2019 届本科生学士学位论文　　学校代码：10269

華東師範大學

East China Normal University

本科生毕业论文

Nvidia 新架构 GPU 为机器学习应用带来的性能提升的研究与评估

Research on performance of ML applications using Nvidia new GPUs

姓　　名：刘子汉

学　　号：10152130243

学　　院：计算机科学与软件工程学院

专　　业：计算机科学与技术

指导教师：钱莹

职　　称：副教授

2019　年　5　月

目　录

摘 要

本文主要针对 Nvidia 新架构的 GPU（图灵架构）为机器学习应用带来的性能提升进行研究，由于目前实际使用中的应用很难达到 Nvidia 官方宣传的性能提升幅度，故本文将从问题类型、代码结构结合硬件、指令特征对这一现象进行研究，并提出相应的建议。本文主要采用定量方法，通过不同世代的硬件和 SDK 进行横向比较，以及同一世代硬件、SDK 和不同类型应用进行纵向比较；并总结出特征。在研究中较为重要的部分为新硬件中加入的张量核心（Tensor Core）以及对应的线性代数库 CUTLASS，文章将通过混合矩阵运算、矩阵乘法、卷积运算等对其进行评估；其他还涉及了传统的矩阵运算库 CUBLAS、模型优化器 TensorRT 以及最为基本的浮点计算、内存种类等。

根据实验结果，新架构硬件中张量核心对于机器学习应用的类型、计算类型、超参数等条件敏感；要达到期望的性能，输入数据规模、形状、运算占比等方面有较为严苛的需求；在矩阵较为稀疏、输入规模较小时 CUSPARSE 稀疏矩阵库和基于纹理内存的方法能取得更高性能；而计算输入较为规律、符合硬件形状时张量核心能带来显著提升。至于网络推理阶段，TensorRT 在各种情况下均能带来明显的提升。在实际应用中，训练阶段应根据任务特征合理选择硬件、SDK 和内存系统使用；而在推理阶段应利用 Tensor Core 提升吞吐量。

关键词： Tensor Core，TensorRT，通用矩阵乘法，图灵架构

Abstract

This paper is focusing on the performance improvement in Machine Learning application brougnt by Nvidia’s new architecture (Turing architecture) GPU. Since currently the Machine Learning application actually in used can hardly get as much improvement as mentioned in Nvidia’s official White Paper, so, this paper will research this situation through the type of the application, the structure of the source code combining with feature of the hardware and instructions, thus give corresponding recommendation about coding. This paper uses quantitative methods, doing both horizontal comparation with hardware and SDK of different generations and vertical comparation with different types of problem running on the same generation of hardware and SDK, through which the pattern and feature can be extracted. Among all the new features, the most important is Tensor Core and corresponding library CUTLASS (CUDA Template Linear Algebra Subroutine), this paper evaluate this unit through GEMM, Matrix Multiple, Convolution, etc. Also, traditional matrix library CUBLAS, optimizer TensorRT, Float Point and GRAM are also mentioned.

In the conclusion, Tensor Core in the new architecture GPU is very sensitive to the type of applications, type of calculations, meta parameter, etc., to achieve expected performance, the scale of the data, shape of the data and type of calculations should be well fit to the hardware. Moreover, in some situation including the input matrixs are sparse and the scale of the input data is small, library oriented to sparse matrix (CUSPARSE) and methods based on texture memory will gain much higher performance, and in situation that the input fit the hardware well, the Tensor Core can bring the application a significant improvement in performance. When it comes to the inference stage, TensorRT can bring a significant improvement in almost all the situation.

So, in the training stage of actual application, the usage of hardware, SDK, memory, etc. should be chosen appropriate based on the feature of the applications, and in the inference stage, do not hesitate to use TensorRT!

Keywords： Tensor Core, TensorRT, GEMM, Turing Architecture

第一章 引言

1.1 研究背景

近年来，人工智能在全球无论是否是计算机相关行业中，都掀起了一股热潮，尤其是深度学习更是赚足了眼球。作为深度学习应用中计算能力支撑的并行计算硬件与软件更是迅猛发展，而英伟达 (Nvidia) 更是在并行硬件领域独占鳌头。

在 2017 年，英伟达发布了一款基于伏特架构 (Volta) 的面向深度学习的 GPU，Tesla V100[1]，其中搭载了一些实验性的新技术；之后在 2018 年第三季度，英伟达发布了新一代图灵架构 (Turing)，在该架构中，正式引入了许多革命性的新技术，同时也对原有技术做了很大的改进。有面向深度神经网络应用的张量核心 (Tensor Core)[2]，能够大幅加速在神经网络训练、推理中的混合精度矩阵计算，该核心最先实验性地搭载于 Tesla V100，在图灵架构中上至面向深度学习推理的 Tesla T4，下至面向游戏玩家的 RTX 2080Ti 都搭载了这款核心；用于更高效搭建分布式计算平台的第二代端对端互联总线 (NV Link 2.0)[3]，相对于原有的 QPI 等总线，该总线能够直接互连 GPU，且提供远高于原先 SLI 技术所能提供的带宽；以及针对游戏玩家推出的实时光线追踪技术 (RTX)，该技术不在本文的讨论范围内。同时，由于 GPU 中 CUDA 计算单元架构包括流多处理器，纹理单元等的优化，在性能大幅提升的同时，热设计功耗 (TDP) 仍然维持在了上一代硬件的 250W。

在并行软件方面，与硬件一起，英伟达将其面向并行程序开发的 SDK CUDA 的版本更新到了 10.0，在游戏应用、通用计算方面针对新架构的特性进行了优化；同时发布了基于 CUDA 10.0 的进行线性代数计算的模板库 CUTLASS(CUDA Template Linear Algebra Subroutines)[4] 以利用其张量核心进行高效的代数运算。

然而，官方文档给出的性能提升仅仅包括单一模块的理论性能提升，如传统 CUDA 核心的浮点数值计算的理论峰值，新加入的张量核心的混合矩阵计算 (GEMM) 的理论峰值；NV Link 2.0 的理论峰值带宽等。在实际使用中，用户反映在网络推理方面以及基于支持新硬件的相关框架开发的机器学习应用中，提升并没有官方白皮书给出的的 9 倍之多 [5]，且同类型不同规模应用的性能提升幅度并不一致，性能提升对神经网络中参数数量、网络层数等因素较为敏感。实际上，官方给出的文档中的提升也仅为绝对计算性能的提升，没有考虑应用类型、平台构建等条件。且目前关于新架构 GPU 的研究主要集中在大型计算节点的扩展效率 [6]，基于 GPGPU-Sim 模拟的性能考察等 [7]，这些研究或是停留在表征性能层面、没有深入到代码或是中间代码层面；或是使用模拟技术、在 PC 机上进行模拟，尽管目前对于硬件的模拟运行的匹配度能够达到较高的水准，但是仍然有一定偏差，目前 GPGPU-Sim 的稳定版支持的最高的 CUDA SDK 版本为 4.0，开发版本支持的最高的 CUDA SDK 版本为 9.2。本文将直接针对真实的，单一的，图灵架构的 GPU：RTX 2080Ti 进行深入，结合版本最新的 CUDA SDK 10.0 以及对应的软件库包含 CUTLASS，CUBLAS 等，从架构、PTX 中间代码层面、SASS 机器码层面对 GPU 在使用 GPU 加速的机器学习应用中的性能以及性能提升进行研究和评估；根据研究和评估结果以及分析得到的原因对现有 CUDA 代码进行优化；且将结合目前对于新老架构的对比研究 [8]，将新架构与麦克斯韦架构的 GPU：GTX Titan X 与帕斯卡架构的 GPU：GTX 1080Ti 进行横向对比，从实际替换成本、环境搭建成本、维护成本、性能/功耗比等角度对新架构进行评估与进一步设想。

在最近刚结束的 GTC 2019 会议中，Nvidia 发布了若干面向机器学习的硬件、软件。包括专为张量计算

设计的 Turing Tensor Core GPU，嵌入式平台的 Jetson Nano[9]，将机器学习相关计算库整合起来的 CUDA X[10]，这些都将在后文提到，但由于这些本质上都基于目前的 Turing 架构，故不会单独进行详细地说明。

1.2 目前展开的工作

本文的研究同时涉及硬件和软件：Nvidia 新架构的 GPU 与使用 GPU 加速计算的机器学习应用，注意这里的机器学习应用并不只是现在大行其道的深度学习，还包括传统的基于概率模型的方法等。

在近年来深度学习迅猛发展之前，关于使用 GPU 并行化机器学习算法的研究就已有很多，甚至可以说正是基于 GPU 的强大并行计算能力，深度学习才能发展如此迅速。早在 2005 年，D. Steinkraus 等人便对在 GPU 上实现机器学习算法进行了研究，当时实现的算法是 OCR，如今 OCR 能够非常快速、方便地通过各种框架、语言实现。然而这一研究奠定了使用 GPU 加速机器学习应用的基础 [21]。之后的几年中，除去 OCR 外，基于 GPU 的各种并行机器学习算法包括 kNN[22]，支持向量机 [23] 都慢慢成熟。由于贝叶斯网络的精确推理是个 NP 难的问题，其性能受限于硬件水平，然而根据 Md Vasimuddin 等人的研究 [24]，通过并行方法将延迟降低了许多。由于传统机器学习方法有着延迟低、模型小、训练时间短、可解释性强等优点，在深度学习迅猛发展的今天，传统机器学习方法仍然占有很大的市场，故评估 GPU 对传统机器学习加速的性能是十分有必要的。本文中在评估传统机器学习的部分便采用了并行的支持向量机算法 (SMO-SVM)，如今，不管是在工业生产领域还是学术研究领域，该算法仍占有一席之地。

之后不久，深度学习、神经网络便迎来了爆炸式的发展。实际上在二十世纪末，便已经有完整的神经网络算法的理论基础；在 1979 年，日本学者福岛邦彦提出了 Neocognition 模型，其中使用多层网络以及神经元对图像特征进行提取和筛选被认为是启发了卷积神经网络的开创性研究 [25]。1989 年，LeCun 首次在论文中提出了“卷积”一次，卷积神经网络因此得名 [26]。在 1993 年，贝尔实验室对 LeCun 的工作进行了代码实现，并大量部署于手写支票识别系统，然而限于当时计算能力低下，基于神经网络的研究也停滞在了理论阶段，其主要原因便是网络中需要训练的参数太多，网络结构复杂，在当时没有芯片能满足如此高的性能要求 [27]。然而，随着高性能 GPU 芯片的出现，基于神经网络的方法正如日中天地发展。从 TinyCNN 等较轻量级、功能单一的库，到 TensorFlow、PyTorch、Caffe、Chain 等完整、易于使用的框架，这些工具或是本身就是基于 GPU 编写的，或是慢慢更新对 GPU 的支持；目前市面上的绝大部分该类产品均支持使用 GPU 运行，单单使用 CPU 进行深度学习训练已经成为历史。在本文中，卷积神经网络由于它的广泛性、高性能、典型性，在本文中被选为深度学习部分评估的主要载体。

而准确地对 GPU 以及机器学习的性能进行评估也尤为重要。且为了深入研究 GPU 对于机器学习应用的性能提升的幅度以及不同提升幅度的不同原因，单单是对训练时间进行统计、评估是不够的。因本文涉及 Nvidia 新架构 GPU 中新加入的硬件以及对应 CUDA 中新加入的 API 等，故指令、运行流级别的分析是有必要的。Ali Bakhoda 等人曾设计实现了一种在软件层面对 CUDA 执行流进行指令级别的模拟和仿真的系统 GPGPU-SIM，该系统是基于 Kepler 架构的硬件以及 CUDA 3.0 版本，在缓存命中率、分支、指令乱序执行等方面能达到 90-95% 与真实硬件的吻合程度 [28]。在之后 Mahmoud Khairy 等人对该系统进行了改进，使其支持伏特架构 (Volta) 的 GPU 以及对应的 CUDA 9.0 [29]。然而，目前 GPU 硬件已经更新到图灵架构 (Turing)，基于安培架构 (Ampere) 的硬件也即将发布；对应的 CUDA 版本已经更新到了 10.1，在指令执行、调度方式等方面都发生了很多的改变；且 Mahmoud Khairy 等人的工作主要着重于 GPU 的内存等方面。故能够准确评估新硬件的系统非常必要。本文中选用了 nvprof、NSight 等公开的工具，这些

工具能从指令运行时间、访存、缓存命中率等方面对 CUDA 应用程序进行评估 [30]。

在新架构的 GPU 中，最为重要的便是新加入的计算单元：张量核心 (Tensor Core)，该运算单元能为深度神经网络中大量存在的张量计算带来明显的提升 [5]，然而这些数据是 Nvidia 官方白皮书中给出的数据，开发者社区中反映很少有情况能获得如 Nvidia 官方宣传所能得到的性能提升；而关于 Tensor Core 的研究少之又少，故本文并非旨在填补这方面研究的空白，姑且在新的方向进行一些稚嫩的尝试；且由于在 Nvidia 进行实习工作，有机会接触到许多内部资料，本文是一个很好的契机。

当然，仅有理论计算性能的研究是无力的，最终本文还是会回归实际，使用实际应用中的模型，如各种结构的神经网络、广泛使用的支持向量机并行库对新架构的 GPU 进行评估。从广为各大厂商使用的深度学习性能评估工具 DeepBench[31]，到使用 cudnn 从 C++ 源码实现的卷积神经网络，再使用 Tensor Flow 框架实现的各种结构的网络，包括 LeNet-5[26]，ResNet[32]，MobileNet[33] 等，本文将由下而上对新架构硬件再浮点精度计算、张量计算、卷积计算、矩阵计算等方面进行评估，不求全面，只求能给出启发。

1.3 我们的工作

近年来，机器学习尤其是深度学习发展迅猛，各种方便程序员搭建模型的框架层出不穷。考虑到机器学习应用的计算量要求日益攀升，这些框架都陆续推出了基于 GPU 的版本。为方便程序员搭建模型，框架本身对硬件的操作进行了抽象。然而，正是因为这一层抽象，忽略了许多硬件层面的细节，使得框架无法完全利用硬件的性能。这也导致了许多用户反映在实际应用中，新架构的性能提升并没有官方给出的文档数值、硬件参数 (包括流处理器、纹理/光栅单元)、甚至价格上涨幅度那么多。

为了尽可能在实际应用场景中提升硬件性能的利用率，本文将从如下层面对基于 CUDA 以及相关框架的机器学习应用进行研究与评估；挖掘理论与实际不符的原因；并做出适当的修改和建议。

- CUDA 源码
- CUDA 源码编译出的 PTX 中间代码
- 基于 CUDA 的框架的应用源码

因 CUDA SDK 10.0 发布不久，目前许多框架还未对该 SDK 进行相关优化；一些既存的 CUDA 应用仍是基于 CUDA SDK 9 甚至 CUDA SDK 8 进行编译的。所以本文将结合对于上述三个层面的分析结果，结合新硬件、新架构、新 SDK 的特征，在源码层面进行调整并给出一些编写相关程序时的建议；力图尽可能多地发掘新硬件、新架构的潜力。

1.4 本文的组织结构

本文在第 2 章中介绍了该研究的背景和相关的工作。首先介绍了基于 GPU 的机器学习应用与 CUDA 的相关背景知识。由介绍基于 GPU 的机器学习应用引出 CUDA 的相关介绍，包括 CUDA 应用的编程模型、编译过程、调用/执行方式。然后介绍了目前对于评估、模拟 GPU，尤其是 CUDA 应用性能开展的相关工作；由超微半导体 (AMD) 开发的 GPU 也具有通用计算功能，然而目前市面上还没有基于 AMD 开发的 GPU 的相关 SDK 或框架，故本文不做讨论。最后介绍了基于 CUDA 的可执行程序的汇编代码结构和使用 CUDA 源码编译得到的 PTX 中间代码的结构，供之后的分析使用。

本文在第 3 章中首先简要介绍实验动机、实验步骤以及实验结果。然后介绍了实验所需的工具、环境以及搭建方式等。接着详细介绍了我们的主要工作，包括基于单一功能、测试用的应用的 Benchmark、基于 CUDA 源码的机器学习应用的研究过程、针对汇编代码与 PTX 中间代码的研究过程、针对基于 CUDA 的相关框架的机器学习应用的研究过程以及根据分析得出的结果给出的修改、建议等。最后给出了各项实验的结果和对比，并进一步分析原因。

本文最后在第 4 章进行总结，并给出之后改进与深入工作的设想和预期。

第二章 背景及相关工作

2.1 NVIDIA GPU 硬件结构

2.1.1 GPU 芯片总体结构

在介绍新老架构区别之前，本节首先自顶向下简要介绍一下 NVIDIA GPU 芯片的结构。一块 GPU 芯片拥有若干图形处理器簇 (Graphics Processing Cluster, GPC)，由外围总线进行调度管理；一个图形处理器簇上有若干纹理处理器簇 (Texture Processing Cluster, TPC)；需要注意的是以上两种结构在编写 CUDA 程序时并不暴露。一个纹理处理器粗上有若干流多处理器单元 (Stream Multiprocessor, SM)，也是本文关注的重点。流多处理器单元被一个线程块调度器管理，所有流多处理器单元通过全局内存总线经过 L2 缓存共享全局内存。每个流多处理器单元中由若干流处理器 (Stream Processpr, SP)，然而这一概念随着流多处理器单元中运算单元种类的增加而被弱化了。在一个流多处理器单元内部的流处理器共享一个指令缓存，每个流处理器拥有自己的线程束调度器与寄存器文件；流处理器中包含若干种执行单元，有浮点单元，整数单元，在新架构中还加入了张量单元 (Tensor Core)，在 RTX 2080TI 上具体的参数为：一个 SM 包含 64 个单精度浮点算术单元，32 个双精度浮点算术单元，64 个 32 位整形算术单元，8 个混合精度张量单元，4 个线程束调度器和 16 个特殊功能单元；所有流处理器通过显存纵横矩阵 (CrossBar) 访问共享内存，或被称为 L1 缓存 [12]。

2.1.2 流多处理器单元 (SM)

上文提到过，六多处理器单元 (SM) 是本文关注的重点，其原因是每一次 NVIDIA GPU 芯片更新都会伴随着其计算能力 (Compute Capability) 的更新，计算能力指的是流多处理器单元 (SM) 支持的运算的等级，分为 Major 和 Minor。其中 Major 代号代表架构的更新，这也会带来许多新的硬件支持的运算，而 Minor 代号则代表同一架构下不同定位的流多处理器产品。如伏特架构的计算能力为 7.2，图灵架构的计算能力为 7.5，Major 代号一样就代表这两种架构其实并无太大修改，而 Minor 代号则代表伏特架构中流多处理器的类型是 Heavy，图灵架构中流多处理器的类型是 Lite。Lite 和 Heavy 一般用于区分消费级/工作站级 GPU，分别对应 GeForce 和 Tesla 代号。

2.1.3 存储模型与管理

NVIDIA GPU 的存储模型与其存储管理系统也是另一个重点。传统 CPU 编程模型中，寄存器、缓存等资源都是由 CPU 自行管理，而不开放给程序员。其原因在于 CPU 拥有的寄存器、缓存资源较为紧缺，为提高指令级并行能力，需要采用多队列乱序发射与寄存器重命名等技术。相对得，GPU 有较为充足的物理寄存器、缓存资源，程序员也对这部分资源掌握有一定的控制权 [14]。CUDA 中的存储设备如表2-1 所示。

需要注意的是，常量内存与纹理内存都是全局内存的一种虚拟地址形式。和常量内存一样，纹理内存也是一种只读内存；但是在访存、缓存加载方式上与其余存储系统存在较大差异，而这种差异会在某些应用中极大提高性能，在本文的实验中大量利用了纹理内存的特性，故将在下一节详细介绍纹理内存。

表 2-1 CUDA 存储系统层级

Table 2-1 CUDA storage system hierarchy

项目	大小	延迟 (时钟周期)	访问权限
寄存器文件	8KB-64KB/SM	10^0	GPU 端
共享内存 (L1,L2)	16KB-128KB/SM	10^1	GPU 端
常量内存	N/A	N/A	N/A
纹理内存	N/A	N/A	N/A
全局内存	-GB	10^2	CPU 端/GPU 端

2.1.4 纹理内存 (Texture Memory)

2.2 伏特/图灵架构新硬件

2.2.1 在流多处理器单元层面的差异

2.2.2 张量核心 (Tensor Core)

2.3 软件

本节将自顶向下介绍 NVIDIA GPU 硬件对应的不同层级的编程软件。

2.3.1 机器学习框架 (Tensor Flow)

2.3.2 CUDA C

2.3.3 机器码与中间代码 (SASS, PTX)

2.4 基于 GPU 的机器学习应用

表 3-1 实验环境

Table 3-1 Environment

项目	内容
CPU	AMD Ryzen ThreadRipper 2990WX 32C64T @ 3.0GHz
主板	MSI X399
内存	CORSAIR DDR4 3200 @ 16-15-15-15-34-1T 128GB
GPU	NVIDIA Geforce RTX 2080TI (Turing)
硬盘	INTEL750 NVMe PCIe 1.2TB * 2 @ RAID 0
系统	Windows 10 64-bit build 17763
CUDA	10.1, 10.0, 9.2, 9.0
其他	Jetson TX2 *

* 该硬件由 NVIDIA 提供。

表 3-2 实验工具

Table 3-2 Tools

项目	内容
Python 3.6	用于进行数据统计、编写 TensorFlow 应用
Conda 4.5.12	用于创建、管理、隔离 Python 环境
TensorFlow	1.12.0 和 1.13.0 版本的源码，用于对比、研究、调整
Bazel 0.16.0	用于从源码构建 TensorFlow
Msys2	用于从源码构建 TensorFlow
CMake 3.1.0	用于构建 CUTLASS
Nsight 6.0	用于后台监听 CUDA 应用，捕捉 Trace
nvprof	用于分析 CUDA 程序的 API 调用、分支效率等
git	版本控制
Perforce	版本控制
Ubuntu 16.04 Physical	用于执行 GPGPU-SIM 应用
GPGPU-SIM 3.2	用于从指令级别模拟 CUDA 程序
Visual Studio 2017	搭配 10.0.17763.0 版本的 SDK

第三章 评估 NVIDIA 新架构 GPU 的机器学习应用性能

3.1 实验工具与环境

3.1.1 实验环境

表3-1 中列出实验环境。

3.1.2 实验工具

实验中使用到了若干软件工具，如表3-2 列出。

3.2 实验详细过程

3.2.1 基于测试样例的 Benchmark

为了为接下来的实验设定基准，这一步先使用用途单一的测试样例测试绝对性能以及相应的提升，因不同架构的硬件各项参数 (包括流处理器数量、显存容量等) 不尽相同，所以直接对比不同架构硬件的性能是没有意义的，这里选择对比不同架构硬件在不同 SDK 下性能提升的比例。此处选用了 CUDA 10.0，CUDA 9.2，CUDA 9.0 三种 SDK，同时选用 9.2 与 9.0 的原因是因为 9.2 版本是为了图灵架构的 GPU Tesla V100 发布的 [35]，也在本文的研究范围内。

因为本文主要讨论新架构 GPU 在机器学习应用中带来的性能提升，故选用的评测样例大部分都与机器学习应用相关；主要从以下角度进行评估：通用矩阵乘法 (GEMM, General Matrix Multiply)、矩阵乘法运算性能、卷积运算性能、神经网络推理性能以及结合框架的综合性能。在评估这些性能时也会包含单/双精度浮点计算性能。

3.2.1.1 通用矩阵乘法 (GEMM, General Matrix Multiply)

待评测项目中最为重要的是通用矩阵乘法 (GEMM, General Matrix Multiply)，新架构对该运算进行了硬件、指令级别的优化，是与老架构最鲜明的区别所在。其混合体现在：运算中同时有加法和乘法，且精度同时涉及半精度浮点、单精度浮点和 8 位整数。与矩阵乘法相比，通用矩阵乘法被定义为：

$$C \leftarrow \alpha AB + \beta C$$

若将 β 置为 0，则该运算变为矩阵乘法运算。通用矩阵乘法这一运算在神经网络训练、推理中十分常见，根据官方文档，目前 Tensor Core 仅能用在 CNN/RNN 等特定结构的神经网络上，且只能用于前馈和反馈两部分。这个范围看起来很宅，然而在深度学习中占到了非常高的比重。式中操作数分别代表输入、权重和偏置，下文将简写为矩阵乘加。NVIDIA 在新的伏特架构与图灵架构中加入的张量核心 (Tensor Core) 正是专门加速这种运算的硬件；对应新硬件，在 PTX 中间代码层面新增了 $wmma$ 指令，在 SASS 机器代码层面则增加了对应的 $hmma$ 指令。该指令的作用为以指定的精度计算两个输入矩阵的乘积并将计算结果累加到指定的精度的矩阵中；指令进行的具体操作、操作数的数据精度、形状、存储方式 (行主元素/列主元素) 等通过指令中特定的字段指定。

在底层的实现中，张量核心以 4×4 的矩阵作为最小的计算单元，被称为 $tile$，任何输入都会被划分为 $tile$ 进行分块运算。在伏特架构以前 (Volta) 的帕斯卡架构 (Pascal)，一次 4×4 矩阵乘加需要首先调用 16 次整数点积运算 (若硬件支持 $idp/idp4a$ 指令)，再将结果累加到乘加矩阵中。而使用 Tensor Core 则仅通过 $hmma$ 指令直接完成。根据官方文档给出的描述，这种机制能使伏特架构相比帕斯卡架构再 FP16, INT8, INT4 京都中分别提供 8 倍、16 倍、32 倍的吞吐量提升。实际测试中，Tesla V100 再 FP16 精度下的 $m = 2048, k = 2048, n = 2048$ 规模的矩阵乘加中比 Tesla P100 块 9.3 倍 [5]，这也是上文提到的官方宣称的 9 倍。本节将在各种规模、精度、形状的情况下考察 Tensor Core 实际能够带来的性能提升并探究相应原因。

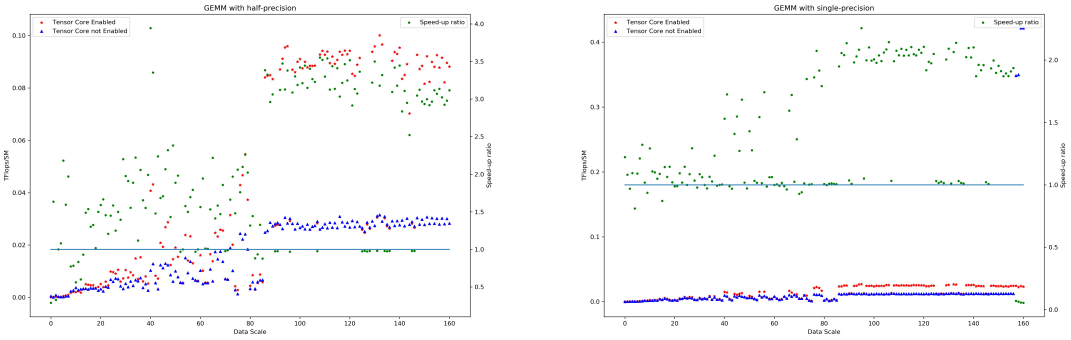

图 3-1　半精度/单精度 GEMM 性能

Figure 3-1　Performance of GEMM at Half and Single

l　实验结果　根据开发者社区的反映，新架构硬件性能的差别主要体现在问题规模、问题类型等方面 (张量维度、形状，训练/推理任务等)，而 NVIDIA 官方仅给出一种规模的结果，所以本节使用了自行编写的一系列测试用例，辅以深度学习测试套件 DeepBench，在开启和关闭新架构中张量核心的情况下进行测试。实验性能使用 TFlops/s 统计，方法为简单的运算数除以运算时间，运算时间的统计采用 CUDA 内置的 $cudaEvent$ 记录。

首先评估的是在不同问题规模下，开启和关闭新架构中的张量核心所能达到的性能，如图3-1 所示。随着问题规模的上升，总体加速比呈上升趋势，在大规模数据时半精度通用矩阵运算性能的加速比能达到 3 到 3.5 倍、单精度通用矩阵运算性能的加速比能达到 2 倍。然而，单纯考察数据规模发现加速比差距非常大，甚至是在大规模数据中仍然存在开启张量核心后性能不如不开启张量核心的情况，结合文档在通用矩阵运算一章中在指令中需要指明运算的最小单元这一点中 [34]；可以推测出输入矩阵的“形状”对张量核心的性能由较大影响。

为了研究输入矩阵“形状”对于加速比的影响，由于两个输入矩阵涉及三个维度，故采用控制变量法，控制 m, n, k 中某一维度考察另外两个维度对于加速比的影响。由于测试数据中存在部分离群值 ($N \geq 500000$)，这会对作图精度产生极大影响，故先予以剔除。实验结果如图3-2 所示。可见在两个输入矩阵的三个维度中，两矩阵共享的维度 K 对于性能的影响最为显著。

关于矩阵的形状、维度对于性能的影响，其原因将在后文结果分析中详细说明。以上实验数据以及性能旨在考查开启和关闭张量核心时性能提升幅度，故开启的线程块数量和线程数量较小，相应的 GPU 占用率也较小，导致所得性能并非 GPU 峰值性能。为测量峰值性能，这里还是用 NVIDIA 官方发布的用于线性代数计算的模板库 CUTLASS(CUDA Template Linear Algebra Subroutines)，该模板库根据新架构硬件特性编写，提供了许多测试样例供参考，这里使用 CUTLASS 测试得到的性能作为峰值性能基准。

CUTLASS 库中的 GEMM 运算有多种精度可供选择：HGEMM、SGEMM、DGEMM、CGEMM、ZGEMM 和 IGEMM，分别代表半精度浮点、单精度浮点、双精度浮点、单精度复数、双精度复数和八位整数。鉴于之前的测试并不涉及复数，此处也不选用复数精度作为基准。需要注意的是测试样例后缀中存在 _$[n/t][n/t]$ 分别代表运算中输入矩阵的数据存储、分布方式，即行列是否转置 (如上文提到的，CUDA 中矩阵存储分行主元素和列主元素存储)。图3-3 为测得的性能基准。

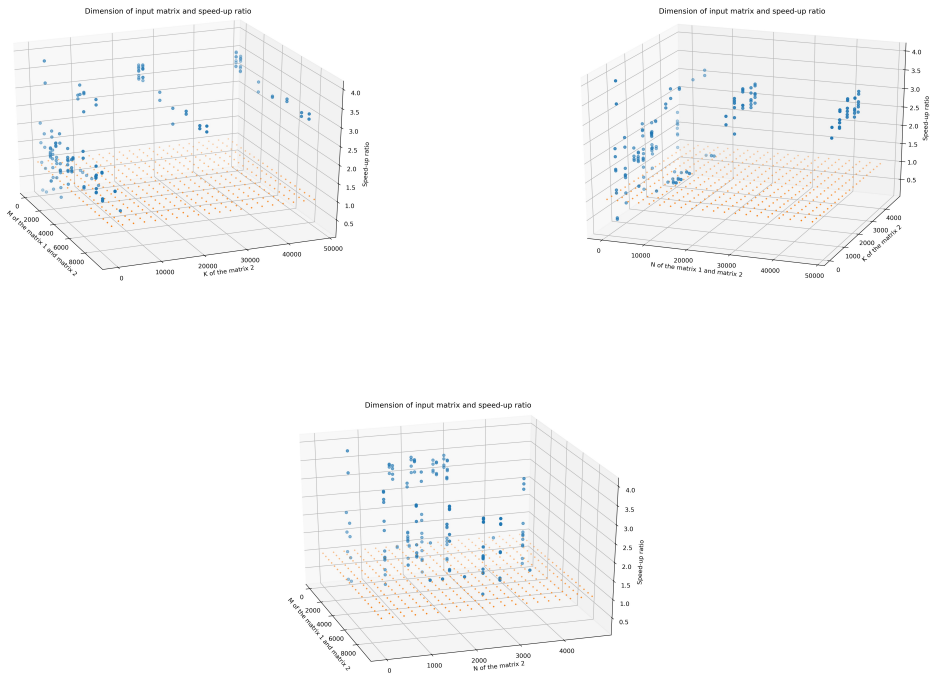

图 3-2 输入矩阵维度与加速比的关系

Figure 3-2 Relationship of input matrix dimension and speed-up ratio

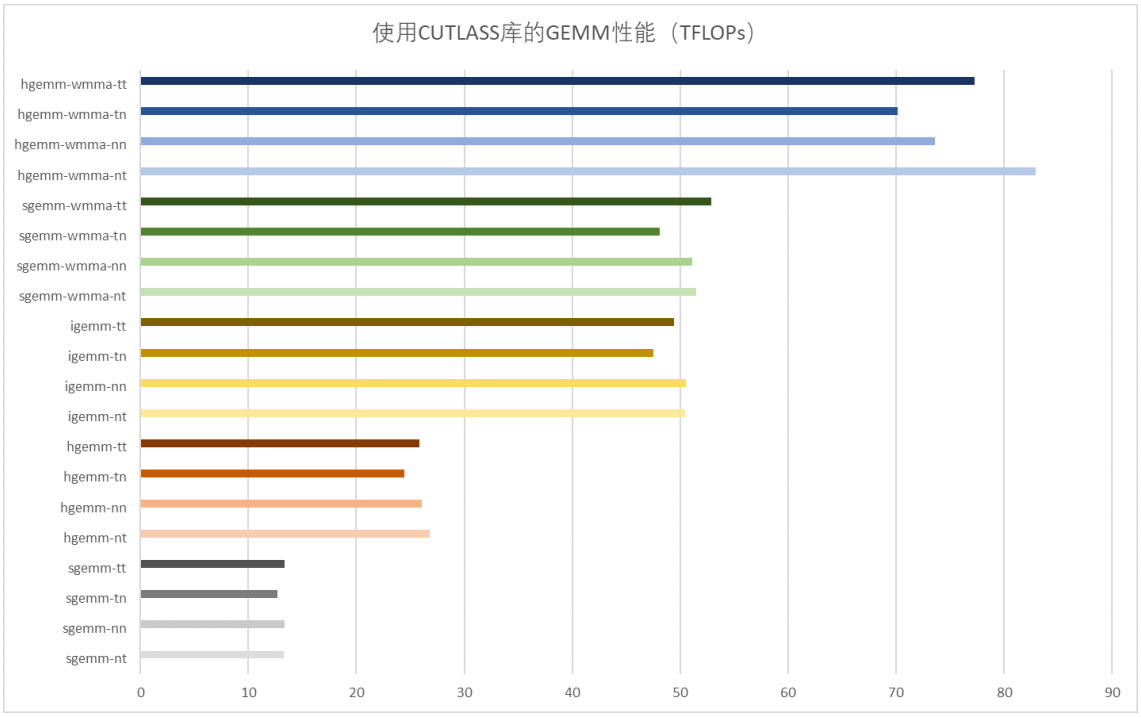

图 3-3 使用模板库测得的 GEMM 性能

Figure 3-3 GEMM Performance with CUTLASS

II 结果分析

3.2.1.2 矩阵乘法运算

I 实验结果

II 结果分析

3.2.1.3 卷积运算

I 实验结果

II 结果分析

3.2.1.4 神经网络推理

I 实验结果

II 结果分析

3.2.2 基于 CUDA 源码的应用

3.2.2.1 卷积神经网络

I 实验结果

II 结果分析

3.2.2.2 并行支持向量机

I 实验结果

II 结果分析

3.2.3 基于 TensorFlow 框架的应用

I 实验结果

II 结果分析

第四章 总结与展望

本文主要通过自底向上的方式对 Nvidia 最近发布的新架构硬件 (伏特、图灵架构) 在机器学习应用中的性能提升进行了评估，其中在新架构中新加入的张量核心 (Tensor Core) 是本文考察的重点，并根据评估结果给出了编程建议，以及对于下一代硬件的一些合理设想。

在正式评估之前，本文对基于 CUDA 的 GPU 编程模型进行了简要介绍，其中包含 CUDA 应用程序的编写步骤、调用方式、内存模型等，同时还对中间层的 PTX 代码进行了简要介绍，这些知识对于后文性能分析部分有较大的帮助。接下来本文便从三个层级对基于 CUDA 的机器学习应用在新架构硬件上的性能提升幅度进行了评估。

首先，本文涉及的最低层面便是用途单一、专为评估绝对性能设计的简单应用进行基准测试，这些应用大部分是 Nvidia 官方发布的测试用例。这些测试用例涵盖混合矩阵运算 (GEMM)、矩阵乘法、卷积核神经网络推理。根据实验得到的结果，在混合矩阵运算 (GEMM) 方面，新架构硬件能在操作数形状、尺寸与硬件参数、调用特征较为匹配的情况下取得大幅度的性能提升，而这些显著的性能提升是采用“用精度换速度”的策略，计算时数据精度均为 FP16/INT8 等低于传统的精度；然而在不匹配的情况下，性能下降极为明显。在矩阵乘法方面，我们评估了 cuBLAS 在新架构上的性能提升，结果是在所有情况下使用 cuBLAS 进行矩阵乘法运算优势都极为明显，且不依赖于操作数的形状、尺寸。在卷积运算方面，基于混合矩阵运算的卷积计算在新架构上相对于原有的基于快速傅里叶变换的计算方法在大规模输入时提升明显，且精度更高；然而在输入规模较小时，使用纹理内存进行直接计算占绝对优势。考虑到目前许多神经网络中的卷积计算的图像规模多为 100-1000 数量级，在该数量级上使用纹理内存进行直接计算的方法性能较强，实际应用中应考虑这种方法。以上三种大多是在网络训练阶段涉及的计算，而在网络推理阶段，本文尝试了一种新的模式，即使用 TensorRT 对训练好的网络结构进行优化并在目标硬件上进行推理。尽管 TensorRT 目前仅能运行于特定硬件，但是根据本文的实验，使用 TensorRT 能为网络推理带来极大的吞吐量提升。

在完成基准测试之后，本文移步基于 CUDA 源码构建的机器学习应用。在该部分中本文分为深度学习应用和传统机器学习应用。深度学习应用选用了结构较为简单的卷积神经网络，而传统机器学习应用选择了支持向量机。在卷积神经网络部分，本文将计算分为前向传播，反向传播更新连接参数，反向传播更新卷积核参数三个部分，分别考察新架构对于这三个部分的提升幅度。实验结果令人意外，除去前向传播中新架构能带来 30%-50% 的提升外，另两个部分中开启新架构甚至会降低性能。其原因为反向传播部分多为梯度计算，能使用混合矩阵计算 (GEMM) 从而利用到 Tensor Core 的部分较少，而开启 Tensor Core 又会对调度、同步、访存和其他指令的发射带来影响，故反向部分会造成性能下降。而前向传播部分由于卷积核、步进、填充的存在，无法保证每一层操作数的形状都能适用于 Tensor Core，故提升幅度极为有限。值得注意的是，通过将卷积操作更换为第二章中提到的纹理内存方式，总体性能得到了一定的提升。在支持向量机部分，本文则根据支持向量机输入矩阵较为稀疏的特征，分别考察了使用专为稀疏矩阵设计的 API 和使用 Tensor Core 的 API 进行评估，实验结果表明在数据量较大时，特征矩阵会愈稀疏，专为稀疏矩阵设计的 API 性能较好，而数据量较小时，仍然是使用 Tensor Core 的 API 性能较好。

之后，本文对基于 TensorFlow-GPU 框架的应用进行评估。在这一部分我们搭建了一个简单的基于 TensorFlow-GPU 的卷积神经网络，目的在于考察在最贴近真实应用场景时如何尽可能利用新硬件提升性

能。本文从神经网络的超参数、网络结构、卷积计算方式、数据精度和推理等方面进行考察；结果发现增加网络的批大小能带来显著的训练速度提升，但是过大的批会导致网络准确度下降，实际应用中应权衡这两点；而由于输入的图片尺寸较小，本文通过修改 Tensor Flow 源代码，将内建的卷积计算方式更换为使用纹理内存的直接方式后，取得了较为明显的提升且网络准确度仍然维持在较高的水平，然而由于这种方式局限大，且更改源码需要重新编译、安装，这个过程极为麻烦，故实际应用中不推荐对源码进行更改；在数据精度方面，使用 FP16/INT8 代替 FP32 并不会对网络总体准确度带来太大的下降，而在训练速度上提升很明显，实际应用中在准确度要求不高的情况下可以考虑用低精度数据替换；网络结构方面，将卷积核大小改为适合 Tensor Core 计算的形状能给训练速度带来一定提升，但是会极大降低网络准确度，实际应用中不推荐使用。最后，本文使用 TensorRT 对训练好的模型进行推理，在延迟方面有 40

通过以上实验，可以总结出新架构的确能在特定情况下为机器学习中大量存在的矩阵混合运算带来明显的性能提升，从而提升总体机器学习应用的性能，然而目前为止，硬件仍然对输入、结构等较为敏感。且有些情况下仍然有性能更高的传统方式。所以在实际编码时，应根据问题规模、算法、结构、数据分布等方面合适选择不同方法，而不是一味使用新硬件提供的方法。

最后，根据实验中发现的一些问题，本文做出了合理地设想，如计算规模更大、跨越多个线程束执行混合矩阵运算的新指令，以单个线程为粒度的同步机制等；这些设想是否会应验，或者在一定程度上实现，只能交给时间去判断，这里仅仅提出我们的设想供启发。

参考文献

[1] NVIDIA. NVIDIA TESLA V100 TENSOR CORE GPU[A]. 2019.

[2] NVIDIA. NVIDIA TENSOR CORES, The Next Generation of Deep Learning[A]. 2019.

[3] NVIDIA. NVLINK FABRIC[A]. Advancing Multi-GPU Processing. 2019.

[4] KERR A, MERRILL D, DEMOUTH J, et al. CUTLASS: Fast Linear Algebra in CUDA C++[A]. 2019.

[5] NVIDIA. NVIDIA TESLA V100 GPU ARCHITECTURE[R]. [S.l.]: NVIDIA Corp., 2017: 14-15.

[6] KURTH T, TREICHLER S, ROMERO J, et al. Exascale Deep Learning for Climate Analytics[C]. in: Super Computing Conference. [S.l. : s.n.], 2018.

[7] RAIHAN M A, GOLI N, AAMODT T M. Modeling Deep Learning Accelerator Enabled GPUs[J/OL]. CoRR, 2018, abs/1811.08309. http://arxiv.org/abs/1811.08309.

[8] MIKI Y. Gravitational octree code performance evaluation on Volta GPU[J/OL]. CoRR, 2018, abs/1811.02761. http://arxiv.org/abs/1811.02761.

[9] NVIDIA. JETSON NANO, Bringing the Power of Modern AI to Millions of Devices[A]. 2019.

[10] NVIDIA. NVIDIA CUDA-X AI, NVIDIA GPU-Acceleration Libraries for Data Science and AI[A]. 2019.

[11] NVIDIA. CUDA Zone[A]. 2019.

[12] KHAIRY M, JAIN A, AAMODT T M, et al. Exploring Modern GPU Memory System Design Challenges through Accurate Modeling[J/OL]. CoRR, 2018, abs/1810.07269. http://arxiv.org/abs/1810.07269.

[13] HARRIS M. An Even Easier Introduction to CUDA[A]. 2017.

[14] COOK S. CUDA Programming: A Developer's Guide to Parallel Computing with GPUs[M]. [S.l.]: Morgan Kaufmann, 2012: 99-102.

[15] HAKURA Z S, GUPTA A. The Design and Analysis of a Cache Architecture for Texture Mapping[C]. in: Proceedings of the 24th International Symposium on Computer Architecture, Denver, Colorado, USA, June 2-4, 1997. [S.l. : s.n.], 1997: 108-120.

[16] FLYNN M J. Some Computer Organizations and Their Effectiveness[J]. IEEE Trans. Computers, 1972, 21(9): 948-960. DOI: 10.1109/TC.1972.5009071.

[17] NVIDIA. Ampere Block Diagram[A]. 2019.

[18] SARKAR S, MITRA S. A Profile Guided Approach to Optimize Branch Divergence While Transforming Applications for GPUs[C]. in: Proceedings of the 8th India Software Engineering Conference, ISEC 2015, Bangalore, India, February 18-20, 2015. [S.l. : s.n.], 2015: 176-185. DOI: 10.1145/2723742.2723760.

[19] TIRUMALA A, GIROUS O, NELSON P, et al. Threads-are threads Functional Description, SM Branch ISA and Convergence Barrier Unit[A]. 2015.

[20] LI H, KUMAR A, TU Y. Performance modeling in CUDA streams - A means for high-throughput data processing[C]. in: 2014 IEEE International Conference on Big Data, Big Data 2014, Washington, DC, USA, October 27-30, 2014. [S.l. : s.n.], 2014: 301-310. DOI: 10.1109/BigData.2014.7004245.

[21] STEINKRAU D, SIMARD P Y, BUCK I. Using GPUs for Machine Learning Algorithms[C]. in: Eighth International Conference on Document Analysis and Recognition (ICDAR 2005), 29 August - 1 September 2005, Seoul, Korea. [S.l. : s.n.], 2005: 1115-1119. DOI: 10.1109/ICDAR.2005.251.

[22] KOMAROV I, DASHTI A, D'SOUZA R. Fast k-NNG construction with GPU-based quick multi-select[J]. CoRR, 2013, abs/1309.5478.

[23] KEERTHI S S, SHEVADE S K, BHATTACHARYYA C, et al. Improvements to Platt's SMO Algorithm for SVM Classifier Design[J]. Neural Computation, 2001, 13(3): 637-649. DOI: 10.1162/089976601300014493.

[24] VASIMUDDIN M, CHOCKALINGAM S P, ALURU S. A Parallel Algorithm for Bayesian Network Inference Using Arithmetic Circuits[C]. in: 2018 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2018, Vancouver, BC, Canada, May 21-25, 2018. [S.l. : s.n.], 2018: 34-43. DOI: 10.1109/IPDPS.2018.00014.

[25] 邦彦 福. 位置ずれに影響されないパターン認識機構の神経回路モデル — ネオコグニトロン —[J]. 電子情報通信学会論文誌 A, 1979, J62-A(10): 658-665.

[26] BENGIO Y, LECUN Y, HENDERSON D. Globally Trained Handwritten Word Recognizer Using Spatial Representation, Convolutional Neural Networks, and Hidden Markov Models[C]. in: Advances in Neural Information Processing Systems 6, [7th NIPS Conference, Denver, Colorado, USA, 1993]. [S.l. : s.n.], 1993: 937-944.

[27] FARHAT N H. Photonic Neural Networks and Learning Machines[J]. IEEE Expert, 1992, 7(5): 63-72. DOI: 10.1109/64.163674.

[28] BAKHODA A, YUAN G L, FUNG W W L, et al. Analyzing CUDA workloads using a detailed GPU simulator[C]. in: IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS 2009, April 26-28, 2009, Boston, Massachusetts, USA, Proceedings. [S.l. : s.n.], 2009: 163-174. DOI: 10.1109/ISPASS.2009.4919648.

[29] KHAIRY M, JAIN A, AAMODT T M, et al. Exploring Modern GPU Memory System Design Challenges through Accurate Modeling[J]. CoRR, 2018, abs/1810.07269.

[30] NVIDIA. NVIDIA Nsight Systems[A]. 2019.

[31] NARANG S, BAIDU. DeepBench[A]. 2016.

[32] HE K, ZHANG X, REN S, et al. Deep Residual Learning for Image Recognition[C]. in: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016. [S.l. : s.n.], 2016: 770-778. DOI: 10.1109/CVPR.2016.90.

[33] HOWARD A G, ZHU M, CHEN B, et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications[J]. CoRR, 2017, abs/1704.04861.

[34] NVIDIA. Parallel Thread Execution ISA Version 6.4. 2019.

[35] NVIDIA. CUDA Toolkit Documentation v9.2.148[A]. 2018.

致谢

距离 2015 年 09 月，已经过去了将近四年。

四年前刚刚踏入大学校园的种种仿佛还就在眼前。回忆起曾经高中学业的紧张、择校选专业时的忐忑，再看现在未来已经基本定型的情况，不由心生感慨：能找到自己真正的兴趣并在目前的实习、之后的进一步学习和工作中予以实践，着实是一件幸运的事。

首先，感谢家人的陪伴，感谢父母对于我学业无论是在经济上还是在生活上全力的支持，没有你们的帮助，我甚至没有能力承担学业。有时我会学习到深夜，在大家都已经进入梦乡的时间，你们还会为我煮上一碗面。正是因为你们，我才能完成论文、完成学业。

在大学的前几年，我也迷茫过，在别的同学做项目、打比赛的时候我也焦虑过；但是幸好我被给予了一个延续我高中以来的梦想：研究硬件、系统结构的机会，钱老师的并行计算课程、魏老师的计算机组成课程、王老师的嵌入式原理课程、肖老师的算法课程、陆老师的 C++ 课程、等等……这些课程可谓是为我打开了一扇大门，我得以系统地学习我曾感兴趣的知识；也正是因为这门课程，我也确定了本文的主题，确定了研究生的学习方向，确定了目前的实习，以及将来的职业目标。

说到职业，这里不得不提到在英伟达 (Nvidia) 实习时，公司以及同事对我的帮助，正是因为这份实习，让我有机会接触到无数的涉及 GPU 底层架构的文档，让我有机会深入到 GPU 级别的汇编代码进行编程，这些经验、资料对本文的写作带来了极大的帮助，当然，这一切都归功于 Edward 先生愿意给予我这次实习机会，并悉心指导我。

在论文的撰写中，我还得到了许多同学的协助；有同样对并行计算感兴趣的吕同学与我耐心的探讨，有姚同学给我提出的建议，有沈同学与我分享行业最新信息，还有各位一起娱乐的群友们为我带来的欢乐与放松……这些无一不让我在紧张的论文撰写中得以卸下一些压力。当然，不只是大学中的同学们，这里也感谢我自初中以来的同学，也是我的女友的 Vega 姜小姐九年以来的陪伴以及在身心上给予我的支持。

论文总有一天会完成上交，学生生涯总有一天会迎来结束。然而对新知识的探求正是支撑起我们计算机学子前进的基石。不求对世界做出什么改变，不求对人类做出什么贡献，只求在未来的道路里不忘初心、坚守道德、尽力而为、劳逸结合、保持童心、有始有终、乐观对待、做自己想做的事，并且无憾一生。

Arrivederci.