基于多核的并行操作转换算法

黎明丽 蔡维纬 吕 晓 何发智
(武汉大学 武汉 400072)

摘 要 操作转换算法是实时协同编辑系统首选的并发控制算法,它不仅能提供不受限的交互,而且维护分布式操作的意图一致性。然而随着操作数目的增多,操作的响应时间也会延长。结合多核多线程技术的发展,提出了第一个并行的操作转换算法,其能减少远程操作集成到本地站点的时间开销。对传统的串行算法进行了改造,使得具有计算依赖的过程能够并行化。实验结果表明,提出的算法相较于传统算法具有较大的优势,在处理较大操作历史的情况下依然能够保证操作合理的响应时间。

关键词 操作转换,并行计算,多核多线程,实时协同编辑,数据一致性
中图法分类号 TP302 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.6.017

Parallel Operational Transformation Algorithm in Multi-core

LI Ming-li CAI Wei-wei LV Xiao HE Fa-zhi
(Wuhan University, Wuhan 400072, China)

Abstract Operational transformation algorithm is the first choice of real-time collaborative editing systems. As a concurrency control strategy, it not only supports unconstrained interactions, but also maintains the intention consistency of distributed operations. However, as the number of executed operations increases, the performance degrades, affecting the responsive time of operations. Combining the development of multi-core and multi-threading, this paper proposed the first parallel operational transformation algorithm, which can greatly reduce the time costs of integrating remote operations. The traditional sequential algorithm is modified, so that the procedure with computation-dependency can be parallelized. Extensive experiments show that the proposed algorithm takes large advantage over the traditional algorithm, and even though the operation history is very large, it still provides a decent responsive time.

Keywords Operational transformation, Parallel computing, Multi-core and multi-threading, Real-time collaborative editing, Data coherence

1 概述

实时协同编辑系统基于 Lesile Lamport(2013 年图灵奖得主)的分布式事件关系[1],采用全复制式结构来提高任务的并行性、响应性和协作性,但也给共享对象的一致性维护带来巨大挑战。基于操作的乐观并发控制方法、操作转换 OT(operational Transformation)算法被广泛应用于各种多媒体文件(Microsoft Office[2]、Bitmap[3]、XML[4]、CAD[5]、3D 图形[6])的协同编辑。

Ellis 在 1989 年提出了第一个 OT 算法(dOPT)[7],其通过转换函数修改操作的执行形式,使并发的操作能够乱序执行。基于 dOPT 算法,史美林等人提出了面向对象数据模型的并发控制方法[8]。Ressel 根据 dOPT 的"puzzle"(某些场景下各个站点的结果不一致)提出了转换函数正确性的充分必要条件(TP1 和 TP2)[9]。Imine 等人通过自动机合成了满足 TP1 和 TP2 条件的非全序转换函数[10]。操作转换的框架包括转换函数和控制过程两个部分,前者确保两个操作的可交换性,后者确定操作的发送、接收和具体转换过程。大部分 OT 算法通过控制过程施加唯一的转换顺序,摆脱了 TP2 条件。在协同编辑环境下,并发操作有可能产生于不同的文档状态,导致操作对象的位置不具有可比性。因此,控制过程的另一个功能是根据操作历史构造出远程操作时产生的文档状态。

OT 能够实现更加严格的一致性。Sun 等人[11]最早提出 OT 算法需要维护操作的意图一致性,然而并没有给出意图明确的定义。廖斌等人提出只有每一步都保持意图,最终的结果才满足意图一致性[12]。Li du 等人确认了这种"立即模式",并明确地将操作的意图定义成操作作用对象之间的位置关系[13]。

OT 根据操作历史来计算出远程操作的执行形式,并保证各站点不同的操作执行顺序产生一致的结果。远程操作的响应时间严重依赖于操作历史的大小。尽管当前效率最高的

到稿日期:2015-07-11 返修日期:2015-09-22 本文受湖北省自然科学基金(2015CFB254)资助。
黎明丽(1971—),女,硕士,主要研究方向为远程数字诊疗系统、协同并行医疗信息系统;蔡维纬(1988—),男,博士生,主要研究方向为实时信息系统、协同计算;吕 晓(1983—),女,博士生,讲师,主要研究方向为实时信息系统、协同计算;何发智(1968—),男,博士,教授,主要研究方向为图形图像处理、协同计算。

OPTIC 算法[14]只需要 O(n)的渐进时间复杂度，但是其不能很好地适应大规模协同编辑环境。

近两年，随着超线程处理器和多核处理器技术的广泛使用，单 CPU 2 核(4 线程)、4 核(8 线程)的个人 PC 已经十分常见。利用多核多线程提高桌面应用程序的性能显得经济可行。据我们所知，本文提出的算法是第一个多核 CPU 环境下的并行操作转换算法，旨在提高集成远程操作的效率，在大规模协同操作的情况下依然能够保持自然和谐的交互。

2 协同编辑模型

协同编辑系统由许多站点通过网络连接在一起，每个站点拥有待加工文档的一个副本。每个站点的用户可以自由地编辑文档的任何一个部分，编辑操作会被发送到其他协同站点；站点接收到远程操作时，首先计算该操作正确的执行形式，然后再修改当前文档。通常，将共享文档建模成一串抽象数据类型的线性集合。每个数据元素可以代表一个字符、一个段落、一个 XML 节点等等。相关研究表明，这种结构很容易扩展到其他复杂的多媒体文档结构。

文档状态仅由两个元操作(插入、删除)来改变，$ins(p,e)$ 表示在位置 p 插入元素 e，$del(p)$ 表示删除位置 p 处的元素。每个站点记录本地产生和接收到的操作，称作操作历史(记为 H)。假设文档初始状态为 DS_0，当前的操作历史为 H_n，则当前文档状态 $DS_n=DS_0\cdot H^n$。每个操作由站点 sid 和逻辑时钟 $clock$(该站点已执行的操作数)来唯一标识操作的 id。

2.1 操作转换的基本原理

两个用户共同编辑文档(初始为“abc”)，站点 1 发起操作 $o_1=ins(2,x)$，站点 2 同时发起操作 $o_2=del(1)$。本地产生的操作立刻执行，然后广播给其他站点。当站点 2 接收 o_1 时，由于 b 已经被删除，操作 o_1 的正确执行形式为 $o_1'=ins(1,x)$。同理，当站点 1 接收 o_2 时，o_1 不会影响 o_2，所以仍然执行 $del(1)$。最终，两个站点都获得了一致的结果，如图 1 所示。

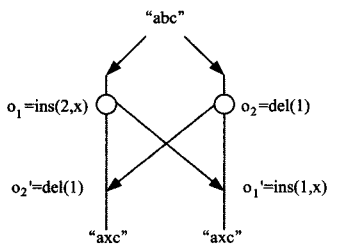

图 1 两个站点的操作转换实例

2.2 转换函数

算法用到两种转换函数，包含转换函数 IT 和排除转换函数 ET。$o_1'=IT(o_1,o_2)$ 包含了 o_2 的执行对 o_1 产生的影响，而 $o_1=ET(o_1',o_2)$ 排除了 o_2 对 o_1 的影响，两者在大部分情况下互为逆过程。关于 ET 和 IT 详细的定义可以参见文献[13]。

2.3 一致性模型

操作转换算法实现协同编辑系统的一致性标准包括：因果一致性、结果一致性、意图一致性。根据 Lamport 的 happened-before 定义，时间上先后关系的操作在分布式站点上的执行也应该保持这种关系。由于每个站点 H 中的操作顺序不一样，简单地执行接收到的操作将会导致不一致的结果。操作转换算法利用转换函数和控制过程改变远程操作的执行形式，保证所有站点在静默状态下的结果一致性。常见的将操作序列化的方法尽管能够得到一致的结果，但是这种人为的顺序往往违背了操作的意图。例如，共享文档 $DS_0=$“abc”，站点 1 执行 $o_x=ins(1,d)$，站点 2 同时执行 $o_y=del(2)$，如果将 o_x 置于 o_y 之前执行，o_y 将会删除字符 b 而不是 c，违背了操作的真实意图。操作转换算法能够较大限度地保持操作的意图，即实现意图的一致性。

3 串行的 OPTIC 算法

3.1 因果依赖关系

基于状态向量(或向量时间戳)定义的因果关系具有很大的局限性，并且很难适应动态的网络环境。事实上，在逻辑时间上具有因果关系的两个操作，在语义上是可以交换执行的，换句话说是可以并发执行的。Imine 等人[14]提出了一种基于操作位置关系的语义依赖关系。

站点的操作历史 H 中两个相邻操作 o_1 和 o_2(o_1 在 o_2 之前)，当且仅当出现以下情况之一时：

1) $o_1.t=o_2.t=ins$；$o_1.p=o_2.p$；$o_1.id\geqslant o_2.id$；

2) $o_1.t=o_2.t=ins$；$o_1.p=o_2.p+1$；$o_1.id\leqslant o_2.id$；

3) $o_1.t=ins$；$o_2.t=del$；$o_1.p=o_2.p$。

称 o_2 因果依赖于 o_1，记为 $o_1\Rightarrow o_2$；否则，o_1 和 o_2 是并发关系，记为 $o_1\parallel o_2$。

从定义可以看出，语义上具有因果关系的操作必定在时间上具有先后关系，反之则不一定成立。因此，语义因果依赖被认为是最小因果关系集，能够极大简化算法集成本地和远程操作的过程。操作的最小因果依赖如图 2 所示。

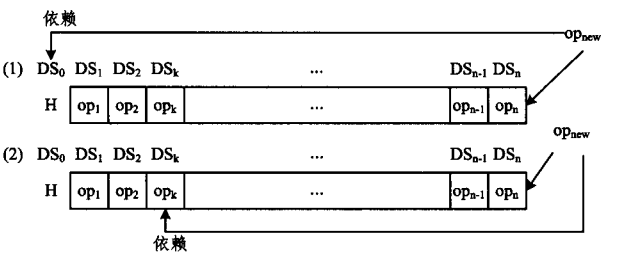

图 2 操作的最小因果依赖

3.2 算法描述

针对本地操作 o，新产生的操作立即被执行，然后计算出它在定义状态下的形式，并广播到其他站点。从右到左扫描 H，调用 ET 逐一排除在它前面执行的操作的影响。如果遇到因果依赖的操作 $H[i]\Rightarrow o$，则停止并记录相关的依赖信息。

算法 1 IntegrateLocal(o)；r

```
1. execute(o); //同时加入 Hi 或 Hd
2. o'=o;
3. for(i=|Hd|-1;i≥0;i--)
4.    o'=ET(o',Hd[i]);
5. endfor
6. if o.t=ins
7.    for(i=|Hd|-1;i≥0;i--)
8.       IT(Hd[i],o');
9. endfor
10. endif
11. for(k=|Hi|-1;k≥0;k--)
```

```
12.  if Hi[k]⇒o then
13.    r=(a,o′);// a=Hi[k]. id,dfx,Hi[k]. p
14.    return r;
15.  else
16.    o′=ET(o′,Hi[k]);
17.  endif
18. endfor
19. return (null,o′);
```

针对远程操作 o，首先判断其因果依赖的操作是否已在 H 中，如果不存在，则将其加入等待队列。假设 $H[i]\Rightarrow o$，则调用 $IT(o,H[i{:}n])$ 得到 o'，执行 o'。显然 o 不依赖任何操作时，$i=0$。

算法 2　IntegrateRemote(r)

```
1. k=0;
2. o′=o;
3. if r. a≠null
4.   Hi[k]⇒o//根据 id 查找到 Hi[k]
5.   o′. p=Hi[k]. p or o′p= Hi[k]. p+1;
6. endif
7. for(;k≤|Hi|;k++)
8.   o′=IT(o′,Hi[k]);
9. endfor
10. if(o. t=ins)
11.   for(i=0;i≤|Hd|−1;i−−)
12.     IT(Hd[i],o′);
13.   endfor
14. endif
15. for(i=0;i≤|Hd|−1;i−−)
16.   o′=IT(o′,Hd[i]);
17. endfor
18. execute(o′);
```

通常远程操作的响应时间只考虑从接收到远程操作到该操作反应到用户 UI 上的时间，也就是 IntegrateRemote 的计算开销。

4　并行的 OPTIC 算法

从第 3 节的算法描述可以看出，算法 1 的 11－18 行和算法 2 的 6－8 行、11－13 行存在循环依赖关系，这给并行化带来了极大的困难。因此，本文提出了并行 OPTIC 算法的计算模型，同时给出算法的具体实现。

4.1　逻辑文档模型

相比用户视图(UView)，逻辑文档模型(LView)保留了被删除的对象，并且置其 visible 属性为 false。用户视图上的操作和逻辑文档上的操作可以互相转换，即 UViewToLView 和 LViewToUView。传统的 OPTIC 算法发送给其他站点的操作是作用在 UView 的，而并行 OPTIC 的操作是基于 LView 的。逻辑文档模型如图 3 所示。

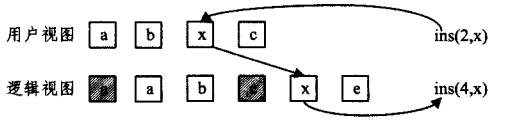

图 3　逻辑文档模型

4.2　位置依赖关系

由于 LView 上的操作不用考虑删除操作的影响，涉及 H_d 的计算步骤都可以移除，而且 LViewToUView 的过程是很容易并行的。

从语义因果依赖的定义可以看出，操作只可能依赖于插入操作，而且操作之间具有特殊的位置关系。假如 H 中存在远程操作 o_1 的依赖操作 o_2，在 LView 上可以快速地计算出 o_1 的位置参数($o_1.p=o_2.p$ 或 $o_2.p+1$)，然后调用 LViewToUView，更新用户 UI。

假如本地操作 o_1 不依赖于 H 中的任何操作，则 o_1 经过排除转换之后，得到了定义在 DS_0 状态的操作 o_1'。那么，o_1' 必定和 DS_0 中的某个元素存在先后关系。定义 DS_0 中元素的标识符为$\langle -1,p\rangle\{p=1,2,\cdots,n\}$。针对 DS_0 为空的情况，定义$\langle -1,-1\rangle$为哨兵元素。扩展之后的位置依赖关系可以适用于所有情况。

4.3　并行 OPTIC 算法的实现

算法使用 vecLView 来表示逻辑视图，保存所有的元素的 id 以及可见性 visible。集成本地操作的主要代码如下。

```
Request PIntegrateLocal(Operation op)
{
  int n=j=0,upos,len;
  execute(op); // 执行 o
  upos=op. getPos();
  len=vecLView. size();
  while(j<len&&n<upos)
  {
      if(vecLView[j]. isVisible())
          n++;
      j++;
  }
  lpos=j;
  op. setPos(lpos);
  Request r= GenerateRequest(vecLView[j−1],op);
  //产生请求包含依赖信息和操作
  return r;
}
```

由于本地操作都是立即执行，不影响操作的响应性，集成过程仍然是串行的。当集成远程操作时，算法利用了位置依赖关系以及逻辑文档模型实现并行计算操作的执行形式，主要代码如下。

```
void PIntegrateRemote(r)
{
   int i,lpos,vnum,len=vecLView. size();
   DepInfo dep=null;
   Operation op=null;
   Initialize(dep,op,r);
   Element ele= dep. getID();
   #pragma omp parallel for
   for(i=0;i<len;i++)
   {
     if(vecLview[i]. equals(ele))
     break;
   }
   Calculate(op,dep. getDF());//计算依赖位置
```

```
    lpos=op.getPos();
    #pragma omp parallel for reduction(:+vnum)
    for(k=0;k<lpos;k++)
    {
      if(vecLView[k].isVisible())
        vnum+=1;
    }
    op.setPos(vnum);
    execute(op);
}
```

5 实验结果与分析

为了评价并行 OPTIC 算法的性能，以 Visual Studio2010 为开发平台，采用 OpenMP 4.0 多线程库实现了上述的并行操作转换算法。测试程序模拟两个站点的协同过程，记录算法集成远程操作的计算时间。程序运行的环境为 8 核 Intel i7-5960X(3.0GHz)CPU，16GB DDR3 内存。图 2 为其中一个算例的优化结果。为了测试多核并行计算对模拟退火算法的加速效果，针对不同的协同过程，分别运行了在 4、8、16 个并行线程情况下的结果，如图 4、图 5 所示。

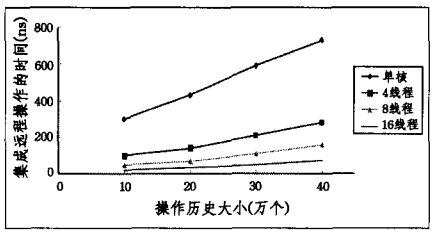

图 4 操作分布(ins:80%,del:20%)

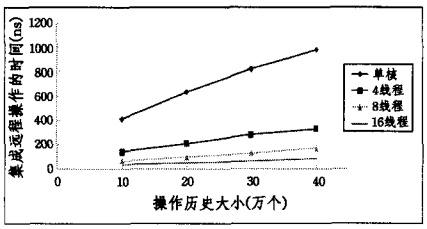

图 5 操作分布(ins:90%,del:20%)

单核情况下的串行 OPTIC 算法，随着操作历史中操作数目的增多，集成一个远程操作所需时间呈线性增长趋势，并且插入操作越多，时间开销也越大。而对于多核情况下的并行 OPTIC 算法，操作分布几乎没有产生什么影响，随着并行线程的增多，算法的计算时间极大地减少。考虑将 50ms 作为最大响应时间，16 线程的并行 OPTIC 算法能够一次性集成 800 多个操作(H 中保存 40 万个操作，90%为插入操作)，而串行版本只能集成 50 个。

结束语 本文首先分析了当前操作转换算法遇到的挑战，大规模协同环境下协同操作造成不自然的交互。通过改进当前最好的 OPTIC 算法的计算模型，把循环计算依赖的过程并行化。实验结果证实了并行的 OPTIC 算法能够充分发挥多核的能力，提高操作的响应性。下一步将考虑利用 GPU 通用计算能力，进一步提高操作转换算法在 P2P 环境下的适用性。

参 考 文 献

[1] Leslie L. Time, clocks, and the ordering of events in a distributed system[J]. Communications of the ACM, 1978, 21(7): 558-565

[2] Sun Cheng-zheng, Xia S, Sun D, et al. Transparent adaptation of single-user applications for multi-user real-time collaboration [J]. ACM Transactions on Computer-Human Interaction, 2006, 13(4): 531-582

[3] Wang Xue-yi, Bu Jia-jun, Chen Chun. Achieving undo in bitmap-based collaborative graphics editing systems[C]// ACM Conference on Computer Supported Cooperative Work. 2002: 68-76

[4] Ignat C-L, Norrie Moira C. Customizable collaborative editor relying on treeOPT algorithm[C]// European Conference on Computer-Supported Cooperative Work. 2003: 315-334

[5] Liu Hua-jun, He Fa-zhi, Li Xiao-xia, et al. A less constraint concurrency control and consistency maintaince in collaborative CAD system[J]. Chinese Journal of Electronics, 2013, 22(1): 15-20

[6] Agustina, Sun Cheng-zheng, Xu Dong. Operational transformation for dependency conflict resolution in real-time collaborative 3D design systems[C]// ACM conference on Computer Supported Cooperative Work. 2012: 1401-1410

[7] Ellis C A, Gibbs S J. Concurrency control in groupware systems [C]// ACM SIGMOD International Conference on Management of Data. 1989: 399-407

[8] Yang Guang-xin, Shi Mei-lin. Object Data Model Based Concurrency Control in Fully-Replicated Architecture [J]. Chinese Journal of Computers, 2000, 23(2): 113-125(in Chinese)
杨光信，史美林. 全复制结构下基于对象数据模型的并发控制[J]. 计算机学报，2000，23(2)：113-125

[9] Matthias R, Doris N-R, Gunzenuser G R. An integrating transformation-oriented approach to concurrency control and undo in group editors[C]// ACM Conference on Computer Supported Cooperative Work. 1996: 288-297

[10] Randolph A, Boucheneb H, Imine A, et al. On Synthesizing a Consistent Operational Transformation Approach [J]. IEEE Transactions on Computers, 2015, 64(4): 1074-1089

[11] Sun Cheng-zheng, Jia Xiao-hua, Zhang Yan-chun, et al. Achieving convergence, causality preservation, and intention preservation in real-time cooperative editing systems[J]. ACM Transactions on Computer-Human Interaction, 1998, 5(1): 63-108

[12] Liao Bin, He Fa-zhi, Jing Shu-xu. Survey of Operational Transformation Algorithms in Real-time Computer Supported Cooperative Work[J]. Journal of Computer Research and Development, 2007, 44(2): 326-333(in Chinese)
廖斌，何发智，荆树旭. 实时协同工作系统中操作转换算法综述[J]. 计算机研究与发展，2007，44(2)：326-333

[13] Li Du, Li Rui. An admissibility-based operational transformation framework for collaborative editing systems[J]. Computer Supported Cooperative Work, 2010, 19(1): 1-43

[14] Imine Abdessamad. A Flexible concurrency control for real-time collaborative editors[C]// International Conference on Distributed Computing Systems Workshops. 2008: 423-428