软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2020,31(6):1839−1859 [doi: 10.13328/j.cnki.jos.005641]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于概率模型检验的云渲染任务调度定量验证*

高洪皓[1,2], 缪淮扣[2,4], 刘浩宇[2], 许华虎[2,3], 于芷若[2]

[1](上海大学 计算中心,上海 200444)

[2](上海大学 计算机工程与科学学院,上海 200444)

[3](上海大学 信息化办公室,上海 200444)

[4](上海市计算机软件评测重点实验室,上海 201112)

通讯作者: 许华虎, E-mail: huahuxu@staff.shu.edu.cn

摘 要: 云渲染技术已被广泛应用于影视和动漫等行业.与传统的渲染农场和租赁市场模式不同,云渲染系统依托云计算基础设施提供多种软件服务进行渲染作业的方式,正逐渐成为新兴的计算模式.由于任务执行和资源操作等作业调度对于用户而言是透明的,这要求云渲染系统应具备智能化以实现计算资源优化调度和多端任务管理,并对系统可靠性提出了更高要求.针对这一问题,提出了采用概率模型检验对云渲染系统任务调度进行定量评估.首先,考虑渲染服务失效等因素引发的随机系统异常和指令错误,如文件损坏和渲染任务超时等,提出了基于离散马尔可夫链(DTMC)的概率模型对云渲染系统的文件准备模块、资源请求模块、渲染任务执行模块进行形式化建模;其次,从服务质量属性角度提出了 9 类验证性质用于定义云渲染系统的可靠性,采用概率计算树逻辑(PCTL)描述检验性质公式并执行工具 PRISM 计算和验证渲染系统可靠性;最后,结合案例和实验证明了该方法的可行性和有效性,尤其是对改进前后云渲染系统进行定量检验,可用于指导如何进行失效恢复和任务切换.因此,该方法在一定程度上可提高云渲染系统的可靠性.

关键词: 云渲染系统;任务调度;概率模型检验;PRISM 定量验证;可靠性分析

中图法分类号: TP391

中文引用格式: 高洪皓,缪淮扣,刘浩宇,许华虎,于芷若.基于概率模型检验的云渲染任务调度定量验证.软件学报,2020,31(6):1839−1859. http://www.jos.org.cn/1000-9825/5641.htm

英文引用格式: Gao HH, Miao HK, Liu HY, Xu HH, Yu ZR. Applying probabilistic model checking to the quantitative verification of task scheduling for cloud rendering system. Ruan Jian Xue Bao/Journal of Software, 2020,31(6):1839−1859 (in Chinese). http://www.jos.org.cn/1000-9825/5641.htm

Applying Probabilistic Model Checking to the Quantitative Verification of Task Scheduling for Cloud Rendering System

GAO Hong-Hao[1,2], MIAO Huai-Kou[2,4], LIU Hao-Yu[2], XU Hua-Hu[2,3], YU Zhi-Ruo[2]

[1](Computing Center, Shanghai University, Shanghai 200444, China)

[2](School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China)

[3](Information Office, Shanghai University, Shanghai 200444, China)

[4](Shanghai Key Laboratory of Computer Software Testing and Evaluating, Shanghai 201112, China)

Abstract: Cloud rendering has been widely used as a new computing architecture for the industries of film, television and animation. However, it is different from traditional methods, such as the render farm and rental market, which can provide a variety of rendering

* 基金项目: 国家自然科学基金(61502294, 61572306); 赛尔网络下一代互联网技术创新项目(NGII20170513)

Foundation item: National Natural Science Foundation of China (61502294, 61572306); Innovation Project of Next Generation Internet Technology of CERNET (NGII20170513)

收稿时间: 2018-04-13; 修改时间: 2018-06-12; 采用时间: 2018-08-29

software in the cloud to recede workloads based on cloud infrastructures. In general, task executions and resource operations of task scheduling are transparent to the user. This requires that the cloud rendering system should have the intelligent ability to perform the optimal resources scheduling and multi-terminal tasks management. Thus, the reliability of the cloud rendering system is a core research problem. To this end, the probabilistic model checking technology is employed to carry out the quantitative verification and performance evaluation of the cloud rendering process focusing on task scheduling. First, the rendering service failure will cause stochastic exceptions and instruction errors when cloud rendering is working, i.e., damaged files and task timeout. To this end, the DTMC-based probabilistic model is proposed to formalize the file preparation module, resource request module, and rendering task execution module. Second, considering QoS attributes, nine types of reliability property are introduced to quantitatively verify the cloud rendering system, based on which PCTL is used to describe the verification formula to execute the supporting tool PRISM. Finally, the feasibility and effectiveness of proposed method are demonstrated by case study and experiments, especially the performance of task scheduling can be guaranteed by system recovery and task switching according to the quantitative result generated from formal verifications. Therefore, the proposed method can improve the reliability of the cloud rendering system.

Key words: cloud rendering system; task scheduling; probabilistic model checking; PRISM based quantitative verification; reliability analysis

影视动漫行业的渲染是对视频动画原模型进行贴图、光照、阴影、纹理等逐帧计算的复杂过程[1−3].近年来,AR 和 VR 产业的繁荣发展对计算机性能和 GPU/CPU 计算能力提出了更高的性能要求,例如以 3D 图像和高清视频技术为代表的视频动画,将决定画质最终呈现的视觉效果[4,5].由于 3D 模型和贴图设计越来越细致,渲染任务将消耗大量的计算资源和存储资源,同时也增加了更多时间来输出相关渲染结果文件.但是在互联网+的时代,影视动漫行业的创意具有时效性和紧迫性,其中缩短创作时间、减少资金成本以及提高渲染质量将是这类企业生存之道和发展关键.面对这一问题,越来越多的企业青睐于使用云渲染来降低构建私有渲染系统的经济负担和技术难度.云计算技术被引入到渲染领域[6],利用互联网网络的高速、互联能力,将本应由计算能力有限的个人计算机执行的渲染任务分发给部署在网络的高性能服务器集群上,再通过对计算节点进行统一管理和调度,为用户提供便捷高效的渲染服务.因此,基于云服务实现渲染任务的计算模式正逐渐成为影视动漫行业节约时间和成本的重要实现手段.

云渲染系统提供的渲染服务属于软件即服务(SaaS)层面,以渲染作业为核心向外提供服务,具有资源虚拟化、按需使用、按需付费等特点.用户通过 Web 浏览器或客户端软件,提交需要渲染的模型文件并设定相应参数后,云渲染系统会根据用户要求分配计算资源和存储资源来完成相关渲染任务[7−9].由于互联网的异构性、动态性和协同性,各种异常随时会导致服务执行失效和服务中断,严重影响渲染业务流程整体的正确性和可靠性.因此,云渲染系统需要不断地调整渲染任务和执行动态配置策略,以适应不断变化的用户需求和不稳定的运行环境.其中,云渲染服务提供商通常定义服务等级协议(SLA)来明确服务质量 QoS 参数和需履行的服务职责[10−12].但如何检验渲染业务任务调度是正确的并且满足预期服务质量需求,却是确保渲染任务能动态且高效执行的前提.

失效威胁是云渲染面临的最严峻的挑战,已有学者研究云计算的服务质量:一部分是用户可感知属性的研究[13,14],例如可用性和响应性属性,用户在使用渲染系统的过程中能够直接感受到系统是否可以访问,点击按钮后系统是否能够及时反馈等.另一部分是用户不可感知属性的研究[15],例如系统可靠性属性,表示系统在指定的一段时间内正常运行的概率.由于云服务资源的高度虚拟化,云渲染系统所使用的计算节点、存储设备和系统的具体执行过程对用户而言是透明的,而在其内部服务流程结构复杂且存在随机失效现象.在云渲染系统执行渲染任务的全期间,用户只能看到渲染进度和渲染结果,却很难判断云渲染系统是否按照约定提供了相应质量的服务.因此,需要服务提供商对系统进行评测后将可靠性计算结果反馈给用户.本文考察的可靠性因素是指云渲染系统模块或服务器出现各种失效的随机概率,通过形式化验证方法综合评估系统可靠性,确保向用户提供的渲染服务是可信的.传统软件测试采用数理统计方法得出系统执行成功的次数与总执行次数的比率或系统故障的时间与总运行时间的比率,以此来衡量系统的可靠性.传统软件测试只能使得错误尽可能少地出现,而采用形式化验证方式,尤其是采用概率模型检验技术对云渲染系统任务调度进行定量检验,则可以同时证明相关

服务流程的正确性和可靠性.

基于上述问题,本文针对云渲染系统的任务调度问题[16],对系统执行渲染任务的路径和失效概率状态等采用形式化方法进行建模,定量检验调度任务,评估服务流程性能,找出产生异常的环节或处理节点,提高渲染任务的成功率,并确保云渲染系统的可靠性.首先,针对云渲染系统的文件准备模块、资源请求模块、渲染任务执行模块的概率建模问题,将云渲染系统任务调度的服务流程状态集合分割成“正常”和“异常”两类,采用离散时间马尔可夫链 DTMC 对系统各功能模块进行概率建模;其次,从服务质量属性 QoS 定义角度,结合渲染需求给出多角度的可靠性属性检验公式,使用模型检验工具 PRISM[17,18]计算概率,并验证云渲染系统可靠性;最后,在实验部分,通过分析定量检验结果给出任务调度相关策略,用于指导云渲染系统的任务调度、资源调度和负载均衡.实验对比分析改进前后的系统可靠性,验证本文方法在提高云渲染系统可靠性方面具有较高应用价值.

本文第 1 节介绍了云渲染系统概率建模方法.第 2 节提出了可靠性相关属性定义和其对应的概率计算树逻辑 PCTL 公式.第 3 节通过实验分析证明了本文方法的可行性.第 4 节介绍了相关工作.第 5 节总结了研究内容,提出今后研究方向.

1 云渲染概率模型建模

1.1 案例和动机

本节通过一个平台实例介绍云渲染系统形式化建模和验证的必要性.特别是面向概率失效而引起的系统稳定性问题,急需能对渲染任务调度进行定量评估的解决方案.如图 1 所示是云渲染系统拓扑图,平台主要包括 Web 服务器、文件服务器、平台数据库、节点管理服务器和云渲染资源节点,通过网络实现互联和调度.首先,用户通过终端连接云渲染系统 Web 服务器,将待渲染文件上传到文件服务器中;随后,Web 服务器向节点服务器请求云渲染资源,节点服务器收到请求后分配云渲染资源节点;接着,待渲染任务获得云渲染资源后,由 Web 服务器将用户的渲染任务提交至节点管理服务器,节点管理服务器将任务分割并分发至计算节点执行渲染子任务,云渲染资源中的存储节点保存数据块和渲染子任务执行产生的中间结果;接着,待所有子任务执行完后,节点管理服务器合并存储节点的渲染结果,将最终渲染结果上传至云渲染平台文件服务器;最后,由云渲染平台 Web 服务器通知用户渲染任务完成.

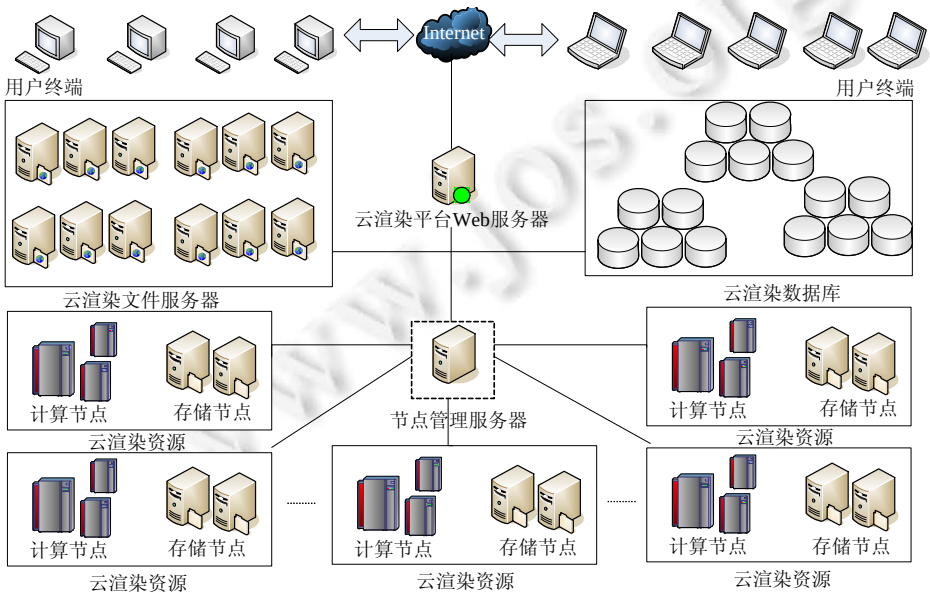


Fig.1 Topology of general cloud rendering system

图 1 云渲染系统拓扑图

然而在云渲染系统环境下,为了完成渲染任务需要庞大的存储资源保存待渲染文件,通过性能强大的 CPU 和 GPU 计算以满足执行大规模渲染任务的需求.高效的任务调度和渲染流程是保证渲染任务执行的前提.为了说明任务调度,如图 2 所示是上海大学承担的上海市科委专项项目云渲染平台所涉及的相关流程.该系统任务调度的服务流程分为校验待渲染文件、请求云渲染资源、渲染子任务分配、节点执行渲染子任务、合并渲染结果等模块.云渲染系统的执行步骤包括开始、校验待渲染文件、请求云渲染资源、获得云渲染资源、分配渲染子任务、执行渲染任务、保存中间结果、合并渲染结果、结束.网络的不稳定性容易产生数据丢包现象,引起数据不一致、文件损坏、执行参数不一致等问题,进而导致渲染帧质量不高、结果输出时间过长等问题,更有可能出现渲染任务无法顺利执行或渲染服务异常等结果.因此,云渲染系统的任务调度流程在关注服务功能执行情况外,还需要考虑异常情况和概率因素.

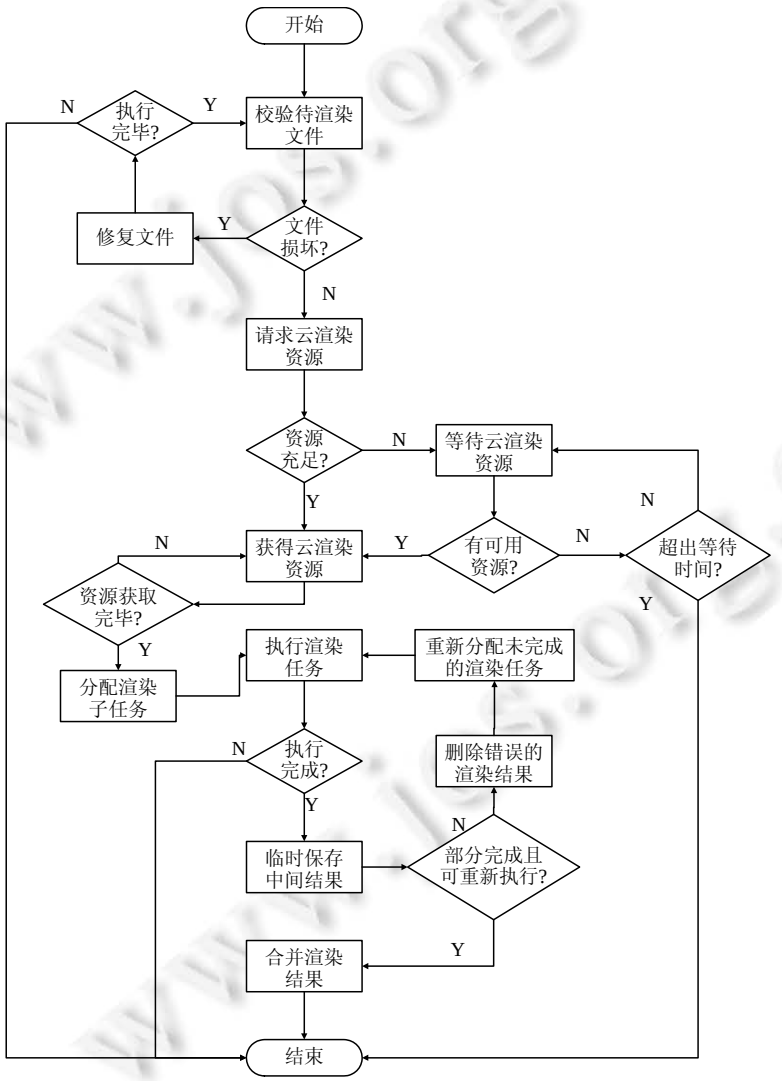


Fig.2 Flow chart of task scheduling in cloud rendering system

图 2 云渲染系统任务调度的流程图

一方面,由于云渲染服务资源是有限的,且渲染任务需要大量的计算和存储支持,常伴有请求资源不足等问题.其中,云渲染资源主要是指执行渲染任务的虚拟主机,是用于保存渲染结果的虚拟存储单元.云渲染任务需

要等待足够云渲染资源后才能继续执行.若等待请求云渲染资源时间过长,表明系统同时执行了较多用户提交的渲染任务,虚拟主机正在执行繁重的渲染任务.云渲染资源的可用性是任务调度的重要考虑因素.另一方面,节点执行渲染子任务不总是成功的,各种软硬件都容易出错致使渲染子任务异常.因此当整个渲染任务完成时,需要检查所有子任务是否都成功执行:若有执行失败的子任务,重新分配这些渲染子任务并尝试执行;当尝试次数过多时,渲染任务被迫中止.只有所有渲染子任务成功执行,产生了正确的结果后,云渲染系统才会合并中间结果并将渲染最终结果交付给用户.因此,云渲染子任务的成功执行率也是任务调度的重要考虑因素.根据上述原因分析,对云渲染系统中不同模块异常与可靠性进行关联分析,给出如下 3 个角度的异常现象.

1) 文件准备模块的异常现象

用户通过网络将待渲染的视频、图片和脚本等原始文件上传到云渲染系统的文件服务器.根据文件大小,一般先被分割成若干小块后存储到存储节点中.考虑到传输或存储文件过程可能由于网络、硬件等不确定因素导致文件损坏或者不兼容,云渲染系统在使用文件之前要先检查文件的有效性.如果校验结果为文件无效或损坏,系统将执行文件修复程序,此时可认为系统进入了异常状态,即文件异常.如果系统发生多次异常或者大量文件异常,则可靠性较低.另一方面,如果修复程序成功地将文件恢复,系统可继续执行下一步操作,即恢复到正常状态.这种异常恢复能力可定义为异常恢复率.当文件异常恢复率较高时,可认为系统可靠性较高.

2) 云渲染资源请求模块的异常现象

云渲染资源包括用于执行渲染任务的虚拟主机(即计算节点)和保存渲染结果的虚拟存储单元(即存储节点).通常,云渲染服务资源是有限的且需要共享,加上渲染任务需要大量的计算和存储支持,可能导致渲染任务无法及时得到满足.因此,若请求资源没有得到满足,渲染任务先进入等待状态.此时可认为云渲染系统进入了异常状态,即资源异常.无期限的等待将是致命的缺陷.若系统能在规定时间内及时分配给目标渲染任务足够的云渲染资源,则可恢复执行下一步操作.若等待请求云渲染资源时间超时,可认为此系统无法承受当前工作负载的渲染任务请求,则直接终止任务.由此可定义资源异常恢复率为:在资源不足的情况下,系统能及时调度获得目标资源的能力.云渲染资源异常率和资源异常恢复率影响着云渲染系统的可靠性.

3) 渲染任务执行模块的异常现象

渲染节点执行子任务不总是成功的,即,有可能子任务虽然执行完毕但是输出了无效的中间结果.因此,检查所有子任务是否都产生了正确有效的结果是必要的.若有的子任务执行失败,则需要重新分配并执行.重新执行子任务也存在一定的失败概率.当多次尝试重新执行子任务,渲染效果仍旧没有达到预期效果,可认为执行失败.如果所有子任务都成功执行且效果正确,则渲染系统合并所有中间结果后得到完整的渲染结果.极端情况下,如果节点执行任务出现严重错误导致整个系统无法正常工作,例如多个节点断电或磁盘损坏等,将直接终止任务.渲染异常率表示节点执行渲染子任务失败的概率.渲染异常恢复率表示当有子任务执行失败时,系统重新分配子任务并且最终执行成功的概率.因此,渲染异常率和渲染异常恢复率也影响着云渲染系统的可靠性.

1.2 基于DTMC的形式化描述

针对失效概率问题,提出概率模型用于云渲染系统形式化建模和验证,从业务模型逻辑角度进行功能建模,从成功执行概率角度进行非功能性建模.云渲染系统任务调度需要虑内部流程结构和渲染服务,因此采用离散马尔科夫链模型 DTMC[19]对云渲染系统任务调度进行流程建模.

定义 1. 云渲染系统任务调度的概率模型定义为六元组:$RM=(S_{normal},S_{exception},s_{init},P,AP,L)$,其中,

1) S_{normal} 是有限非空状态集合,表示云渲染系统处于正常运行的状态空间.包含所有的正常状态,例如请求资源和分配任务等,以及初始状态和成功结束状态;
2) $S_{exception}$ 是有限非空状态集合,表示云渲染系统处于异常运行的状态空间,包含所有的异常状态.除了修复文件、等待资源、重新渲染等异常状态,还包含文件修复失败、资源不足、渲染执行失败等出错状态;
3) $s_{init}\in S_{normal}$ 代表初始状态;
4) $L:S\to 2^{AP}$ 标签函数,用来描述状态上的命题集合,标识具体状态所对应的正在执行的任务操作或出错

情况.其中,AP 是原子命题集合,$S=S_{normal}\cup S_{exception}$ 是所有状态空间集合;

5) $P:S\times S\rightarrow[0,1]$是状态迁移概率函数,表示云渲染系统中从某一个状态是否会迁移转换到另一个状态以及状态迁移概率.

概率模型的每个状态对应云渲染系统任务调度的一个任务节点,每次任务执行视为执行渲染服务或使用资源.该云渲染系统模型描述了流程状态、状态变迁信息和概率.此外,将系统状态集合分成正常和异常两种状态集,用执行步骤与出错描述作为状态,并加入错误状态描述出现的异常情况.为了将任务调度的流程转化为相应的概率模型,给出了如下几种通用转化规则:

定义 2. 云渲染系统任务调度的形式化建模转化规则包括顺序执行、条件分支和循环操作等.

1) 顺序执行:按照渲染流程任务依次执行操作.图 3 给出了顺序任务到概率模型的转换规则;
2) 条件分支:当流程中遇到条件判断时会执行两种不同的渲染操作,从而产生两个不同的计算结果.在概率模型中可以直接展现成两条概率路径,但要求状态转移概率值相加为 1.图 4 给出了条件分支任务到概率模型的转换规则,其中,p 和 $1-p$ 分别是两个分支的发生的概率;
3) 循环操作:当流程中遇到条件判断时,执行结果中有一分支是返回执行跳转前的步骤以构成循环.图 5 给出了循环任务到概率模型的转换规则,其中,$1-p$ 是发生自循环的概率.

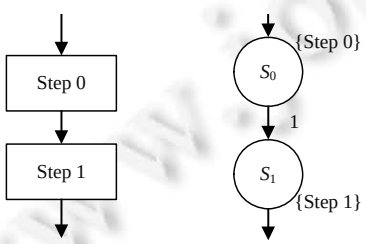


Fig.3 Transformation rule for sequence

图 3 顺序任务转化

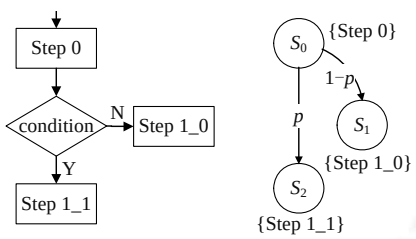


Fig.4 Transformation rule for branch

图 4 条件分支任务转化

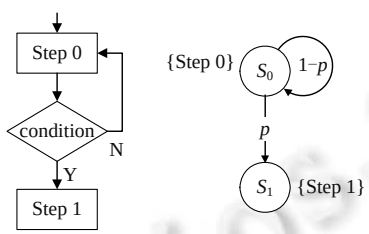


Fig.5 Transformation rule for loop

图 5 循环任务转化

1.3 云渲染系统概率模型

根据上节定义 1,结合图 2 所示平台,将云渲染主要流程分为 3 个模块:准备待渲染文件、获取云渲染资源、执行渲染任务.表 1 所示云渲染系统模型的状态定义信息.

定义 3. 将云渲染系统模型的状态定义分为:

1) 云渲染系统的执行成功状态记为 s_{succ},失败状态记为 s_{fail};
2) 其他状态用 $s(a,b)$表示,表示 a 为功能模块的编号,b 为 a 功能模块中某一步操作或者某一错误的编号.

结合定义 2 转化规则,如图 6 所示是图 2 云渲染系统的概率模型实例.

Table 1 State definition for cloud rendering system

表 1 云渲染系统模型的状态定义表

编号	模块名称	执行步骤与出错描述	状态名称
1	准备待渲染文件	校验文件	$s(1,1)$
		修复文件	$s(1,2)$
		文件无效	$s(1,3)$
2	请求云渲染资源	请求云渲染资源	$s(2,1)$
		等待云渲染资源	$s(2,2)$
		获取云渲染资源	$s(2,3)$
		云渲染资源不足	$s(2,4)$
3	执行渲染任务	分配渲染子任务	$s(3,1)$
		执行渲染子任务	$s(3,2)$
		保存中间结果	$s(3,3)$
		删除错误渲染结果	$s(3,4)$
		重新分配渲染子任务	$s(3,5)$
		合并渲染中间结果	$s(3,6)$
		渲染任务执行失败	$s(3,7)$

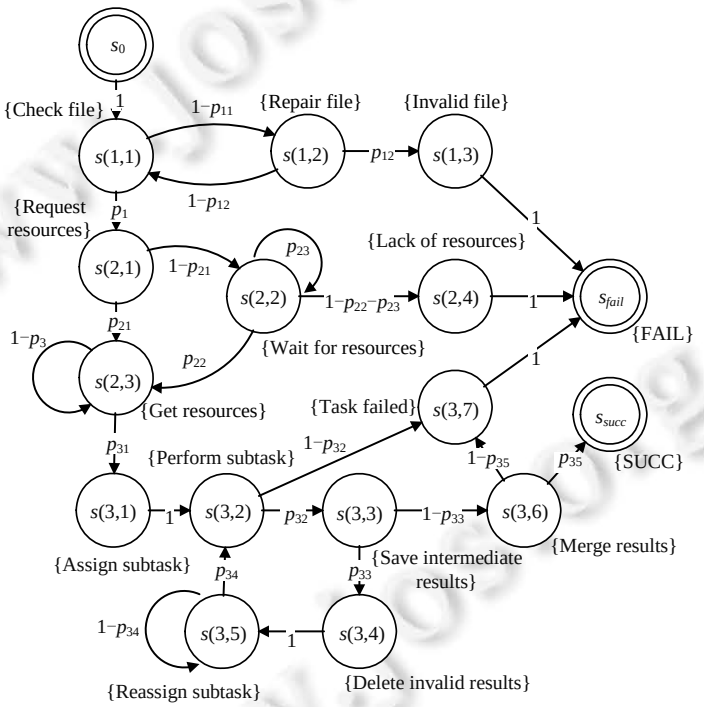


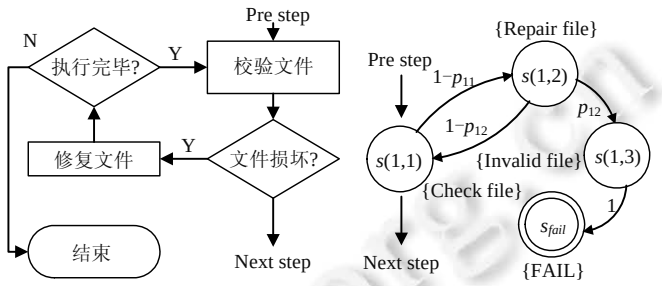
Fig.6 An example of probabilistic model for cloud rendering system

图 6 云渲染系统的概率模型实例

如图 7 所示是图 6 模型的准备渲染文件模块,包括校验文件完整性和修复损坏的文件.该模块有两个执行步骤和一个出错描述,分别对应状态 $s(1,1)$,$s(1,2)$,$s(1,3)$.校验文件时可能会出现文件损坏异常,需要执行修复文件操作以恢复到正常状态.如果修复操作执行失败,云渲染系统无法继续执行任务,则进入失败状态.因此,在跳转到失败状态 s_{fail} 前加上一个错误状态 $s(1,3)$,用于描述系统运行失败是由于文件无法修复导致的.而状态 $s(1,2)$ 到 $s(1,1)$ 变化描述了可修复的概率迁移.

如图 8 所示是图 6 模型的请求云渲染资源模块,包括请求云渲染资源、等待云渲染资源和获得云渲染资源这 3 个步骤,分别对应概率模型中的状态 $s(2,1)$,$s(2,2)$,$s(2,3)$.在请求云渲染资源时,可能因资源被其他渲染任务

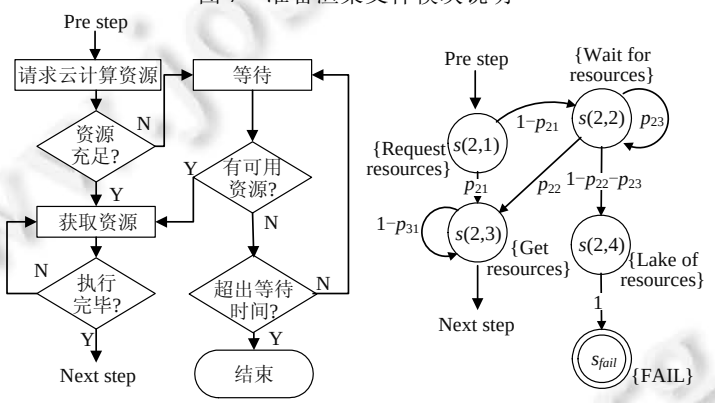
占用而无法足够分配,使得系统会进入等待云渲染资源的异常状态.当等待时间超出限制时间时,系统返回云渲染资源不足的错误警告,表示运行失败.因此,在失败状态 s_{fail} 前添加一个云渲染资源不足的错误状态节点.此外,等待云渲染资源和获取云渲染资源是耗时操作,所以状态 $s(2,2)$和 $s(2,3)$存在自循环.



(a) 模块1的执行步骤 (b) 模块1的概率模型

Fig.7 Module of rendering file preparation

图 7 准备渲染文件模块说明



(a) 模块2的执行步骤 (b) 模块2的概率模型

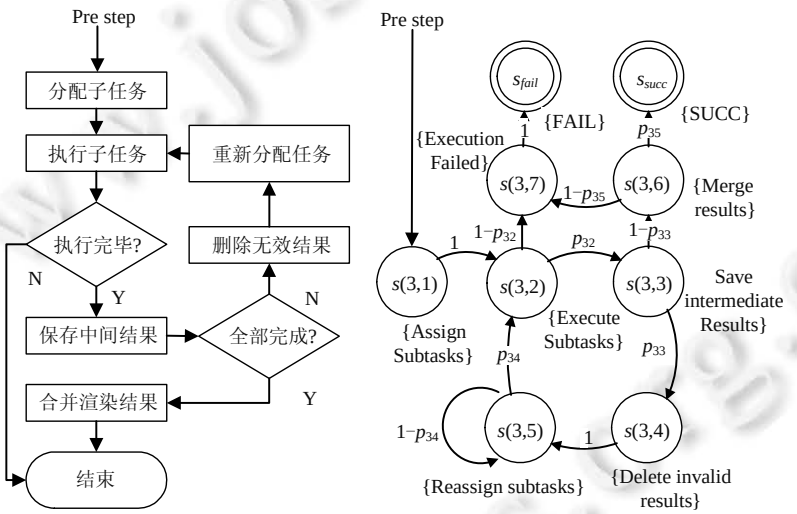
Fig.8 Module of rendering resource request

图 8 请求云渲染资源模块说明

如图 9 所示是图 6 模型的渲染任务执行模块,包括分配渲染子任务、执行渲染子任务、保存中间结果、删除错误中间结果、重新分配未完成的渲染子任务、合并渲染结果等,分别对应状态 $s(3,1)$,$s(3,2)$,$s(3,3)$,$s(3,4)$,$s(3,5)$,$s(3,6)$.执行渲染子任务时,可能因为程序漏洞或硬件损坏导致渲染任务执行失败,所以在失败状态 s_{fail} 前添加渲染执行失败的状态 $s(3,7)$.然而即使渲染子任务执行成功了,也不能保证产生的中间结果是正确的,在这种情况下,系统会尝试重新分配这些渲染子任务给计算节点执行.如果尝试次数超出允许的最大次数,则进入状态 $s(3,6)$表示合并正确的中间结果.如果渲染子任务没有全部完成,只合并了部分中间结果,也视为执行失败,即状态 $s(3,6)$仍有一定概率迁移到错误状态 $s(3,7)$.当所有渲染子任务都产生正确结果,则视为系统运行成功,即状态 $s(3,6)$迁移到成功状态 s_{succ}.

概率模型状态转移的概率参数值可以从实际云渲染系统中统计获得.在概率模型的状态迁移概率矩阵 P 中,行列分别对应云渲染系统概率模型的状态,其中,p_{ij} 表示状态 i 迁移到状态 j 的概率值.如下矩阵 P 是图 6 云渲染系统的概率模型各状态之间的迁移概率信息示例.

$$P=\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & p_{11} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-p_{11} & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 1-p_{21} & p_{21} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & p_{23} & p_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-p_{22}-p_{23} & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 1-p_{31} & p_{31} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & p_{32} & 0 & 0 & 0 & 0 & 0 & 0 & 1-p_{32} & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p_{33} & 0 & 1-p_{33} & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & p_{34} & 0 & 0 & 1-p_{34} & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p_{35} & 0 & 0 & 1-p_{35} & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 1-p_{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p_{12}\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix}.$$



(a) 模块3的执行步骤 (b) 模块3的概率模型

Fig.9 Module of rendering task execution

图 9 执行渲染任务功能模块说明

2 可靠性检验性质描述

可靠的渲染服务不仅满足预期用户对渲染作业的功能性需求,而且在受到干扰(外部攻击、错误操作或网络环境影响)时,具有连续提供渲染服务的能力,不会出现用户需求以外的错误行为.本节考虑云渲染服务质量属性,基于 PCTL 性质描述方法从系统运行结果、系统出现异常的概率、异常恢复概率这 3 个方面定义可靠性检验公式.

2.1 概率计算树逻辑PCTL

概率计算树逻辑 PCTL 用概率运算符号 $P_{\sim p}$ 定量地扩展了计算树逻辑(CTL)的路径量词 A(all)和 E(exists).基于 PCTL 能够定量描述系统性质以及概率事件等.

定义 4. 每个 PCTL 概率计算树逻辑公式由原子命题、逻辑运算符、概率运算符以及时态运算符构成[18,19],通过以下公式构造获得:

1) 命题常量{true,false}和原子命题 p 是 PCTL 公式;
2) 如果ϕ,φ是 PCTL 性质,则$\neg\phi,\phi\wedge\varphi,\phi\vee\varphi,\phi\rightarrow\varphi,\boldsymbol{X}\phi,\boldsymbol{F}\phi,\boldsymbol{G}\phi,\phi\boldsymbol{U}\varphi$和$\phi\boldsymbol{U}^{\leqslant k}\varphi$是 PCTL 公式;
3) 如果φ是 PCTL 性质,则 $P_{\sim p}(\varphi)$是 PCTL 公式,称为极值公式;
4) 如果φ是 PCTL 性质,则 $P_{=?}(\varphi)$是 PCTL 公式,称为计算公式.

每个概率计算树逻辑 PCTL 公式由概率运算符和路径公式组成,其中,路径操作符包括 ***X***(next),***F***(future),***G***(globally),***U***(until)这 4 种.概率运算符分为两类.

- 第 1 类是有关定量判断限定符$\sim\in\{<,\leqslant,>,\geqslant\}$,描述满足性质的路径概率是否符合用户指定阈值.概率操作符$\sim p$表示可达概率,其中,$p\in[0,1]$是给定的限定(界限)概率值.$k\in\mathbb{N}_{>0}$表示路径的步长,$\leqslant k$表示在 k 步执行之内;
- 第 2 类是有关计算定量数值符号=?,计算满足性质的路径概率值.

概率模型检验工具 PRISM 已实现对带概率约束的系统行为进行建模和验证,支持对 PCTL,PCTL*等定量性质的执行.实例说明:如性质公式 $\boldsymbol{P}_{\leqslant 0.75}(\text{true}\boldsymbol{U}^{\leqslant 5}login)$表示至少在 5 次操作内成功并登录的概率最大值为 0.75,而性质公式 $\boldsymbol{P}_{=?}(\text{true}\ \boldsymbol{U}\ \text{fail})$表示系统出现故障的概率;如性质公式 $\boldsymbol{P}_{=?}(\boldsymbol{F}\ TaskFailed\ )$表示计算将来出现任务失败的概率;又如,性质公式 $P_{=?}(\boldsymbol{F}(RequestResources\rightarrow\boldsymbol{F}\ GetResources))$表示计算发出请求资源后获得资源的概率.

2.2 系统运行结果描述

云渲染系统执行渲染任务成功时会输出渲染结果,而失败分两种情况:一是执行渲染任务之前出错,导致任务提前终止;二是执行部分渲染子任务出错,导致系统只输出部分渲染结果并终止.针对云渲染系统运行状态和渲染作业结果,基于 PCTL 语法定义以下定量检验性质.

性质 1. 系统运行成功的概率:

$$P_{=?}[F\leqslant step(exception\leqslant upper\ \&\ all\_complete=\text{true})].$$

性质公式描述了系统状态迁移 *step* 步内,即在执行渲染任务过程中,异常次数未超出限制且完成了全部渲染子任务的概率.此时,若系统运行成功的概率设置为大于 95%,则通过比较执行结果与 95%得出定量结论.

性质 2. 系统运行失败的概率:

$$P_{=?}[F\leqslant step(exception\leqslant upper\ \&\ s=15\ \&\ part\_complete=\text{false})].$$

性质公式表示系统状态迁移 *step* 步内,即在执行渲染任务过程中,异常次数未超出限制、且未完成任何渲染子任务就到达失败状态的概率.其中,s=15 表示失效状态,其他相关编码见后文表 4 所示.此时,若系统运行失败的概率设置为小于 1%,则通过比较执行结果与 1%得出定量结论.

性质 3. 系统不能完成全部渲染子任务概率:

$$P_{=?}[F\leqslant step(exception\leqslant upper\ \&\ part\_complete=\text{true})].$$

性质公式表示系统状态迁移 *step* 步内,即在执行渲染任务过程中,异常次数未超出限制、但只完成部分渲染子任务的概率.此时,若系统不能完成全部渲染子任务概率设置为小于 95%,则通过比较执行结果与 95%得出结论.

上述性质 1~性质 3 的公式中,*step* 表示系统状态迁移的步长,*exception* 表示系统出现异常状态的总次数,*upper* 表示允许系统出现异常最大次数.布尔变量 *all_complete* 表示系统是否完成所有的渲染子任务,布尔变量 *part_complete* 表示系统是否只完成部分渲染子任务.

2.3 系统出现异常描述

云渲染系统进入异常状态主要分为 3 种情况:(1) 接收用户提交的云渲染任务和存储待渲染文件的过程中有一定机率造成文件损坏,使得渲染文件不可用,若修复文件失败,则整个渲染任务无法进行;(2) 请求云渲染资源时,可能会遇到计算资源不足的情况,导致当前云渲染任务长时间等待分配云渲染资源,整个渲染任务无法正常进行;(3) 由于计算节点配置各不相同,计算能力也有差异,不能保证每个渲染子任务都成功执行并得到正确结果.整个渲染任务结束后,系统将尝试重新分配和执行未完成的子任务,但可能发生多次尝试后仍然失败的情

况.针对云渲染系统的上述异常情况,基于 PCTL 语法定义以下定量检验性质.

性质 4. 用户上传的文件损坏的概率:

$$P_{=?}[F\leqslant step(repair\_file>0\ \&\ exception>0)].$$

性质公式表示系统状态迁移 *step* 步内,即在执行渲染任务过程中,出现过异常且修复过文件的概率.此时,若用户上传的文件损坏的概率设置为小于 1%,则通过比较执行结果与 1%得出结论.

性质 5. 系统等待分配云渲染资源超时的概率:

$$P_{=?}[F\leqslant step(wait\_resources>0\ \&\ exception>0)].$$

性质公式表示系统状态迁移 *step* 步内,即在执行渲染任务过程中,出现过异常且等待过云渲染资源的概率.此时,若系统等待分配云渲染资源超时的概率设置为小于 1%,则通过比较执行结果与 1%得出结论.

性质 6. 节点执行渲染子任务产生不正确结果的概率:

$$P_{=?}[F\leqslant step(rerendering>0\ \&\ exception>0)].$$

性质公式表示系统状态迁移 *step* 步内,即在执行渲染任务过程中,出现过异常且重新尝试过执行渲染子任务的概率.此时,若节点执行渲染子任务产生不正确结果的概率设置为小于 1%,则通过比较执行结果与 1%得出结论.

上述性质 4~性质 6 的公式中,*step* 表示系统状态迁移的步长,*exception* 表示系统出现异常状态的总次数,*repair_file* 表示系统修复文件的次数,*wait_resources* 表示系统进入等待计算资源的状态的次数,*rerendering* 表示系统重新执行失败的渲染子任务的次数.

2.4 系统异常恢复描述

对于云渲染系统这种结构复杂的系统,运行时出现异常是不可避免的.但可靠性也表现为从异常状态中恢复的能力.针对云渲染系统异常恢复,基于 PCTL 语法定义以下定量检验性质:

性质 7. 用户上传的待渲染文件损坏时,系统能够及时修复的概率:

$$\frac{P_{=?}[F\leqslant step(file\_ok=\text{true}\ \&\ repair\_file>0)]}{P_{=?}[F(repair\_file>0\ \&\ exception>0)]}.$$

在该性质公式的百分比中:分母为系统执行过程中出现文件损坏的概率;分子为系统状态迁移 *step* 步内,修复文件且将文件修复完好的概率.整个公式表示系统在进入文件损坏异常时,修复文件成功的概率.此时,若系统能够及时修复的概率设置为大于 95%,则通过比较执行结果与 95%得出结论.

性质 8. 系统等待分配云渲染资源超时后,能通过资源分配策略,经过有限的状态迁移后获得足够资源的概率:

$$\frac{P_{=?}[F\leqslant step(wait\_resources>1\ \&\ resources\_ok=\text{true})]}{P_{=?}[F(wait\_resources>1\ \&\ exception>0)]}.$$

在该性质公式的百分比中:分母为系统执行过程中出现长时间等待云渲染资源情况的概率;分子为系统状态在迁移 *step* 步内,等待云渲染资源且成功获得云渲染资源的概率.整个公式表示系统在进入长时间等待云渲染资源异常时,能成功获取到云渲染资源的概率.此时,若经过有限的状态迁移后获得足够资源的概率设置为大于 95%,则通过比较执行结果与 95%得出结论.

性质 9. 当部分渲染子任务执行失败时,系统通过重新分配子任务给计算节点进行渲染并执行成功,得到正确结果的概率:

$$\frac{P_{=?}[F\leqslant step(rerendering>0\ \&\ all\_complete=\text{true})]}{P_{=?}[F(rerendering>0\ \&\ exception>0)]}.$$

在该性质公式的百分比中:分母为系统执行过程中渲染子任务执行失败的概率;分子为系统状态迁移 *step* 步内,重新执行过渲染子任务且最终完成所有渲染子任务的概率.整个公式表示系统在出现渲染异常时,能重新分配并完成所有任务的概率.此时,若系统通过重新分配子任务给计算节点进行渲染并执行成功,得到正确结果的概率设置大于 95%,则通过比较执行结果与 95%得出结论.

上述性质 7~性质 9 的公式中,*step* 表示系统状态迁移的步长,*exception* 表示系统出现异常状态的总次数,*repair_file* 表示系统修复文件的次数,*wait_resources* 表示系统进入等待计算资源状态的次数,*rerendering* 表示系统重新执行失败的渲染子任务的次数,布尔变量 *file_ok* 表示用户文件是否完好无损,布尔变量 *resources_ok* 表示系统是否获得足够的云渲染资源,布尔变量 *all_complete* 表示系统是否完成所有的渲染子任务.

3 实验与分析

本节实验主要是采用 PRISM 工具执行定量验证,实验对象为图 2 所示流程实例.为了进行仿真实验,本文使用 20 台计算机搭建了小型云渲染系统的存储和计算节点.实验环境具体配置见表 2.

Table 2 Computer configuration of experiment

表 2 实验环境计算机配置

组别	操作系统	CPU	显卡	内存	存储	数量
1	CentOS 7	Intel i5-4590	华硕 GTX750	2G	500G	3
2	CentOS 7	Intel i5-6500	华硕 GTX750	4G	256G	4
3	CentOS 7	Intel i3-6100	华硕 GTX750	2G	1T	3
4	CentOS 7	Intel i3-4160	技嘉 GTX760	2G	500G	3
5	CentOS 7	Intel i5-4460	技嘉 GTX760	2G	1T	4
6	CentOS 7	Intel i3-4160	技嘉 GTX760	4G	1T	3

该云渲染系统运行一周后,从系统日志中得到状态迁移概率矩阵 P 中的各项参数概率值.表 3 是 PRISM 代码中描述概率的变量信息和相关赋值操作.表 4 所示 PRISM 代码中描述变量的值与状态名称对应关系.具体云渲染系统概率模型详见附录 1 的代码.

Table 3 Probability of parameters from system running log

表 3 系统运行日志的参数概率值

const double p_{11}=0.985 8;
const double p_{21}=0.989 2;
const double p_{23}=0.953 7;
const double p_{22}=0.017 3;
const double p_{31}=0.977 6;
const double p_{32}=0.985 8;
const double p_{33}=0.014 3;
const double p_{34}=0.988 9;
const double p_{35}=0.982 9;
const double p_{12}=0.018 9;

Table 4 Mapping between state and value

表 4 状态名称对应表

状态名称	变值
s_0	0
s(1,1)	1
s(1,2)	12
s(1,3)	16
s(2,1)	2
s(2,2)	3
s(2,3)	4
s(2,4)	13
s(3,1)	5
s(3,2)	6
s(3,3)	7
s(3,4)	8
s(3,5)	9
s(3,6)	10
s(3,7)	14
s_{fail}	15
s_{succ}	11

3.1 基于PRISM的可靠性检验实验

得到云渲染系统任务调度的概率模型后,根据第 2 节描述的可靠性检验性质,执行概率模型检验.通过分析执行结果的定量数值,评估云渲染系统是否满足预期可靠性需求.

图 10 是性质 1 的检验结果,其中,横坐标表示执行步长,纵坐标表示概率值.实验分别设置 *upper* 为 0~5.观察结果可知:系统未出现异常状态完成渲染任务的概率低于 95%;系统出现 1 次异常状态完成渲染任务的概率大于 95%;系统出现 2 次及以上异常状态完成渲染任务的概率接近 97%.允许的异常状态次数越多,渲染系统运行成功的概率越高.这是由于云渲染系统具有异常状态修复能力.因此,系统满足性质 1,即该云渲染系统在允许出现异常的前提下,执行渲染任务成功的概率大于 95%.

图 11 是性质 2 的检验结果,其中,横坐标表示执行步长,纵坐标表示概率值.实验分别设置 *upper* 为 1~5.从图

中结果发现:系统状态转移次数在 6 次以内,渲染任务失败的概率非常小,其原因可能是文件异常和云渲染资源不足;而随着状态转移次数的不断增加,系统执行渲染子任务失败的概率明显上升;另外,随着允许出现异常状态最大次数的增加,系统运行失败的概率也随之升高,接近 1.5%.因此,该云渲染系统不满足性质 2,即系统状态迁移 6 步之外,执行渲染任务出错的概率大于 1%.

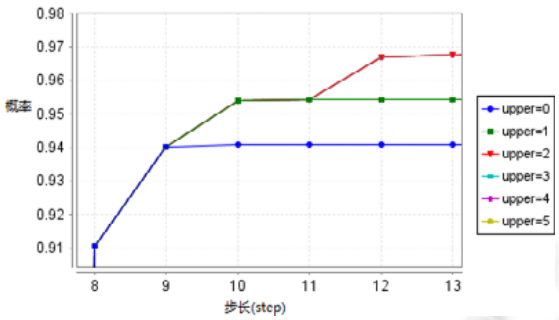


Fig.10 Verification result of property 1

图 10 性质 1 检验结果

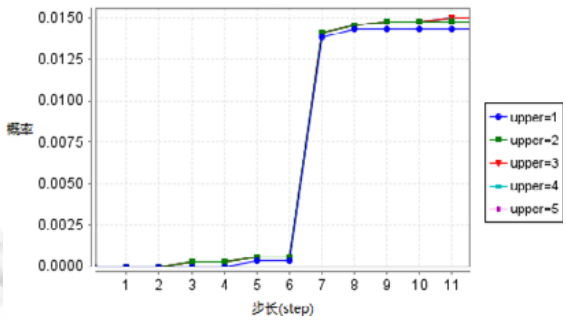


Fig.11 Verification result of property 2

图 11 性质 2 检验结果

图 12 是性质 3 的检验结果,其中,横坐标表示执行步长,纵坐标表示概率值.实验分别设置 *upper* 为 1~5.观察结果可知:随着状态迁移次数的增加,系统不能完成全部渲染子任务的概率不断升高;当系统仅允许 1 次异常时,部分任务失败的概率小于 1.675%;当系统的允许异常次数在 2 次及以上时,部分任务失败的概率超过 1.675%.由于允许异常次数的增加,给了云渲染系统更多的机会去重新执行之前失败的渲染子任务.因此,该云渲染系统不满足性质 3,即系统完成全部的渲染子任务的概率大于 1%.

图 13 展示了性质 4~性质 6 的检验结果.观察结果可知:文件损坏的异常出现得比较早,是因为执行渲染任务前首先要对待渲染文件进行检验;出现长时间等待云渲染资源的概率极小,不足 0.25%;渲染异常与文件异常的概率较大,超过 1.25%.因此,该云渲染系统满足性质 5,但不满足性质 4 和性质 6.即系统运行过程中,发生文件损坏异常和渲染异常的概率大于 1%,而发生长时等待云渲染资源的异常概率小于 1%.

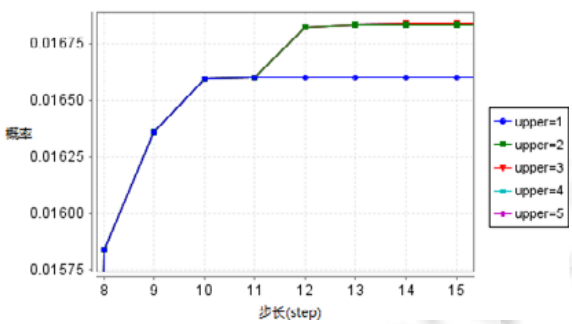


Fig.12 Verification result of property 3

图 12 性质 3 检验结果

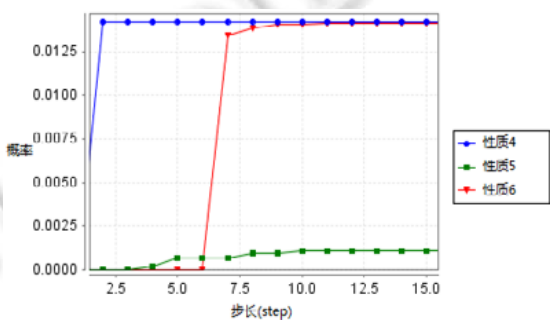


Fig.13 Verification result of property 4~6

图 13 性质 4~性质 6 检验结果

图 14 展示了对性质 7~性质 9 的检验结果.观察结果可知:文件异常较渲染异常能更早得到修复,这也符合云渲染系统的实际任务执行流程.本实验的云渲染系统对 3 种异常的恢复能力相对较高,恢复到正常状态的概率均大于 95%.对文件异常的修复能力也较高,修复成功率大于 97%.因此,该云渲染系统满足性质 7 至性质 9,即系统运行过程中对文件异常,云渲染资源异常以及渲染异常的成功修复概率均大于 95%.

通过对上述实验结果的分析,统计出该云渲染系统可靠性检验结果,见表 5.

在本实验中,云渲染系统满足了 5 条检验性质,即性质公式 1、性质公式 5、性质公式 7~性质公式 9.考虑到

性质公式 2~性质公式 4、性质公式 6 不满足,通过对定量数值进一步分析,给出了如下结论和建议.

1) 从系统执行结果的角度看,系统在理想状态(零异常发生)下执行渲染任务成功率为 94%.出现异常次数越多,表明系统的稳定性越差.由于云渲染系统具备较好的异常状态修复能力,当允许的异常次数增加,系统成功运行的概率也随之提高.允许的异常次数大于 2 时,系统成功运行的概率接近 97%;
2) 从异常率的角度看,文件发生异常的概率比较高,说明文件传输或文件存取模块的程序存在漏洞,也有可能是因为存储节点的硬件质量问题或硬件稳定性不达标.对于防范文件异常而言,需要更换高质量的存储设备并优化文件传输和存取程序.渲染异常和文件异常的发生概率同样高,这可能是由于计算节点的渲染算法或 CPU 和 GPU 计算压力过大导致.对于防范渲染异常,需要优化渲染算法或者考虑选用计算能力更强的计算机作为计算节点.云渲染资源不足的异常率较低,说明在本实验环境下该云渲染系统有充足的计算节点和存储节点,能够满足渲染任务的资源需求;
3) 从异常恢复率角度看,若损坏文件被成功修复的概率很高,说明文件备份和还原修复的功能良好.当渲染任务长时间等待云渲染资源时,若及时分配请求资源的概率较高,表明云渲染资源的分配和调度策略是适用于当前云渲染系统的.当部分渲染子任务失败时,若能及时删除并重新分配子任务,表明云渲染系统对渲染任务的分配和调度能力较高.总体来说,系统进入异常状态后,能恢复到正常状态的概率是评估可靠性的重要指标.

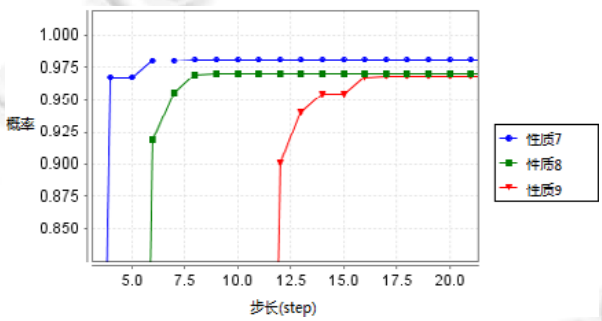


Fig.14 Verification result of property 7~9

图 14 性质 7~性质 9 的检验结果

Table 5 Verification result of all given properties

表 5 云渲染系统可靠性检验结果

编号	性质	结果
1	系统执行渲染任务成功的概率大于 95%	满足
2	系统运行失败的概率小于 1%	不满足
3	系统未能完成全部渲染子任务的概率小于 1%	不满足
4	用户上传的文件损坏的概率小于 1%	不满足
5	系统等待分配云渲染资源超时的概率小于 1%	满足
6	节点执行渲染子任务未产生正确结果的概率小于 1%	不满足
7	待渲染文件损坏时,系统能够及时修复的概率大于 95%	满足
8	系统等待分配云渲染资源超时后,能通过一定的资源分配策略,经过有限的状态迁移获得足够资源的概率大于 95%	满足
9	当部分渲染子任务执行失败时,系统通过重新分配子任务给计算节点进行计算并执行成功,得到正确结果的概率大于 95%	满足

3.2 基于检验结果的改进实验

综合上述 3 个方面的分析可知:如果要提高渲染系统可靠性,需要对文件准备模块和渲染执行模块进行优化配置.本节基于检验结果对云渲染系统提出改进方案以提高系统可靠性,之后再次执行概率模型检验对可靠性检验性质进行校验,以期能提高相关定量数值.

首先,文件准备模块主要执行校验文件和修复文件等操作,所以文件的存储方式和存储节点的性能需要额外关注.在本实验平台下,在文件存储到节点前,系统读取预先设定好的存储块阈值,即根据*BLOCK_SIZE*的值来决定是否分割文件.如果文件大小超过预设大小,则文件会被分割成小于或等于 *BLOCK_SIZE* 大小的数据块后再存储到节点中.此外,数据块会有多个副本存储在不同节点上,但是由于实验系统使用普通机器作为云渲染主机,其存储和读取数据的性能较低.如果数据块副本数量适当增加,也可以提高存储系统的容错率,从而提高存储和读取可靠性.表 6 代码是为云渲染系统的 HDFS 副本数量.

Table 6 Copy parameter setting of HDFS in cloud rendering system

表 6 云渲染 HDFS 副本数量设置

```
…
  ⟨property⟩
    ⟨name⟩ dfs.replication ⟨/name⟩
    ⟨value⟩ 4 ⟨/value⟩ ⟨!--数据块副本数量调整为4--⟩
  ⟨/property⟩
…
```

其次,渲染执行模块主要由计算节点接收和执行渲染子任务,要求执行渲染任务的计算节点能够承担高计算量并拥有足够的稳定性,使得每个子任务顺利执行.本实验系统中的各个设备的 CPU 和 GPU 性能都较低,容易导致执行渲染子任务失败.调整后的实验设备硬件配置见表 7.从表中可以看出:CPU 更换成英特尔 i7 或 i5 处理器,内存为 8G 容量,同时更换了性能较高的显卡.由于本实验系统满足性质 5,即出现云渲染资源不足的异常概率较小,因此可以认为当前系统云渲染资源充足,不需要增加计算机设备数量.表 7 是更新后的实验配置,其中计算机数量与表 2 保持一致.

Table 7 Updated computer configuration for experiment

表 7 更新后的计算机配置

组别	操作系统	CPU	显卡	内存	存储	数量
1	CentOS 7	Intel i5-7600	影驰 GTX960	8G	500G	3
2	CentOS 7	Intel i7-6700	影驰 GTX960	8G	256G	4
3	CentOS 7	Intel i5-6500	影驰 GTX960	8G	1T	3
4	CentOS 7	Intel i7-6700	影驰 GTX960	8G	500G	3
5	CentOS 7	Intel i5-7600	影驰 GTX960	8G	1T	4
6	CentOS 7	Intel i5-7500	影驰 GTX960	8G	1T	3

根据上节内容调整策略后,向系统提交同样数量和类型的渲染任务.待系统运行数周之后,再从系统日志管理模块中获取新的概率参数值.表 8 是改进后云渲染系统的概率变量参数和数值.

Table 8 Probability of parameters after improvement

表 8 改进后的变量参数和概率数值

const double p_{11}=0.994 6;
const double p_{21}=0.990 3;
const double p_{23}=0.952 1;
const double p_{22}=0.028 6;
const double p_{31}=0.983 1;
const double p_{32}=0.987 9;
const double p_{33}=0.008 5;
const double p_{34}=0.993 6;
const double p_{35}=0.992 9;
const double p_{12}=0.004 3;

接着,执行概率模型检验,得到图 15~图 18 的实验数据.

图 15 是对性质 1 的检验结果.对照图 9 可知:改进前后的系统运行成功的概率随系统状态转移步数的变化趋势相似,但是改进后的系统运行成功的概率明显提高.改进之前系统运行成功的概率接近 97%,改进之后的概率超过了 97.5%.因此,系统改进后仍满足性质 1,即系统运行成功的概率大于 95%.

图 16 是对性质 2 的检验结果.对照图 10 可知:系统状态转移次数在 6 次以内,出现任务失败的概率非常小;随着状态转移次数的不断增加,出现系统运行失败概率骤增现象;当允许的异常状态次数增加后,系统运行失败的概率小幅上升;改进前的失败概率接近 1.5%,改进后的失败概率值下降,接近 1%.改进后的云渲染系统虽然仍不满足性质 2,但是可靠性已有小幅度提高.因此,该改进策略方向正确,具有一定的参考性.

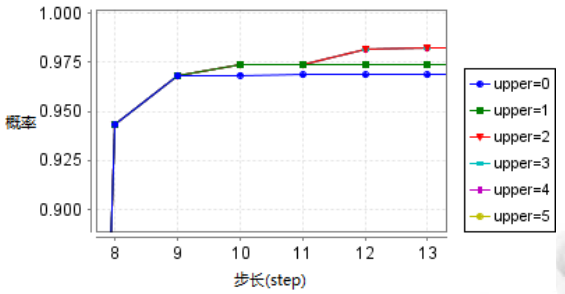


Fig.15 Verification result of property 1

图 15 改进后对性质 1 的检验结果

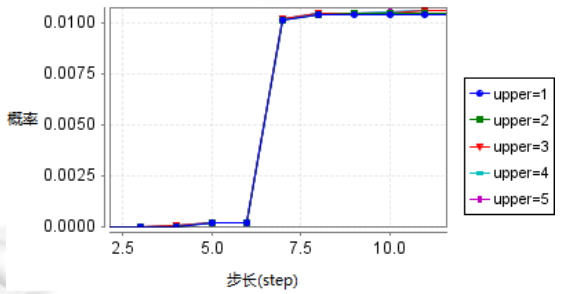


Fig.16 Verification result of property 2

图 16 改进后对性质 2 的检验结果

图 17 是对性质 3 的检验结果.对照图 11 可知:随着系统状态迁移次数的增加,系统不能完成全部渲染子任务的概率不断升高;改进后,当系统仅允许 2 次以内异常时,部分任务失败的概率小于 0.7%;当系统的异常次数在 2 次以上时,系统只完成部分渲染子任务的概率增加,超过了 0.7%,但均没有超过 1%.因此,改进后的云渲染系统满足性质 3,即系统不能完成全部的渲染子任务的概率小于 1%.

图 18 展示了对性质 4~性质 6 的检验结果.对照图 12 可知:系统文件异常和渲染异常发生的概率明显下降;文件异常发生的概率略高于 0.8%,但低于 1%;渲染子任务执行异常发生的概率在 0.5%~0.6%之间;由于系统改进方案没有涉及云渲染资源总量,云渲染资源不足的异常发生概率几乎保持不变.改进后的云渲染系统同时满足性质 4~性质 6,即云渲染系统 3 个功能模块发生异常的概率都小于 1%.

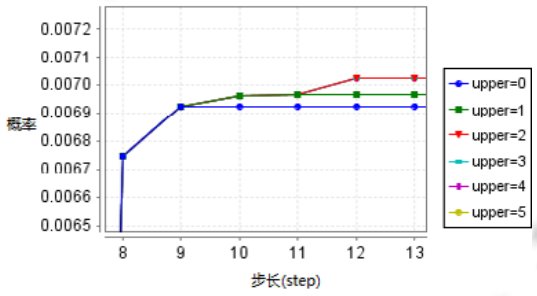


Fig.17 Verification result of property 3

图 17 改进后对性质 3 的检验结果

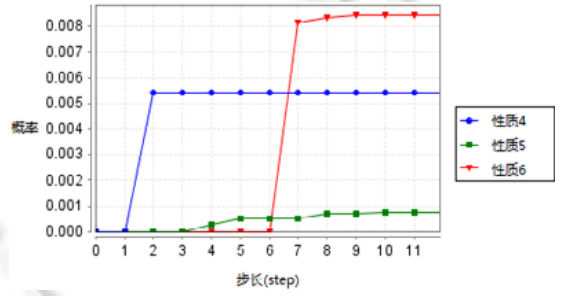


Fig.18 Verification result of property 4~6

图 18 改进后对性质 4~性质 6 的检验结果

图 19 展示了对性质 7~性质 9 的检验结果.对照图 13 可知:系统改进前后各个功能模块的异常恢复率都高于 95%.改进后的异常恢复率小幅提高,其中,由于改进方案中增加了数据块的副本数,文件异常恢复率更加接近 98%.在云渲染资源不足的情况下,由于改进方案中提升了计算节点的 CPU 性能,异常恢复率和渲染子任务执行异常的恢复率也提升到了 97.5%以上.因此,改进后的系统仍然满足性质 7~性质 9,即 3 个功能模块的异常恢复率均大于 95%.

综上所述,表 9 是对检验结果汇总.

总的来说,改进后的云渲染系统满足 8 条可靠性性质.相较于改进前,改进后的云渲染系统的可靠性水平有所升高.改进后的系统同时满足了性质 3、性质 4 和性质 6,说明改进后的云渲染系统出现文件损坏和文件无效的异常概率下降,节点执行渲染子任务失败的概率也有所下降,这些都有助于确保渲染作业正确执行.

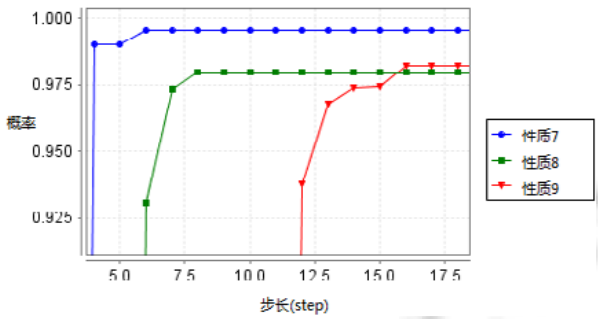


Fig.19 Verification result of property 7~9

图 19 改进后对性质 7~性质 9 的检验结果

Table 9 Verification result of all given properties after improvement

表 9 改进后的系统可靠性性质检验结果汇总

编号	性质	结果
1	系统执行渲染任务成功的概率大于 95%	满足
2	系统运行失败的概率小于 1%	不满足
3	系统未能完成全部渲染子任务的概率小于 1%	满足
4	用户上传的文件损坏或无效的概率小于 1%	满足
5	系统等待分配云渲染资源超时的概率小于 1%	满足
6	节点执行渲染子任务执行失败的概率小于 1%	满足
7	待渲染文件损坏时,系统能够及时修复的概率大于 95%	满足
8	系统等待分配云渲染资源超时后,能通过一定的资源分配策略,经过有限的状态迁移后获得足够资源的概率大于 95%	满足
9	当部分渲染子任务执行失败时,系统通过重新分配子任务给计算节点进行计算并执行成功,得到正确结果的概率大于 95%	满足

4 相关工作

渲染任务是计算密集型和数据密集型作业.传统以单机模式的渲染方法存在计算效率低和耗时等问题.因此,研究人员开始关注分布式计算方法以多机联合辅助形式实现渲染任务.Mahsa 等人[20]提出了一种支持 3D 视频远程渲染模型,能在减少视频传输延迟的同时保障渲染质量.由于网络和云的不确定性,延迟传输是导致系统异常的重要因素,提高可靠性问题是云渲染的任务调度和优化的核心.Donghyeok 等人[21]提出了一种基于移动云的交互式 3D 渲染和流媒体系统,包括支持软件定义网络(SDN)的自适应云资源管理模块和上下文感知的 3D 渲染和流媒体模块,以提高渲染和流服务的服务质量.但该方法的自适应云资源管理和上下文感知涉及大规模流式数据的存储和交换,因此,如何评估最优渲染服务器是非常重要的研究点.Audrius 等人[22]提出了基于 Game Large 架构模式的远程渲染系统.当客户端的硬件或网络不通畅时,系统通过视频流将图形渲染切换到服务器,并将结果压缩为能够在客户端使用的视频流.这一混合模式虽然提高了任务持续性,但是仍然缺乏大规模云端服务器应用的数据交换和服务器性能监控等相关方法和技术.Li 等人[23]提出了一种高效的两层结构的作业调度和任务调度策略(RF-FD)用于集群渲染系统,其中:RF 策略利用低优先级作业使得作业在被阻塞时最大化资源利用率;而 FD 策略利用资源使用反馈来选择渲染器并为其分配相应线程,并将渲染节点划分为细粒度渲染单元以平衡负载分布.从技术角度看,这些策略和方法有助于实现任务调度的可靠性,但是策略制定和自动化执行策略还需要进一步研究和提高.

针对云计算环境,相关分布式协作平台和研究成果正促使传统的渲染服务向云渲染模式方向发展.Liu 等人[24]提出了基于 Hadoop 的大型渲染系统,将场景解析后每一帧渲染对应的场景文件存储到 HDFS 上,利用它的数据本地性减少数据传输消耗的时间;同时,利用 MapReduce 编程模型完成作业的调度.Zhou 等人[25]针对超大规模渲染云问题,提出一种 5 级软件架构的黄金农场集群渲染平台,包括基础架构层、平台服务层、应用服务层、服务管理和访问接口.开发并部署在“天河-1A”超级计算机上,能够管理多达 2 000 个高性能渲染节点,支持

同时使用 24 000 个 CPU 进行并行任务渲染.由于 3D 模型的大小和复杂性,渲染过程耗时且生产力低下.Kijsipongse 等人[26]评估了 BitTorrent 文件系统,通过对等方式传播数据,并使用本地缓存来缩短渲染时间,改进了分布式动画渲染的通信性能.Wu 等人[27]设计了云三维动画渲染系统,通过云计算模式,可以缩减动画的渲染时间,并且显著提升渲染效果.虽然这些平台在一定程度上解决了不同类型的渲染任务,但是值得注意的是:由于网络不稳定性,任务调度的可靠性方面需要进一步研究.

渲染通常在分布式计算环境下进行,其执行大量并行计算来加速渲染作业,且异构网络下的数据服务成本受限.因此,越来越多的学术界和工业界的研究者关注任务调度中数据流和工作负载等方面问题,致力于提高渲染效果.Cristian 等人[28]针对二维图像的输出序列中帧丢失问题,提出了一种并行计算解决方案.该方案由多个基于 GPU 异构节点组成渲染方案实现交替帧渲染.在移动设备上处理 3D 模型较为困难,Yan 等人[29]提出了一种包括自适应分裂和错误处理机制的传输控制方法,并将其与典型的远程渲染系统相集成.该机制可以通过分割操作交易传输来降低传输频率.Vilutis 等人[30]针对任务延迟问题,提出了混合云的工作负载均衡算法用于渲染服务,以确保其基于云的及时交付.Dobashi 等人[31]提出了一种借助求解反演渲染方法来解决渲染参数设置问题,给定非均匀的合成云密度分布,使用遗传算法搜索最佳参数,对渲染合成云的参数进行估计.上述方法虽然考虑到任务/数据可靠性问题,但是在性能计算和评估方面还缺少相关技术支撑.近年来,越来越多的研究人员关注云渲染的可靠性,特别是对网络系统、通信系统引起的不可靠、不可预测性行为进行定量建模和检验.形式化验证技术作为一种可行方法,尤其是概率模型检验技术,能实现对流程进行定量和定性的检验.但目前鲜有概率模型检验技术在云渲染系统上的相关应用研究.与上述工作不同,本文重点关注采用概率模型检验验证云渲染系统任务调度的流程可靠性,并通过定量的验证结果数值指导如何设计云渲染系统优化改进策略,以确保云渲染系统提供可靠的渲染服务.

5 总 结

随着互联网和云计算技术的快速发展,使用云渲染系统有助于企业在较短时间内完成预期的渲染工作,在影视和动画等行业具有广泛应用前景和推广市场.渲染任务执行和调度过程对用户而言是透明的,即提交渲染文件后由渲染系统在云端自动执行相对应的渲染操作并返回渲染结果.然而,网络的不确定性等诸多影响因素使得用户更加关心渲染任务的可靠性,因此,有必要针对云渲染系统任务调度流程进行正确性和可靠性检验.同时,可靠性评估的定量数值可以指导云渲染系统进行任务切换,以确保渲染服务持续和稳定运行.基于此,本文提出了基于概率模型检验的云渲染任务调度定量验证方法.从系统运行成功率、异常率和异常恢复率这 3 个角度分析云渲染系统的可靠性,并定义可靠性检验性质公式,通过 PRISM 执行概率模型检验得到定量结果用于可靠性和系统性能评估.在实验部分,搭建了实验环境,一方面检验和计算云渲染系统可靠性;另一方面,根据性能评估结果执行指导云渲染资源的分配和调度策略.实验证明了本文方法的可行性和有效性,对提高云渲染系统服务可靠性具有指导意义.

在未来工作方面,将增加实验环境的计算节点数量,增加实验用户数和并发量,提交更多的渲染任务,研究分布式渲染数据的收集和处理方法.此外,针对定量检验结果,研究基于服务质量参数的自我评估和自动修正方法,从而增强异常情况下实时任务调度的可操作性.

References:

[1] Song CM, Zhao CW, Liu D, *et al*. Advances in three-dimensional multiscale geometrical analysis. Ruan Jian Xue Bao/Journal of Software, 2015,26(5):1213−1236 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4822.htm [doi: 10.13328/j.cnki.jos.004822]

[2] Sun XP, Wang G, Wang L, *et al*. 3D point cloud shape feature descriptor using 2D principal manifold. Ruan Jian Xue Bao/ Journal of Software, 2015, 26(3):699−709 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4653.htm [doi: 10.13328/j.cnki.jos.004653]

[3] Li DJ. An artistic rendering framework for stereoscopic images. Chinese Journal of Computers, 2014,37(10):2218−2226 (in Chinese with English abstract).

[4] Zhao C, Ma SW, Zhang XF, *et al*. An efficient image compression method based on cloud data. Chinese Journal of Computers, 2017,40(11):2433−2447 (in Chinese with English abstract).

[5] Shi WD, Lu Y, Li Z, *et al*. SHARC: A scalable 3D graphics virtual appliance delivery framework in cloud. Journal of Network & Computer Applications, 2011,34(4):1078−1087.

[6] Guo YK, Han R. Elastic algorithm in cloud computing: Overview and prospect. Journal of Shanghai University (Natural Science Edition), 2013,19(1):1−4 (in Chinese with English abstract).

[7] Gao Z, Sun WJ, Wang JH, *et al*. Optimized interactive remote realistic volume rendering using renderer server and Web server coupling algorithm. Journal of Image and Graphics, 2017,22(3):385−394 (in Chinese with English abstract).

[8] Wang M. A 3D WebGIS system based on VRML and X3D. In: Proc. of the IEEE 2nd Int'l Conf. on Genetic and Evolutionay Computing. IEEE, 2008.197−200.

[9] Shi S, Hsu CH. A survey of interactive remote rendering systems. ACM Computing Surveys, 2015,47(4):57.

[10] Zhang T, Ma JF, Xi N, *et al*. Trust-Based decentralized service composition approach in service-oriented mobile social networks. Acta Electronica Sinica, 2016,44(2):258−267 (in Chinese with English abstract).

[11] Wu ZH, Wu QD, Li P, *et al*. Research and development of service science from the inter-disciplinary of natural science and social science. Bulletin of National Natural Science Foundation of China, 2016,30(6):527−534 (in Chinese with English abstract).

[12] Lin C, Chen Y, Huang JW, *et al*. A survey on models and solutions of multi-objective optimization for QoS in services computing. Chinese Journal of Computers, 2015,38(10):1907−1923 (in Chinese with English abstract).

[13] Calinescu R, Ghezzi C, Johnson K, *et al*. Formal verification with confidence intervals to establish quality of service properties of software systems. IEEE Trans. on Reliability, 2016,65(1):107−125.

[14] Benlian A, Koufaris M, Hess T. Service quality in software-as-a-service: developing the SaaS-qual measure and examining its role in usage continuance. Journal of Management Information Systems, 2011,28(3):85−126.

[15] Cancian MH, Hauck JCR, von Wangenheimm CG, *et al*. Discovering software process and product quality criteria in software as a service In: Babar MA, Vierimaa M, Oivo M, eds. Proc. of the Int'l Conf. on Product Focused Software Process Improvement (PROFES 2010). LNCS 6156, Berlin, Heidelberg: Springer-Verlag, 2010. 234−247.

[16] Xu YM, Li KL, He LG, *et al*. A hybrid chemical reaction optimization scheme for task scheduling on heterogeneous computing systems. IEEE Trans. on Parallel & Distributed Systems, 2015,26(12):3208−3222.

[17] PRISM. http://www.prismmodelchecker.org

[18] Kwiatkowska M, Norman G, Parker D. PRISM 4.0: Verification of probabilistic real-time systems. In: Proc. of the 23rd Int'l Conf. on Computer Aided Verification. LNCS 6806, Berlin, Heidelberg: Springer-Verlag, 2011. 585−591.

[19] Baier C, Katoen JP. Principles of Model Checking. MIT Press, 2008. 745−907.

[20] Shi S, Kamali M, Nahrstedt K, *et al*. A high-quality low-delay remote rendering system for 3D video. In: Proc. of the ACM Int'l Conf. on Multimedia. ACM, 2010. 601−610.

[21] Ho DH, Kim HN, Kim W, *et al*. Mobile cloud-based interactive 3D rendering and streaming system over heterogeneous wireless networks. IEEE Trans. on Circuits and Systems For Video Technology, 2017,27(1):95−109.

[22] Nave I, David H, Shani A, *et al*. Games@large graphics streaming architecture. In: Proc. of the IEEE Int'l Symp. on Consumer Electronics. IEEE, 2008. 1−4.

[23] Li Q, Wu WG, Sun ZY, *et al*. An efficient two-level hierarchy job scheduling and task dispatching strategy for cluster rendering system. Journal of Information Science and Engineering, 2017,33(2):463−483.

[24] Liu WF, Gong B, Hu Y. A large-scale rendering system based on hadoop. In: Proc. of the Int'l Conf. on Pervasive Computing and Applications. IEEE, 2011. 470−475.

[25] Zhou WN, Lu YQ, Gao PD, *et al*. A new software architecture for ultra-large-scale rendering cloud. In: Proc. of the 11th Int'l Symp. on Distributed Computing and Applications to Business, Engineering & Science. IEEE, 2012. 196−199.

[26] Ekasit K, Namfon A. Improving the communication performance of distributed animation rendering using BitTorrent file system. Journal of Systems and Software, 2014,97:178−191.

[27] Wu JQ, Xu HH, Vassilev T. Design of 3D animation rendering platform based on cloud computing. In: Proc. of the 2016 4th Int'l Conf. on Enterprise Systems. IEEE, 2016. 153−159.

[28] Perez-Monte CF, Perez MD, Silvio R, *et al*. Modelling frame losses in a parallel alternate frame rendering system with a computational best-effort scheme. Computers & Graphics, 2016,60:76−82.

[29] Yan YJ, Liang XH, Xie K, *et al*. ASEHM: A new transmission control mechanism for remote rendering system. Multimedia Tools and Applications, 2014,69(3):585−603.

[30] Vilutis G, Sutiene K, Kavaliunas R, *et al*. Load balancing strategy for hybrid cloud-based rendering service. Elektronika IR Elektrotechnika, 2014,20(2):79−84.

[31] Yoshinori D, Wataru I, Ayumi O, *et al*. An inverse problem approach for automatically adjusting the parameters for rendering clouds using photographs. ACM Trans. on Graphics, 2012,31(6):1−10.

附中文参考文献:

[1] 宋传鸣,赵长伟,刘丹,等.3D 多尺度几何分析研究进展.软件学报,2015,26(5):1213−1236. http://www.jos.org.cn/1000-9825/4822.htm [doi: 10.13328/j.cnki.jos.004822]

[2] 孙晓鹏,王冠,王璐,等.3D 点云形状特征的二维主流形描述.软件学报,2015,26(3):699−709. http://www.jos.org.cn/1000-9825/4653.htm [doi: 10.13328/j.cnki.jos.004653]

[3] 李大锦.艺术化立体图像的渲染.计算机学报,2014,37(10):2218−2226.

[4] 赵琛,马思伟,张新峰,等.基于云数据的高效图像编码方法.计算机学报,2017,40(11):2433−2447.

[6] 郭毅可,韩锐.云计算中的弹性算法:概要和展望.上海大学学报(自然科学版),2013,19(1):1−4.

[7] 高瞻,孙万捷,王杰华,等.渲染器与 Web 服务器耦合实现远程体渲染的交互优化.中国图像图形学报,2017,22(3):385−394.

[10] 张涛,马建峰,习宁,等.面向服务移动社交网络中基于信任的分布式服务组合方法.电子学报,2016,44(2):258−267.

[11] 吴朝晖,吴启迪,李平,等.自然科学与社会科学交叉视角下的服务科学研究与发展.中国科学基金,2016,30(6): 527−534.

[12] 林闯,陈莹,黄霁崴,等.服务计算中服务质量的多目标优化模型与求解研究.计算机学报,2015,38(10):1907−1923.

附录 1 PRISM 构建云渲染系统概率模型

根据 PRISM 建模语言语法规则,所构建的云渲染系统概率模型名为“*CloudRenderingSystem*”,其核心代码如下所示.

```
1  module CloudRenderingSystem
2  s:[0..16] init 0;
3  repair_file:[0..max_repair_file] init 0;
4  wait_resources:[0..max_wait_resources] init 0;
5  rerendering:[0..max_rerendering] init 0;
6  exception:[0..max_exception] init 0;
7  all_complete:bool init false;
8  part_complete:bool init false;
9  file_ok:bool init false;
10 resources_ok:bool init false;
11 [Begin]s=0→(s′=1);
12 [DealFile]s=1 & repair_file<max_repair_file & exception<max_exception→p11:(s′=2) & (file_ok′=true)+
   1−p11:(s′=12) & (repair_file′=repair_file+1) & (exception′=exception+1);
13 [RequestResources]s=2 & wait_resources<max_wait_resources & exception<max_exception→1−p21:
   (s′=3) & (wait_resources′=wait_resources+1)+p21:(s′=4);
14 [WaitResources]s=3 & wait_resources<max_wait_resources & exception<max_exception→p23:(s′=4)+
   p22:(s′=3) & (wait_resources′=wait_resources+1) & (exception′=exception+1)+1−p23−p22:(s′=13);
15 [GetResources]s=4→1−p31:(s′=4)+p31:(s′=5) & (resources_ok′=true);
16 [Distribution]s=5→(s′=6);
17 [StartRendering]s=6→p32:(s′=7)+1−p32:(s′=14);
18 [EndRendering]s=7 & rerendering<max_rerendering & exception<max_exception→1−p33:(s′=10)+p33:
   (s′=8) & (rerendering′=rerendering+1) & (exception′=exception+1);
19 [Delete]s=8 & exception<max_exception→(s′=9) & (exception′=exception+1);
20 [Redistribution]s=9 & exception<max_exception→1−p34:(s′=9) & (exception′=exception+1)+p34:(s′=6);
21 [Complete]s=10 & exception<max_exception→p35:(s′=11) & (all_complete′=true)+1−p35:(s′=14) &
```

(*part_complete′*=true);

22 [*RepaireFile*]*s*=12 & *repair_file*<*max_repair_file* & *exception*<*max_exception*→p_{12}:(*s′*=16) & (*exception′*=*exception*+1)+1−p_{12}:(*s′*=1);

23 [*ResourcesException*]*s*=13 & *exception*<*max_exception*→(*s′*=15) & (*exception′*=*exception*+1);

24 [*FileException*]*s*=16 & *exception*<*max_exception*→(*s′*=15) & (*exception′*=*exception*+1);

25 [*RenderingException*]*s*=14 & *exception*<*max_exception*→(*s′*=15) & (*exception′*=*exception*+1);

26 [*FileError*]*s*=12 & *repair_file*=*max_repair_file*→(*s′*=15);

27 [*ResourcesError*]*s*=3 & *wait_resources*=*max_wait_resources*→(*s′*=15);

28 [*RenderingError*]*s*=8 & *rerendering*=*max_rerendering*→(*s′*=15);

29 [*Error*]*exception*=*max_exception*→(*s′*=15);

30 endmodule

代码的第 2 行~第 10 行对模型参数进行定义,具体变量描述见表 10.代码第 11 行~第 22 行为系统状态迁移的定义,其中,在一些特定的状态迁移中加入了对异常计数类参数的自增操作,是为了记录系统到达某一状态时的历史信息.代码 23 行~第 29 行对边界条件进行判断,例如:异常发生的次数超过边界值,则直接进入失败状态.

Table 10 Parameters introduction for *CloudRenderingSystem* model

表 10 *CloudRenderingSystem* 模型的参数变量定义

变量名	变量类型	变量描述
s	int	模型中的状态,值代表状态序号
repair_file	int	系统修复文件的次数
wait_resources	int	系统进入等待计算资源状态的次数
rerendering	int	系统重新执行失败的渲染子任务的次数
exception	int	系统出现异常状态的总次数
all_complete	bool	系统是否完成所有的渲染子任务
part_complete	bool	系统是否只完成部分渲染子任务
file_ok	bool	用户文件是否完好无损
resources_ok	bool	系统是否获得足够的云渲染资源

高洪皓(1985－),男,浙江临海人,博士,副教授,CCF 高级会员,主要研究领域为软件工程,服务计算,概率模型检验,云边协同计算.

缪淮扣(1953－),男,教授,博士生导师,CCF 杰出会员,主要研究领域为软件形式方法,软件工程.

刘浩宇(1993－),男,硕士生,主要研究领域为云渲染系统,系统可靠性.

许华虎(1966－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为网络安全,多媒体,教育信息化,大数据.

于芷若(1994－),女,硕士生,主要研究领域为软件工程,模型检验.