事务标识的研究与实现*

王意洁 王勇军 胡守仁

(长沙工学院并行与分布处理国家重点实验室 长沙 410073)

摘要 事务标识的分配是影响嵌套事务执行效率的重要因素之一.文章深入分析了在嵌套事务模型下,事务处理对事务标识的需求.以此为基础,提出了一系列实用有效的分配策略.其中,基于位的事务标识分配策略实际应用于自行研制的面向对象数据库系统KDOODB(KeDa object oriented database)中.最后给出了性能测试的结果.

关键词 数据库,事务处理,嵌套事务,事务标识,分配.

中图法分类号 TP311

事务标识(transaction identifier)的分配是影响嵌套事务[1~3]执行效率的一个重要因素,同时也是一个极容易被忽视的因素.事务标识的分配不当将导致占用过多的存储空间和处理机时间.目前,已有越来越多的系统开始提供对嵌套事务的支持,但是它们还没有充分考虑事务标识的分配问题.

我们从事务处理对事务标识的需求入手,结合嵌套事务模型的具体特点,提出了一系列实用、有效的事务标识分配策略:基于位的事务标识分配策略、改进的事务标识分配策略和优化的事务标识分配策略.它们将事务层次结构的有关信息有效地记录于事务标识中.与传统的基于树和哈希表的策略相比,它们兼顾了对存储空间和处理机时间的有效利用,提高了嵌套事务的运行效率.我们通过编制模拟测试程序,对它们进行了比较全面的性能评价.基于位的事务标识分配策略具体在自行研制的面向对象数据库系统KDOODB(KeDa object oriented database)中实现,为KDOODB系统的有效事务管理奠定了坚实的基础.

1 事务处理对事务标识的需求

通常,事务处理系统由事务管理器、恢复管理器、锁管理器、死锁管理器和缓存管理器构成.它们对事务标识的需求可以归纳为:

(1) 可以直接从子事务的标识得到其父事务的标识;

(2) 事务标识对事务层次结构的宽度和广度提供良好的支持(即对宽度和广度没有限制);

(3) 事务标识存储结构能够提供足够多的事务标识(即对事务标识的数目没有限制);

(4) 根据事务标识可以判断两个事务是否具有祖先-后代关系;

(5) 根据事务标识可以找出两个事务的最高层的非共同祖先;

(6) 事务标识存储结构应该灵活,长度可变,能够充分利用存储空间,有效地存储长标识和短标识.

2 事务标识分配策略

2.1 基本策略

我们对事务标识分配策略的研究是以一种传统的基本策略为基础的.在这种基本策略中,事务标识由一个

---

* 作者**王意洁**,女,1971年生,博士,主要研究领域为数据库,网络计算,并行分布处理技术,神经网络.**王勇军**,1971年生,博士,主要研究领域为数据库,并行分布处理技术,虚拟现实技术.**胡守仁**,1926年生,教授,博士生导师,主要研究领域为面向对象数据库,虚拟现实,高性能机体系结构,并行分布处理技术.

本文通讯联系人:王意洁,长沙 410073,长沙工学院计算机系并行与分布处理国家重点实验室

本文1998-05-05收到原稿,1998-08-05收到修改稿

可变长度的整型数组来表示.对于顶层事务而言,表示事务标识的整型数组由一个元素构成,随着事务层次的逐渐降低,表示事务标识的整型数组的元素个数逐渐增加,每当一个新的子事务产生,它的事务标识由其父事务标识和一个用于与兄弟事务区分的新元素构成.这个基本策略比较简单,并且也基本满足了对事务的某些需求(即需求1、需求4和需求5).但是,它的存储空间开销很大.假设采用4字节的长整数,不管位于第N层的事务有多少个,每个N层事务的标识都需要$4\times N$字节表示,这将造成极大的空间浪费.

2.2 基于位的事务标识分配策略

基本策略满足了对事务的某些需求,可以说,它具有一些比较好的特性.但是,它也有一个致命的缺点——大量的存储空间浪费.在"扬长避短"的原则指导下,我们提出了基于位的事务标识分配策略.它对存储空间的使用是以位为单位来考虑,而不是以长整数(4字节)为单位来考虑的,从而有效地避免了存储空间的浪费.

在基于位的事务标识分配策略中,子事务的标识包含了其父事务的标识,一个事务标识由若干个元素构成,每个元素与事务层次结构中的一个层次相对应.每个元素由一个基于位的编码序列表示,编码序列由编码单元组成,各编码单元的值的总和即为元素的值.编码单元的基于位的长度是预先定义的,所以一个编码单元能够表示的数值范围也是预先定义的.当元素的值超过一个编码单元的表示上限(设n是编码单元的长度,即2^n-1)时,将该编码单元置为全满标志(即全零),并分配一个新的编码单元.如果元素的值又超过了两个编码单元的共同表示上限(即$2\times(2^n-1)$),则再分配一个新的编码单元.依此类推,直到满足要求为止.事务标识的第1个编码序列用于表示与元素相对应的编码单元的数目.

实际上,在基于位的事务标识分配策略(如图1所示)下,事务标识由一系列编码单元构成,它们在逻辑上组成了若干个编码序列,每个编码序列表示一个元素.通过判断编码单元的具体内容(即是否设置了全满标志)来确定编码序列的开始与结束:从事务标识的第1个编码单元开始,依次判断每一个编码单元,直到找到第1个未置全满标志的编码单元,到此为止便是第1个编码序列;下一个编码单元标志着第2个编码序列的开始,依此类推.将编码序列中每个编码单元的值相加就得到了相应的值.编码序列可以表示的不同的值的数目为$N\times(2^n-1)$(设N为编码单元的数目).

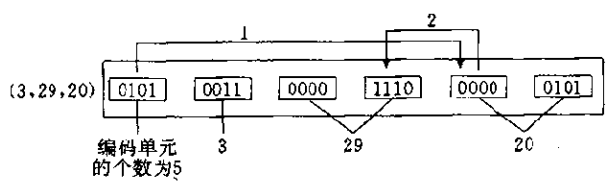

图1　基于位的事务标识分配策略的事务标识结构

基于位的事务标识分配策略满足了对事务标识的各种需求.

需求1——首先,读出事务标识的第1个编码序列,得到与元素相对应的编码单元的数目Sum;然后,读出下面的第$(Sum-1)$个(即倒数第2个)编码单元,判断它是否设置了全满标志:若否,则前$(Sum-1)$个编码单元就表示其父事务的标识;若是,则继续读出第$(Sum-2)$个(即倒数第3个)编码单元并判断它是否设置了全满标志,直到找到未置全满标志的编码单元(设其为第$(Sum-k)$个编码单元),那么前$(Sum-k)$个编码单元表示其父事务的标识(如图1所示);

需求2——由于对编码序列的长度和个数没有任何限制,所以可支持任意宽度和广度的事务层次结构;

需求3——由于对编码序列的长度和个数没有任何限制,所以它提供的事务标识无范围限制;

需求4——通过判断长事务标识是否包含短事务标识即可确定两事务是否有祖先-后代关系;

需求5——从第2个编码序列开始,依次比较两个事务标识的编码序列,直到发现第1个不同的编码序列为止,即找到了两个事务的最高层非共同祖先;

需求6——事务标识的存储结构长度可变,以位为单位进行动态空间分配,而且无需编码单元计数器,因此能够更加充分地利用存储空间,有效地存储长标识和短标识.

2.3 改进的事务标识分配策略

基于位的事务标识分配策略满足了对事务的各种需求,它以位为单位来使用存储空间,比较有效地避免了存储空间的浪费.通过观察我们发现,事务标识的元素值越大,编码序列中全满编码单元的数目越大.实际上,在表示事务标识的元素值时,只需记录编码序列中全满编码单元的数目即可,而不必将全满编码单元一一列出.为

此，我们在基于位的事务标识分配策略的基础上，提出了改进的事务标识分配策略.

与基于位的事务标识分配策略相比，在改进的事务标识分配策略中，对应于每个元素的编码序列由编码单元和编码单元计数器组成. 若元素的值超过一个编码单元的表示上限(设 n 是编码单元的长度，即 2^n-1)时，则将编码单元计数器的值加 1，并将元素的值减去表示上限(2^n-1)；如果修改后的元素值又超过了编码单元的表示上限，则再将编码单元计数器的值加 1，并将元素的值减去表示上限(2^n-1). 依此类推，直到满足要求为止. 我们利用编码单元计数器来记录一个编码序列中的编码单元(包括全满编码单元)的个数，它位于编码单元的前面(如图 2 所示). 编码单元计数器由计数器单元构成，各计数器单元的值的总和即为元素的值，计数器单元的长度也是预先定义的. 当计数器的值超过一个计数器单元的表示上限(设 m 是计数器单元的长度，即 2^m-1)时，则将该计数器单元置为全满标志(即全零)，并分配一个新的计数器单元. 如果计数器的值又超过了两个计数器单元的共同表示上限(即 $2\times(2^m-1)$)，则再分配一个新的计数器单元. 依此类推，直到满足要求为止. 事务标识的第 1 个编码序列用于表示事务的层次(即事务标识的元素个数).

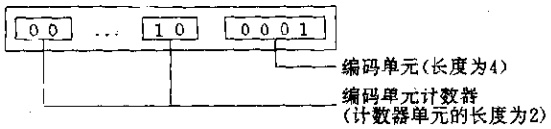

图2　改进的事务标识分配策略的编码序列

为了确定事务标识中某个元素的值，首先，读出元素的前 m 位(m 是计数器单元的长度)，如果计数器单元被设置了全满标志，则继续读出下面的 m 位，直到找到一个未置全满标志的计数器单元，将已读出的计数器单元的值相加就得到了计数器的值 Num；然后，读出下面的 n 位(n 是编码单元的长度)，得到编码单元的值 NU；那么，元素的值为 $(2^n-1)\times(Num-1)+NU$. 编码序列可以表示的不同的元素值的数目为 $k\times(2^m-1)\times(2^n-1)$($k$ 是计数器单元的个数).

改进的事务标识分配策略较好地满足了对事务标识的各种需求.

需求 1——首先，读出事务标识的第 1 个编码序列，得到事务的层次 L；然后，读出下面的 $(L-1)$ 个编码序列，就得到了其父事务的标识；

需求 2——通过扩展编码单元计数器的长度(即增加计数器单元的数目)，可以支持任意宽度和广度的事务层次结构；

需求 3——通过扩展编码单元计数器的长度，可以提供足够多的事务标识；

需求 4——通过判断长事务标识是否包含短事务标识，即可确定两个事务是否有祖先-后代关系；

需求 5——从第 2 个编码序列开始，依次比较两个事务标识的编码序列，直到发现第 1 个不同的编码序列为止，即找到了两个事务的最高层非共同祖先；

需求 6——事务标识的存储结构长度可变，以位为单位进行动态空间分配，因此充分地利用了存储空间，而且能够有效地存储长标识和短标识.

2.4　优化的事务标识分配策略

改进的事务标识分配策略较好地满足了对事务的各种需求，尤其是它具有良好的可扩展性. 另外，它通过省略全满编码单元来节省存储空间，但是，它对编码单元计数器中的许多全满计数器单元并没有进行有效的处理，所以，它还是没有充分利用存储空间. 针对改进的事务标识分配策略的不足，我们提出了优化的事务标识分配策略，有效地处理了全满计数器单元，实现了充分利用存储空间的目的.

优化的事务标识分配策略的主要思想(如图 3 所示)是：元素仍然由基于位的编码序列表示，编码序列由编码单元和编码单元计数器组成；编码单元的取值和表示方式与基于位的事务标识分配策略相同；编码单元计数器由计数器单元和特殊计数器构成，计数器单元的基于位的长度是预先定义的，所以一个计数器单元能够表示的数值范围也是预先定义的. 若编码单元计数器的值超过一个计数器单元的表示上限(设 m 是编码单元的长度，即 2^m-1)，则

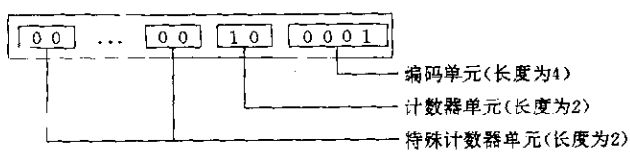

图3　优化的事务标识分配策略的编码序列

将特殊计数器的值加1，并将编码单元计数器的值减去表示上限(2^m-1)；如果修改后的编码单元计数器值又超过了计数器单元的表示上限，则再将特殊计数器的值加1，并将编码单元计数器的值减去表示上限(2^m-1)．依此类推，直到满足要求为止．我们利用特殊计数器来记录一个编码序列中的计数器单元(包括全满编码单元)的个数，它位于计数器单元的前面，如图3所示．

各特殊计数器单元的值的总和即为特殊计数器的值，特殊计数器单元的长度也是预先定义的．当特殊计数器的值超过一个特殊计数器单元的表示上限(设r是特殊计数器单元的长度，即2^r-1)时，则将该特殊计数器单元置为全满标志(即全零)，并分配一个新的特殊计数器单元．如果特殊计数器的值又超过了两个特殊计数器单元的共同表示上限(即$2\times(2^r-1)$)，则再分配一个新的特殊计数器单元．依此类推，直到满足要求为止．事务标识的第1个编码序列用于表示事务的层次(即事务标识的元素个数)．

为了确定事务标识中的某个元素的值，首先，读出元素的前r位(r是特殊计数器单元的长度)，如果特殊计数器单元被设置了全满标志，则继续读出下面的r位，直到找到一个未置全满标志的特殊计数器单元为止，将已读出的特殊计数器单元的值相加就得到了特殊计数器的值Num；然后，读出下面的m位(m是计数器单元的长度)，得到计数器单元的值NC；最后，读出下面的n位(n是编码单元的长度)，得到编码单元的值NU；那么，元素的值为$(2^r-1)\times((2^m-1)\times(Num-1)+NC-1)+NU$．编码序列可以表示的不同的元素值的数目为$k\times(2^r-1)\times(2^m-1)\times(2^n-1)$($k$是特殊计数器单元的个数)．优化的事务标识分配策略也能较好地满足对事务标识的各种需求．

3　性能评价

为了合理评价基于位的事务标识分配策略、改进的事务标识分配策略和优化的事务标识分配策略的性能，我们选择传统的事务标识分配策略作为对比测试对象，通过编制模拟程序而进行了大量的性能测试．在分析测试结果时，我们发现，基于位的事务标识分配策略、改进的事务标识分配策略和优化的事务标识分配策略的性能存在如下规律：

(1) 三者的空间开销呈递减趋势，当事务层次结构较简单时，这种递减趋势不明显；

(2) 三者的时间开销呈递增趋势，递增幅度较小．

鉴于此，这里主要介绍对比测试对象、环境以及基于位的事务标识分配策略的对比测试结果．

3.1　对比测试对象及环境

我们选择一种基于树的传统策略作为对比测试对象，这种基于树的传统策略在PRIMA系统[4]中实现．它的基本思想是：事务由事务控制块表示，事务控制块包括事务标识、指向父事务控制块的指针、指向第1个子事务控制块的指针、指向前一个兄弟事务控制块的指针、指向后一个兄弟事务控制块的指针等信息；每个事务的标识是通过一个计数器依次分配的；树的根对应于顶层事务．

我们利用C++语言实现了传统策略和基于位的事务标识分配策略，运行环境是586微机(166Mhz，32MB内存)．用于测试的事务层次结构的深度取为30．在基于位的事务标识分配策略中，编码单元的长度均取8位．取100 000次运行结果的平均值作为测试结果．

3.2　测试结果

测试内容包括：① 存储空间利用情况；② 获得父事务标识的时间开销；③ 判定祖先-后代关系的时间开销，找出最高层的非共同祖先的时间开销．测试结果如下：

• 存储空间利用情况

为了便于比较，假定每个事务至多有(2^8-1)个子事务．在传统策略中，用于标识事务和维护事务之间关系的存储空间大小是固定的，与事务的层次无关，每个事务需20字节．在基于位的事务标识分配策略中，顶层事务的标识长度是2字节，事务层次每增加一层，事务标识就增加一个编码序列(1字节)．如图4所示，当事务层次小于19时，基于位的事务标识分配策略的事务标识所占用的存储空间少于传统策略的事务控制块所占用的存储空间；当事务层次大于19时，基于位的事务标识分配策略的事务标识所占用的存储空间多于传统策略的事务控

制块所占用的存储空间.

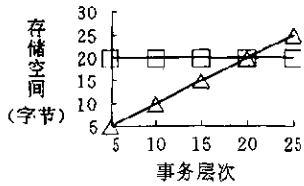

图4 存储空间利用情况

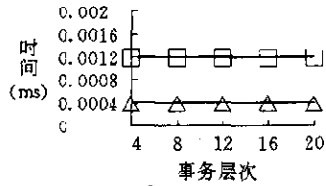

图5 获得父事务标识的时间开销

· 获得父事务标识的时间开销

在传统策略中,为了得到某事务(设其标识为 tid)的父事务标识,首先,找到与 tid 对应的事务控制块;然后,通过其上的指针找到父事务控制块,得到父事务标识.在基于位的事务标识分配策略中,首先,读出第 1 个编码序列,得到事务层次 L;然后依次读出下面的$(L-1)$个编码序列,就得到了父事务标识.对于上述两种策略而言,获得父事务标识的时间开销(如图 5 所示)基本与事务层次无关.

· 判定祖先-后代关系的时间开销

在传统策略中,利用自底向上的方法来判定事务之间的祖先-后代关系,它的时间开销与两个事务之间的层次差额有关.在基于位的事务标识分配策略中,从两个事务标识的第 2 个编码序列开始,依次读取每个字节并进行比较,从而判定事务之间的祖先-后代关系.它的时间开销与两个事务之间的层次差额无关,而与短标识的事务层次有关.为了便于比较,我们取子事务的层次均为 21,通过改变父事务的层次来观察时间开销的变化(如图 6 所示).随着父事务的层次增加,在传统策略中,向上搜索的路径逐渐缩短,因而时间开销减少;在基于位的事务标识分配策略中,比较的字节数增加,因而时间开销增加.

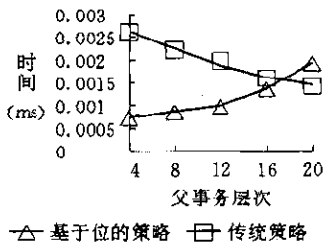

图6 判定祖先-后代关系的时间开销

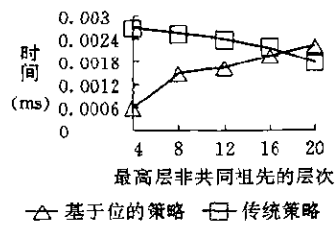

图7 找出最高层的非共同祖先的时间开销

· 找出最高层的非共同祖先的时间开销

在传统策略中(设两个事务 $T1$ 和 $T2$ 的层次分别为 $L1$ 和 $L2$,且 $L1>L2$),首先,通过事务控制块中的指针,找到事务 $T1$ 的位于层次 $L2$ 上的祖先 $A1$;然后,事务 $T2$ 和事务 $A1$ 同时利用指针自底向上搜索,并记录每次搜索到的事务,直到找到第 1 个相同的祖先,那么,上次搜索到的事务便是最高层的非共同祖先.它的时间开销和两个事务与非共同祖先之间的层次差额有关.在基于位的事务标识分配策略中,从两个事务标识的第 2 个编码序列开始,依次读取每个字节并进行比较,直到找到第 1 个不同的字节,从而找出最高层的非共同祖先.它的时间开销与非共同祖先的层次有关.为了便于比较,我们取两个事务的层次均为 21,通过改变最高层的非共同祖先的层次来观察时间开销的变化(如图 7 所示).随着最高层的非共同祖先的层次增加,在传统策略中,向上搜索的路径逐渐缩短,因而时间开销减少;在基于位的事务标识分配策略中,比较的字节数增加,因而时间开销增加.

4 小 结

本文针对嵌套事务模型的执行效率,首先研究了事务处理对事务标识的需求,在深入研究现有事务标识分配策略的基础上,提出了一系列实用有效的事务标识分配策略:基于位的事务标识分配策略、改进的事务标识分

配策略和优化的事务标识分配策略.它们将事务层次结构的有关信息有效地记录于事务标识中,与传统策略的对比测试结果表明,它们兼顾了存储空间和处理机时间的有效利用,能够更好地满足事务处理对事务标识的各种需求,提高了嵌套事务的运行效率.与基于位的事务标识分配策略相比,改进的事务标识分配策略和优化的事务标识分配策略能够更加有效地利用存储空间,但它们的时间开销略有增加,这符合"以时间换取空间"的规律.

基于位的事务标识分配策略具体实现于我们自行研制的客户/服务器面向对象数据库系统 KDOODB 中,它为 KDOODB 系统的有效事务管理奠定了坚实的基础.

参考文献

1 Harder T, Rothermel K. Concurrency control issues in nested transactions. Journal of Very Large Databases, 1993,2(1):33~42

2 Catriel Beeri, Bernstein P A, Goodman N. A model for concurrency in nested transaction systems. Journal of the Association for Computing Machinery, 1989,36(2):230~269

3 王意洁,王勇军,胡守仁.面向对象数据库管理系统中的事务管理.计算机科学,1996,23(6),59~62
(Wang Yi-jie, Wang Yong-jun, Hu Shou-ren. Transaction management in object-oriented database management systems. Computer Science, 1996,23(6):59~62)

4 Harder T, Meyer-Wegener K, Mitschang B *et al.* PRIMA——a DBMS prototype supporting engineering applications. In: Agrawal R, Bell D eds. Proceedings of the 13th International Conference on Very Large Data Bases. San Mateo, CA: Morgan Kaufman Publishers, Inc., 1987. 47~59

Research and Implementation of Transaction Identifier

WANG Yi-jie WANG Yong-jun HU Shou-ren

(*National Laboratory of Parallel and Distributed Processing Changsha Institute of Technology Changsha* 410073)

Abstract The assignment of transaction identifiers is one of the important factors influencing the performance of nested transactions. In this paper, the requirements which the transaction process poses on the transaction identifiers under the nested transaction model are analysed. Based on this, a series of efficient and practical assignment strategies are proposed, and the bit-based transaction identifier strategy is implemented in the object-oriented database system—— KDOODB (KeDa object oriented database). At last, the performance evaluation results are given.

Key words Database, transaction process, nested transaction, transaction identifier, assignment.