DOI:10.3979/j.issn.1673-825X.2014.01.023

一种针对汽车三维虚拟仿真系统的多线程渲染优化方法

李红波,罗　璇,吴　渝,刘昱晟
(重庆邮电大学 网络智能研究所,重庆 400065)

摘　要:针对汽车三维虚拟现实仿真系统在多核平台下的渲染效率和 CPU 利用率较低的这一问题,提出了一种基于 OpenMP 的多线程渲染优化方法。该方法采取了设置动态线程数量、策略调度以及渲染帧并行绘制等策略,对汽车三维虚拟现实仿真系统渲染过程中的初始化阶段、逻辑阶段以及渲染阶段进行并行优化。在多核平台上进行了实验,结果表明,该方法能有效地提高系统的渲染效率和 CPU 利用率,改善 CPU 的负载均衡。

关键词:并行计算;多核平台;多线程;渲染速率;优化;虚拟现实
中图分类号:TP391.9　　**文献标识码**:A　　**文章编号**:1673-825X(2014)01-0137-06

A multithreaded optimization rendering method for 3D virtual reality simulation system of vehicles

LI Hongbo, LUO Xuan, WU Yu, LIU Yusheng
(Institute of Web Intelligence, Chongqing University Posts and Telecommunications, Chongqing 400065, P. R. China)

Abstract: Since in current 3D virtual reality simulation system of vehicles, the rendering efficiency and CPU utilization under the multi-core platform are both low, a multi-thread rendering optimization method based on OpenMP is presented. In this method, initialization stage, logical stage and rendering stage of the system are optimized by use of strategies including setting the number of dynamics threads, adopting scheduling scheme and rendering parallelly of the rendering frame. The test on the multi-core platform shows that our method gains a high rendering speed, CPU utilization, and better CPU load balancing.

Key words: parallel computing; multi-core platform; multithreading; rendering rate; virtual reality

0　引　言

随着虚拟现实技术的发展,OGRE 等三维渲染引擎越来越广泛地应用在汽车仿真系统的研究和开发中[1]。在这些汽车仿真系统中,为了观察汽车各部件的运行状态,需要在同一系统界面中打开多个实时渲染的视窗。然而,目前的系统即便运行在多核计算机上,也仍使用单线程的串行执行,不能有效使用到所有 CPU 资源,渲染效率低,并造成多核 CPU 资源的浪费。

针对此类问题,Jeff Andrews[2] 创建了一个介于 API 和渲染插件之间的层,采用缓冲区加锁的方式来处理多线程化;陈学亮[3] 使用 Win32 线程库来更新队列、复制对象等多帧渲染的方法对 OGRE 渲染引擎进行多线程并行优化。以上 2 种方法能优化场景组织较为简单的实例渲染效率,但对于场景组织较为复杂的实例,其优化效果不明显。此外,赵建斌[4] 根据 Win32 线程库和 DirectX 的多线程支持,提出了一种渲染帧的线程级并行化方法。该方法对复杂场景的性能提升效果明显,但对逻辑简单、渲染资源少的简单场景,优化效果并不理想。李喆[5] 根据线程数等于 CPU 核数的原则,提出了一种设置线程数量的方法。该方法能提升三维场景的并行渲染效率,但却未提出相应的策略调度来保证负载平衡。

收稿日期:2013-01-07　**修订日期**:2013-09-09　**通讯作者**:李红波　lihongbo@cqupt.edu.cn
基金项目:"核高基"重大专项(2009ZX01038-002-002-2);新世纪优秀人才支持计划和科技部计划扶持项目(2009-593)
Foundation Items: The "CHB" Major Project (2009ZX01038-002-002-2); The Support Plan for New Century Excellent Talents and the Ministry of Science and Technology Plan(2009-593)

本文根据汽车虚拟仿真系统的功能需求,分析现有基于多核平台下并行渲染优化方法的不足,提出一种基于 OpenMP 的多线程并行优化方法。该方法分别对汽车仿真系统渲染过程中的初始化阶段、逻辑运算阶段以及渲染阶段进行并行优化,旨在提升系统渲染效率和 CPU 利用率。

1 总体优化策略和方法描述

本文工作是针对项目组在“核高基”重大专项支持下研发的汽车三维虚拟仿真系统。该系统基于 OGRE 渲染引擎开发,主要模拟不同汽车在不同环境、不同路面条件下的 ABS 功能、平顺性、稳定性测试,并真实反映制动过程中车体的运动状态。由于该系统的结构复杂,而目前基于多核平台的并行渲染系统还处于探索阶段[6],现有多线程并行优化方法不能全面地体现仿真系统的优化效果。

现有的多帧渲染并行优化方法[2-3]在渲染阶段并没有对渲染层面进行过多的并行处理,这导致渲染状态的频繁切换,对复杂场景无明显的优化。渲染帧的线程级并行优化方法[4]在逻辑帧的迭代运算处理时,并没运用策略调度和线程数量的控制,对于场景组织简单的实例,DirectX API 延迟线程的创建以及事件信号的发送和等待的时间开销占据绝大部分,使得优化后效果并不理想,甚至下降。基于核数的线程设置方法[5]得到的并非最优线程数,在并行优化时没有提出相应的策略调度来保证负载平衡。由于汽车三维仿真系统这类复杂的图形应用程序对渲染引擎各方面性能有较高要求,上述方法不能全面体现仿真系统渲染的优化效果。

汽车虚拟现实仿真系统运行过程中,场景与资源的加载、车体信息与视口的更新、汽车运行状态的渲染,分别对应引擎渲染过程中的初始化阶段、逻辑运算阶段和渲染阶段。根据系统运行过程以及渲染引擎绘制过程,本文的总体优化方案如图 1 所示。

主要的优化策略和方法包括:①针对在初始化阶段实体结点依赖性最小,无依赖函数之间不存在线程同步与数据竞争的特点,采用 OpenMP 中的 sections 方法,对无依赖函数并行处理,完成场景组织和资源加载的优化。同时,采用动态设置最优线程数量的方法来提升渲染效率;②针对车体各部分信息与视口更新时的大量迭代计算,在逻辑运算阶段,采用策略调度使迭代计算的线程达到负载均衡,以便获得更高的执行效率;③针对多帧渲染和渲染帧线程级并行优化对不同场景渲染优化片面性的缺陷,在渲染阶段采用渲染帧并行绘制的方法来保证对具有不同复杂度的场景的绘制效率。

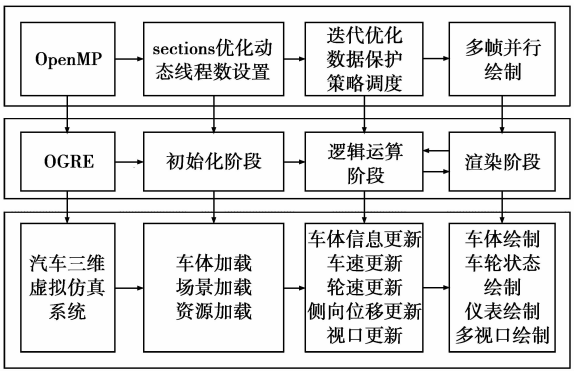


图 1 总体优化策略

Fig. 1 Overall optimization scheme

2 优化方法设计及实施

2.1 初始化阶段的并行优化

在汽车三维仿真系统中,通常使用加速比 $S_p(n)=t_s(n)/t_p(n)$ 评估并行性能,其中,$t_s(n)$ 为串行算法的运行时间;$t_p(n)$ 为并行算法的运行时间。该指标反映了并行性对运行时间的改进程度[7]。多线程并行优化时,并发执行的线程可能发生数据竞争,性能的提升取决于系统中并行任务数量的多少以及任务的规模变化情况。假设有锁保护的串行化时间为 1,可并行化部分在单核 CPU 上的运行时间为 t,CPU 的个数为 n,那么在 n 个对等任务同时运行情况下,总等待时间为 $1+2+\cdots+n=n\cdot(n-1)/2$,耗时最多的一个任务所用时间为 $n+t/n$,加速比为

$$S_p(n)=(t+1)/(n+t/n)=n(t+1)/(n^2+t) \tag{1}$$

从公式(1)看出,随着线程数量增加,若线程开销超过一定范围,会使系统性能下降。在设置线程数量时,需要根据系统能力、应用程序的并行需求选择合适的线程数量,达到性能最优。因此,在仿真系统首次运行之前,要计算出最优线程数。具体算法步骤描述如下。

步骤 1 定义 TheadFlag,表示是否得到最优线程数量;定义 nthreads,表示最优线程数;

步骤 2 若 TheadFlag = True,读取 nthreads 覆盖环境变量 OMP_NUM_THREADS 的值,转步骤 7;

步骤 3　若 TheadFlag = False,获取 CPU 核数 p,$p \times 2^n$ 为当前线程数量;

步骤 4　计算 $n = 0$ 时,模拟粒子势能计算[3,8]的时间 t_n;

步骤 5　$n = n + 1$ 计算此时 t_{n+1};

步骤 6　若 $t_{n+1} > t_n$,取 $p \times 2^n$ 为最优线程数量;否则,重复步骤 5;

步骤 7　利用 omp_set_num_threads(nthreads)函数自动设置线程数量。算法结束。

根据最优线程数量,以汽车虚拟仿真平台中初始化为例,利用 OpenMP[9]的 sections 命令定义一个区块,用 section 命令将此区块划分为不同的并行段。对场景和资源加载这类无依赖函数的并行优化如下。

```
#pragma omp parallel sections num_threads(nthreads){
    #pragma omp section   SenceLoad()
    #pragma omp section   CarLoad()
    #pragma omp section   ResouceLoad()
}
```

2.2　逻辑阶段的并行优化

系统渲染过程中,逻辑帧的生成将消耗大量的时间来完成大量的循环迭代运算。因此,逻辑帧的生成速度是影响渲染系统效率的重要因子。通过对循环迭代运算的并行优化,在保证帧之间相关性的前提下,对逻辑帧“并行”生成,则逻辑帧的生成速率会大大提高,可以提升 CPU 利用率并减少渲染帧的等待时间,提高渲染效率。在多线程并行优化设计时,线程的负载平衡对整体性能影响很大[10-12],对循环迭代做有效的调度策略来确保负载平衡,能保证多个处理器内核大部分时间里都保持工作状态。逻辑帧的生成过程中,每一帧的循环体大小以及循环迭代的规模并不确定,线程数定义为 p,在动态调度的实现过程中,对于循环迭代数为 I 的循环结构,第 i 次调度的任务块大小定义为 C_i,剩下的循环迭代数为 R_i。

$$R_0 = I, C_i = f(R_{i-1}, p), R_i = R_{i-1} - C_i \quad (2)$$

公式(2)中函数 f 中的参数依据具体的问题场景可能有所不同。此时的负载不均衡依赖于各个线程执行时间 $t_j(j = 1, \cdots, p)$ 之间的最大差距[13]。因此,在对系统渲染过程中的迭代循环进行优化时,要遵循如下的选取原则。

1)设置数据保护,避免与渲染线程的同步;

2)采用适当的线程调度策略,以保证线程负载平衡;

3)要尽量选择迭代次数较多、内部结构简单的循环,减少时间开销;

4)要尽量选取最外层循环,以便实现最大程度的多线程优化,提高优化效率;

系统渲染过程中耗时间较多的模块为 Rendersystem_Dire3D9,OgreMain[3],分别对其对应文件中符合要求的循环迭代做并行优化。以 Rendersystem_Dire3D9 模块内其中一个循环迭代为例,设置保护数据 mRenderTargets、线程数量 nthreads、采用 guided 调度策略进行并行优化如下。

```
#pragma omp parallel for private (mRenderTargets) schedule(guided)  num_threads(nthreads)
for(size_t x = 0;
x < OGRE_MAX_MULTIPLE_RENDER_TARGETS; ++x)
{
if(mRenderTargets[x] != NULL)
pSurf[x] =
mRenderTargets[x] -> getSurface(D3D9RenderSystem::getActiveD3D9Device());
}
```

对于其他计算量大的循环迭代根据具体情况设置好相应的数据保护以及调度策略。

2.3　渲染阶段的并行优化

在渲染过程中,场景管理器通过计算当前帧中所有可见实体对象,并将其按顺序排列,由此组成渲染队列。渲染队列是渲染阶段确保绘制流程和提高绘制效率的关键。在渲染队列中,根据传入的对象和自身的配置按照一定规则将可视实体划分为多个优先级,优先级高的实体先于优先级低的实体绘制,通过各个实体的优先级序列,确保正确的渲染流程。具有相同优先级的实体则依照渲染对象与摄像机的距离或者通路(pass),再次细分绘制顺序。将可视距离相同的实体和具有相同通路的物体放在一起进行绘制,可保证绘制的连续性,减少渲染状态的切换。场景管理器为渲染队列中的对象分配索引编号,根据索引编号的不同,将背景、场景和界面划分开,再对渲染队列中物体逐个进行渲染。

根据渲染队列的绘制特点,为有效解决多帧渲染导致的数据竞争和同步问题,本文对所有可见对象进行多线程同步并行渲染。在多帧渲染优化方案

的基础上,将渲染循环分成信息获取和执行绘制 2 个阶段。在信息获取阶段通过统计所有可见的静态几何体、场景实体以及通路的个数。若场景实体数大于最大通路数,将天空体、用户界面和场景实体放入不同的子线程中并行获取绘制信息。在执行绘制阶段,则通过主线程的调用,最后,执行所有通路相同线程的并行绘制;否则,按通路排序和可视距离直接绘制实体。这样可实现渲染帧的多线程同步执行,解决复杂场景和简单场景的优化效率。实现原理如图 2 所示。

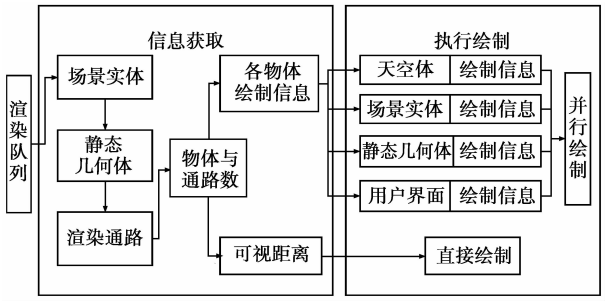


图 2 渲染阶段的并行优化方案

Fig. 2 Parallel optimization scheme of the rendering stage

具体算法描述如下。

步骤 1 统计当前帧渲染队列中所有可见的静态几何体、场景实体以及渲染通路个数 N_1, N_2, N_p;

步骤 2 若 $N_1 + N_2 < N_p$,根据可视距离依次对具有相同通路的物体直接执行绘制,转步骤 6;

步骤 3 若 $N_1 + N_2 > N_p$,创建子线程,采用 guided 策略调度,并行获取天空体、用户界面以及各场景实体的绘制信息;

步骤 4 根据通路个数 N_p 设置数据保护;

步骤 5 通过个物体的绘制信息依次对具有相同通路的物体子线程执行并行绘制;

步骤 6 当前帧绘制结束,清空渲染队列。算法结束。

3 实验结果与讨论

多线程图形应用程序性能的标准主要有 FPS (frames per second),CPU 利用率和均衡负载,为了验证本文优化方法,实验环境为 IntelCore I5 2320 双核处理器,4 GByte 内存,NVIDIA GeForce GTS 560 图形处理器。采用超线程,可并行处理 4 个线程。

3.1 OGRE 引擎优化结果

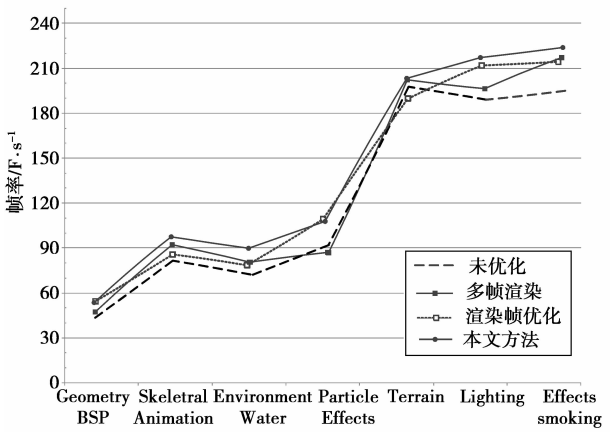
OGRE 版本为 1.7.2,采用 2.1,2.2 和 2.3 中并行方法对 OGRE 渲染系统以及标准测试程序进行并行优化,标准测试程序使用 debug 调试。优化前后的结果如表 1 所示。

表 1 OGRE 渲染引擎性能对比

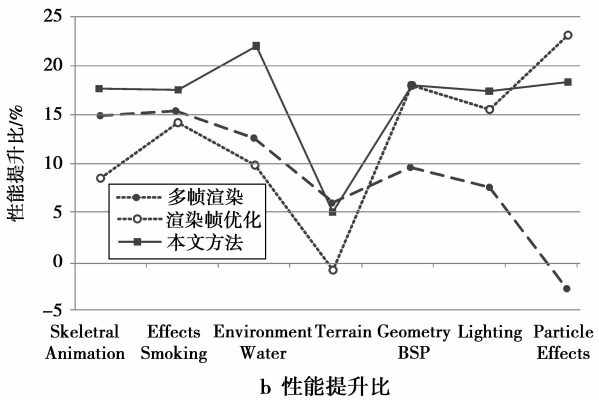
Tab. 1 Performance contrast of OGRE rending engine

标准测试程序	优化前 FPS	优化后 FPS	提高比/%
Skeletral Animation	84.850 4	99.339 2	17.1
Effects_Smoking	195.596 7	228.946 3	17
Environment_Water	76.083 9	92.182 6	21.2
Terrain	198.706 3	208.840 3	5.1
Geometry_BSP	47.442 5	54.772 7	15.4
Lighting	190.502 8	222.876	16.9
Particle_Effects	94.124 7	110.878 9	17.8

从表 1 对比数据可得,并行优化能大大提升测试程序的渲染效率,大部分帧率提升 15% 以上。粒子系统实例和 BSP 场景管理实例这些较复杂场景,性能提升也为 17.8% 和 15.4%。对于逻辑运算较为简单,渲染几何资源较少的地形管理实例,优化后性能提升 5.1%。性能提升曲线如图 3 所示。



a 帧率曲线



b 性能提升比

图 3 优化后性能

Fig. 3 Performance after optimization

图 3a 中是多帧渲染优化、渲染帧优化和本文三种优化方法下渲染效率以及原始 OGRE 渲染效率的比较。由图 3a 可知,本文优化方法能有效提高 OGRE 渲染效率。对于某些测试程序,渲染帧频要比多帧渲染优化、渲染帧优化方法更高,完全能够满足高质量连续画面渲染的要求。

图 3b 中是多帧渲染优化、渲染帧优化和本文 3 种优化方法性能提升曲线。本文方法在大多数测试程序的性能提升优于这 2 种方法。虽然在粒子系统实例中,性能提升比稍逊于渲染帧优化方法,然而,在对于逻辑运算简单、渲染几何资源少的地形管理实例测试时,性能并未下降。从总体性能提升来看,本文优化方法效果是理想的。

3.2　汽车虚拟现实仿真系统优化结果

根据本系统对于汽车侧滑、翻滚、甩尾等动作体现力较强的需求,要同时对车体的正面、侧面、各轮子进行观察,记录其运动状态以及参数。由此,采用多视口窗体,将每个视口对应车体各个需要观察运动状态的部分,每个部分在一个独立的线程上完成。利用 OGRE 实现的多视口如图 4 所示,渲染效率(平均 FPS)和 CPU 利用率如表 2 所示。

表 2　汽车虚拟仿真平台性能优化对比

Tab. 2　Performance contrast of 3D virtual reality simulation platform

性能参数	优化前	优化后	提高比/%
渲染效率	49.6	60.2	21.37
CPU 利用率	43%	58%	34.8

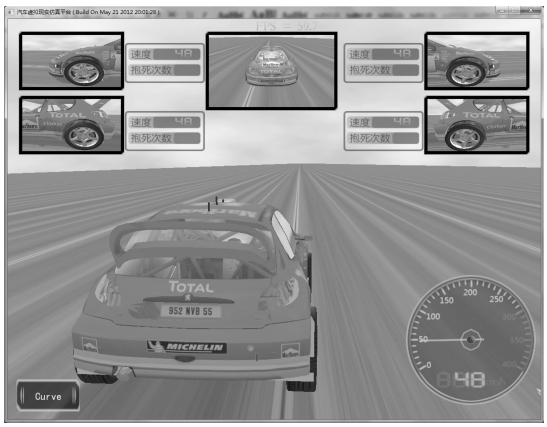


图 4　汽车虚拟仿真平台中多视口

Fig. 4　Multiple views in 3D virtual reality simulation platform

从表 2 中看出,优化前后渲染效率提升了 21%。优化前 CPU 的利用率为 43%,优化后为 58%,提升了约 34%。渲染效率和 CPU 的利用率都有了较理想的提升。图 5 是 CPU 使用情况对比,通过 CPU 使用记录对比可以看出,根据合适的调度策略优化,每个处理核心都得到了充分的调度,负载更加均衡。

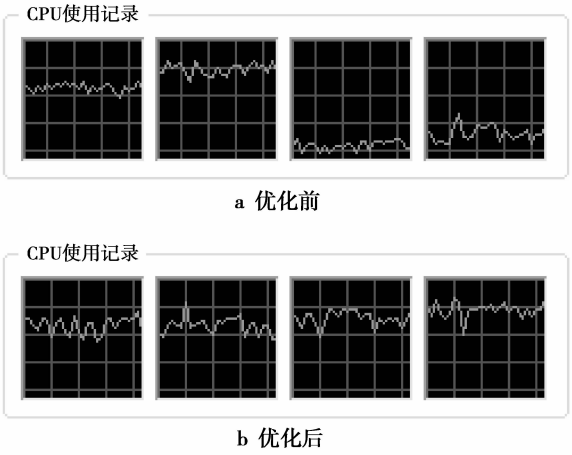


图 5　汽车虚拟仿真系统优化前后 CPU 使用情况对比

Fig. 5　Contrast of CPU usage in 3D virtual reality simulation platform

4　结束语

多线程并行渲染的方式能大大提高图形应用程序的性能。本文采用了一种基于 OpenMP 多线程并行渲染优化方案,对基于 OGRE 渲染引擎的汽车三维虚拟仿真系统进行并行渲染优化。该方案能有效提高 OGRE 渲染引擎的性能、汽车虚拟仿真系统的渲染效率和 CPU 利用率,对于线程级并行方法在图形渲染系统中的研究具有一定参考价值。

参考文献:

[1]　苏永生,赵冬斌. 基于 OGRE 的车辆自适应巡航控制三维仿真[J]. 交通运输系统工程与信息,2012,12(2):47-52.
SU Yongsheng, ZHAO Dongbin. 3D Simulation of Adaptive Cruise Control Based on OGRE[J]. Journal of Transportation Systems Engineering and Information Technology, 2012, 12(2): 47-52.

[2]　JEFF Andrews. Threading the OGRE 3D Render System [EB/OL]. (2006-09-06) [2012-12-10]. http://www.intel.com/cd/ids/developer/asmona/eng/dc/games/331359.htm.

[3]　陈学亮. 基于多核平台优化的 OGRE 3D 渲染引擎[D]. 杭州:浙江大学,2007.
CHEN Xueliang. An Optimized OGRE 3D Render Engine Based by Multi-Core Platform [D]. Hang Zhou:

Zhejiang University, 2007.

[4] 赵建斌,李灵巧,杨辉华. 线程级并行计算在图形渲染引擎中的研究[J]. 计算机工程与设计,2011,32(12):41-43.
ZHAO Jianbin, LI Lingqiao, YANG Huihua. Research on graphic rendering engine with thread-level parallelism method [J]. Computer Engineering and Design, 2011, 32(12):41-43.

[5] 李喆,郑晓薇. 多核并行技术在三维场景加载中的应用[J]. 计算机工程,2011,37(6):245-246.
LI Zhe, ZHENG Xiaowei. Application of Multi-core Parallel Technology in 3D Scene Loading[J]. Computer Engineering, 2011, 37(6):245-246.

[6] 余江峰,陈景广,程亮,等. 三维地形场景并行渲染技术进展[J]. 武汉大学学报:信息科学版,2012,37(4):463-466.
SHE Jiangfeng, CHEN Jingguang, CHENG Liang, et al. Review on 3D Terrain Parallel Rendering[J]. Geomatics and Information Science of Wuhan University. 2012, 37(4): 463-466.

[7] 陈国良,孙广中,徐云,等. 并行算法研究方法学[J]. 计算机学报,2008,31(9):1494-1501.
CHEN Guoliang, SUN Guangzhong, XU Yun, et al. Methodology of Research on Parallel Algorithms[J]. Chinese Journal of Computers, 2008,31(9):1494-1501.

[8] AYGUADE E, BLAINEY B, DURAN A, et al. OpenMP Shared Memory Parallel Programming : Is the Schedule Clause Really Necessary in OpenMP[C]// International Workshop on OpenMP Applications and Tools, WOMPAT 2003 Toronto. Canada: Lecture Notes in Computer Science, 2003, 147-159.

[9] 蔡佳佳,李名世,郑锋. 多核微机基于 OpenMP 的并行计算[J]. 计算机技术与发展, 2007,17 (10):87-91.
CAI Jiajia, LI Mingshi, ZHENG Feng. OpenMP-Based Parallel Computation on Multi-Core PC [J]. Computer Technology and Development, 2007,17 (10):87-91.

[10] OLIVIER S L,PORTERFIELD A K, WHEELER K B, et al. OpenMP Task Scheduling Strategies for Multicore NUMA Systems[J]. International Journal of High Performance Computing Applications. 2012,26(3),110-124.

[11] 汪伟,范秀敏,武殿梁. 虚拟现实应用中的并行渲染技术[J]. 计算机工程,2009,35(3):282-285.
WANG Wei, FAN Xiumin, WU Dianliang. Parallel Rendering Technology in Virtual Reality Applications [J]. Computer Engineering, 2009,35(3):282-285.

[12] 赖建新, 胡长军, 赵宇迪,等. OpenMP 任务调度开销及负载均衡分析[J]. 计算机工程, 2006,32(18):58-60.
LAI Jianxin, HU Changjun, ZHAO Yudi,et al. Analysis of Task Schedule Overhead and Load Balance in OpenMP [J]. Computer Engineering,2006,32(18):58-60.

[13] 任小西,唐玲,李仁发. OpenMP 多线程负载均衡调度策略研究与实现[J]. 计算机科学,2010,37(11):148-151.
REN Xiaoxi, TANG Ling, LI Renfa. Study and Implementation of OpenMP Multi-thread Load Balance Scheduling Scheme[J]. Computer Science,2010,37(11):148-151.

作者简介:

李红波(1970-),男,甘肃兰州人,高级工程师,硕士,主要研究方向为数字媒体、机器视觉、虚拟现实。E-mail: lihongbo@ cqupt. edu. cn。

罗 璇(1987-),男,湖北黄陂人,硕士研究生,主要研究方向为数字媒体,虚拟现实。E-mail:hongyue1987m@ yahoo. com. cn。

吴 渝(1970-),女,重庆人,教授,博士生导师,主要研究方向为网络智能、数字媒体、数据挖掘。E-mail: wuyu@ cqupt. edu. cn。

刘昱晟(1987-),男,重庆人,硕士研究生,主要研究方向为数字媒体、虚拟现实。E-mail: liuyusheng. cn@ gmail. com。

(**编辑:**刘 勇)