DOI: 10.13482/j.issn1001-7011.2019.09.042 投稿网址: http://hdzrxb.cbpt.cnki.net

基于SMV模型检测工具的分布式动漫渲染系统软件建模与分析

洪志国, 王永滨, 石民勇, 于水源
(中国传媒大学 计算机与网络空间安全学院, 北京 100024)

摘 要: 利用网络资源搭建分布式动漫渲染系统是提升渲染速度、克服动漫制作效率瓶颈的有效方式。分布式动漫渲染系统软件的健壮性和可用性是渲染系统稳定高效运行的重要保障。因此,从模型检测角度对软件开发进行建模与分析将有效地预防和消除程序中的Bugs,保证程序设计的正确性。基于模型检测方法对系统进行了建模,采用计算树逻辑(Computational tree logic, CTL)对系统待验证的性质进行了描述,并进一步通过符号模型检查(Symbolic model verification, SMV)工具验证了所构建模型的相关性质,为提高系统软件开发的正确性提供了重要的理论依据。

关键词: 模型检测; 动漫渲染; SMV; 状态机
中图分类号: TP311.1 **文献标志码:** A **文章编号:** 1001-7011(2020)03-0362-05

SMV model checking tool-based modeling and analysis of distributed animation rendering system software

HONG Zhiguo, WANG Yongbin, SHI Minyong, YU Shuiyuan
(School of Computer Science and Cybersecurity, Communication University of China, Beijing 100024, China)

Abstract: Utilizing network resources to build distributed animation rendering system is an effective way to improve the rendering speed and overcome the bottleneck of producing animation. The robustness and availability of distributed animation rendering system is an important guarantee for the stable and efficient operation. Therefore, from the perspective of model checking in software development for modeling and analysis, it will effectively prevent and eliminate bugs in the program, ensuring the correctness of programming. Based on model checking method for system modeling, the properties to be verified of system is described using computational tree logic (CTL). Furthermore, the related properties are verified with symbolic model verification (SMV). This methodology offers important theoretical accordance to improve the correctness of software development.

Keywords: model checking; animation rendering; SMV; the state machine

收稿日期: 2016-05-25
基金项目: 国家科技支撑计划课题资助项目(2013BAH54F03);中国传媒大学优秀中青年教师培养工程资助项目(YXJS201508);中国传媒大学理工科规划项目(3132015XNG1504)
通讯作者: 洪志国(1977-),男,副教授,博士,主要研究方向:大数据技术、数学建模、网络性能分析、计算机仿真和动漫渲染,E-mail: hongzhiguo@cuc.edu.cn
引文格式: 洪志国,王永滨,石民勇,等. 基于SMV模型检测工具的分布式动漫渲染系统软件建模与分析[J]. 黑龙江大学自然科学学报,2020,37(3):362-366.

0 引　言

“计算密集型”的渲染阶段因其计算复杂度大而成为动漫产品制作和生产环节的主要瓶颈之一。分布式动漫渲染系统/平台充分利用网络节点[1]、集群存储系统[2]或云计算平台[3-5]执行渲染计算,是提升渲染效率的有效手段。最大限度地提高分布式动漫渲染系统的效率是动漫制作者和渲染服务提供者共同追求的目标。较短的渲染时延能让动漫制作者快速地欣赏其动漫作品的产品级渲染输出,极大地改善动漫制作者对动漫渲染系统的使用体验。在研发分布式动漫渲染系统环节,往往因为顶层设计考虑不周、集成了不同的含 Bugs 的模块等原因导致系统运行的异常。基于黑盒测试和白盒测试等方式可以发现软件中的部分 Bugs,但因相关测试手段不能覆盖所有可能路径而难以发现部分潜在的 Bugs。

模型检测(Model checking, MC)方法从理论上保证了软件中各模块(线程)之间的时序逻辑,对提高软件的健壮性和可用性具有重要的指导作用。模型检测是一种面向有穷状态并发系统的验证技术,是形式化验证中非常重要的一种方法。它的研究始于 20 世纪 80 年代初,Clarke 等提出了用于描述并发系统性质的 CTL 逻辑,设计了检测有穷状态系统是否满足给定 CTL 公式的算法,并实现了一个原型系统 EMC[6]。这一工作为对并发系统的性质进行自动验证开辟了一条新的途径,成为近二十年来计算机科学基础研究的一个热点[7]。目前,模型检测技术被应用于硬件设计[8-9]、协议分析[10]、分布式软件系统交互行为分析[11]、软件开发[12-13]和 Web 服务[14]等方面。SMV 是一款流行的模型检测分析软件,它采用描述有限状态并发系统的规范语言,支持对所建模型相关性质进行验证。

1 分布式动漫渲染系统的软件架构

采用 B/S 架构搭建 Web 服务器提供渲染状态信息,便于用户(动漫制作者)查看所提交场景的渲染进度。渲染管理软件和渲染节点之间通过网络 Socket 进行通信。采用如图 1 所示的分布式动漫渲染系统软件架构,其中实线表示“控制流(请求/响应)”,虚线表示“数据流(渲染输入/输出)”。

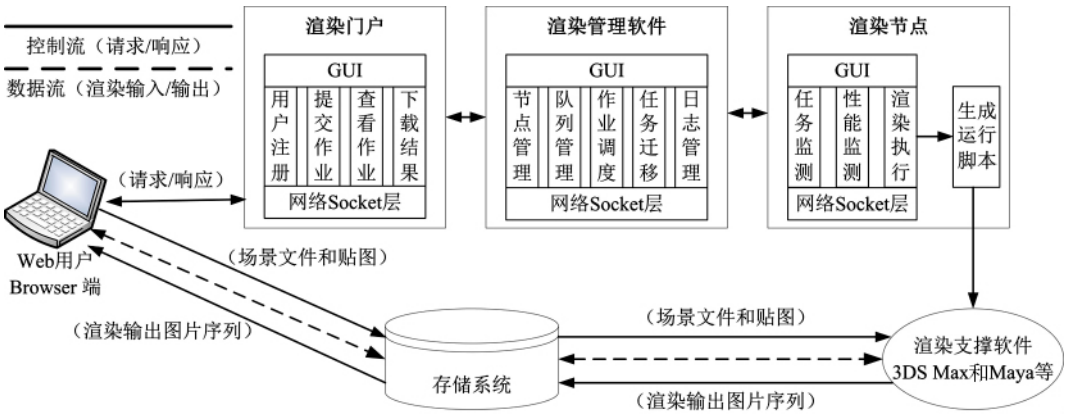

图 1　分布式动漫渲染系统软件架构

Fig. 1　Software architecture of distributed animation rendering system

分布式动漫渲染系统如图 1 所示,包括 Web 用户 Browser 端、渲染门户、存储系统、渲染管理软件、渲染节点、渲染支撑软件 3DS Max 和 Maya 等。用户通过渲染门户提供的 Web Services 进行作业的提交,提交过程包括两部分:(1)向渲染管理服务器提交与作业有关的一些信息,如作业名、总帧数、三维制作软件类型、优先级、内容分类和渲染引擎等;(2)通过 FTP 将采用 Maya 或 3DS Max 制作的渲染工程文件和素材上传到存储系统。

用户在提交的场景中常采用绝对路径引用了相关贴图集,这样当场景文件和贴图集分发到不同节点执行渲染计算时,会因为缺少贴图导致渲染异常或渲染输出错误。因此,为提高渲染系统输出的正确率,需要对场景文件进行预处理: 重新打开并编辑场景文件,采用 3DS Max 或 Maya 等工具的内嵌式脚本对贴图路径

进行修改，便于场景的正常渲染。对于较复杂的场景而言，渲染计算将占用大量 CPU 或 GPU 以及 RAM 等资源，该环节不允许其他程序(线程)对场景文件进行编辑。由此可见，预处理和渲染过程分别对场景文件进行了占用，这两个过程对场景资源具有互斥使用的约束。类似地，预处理过程和渲染过程中内容的不同状态存在着如消息触发和时序约束：渲染管理服务器只有等从 FTP 服务器下载到完整的场景文件后才能进入预处理。

2 模型检验

以预处理和渲染计算为主要分析对象，基于 SMV 工具进行建模和分析。为提高分布式动漫渲染系统的效率，需要有并发执行的线程来完成渲染计算任务。设置 Preprocessing(预处理)线程和 Rendering(渲染)线程。Preprocessing 线程和 Rendering 线程共同拥有作业状态信息表，可以对其进行操作。

2.1 有限状态机的消息定义

为了便于信息表的读写，使用 XML 格式的文件表示作业状态信息表，这两个线程对该文件的操作也是互斥的。基于渲染业务流程，分别构建了如图 2 和图 3 所示的有限状态机。Preprocessing 线程受消息驱动的有穷状态机如图 2 所示，Preprocessing 线程的状态集合为：{等待(Waiting)、初始化(Initializing)、处理(Processing)、退出(Exiting)}；Rendering 线程受消息驱动的有穷状态机如图 3 所示，Rendering 线程的状态集合为：{等待(Waiting)、初始化(Initializing)、渲染(Rendering)、退出(Exiting)}。

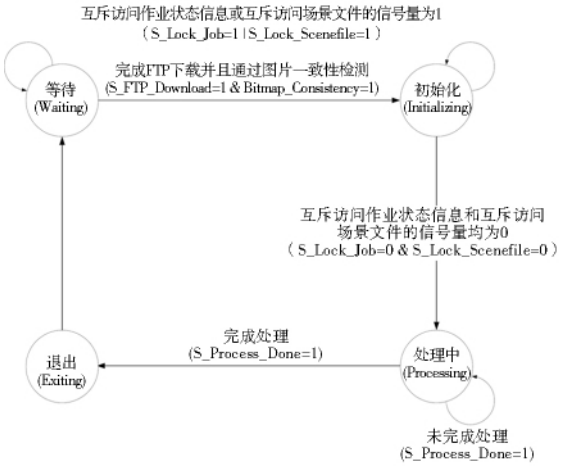

图 2 Preprocessing 线程受消息驱动的有穷状态机

Fig. 2 Finite state machine of Preprocessing thread driven by messages

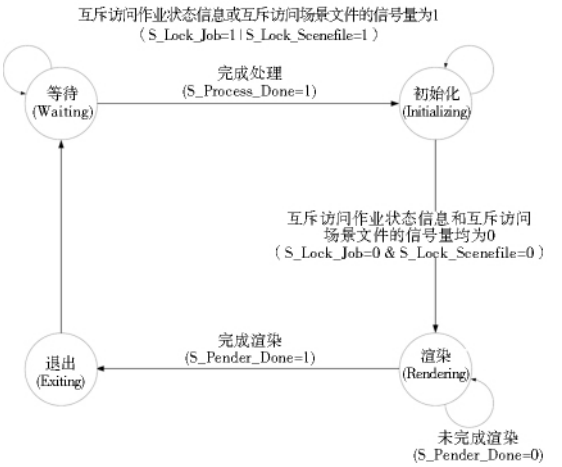

图 3 Rendering 线程受消息驱动的有穷状态机

Fig. 3 Finite state machine of Rendering thread driven by messages

2.2 消息驱动分析

程序中的两个线程各状态之间的转移基于消息驱动原理，当转移条件满足时，实现一个状态向另一个状态的迁移，否则该状态处于自等待状态。Preprocessing 线程和 Rendering 线程响应的消息说明如表 1 所示。

表 1 Preprocessing 线程和 Rendering 线程的响应消息说明

Table 1 Description of messages of Processing and Rendering threads

变量	类型	代表意义
S_FTP_Download	Boolean(布尔型)	场景文件是否从 FTP 服务器成功下载
S_Bitmap_Consisitency	Boolean(布尔型)	是否完成了渲染场景中贴图集的一致性检查
S_Lock_Job	Boolean(布尔型)	用户互斥访问作业状态信息表的信号量
S_Lock_Scenefile	Boolean(布尔型)	用户互斥访问场景文件的信号量
S_Process_Done	Boolean(布尔型)	Preprocessing 线程进行渲染的完成情况
S_Render_Done	Boolean(布尔型)	Rendering 线程进行预处理的完成情况

2.3　系统的 CTL 属性

为了使用 SMV 对系统软件进行检测,给出了如下的规范描述:

(1) 第一则安全性(safety 1) : assert G ~(M_Process. State = Processing & M_Render. State = Rendering);

(2) 第一则活性(Liveness 1) : assert G(M_Process. State = Waiting - > F M_Render. State = Waiting);

(3) 第二则活性(Liveness 2) : assert G(S_Process_Done - > F M_Render. State = Rendering);

M_Process、M_Render 分别表示 Preprocessing 线程和 Rendering 线程。G、F、- >、& 是 CTL 的符号: AG 表示所有路径上的所有状态; AF 表示所有路径上的最后状态; - > 表示逻辑蕴含,即可以推断出某个结论, & 表示逻辑"与"。

这些属性的具体含义为:

(1) Safety 1 表示不存在 Preprocessing 线程的状态为"Processing(处理中)",而且 Rendering 线程的状态为"Rendering(渲染中)"的情况,这样保证了对作业状态信息表和场景文件的互斥访问,避免了系统软件运行时的异常。

(2) Liveness 1 表示如果 Preprocessing 线程完成预处理操作后,那么最终会导致 Rendering 线程也处于等待状态。

(3) Liveness 2 表示如果预处理完成信号量为 1 时,那么最终 Rendering 线程进入渲染状态。即动漫场景完成预处理后,Rendering 线程将会执行渲染计算。

2.4　基于 SMV 的验证

根据图 2 和图 3 所示的有限状态机编写 SMV 格式的代码,在 Windows 7 的 32 位操作系统运行 SMV experimental release 10-11-02P46 版工具进行验证[15],考查分配的二分决策图(Binary decision diagram,BDD)节点数随渲染作业数的变化情况,结果如图 4 所示。可以看出,随着渲染总帧数的增加,模型检测的状态数也相应地增加,表现为检测第一则活性 Liveness 1、第二则活性 Liveness 2 和第一则安全性 Safety 1 所分配的 BDD 节点数整体上均呈现上升的趋势。其中检测 Liveness 2 所分配的 BDD 节点数多于 Liveness 1 和 Safety 1 的 BDD 节点数,而检测 Liveness 1 和 Safety 1 两个属性所分配的 BDD 节点数在各种渲染总帧数的情况下几乎相等。同时,在渲染总帧数分别为 100、200、300、400、500、600、700、800、900 和 1 000 的情况下,通过实验观察到 Liveness 1、Liveness 2 和 Safety 1 的结果都为 True。这说明在总帧数为 1 000 及以下的情况下,根据渲染业务所定义的 Preprocessing 线程和 Rendering 线程的交互模型是正确的、安全的。

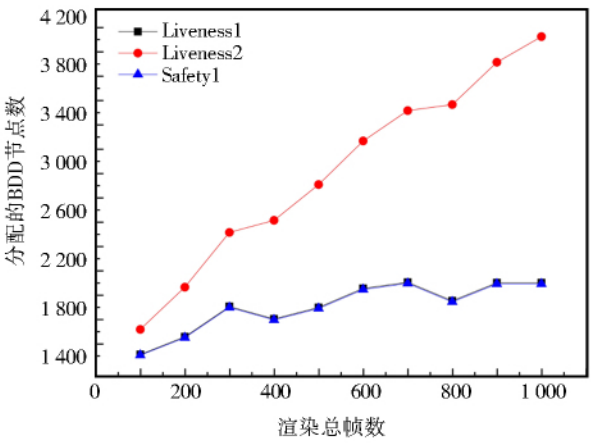

图 4　分配的 BDD 节点数随渲染总帧数的变化情况

Fig. 4　Variation trend of BDD nodes allocated with the increase of total frames of rendering

3　结　论

将模型检测技术应用到分布式动漫渲染系统软件设计中,对系统的预处理和计算过程进行了建模与分

析。该技术基于渲染系统的业务流程，对预处理线程和渲染线程的有限状态机进行了分析，用CTL描述系统的安全性和活性，进一步使用SMV模型检测工具对相关代码进行验证，得出了所设计交互模型是正确和安全的结论。

参考文献

[1] 王永滨，洪志国，曹轶臻，等. 面向广域网的动漫渲染任务分解支持方法及实现系统 [P]. 中国专利：CN201010543756.2，2011－06－08.

[2] 刘刚，李金，李瑞东，等. 一种NAS集群存储系统及其在动漫渲染行业中的应用 [J]. 计算机研究与发展，2012，S1：305－308.

[3] 宋旸，朱毅. 基于云计算的动漫渲染技术的研究 [J]. 电脑知识与技术，2011，30：7448－7449.

[4] 廖宏建，杨玉宝，唐连章，等. 基于云计算的动漫渲染实验平台研究与实现 [J]. 实验室研究与探索，2012，31(7)：68－71.

[5] 刘葵，葛志游. 云计算技术在动漫渲染行业的应用拓展研究 [J]. 广州大学学报(自然科学版)，2013，12(2)：67－70.

[6] CLARKE E M，EMEN E A，SISTLA A P. Automatic verification of finite-state concurrent systems using temporal logic specifications [C]. In ACM Transactions on Programming Languages and Systems，1986，8(2)：244－263.

[7] 林惠民，张文辉. 模型检测：理论、方法与应用 [J]. 电子学报，2002(S1)：1907－1912.

[8] 杨莹，王永滨，闫坚，等. SMV在开发数字电视硬盘机顶盒中的应用 [J]. 系统仿真学报，2005，17(z1)：190－192.

[9] 逄涛，段振华. WISHBONE片上总线符号模型检测 [J]. 计算机研究与发展，2014，51(12)：2759－2771.

[10] 邱卫，杨英杰，汪永伟，等. 基于改进遗传算法和隐Markov模型的协议异常检测方法 [J]. 计算机应用研究，2016，33(4)：1164－1168.

[11] 张琛，段振华，田聪，等. 分布式软件系统交互行为建模、验证与测试 [J]. 计算机研究与发展，2015，52(7)：1604－1619.

[12] 魏欧，石玉峰，徐丙凤，等. 软件模型检测中的抽象模型研究综述 [J]. 计算机研究与发展，2015，52(7)：1580－1603.

[13] 马玲. 分布式在线动漫渲染系统及模型检验 [D]. 北京：中国传媒大学，2010.

[14] 唐郑熠，许道云，王晓峰，等. 基于模型检测技术的语义Web服务自动组合 [J]. 吉林大学学报(工学版)，2013，43(2)：391－396.

[15] MCMILLAN K L. The SMV language [M]. Californy：Cadence Berkeley Labs，1999.