Parallel Programming (English)

Lecture Content and Study Guide

作者：Weifeng Liu

组织：Department of Computer Science and Technology, College of Information Science and Engineering,
China University of Petroleum-Beijing

时间：Spring semester of 2021-2022, April 19-June 10

版本：2.00

Teaching Assistant：*Wenhao Dai*

Latex template：*ElegantLATEX Program*

目录

1 First Week (2021.04.19 15:25 - 17:00)

Lecture Content 1

1.1 Sections One & Two (Introduction of the Course) 1

1.1.1 Part One: Introduction to what are computing and supercomputing, and why we need to learn parallel programming, about 60 minutes 1

1.1.2 Part Two: Introduction to self-study content in the next two weeks (Basic knowledge of parallel processors and OpenMP programming model), about 15 minutes. 1

1.1.3 Part Three: Introduction to this week's homework, namely homework 1 (dense general matrix-matrix multiplication, and a performance comparison with OpenBLAS) and homework 2 (quick sort and merge sort, and a performance comparison with qsort()), about 15 minutes 4

2 Second Week (2022.04.26 15:25 - 17:00)

Lecture Content 6

2.1 Section One (In-Class Test) 6

2.2 Section Two (Introduction to Basic GPU Computing, and the final project) . . . 6

2.2.1 Part One: Introduction to self-study content in the next week (Basic knowledge of GPU computing and CUDA programming model), about 5 minutes. 6

2.2.2 Part Two: Introduction to this week's homework, namely homework 3 (dense general matrix-matrix multiplication using CUDA, and a performance comparison with cuBLAS), about 5 minutes 10

2.2.3 Part Three: Introduction to the final project - GraphChallenge, about 15 minutes 12

2.2.4 Part Four: Introduction to computational thinking for CUDA parallel programming, about 20 minutes 12

3 Third Week (2022.05.03 15:25 - 17:00)

Lecture Content 13

4 Fourth Week (2022.05.10 15:25 - 17:00)

Lecture Content 14

4.1 Section One (In-Class Test) 14

4.2 Section Two (Introduction to Parallel Sparse Matrix Computations on GPUs) . 14

4.2.1 Part One: Introduction to self-study content in the next week (advanced optimizations of GPU computing and CUDA programming model), about 5 minutes. 14
4.2.2 Part Two: Introduction to this week's homework, namely homework 4 (sparse matrix-multiple vector multiplication (SpMM) using CUDA, and a performance comparison with cuSPARSE), about 5 minutes 17
4.2.3 Part Three: Introduction to sparse matrix, its use in neural networks, the CSR storage format and the parallelization of sparse matrix algorithms, about 35 minutes 19

第一章 First Week (2021.04.19 15:25 - 17:00) Lecture Content

1.1 Sections One & Two (Introduction of the Course)

The two 45-minute classes will use multiple kinds of materials for introducing high performance computing and its recent development in China and all over the world. How the course will be teaching will be introduced as well. There are three parts:

(1) Part One: Introduce computing, supercomputing and parallel programming; (about 60 minutes)

(2) Part Two: Introduce the self-study content in the next week; (about 15 minutes)

(3) Part Three: Introduce the homework in the next two weeks; (about 15 minutes)

1.1.1 Part One: Introduction to what are computing and supercomputing, and why we need to learn parallel programming, about 60 minutes

Firstly, the requirement of computing and supercomputing technologies is introduced. Several video clips and some performance numbers are given. Also, from the perspective of the history of microprocessor development, three questions are used to discuss why it is necessary to learn parallel programming. This part uses slides to explain.

1.1.2 Part Two: Introduction to self-study content in the next two weeks (Basic knowledge of parallel processors and OpenMP programming model), about 15 minutes.

In the next two weeks, students will need to study some of the following two courses at Bilibili by yourself, with a total length of about 106 minutes and about 336 minutes, respectively.

Bilibili course 1. 新竹清华大学：并行计算与并行编程课程（周志远教授，2018 年秋季学期） https://www.bilibili.com/video/BV1Yt411W7td

（1）Lecture 1C：什么是并行计算（21 分 59 秒）https://www.bilibili.com/video/BV1Yt411W7td?p=3

00:00-08:28, 并行计算的定义

08:38-15:58, 并行计算和分布式计算的区别

16:00-21:58, 为什么需要并行计算

（2）Lecture 1D：为什么我们需要并行计算？（23 分 19 秒）https://www.bilibili.com/video/BV1Yt411W7td?p=4

00:00-12:57, 为什么需要并行计算

12:57-23:18, 并行计算的趋势

（3）Lecture 2A：并行计算机与并行模型的分类（23 分 00 秒）https://www.bilibili.com/video/BV1Yt411W7td?p=5

00:00-14:50, Flynn 的四个经典分类

14:50-23:00, 共享内存和分布式内存

（4）Lecture 3A：“超级计算机” 的概念应如何界定？（16 分 22 秒）https://www.bilibili.com/video/BV1Yt411W7td?p=7

00:00-09:11, 超级计算机的定义以及衡量性能的方式

09:11-11:50, HPL Benchmark

11:50-16:00, 为什么要使用 HPL benchmark?

（5）Lecture 3B：超级计算机的效能（21 分 03 秒）https://www.bilibili.com/video/BV1Yt411W7td?p=8

00:00-06:25, 超算为什么可以算这么快?

06:25-13:25, TOP500 排名

13:25-21:02, TOP500 发展趋势

（6）Lecture 8C：Shared-Memory Programming: OpenMP（23 分 58 秒）https://www.bilibili.com/video/BV1Yt411W7td?p=23

00:00-04:39, OpenMP 编程模型

04:39-08:15, OpenMP 编程例子

08:15-12:26, OpenMP 指导语句

12:26-23:57, OpenMP 线程设置

Bilibili course 2. UC Berkeley CS267 Applications of Parallel Computers (Kathy Yelick et al., Spring 2018)https://www.bilibili.com/video/BV1qV411q7RS

（1）Lecture 1 Introduction（1 小时 24 分 28 秒）https://www.bilibili.com/video/BV1qV411q7RS?p=1

09:56-17:00, Overview

17:52-37:05, Some of the World’s Fastest Computer

    17:52-30:45, Top500 list

    30:35-37:05, figures of computer development

37:05-55:25, Science using High Performance Computing (science examples)

55:25-72:04, Why the Fastest (all since 2005) Computers are Parallel Computers

    55:25-56:50, Limitations

    56:50-72:04, Moore’s Low

72:04-81:25, Measuring Performance

    72:04-73:32, Runtime

    73:32-77:20, Speedup (Amdahl’s Law)

    77:20-81:25, parallel efficiency

81;25-84:27, What’s in this course

（2）**Lecture 2 Single Processor Machines Memory Hierarchy and Processor Features**（1 小时 21 分 34 秒）https://www.bilibili.com/video/BV1qV411q7RS?p=2

07:00-17:36, Costs in modern processors (Idealized models and actual costs)

17:36-50:45, Memory hierarchies

17:36-27:20, Temporal and spatial locality

27:20-40:14, Basics of caches

40:14-50:18, Use of microbenchmarks to characterized performance

50:18-63:10, Parallelism within single processors

51:18-53:59, Pipelining

55:49-63:10, SIMD units

63:10-80:10, Case study: Matrix Multiplication

63:21-67:40, Use of performance models to understand performance

68:56-71:50, Simple cache model

71:50-74:50, Matrix-vector multiplication

74:19-80:10, Naïve vs optimized Matrix-Matrix Multiply

（3）**Lecture 3 Optimizing Matrix Multiply (cont.), Introduction to Data Parallelism**（1 小时 23 分 25 秒）https://www.bilibili.com/video/BV1qV411q7RS?p=3

01:28-46:40, Matrix Multiplication

03:30-09:56, L2 recap: memory hierarchies and tiling

09:56-30:28, Cache Oblivious algorithms

30:28-41:05, Practical guide to optimizations

41:06-41:47, Beyond $O(n^3)$ matrix multiply

41:47-45:21, Basic Linear Algebra Subprogram (BLAS)

48:46-82:40, Introduction to parallel machines and programming

49:46-54:30, Parallel Machines and Programming

54:30-82:20, Data parallel

（4）**Lecture 4 Shared Memory Parallelism**（1 小时 26 分 47 秒）https://www.bilibili.com/video/BV1qV411q7RS?p=4

10:38-20:05, Shared memory parallelism with threads

20:05-25:42, What and why OpenMP

25:42-30:07, Parallel programming with OpenMP

30:07-85:30, Introduction to OpenMP

30:07-32:44, Creating parallelism

32:44-45:14, Caching

45:14-49:39, Synchronizing

51:52-66:20, Parallel Loops

66:20-73:24, Data sharing

73:24-85:30, Tasks

1.1.3 Part Three: Introduction to this week's homework, namely homework 1 (dense general matrix-matrix multiplication, and a performance comparison with OpenBLAS) and homework 2 (quick sort and merge sort, and a performance comparison with qsort()), about 15 minutes

Configure the Programming Environment

It is recommended to install a Linux system, which usually comes with gcc. If one wants to use Microsoft Windows, MinGW (`https://sourceforge.net/projects/mingw`) is generally needed to manually install. After the installation is successful, gcc and OpenMP should be ready for programming homework of this course.

Homework 1. Todo

(1) Run the given serial code of general matrix-matrix multiplication (GEMM). Try to exchange the order of the loops, try to add OpenMP directive `#pragma omp parallel for` to different loops to parallelize them, verify the correctness of the results, test performance, and write them in the report;

(2) Install OpenBLAS. Test the given code that calls GEMM in OpenBLAS, compare the performance with your own parallel code, and write it in the report.

Homework 1. Report Requirements

(1) Test requirements: multiply two square matrices (a matrix with the same number of rows and columns) A and B to get $C = AB$, and the number of rows and columns are both 100-2000 (take a point every 100 intervals), a total of 20 sets of test data, record the running time of each set;

(2) The test method includes at least: (a) the original serial code, (b) several kinds of serial codes with the order of the loops exchanged, (c) the parallel code of adding OpenMP instruction statements on different for loops, (d) the parallel code of OpenBLAS, (e) Original GEMM code optimized in any way;

(3) Submission deadline: April 29, 2022. Submit the report (student ID number + name.pdf) to your teaching assistant.

Homework 2. Todo

(1) Run the given quicksort and mergesort serial codes, find the opportunity to use OpenMP task parallel, add OpenMP instruction statement `#pragma omp task` to parallelize it, and verify the result correctness, test performance, write them in the report;

(2) Call the qsort() function. Test the code of the given qsort() function in the C standard library, compare the performance with your own parallel code, and write it in the report.

Homework 2. Report Requirements

(1) Test the OpenMP parallel algorithms of quicksort and mergesort under the following conditions. The sorting result is required to be in the ascending order, the calculation result should be correct, and the lengths of the input data are 10^1, 10^2, 10^3, 10^4, 10^5, 10^6, 10^7, 10^8,

and a total of eight sets of tests Data, record the running time of each set;

(2) The test method includes at least: (a) running time under different threads, (b) memory space consumption under different threads, (c) performance under different input data, using three different input data: (I) random number sequence, (II) ascending sequence, (III) descending sequence, (d) performance of qsort() code, (e) original sorting code optimized in any way;

(3) Submission deadline: April 29, 2022. Submit the report (student ID number + name.pdf) to your teaching assistant.

Unified requirements for English homework reports

(1) Use Latex to write (Note: do not use Microsoft Word), and submit the pdf file. It is recommended to write homework on `https://www.overleaf.com/`;

(2) Use Python Matlibplot to draw your figures;

(3) The hardware environment of the machine (CPU model, number of cores, frequency, memory capacity, etc.) should be described in the report;

(4) The report language must be English.

第二章 Second Week (2022.04.26 15:25 - 17:00) Lecture Content

2.1 Section One (In-Class Test)

This 45-minute in-class test will give ten questions to evaluate the study effect of the last one week. The quiz will contain the knowledge of three parts:

(1) the MOOC teaching videos at Bilibili given in the first week,

(2) the homeworks about parallel programming using OpenMP.

2.2 Section Two (Introduction to Basic GPU Computing, and the final project)

This 45-minute class consists of three parts:

(1) Part One: Introduce the self-study content in the next week; (about 5 minutes)

(2) Part Two: Introduce the homework for the third week; (about 5 minutes)

(3) Part Three: Introduce the final project. (about 15 minutes)

(3) Part Four: Introduce CUDA programming. (about 20 minutes)

2.2.1 Part One: Introduction to self-study content in the next week (Basic knowledge of GPU computing and CUDA programming model), about 5 minutes.

In the next week, students will need to study some of the following three courses at Bilibili by yourself, with a total length of about 177 minutes, 97 minutes and 70 minutes, respectively.

Bilibili course 3. UIUC（伊利诺伊大学香槟分校）：NVIDIA CUDA 高度并行处理器编程课程（Wen-mei Hwu（胡文美）教授，2008 年） https://www.bilibili.com/video/BV1tC4y1H7QG

（1）Introduction and Motivation（31 分 42 秒） https://www.bilibili.com/video/BV1tC4y1H7QG?p=1

00:00-02:57 CUDA 背景资料

02:58-04:05 CPU、GPU 设计理念差别

04:06-05:30 Control 部分的差别

05:31-06:59 Cache 部分的差别

07:00-07:58 GPU 发展趋势

07:59-08:45 并行编程在 2008 年前没有大获成功的原因

08:46-09:47 1. killer micros（研发出并行后已经有更快的串行处理器）

09:48-11:57 2. 网络延迟导致并行变慢

11:58-12:57 3. 并行广泛使用

12:58-14:24 4. 浮点运算

14:25-16:31 GPU 计算的扩展性

16:32-19:33 加速的好处、降低误差、提高准确度

19:34-20:24 GPGPU 简介

20:25-21:57 GPGPU 的限制

21:58-23:42 GPU 擅长数据并行处理

23:43-26:00 CUDA：CPU 和 GPU 协作

26:01-27:25 应用并行

27:26-31:42 应用并行加速

（2）CUDA 编程模型（1 小时 26 分 25 秒）https://www.bilibili.com/video/BV1tC4y1H7QG?p=2

00:00-00:56 Overview

00:57-05:17 CUDA

05:18-11:27 CUDA Devices and Threads

11:28-13:33 G80 - Graphics Mode

13:34-15:12 G80 CUDA mode

15:13-22:24 CUDA language - C extended

22:25-25:14 Thread

25:15-28:30 Thread Block

28:31-30:34 Block ID、Thread ID

30:35-32:32 CUDA Memory —Global Memory

32:33-34:26 CUDA API

34:27-35:22 CUDA 内存分配—cudaMalloc()

35:23-36:47 CUDA 内存销毁—cudaFree()

36:48-38:19 内存分配代码举例

38:20-39:50 CUDA 数据传输

39:51-41:49 数据传输代码举例

41:50-46:30 CUDA 函数声明

46:31-49:44 __device__ 函数

49:45-54:35 线程创建代码示例

54:36-56:21 矩阵乘法举例

56:22-60:07 矩阵乘法 C 语言中的简单主机版本（CPU）

60:08-62:07 输入矩阵数据传输（GPU）

62:08-63:21 输出矩阵数据传输（GPU）

63:22-68:55 Kernel 函数

68:56-70:35 Kernel 调用

70:36-72:26 矩阵计算所需线程

72:27-73:13 处理任意矩阵思路

73:14-76:07 CUDA 程序编译过程和 NVCC 编译器

76:08-76:38 CUDA 链接库

76:39-78:51 Debugging 两种方法

78:52-80:36 仿真模式的陷阱

80:37-86:25 浮点计算的陷阱

（3）CUDA 存储（1 小时 00 分 29 秒） https://www.bilibili.com/video/BV1tC4y1H7QG?p=3

00:00-07:24 CUDA 的变量类型限定符的区别

07:25-08:59 CUDA 的变量类型限定符应该在哪儿定义

09:00-11:09 CUDA 存储器的实现原理和 share memory（共享内存）在其中的作用

11:10-13:29 一个 CUDA 通用编程策略介绍

13:30-15:14 Pointers 变量在 kernel 中使用的限制

15:15-19:09 GPU 中的原子整数操作

19:10-20:14 共享内存的矩阵乘法介绍

20:15-24:44 G80 的运算过程以及 G80 的性能

24:45-28:19 如何使用共享内存来重用内存数据

28:20-31:03 分块矩阵乘法的过程

31:04-34:04 分块矩阵乘法的优点

34:05-35:04 CUDA 内核代码执行前的配置

35:05-36:34 CUDA 内核代码介绍

36:35-41:34 共享内存中的数据初始化

41:35-44:39 CUDA 内核代码运算结果介绍

44:40-45:18 CUDA 内核代码运算结果如何保存

45:19-46:49 共享内存的矩阵乘法

46:50-60:29 CUDA 程序的典型结构总结

Bilibili course 1. 新竹清华大学：并行计算与并行编程课程（周志远教授，2018 年秋季学期） https://www.bilibili.com/video/BV1Yt411W7td

（1）Lecture 13B：Data Partitioning/Introduction to GPU（10 分 36 秒） https://www.bilibili.com/video/BV1Yt411W7td?p=38

11:11-13:12 介绍视频的内容分配，异构计算和 GPU 介绍。

13:13-14:45 介绍 CPU 扩展的末路，性能越来越难提升。

14:46-17:48 介绍并行计算机的趋势，从单核时代到多核时代再带分布式系统时代，异构系统时代。

17:49-18:52 介绍异构计算，异构计算是由不同类型的计算单元组成的集成系统。

18:53-20:16 介绍计算范式的转变，从 cpu 到 gpu。

20:17-21:46 介绍 GPU/Xeon Phi 进入世界前 500，成为世界最快的超级计算机。

（2）Lecture 14A：Hardware for Heterogeneous Computing: CPU & GPU （15 分 06 秒） https://www.bilibili.com/video/BV1Yt411W7td?p=39

09:08-10:42 回顾上节课的异构计算的定义和 GPU/Xeon Phi。

10:43-12:53 介绍 GPU 服务器，与商用服务器相同的硬件架构，但 CPU 和 GPU 之间的内存复制成为主要瓶颈。

12:54-16:05 介绍异构系统架构，旨在提供一种通用的系统架构，以便为所有设备设计更高级别的编程模型。

16:06-18:47 介绍 AMD 加速处理器（APU），旨在在单个芯片上融合 CPU 和 GPU。

18:48-24:13 介绍 GPU，图形处理器，专为快速显示和可视化而设计的专用芯片。

（3）Lecture 14B：Heterogeneous System Architecture （26 分 53 秒） https://www.bilibili.com/video/BV1Yt411W7td?p=40

00:00-03:16 介绍 GPGPU，专门用来做计算的 GPU。

03:17-09:19 介绍系统架构，GPU 和 CPU 用 PCI-E 方式连接。

09:20-17:19 介绍 GPU 框架图，由多个流多处理器组成，内存架构从全局内存到共享内存，再到本地寄存器。

17:20-20:29 介绍流多处理器，每个流多处理器是向量机，共享寄存器文件，缓存可编程，硬件调度用于线程执行和硬件上下文切换。

20:30-26:52 介绍支持 NVIDIA cuda 的 GPU 产品，有 tegra x2，geforce 等。

（4）Lecture 14C：Specification and Capability of GPU （20 分 10 秒） https://www.bilibili.com/video/BV1Yt411W7td?p=41

00:00-01:26 回顾上节课中的 GPU 框架图和支持 NVIDIA cuda 的 GPU 产品。

01:27-10:22 介绍 NVIDIA GPU 的硬件规格，包括 tesla k40，tesla p100，tesla v100，geforce gtx 1080 的架构比较。

10:23-12:54 介绍 NVIDIA GPU 体系结构路线图，从 2008 的 1.0 到 2018 的 7.0 计算能力的发展过程。

12:55-15:26 介绍 GPU 计算能力，即 GPU 设备的编程能力

15:27-17:09 介绍 CUDA SDK 设备查询，使用 deviceQuery 指令查询。

17:10-20:09 介绍 cuda 工具包，包括编译器 nvcc，工具 IDE 等，BLAS 库等，和一些例子代码以及文献。还有几个 CUDA SDK 版本的计算能力和架构介绍。

（5）Lecture 14D：What is Compute Unified Device Architecture (CUDA) （25 分 18 秒） https://www.bilibili.com/video/BV1Yt411W7td?p=42

00:00-02:48 介绍章节内容安排，包括编程模型，CUDA 语言，示例代码研究，cpu 和 gpu 同步，多 GPU。

02:49-04:50 介绍什么是 CUDA，CUDA 是计算统一设备架构，NVIDIA GPU 编程的编译器和工具，支持 GPU 的异构计算和强大功能，cuda api 扩展了 C/C ++ 编程语言，表达 SIMD 并行性，提供硬件的高级抽象。

04:51-08:17 介绍 CUDA 程序流程，主机 CPU 存储器将数据复制到设备 GPU 存储器，

设备 GPU 存储器将结果复制到主机 CPU 存储器，主机 CPU 调用 GPU 函数，设备 GPU 执行程序。

08:18-11:40 介绍 CUDA 编程模型，CUDA 是具有并行 kernel 的串行程序，串行 C 代码在主机线程中执行，并行内核 C 代码在跨多个处理单元的许多设备线程中执行。

11:41-14:17 介绍 cuda 编程框架，gpu 代码以 __global__ void my_kernel() 形式定义，cpu 代码部分以 my_kernel< < < gridsize，blocksize> > > 形式调用。

14:18-19:02 介绍内核 kernel，内核是许多并发线程，一次在设备中执行一个内核，许多线程执行每个内核，cuda 线程可能是物理线程，如 NVIDIA 线程，也可能是虚拟线程。

19:03-23:36 介绍并发线程的层次结构，线程被分组为线程块，内核是线程块网格，同一块中的线程可以同步，但不能在块之间同步。

23:37-25:17 介绍软件映射情况，软件是从线程带集合到块再到线程，硬件是从 GPU 到多核处理器再到核。

Bilibili course 2. UC Berkeley CS267 Applications of Parallel Computers (Kathy Yelick et al., Spring 2018)https://www.bilibili.com/video/BV1qV411q7RS

（1）**Lecture 8 An Introduction to CUDA/OpenCL and Graphics Processors (GPUs)**（1 小时 10 分 09 秒）https://www.bilibili.com/video/BV1qV411q7RS?p=9

01:30-03:14, Comparison of CPU and GPU

03:15-05:20, Development of processors

05:21-06:29, Latency and Bandwidth

06:30-09:32, Structure of CPU and GPU

09:33-11:42, CPU to GPGPUs

11:43-15:48, Thread divergence

15:49-16:30, Summary of Latency and Throughput

16:33-20:00, SIMD and SIMT

20:01-29:17, Hardware of CUDA programming models

29:18-59:40, Introduction to CUDA programming

59:41-61:13, Introduction to OpenCL

61:14-62:33, Imperatives for efficient CUDA code and profiling

62:33-70:09, What’s next

2.2.2 Part Two: Introduction to this week’s homework, namely homework 3 (dense general matrix-matrix multiplication using CUDA, and a performance comparison with cuBLAS), about 5 minutes

Configure the GPU Programming Environment

It is recommended to install a CUDA SDK (including NVIDIA GPU driver) in a Linux system. Virtual Machine should not work for running CUDA code, though the system has an NVIDIA GPU. After the installation is successful, CUDA should be ready for programming homework of this course.

Homework 3. Todo

(1) Run the given basic CUDA code of general matrix-matrix multiplication (GEMM) using CUDA global memory. Try to compare the code and performance difference between CUDA code and the OpenMP code in homework 1, verify the correctness of the results, test performance (note that cudaDeviceSynchronize() should be called before timing function, and a kernel call should run 10 times, and the average execution time should be recorded), and write them in the report;

(2) Run the given optimized CUDA GEMM code using shared memory. Try to compare the performance difference between the optimized CUDA code and the basic CUDA code in the above step, verify the correctness of the results, test and compare performance, and write them in the report;

(3) Install cuBLAS (should be already there if CUDA SDK is installed correctly). Test the given code that calls GEMM in cuBLAS, compare the performance with the above two versions of the CUDA GEMM code (using global memory and shared memory, respectively), and write the performance comparison in the report.

Homework 3. Report Requirements

(1) Test requirements: multiply two square matrices (a matrix with the same number of rows and columns) A and B to get $C = AB$, and the number of rows and columns are both 100-2000 (take a point every 100 intervals), a total of 20 sets of test data, record the running time of each set;

(2) The test method includes at least: (a) your OpenMP GEMM code, (b) the parallel GEMM code of OpenBLAS, (c) the basic CUDA GEMM code using global memory, (d) the optimized CUDA GEMM code using shared memory, (e) the GEMM code using cuBLAS, (f) the CUDA GEMM code optimized by yourself with your own techniques (if any);

(3) Submission deadline: May 6, 2022. Submit the report (student ID number + name.pdf) to your teaching assistant.

Unified requirements for English homework reports

(1) Use Latex to write (Note: do not use Microsoft Word), and submit the pdf file. It is recommended to write homework on `https://www.overleaf.com/`;

(2) Use Python Matlibplot to draw figures;

(3) The hardware environment of the machine (CPU model, number of cores, frequency, memory capacity, etc. and GPU model, number of cores, frequency, memory capacity, etc.) should be described in the report;

(4) The report language must be English.

2.2.3 Part Three: Introduction to the final project - GraphChallenge, about 15 minutes

GraphChallenge is organized by MIT and Amazon, it encourages community approaches to developing new solutions for analyzing graphs and sparse data derived from social media, sensor feeds, and scientific data to enable relationships between events to be discovered as they unfold in the field.

Sparse Deep Neural Network Graph Challenge is the newest challenge published in 2019. This challenge performs neural network inference on a variety of sparse deep neural networks.

You can find slides, paper, example serial code, and example data sets from `https://graphchallenge.mit.edu/challenges`.

Our final project is a more specific version of the *Sparse Deep Neural Network Graph Challenge*. The dataset and serial code will be introduced.

Requirements of the project

- Work individually, ask your TA for technical supports;
- Use sparsity of the matrices, and MPI+CUDA to program;
- Optimizations (data structure and algorithm design, sparsifying, MPI, CUDA, report writing, etc.) should be carefully considered;
- All tests will be done on our 4-node 32-GPU cluster;
- Hand in your code and a final report describing your algorithm and performance.

Deadlines of the project

- 1st Deadline is Week 5
    - Use slides to show your plan and your progress
    - All students should come to the stage to present
    - No more than 1.5 minutes
- 2nd Deadline is Week 8
    - Use slides to show your final performance results
    - All students should come to the stage to show your achievements
    - No more than 1.5 minutes
    - Hand in your code and a final report

2.2.4 Part Four: Introduction to computational thinking for CUDA parallel programming, about 20 minutes

From the perspective of the Single Program Multiple Data (SPMD), CUDA programming will be introduced with vector addition and GEMM examples. Students will be encouraged to think parallel. This part uses PPT and black board to explain, and teacher and students will interact to build a better study experience.

第 三 章 Third Week (2022.05.03 15:25 - 17:00)

Lecture Content

This class is removed because of the May Day holiday.

第四章 Fourth Week (2022.05.10 15:25 - 17:00) Lecture Content

4.1 Section One (In-Class Test)

This 45-minute in-class test will give questions to evaluate the study effect of the last week. The quiz will contain the knowledge of two parts:

(1) the MOOC teaching videos at Bilibili given in the last week,

(2) the third homework about parallel GEMM using CUDA global memory and shared memory, respectively.

4.2 Section Two (Introduction to Parallel Sparse Matrix Computations on GPUs)

This 45-minute class consists of three parts:

(1) Part One: Introduce the self-study content in the next week; (about 5 minutes)

(2) Part Two: Introduce the homework in this week; (about 5 minutes)

(3) Part Three: The teacher teaches the introduction to sparse matrix using CUDA programming. (about 35 minutes)

4.2.1 Part One: Introduction to self-study content in the next week (advanced optimizations of GPU computing and CUDA programming model), about 5 minutes.

In the next week, students will need to study some of the following three courses at Bilibili by themselves, with a total length of about 187 minutes, 82 minutes and 55 minutes, respectively.

Bilibili course 3. UIUC（伊利诺伊大学香槟分校）：NVIDIA CUDA 高度并行处理器编程课程（Wen-mei Hwu（胡文美）教授，2008 年） https://www.bilibili.com/video/BV1tC4y1H7QG

（1）CUDA 线程（1 小时 26 分 25 秒） https://www.bilibili.com/video/BV1tC4y1H7QG?p=4

00:00-07:47 回顾讲解 threads

07:48-12:30 Thread ID

12:31-16:08 CUDA Thread Block 回顾

16:09-20:09 块的可扩展性

20:10-30:17 G80 结构

30:18-47:37 G80 线程调度

47:38-51:31 G80 矩阵选择注意事项

51:32-57:29 G80 共享内存和线程的互动

57:30-64:17 不同 Tiling Size 的速度影响

64:18-66:13 API

66:14-70:00 dim3

70:01-72:07 数学函数

72:08-74:54 G80 SFU 部分

74:55-76:52 主机运行时组件

76:53-86:25 设备运行组件：同步功能

（2）性能的协调（性能调优，Performance Tuning）（1 小时 40 分 34 秒）https://www.bilibili.com/video/BV1tC4y1H7QG?p=5

00:00-07:54 本节目的

07:55-10:31 线程块划分方式

10:32-17:42 Control Flow 指令

17:42-22:29 并行中的 Reduction

22:30-24:37 in-place reduction

24:38-28:37 Reduction 示例代码

28:38-31:31 线程图解

31:32-34:19 示例代码缺陷

34:20-36:13 改进代码

36:14-38:44 改进代码线程图解

38:45-47:55 GPU Memory Coalescing

47:56-50:35 通过 Tile 实现垂直方法

50:36-53:30 垂直更优的原因

53:31-53:52 总结

53:53-55:07 不同 Tiling Size 的速度影响

55:08-58:29 简化问题思路

58:30-69:32 Register File

69:35-73:29 矩阵乘法 register 使用示例

73:30-76:20 动态分割

76:21-78:05 指令层并行和线程层并行区别

78:06-78:37 资源分配示例

78:38-81:52 指令组合注意事项

81:53-85:20 共享 thread

85:21-87:20 Prefetch 减少等待时间

87:21-91:16 prefetch 性能提升

91:17-92:12 关于矩阵乘更优性能的想法

92:13-93:28 unrolling 版本和 tiling 版本代码对比

93:29-100:34 G80 的主要性能下降因素

Bibilili course 1. 新竹清华大学：并行计算与并行编程课程（周志远教授，2018 年秋季学期） https://www.bilibili.com/video/BV1Yt411W7td

（1）**Lecture 15A：Hierarchy of Concurrent Threads （21 分 56 秒）** https://www.bilibili.com/video/BV1Yt411W7td?p=43

00:00-09:55 回顾上次课中的 cuda 编程框架、并发线程的层次结构和软件映射方式。

09:56-12:17 介绍块级调度，块彼此独立以提供可扩展性，kernel 通过将块调度到多处理器来扩展任何数量的并行内核。

12:18-15:48 线程级调度 warp，在多处理器内部，线程以 32 组为一组启动，称为 warp。warp 共享控制部分，其中的线程将执行相同的指令。

15:49-17:00 介绍线程组的限制，使用 deviceQuery 来查询限制，包括线程块的最大值，最大执行并发等。

17:01-21:55 介绍内存层次，每个线程和其本地内存同级，块和每个块的共享内存同级，内核和每个设备的全局内存同级。

（2）**Lecture 15B：Programming Terminology of CUDA（25 分 49 秒）** https://www.bilibili.com/video/BV1Yt411W7td?p=44

00:00-02:53 介绍内存大小限制，使用 deviceQuery 来查询限制。以计算能力 6.1 的 GTX1080 为例，全局内存总量为 11171MB，常量内存总量为 64MB，每块的共享内存总量为 48MB，每块可用的寄存器总数为 65536 个。

02:54-03:52 介绍 cuda 的编程术语，Host 是 cpu，Device 是 gpu，Kernel 是在 GPU 上执行的功能，Thread 是基本执行单元，Block 是一组线程，Grid 是一组块。

03:53-10:02 介绍一些相关的问题与解答，如一个内核可以跨多个多处理器运行等等。

10:03-16:26 介绍 CUDA 语言

16:27-19:16 介绍线程和块 id

19:17-21:31 介绍如何在某内核启动设置下对数组的 100 个元素进行相加

21:32-23:55 介绍功能限定词，__device__ function 在只能从设备调用的设备上执行，__global__ function 在只能从主机调用的设备上执行，__host__ function 在只能从主机调用的主机上执行，Functions without qualifiers 仅为主机编译，__host__ __device__ function 为主机和设备编译。

23:56-25:48 介绍变量类型限定词，__device__ var 驻留在设备的全局内存空间中，有申请的剩余时间，__constant__ var 可从网格中的所有线程以及通过运行时库从主机访问，驻留在设备的常量存储空间中，__shared__ var 驻留在线程块的共享内存空间中，具有块的寿命，仅可从块内的线程访问。

（3）**Lecture 16A：Foundation of CUDA language（20 分 28 秒）** https://www.bilibili.com/video/BV1Yt411W7td?p=45

00:00-05:35 回顾上节课的 CUDA 语言、功能限定词、变量类型限定词等。

05:36-09:43 介绍设备内存操作。

09:44-11:55 介绍 cudaMemcopy 的四个可能的值，cudaMemcpyHostToHost 是从 host 到 host，cudaMemcpyHostToDevice 是从 host 到 device，cudaMemcpyDeviceToHost 是从 device 到 host，cudaMemcpyDeviceToDevice 是从 device 到 device。

11:56-17:01 介绍程序编译，包含 cuda 语言的任何源文件都必须使用 nvcc 进行编译，nvcc 将主机上运行的代码与设备上运行的代码分开。

17:02-18:00 例题 1 hello world

18:01-20:27 例题 2 两数相加的错误示范

（4）Lecture 16B：Pseudocode for CUDA （15 分 41 秒） https://www.bilibili.com/video/BV1Yt411W7td?p=46

00:00-04:08 例题 2 两数相加的正确示范

04:09-11:20 例题 3 向量相加，展示了 3 种使用不同线程块和线程配置的方法。

11:21-13:00 介绍 blockdim.x 使用的一般情况。

13:01-15:41 介绍 blockdim.x 使用的另一个例子，以及 kernel 的分支。

Bilibili course 2. UC Berkeley CS267 Applications of Parallel Computers (Kathy Yelick et al., Spring 2018)https://www.bilibili.com/video/BV1qV411q7RS

（1）Sparse-Matrix-vector-Multiplication and Iterative Solve（55 分 44 秒）https://www.bilibili.com/video/BV1qV411q7RS?p=16

10:10-28:30, Sparse matrix formats and basic SpMV

28:30-52:25, Register blocking and autotuning SpMV

52:25-59:05, SpMV on multicore

59:05-65:54, Distributed memory SpMV

4.2.2 Part Two: Introduction to this week's homework, namely homework 4 (sparse matrix-multiple vector multiplication (SpMM) using CUDA, and a performance comparison with cuSPARSE), about 5 minutes

Configure the GPU Programming Environment

It is recommended to install a CUDA SDK (including NVIDIA GPU driver) in a Linux system. Virtual Machine should not work for running CUDA code, though the system has an NVIDIA GPU. After the installation is successful, CUDA should be ready for programming homework of this course.

Homework 4. Todo

(1) Multiply a square matrix A of size $m \times m$ and a tall-and-skinny matrix B of size $m \times n$ (m should be much larger than n) to get $C = AB$, and run the given basic code of sparse matrix-multiple vector multiplication (SpMM) using the CSR format for A. Try to compare the performance difference of

(a) a serial GEMM code, where A is dense,

(b) an OpenMP GEMM code, where A is dense,

(c) a global memory version of CUDA GEMM code, where A is dense,

(d) a shared memory version of CUDA GEMM code, where A is dense,

(e) a cuBLAS version of CUDA GEMM code, where A is dense,

(f) a serial SpMM code, where A is sparse and in the CSR format, and B is dense,

(g) an OpenMP SpMM code, where A is sparse and in the CSR format, and B is dense,

(h) a scalar-version of CUDA SpMM code, where A is sparse and in the CSR format, and B is dense,

(i) a vector-version of CUDA SpMM code, where A is sparse and in the CSR format, and B is dense,

(j) a cuSPARSE (should be already there if CUDA SDK is installed correctly) of CUDA SpMM code, where A is sparse and in the CSR format, and B is dense.

Verify the correctness of the results, test performance (note that cudaDeviceSynchronize() should be called before timing function, and a kernel call should run 200 times, and the average execution time should be recorded), and write a performance comparison in the report;

Homework 4. Report Requirements

(1) Test requirements: m (the order of A) should be between 100 and 2000 (take a point every 100 intervals, a total of 20 sets of test data), n (the number of column of B) should be 16, 32 and 64, record the running time of each parameter combination (so for each method, $60 = 20 \times 3$ results should be recorded). A matplotlib heatmap (see `https://matplotlib.org/stable/gallery/images_contours_and_fields/image_annotated_heatmap.html`) should be use to show the results of each method. Thus in total 10 heatmaps should show up in the final report.

(2) You are encouraged to write your own optimized SpMM code using CUDA. If you do so, add yours as the 11th set of performance data;

(3) Submission deadline: May 20, 2022. Submit the report (student ID number + name.pdf) to your teaching assistant.

Unified requirements for English homework reports

(1) Use Latex to write (Note: do not use Microsoft Word), and submit the pdf file. It is recommended to write homework on `https://www.overleaf.com/`;

(2) Use Python Matlibplot to draw figures (note: this time you should draw heatmap);

(3) The hardware environment of the machine (CPU model, number of cores, frequency, memory capacity, etc. and GPU model, number of cores, frequency, memory capacity, etc.) should be described in the report;

(4) The report language must be English.

4.2.3 Part Three: Introduction to sparse matrix, its use in neural networks, the CSR storage format and the parallelization of sparse matrix algorithms, about 35 minutes

From the perspective of the sparse neural networks, sparse matrix and its CSR format will be introduced. Also, some performance considerations of using CUDA for sparse problems will be listed. This part uses PPT and black board to explain, and teacher and students will interact to build a better study experience.