Marco Seman

Professor Jie Shen

December 2, 2018

Assignment #4

Machine Type: Windows 8

Interpreter Type: Visual Studios 2017

Table of Contents

[**Question 1 MPI (10 points)** 2](#_Toc531792686)

[**Question 2 OpenMP (10 points)** 3](#_Toc531792687)

[**Question 3 (10 points).**  6](#_Toc531792688)

[**Question 4 (10 points).** 7](#_Toc531792689)

# **Question 1 MPI (10 points)**

Write a one-page summary about MPI. You may include the following points:

1. What is MPI ?
2. When should we use MPI ?
3. What is main mechanism of MPI ?
4. What are the drawbacks of MPI ?

MPI is a Message Passing Interface Standard which is a standardized way of exchanging messages through several computers that are running on a parallel program across distributed memory. Running parallel computing through multiple processors are called nodes where each node works on a portion of the computing problem. The nodes are then coordinated by the action that each parallel node is switched between other nodes and provide command and control over the entire parallel cluster. Today, MPI is used for working on the draft of the MPI-3 standard for improving scalability, enhance performance, include multicore cluster, and cluster support. MPI is also available for C/C++ and Fortran.

Message Passing Interface should be used when a user needs to create parallel applications/computing. Parallel computing is efficient for five reasons which are: standardization, portability, performance opportunities, functionality, and availability. Standardization is MPI’s only message passing library which is standard and is supported on all HPC platforms. Portability is useful in MPI where it enables there to be very little modifications to the programmer’s source code when running it on a different platform. Performance opportunity for MPI is useful since it exploits native hardware features which optimize performance as well as it being free to develop optimized algorithms. For functionality, there are over 430 routines. And for availability in MPI, there are multiple implantations that are available, and both public and vendor domain.

For the mechanisms of MPI, it provides support of development of modular programs and the sequential and parallel composition of program components. A communicator is one of MPI’s main mechanisms for establishing communication “universes”. The communication mechanism also is used to distribute the application development which offers multiple primitives from point to point as well as process control such as startup and shutdown. In communication, it is important that all computers are communicating at high performances through multithreading programming models and message passing interface to achieve high levels of parallelism and scalability in applications.

There are a few drawbacks with it comes to Message Passing Interface. The first disadvantage is that the programmer is responsible to provide data in another processor, explicitly design how data communicates, and to synchronize between tasks. Second disadvantage is that there is no compilation time optimization, therefor it allows sophisticated analysis allowing in-lining. The third drawback is that it blocks sends and receives which the blocking model suggests a mismatch between the user’s model and the MPI model of parallel computing.

# **Question 2 OpenMP (10 points)**

Write an OpenMP program as follows:

// Marco Seman

// Professor Jie Shen

// Assignment 4 Question 2

// December 12, 2018

//

#include "stdafx.h"

#include "iostream"

#include <time.h>

using namespace std;

int main()

{

// n: tested using goto statement

int thread\_count, i;

/\* ask users to input an n as the length of arrays \*/

int n;

cout << "Enter Array Size: ";

cin >> n;

int \*arr = new int[n];

/\* dynamically allocate three one-dimension float arrays x, y and z with the length of n \*/

float\* x = NULL;

x = new float[n];

float\* y = NULL;

y = new float[n];

float\* z = NULL;

z = new float[n];

cout << "Input the number of threads to be used: ";

cin >> thread\_count;

clock\_t start, end;

float duration, sum\_time;

start = clock();

#pragma omp parallel for num\_threads(thread\_count)

for (i = 0; i<n; i++)

{

x[i] = i\*i;

y[i] = exp((double)x[i]);

z[i] = y[i] + sin(3.14);

}

end = clock();

duration = (double)(end - start) / CLOCKS\_PER\_SEC; // duration in seconds

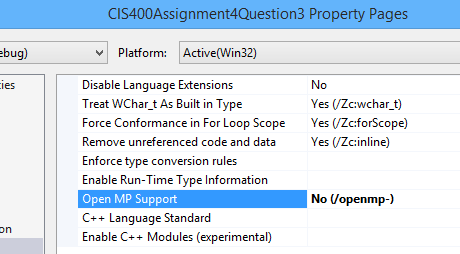
cout << "Duration = " << duration << " (sec) " << endl << endl;

//goto n; used this to test

system("pause");

return 0;}

1. In Microsoft Visual Studio, turn off the pre-compiled header flag and turn on the flag for openmp under project property🡪 C/C++ 🡪 Language 🡪 Open MP Support.



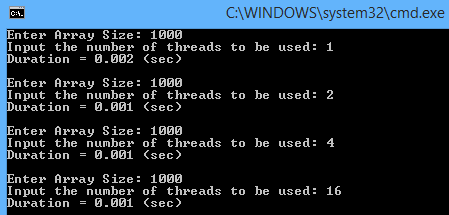
1. Compile the Open MP project and grab the output as a screenshot. Insert the screenshot into the MS word document for this assignment.

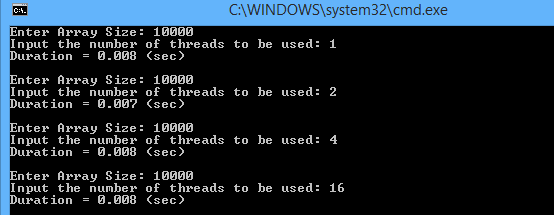
c) Use time function as follows:

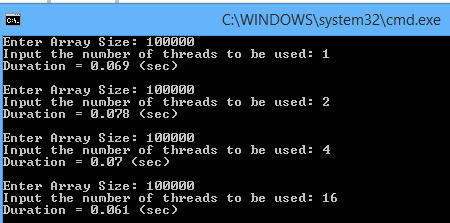
Record the execution time for the Open MP problem on one machine with the following different setting:

Number of threads: 1, 2, 4, 16

Number of elements: 1000, 10000, 100000







Draw a bar graph to show the difference in the execution time with respect to the number of threads in the cases of each *n*. Give a short discussion about what you found.

­

In your document, you need to present the hardware information of your test machine, including machine type, CPU type, the number of cores, CPU clock rates, and graphics card.

IDE: Microsoft Visual Studio

Machine Type: HP Sleekbook TouchSmart 15-b153nr

CPU Type: AMD A8-4555M

Number of Cores: Quad Core

Clock Speed: 1.60 GHz

Graphics Card: API with Radeon™ HD Graphics

Operating System: 64 Bit Windows 8.1

RAM: 6 GB

# **Question 3 (10 points).** Write a one-page summary about CUDA programming framework. You may include the following points:

1. What is CUDA programming framework ?
2. Who developed it ?
3. What are the main features for GPGPU ?
4. What is the purpose of \_\_global\_\_ specifier ?
5. What is the purpose of \_\_device\_\_ specifier ?
6. Explain blocks and threads for CUDA kernel calls

Cuda programing framework is a programming model and parallel computing platform that was developed by Nvidia, so they can compute their own graphics processing units. CUDA allows software developers and software engineers to use GPU enabled general purpose processing to increase the speed of compiled intensive applications by harnessing the power of GPU for the parallelizable part of computation.

Cuda was developed and designed by Ian Buck and his team of researchers in 2003. The platform was the first widely adopted programming model that extended C with data parallel constructs. Nvidia hired Ian Buck to lead the laugh of CUDA in 2006 which was the first commercial solution for general purpose computing on GPUs.

GPUPU contains a variety of features for its programmable concepts. One of its features is programmable processors which allows the programmer to run a kernel on streams of data. The programmable processors are also vertex, primitive, and computable pipelines. Rasterizer is another feature that interpolates vertex that fragments the coordination of the computed color scheme. Texture unit is used as a read only memory interface and Framebuffer is used as a write only memory interface.

The purpose of \_\_Global\_\_ functions is to call functions from the host and execute them in the device. \_\_Global\_\_ also runs on the GPU which is called from the CPI where it is then executed using the <<<dim3>>> arguments.

The purpose of \_\_Device\_\_ function is that is runs on the GPU and called from the GPU so it can be used with variables too. In other words, \_\_device\_\_ can be called from the device only where it is then executed by the device only. The unlike global, device function cannot be called from the host code unless you declare \_\_host\_\_ \_\_device\_\_.

A block and thread for CUDA is part of the operating system and programming abstraction that represents a group of threads that can be executed serially or in parallel. Parallel computing is a CUDA platform that enables the ability for higher level languages to use exploit parallelism. The treads in CUDA are an abstract entity enabling execution of a kernel which is a small program or function. For most graphics cards, 512 is the limit per block. If the problem is bigger than 512, we must use a combination of blocks and threads.

The benefit of using thread blocks in CUDA is that it guarantees that the hardware can run efficiently because it has so much flexibility. For example, if one thread block is completed quickly, the stream processors can immediately another thread block without waiting for any other blocks to complete. Scalability is the biggest advantage of thread blocks since it makes no guarantees where it can be run or how many thread blocks can be run, meaning that the user can scale all the way down to a GPU that will be running the single stream processor up the the massive GPU used in super computers.

# **Question 4 (10 points).** Write a one-page summary about RUST programming language. You may include the following points:

1. What is RUST programming languages ?
2. Who developed it ?
3. What is the concept of ownership ?
4. What is the concept of borrowing ?
5. What is the concept of lifetime ?
6. What are the main benefits of using this language ?

Rust is a system programming language with a focus on safety, especially thread safety and memory safety. Rust supports both functional and imperative paradigms. Rust Syntactically follows C# and was designed by Grandon Hoore at Mozilla research in 2010 and was ranked the first place for most loved programming language by stack overflow. The website for rust is: [www.rust-lang.org](http://www.rust-lang.org).

The central feature in Rust is ownership. Ownership’s key idea is that there is exactly one binding to any given memory allocated in heap. The stack stores values in order it gets them in and removes the last value in the opposite order, in other words, last one in, first one out. Ownership is fast because of the way it accesses data. The heap is used to store data with a size that is unknown at compile time since it is less organized and therefore slower.

Borrowing is when we do not take over the ownership, the ownership is only borrowed using pass by reference. There is a type of reference called &mut T which is known as a mutable reference that allows you to mutate the resource you are borrowing. There are two rules when it comes to borrowing, the first rule is that you may only have one or more reference (&T) to a reference. The second rule is that you are only allowed to have one mutable reference (&Mut T).

Lifetime is a constructor in the compiler which ensures that all the borrowed values are valid. The lifetime of a variable is started from when it is created, then the variable will end once it is destroyed. A borrow feature has a lifetime that is determined when a variable is declared. The borrow is valid until it is destroyed by the lender. At execution time, the compiler must know the lifetime of the value that the pointer refers to which is the reason why lifetime parameters exists in Rust.

There are a variety of benefits when it comes to using Rust in modern systems. The first main benefit of using Rust is its memory safety. Memory safety is an important factor since it prevents errors that cause security vulnerabilities through RAM access. Dangling pointers and data race are some memory errors that occur in programming languages. Dangling pointers do not occur in Rust due to the special type of pointers called smart pointers. Smart pointers free up memory allocations when the data is out of cope which avoids the formation of dangling pointers that de-allocate memory.

Another benefit of Rust is that it is compatable with Windows, Mac, and Linux. For windows, it contains precompiled binary installers that are available on the Rust website. When running a program on rust and it is successfully compiled, Rust will automatically generate a binary executable. Cargo is another main benefit since it involves multiple functionalities such as: building code, download required dependencies, and builds the dependencies.

**Submission of Your Work:**

(1) a zipped file that contains a) source code, and an MS word document.

(2) The Word document should contain the following information

* Your name
* Machine type
* Interpreter type
* Description of your code design and implementation
* Inclusion of your source
* Test case design and screen shots