基于Docker的高性能并行异构系统设计与实现

摘要

在高性能计算系统中构建和部署基于图形处理加速器的软件环境是一项极具挑战性的任务。高性能应用程序必须可靠地运行在多个平台和环境中，并在解决复杂的软件堆栈依赖关系的同时充分利用图形处理加速器的资源。容器通常用于在多台机器上无缝部署基于CPU的应用程序。有了这个用例，容器就是硬件无关的和平台无关的。使用英伟达图形处理加速器显然不是这样，因为它使用专用的硬件，并且需要安装英伟达驱动程序。因此，Docker Engine本身并可以直接访问到英伟达图形加速器并发挥其效率。

容器是一种轻量级的虚拟化技术，具备便携式，易于构建和部署，占用空间小，运行时间少等优势。通过将应用程序及其环境封装到标准的软件单元中来解决部署复杂环境的问题。为了使Docker 容器在使用英伟达图形加速器的同时具备便携式和跨平台的特点，我们使用的解决方案是分离图像与英伟达驱动程序之间的联系。启动目标机器上的容器时，将安装所需的字符设备和驱动程序文件。

本文研究分析了基于Volta架构的图形处理器和统一计算设备架构，如合作组织，动态平衡度等编程模型。笔者通过在实际工作中遇到的问题和需求为Docker提供了运行时的扩展应用，它无缝隙的支持基于Volta架构的图形处理器，并在容器内能实现所有最新CUDA提供的技术方案，它为容器化应用程序提供了访问图形加速器和主机系统的专用机制，从而有效地解决了高性能计算资源在容器内的移植性问题。

在实际工作中，优化后的容器将应用程序包装到隔离的虚拟环境中，通过此项技术，大大简化了CUDA和操作系统在数据中心环境部署工作，运维和测试人员也可以轻松地集成和隔离加速应用程序，无需任何修改，并将其部署在任何支持GPU的基础设施上,极大地节省了资源。

Abstract

Building and deploying a software environment based on a graphics processing accelerator in a high-performance computing system is a challenging task. High-performance applications must reliably run on multiple platforms and environments and take full advantage of the resources of the graphics processing accelerator while resolving complex software stack dependencies. Containers are typically used to seamlessly deploy CPU-based applications on multiple machines. With this use case, the container is hardware-independent and platform-independent. The use of NVIDIA graphics processing accelerator is clearly not the case because it uses dedicated hardware and requires the installation of NVIDIA drivers. As a result, the Docker Engine itself does not support encapsulating the NVIDIA graphics accelerator inside the container.

A container is a lightweight virtualization technology that is portable, easy to build and deploy, takes up less space and runs less time. Solve the problem of deploying complex environments by encapsulating the application and its environment into standard software units. In order to make the Docker image with portable and cross-platform features while using the NVIDIA graphics accelerator, the solution used by the container is to make the image irrelevant to the NVIDIA driver. When you start the container on the target machine, the required character device and driver files are installed.

This paper analyzes and analyzes the graphics processor based on Volta architecture and the unified computing device architecture, such as cooperative organization, dynamic balance and other programming model. The author provides Docker with run-time extended applications through the problems and requirements encountered in real work. It seamlessly supports the graphics processor based on the Volta architecture and enables the implementation of all the latest CUDA technology solutions in the container. Provides a specialized mechanism for accessing graphical accelerators and host systems for containerized applications, effectively addressing the portability of high-performance computing resources in containers.

In actual work, the optimized container wraps the application into an isolated virtual environment, greatly simplifying the deployment of CUDA and the operating system in the data center environment, and the ease of integration and isolation Accelerate applications, without any modification, and deploy them on any GPU-enabled infrastructure, dramatically saving resources.

**Keywords**: GPU, Docker, High Performance Computing, Container, Virtualization, Heterogeneous System, CUDA

目录

1. 绪论
   1. 课题背景和意义
   2. 国内现状和国外研究进展
   3. 课题研究的创新点
   4. 论文的主要目标
   5. 本文的组织结构
2. 系统相关技术研究

2.1 虚拟化容器技术

2.2 高性能异构技术

2.3 Postgresql数据库技术

2.4 Go编程

2.5 基于MVC的Django框架

2.6 Python编程

2.7 前端相关技术

1. 基于CUDA的高性能平台设计

3.1 CUDA编程模型与环境

3.2 合作组织

3.3 动态平衡度

3.4 NVML

3.5 GPU P2P

3.6 GPUDirect or NVLINK

3.7 基于Volta架构的硬件平台

第四章 容器内部实现的异构系统及Web服务

4.1 Docker容器的组织结构

4.2 Docker容器的外部扩展

4.3 基于Django的Web框架

4.4 数据库服务

4.5 前端服务

第五章 高性能异构系统的实现和测试

5.1 可访问GPU的Docker容器

5.2 Cholesky算法实现及移植

5.3 性能调优设计

5.4 Linpack测试系统

5.5 基于Django开发的Web服务

第六章 结论与展望

6.1 总结

6.2 展望

致谢

第一章 绪论

1.1 课题的背景介绍

长期以来计算机科研人员最为担忧的是软件的可移植性问题。起初在20世纪50年代后期，人们创造出COBOL编程语言，这主要用来降低软件移植到新平台上去的复杂性和成本[1]。自那时起已经有60多年了，目前人们仍然在努力应对这些问题的根本性挑战。在高性能计算（HPC）领域，人们越来越多的把注意力聚焦在便携性上，这是因为在不损失高水平计算性能的前提下，高性能应用程序通常需要被运行在各种各样的平台和环境中。

从最初人们在COBOL中的工作，许多创新使得我们更加关注多年来软件的可移植性。让我们考虑下面的几个方面：编程语言（例如COBOL，Fortran，C，C++，Python），便携式库（例如Boost C++，PETSc），通用操作系统（例如嵌入式系统到超级计算机的Linux），计算机平台标准（例如IBM-PC,基于x86的架构），软件模式（例如通过可重用的软件组件的可移植性）。尽管这些努力已经成为人们处理软件的重要工具，但是由于对于如：性能，资源限制，依赖机器的功能，与操作系统有关的要求，软件库和各种工具的可用性日益增长的需求使得软件可移植性成为一个极为复杂的问题。因此，开发人员通常会将可移植性作为与其他要求的权衡。

在高性能计算需求的背景下，人们要求程序同时能兼顾高性能和可移植性[6]。特别是在超级计算机架构中这项要求特别具有挑战，以为配置和软件环境在系统和供应商之间差异很大。我们要求高性能软件能适应各种不同的系统环境。工程师们最需要考虑的是在跨超级计算机站点移植代码任务时候产生的时间和金钱成本。然而，大多数情况下，用户和开发人员都希望将其时间花在新的科学研究和软件开发商，而不是在平台和环境之间移植应用程序。因此，简化和加速应用程序的移植的工作流程可以大大提高用户和开发人员的生产力。

虚拟化技术在过去十多年中呈现快速增长的趋势，它展现出可以胜任轻松移植和部署应用程序，尤其在云环境中被使用的最为广泛。

本文从实际需求出发，结合最新的Volta架构与CUDA的配合使用，采用英伟达公司GPU来满足人民日益增长的对高性能计算的需求。在公司内部，我们又需要快速搭建异构环境，将平时工作中对GPU的测试记录写入数据库，并对其进行纵向对比，从而保证公司产品不会发生回归性质的性能衰退。本人使用市面上多种技术结合，选择Docker作为基础容器，将应用算法封装进入容器，并采用基于Django的web开发框架配合后端postgresql数据库和前端nginx服务器，为用户提供快速简便的操作界面，并提供稳定长久的数据支持。

1.2 国内现状和国外研究进展

1.2.1 虚拟化技术现状

上世纪60年代开始，美国的计算机学术界就有了虚拟技术思想的萌芽。1959年克里斯托弗（ChristopherStrachey）发表了一篇学术报告，名为《大型高速计算机中的时间共享》（TimeSharinginLargeFastComputers），他在文中提出了虚拟化的基本概念，这篇文章也被认为是虚拟化技术的最早论述。L.W. Comeau 和 R.J. Creasy 创造性地设计了一种名为 CP-40 的新型操作系统，该操作系统实现了虚拟内存和虚拟机。

虚拟化技术在 20 世纪 60 年代首次出现，由IBM 率先实施：对大型机进行逻辑分区以形成若干独立虚拟机的一种方式。这些分区允许大型机进行“多任务处理”：同时运行多个应用程序和进程。原因是当时大型机是十分昂贵的资源，因此设计虚拟化技术来进行分区，作为一种充分利用投资的方式，解决了大型机的僵化和使用率不足的情况。

在 20 世纪 80 年代和 90 年代，由于客户端-服务器应用程序以及价格低廉的x86服务器和台式机组成了分散的计算机架构，大型机上的虚拟化技术处于停滞不前的状态。

在20世纪，虚拟化技术基本上都是服务器虚拟化，进入了21世纪，随着IT的发展，虚拟化的思路被借用到服务器以外的领域(包括存储，网络，桌面应用等)，形成了各种各样的虚拟化技术。本文重点从桌面应用来看，开始出现了应用虚拟化（也称桌面虚拟化）的技术，该技术把应用程序的人机交互逻辑（应用程序界面、键盘及鼠标的操作、音频输入输出、读卡器、打印输出等）与计算逻辑隔离开来，客户端无需安装软件，通过网络连接到应用服务器上，计算逻辑从本地迁移到后台的服务器完成，实现应用的快速交付和统一管理。

1.2.2 高性能计算平台现状

自1946年第一台电子计算机ENIAC问世至今，超级计算机的发展已先后经历了5个阶段或5代，即早期的单处理器巨型机、向量处理系统、大规模并行处理系统、共享内存处理系统和机群系统。

1983年，中国第一台被命名为“银河”的亿次巨型电子计算机在国防科技大学诞生。它的研制成功向全世界宣布：中国成了继美、日等国之后，能够独立设计和制造巨型机的国家。

1992年，国防科技大学研制出银河－II通用并行巨型机，峰值速度达每秒10亿次，主要用于中期天气预报。

1993年，国家智能计算机研究开发中心（后成立北京市曙光计算机公司）研制成功曙光一号全对称共享存储多处理机，这是国内首次以基于超大规模集成电路的通用微处理器芯片和标准UNIX操作系统设计开发的并行计算机。 1995年，曙光公司又推出了曙光1000，峰值速度每秒25亿次浮点运算，实际运算速度上了每秒10亿次浮点运算这一高性能台阶。曙光1000与美国Intel公司1990年推出的大规模并行机体系结构与实现技术相近，与国外的差距缩小到5年左右。

1997年，国防科技大学研制成功银河－III百亿次并行超级计算机系统，峰值性能为每秒130亿次浮点运算。

1997至1999年，曙光公司先后在市场上推出曙光1000A，曙光2000－I，曙光2000－II超级服务器，峰值计算速度突破每秒1000亿次浮点运算。 1999年，国家并行计算机工程技术研究中心研制的神威I计算机，峰值运算速度达每秒3840亿次，在国家气象中心投入使用。

2004年，由中科院计算所、曙光公司、上海超级计算中心三方共同研发制造的曙光4000A实现了每秒10万亿次运算速度。

2008年，“深腾 7000”是国内第一个实际性能突破每秒百万亿次的异构系统集群系统，Linpack能突破每秒106.5亿次。

2008年，曙光5000A实现峰值速度230万亿次、Linpack值180万亿次。作为面向国民经济建设和社会发展的重大需求的网格超级服务器，曙光5000A可以完成各种大规模科学工程计算、商务计算。

2009年10月29日，中国首台千万亿次超级计算机“天河一号”诞生。这台计算机每秒1206万亿次的峰值速度和每秒563．1万亿次的Linpack实测性能，使中国成为继美国之后世界上第二个能够研制千万亿次超级计算机的国家。 2011年11月，中国第一台实测性能超千万亿次的超级计算机曙光“星云”在国家深圳超算中心全面开通运行。“星云”系统峰值为每秒3000万亿次，实测Linpack性能达到每秒1271万亿次，是世界第三台、中国第一台实测双精度浮点计算超过千万亿次的超级计算机。

由日本政府出资、富士通制造的巨型计算机“K Computer”目前落户于日本理化研究所，并成功从中国手中夺回运算速度排行榜第一的宝座。“K Computer”目前的运算速度为每秒8000万亿次，而到2012年其完全建成时，运算速度将达到每秒一万万亿次。“K Computer”比现居第二的中国超级计算机速度快出约3倍，甚至比排名第2至第6的计算机运算速度总和还要快。

进入新世纪，随着研制高端计算机系统的诸多关键技术被攻克(尤其是机群技术)，我国自行研制的高端计算机系统已开始形成自己的品牌系列和一定的市场规模，其发展呈现星火燎原之势头。近两年，随着“神威”、“银河”、“曙光”、“深腾”、“天梭”等一批知名产品的出现，使我国成为继美、日之后第三个具备高端计算机系统研制能力的国家，被誉为世界未来高端计算市场的“第三股力量”。

1.2.3 容器技术现状

Docker利用Linux 的一些内核机制例如 [cGroups](https://www.kernel.org/doc/Documentation/cgroups/cgroups.txt)、命名空间和 [SElinux](http://selinuxproject.org/page/Main_Page) 来实现容器之间的隔离。起初 Docker 只是 [LXC](https://linuxcontainers.org/) 容器管理器子系统的前端，但是在 0.9 版本中引入了 [libcontainer](http://blog.docker.com/2014/03/docker-0-9-introducing-execution-drivers-and-libcontainer/)，这是一个原生的 go 语言库，提供了用户空间和内核之间的接口。

容器技术需要解决最为核心的问题是针对软件的创建、发布和运行。它通过将运行环境和应用程序打包到一起，来解决部署的环境依赖问题，整整做到跨平台的发布和使用。容器会比虚拟机更高效，因为它们能够分享一个内核和分享应用程序库。

最初一家法国公司DotCloud提供PaaS服务，它能对支持多种语言的运行环境，如Ruby、Java、Python等。可是在PaaS领域有太多巨头已经布局，DotCloud考虑如果不开源，很难与巨头竞争，所以就考虑将Docker项目开源，至少能在开源社区得到个好名声。2013年3月，Docker正式以开源形式发布，此举让容器领域有了新的春天，截止2015年11月，Docker在Github上收到超过25600个赞，超过6800次克隆，以及超过1100名代码贡献者，成为20个最具影响力的Github开源项目。目前，世界上几乎所有的科技公司都在拥抱以Docker为代表的虚拟化生态圈。

1.2.4 异构系统现状

当前图形设备处理器易于获得并且价格可以被广泛接受，被认为是搭建异构系统的理想硬件设备。基于GPU硬件架构的特点，如：晶体管数量庞大，只有少数逻辑控制和缓存部分，特别适合用来做高任务量且逻辑相对独立的矩阵运算。所以CPU是专门为顺序串行处理而优化的几个核心组成，而GPU则有用一个数以千记的更小、更高效的核心组成的大规模并行计算架构。

NVIDIA公司于2007年发明了基于CUDA的并行编程生态系统，它对C语言进行了扩展，实现了异构系统模型，因为CUDA提供了一整套开发生态系统，包括基于编译器，调试器，调优器，集成开发环境，和各类加速应用数学库等，让基于GPU的编程变得从此简单。

在2017年GPU技术会议上，NVIDIA向全球发布了CUDA 9，最新版本的CUDA提供了空前强大的并行计算平台和编程模型。比如合作组织，它重新定义了在kernel内部线性启动机制。还有针对NPP运行库的优化，相比Intel公司的多核至强系列处理器，在图像处理方面的性能提升了20-100倍。

1.3 论文的创新点

重点思路

支持访问NVIDIA GPU的Docker容器

算法的方法优化，在GPU上实现的算法方法优化，如Cholesky分解

待扩展

1.4 论文的主要目标

重点思路

Django的定制化服务，封装在容器内部，服务于测试任务

深入理解Votla的硬件特性

基于Go语言的Docker应用扩展

MVC模型的编程思想

前端框架实现订制服务

后端数据库模型设计

待扩展

1.5 本文的组织结构

待完善

第二章 系统相关技术概述

2.1 虚拟化容器技术

大概描述：

虚拟化是提高现有计算资源利用率的常用策略，传统上，使用它就意味着为了获得虚拟机的便利而需要接受性能大幅降低的代价。如今，这种性能上的降低已经得到很大的缓解。由于更快的处理器的出现以及更高效的虚拟化解决方案的诞生，现在我们也能把一台普通的台式计算机作为虚拟机的宿主机来使用。

在日常工作中，处理客户问题要求能快速搭建出和客户问题想匹配的相关系统环境。另外在测试工作中，亦要求工程师能提高工作效率，不要把过多的精力放在搭建系统环境上，而是能更深入的对平台和GPU进行验证。

待扩展

2.2 高性能异构平台

2.3 Postgresql数据库技术

2.4 Go编程

2.5 基于MVC的Django框架

2.6 Python编程

2.7 前端相关技术

第三章 基于CUDA的高性能平台设计

3.1 CUDA编程模型与环境

3.2 合作组织

3.3 动态平衡度

3.4 NVML

3.5 GPU P2P

3.6 GPUDirect or NVLINK

3.7 基于Volta架构的硬件平台

第四章 容器内部实现的异构系统及Web服务

4.1 Docker容器的组织结构

4.2 Docker容器的外部扩展

4.3 基于Django的Web框架

4.4 数据库服务

4.5 前端服务

第五章 高性能异构系统的实现和测试

5.1 可访问GPU的Docker容器

5.2 Cholesky算法实现及移植

5.3 性能调优设计

5.4 Linpack测试系统

5.5 基于Django开发的Web服务

第六章 结论与展望

6.1 总结

6.2 展望

致谢