一种基于 XML的可变粒度冲突避免策略

史 昊, 庄 毅, 王 凯

(南京航空航天大学 信息科学与技术学院, 江苏 南京 210016)

[摘要] 针对协同设计过程中的数据一致性问题, 分析了已有的并发冲突避免策略, 结合 CATIA的特殊性提出了一套可变粒度锁的冲突避免策略, 设计了总体框架及核心算法. 以节点间关联性作为判断依据进行节点仲裁选择, 并提出节点关联集的维护算法. 在协同设计平台中, 利用 XML实现并发控制机制, 对提出的算法进行了实验, 验证了用户可互斥地访问对象, 并保证一致性.

[关键词] 协同设计, 冲突避免, 锁机制, 节点关联

[中图分类号] TP 393 [文献标识码] A [文章编号] 1672-1292(2008)04-0077-05

An Alterable Granularity Collision Avoidance Strategy Based on XML

Shi Hao, Zhuang Yi, Wang Kai

(College of Information Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

Abstract Aiming at the problem of data consistency in collaborative design, the paper analyzes the available strategies of subsequent collision avoidance, puts forward one collision avoidance strategy of alterable granularity lock by combining with the particularity of CATIA. It designs the global frame and kernel algorithm. It adopts relevancy of nodes as the deciding criterion to arbitrate and choose nodes, and puts forward the algorithm of maintaining node relevant aggregation. In the collaborative design platform, XML is used to carry out the subsequent control mechanism, and the proposed algorithm. The experiment indicates that user can access model resource mutually exclusively and ensure the consistency.

Key words collaborative design, collision avoidance, lock mechanism, node relevancy

基于 CATIA的协同造型系统中, 若多个用户并发的对对象进行操作, 很可能出现冲突. 为保证总体设计的一致性, 系统必须提供有效的并发控制机制[1]. 根据并发操作之间的关系, 主要存在两种类型的基本并发控制方法: 并发冲突避免以及并发冲突检测与消解.

现有的并发冲突避免策略有令牌机制和锁机制等. Li等人提出了基于 token passing的令牌持有机制[2], 在服务器维护 token令牌, 持有该令牌的设计者可以对对象进行编辑, 而其他设计者就必须等待令牌的重新分配, 可保证系统不产生冲突, 这种方法并发度很低. 文献[3]中运用了一种以特征为粒度的锁机制, 设计者可以对特征进行加锁, 以阻止同时期其他设计者的访问. 这种并发控制方法提高了系统的并发度, 但设计者有时需要频繁切换加锁对象, 降低了系统的效率. 文献[4]提出一种基于预测的智能锁, 依据图像区域操作次数帮助用户预测未来的操作区域, 并提前锁定该区域. 该方法可以预防潜在的操作冲突, 但提前锁定预测的区域可能会造成资源的浪费, 且会出现很多无用预测和锁定, 增大了系统的开销.

本文提出了一个适用于集中式协同设计系统的可变粒度加锁机制, 并采用节点关联关系作为首要的节点选择依据. 实验结果表明能够减少不同节点之间冲突的发生, 实现了用 XML存储锁的信息, 能够保持语义, 提高传输速度和可扩展性.

1 并发控制机制

1.1 总体框架

在飞行器协同设计平台中利用锁机制进行并发控制, 能够在用户协同工作时保证数据的一致性, 本文

收稿日期: 2008-06-18
基金项目: 国家 863 计划(2006AA706103)资助项目.
通讯联系人: 庄 毅, 教授, 研究方向: 分布式计算和网络安全. E-mail: zhuangy@263.net

提出的并发控制机制如图 1所示.

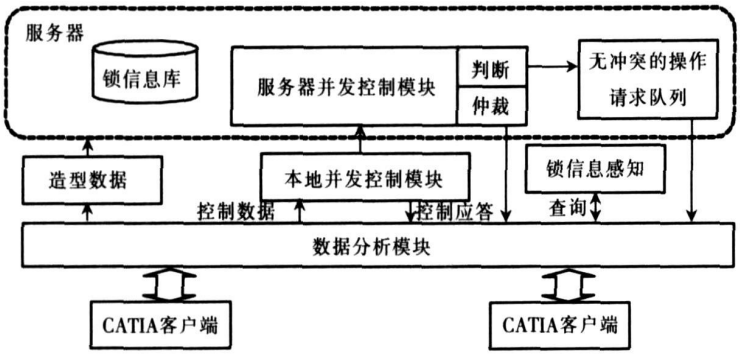

图 1 并发控制机制框架图

Fig.1 The frame graph of subsequent control mechanism

用户登陆 CATIA 协同平台, 参加对某一产品的协同设计. 数据分析模块对数据进行解析, 分离出造型数据、控制数据及查询数据. 用户对对象的加锁请求由并发控制模块进行处理, 对有冲突的请求作出应答, 并将得到的无冲突操作请求队列返回给客户端. 锁信息感知模块在本地存有整个产品的锁信息, 用户可以在设计前进行查询, 减少设计时的无用加锁请求.

在并发控制机制中, 为了减轻服务器负担, 采用两级检测方式. 在本地并发控制模块中, 本地客户端存有设计对象的锁信息, 并及时更新. 用户对对象的加锁请求首先经过本地并发控制模块, 若它可以对请求作出冲突判断, 则返回控制应答, 否则将请求发送至服务器进行判断和仲裁. 本地客户端的锁信息存在不全面问题, 服务器端锁信息的变化客户端可能并未感知到, 因此对于本地并发控制模块未作出应答的加锁请求, 服务器对其进行处理. 虽然采用两级检测, 增加了系统复杂度, 但由于本地判断速度快, 客户端和服务器端之间的通信量降低, 系统效率得到提高.

1 2 可变粒度加锁策略

CAD协同控制的锁机制目前主要有两种形式, 一种是以整个模型为加锁的对象, 粒度过大, 导致并发度很低; 另一种是以特征为加锁的对象, 细粒度提高了并发度, 但可能需要频繁进行加锁操作, 增大消耗, 同样导致整个系统效率的降低[5,6].

运用 CATIA 进行设计开发时, 设计的产品模型对应于一棵产品结构树 T, 其中节点由根至叶子依次为产品、部件、零件和特征. 一个 CATIA 产品划分为 4个不同层次的元素集合, 4部分的组合 < productID, assemblyID, partID, featureID > 才能标识一个具体的底层特征.

考虑既要避免模型各个层次上的冲突, 又不至于加锁的粒度过大或过细, 本文运用可变粒度的加锁方式, 产品结构树任意层次的节点都可以作为加锁对象, 只要它满足加锁条件.

对于 CATIA 模型中的任一节点 n, 其直接影响集可表示为:

$Aff(n) = \{n, F(n), S(n)\}$, 其中, $F(n)$为 n的父节点的集合, $S(n)$为 n的子节点的集合.

在可变粒度锁策略中, 节点的加锁条件与其直接影响集有关. 只有节点 n的直接影响集 $Aff(n)$中的节点均处于未锁定状态, 设计者对节点 n的加锁请求才被允许; 如果设计者对节点 n已加锁, 那么 $Aff(n)$中包含节点的加锁请求都将被拒绝. 根据这种特点, 若用户在短期内需要对产品的某一部分进行大的改动, 可以将需要改动的最小极大区域集进行加锁, 这样可以避免过于频繁的加 解锁操作; 而用户对模型进行小范围操作时, 只需对改动的特征加锁, 不影响其他用户的操作, 可提高并发度.

1 3 AGL算法

在基于 CATIA 的协同平台中, 注册登陆模块首先检查用户权限, 若用户通过验证, 此后参加某一产品的设计并被赋予相应角色. 依据本文提出的并发控制机制设计可变粒度锁算法 AGL (alterable granularity lock Algorithm), 在协同设计处于起始状态时, 产品中所有节点皆为未锁状态. 假设协同设计的产品为 P, 用户操作的目的节点为 N, 具体算法描述如下:

```
var N: integer init 0;                          //N起始状态未锁
    f[q]: integer for each q  P init 0;         //产品 P的锁信息
```

```
    operation: ( create, modify, delete);                    /操作类型
begin
    download fl[ q] to client and update timely;
    do lock request for the operation;
    if operation= create then
        Create(N);
        amend fl[ ];                                         /修改锁信息
        lkN: = 0;                                            /设置 N 初始未锁
        Send N and fl[ ] to Server;                          /将 N 及锁信息文件传到服务器
    else
        if LocateLk(N) = 1 then                              /本地检测已锁
            give up the operation;
        else
            send lock request to server;
            if Lk(N) = 1                                     /服务器检测已锁
                give up the operation;
            else
                lkN: = 1;
                Lock(N);                                     /进行加锁
                if operation = delete then
                    Delete(N);
                    amend fl[ ];                             /若是删除 N, 则修改锁信息文件
                else
                    Modify(N);
                    lkN: = 0;
                    Unlock(N);                               /解锁
End
```

1 4　锁信息描述

CATIA 模型文件具有层次性的特点, 其数据是结构化数据. XML是 W 3C 在网络环境下推出的面向数据处理的语言, 采用树状的层次结构, 适合描述结构化数据. XML具有可扩展性, 能够根据需要添加设计者需要表达的各种信息.

因此, 考虑采用 XML的形式来描述和存储锁信息, 能够在不影响模型文件树形结构的同时, 自行设计锁信息文件的模式库. 结合解析的 CATIA 模型文件的结构, 利用 DOM 模型生成其对应的锁信息 XML文件, 通过遍历它来检索和查询锁信息. 模型的锁信息 XML文件结构在实现示例中有具体描述, 这里不再赘述.

2　并发仲裁机制

当多用户同时对同一节点请求加锁, 只能接受一个用户的请求, 此时需要仲裁机制进行判断选择. 传统的仲裁策略是采用先来先服务, 但这种方法欠灵活, 效率较低. 结合 CATIA 环境下产品设计的特殊性, 为了减小不同用户操作节点间冲突的产生概率, 本文提出一种根据节点关联集来进行仲裁的方法.

2 1　相关定义

定义 **1**　用户已有锁节点集: 用户已加锁的节点集合, 记为 allock(用户 ID).

定义 **2**　节点关联: 节点之间存在的直接约束关系, 以保持语义.

定义 **3**　节点关联集: 与节点 i存在节点关联的节点的集合, 记为 relate(i).

节点关联集在协同设计过程中需要动态维护, 以适应用户对产品模型的随时修改. 假设节点 i的节点关联集为 relate(i), 其维护具体算法描述如下

```
var operationi ( create, modify, delete);
```

```
    p: node;
begin
    if operation i = create then                    //若加入节点 i
        create relate( i);                          //创建 i的节点关联集
        for each p  relate( i)
            add i to relate( p);                    //对与 i关联的节点 p,将 i加入 relate( p)中
    else if operation i = delete then //若删除节点 i
        for each p  relate( i)
            remove i from relate( p);               //对与 i关联的节点 p,在 relate( p)中删去 i
        delete relate( i);                          //删除 relate( i)
        else                                        //若修改节点 i则根据修改后的关联情况维护 relate( i)及 relate( p)
            if create restriction between i and p then
                add i to relate( p);
                add p to relate( i);
            if delete restriction between i and p then
                remove i from relate( p);
                remove p from relate( i);
end
```

2 2 仲裁算法

在并发控制机制中,当有多个用户 A, B, 同时申请一个空闲节点锁时,需要运用仲裁算法来进行选择.本文运用一种根据节点关联集来进行仲裁的方法,假设空闲节点 i同时申请该节点锁的用户列表为 userlist,其具体算法描述如下:

```
var allock( u)  for each u  userlist,
    relate( i);
begin
    U: = {u  userlist|max( number of relate( i)  allock( u) )}; //相交元素最多的用户
    if length( U) = 1 then
        allow u  U lock i;                          //取该用户为锁拥有者
        {u|u  U  u  userlist} wait;                 //其他等待
else
        choose using FCFS;                          //用先来先服务的策略进行仲裁
end
```

3 实现

现阶段,Web Services技术以它的开放性和可扩展性而得以广泛得运用,它同样成为协同设计领域新的热点[7].本文提出的锁机制用于基于 Web Services的 CATIA环境下协同设计平台中,支持多用户对同一对象进行并发协同操作和交流.在平台中,采用 Web Service技术实现分布式计算,在 Microsoft visual studio环境下开发 Web服务,利用 MSSOAP3 0开发包实现客户端与服务器端之间的数据传递,客户端利用 CAA组件对 CATIA进行二次开发.

为了检验并发控制机制的有效性,将开发的客户端部署在实验室的主机上,将 Web Services部署在实验室的服务器上.用户运行客户端,并且登录到服务器上,将 CATIA模型 CAAPstBlockMovable_ chain. CATProduct下载到本地打开,进入协同状态,可以浏览或设计该模型.此时生成的锁信息文件如下所示:

```
<? xml version= 1 0?>
< chain lock= "0" >
    < link 1 lock= "0" />
    < link 2 lock= "0" />
    < Part1 1 lock= "0" />
< /chain>
```

节点状态均为未锁. 对 link. 1零件作加锁操作,可以对 link. 1进行设计. 此时另一用户也参与到该次协同设计中,对模型的状态进行查询,得到结果如图 2(a)所示,若此时对 link. 1提出加锁请求,则被拒绝,如图 2(b)所示.

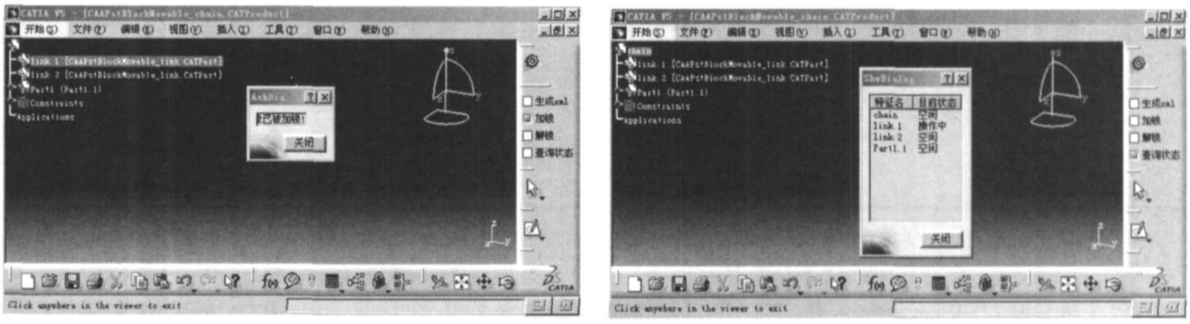

图 2　客户端显示图

Fig.2　The vision of client

在该平台中运用了所述的并发冲突避免策略,实验验证能够保证用户对模型的互斥访问,而采用基于节点关联集的仲裁机制,最大可能地保证了约束关系紧密的节点由同一个用户控制,因此减少了节点间冲突的发生,利于系统的一致性维护.

4　结语

并发控制是 CSCW 中的关键问题. 本文首先简单介绍了传统并发避免的策略. 然后提出了一套冲突避免的方案,采用可变粒度的锁机制,以节点间的关联性为加锁的仲裁依据,并以 XML的形式记录及传输锁信息. 该方法从 CATIA模型的特性考虑,并以减少特征间冲突为目标,为实现协同平台打下了良好的基础. 以后的工作是进一步研究并发冲突检测和消解算法,以保证整个系统的数据一致性,使系统能够高效运行.

[参考文献] (References)

[1] 史美林,向勇,杨光信. 计算机支持的协同工作理论与应用[M]. 北京:电子工业出版社, 2000: 154-155.
Shi Meilin, Xiang Yong, Yang Guangxin. Theory and Application of Computer Supported Cooperative Work[M]. Beijing: Publishing House of Electronics Industry, 2000: 154-155. (in Chinese)

[2] Li W D, Ong S K, Jerry Y H Fuh, et al. Feature based design in a distributed and collaborative environment[J]. Compute Aided Design, 2004, 36(9): 775-793.

[3] Li Min, Gao Shuming, Jerry Y H Fuh, et al. A fine granular concorrency control mechanism for a peer-to-peer cooperative design environment[C] // Proceedings of the 11th International Conference on Computer Supported Cooperative Work in Design. Australia, 2007: 180-185.

[4] 卢刚,惠怀海,卜佳俊,等. 基于预测的图案协同设计中智能锁研究[J]. 重庆大学学报, 2005, 28(5): 68-71.
Lu Gang, Hui Huaihai, Pu Jiajun, et al. Intelligent locking in collaborative pattern design based on forecast[J]. Journal of Chongqing University, 2005, 28(5): 68-71. (in Chinese)

[5] Tang M, Chou S C, Dong J X. Collaborative virtual environment for feature based modeling[C] // Proceedings of 2004 International Conference on Virtual Reality Continuum and its Applications in Industry. Singapore: ACM, 2004: 120-126.

[6] Chen J Y, Ma Y S, Wang C L, et al. Collaborative design environment with multiple CAD systems[J]. Computer-Aided Design and Applications, 2005, 2(1/4): 367-376.

[7] Johnson J E, Langworhty D E, Lamport L. Formal specification of a Web services protocol[C] // Proceedings of the 1st International Workshop on Web Service and Formal Methods. Pisa, Italy, 2004: 147-158.

[责任编辑: 刘　健]