面向 CUDA 的 GPU 虚拟化研究及其应用*

马业+, 袁家斌, 吕相文

(南京航空航天大学 计算机科学与技术学院,江苏 南京 210016)

Research and Application of CUDA-based GPU Virtualization*

MA Ye, YUAN Jia-Bin, LV Xiang-Wen

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

+ Corresponding author: E-mail: maye1766@126.com, http://www.nuaa.edu.cn/nuaanew

Abstract: Considering the powerful parallel computing ability of GPU and the isolation characteristic of virtualization,the GPU virtualization technology that support dynamic scheduling and multi-user concurrent is presented. CUDA manage end utilize centralized, flexible mechanism to manage the GPU resources. Simultaneously,CUDA manage end manage GPU internal tasks. We set a value of integrated load evaluation to achieve load balance.We design a distributed heterogeneous system to simulate the application scenarios.At the same time,We verify the feasibility and efficiency of GPU virtualization that applied in the designed system.

Key words: GPU virtualization; CUDA; distributed system

摘 要: 为高效、充分的利用 GPU，针对 GPU 的特点，结合虚拟化环境的安全、隔离等特性，借鉴已有的虚拟化技术，提出了一种可动态调度、支持多用户并发的 GPU 虚拟化技术。CUDA 管理端采用集中、灵活的机制对 GPU 资源进行统一管理，对 GPU 内部任务统一调度。GPU 内部任务调度通过设置综合负载评价值实现负载平衡，避免了资源的利用不充分。设计了一个面向科学计算的分布式异构系统来模拟 GPU 虚拟化的应用场景，并通过在设计的系统上进行大规模矩阵运算实验，验证了 GPU 虚拟化技术应用在计算系统中的可行性和高效性。

关键词: GPU 虚拟化；CUDA；分布式系统

中图法分类号: TP393 **文献标识码**: A

* Supported by the National High-Tech Research and Development Plan of China under Grant No.2009AA044601(国家高技术研究发展计划(863)); the National Natural Science Foundation of China under Grant No.61139002(国家自然科学基金重点项目); A Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions (江苏高校优势学科建设工程资助项目); NUAA Research Funding.NO.NS2010230(南京航空航天大学基本科研业务费专项科研项目资助); Research Fundation of Graduate Innovation Center of NUAA.NO.kfjj20110128(南京航空航天大学研究生创新基地开发基金)

作者简介: **马业**(1988－),男,硕士,主要研究领域为虚拟化，高性能计算;**袁家斌**(1968－),男,江苏兴化人,博士,教授,主要研究领域为信息安全,高性能计算,量子密码;吕相文(1987－),男,博士,主要研究领域为云计算，量子密码.

随着一些支持 GPGPU 计算的技术（例如 CUDA[1]）的出现，GPGPU 的应用也越来越广泛，使用 GPU 集群来进行以海量数据计算为基础的科学研究的例子不胜枚举。但是，一方面 GPU 的功耗较大，如果每个节点都配备 GPU，则可能大大增加集群的功耗；另一方面，GPU 的并行计算能力强大，而在大部分运算中，GPU 作为协处理器，仅仅加速代码中的并行部分，使得 GPU 的使用率不高。为了解决以上问题，出现了 vCUDA[2]、Gvim[3]、gVirtus[4]等 GPU 虚拟化方案。当前，结合 GPU 虚拟化技术构建 GPU 虚拟集群，使用云计算的模式来提供针对科学计算的服务为很多科学应用提供了新的可能。云计算的便捷性、屏蔽资源异构性及其资源易获取性保证了应用性能不降低的同时，大幅度降低了使用成本。

本文借鉴已有的虚拟化技术，将 GPU 引入虚拟机，使得虚拟机可以使用 GPU 强大的并行能力加速自身计算任务。在 GPU 虚拟化技术中引入负载平衡技术以提高 GPU 资源使用率，同时设计一个分布式应用系统模拟 GPU 虚拟化的应用场景，并通过实验验证 GPU 虚拟化的可行性和高效性。

1 相关工作

1.1 通用并行计算架构CUDA

当前计算正在从 CPU 处理向 CPU 与 GPU 协同处理的方向发展，为了实现这一新型计算模式，NVIDIA 提出了 CUDA 并行计算架构。CUDA(Compute Unified Device Architecture)是一个新的基础架构，这个架构可以使用 GPU 来解决商业、工业以及科学方面的复杂计算问题。

CUDA 软件栈包含多个层：设备驱动程序、应用程序编程接口（API）及其运行时、两个较高级别的通用数学库，即 CUFFT 和 CUBLAS[5]。运行时库可分割为宿主组件、设备组件和通用组件。宿主组件运行在宿主上，提供函数来通过宿主控制和访问一个或多个计算设备。设备组件运行在设备上，提供特定于设备的函数。通用组件提供内置向量类型和 C 标准库的一个子集，宿主和设备代码都将支持此子集。宿主运行时组件包括两个 API：CUDA 驱动程序 API，CUDA 运行时 API。CUDA 运行时 API 是在 CUDA 驱动程序 API 的基础之上实现的。这两个 API 是互斥的，因此一个应用程序仅能使用其中之一。CUDA 运行时提供了隐式初始化、上下文管理和模块管理，从而简化了设备代码管理。NVCC 生成的 C 宿主代码基于 CUDA 运行时，因此链接到此代码的应用程序必须使用 CUDA 运行时 API。对于程序员来说，CUDA 的执行模型就是一系列并行执行的线程的集合。

1.2 GPU虚拟化

在图形处理方面，VMWARE[6]主持开发的一个特定的 GPU 虚拟化架构 VMware's Virtual GPU。这个方案有些类似于设备仿真的方法，但是它包含了 API 远程处理的特点。还有一个是基于 OPENGL 的解决方案 VMGL[7]，它是一种比较通用的方案，不依赖于特定的虚拟化平台和 GPU。

在通用计算方面，主要是基于 CUDA 的解决方案，例如 vCUDA、Gvim、gVirtus 等等。vCUDA 采用了 XML-RPC 的方式来处理客户端和服务器端的通信，这使得开发难度大大降低，但是也带来了不小的通信开销。Gvim 则依赖于特定的 XEN 虚拟化平台，但是它利用 XEN 中特定的通信方式例如 XENLOOP 等，降低了通信开销。gVirtus 方案则是一个较通用的解决方案，结合了 vCUDA 的平台无关性和 Gvim 的低通信开销的特点。VCL[8]则是一个类似的基于 OPENCL 的解决方案。

2 面向 CUDA 的 GPU 虚拟化

2.1 GPU虚拟化

GPU 虚拟化解决了传统的虚拟化方案在新的编程接口没有提供虚拟化硬件的问题，通常使用基于动态库拦截的方案。本文参照 C/S 模式(如图 1)，利用开源代码实现了三个用户层模块：位于虚拟机的 CUDA 客户端、位于 VMM[9](Virtual Machine Monitor)或者特权虚拟机的 CUDA 服务端、CUDA 管理端。

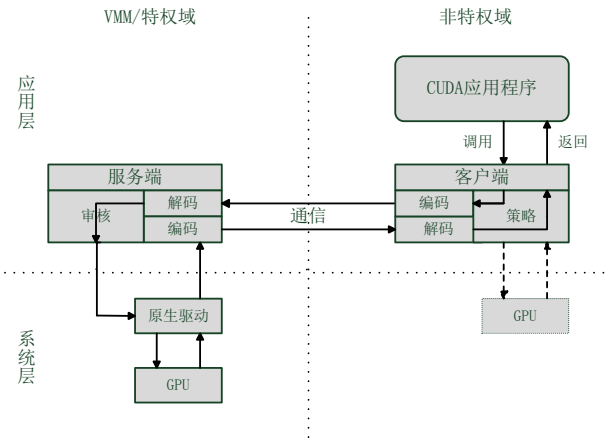


图 1 GPU 虚拟化框架

2.1.1 CUDA 客户端

客户端组件面向应用程序，其作用包括：1)拦截应用程序中 CUDA 调用；2)选择通信策略，为虚拟化提高更高层语义的支持；3)对调用和参数封装，编码；4)对服务端返回的数据进行解码，并返回给应用。

其中通信策略很广泛，可以利用各虚拟化平台自带的高效通信方式，例如 Xen 下的 XenRPC[10]、VMware 下的 VMCI[11]，或者使用通用的方式，例如 socket、远程过程调用(remote procedure call,RPC)，或者直接引入需要特殊设备的 InfiniBand[12]架构。其中 RPC 是最快速、便捷的开发方式。本文采用 socket 的方式进行通信，传输数据为字节级，传输数据可自定义，并在需要的场合对数据进行加密，数据安全性较强,适用于各种不同的虚拟化平台。

2.1.2 CUDA 服务端

服务端组件面向真实物理硬件，其作用包括：1)接收客户端的数据包，并解析出调用和参数；2)对调用和参数进行审核；3)利用物理 GPU 运算审核通过的调用；4)将结果编码，并返回给客户端；5)对计算系统中支持 CUDA 的 GPU 进行管理。

服务端应对客户端的请求时，为每个用户分配独立的服务线程，实现了多用户并发，即同一个 GPU 上可以安排不同的用户服务线程，使得单 GPU 多用户共享。

2.1.3 CUDA 管理端

在实现 CUDA 编程接口虚拟化的基础上，将 GPU 强大的计算能力和计算资源在更高的逻辑层次上进行隔离、划分、调度。管理端对 GPU 资源进行统一管理，采用集中、灵活的机制，实现：1)动态调度：当用户所占资源空闲时间超过一定阈值或者任务结束时，管理端回收该资源，当该用户再次发布计算任务时，重新为其任务分配 GPU 资源；2)负载平衡：当局部计算压力过大时，调整计算负载，通过动态调度时选择合适的 GPU 资源来分散计算负载；3)故障恢复：当出现故障时，将任务转移到新的可用 GPU 资源上。

2.2 GPU内部任务调度

GPU 内部任务调度由 3.1 节中管理端完成。利用 GPU 虚拟化技术，使得多个虚拟计算节点可以共享 GPU 物理资源。由 GPU 虚拟化中服务器端运行时，向管理端注册 GPU 物理资源。由文献[13]可知，GPU 运算的速度并非取决于 block 数量或者 thread 数量，而是与 SM 数量密切相关，并且运算效率与 GPU 本身的内核数和时钟频率乘积大概成正比。并且 GPU 全局内存的大小对计算效率的影响也非常大，如果计算规模大于 GPU 全局内存规模，计算不能在 GPU 上一次完成，会引入额外的通信开销。故注册信息中主要包含处理核心的数量和时钟频率以及全局内存。管理端针对注册信息调度 GPU 任务。本文设置一个综合负载评价值(I)：$I=\frac{N}{\alpha*P*R+\beta*G}$，其中 N 为当前 GPU 上的任务数，P 表示处理核心的数目，R 表示处理核心的时钟频率，G

为全局内存。α为处理能力的影响因子，β为全局内存的影响因子，α、β为任意指定的0-1之间的常数。α、β的设置取决于对负载的属性，如负载对处理能力要求高，则可以适当提高α的值，如对负载内存需求较高，则可适当提高β的值，使得评价值更加贴近真实情况。如图5为GPU内部任务调度流程。

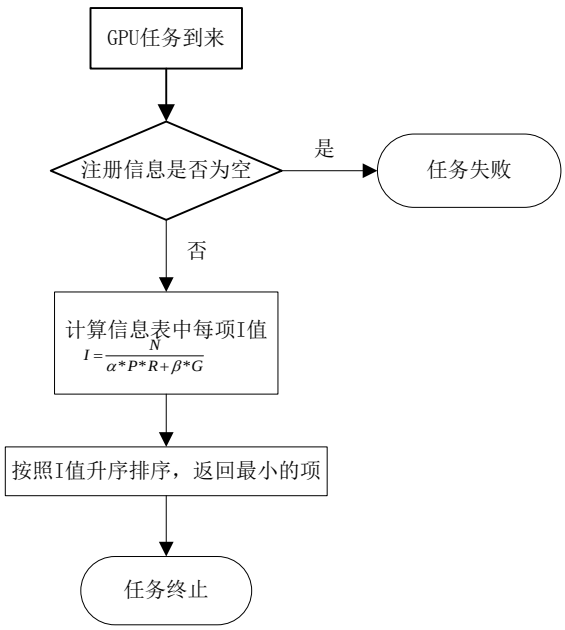


图2 GPU内部任务调度流程

对于服务端存在多个GPU物理硬件的情况，当捕获到cudaSetDevice()函数时，服务端利用和上述一样的机制，结合I，将当前最优的设备ID返回给客户端，这样保证了单个服务端多GPU间的负载平衡，最大化的利用GPU的并行处理能力。

3 GPU虚拟化的系统应用

3.1 系统实现目标

本文针对科学计算的问题，应用GPU虚拟化技术，设计了一个任务级的分布式异构科学计算系统，实现了基于计算资源特征的任务分配和调度，进而实现了高效的、透明的、并行的计算资源利用机制，有效的保证了计算性能，大幅度提高了易用性，同时降低了使用成本。系统主要实现了如下目标：1)采取了灵活的调度管理，把任务合理的划分并分配到有限的计算资源上；2)通过设置配置服务器，有效的避免了单点故障的问题；3)具有良好的扩展性，能够动态的加入计算资源；4)建立容错机制，并尽量减少通信量；5)实现CPU和GPU的协作，充分利用可用的计算资源。

3.2 系统总体架构

系统主要由分发服务器、配置服务器、调度服务器、计算资源服务器构成。其中计算资源服务器由GPU计算资源服务器、CPU计算资源服务器构成。架构如图3所示：

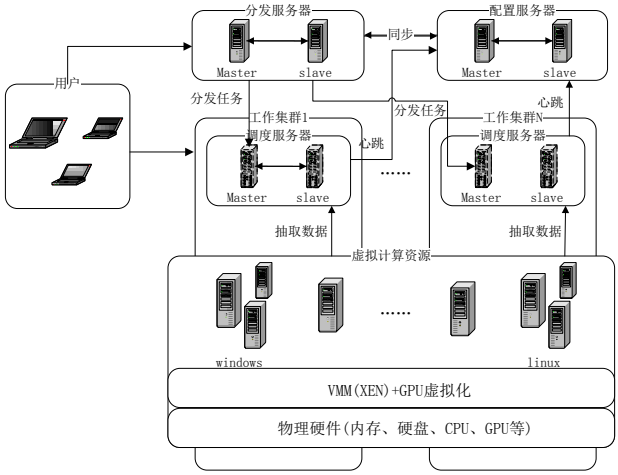


图 3 系统整体架构

分发服务器与用户直接交互，对外提供服务，并负责对任务进行分发，同时在任务级评估各个计算集群的运算效率。配置服务器监听已加入系统的计算集群调度服务器，维护一张索引表，每当有计算集群状态发生变化(例如新加入集群或者有调度服务器不可用)时，配置服务器会根据当前可用的集群重新生成索引表。集群中调度服务器对分发的任务进行划分，根据调度策略选择最优的方案，并监测集群中的计算节点。计算资源服务器由具有 GPU 加速的计算节点和普通计算节点组成。针对具有 GPU 硬件的计算节点，通过 GPU 虚拟化技术，将 GPU 虚拟化成一个共享资源。采用虚拟化技术，使得在一个物理机器中多个系统上运行的不同任务相互隔离，既提高了安全性，又提高了计算资源的利用率。

3.3 系统工作流程

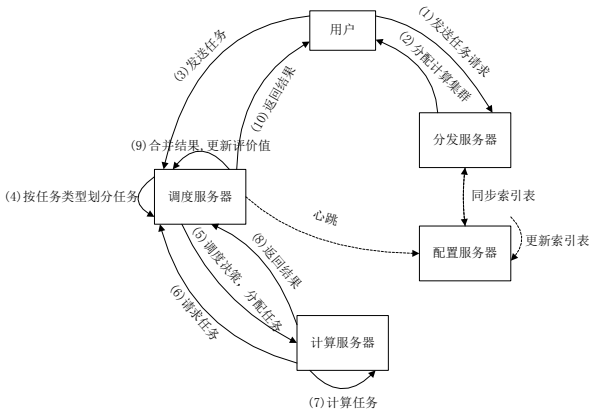


图 4 系统工作流程

工作流程如图 4 所示。其中虚线表示定时任务，计算节点和配置服务器间设置心跳，以便及时地更新索引表。分发服务器与配置服务器间定时同步索引表，以便让新加入的集群及时地提供计算资源，或者不可用的集群上的任务尽快重新调度到可用节点上计算。其中图中步骤 7 涉及的 GPU 虚拟化工作流程见图 5，其中虚线为 GPU 注册部分流程。

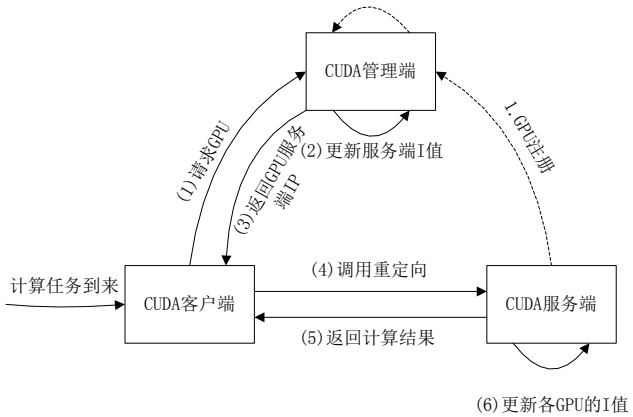


图 5 GPU 虚拟化部分工作流程

3.4 任务调度管理

系统适应多种计算的业务，简化调度管理的模型，根据系统的数据流动顺序设置了三个调度层次：

（1）以计算集群为划分基础的粗粒度任务调度。分发服务器掌握着各个集群的当前负载，了解各个集群的计算业务、总体计算能力和目前任务情况。计算任务从外部网络的用户到达时，先对其进行初步的筛选，把它发送到对应的支持该业务的计算集群里。如果对应的集群有多个，再综合集群的性能和当前负载确定计算任务的目的集群。

（2）以计算集群中的计算节点为基础的细粒度任务调度。调度者与被调度者之间不需要进行全部计算数据的传输，两者已经预先协商好所要计算的数据的总体范围，每一个任务片只是这个总体范围的一部分，两者每次只要传送该部分的起点和终点。

这一层次的调度有以下特点：

调度者被动式调度。调度者并不向被调度者发送相关信息，计算的信息存放在调度者上，组织结构采用集中式模型，资源推送采用抽取模型，被调度者主动连接调度者，进行提交信息和索取任务的操作。

小规模操作任务的划分。根据 GPU 并行计算和 CPU 计算的特点，计算任务被最小限度地划分，保证任务规模足够小，各个任务之间相互无关，某一任务出错时，不至于产生较大范围的消极影响。

实时信息反馈的调度。被调度者每次不仅提交任务完成的进度和结果，还把机器的性能和任务信息一并发送，在机器性能和任务负载实时反馈之下，调度者智能地实现负载均衡的设置。

（3）GPU 内部基于线程的任务调度。这一层次的调度由由 GPU 虚拟化中的管理端完成。单任务时，每一个线程都是采用相同的算法，计算过程已经固定，不同的是每一个计算过程的输入数据不一样，各个硬件线程宏观上同时开始执行任务，由于输入数据的差异，输出结果不相同，计算所用的时间也不尽相同。多任务时，硬件线程采用不同的算法，输入数据也不相同，计算所需的时间会存在差异。

3.5 可扩展性设计

可扩展性[12]其本身是一个多方面的概念集合，其中两个重要的考量是机器规模的可扩展性和问题规模的可扩展性。在机器规模的可扩展性上，本文使用了三个层次的调度，在每个层次都可以动态的添加软硬件资源，在最顶层的粗粒度调度上，动态的加入集群，分发服务器通过索引表向集群分发任务，并通过新加入集群完成任务的情况更新索引表。计算集群中新加入计算节点，会自动向调度服务器索取任务，通过这样的方式，调度服务器可以获取新加入计算节点的计算状态，通过设定的策略进行调度任务。从时间分布上看，多个计算节点的工作呈流水线方式(如图 6)，t1 表示计算节点和调度服务器交互的时间，t2 表示计算节点执行任务的时间，t1+t2 近似为计算节点的一个工作周期。每个计算节点的 t1 时间都是与调度服务器不断接受计算节点返回计算结果和分发任务的过程，所有的 t1 时间之后就是调度服务器工作时间。每个计算节点的工作周期中只有 t1 时间与调度服务器交互，其他时间均为单独计算时间。当计算节点完成 t1 时间的交互后，其他

名称	数量	硬件环境描述	软件环境描述
分发服务器	1	Intel Core i5(3.33GHZ)、4GB 内存	Centos 5.5
配置服务器	1	Intel Core i5(3.33GHZ)、4GB 内存	Centos 5.5
调度服务器	1	Intel Core i5(3.33GHZ)、4GB 内存	Centos 5.5、Cuda 3.2.16
计算资源服务器	20	（虚拟机环境）vCPU*2、2GB 内存、Tesla C2050	Centos 5.5、Cuda 3.2.16 Xen 4.0.1

表 1 系统软硬件环境

的计算节点就可以与调度服务器交互。

假设如图 6 所示只有 4 个计算节点，每个计算节点只需要与调度服务器交互一次就完成对应计算任务，那么完成所有任务的时间是 4*t1+t2，经过实验测试，t1 远小于 t2，可以忽略 t1 的影响，实际完成时间为 t2，如果只有一个计算节点，完成时间为 4*（t1+t2），约为 4*t2。由此可知，随着客户端程序数量的线性增加，这个系统的计算能力是线性增加的。

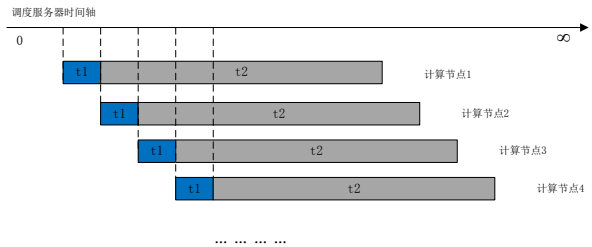
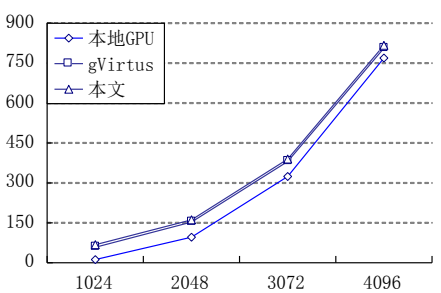


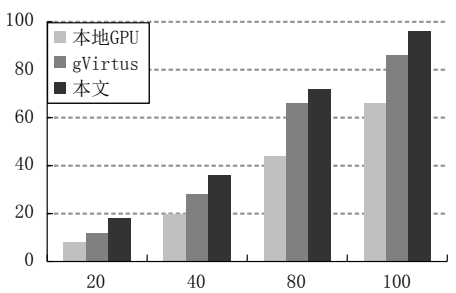
图 6 系统内工作流水线图

4 性能测试与分析

本文选取科学计算中较为常见的典型应用进行分析，从矩阵乘法方面对于系统的性能进行测试，并对结果进行分析。系统的体系结构由:分发服务器、配置服务器、调度服务器、计算资源服务器组成。系统基于软、硬件环境如表 1 所示。其中计算节点使用虚拟机，宿主物理机为两台服务器，一台配置为 Intel Xenon E7-4830、Tesla C2050*4、48GB 内存，另一台为 Intel Core i5-2300、Tesla C2050*2、16GB 内存。对于 GPU 虚拟化组件中的特权域，目前并不存在适用于 XEN 下 DOMAIN 0 中的显卡驱动，本文通过在 XEN 中非特权虚拟机中使用 pci pass through 技术，使得该非特权域获得物理 GPU 的访问权，并在此非特权域部署 CUDA 服务端组件，使其充当特权域的角色。实验中使用的矩阵乘法实现为 CUDA GPUCOMPUTING SDK 3.2.16 中简单矩阵乘法。本文设置 α 为 0.5， β 为 0.5。



（a）



(b)

图 8 (a)单任务下任务时间效率比较，X 轴为矩阵阶数，Y 轴为计算时间，单位为 ms
(b)多任务完成数比较，X 轴为任务个数，Y 轴为完成任务个数

针对单个任务利用 GPU 虚拟化的效率对比，由图 8(a)可以看出，本地 GPU、gVirtus 与本文，在任务规模变大时，本文的效率慢慢接近于本地 GPU 和 gVirtus。因为本文比本地 GPU 的方式增加了 CUDA 服务端与 CUDA 客户端之间的通信开销，比 gVirtus 增加了调度开销，在计算规模变大时，计算时间同时增加，则开销在总时间的比例越来越少，可忽略不计。同时本文的方式支持多用户并发、负载均衡，在多 GPU 和多任务的情况下明显优于另外两种方式。由图 8(b)可知，在给定时间内，本文完成任务数明显大于另外两种方式。

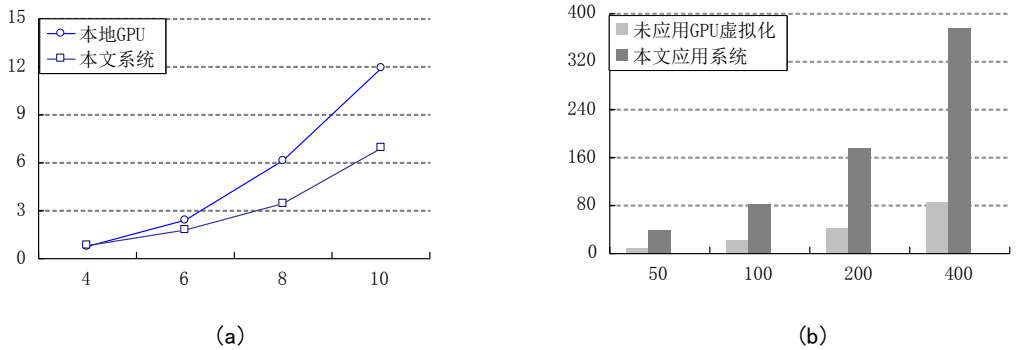


图 9 (a)单任务下应用系统完成任务时间效率比较，X 轴为矩阵阶数，单位为千，Y 轴为计算时间，单位为 s
(b)多任务下应用系统任务完成数比较

针对单个大规模矩阵相乘运算，由图 9(a)可以看出应用了 GPU 虚拟化技术的系统因为任务划分以及多线程并发，在问题规模扩大时运算时间呈现线性增加的趋势，而在本地 GPU 上呈指数倍数增加。针对多个矩阵运算任务，由图 9(b)可以看出，在给定时间内，本文应用系统的性能具有明显优势。应用了 GPU 虚拟化技术，同时设置了三层调度结构，可以将计算资源充分利用，并且针对矩阵运算这种可分割任务，系统设置的任务划分可以将任务运行时间有效减少。

5 结束语

结合分布式系统思想和可扩展性设计的原则，本文设计了一个面向大规模科学计算的分布式系统模拟 GPU 虚拟化技术的应用场景。GPU 虚拟化使 GPU 可以在多计算节点间共享，有效地通过 GPU 加速提高大多数类型的计算任务。同时利用 GPU 虚拟化提高了 GPU 使用率和计算任务的计算效率。此外，仍存在一些问题有待进一步该进。例如 GPU 虚拟化中的容错机制不够完善，GPU 虚拟化的效率和本地 GPU 之间仍有差距。相关的改进与测试将在后续工作中展开，并对容错机制进一步深入研究，最终希望将 GPU 虚拟化技术可靠性更高，应用系统更完善，能够以 SaaS[13](Software as a service)的模式提供服务。

致谢　感谢云计算组各位同学为本文的研究做出的贡献。

References:

[1] CUDA: Compute Unified Device Architecture［EB/OL］.http://www.nvidia.com/object/cuda_home_new.html. 2012.

[2] Lin Shi, Hao Chen, Jianhua Sun,et al.vCUDA: GPU-Accelerated High-Performance Computing in Virtual Machines[J]. Computers, IEEE Transactions on , vol.61, no.6, pp.804-816, June 2012 .

[3] Vishakha Gupta, Ada Gavrilovska, Karsten Schwan, Harshvardhan Kharche, Niraj Tolia et.al.GViM: GPU-accelerated Virtual Machines[J].3rd Workshop on System-level Virtualization for High Performance Computing (HPCVirt), in conjunction with EuroSys 2009, Nuremberg,Germany, Mar.

[4] Raffaele Montella, Giuseppe Coviello, Giulio Giunta, et al.A GPU Accelerated High Performance Cloud Computing Infrastructure for Grid Computing Based Virtual Environmental Laboratory[J]. Advances in Grid Computing ,2011:121-146

[5] Zhang Shu,Chu Yan-Li,et al.CUDA-enabled GPU high performance computing[M]. Beijing:China WaterPower Press,2009.

[6] Vmware[EB/OL].http://www.vmware.com.2012.

[7] H.Andres Lagar-Cavilla, Niraj Tolia,et al.VMM-independent graphics acceleration[J].Proceedings of the 3rd international conference on Virtual execution environments.June 2007:33 - 43.

[8] Amnon Barak,Amnon Shiloh.The Virtual OpenCL (VCL) Cluster Platform[EB/OL].http://www.mosix.org.2012.

[9] P Barham, B Dragovic, K Fraser,et al.Xen and the art of virtualization[J].SOSP '03 Proceedings of the nineteenth ACM symposium on Operating systems principles.2003:164-177.

[10] Chen Hao,Peng Cuifen,Sun Jianhua,et al.XenRPC:Design and Implementation of Security VM Remote Procedure Call[J].Journal of Computer Research and Development.49(5):996-1004,2012.

[11] VMCI Overview[EB/OL].http://pubs.vmware.com/vmci-sdk/VMCI_intro.html.2012.

[12] HAO Shui-xia,ZENG Guo-sun,TAN Yi-ming.Scalability Analysis of Heterogeneous Computing Based on Computation Task and Architecture to Match[J].38(11):2585-2589,2010.

[13] Li Wenliang.Key Technology Research of GPU Cluster Scheduling Management System[D].Wuhan: Huazhong University of Science&Technology.2011.

[14] WANG Zhuo-Hao,ZHAO Zhuo-Feng,FANG Jun,WANG Xi-Cheng.A SaaS-Friendly Service Community Model and Its Application in the Nationwide Service Network for Sharing Science and Technology Information[J].Chinese Journal of Computers.33(11): 2033-2043,2010.

附中文参考文献:

[5] 张舒,褚艳利等.GPU 高性能运算之 CUDA[M].北京:中国水利水电出版社.2009.10.

[10] 陈浩,彭萃芬.孙建华等.XenRPC:安全的虚拟机远程过程调用设计与实现[J].计算机研究与发展,2012,49(5):996-1004.

[12] 郝水侠,曾国荪,谭一鸣等.计算任务与体系结构匹配的异构计算可扩展性分析[J].电子学报,2010,38(11):2585-2589.

[13] 李文亮.GPU 集群调度管理系统关键技术的研究[D].武汉：华中科技大学.2011.

[14] 王卓昊,,赵卓峰,,房俊,,等.一种 SaaS 模式下的服务社区模型及其在全国科技信息服务网中的应用[J]. 计算机学报, 2010, 33(11): 2033-2043.