浙江理工大学学报(自然科学版),第 39 卷,第 1 期,2018 年 1 月
Journal of Zhejiang Sci-Tech University (Natural Sciences)
Vol. 39, No. 1, Jan. 2018

DOI:10.3969/j.issn.1673-3851(n).2018.01.015

基于 BIM 子模型的分布式版本控制研究

李俊男[1],廖梦纯[1],桂　宁[1,2]

(1.浙江理工大学信息学院,杭州 310018;2.嘉兴国电通新能源科技有限公司,浙江 嘉兴 314000)

摘　要:在建筑信息模型(BIM)协同开发过程中,各参与方通常仅关注其领域相关的子模型数据而非复杂庞大的全局模型。为领域相关的子模型开发提供系统性的分布式开发功能,特别是版本控制支持,是 BIM 协同开发急需解决的问题。文章提出了一种基于 BIM 子模型的分布式版本控制框架,实现了分布式版本控制支持。建立了基于操作追踪的版本控制机制,通过记录用户对模型的操作,对操作序列进行叠加、逆反以实现版本跟踪;构建了相互耦合并独立工作的子模型层,不依赖于全局模型,实现设计人员在工作过程中对需求子模型的版本控制;构建了局部更新全局协同的工作模式,实现分布子模型与全局模型的版本协同。该系统已在开源 BIMServer 平台上实现,仿真结果表明,该系统在子模型支持、版本响应时间、存储资源占用及网络传输速度等方面均有明显的优势。

关键词:建筑信息模型(BIM);版本控制;分布式;子模型

中图分类号:TP319　　**文献标志码**:A　　**文章编号**:1673-3851 (2018) 01-0090-08

0　引　言

建筑信息模型(Building information modeling, BIM)是以建筑工程项目的相关数据为模型基础模拟建筑物真实信息的数字化模型[1]。它能将建筑生命周期不同阶段的资源、数据相互关联,可供不同参与方共同使用[2-5]。在 BIM 项目的不同阶段,由于协同设计过程中的多方人员参与设计需共用同一个 BIM 模型,这会导致对象模型的不一致性及反复性问题[6]。同时,由于各个 BIM 项目参与方通常关注的是 BIM 模型中的部分信息,如电气设计人员主要在电气设计子模型上进行工作,因而他们主要关注电气设计子模型。如何有效地支持符合参与方需求子模型的修改和追踪,是目前 BIM 协同工作平台亟待解决的问题之一。

工业基础分类(Industry foundation classes, IFC)是 BIM 数据共享和描述的国际标准[7-9],目前主流的 BIM 平台主要基于 IFC 标准,如 IFC Web Server[10]、XBIM[11]等。IFC Web Server、XBIM 主要提供对 IFC 数据处理、浏览展示以及对模型数据的分析功能。Chen 等[12]构建了一个 BIM 数据服务器,通过构建针对建筑设计和结构设计不同需求的数据视图,实现建筑设计和结构设计的协同工作。但是这些平台通常不支持模型的版本管理功能。在模型的版本管理上,BIMServer[13]是目前最为广泛使用的开源 BIM 平台,对于模型的管理提供了版本比较功能,然而它并不能很好地跟踪用户的修改,也不支持分布式的子模型的设计模式。商业软件一般提供了较强的协同设计支持,如:Autodesk Revit® 提供了"链接模型"和"工作集"两种模式来支撑协同工作,Autodesk Vault® 提供了对版本控制的支持。然而,这些工具采用集中式的模型管理,并未提供适应不同设计方的领域相关子模型支持。因此,不同参与方常常需要存储和传输非必要数据,由此带来不必要的存储、传输和解析开销。

常见的版本管理系统如 Git[14]、CVS[15]等,多是基于文件系统的管理方式,是以文本中行数据作为数据单元,因而不能有效解析 IFC 这类面向对象

收稿日期:2017－06－05　　网络出版日期:2017－12－11

作者简介:李俊男(1992－),男,山东潍坊人,硕士研究生,主要从事智能信息建筑方面的研究。

通信作者:桂　宁,E-mail:ninggui@gmail.com

的 BIM 模型。当 IFC 对象变化时,常常涉及到大量行数据变化,这给 Git 修改跟踪带来了许多问题。并且,IFC 项目模型常常较大(通常有上百兆),当模型规模增大时,Git 版本跟踪性能急剧下降。尽管 Git 实现了分布式的版本控制功能,但是其基于文本管理方案并不能很好地支持基于对象的 IFC 模型的版本管理。

针对上述问题,本文在已有子模型抽取研究[16]的基础上,提出了基于领域相关子模型的分布式版本控制框架,以实现分布式的系统管理。本文的贡献主要有:设计了基于操作跟踪的本地化版本管理机制,以有效记录用户对模型修改;在分布式模型管理上,提出了局部更新全局协同的工作模式,在保证模型一致性的同时,实现了分布式子模型与全局模型的高效同步。

1　分布式子模型版本控制框架

在建筑设计过程中,由于建筑规模的扩大及涉及领域的增多,模型数据量不断增加。在版本控制支持上,目前的 BIM 协同工作平台多采用集中式版本控制模式,各设计人员在完整模型数据的基础上进行工作。随着 BIM 模型数据量的增加,集中式版本控制对网络要求越来越高,服务器压力也不断增长。而在建筑设计过程中,各领域设计人员主要工作集中于其领域相关的部分数据,因此无筛选地获取全局模型,导致模型数据价值密度较低,并且带来不必要的资源开销以及相应的模型数据安全问题[17]。因此,如何为不同领域的参与方提供定制化模型的数据,并使得不同用户能够在定制化模型上工作,成为提高模型数据价值密度的有效解决方案之一。此外,集中式的模型管理较难适应分布式的模型开发,也对数据的安全和隐私控制带来了一定的难度。

为解决上述问题,本文提出了基于操作追踪的分布式子模型版本控制的设计框架,如图 1 所示。与传统的集中式管理模式不同之处在于,本协同设计框架的数据模型分为 2 层:传统集中式的数据中心全局模型层和相互耦合的领域子模型构成的子模型层。设计人员不再直接修改全局模型,而是在基于由独立子模型耦合组成的协同工作网络中操作。在该框架中,全局模型层和子模型层同时提供对应的版本控制支持,用户的修改信息在提交到子模型中后,可以同步到全局不同版本中。不同领域设计人员在协同工作中通过实现对子模型的版本控制,领域子模型与全局模型的数据合并实现全局模型的更新,从而实现局部更新全局协同的工作模式。

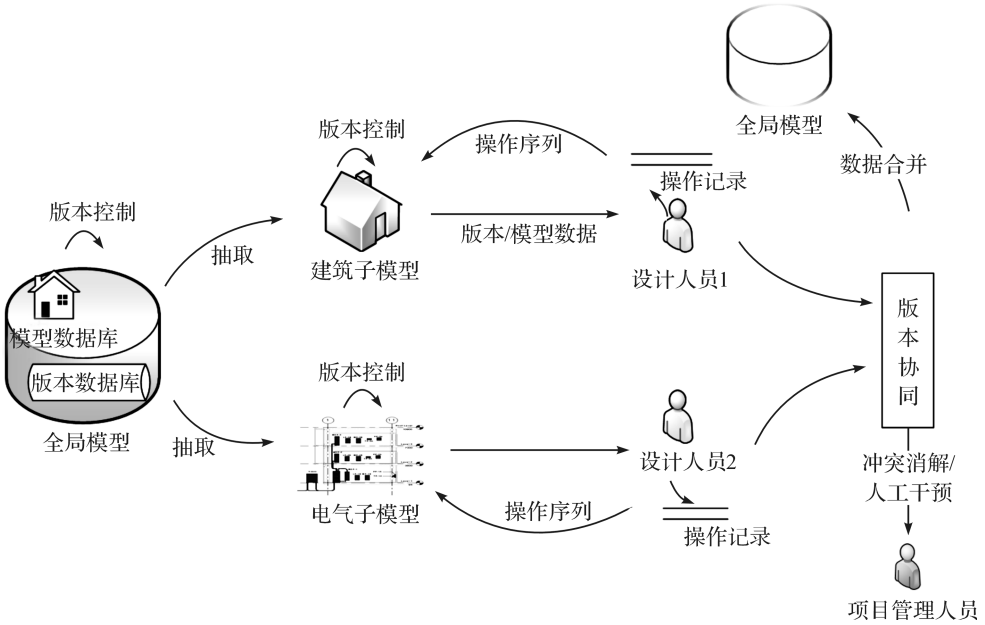

图 1　分布式子模型版本管理的协同设计框架

协同工作过程中,多领域子模型之间数据通常存在重叠,因设计需求不同,各设计人员对于模型数据同时进行修改,这导致在各子模型之间以及子模型与全局模型之间,数据合并时可能出现数据冲突问题,因此需对设计过程中版本数据冲突进行消解。如图 1 右侧所示,本框架使用基于设计人员优先级的冲突消解方案,实现版本数据的合并、统一。

2 基于操作追踪的版本控制

与传统基于版本比较的实现方案不同,本框架实现操作行为与操作数据实体的转换、引入操作追踪机制以记录用户对模型的修改,帮助各领域参与方记录 BIM 模型设计过程中的操作。通过对操作序列进行版本控制,与领域子模型数据叠加从而实现版本演变。通过维护多领域子模型与全局模型数据统一,实现局部更新全局协同的分布式子模型版本控制。

2.1 操作记录定义

为实现操作追踪,需记录设计人员对模型的操作行为,因此需对设计人员操作行为进行数据实体转换。操作记录实体依据时间记录设计人员对模型实体属性的修改,记录主要包括操作符、被操作实体 ID 以及实体属性的修改内容,版本区存储示例如图 2 所示。

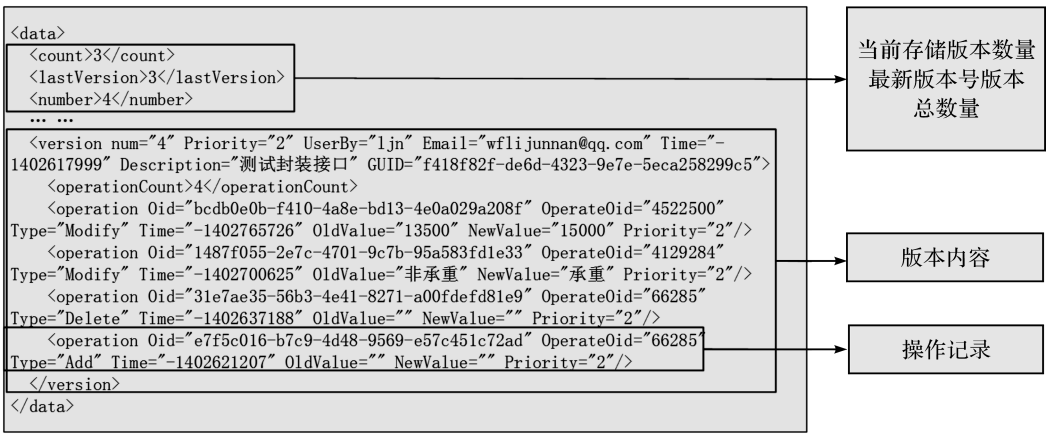

图 2 版本区存储示例

其中操作记录示例包含内容为:

a) 操作 ID(Oid):为保证操作记录等数据唯一性,数据存储过程中添加全局唯一标识符(Globally unique identifier, GUID),作为操作的标识符。GUID 为二进制长度为 128 位的全局唯一数字标识符。图 2 显示的操作记录示例,可以看出操作 ID "e7f5c016-b7c9-4d48-9569-e57c451c72ad"。

b) 被操作实体 ID(OperateOid):标识设计人员对模型进行修改实体 ID,该实体 ID 是其在 IFC 模型中的全局唯一标识,以确保实体 ID 与模型中实体的映射关系。图 2 示例中,被操作实体为 66285。

c) 操作符(Type):表示设计人员该操作记录中的操作属性,包括实体修改、增加及删除 3 种类型。图 2 示例中,操作符为 Add。

d) 优先级(Priority):用于区分设计人员对模型修改的权限。图 2 示例中,优先级为 2。

e) 时间戳(Time):记录设计人员对模型修改的时间。图 2 示例中,时间戳为 1480331991252。

本文对操作追踪定义为:通过完整记录设计人员对 BIM 模型实体的操作,将操作以数据实体的方式存储于版本库中,实现对设计人员操作的追踪。构建的版本控制系统实现模型版本控制依赖于模型数据与版本数据的合并,模型数据独立于版本库存储,只存储原始的模型数据,如版本发生更改时,只需将版本库中所需版本操作记录提取,累加到模型数据中,即可得到所需的模型版本。该种方式有利于提高模型数据传输效率。

2.2 版本记录存储

为了实现有效的模型版本控制,必须对用户的修改进行合理的记录、存储。这里的版本由两部分组成:模型数据和版本数据。

模型数据包含 BIM 模型的数字化描述信息,版本数据包含版本修订情况及设计人员的操作序列等信息。在模型数据中:全局模型包含 BIM 模型的所有模型信息及全局的版本信息,是协同工作中数据总集;子模型中模型数据由全局模型依据领域需求抽取获得,用以实现领域中协同工作。私有库与领域子模型数据一致,用于存储设计人员私有版本,实现本地化版本控制。子模型数据和全局模型之间为松散同步,按照需求同步而非实时同步,从而降低数据传输量,提高版本管理效率。

版本数据分 2 部分组成,版本描述信息及操作序列。版本描述信息包括:版本编号、设计人员 ID、优先级、时间戳等。其中版本编号为记录设计人员提交版本次数的编号,为版本查询提供简明的描述。版本区除记录版本内容外,同时记录版本区当前存储版本数量、版本总数量、最新版本号等版本区信息,如图 2 所示。版本总数量是指设计人员在最初

版本至当前最新版本存储的全部版本的数量,版本区当前存储数量与版本总数量并不相等,当设计人员对模型版本进行回溯操作时,版本管理会将需求版本之后的版本数据存储提取到缓冲区中,该操作会改变版本区中当前版本数量。

2.3　版本库设计

本文中为实现基于操作追踪的版本控制,需记录并阶段性存储领域人员在设计过程中的操作记录,因此设计操作区用于临时存储设计人员操作序列,设计版本区存储版本信息,为版本控制提供数据支持。设计人员在进行版本回溯,版本区会丢失部分版本数据,因此设计缓冲区保存丢失版本数据,保持数据完整性。为提高模型数据加载效率,保存当前版本快照,所以设计快照区。版本库设计的具体内容为:

a) 操作区:用于记录设计人员在新版本提交之前对当前模型文件进行的所有操作数据,当新版本成功提交、存储后,自动清空操作区。

b) 版本区:用于记录提交的完整版本记录,其内容包括设计人员即提交者对于版本记录的描述信息,以及对 IFC 文件中实体修改的记录信息。

c) 快照区:用于存储当前模型版本数据,由版本区中初始版本至当前版本的操作记录合并获得。

d) 缓冲区:用于存储回溯版本后丢失的版本数据,如设计人员回溯到历史版本,则将需求版本与回溯前版本之间版本差异存储于缓冲区中。缓冲区仅在设计人员进行版本回溯后出现。

在工作过程中,版本控制数据流如图 3 所示。当设计人员完成该阶段工作后发出提交版本请求,服务器将操作区中所存操作序列提交到私有库版本区中,同时添加设计人员信息、版本号等版本描述信息。版本提交成功之后服务器将快照区数据置为当前版本,并清空私有库操作区,从而实现版本更新。

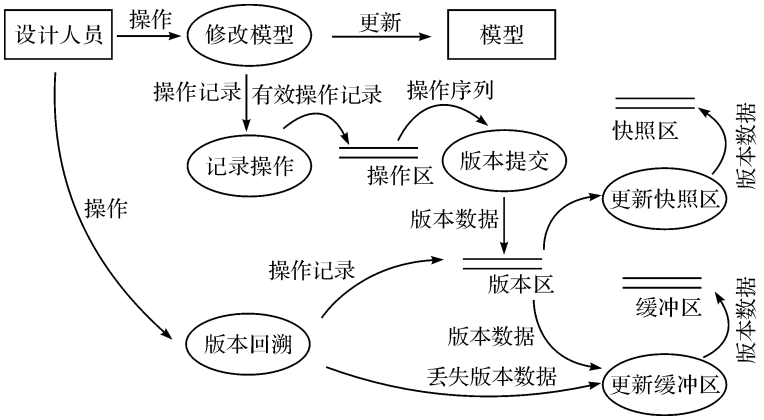

图 3　版本控制数据流图

本文中版本回溯是指通过对当前模型版本进行操作记录的逆操作,回溯至需求版本。逆操作指某一操作与其逆操作叠加后属于无效操作,如对模型某一实体进行删除,操作记录中操作符为 Delete,其逆操作操作记录的操作符将更改为 Add。

设计人员请求版本回溯时,将版本区中当前版本与需求版本之间版本数据提取到缓冲区中,并更新快照区,实现版本回溯。版本前进时会读取缓冲区中版本数据加载至版本区,对版本区数据重新叠加更新至快照区,与子模型数据整合得到设计人员的需求模型版本。

2.4　子模型与全局模型协同

在本文构建分布式子模型版本控制框架中,同时支持在全局模型层和子模型层的版本控制,全局模型层的版本控制机制与子模型层版本控制机制一致。

在 BIM 协同工作中,由全局模型抽取获得多个领域子模型,抽取过程中同时抽取子模型中相关引用数据,以保证子模型的完备性,抽取的多个子模型间允许存在交集。[16] 图 4 为版本同步流程图,设计人员在客户端对子模型进行版本控制,存储于子模型版本数据库,实现该领域设计人员的版本协同。

当模型版本更新时向服务器版本库进行推送,服务器检测客户端模型版本库与服务器版本库差异,若服务器版本库中数据为客户端数据的子集,则依据时间戳将产生的增量版本数据追加到服务器版本库中;若客户端模型版本提交时,服务器版本库中存在新的提交记录,则需要拉取最新版本后进行版本提交。客户端向服务器拉取版本数据机制与版本推送机制类似,通过比对客户端版本库与服务器版本库差异,对子模型版本库推送增量版本数据。类

似于领域设计人员版本协同，领域子模型版本库与全局模型版本库进行数据同步，实现领域子模型与全局模型协同。

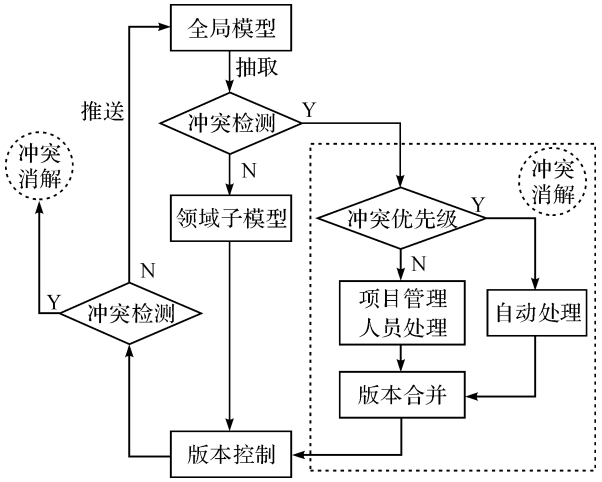

图 4 分布式模型版本同步流程

在版本推送与拉取过程中，可能存在多个子模型中对交集实体的冲突操作，产生版本数据冲突，所以需对版本冲突消解后才可完成版本推送与拉取。本文采用基于操作记录优先级及领域相关度的冲突消解策略。在多个子模型版本协同工作过程中，如合并的版本数据中不存在重复的被操作实体 ID，则依照时间戳顺序合并操作记录；若合并过程产生数据冲突，则对操作记录优先级进行判断，如产生冲突的操作记录中存在高优先级的记录，则选取较高优先级的操作记录作为冲突消解的处理结果；若无法根据优先级的方式解决冲突，则依据领域相关度交付给领域管理人员进行冲突消解。

3 系统实现及分析

3.1 系统架构实现

BIMServer 是目前最为广泛使用的基于 IFC 标准的开源 BIM 服务器平台。BIMServer 属于 B/S 架构，维护和升级方式简单；支持 IFC Engine 等多个解析引擎，支持 IFC 模型的 3D 展示；使用嵌入式数据库系统 BerkeleyDB 作为模型存储，其高效性、可伸缩性切实的适应了 BIM 服务器对性能上的需求；提供了完整的源码和 SDK，易于扩展。因此本文选择 BIMServer 作为系统的开发平台。

当前 BIMServer 使用 Berkeley DB 内存数据库（键值对存储）来存储模型数据，实现数据读写速度优化。但是其仅支持较为简单的数据记录，并不适合作为版本管理的数据库。由于 XML 的跨平台性和统一性等优点，使其适合作为数据交换的媒介，同时 XML 拥有良好的可扩展性以及方便的信息检索。因此，采用基于 XML 的模型版本库实现方式。本文中模型包含 2 部分数据，即 BerkeleyDB 中存储的模型数据及以 XML 格式存储的版本数据。

同时，由于 BIMServer 内建的 Plugin 机制只能提供有限的扩展能力，不能很好地支持平台所需的接口定义[18]，所以构建一个独立的 Servlet 作为版本控制功能数据交互容器，实现对设计人员操作记录的传输以及版本控制模块的调用。由此，系统架构的原型设计图如图 5 所示，该架构分为以下 2 个部分：

a) 模型数据的管理和展示，模型数据管理主要通过实现重用及封装 BIMServer 模块实现。

b) 版本控制及协同、添加 VersionManage 模块，其数据来源依赖于 BIMServer，但方法调用与数据存储独立于 BIMServer。版本控制及协同模块独立于 BimServer 内建的接口，在协同设计过程中该模块从版本库中读取版本数据，BIMServerService 模块抽取 BerkeleyDB 中模型数据，版本数据与模型数据整合得到模型版本。

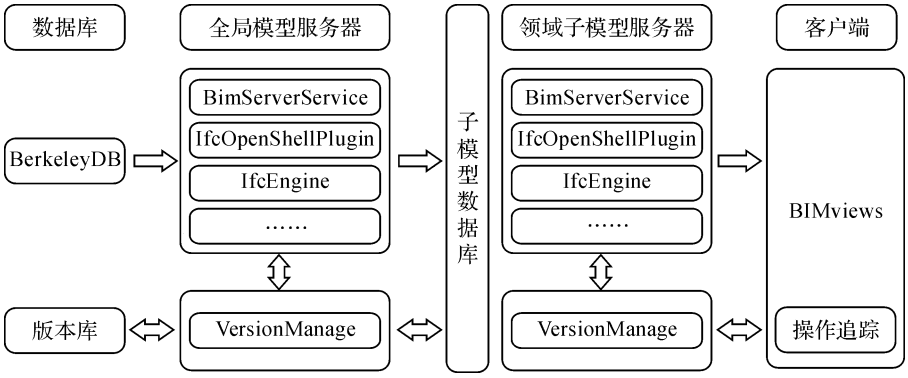

图 5 原型系统框架设计

3.2　子模型版本管理实现

为实现子模型分布式管理功能,设计了 5 类接口:配置项接口、序列化接口、版本控制接口、远程交互接口以及冲突消解接口,如图 6 所示,其中部分接口信息如下:

a) 配置项接口:包括 setConfig 等 8 个接口,用于设置版本库路径、设计人员及项目 ID 等信息,以及实现私有库与中央版本库转换。

b) 序列化接口:用于实现设计人员操作记录与数据实体转换,通过 CreateOpertion 等方法实现操作行为与操作记录数据实体的转换。通过 serialize 及 deSerialize 方法对操作记录对象进行序列化,实现数据存储及读取。

c) 版本控制接口:用于实现本地化版本控制,该类接口实现功能包含:提交操作记录到操作区;提交新版本到版本区;获取版本、快照等数据;更新快照区;版本前进及回溯。

d) 远程交互接口:通过读取版本库配置文件,实现版本库数据的推送拉取及克隆。

e) 冲突消解接口:实现冲突消解,预留部分接口,用于添加冲突处理规则以适用多应用场景下的冲突消解。

图 6　分布式子模型版本控制接口

3.3　性能比较

通过本文构建的分布式子模型版本控制框架与目前主流的版本管理系统 Git 进行比较。为了衡量系统在不同模型大小下的性能,选取的文件大小在 0～100 MB 之间,包含 0.02、1.44、5.28、10.80、29.60、61.10、95.90 MB 等 10 组 IFC 模型,来源于 GitHub 的 BIMServer 项目。IFC 模型的修改对象为模型中的“墙”实体 WallStandCase,在其属性进行的修改后,分别使用 Git 及本文构建的版本控制进行版本提交,并计算其系统开销。系统性能记录选取的指标为:版本提交耗时及提交过程中数据传输量。其中版本提交耗时只记录 Git 本地提交耗时。为了防止单次数据的波动性,数据为对模型的 30 次操作的平均值。客户端配置为 Windows 10 64 位操作系统,CPU 型号为 i7-3770,8 G 内存。

由于 Git 以文本中行数据作为数据单元,当删除 IFC 实体属性时,会改变文件结构,引起大量的版本修订记录,导致 Git 性能下降。图 7 显示,Git 版本提交耗时及版本提交传输量随模型的大小和复杂度呈上升趋势。由于本文构建的版本控制基于设计人员的操作追踪,只需记录该次设计人员对模型修改的操作记录,并不存储模型数据,需记录数据数量远小于 Git 记录 IFC 文件版本修改的数据信息,因此在模型数据量逐渐增大的情况下,性能优于 Git 版本控制。当模型文件大小在约为 100 MB 时,本文构建的版本控制平台提交耗时约为 Git 提交耗时的 4.75%,数据量传输仅为 Git 版本传输数据量的 0.1%。仿真结果表明本平台在子模型支持、版本响应时间、存储资源占用及网络传输速度等方面均有明显的优势。

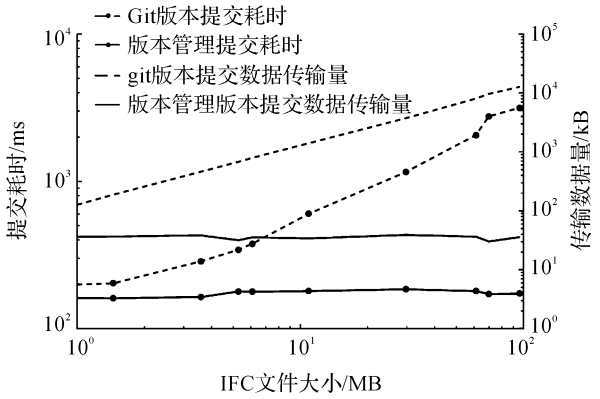

图 7 与 Git 在版本提交的性能比较

4 结 论

本文通过操作追踪机制实现对 IFC 模型版本的记录，并通过该机制实现 BIMServer 环境中的分布式子模型版本控制，使得 BIM 子模型在协同工作模式下具有更优的应用效率及应用价值。另外，通过使用数据量较小、数据价值密度更高的子模型实现版本控制，有效地解决 BIM 协同工作中模型性能低、模型版本管理困难等问题。

由于系统采用了分布式模型的架构，在子模型与全局模型的版本推送、拉取过程中可能会出现模型数据不一致问题。目前的系统主要采用了基于操作优先级冲突解决方案实现。后续将研究基于设计人员特性的冲突消解机制，以更好地体现不同设计人员在不同类型的子模型中的优先级。另外，由于现有的设计软件缺乏对于 IFC 模型的设计支持，拟研究将 Autodesk 公司的 Revit 系列软件中对建筑模型的编辑功能与 BIMServer 的模型管理相整合，以实现多终端浏览器为客户端，BIMServer 为服务器，Revit 为建筑编辑及版本管理的 B/S 架构的一体化应用平台。

参考文献：

[1] Eastman C, Teicholz P, Sacks R, et al. BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors[M]. New Jersey, US: Wiley Publishing, 2008:101-102.

[2] Soibelman L, Sacks R, Akinci B, et al. Preparing civil engineers for international collaboration in construction management[J]. Journal of Professional Issues in Engineering Education & Practice, 2011, 137(3): 141-150.

[3] Son J W, Rojas E M. Understanding collaborative working processes of temporary project teams in large-scale construction projects[C]//Building a Sustainable Future: Proceedings of the 2009 Construction Research Congress, 2009:856-865.

[4] 满庆鹏，李晓东. 基于普适计算和 BIM 的协同施工方法研究[J]. 土木工程学报，2012，45(S2)：311-315.

[5] Liu Y, Nederveen S V, Hertogh M. Understanding effects of BIM on collaborative design and construction: An empirical study in China[J]. International Journal of Project Management, 2017, 35(4): 686-698.

[6] 曹健，张友良. 协同设计的版本管理[J]. 计算机集成制造系统，1998，4(6)：16-20.

[7] Liebich T. IFC4: The New Building SMART Standard [EB/OL]. (2003-03-12)[2017-06-05]. http://www.buildingsmart-tech.org.

[8] Building SMART International Ltd. IFC Data File Formats and Icons[EB/OL]. (2016-07-15)[2017-02-16]. http://www.buildingsmart-tech.org/specifications/ifc-overview/ifc-overview-summary.

[9] 王勇，张建平，胡振中. 建筑施工 IFC 数据描述标准的研究[J]. 土木建筑工程信息技术，2011(4)：9-15.

[10] Ali I. IFC Web Server Project[EB/OL]. (2016-08-07)[2017-02-16]. https://drive.google.com/file/d/0B3PRiGtq6fseTnhhQjBhNEs0S2c/edit.

[11] Ward A, Benghi C, Černý M, et al. XBIM[EB/OL]. (2016-11-02)[2017-02-16]. https://xbim.codeplex.com/.

[12] Chen P, Cui L, Wan C, et al. Implementation of IFC-based web server for collaborative building design between architects and structural engineers[J]. Automation in Construction, 2005, 14(1): 115-128.

[13] Jiang Y, Ming J, Wu D, et al. BIM server requirements to support the energy efficient building lifecycle[C]// International Conference on Computing in Civil Engineering, 2012:365-372.

[14] Loeliger J. Version Control with Git: Powerful Tools and Techniques for Collaborative Software Development [M]. Sebastopol, US: O'Reilly Media, Inc, 2009.

[15] Derek Robert Price, Ximbiot. CVS: Concurrent Versions System[EB/OL]. (2015-07-21)[2017-02-16]. http://www.nongnu.org/cvs/.

[16] 桂宁，王楚涵，董彦松，等. 基于工业基础分类的协同工作子模型抽取研究[J]. 工业建筑，2016，46(8)：176-81.

[17] 陈小波. "BIM& 云"管理体系安全研究[J]. 建筑经济，2013(7)：93-96.

[18] 马智亮，毛娜. 基于 BIMserver.org 开发网络 BIM 工具软件[C/OL]//第一届全国 bim 学术会议，2015.(2005-11-12)[2017-06-05]. https://www.researchgate.net/publication/283716482.

Research on the distributed version control for BIM-based submodel

LI Junnan[1], *LIAO Mengchun*[1], *GUI Ning*[1,2]

(1. School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou 310018, China;
2. Jiaxing Guodiantong New Energy Co., Ltd, Jiaxing 314000, China)

Abstract: In the process of the BIM collaborative development, different stakeholders usually only focus on their domain-specific data rather than the whole BIM model. Thus, it is important to provide a systematic distributed development function for the submodel development, especially for version control supporting. This paper proposes a distributed version control framework based on BIM submodel and achieves distributed version control support. Besides, version control mechanism based on operation tracing is established. The version tracing is implemented through superposition and reversion of operation sequence. The submodel layer which couples each other and works independently is established. Independent of global model, designer's control of submodel is achieved in the working process. The working mode of partial renewal and global collaboration is established to achieve version collaboration of distributed submodel and global model. The system has been implemented on the BIMServer-an open source platform. The simulation results show that, the system has obvious advantages in many aspects: submodel support, version response time, storage resource occupancy and network transmission.

Key words: building information modeling(BIM); version control; distributed; submodel

(责任编辑: 康　锋)