Test #2

截止时间 6月12日 23:59 **得分** 100 **问题** 40
可用 6月8日 0:01 至 6月12日 23:59 5 天 **时间限制** 60 分钟

说明

This is called a "quiz" on Canvas, but it is really 100-point Test #2.

There is a time limit of 60 minutes on this test. Once you start, you must finish. Canvas will not let you pause and come back later.

This test is Open Notes, Closed Friends, and Closed Internet.

This is your last graded-anything in this class, so do a good job!

Thanks for a *great* quarter!

尝试历史记录

	尝试	时间	分数
最新	尝试 1	34 分钟	100，满分 100 分

! 正确答案将于 6月13日 0:01 提供。

此测验的分数： **100**，满分 100 分
提交时间 6月9日 19:15
此尝试进行了 34 分钟。

问题 1

2.5 / 2.5 分

Function calls on GPU hardware:

○ Happen exactly the same way as CPU hardware implements them

● End up being inlined because there is no stack to store arguments and return addresses

○ Happen through the special “GPU-stack”, which is the same as a CPU-stack

○ Happen through the special “GPU-stack”, which is different from a CPU-stack

问题 2

2.5 / 2.5 分

In Project #3, the Functional Decomposition project, each individual quantity’s function needed to have three barriers. The *first* barrier was there to:

○ Indicate when the Watcher thread could print values

○ Indicate when it was time to increment the month

○ Indicate when that quantity’s next value was done being copied to the global state

● Indicate when that quantity's function was done computing that quantity’s next value

问题 3

2.5 / 2.5 分

MPI Reductions:

○ Are unnecessary because of the number of CPUs

● Are a built-in feature of the MPI API

○ Must be implemented by your application

○ Are unnecessary because of the SIMD units on the CPUs

问题 4

2.5 / 2.5 分

What is the advantage of a Fused-Multiply-Add?

○ It implies that a SIMD operation should be performed

○ You only have to write one line of code instead of two

● It can perform a multiply plus an add in about the same time as it could have done the multiply alone

○ It reduces the possibility of False Sharing

问题 5

2.5 / 2.5 分

Let’s beat the Yellow Robot metaphor to death. The yellow robot’s grippers represent:

● Processing Elements

○ SIMD banks

○ Separate CPU cores

○ Compute Units

问题 6

2.5 / 2.5 分

In MPI, a "derived type" is:

○ Being able to pack multiple MPI_INTs into a single MPI_LONG

○ Being able to pack multiple MPI_CHARs into a single MPI_LONG

○ Linking multiple MPI calls together to send both MPI_FLOATs and MPI_INTs

● You creating a struct that can act just like MPI_FLOAT, MPI_INT, etc.

问题 7

2.5 / 2.5 分

Projects #1 and #5 ran roughly the same code on a CPU and a GPU, respectively. What can you say about their relative performance in Trials/Second?

● The GPU version was way faster

○ The CPU version was way faster

○ Within 10%, the two versions had about the same performance

问题 8

2.5 / 2.5 分

The OpenCL function clCreateFromGLBuffer( ):

○ Allocates an OpenCL device memory buffer

○ Creates an OpenGL graphics vertex buffer object

○ Deletes an OpenCL buffer and replaces it with an OpenGL-compatible vertex buffer object

● Creates an OpenCL device memory pointer from an OpenGL graphics vertex buffer object

问题 9

2.5 / 2.5 分

In Project #3, the Functional Decomposition project, each individual quantity’s function needed to have three barriers. The *third* barrier was there to:

● Indicate when the Watcher thread was done printing values

○ Indicate when that quantity's function was done computing that quantity’s next value

○ Indicate when that quantity’s next value was done being copied to the global state

○ Indicate when it was time to increment the time of day

问题 10

2.5 / 2.5 分

In the CUDA call:

cudaMemcpy(**A,** **B,** NUM_ELEMENTS*sizeof(float), cudaMemcpyHostToDevice); **:**

○ The CPU array A gets copied to the GPU array B

● The CPU array B gets copied to the GPU array A

○ The GPU array B gets copied to the CPU array A

○ The GPU array A gets copied to the CPU array B

问题 11

2.5 / 2.5 分

In your CUDA program, how do you show that a function is the GPU Kernel function?

○ By labeling it with __kernel__

● By labeling it with __global__

○ By labeling it with __local__

○ By labeling it with __device__

问题 12

2.5 / 2.5 分

In running your GPU program on the DGX server, you needed to type, for example:

○ ./montecarlo.cu

● sbatch montecarlo.bash

○ slurm montecarlo.csh

○ ./montecarlo

问题 13

2.5 / 2.5 分

In Project #3, the Functional Decomposition project, each individual quantity's function needed to have three barriers. The *second* barrier was there to:

● Indicate when that quantity's next value was done being copied to the global state

○ Indicate when that quantity's function was done computing that quantity's next value

○ Indicate when it was time to increment the month

○ Indicate when the Watcher thread could print values

问题 14

2.5 / 2.5 分

As of the writing of our class notes, the 2022 IEEE Visualization conference:

● Will be held in Oklahoma City, OK

○ Has been cancelled

○ Will be online-only

○ Will be held in Cambridge, MA

问题 15

2.5 / 2.5 分

A Sphere can be represented as four floats. What are they?

○ The STP of the texture coordinates and the radius

○ The four hyperbolic radii

○ XYZ of the surface normal and the radius

● XYZ of the center position and the radius

问题 16

2.5 / 2.5 分

Let’s beat the Yellow Robot metaphor to death. The yellow robot represents:

● A Compute Unit

○ A SIMD bank

○ A separate CPU core

○ A Processing Element

问题 17

2.5 / 2.5 分

GPU Reductions:

○ Are unnecessary because of the GPU speed

● Must be implemented by the .cl function you write

○ Are unnecessary because of the GPU hardware instruction set

○ Are a built-in feature of the OpenCL API just like they are in OpenMP

问题 18

2.5 / 2.5 分

As of the writing of our class notes, the 2022 SC conference (International Conference for High Performance Computing, Networking, Storage, and Analysis) will be held:

○ In Washington, DC

○ Totally online

● In Dallas, TX

○ In Los Angeles, CA

问题 19

2.5 / 2.5 分

When is it OK to use the less-precise “fast_normalize( )” call instead of the full-precision “normalize( )” call?

○ When using OpenCL for scientific computing

○ Always

● When using OpenCL for computer graphics

○ Never

问题 20

2.5 / 2.5 分

In your C/C++ CUDA program, how do you show that you are making a call to the GPU Kernel function?

○ With the cudaEnqueueNDRangeKernel( ) function

○ With the cudaExecuteKernel( ) function

○ With the >>> … <<< (chevron) syntax

● With the <<< … >>> (chevron) syntax

问题 21

2.5 / 2.5 分

Joe Parallel wants to use OpenCL kernels to implement the graph execution structure shown here. How?

○ He sets up barriers at C and D

○ He turns C and D into special OpenCL reduction functions

● He has A, B, and C each throw events, and has C, C, and D (respectively) wait for those events.

○ He can't -- it is not possible in the current version of OpenCL, but might be in the future

问题 22

2.5 / 2.5 分

In Project #1, you performed a multicore Monte Carlo simulation by using the NUMTRIALS for-loop. In Project #5, you re-created that same simulation using CUDA without any for-loop. Where did that NUMTRIALS for-loop go?

○ It is still there – it has just been replaced with the special CUDA “foreach” capability

● It is not needed – it has been replaced by duplicating the simulation onto thousands of threads

○ You don't need to include it– CUDA is smart enough to figure out what you are trying to do and adds it for you

○ It is still there – it has just been written in CUDA-code instead of C/C++

问题 23

2.5 / 2.5 分

In MPI, a computer’s “rank” is:

○ Its processing power

○ Its priority

● Its integer identifier

○ The number of cores it has

问题 24

2.5 / 2.5 分

MPI follows what parallel programming model?

○ Single Instruction, Multiple Data (SIMD)

○ Single Instruction, Single Data (SISD)

○ Multiple Instructions, Multiple Data (MIMD)

● Single Program, Multiple Data (SPMD)

问题 25

2.5 / 2.5 分

In the OpenCL call “gid = get_global_id(0)”, what does the argument of 0 indicate?

○ Relative to the first element of the dataset

● In the X dimension

○ Since the time at which the program started

○ That you want only one value returned

问题 26

2.5 / 2.5 分

Jane Parallel uses this line of OpenCL code:

status = clEnqueueNDRangeKernel(cmdQueue, kernel, 1, NULL, A, B, C, D, E);

what are the C and D variables used for?

○ The context to use

● They specify how many events to wait for and which ones they are

○ The globalWorkSize and the localWorkSize

○ They specify what event to throw when this kernel is completed

问题 27

2.5 / 2.5 分

Why did Jane Parallel use those typedefs (point, vector, color, sphere) in her OpenCL code?

○ The compiler requires it

○ Those were indeed the real OpenCL names for those types of variables

● It makes it more obvious what her code is doing

○ The OpenCL standard requires it

问题 28

2.5 / 2.5 分

A "CUDA Core":

○ Consists of multiple cores

○ Consists of flow control alone

● Consists of an integer unit and a floating-point unit

○ Consists of flow control, plus an integer unit and a floating-point unit

问题 29

2.5 / 2.5 分

Comparing CPUs and GPUs, it is correct to say:

● CPUs are better with linked-list data structures, GPUs are better with data parallel arrays

○ CPUs are better with integers, GPUs are better with floating-point

○ GPUs are better with integers, CPUs are better with floating-point

○ GPUs are better with linked-list data structures, CPUs are better with data parallel arrays

问题 30

2.5 / 2.5 分

An MPI “Broadcast” operation involves:

○ Many functions: multiple broadcast senders and a single broadcast receiver

○ Many functions: a broadcast sender and one unique broadcast receiver function per CPU

○ Two functions: a broadcast sender and a broadcast receiver

● A single function regardless of if you are sending or receiving

问题 31

2.5 / 2.5 分

What does the OpenCL call “gid = get_global_id(0)” return?

○ It tells you how big the global dataset is

○ It tells you how big the local dataset is

● It tells you where you are in the global dataset

○ It tells you where you are in the local dataset

问题 32

2.5 / 2.5 分

One of the ways that CUDA differs from OpenCL is:

○ In OpenCL, the C/C++ and GPU code are placed in the same file

○ CUDA GPU code looks like C and OpenCL GPU code looks like Python

● In CUDA, the C/C++ and GPU code are placed in the same file

○ CUDA GPU code looks like Python and OpenCL GPU code looks like C

问题 33

2.5 / 2.5 分

The advantage of using SSE SIMD is:

○ It uses a stack variable to hold each multiplication product

○ It uses the stack to hold its pointers

○ You can perform a multiply and an add in one instruction

● You should be able to get a 4x performance improvement in array multiplication

问题 34

2.5 / 2.5 分

What is special about using OpenCL/OpenGL interoperability?

○ The Khronos Group gives you a certificate for doing it

○ It saves electrical power

○ It allows GPU graphics to be driven by CPU multicore computing

● The data never leaves GPU memory

问题 35

2.5 / 2.5 分

What does the function cudaMalloc( ) do?

○ Allocates space in CPU memory

○ Allocates space in both CPU and GPU memory

● Allocates space in GPU memory

○ Pre-allocates space in the GPU cache

问题 36

2.5 / 2.5 分

In MPI, the phrase “scatter/gather” means:

○ To use the MPI_Bcast( ) function to get information to all other CPUs

○ To gather input data from the disk and then scatter it out to different CPUs for computation

● To break a problem up into pieces, give each piece to a separate computers, and then gather up the results

○ To setup Barriers across all CPUs

问题 37

2.5 / 2.5 分

In this class, the letters “MPI” stand for:

○ Many-Processor Interfaces

● Message Passing Interface

○ MegaCalculation Per Instruction

○ Millions of Processor Instructions

问题 38

2.5 / 2.5 分

In CUDA, how many threads are in each Warp?

● 32

○ 16

○ 128

○ 64

问题 39

2.5 / 2.5 分

Jane Parallel uses this line of OpenCL code:

status = clEnqueueNDRangeKernel(cmdQueue, kernel, 1, NULL, A, B, C, D, E);

what is the E variable used for?

○ It specifies how many events to wait for

○ The globalWorkSize

○ The context to use

● It specifies what event to throw when this kernel is completed

问题 40

2.5 / 2.5 分

When OpenCL and OpenGL work together:

○ It is just OpenGL that is able to access the vertex buffer

○ It is just OpenCL that is able to access the vertex buffer

● They take turns accessing the vertex buffer

○ They both access the vertex buffer at the same time

测验分数： **100**，满分 100 分

▶