务同时进行操作，但在任意时刻，数据库中只允许有至多一个写事务，所以对拥有大量更新操作的应用它的性能一般，因为当更新事务数量增加时，不同事务之间的竞争条件也在增加，读事务需要更多的等待时间才能获得与写事务不冲突的锁，导致系统整体响应时间增加；对于读事务数量多于写事务的情况则有很好的性能，当写事务数量较少，系统中竞争条件较宽松时，可以有更多的读事务在写事务进行操作时获得不冲突的读锁，加速事务的整体吞吐。为了测试其性能表现，我们利用一个现有课题成果——一套与总参某研究所合作开发的高并发集群监控系统。该系统针对内部应用的特殊性，对一些实时性极高的内部业务集群进行监控管理，对内提供业务逻辑的实时跟踪、动态分配、负载均衡和数据统计等功能，对外提供业务功能接口、人工管理、实时信息展示和数据报表等功能。为了能够应对其应用环境中链接数量大，事务高并发的特点，其内部数据存储采用了内存数据库模块。由于在实际应用中写事务数量不大，而且操作数据较少，执行较快，因此该模块使用了传统的单写多读并发控制策略，任意时刻都只允许一个单独的写事务操作数据库。通过观察发现，该系统中绝大多是短小事务，每个事务访问的数据集相对较小，事务之间数据交集不大。例如在该系统中，内存数据库中记录着各个计算节点的信息、状态以及它们之间的链路状态，更新主要集中于链路状态信息上，而对于节点信息和状态数据主要是查询事务较多，总体上更新事务比例较小，操作的数据集也比较集中，每次更新的数据量也不大，通过利用本文提出的并发协议对该单写多读策略进行修改，对比考察其在不同并发链接数量的情况下，事务并发量，更新事务所占比例和平均响应时间之间的关系，得出结

果如图 2 以及图 3 所示。

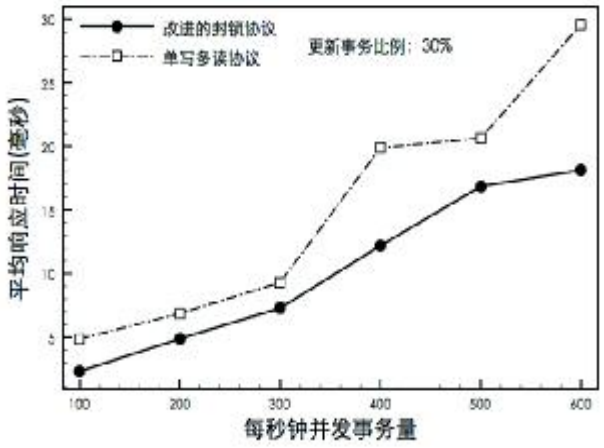

图 2 事务并发度与平均响应时间关系

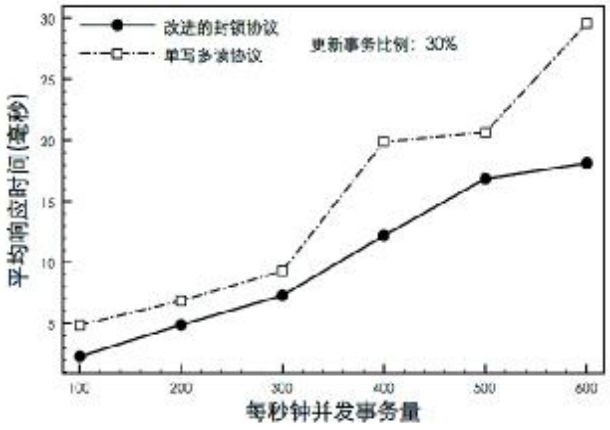

图 3 更新事务比例与平均响应时间关系

图 2 表明在事务之间数据交集不大的情况下，改进后的封锁协议在不同的事务并发场合均优于单写多读协议，因为改进后的协议允许不重叠的读事务和写事务并发执行，加快了系统的整体执行速度。图 3 中，在固定的事务并发度情况下，不同随着更新事务所占比例的提升，单写多读协议中读事务需要等待的时间就越长，整体执行效率就会降低，系统平均响应时间就会增加；利用改进的封锁协议，允许部分读事务和写事务并行操作，减少了读事务的等待时间，提高了系统效率。

4 结语

在与总参某研究所合作开发的高并发集群监控系统中，存在着更新事务比例小、事务短小和事务之间相交数据集不大的应用特点，利用这些特点针对其原有的内存数据库模块中的单写多读并发控制方式进行改进，通过意向锁和事务自有锁粒度转换的方式，允许多个读事务和数据集不相交的单个写事务并行操作，在更新事务所占比例适中的情况下，可以提高内存数据库的事务执行效率和事务吞吐量，在相同时间内促使更多的事务完成，减少平均等待时间，提高系统的平均响应速度。通过实验测试和实际应用中的统计，利用改进后的封锁协议比原有的单写多读协议可以将系统的平均响应时间最高提高 35%，取得了良好的应用效果。

参考文献

1 王珊,肖艳芹,刘大为,覃雄派.内存数据库关键技术研究. 计算机应用, 2007,27(10):2353－2357.
2 萨师煊,王珊. 数据库系统概论.北京:高等教育出版社, 2000:267－268.
3 李国徽,刘云生.实时数据库并发控制.小型微型计算机系统, 2001,22(12):1501－1503.