 **TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**KHOA CÔNG NGHỆ THÔNG TIN**

**======\*\*\*======**

Ảnh có chứa văn bản, biểu tượng, Đồ họa, ảnh chụp màn hình

Mô tả được tạo tự động

**BÁO CÁO BÀI TẬP LỚN**

**HỌC PHẦN: TÍNH TOÁN HIỆU NĂNG CAO**

*Mã lớp: 20241IT6069001*

**ĐỀ TÀI: GIỚI THIỆU VỀ OPENCL**

**GVHD: TS. Hà Mạnh Đào**

**Nhóm sinh viên : 6**

1. **Phạm Văn Hiếu Mã SV: 2022602204 Lớp: KHMT01**
2. **Nguyễn Việt Anh Mã SV: 2022601605 Lớp: KHMT01**
3. **Lê Huy Hậu Mã SV: 2022600351 Lớp: KHMT01**

***Hà Nội - 2024***

# 

# MỤC LỤC

[DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT](#_Toc185111620)

[DANH MỤC CÁC BẢNG](#_Toc185111621)

[DANH MỤC CÁC HÌNH VẼ, ĐỒ THỊ](#_Toc185111622)

[MỞ ĐẦU 1](#_Toc185111623)

[1. Lý do chọn đề tài 1](#_Toc185111624)

[2. Mục tiêu nghiên cứu 1](#_Toc185111625)

[3. Phạm vi và giới hạn 2](#_Toc185111626)

[4. Những kiến thức và kĩ năng cần đạt được 2](#_Toc185111627)

[5. Yêu cầu về đầu ra sản phẩm nghiên cứu 2](#_Toc185111628)

[6. Các công cụ được sử dụng để làm sản phẩm 2](#_Toc185111629)

[7. Cấu trúc của báo cáo 3](#_Toc185111630)

[CHƯƠNG I: TỔNG QUAN 4](#_Toc185111631)

[1.1. Giới thiệu về HPC (High Performance Computing) 4](#_Toc185111632)

[1.2. Giới thiệu về OpenCL 4](#_Toc185111633)

[1.3. Kiến thức cần biết trước khi bắt đầu sử dụng OpenCL 5](#_Toc185111634)

[1.4. Các khái niệm cơ bản trong OpenCL 5](#_Toc185111635)

[1.4.1. Platform 6](#_Toc185111636)

[1.4.2. Device 6](#_Toc185111637)

[1.4.3. Kernel 6](#_Toc185111638)

[1.4.4. Context 7](#_Toc185111639)

[1.4.5. Command Queue 7](#_Toc185111640)

[1.4.6. Buffer 7](#_Toc185111641)

[1.4.7. Event 8](#_Toc185111642)

[1.4.8. Program 8](#_Toc185111643)

[1.5. Mô hình và cấu trúc của một chương trình OpenCL 9](#_Toc185111644)

[CHƯƠNG II: CÔNG CỤ VÀ THIẾT LẬP MÔI TRƯỜNG 10](#_Toc185111645)

[2.1. Cài đặt OpenCL 10](#_Toc185111646)

[2.2. Cài đặt và cấu hình môi trường phát triển OpenCL 15](#_Toc185111647)

[2.3 Các công cụ hỗ trợ OpenCL 21](#_Toc185111648)

[2.3.1. Platform API 21](#_Toc185111649)

[2.3.2. Runtime API 22](#_Toc185111650)

[2.3.3. Kernel Programming Language 23](#_Toc185111651)

[2.3.4. Tóm tắt các công cụ hỗ trợ OpenCL 24](#_Toc185111652)

[CHƯƠNG III: THỰC NGHIỆM 25](#_Toc185111653)

[3.1. Bài toán thực hiện phép cộng hai vector số nguyên 25](#_Toc185111654)

[CHƯƠNG IV: KẾT QUẢ 29](#_Toc185111655)

[KẾT LUẬN 29](#_Toc185111656)

[TÀI LIỆU THAM KHẢO 29](#_Toc185111657)

[PHỤ LỤC 29](#_Toc185111658)

# DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT

# DANH MỤC CÁC BẢNG

# DANH MỤC CÁC HÌNH VẼ, ĐỒ THỊ

# MỞ ĐẦU

## Lý do chọn đề tài

Hiện nay, với sự phát triển mạnh mẽ của các công nghệ mới như là Machine Learning, Deep Learning, Artificial Intelligent,... dẫn đến nhu cầu tính toán trong lĩnh vực khoa học và công nghệ ngày càng cao và trở thành một thách thức vì phương pháp xử lý tuần tự truyền thống với một bộ xử lý không thể nào đáp ứng được. Các phép tính phức tạp và đòi hỏi sự tính toán lớn luôn được tìm thấy và không ngừng tăng lên nhằm để giải quyết các bài toán trong lĩnh vực khoa học và công nghệ. Vì vậy sứ mệnh của lĩnh vực tính toán hiệu năng cao (High Performance Computing) là xây dựng và tạo ra các hệ thống máy tính mạnh mẽ để đáp ứng nhu cầu đó. Nhận thấy được tầm quan trọng đó, nhóm chúng em đã quyết định chọn đề tài “***Nghiên cứu về OpenCL***” với mong muốn nghiên cứu, tìm tòi, phát triển một chương trình sử dụng sức mạnh của GPU để tính toán song song giúp xử lý bài toán một cách nhanh chóng và dễ dàng hơn.

## ****Mục tiêu nghiên cứu****

* **Mục tiêu tổng quát**:
  + Nghiên cứu và tìm hiểu cách hoạt động của OpenCL, từ đó áp dụng vào giải quyết các bài toán cụ thể đòi hỏi tính toán song song.
* **Mục tiêu cụ thể**:
  + Tìm hiểu nguyên lý hoạt động và kiến trúc của OpenCL.
  + Làm rõ các khái niệm quan trọng trong OpenCL như kernel, buffer, program và các thành phần khác.
  + Xây dựng và triển khai một chương trình ứng dụng OpenCL để giải quyết bài toán cụ thể.
  + Đánh giá hiệu năng của OpenCL trên các thiết bị phần cứng khác nhau.

## ****Phạm vi và giới hạn****

* **Phạm vi nghiên cứu**:
  + Tập trung vào nghiên cứu và triển khai các bài toán tính toán song song đơn giản trên nền tảng OpenCL.
  + Thử nghiệm trên các loại phần cứng cụ thể như CPU, GPU từ các hãng phổ biến (Intel, AMD, NVIDIA).
  + Đánh giá hiệu năng dựa trên thời gian thực thi, tốc độ xử lý, và khả năng mở rộng.
* **Giới hạn nghiên cứu**:
  + Chỉ tập trung vào các tính năng cơ bản và ứng dụng phổ biến của OpenCL.
  + Không đi sâu vào các khía cạnh phức tạp như tối ưu hóa chuyên sâu cho từng loại phần cứng.
  + Không đề cập đến các công nghệ xử lý song song khác ngoài mục tiêu so sánh cơ bản.

## Những kiến thức và kĩ năng cần đạt được

* Kiến thức cơ bản về môn học tính toán hiệu năng cao
* Kiến thức cơ bản về OpenCL
* Kỹ năng làm việc nhóm

## Yêu cầu về đầu ra sản phẩm nghiên cứu

* Tiết kiệm thời gian, công sức về xử lý bài toán
* Tài liệu ngắn gọn, đủ ý, dễ đọc, dễ hiểu cho người mới bắt đầu sử dụng OpenCL

## Các công cụ được sử dụng để làm sản phẩm

* **Visual Studio:** 
  + Là một môi trường phát triển tích hợp (IDE) do Microsoft phát triển, được sử dụng để lập trình, gỡ lỗi và xây dựng các ứng dụng cho nhiều nền tảng như Windows, Web, Di động.
  + Hỗ trợ nhiều ngôn ngữ lập trình C++. C#, Python,..
* **OpenCL SDK:**
  + Là bộ công cụ phần mềm cung cấp thư viện, trình biên dịch và công cụ hỗ trợ để lập trình ứng dụng OpenCL, cho phép tận dụng sức mạnh tính toán song song trên các nền tảng như CPU và GPU.
  + Sản phẩm sẽ được sử dụng OpenCL SDK tiêu chuẩn của Khronos Group Inc.

## Cấu trúc của báo cáo

Ngoài phần mở đầu và kết luận, nội dung quyển Báo cáo bài tập lớn còn bao gồm 4 chương:

**Chương I: Tổng quan**

Chương II: Công cụ và thiết lập môi trường

Chương III: Thực nghiệm

**Chương IV: Kết quả**

# CHƯƠNG I: TỔNG QUAN

## ****Giới thiệu về HPC (High Performance Computing)****

* Thông thường, phần mềm được viết cho tính toán tuần tự (serial computation)

● Được chạy trên máy tính đơn với một bộ xử lý trung tâm (CPU).

● Một bài toán (problem) sẽ được chia thành một chuỗi các câu lệnh rời rạc.

● Các câu lệnh được thực hiện một cách tuần tự.

● Tại mỗi thời điểm chỉ thực hiện được một câu lệnh.

* Ý nghĩa đơn giản nhất của tính toán song song là việc sử dụng đồng thời nhiều tài nguyên máy tính để giải quyết bài toán về tính toán.

● Để chạy trên nhiều CPU.

● Một bài toán được chia thành các phần riêng biệt mà có thể được giải quyết đồng thời.

● Mỗi phần được chia nhỏ hơn dưới một dãy các câu lệnh.

● Các câu lệnh của mỗi phần thực thi đồng thời trên các CPU khác nhau

➔ Tính toán song song là một hình thức tính toán trong đó nhiều phép tính được thực hiện đồng thời, hoạt động trên nguyên tắc là những vấn đề lớn đều có thể chia thành nhiều phần nhỏ hơn, sau đó được giải quyết đồng thời ("trong lĩnh vực tính toán").

## ****Giới thiệu về OpenCL****

* **Định nghĩa**:  
  OpenCL (Open Computing Language) là một tiêu chuẩn mở cho lập trình tính toán song song trên các nền tảng đa dạng, từ CPU, GPU và các vi điều khiển.
* **Lịch sử và phát triển**:
  + Được phát triển bởi Khronos Group vào năm 2008.
  + Được hỗ trợ bởi nhiều nhà sản xuất phần cứng lớn, giúp giảm sự phụ thuộc vào công nghệ độc quyền.
* **Ưu điểm**:
  + Hỗ trợ đa nền tảng và tương thích với nhiều loại phần cứng.
  + Cho phép tối ưu hóa hiệu năng tính toán nhờ khả năng tận dụng tối đa tài nguyên phần cứng.
* **Hạn chế**:
  + Đường cong học tập dốc (Steep Learning Curve) so với các công cụ khác như CUDA.
  + Hiệu năng có thể không tối ưu bằng các giải pháp chuyên biệt trên phần cứng cụ thể.

## ****Kiến thức cần biết trước khi bắt đầu sử dụng OpenCL****

* **Ngôn ngữ lập trình**:
  + Thành thạo ngôn ngữ C/C++ vì OpenCL dựa trên C99.
  + Hiểu biết về các khái niệm cơ bản của lập trình song song (thread, synchronization, v.v.).
* **Kiến thức về phần cứng**:
  + Hiểu rõ cấu trúc CPU, GPU, và cách chúng thực hiện các tác vụ tính toán song song.
  + Biết cách làm việc với driver và SDK của phần cứng mục tiêu (ví dụ: AMD ROCm, NVIDIA OpenCL SDK).
* **Công cụ phát triển**:
  + Sử dụng các IDE hỗ trợ lập trình OpenCL như Visual Studio, Code::Blocks, hoặc các trình biên dịch dòng lệnh.

## ****Các khái niệm cơ bản trong OpenCL****

* OpenCL framework có ba thành phần chính là **Platform API, Runtime API** và **Kernel Programming Language:**
  + Platform API: Bao gồm các hàm và chức năng được sử dụng để truy xuất thông tin về các platform và device có sẵn trong hệ thống (Nvidia, Intel, AMD)
  + Runtime API: Bao gồm các hàm và chức năng được sử dụng để thực thi mã OpenCL trên device được lựa chọn bới Platform API
  + Kernel Programming Language: Là ngôn ngữ lập trình dựa trên ngôn ngữ tiêu chuẩn C99, được sử dụng để viết code cho kernel
* Các khái niệm cần biết và API thao tác với chúng:

### Platform

* Định nghĩa: Một platform trong OpenCL đại diện cho một môi trường thực thi mã OpenCL. Một platform có thể chứa nhiều device khác nhau (CPU, GPU,...)
* Một số platform API phổ biến:
  + clGetPlatformIDs(): Lấy danh sách các platform OpenCL có sẵn
  + clGetPlatformInfo(): Lấy thông tin chi tiết về một platform, như tên, phiên bản, và khả năng hỗ trợ.

### Device

* Định nghĩa: Device là phần cứng được sử dụng để thực thi mã OpenCL
* Trong một platform có thể có nhiều device
* Một số device API phổ biến:
  + **clGetDeviceIDs()**: Lấy danh sách các device có sẵn từ một platform.
  + **clGetDeviceInfo()**: Lấy thông tin chi tiết về một device, như loại, bộ nhớ, và tốc độ xử lý.

### Kernel

* Định nghĩa: Kernel là mã nguồn OpenCL chạy trên device, tương tự như một hàm trong lập trình nhưng được thiết kế để chạy đồng thời trên nhiều phần tử dữ liệu
* Một số kernel API phổ biến:
  + **clCreateKernel()**: Tạo một kernel từ một chương trình OpenCL.
  + **clSetKernelArg()**: Thiết lập tham số cho kernel.
  + **clReleaseKernel()**: Giải phóng kernel khi không còn sử dụng.

### Context

* Định nghĩa: Context là một đối tượng trong OpenCL giúp quản lý tất cả các resources (tài nguyên) cần thiết cho việc thực thi kernel. Nó bao gồm thông tin về các device, các buffer và các chương trình
* Một số context API phổ biến:
  + **clCreateContext():** Tạo một context cho các device được chỉ định.
  + **clRetainContext():** Tăng số lượng tham chiếu đến context
  + **clReleaseContext():** Giải phóng context khi không còn sử dụng.

### Command Queue

* Định nghĩa: Command queue là hàng đợi nơi người dùng gửi các lệnh để thực thi trên một device
* Mỗi queue thuộc về một context và một device cụ thể
* Một số queue API phổ biến:
  + **clCreateCommandQueueWithProperties():** Tạo một command queue cho một device.
  + **clEnqueueNDRangeKernel()**: Gửi kernel vào command queue để thực thi trên device.
  + **clFinish()**: Đợi cho tất cả các lệnh trong command queue hoàn thành.

### Buffer

* Định nghĩa: Buffer là một vùng nhớ trên device dùng để lưu trữ dữ liệu
* Nó có thể chứa dữ liệu đầu vào, dữ liệu đầu ra, hoặc bất kỳ thông tin nào khác cần thiết cho kernel.
* Một số buffer API phổ biến:
  + **clCreateBuffer():** Tạo một buffer để lưu trữ dữ liệu.
  + **clEnqueueWriteBuffer():** Ghi dữ liệu từ host vào buffer.
  + **clEnqueueReadBuffer():** Đọc dữ liệu từ buffer về host**.**
  + **clReleaseMemObject():** Giải phóng buffer khi không còn sử dụng.

### Event

* Định nghĩa: Event là một đối tượng trong OpenCL dùng để theo dõi trạng thái của các tác vụ (task) trong command queue.
* Một số event API phổ biến:
  + **clEnqueueMarkerWithWaitList()**: Đặt một dấu hiệu trong command queue để đồng bộ hóa.
  + **clWaitForEvents()**: Chờ cho một hoặc nhiều events hoàn thành.
  + **clReleaseEvent()**: Giải phóng event khi không còn sử dụng.

### Program

* Định nghĩa: Program là một đối tượng đại diện cho tập hợp mã nguồn của một hoặc nhiều kernel
* Một số program API phổ biến:
  + **clCreateProgramWithSource():** Tạo một chương trình OpenCL từ mã nguồn kernel.
  + **clBuildProgram():** Biên dịch chương trình OpenCL.
  + **clReleaseProgram():** Giải phóng chương trình khi không còn sử dụng.

## ****Mô hình và cấu trúc của một chương trình OpenCL****

**A diagram of a computer process

Description automatically generated**

*Nguồn: OpenCL Programming Book Guide*

* Tại chương trình host, ta xác định context, trong hình trên context bao gồm cả CPU và GPU
* Tiếp đó, ta xác định các Command-queues, như trong ảnh ta có 2 queues: In-order queue cho GPU và out-of-order queue cho CPU
* Chúng ta tiếp tục khai báo đối tượng Program để biên dịch và tạo ra các kernels cho các OpenCL devices (CPU và GPU)
* Khai báo các đối tượng Memory để lưu trữ dữ liệu các biến của chương trình host, sau đó các đối tượng Memory này sẽ được sử dụng làm đối số cho các kernel
* Cuối cùng, chương trình host sẽ đưa các kernel vào các queue để thực thi song song.

# CHƯƠNG II: CÔNG CỤ VÀ THIẾT LẬP MÔI TRƯỜNG

## 2.1. Cài đặt OpenCL

Với máy tính cá nhân sử dụng GPU của Nvidia, ta sẽ sử dụng OpenCL trong bộ Nvidia Computing Toolkit.

* **Kiểm tra phần cứng**
* Xác định xem máy tính có sử dụng GPU của **NVIDIA**, **AMD**, hoặc **Intel**.
* Đảm bảo rằng CPU hoặc GPU hỗ trợ OpenCL:
  + NVIDIA: OpenCL đi kèm với driver CUDA.
  + AMD: Driver hỗ trợ OpenCL có sẵn trong Adrenalin hoặc ROCm.
  + Intel: Hỗ trợ OpenCL thông qua Intel OpenCL Runtime.
* **Cài đặt driver**

**a. NVIDIA**

* Tải **CUDA Toolkit** (bao gồm driver NVIDIA hỗ trợ OpenCL).
  + Link: [NVIDIA CUDA Toolkit](https://developer.nvidia.com/cuda-downloads)
* Cài đặt driver bằng cách chạy tệp .exe.

**b. AMD**

* Tải **Adrenalin Software** cho GPU của bạn hoặc **ROCm** (nếu cần).
  + Link: [AMD Drivers](https://www.amd.com/en/support/download/drivers.html)
* Chạy tệp cài đặt và làm theo hướng dẫn.

**c. Intel**

* Tải **Intel OpenCL Runtime** hoặc **Intel oneAPI Base Toolkit**.
  + Link: [Intel oneAPI Toolkit](https://www.intel.com/content/www/us/en/developer/tools/oneapi/base-toolkit-download.html?packages=oneapi-toolkit&oneapi-toolkit-os=windows&oneapi-win=online)
* Chạy tệp cài đặt và làm theo hướng dẫn.

**Bước 1:** Tải và cài đặt phiên bản CUDA Toolkit phù hợp từ trang chủ NVIDIA. https://developer.nvidia.com/cuda-downloads A screenshot of a computer

Description automatically generated

Hình 2.1. Tải và cài đặt phiên bản CUDA Toolkit

**Bước 2:** Mở file vừa tải về để cài đặt

* + Chọn đường dẫn

A screenshot of a computer error

Description automatically generated

Hình 2.2. Chọn đường dẫn

A screenshot of a computer error

Description automatically generated

Hình 2.3. Đợi quá trình kết thúc

* + Chọn Agree and Continue

A screenshot of a computer

Description automatically generated

Hình 2.4. Đồng ý điều khoản và tiếp tục

* + Chọn Express (Cài mọi thành phần của CUDA) hoặc Custom (chọn thành phần muốn cài)

A screenshot of a computer

Description automatically generated

Hình 2.5. Chọn phương thức cài đặt

* + Chọn next và chờ tiến trình cài đặt hoàn thành

A screenshot of a computer

Description automatically generated

Hình 2.6. Chọn Next

A screenshot of a computer

Description automatically generated

Hình 2.7. Hoàn tất quá trình cài đặt

## 2.2. Cài đặt và cấu hình môi trường phát triển OpenCL

* **Cài đặt công cụ phát triển**
* **Visual Studio**: Hỗ trợ tốt nhất cho Windows.
* **VS Code**: Nhẹ và linh hoạt.
* **Code::Blocks** hoặc **Eclipse**
* **Cấu hình môi trường**

**a. Cấu hình Visual Studio**

1. **Thêm đường dẫn thư viện và tệp tiêu đề**:

Mở **Visual Studio**, tạo hoặc mở một dự án C++.A screenshot of a computer program

Description automatically generated

Hình 2.8. Tạo một dự án mới

A screenshot of a computer

Description automatically generated

Hình 2.9. Điền tên và tạo dự án

* + Vào **Project > Properties**.A screenshot of a computer program

    Description automatically generated

Hình 2.10. Chọn Properties

* + Tại **VC++ Directories**, thêm:
    - **Include Directories**: C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v12.6\include

A screenshot of a computer

Description automatically generated

Hình 2.11. Gắn link thư mục Include

* + Tại **Linker > General**, thêm:
    - **Additional library directories**:

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v12.6\lib

A screenshot of a computer

Description automatically generated

Hình 2.12. Gắn link thư mục Library

* + Tại **Linker > Input**, thêm:
    - **Additional Dependencies**:

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v12.6\lib\x64\OpenCL.lib

A screenshot of a computer

Description automatically generated

Hình 2.13. Gắn link thư viện OpenCL.lib

1. **Thiết lập tệp nguồn**:
   * Đảm bảo mã nguồn bao gồm: #include <CL/cl.h>

* **Viết mã OpenCL**

**Mã ví dụ cơ bản:**

#include <CL/cl.h>

#include <stdio.h>

int main() {

cl\_uint platformCount;

cl\_platform\_id platforms[10];

cl\_int err = clGetPlatformIDs(10, platforms, &platformCount);

if (err != CL\_SUCCESS) {

printf("Lỗi: Không thể lấy danh sách nền tảng OpenCL.\n");

return -1;

}

printf("Cai dat thanh cong moi truong OpenCL");

return 0;

}

* **Chạy chương trình**

**Biên dịch với Visual Studio**:

* + Nhấn **Ctrl + Shift + B** để build.
  + Nhấn **F5** để chạy.

A screenshot of a computer

Description automatically generated

Hình 2.14. Kết quả chạy chương trình

* **Kiểm tra và gỡ lỗi**

1. Nếu không tìm thấy thiết bị OpenCL:
   * Kiểm tra xem driver đã được cài đặt đúng chưa.
   * Đảm bảo GPU/CPU hỗ trợ OpenCL.
2. Sử dụng **clinfo** để kiểm tra danh sách thiết bị OpenCL khả dụng.

## 2.3 Các công cụ hỗ trợ OpenCL

### 2.3.1. Platform API

**1. Khái niệm và vai trò**

Platform API: Bao gồm các hàm và chức năng được sử dụng để truy xuất thông tin về các platform và device có sẵn trong hệ thống (Nvidia, Intel, AMD)

* Truy vấn thông tin về các nền tảng và thiết bị (CPU, GPU).
* Thiết lập môi trường để chạy các chương trình OpenCL.

**2. Các hàm chính**

* clGetPlatformIDs: Liệt kê các nền tảng OpenCL khả dụng.
* clGetPlatformInfo: Lấy thông tin chi tiết của một nền tảng.
* clGetDeviceIDs: Liệt kê các thiết bị (CPU, GPU) trên một nền tảng.

**3. Ví dụ minh họa mã nguồn**

#include <CL/cl.h>

#include <stdio.h>

int main() {

cl\_platform\_id platforms[10];

cl\_uint num\_platforms;

clGetPlatformIDs(10, platforms, &num\_platforms);

printf("Number of OpenCL platforms: %d\n", num\_platforms);

return 0;

}

### 2.3.2. Runtime API

**1. Khái niệm và vai trò**  
Runtime API cung cấp các hàm để quản lý chương trình OpenCL trong quá trình thực thi, bao gồm:

* Tạo và quản lý **ngữ cảnh** (context).
* Quản lý **hàng đợi lệnh** (command queue).
* Quản lý **bộ nhớ** (memory buffers).

**2. Các hàm chính**

* **Quản lý ngữ cảnh:**
  + clCreateContext: Tạo ngữ cảnh.
  + clReleaseContext: Giải phóng ngữ cảnh.
* **Quản lý hàng đợi lệnh:**
  + clCreateCommandQueue: Tạo hàng đợi lệnh.
* **Quản lý bộ nhớ:**
  + clCreateBuffer: Tạo bộ đệm trên thiết bị.
  + clEnqueueWriteBuffer: Sao chép dữ liệu từ host vào buffer.

**3. Ví dụ minh họa mã nguồn**

cl\_context context = clCreateContext(NULL, num\_devices, devices, NULL, NULL, &err);

cl\_command\_queue queue = clCreateCommandQueue(context, device, 0, &err);

cl\_mem buffer = clCreateBuffer(context, CL\_MEM\_READ\_WRITE, sizeof(int) \* 100, NULL, &err);

### 2.3.3. Kernel Programming Language

**1. Khái niệm và vai trò**  
Kernel Programming Language là ngôn ngữ được sử dụng để viết các chương trình thực thi trên thiết bị (device) như GPU hoặc CPU.

* Dựa trên chuẩn C99 với các mở rộng cho xử lý song song.

**2. Các đặc điểm chính**

* **Từ khóa đặc biệt:**
  + \_\_kernel: Định nghĩa hàm kernel.
  + \_\_global, \_\_local: Xác định phạm vi bộ nhớ.
* **Hàm tích hợp:**
  + get\_global\_id(dim): Lấy chỉ số toàn cục của work-item.

**3. Ví dụ minh họa mã nguồn Kernel**

\_\_kernel void add\_arrays(\_\_global const int \*A, \_\_global const int \*B, \_\_global int \*C) {

int id = get\_global\_id(0);

C[id] = A[id] + B[id];

}

**4. Các bước sử dụng Kernel trong chương trình OpenCL**

* **Tạo và biên dịch Kernel:**
  + clCreateProgramWithSource: Tạo chương trình từ mã nguồn Kernel.
  + clBuildProgram: Biên dịch Kernel.
* **Thực thi Kernel:**
  + clEnqueueNDRangeKernel: Chạy Kernel song song trên thiết bị.

### 2.3.4. Tóm tắt các công cụ hỗ trợ OpenCL

| **Công cụ** | **Chức năng** | **Ví dụ minh họa** |
| --- | --- | --- |
| **Platform API** | Truy vấn và thiết lập thông tin nền tảng, thiết bị. | clGetPlatformIDs, clGetDeviceIDs |
| **Runtime API** | Quản lý ngữ cảnh, hàng đợi lệnh, bộ nhớ trong chương trình OpenCL. | clCreateContext, clCreateBuffer |
| **Kernel Programming Language** | Viết các chương trình thực thi trên thiết bị (GPU/CPU), hỗ trợ tính toán song song. | \_\_kernel void add\_arrays |

Bảng 2.1. Các công cụ hỗ trợ OpenCL

# CHƯƠNG III: THỰC NGHIỆM

## Bài toán thực hiện phép cộng hai vector số nguyên

* **Mô tả bài toán:**

Bài toán yêu cầu thực hiện phép cộng từng phần tử tương ứng của hai vector số nguyên có kích thước lớn (99999 phần tử). Kết quả của phép cộng này được lưu lại trong một trong hai vector ban đầu.

* **Các bước thực hiện**:
* **Khởi tạo hai vector:**
* Hai mảng data1 và data2 có kích thước 99999 được sử dụng để lưu trữ dữ liệu.
* Các phần tử trong hai mảng được gán giá trị ngẫu nhiên trong khoảng từ 0 đến 99 bằng cách sử dụng hàm std::rand().
* **Thực hiện phép cộng:**
* Với mỗi phần tử ở vị trí i trong hai mảng data1 và data2, thực hiện phép cộng:

data1[i] = data1[i]+data2[i]

Kết quả được lưu đè vào mảng data1.

* **Đo thời gian thực thi:**
* Sử dụng hàm omp\_get\_wtime() để đo thời gian bắt đầu và kết thúc của quá trình thực hiện phép cộng.
* Thời gian thực thi được tính toán và in ra màn hình dưới đơn vị **milliseconds (ms)**.
* **In kết quả:**
* Giá trị của từng phần tử trong mảng data1 sau khi đã thực hiện phép cộng được in ra màn hình.
* **Code:**

A screen shot of a computer program

Description automatically generated

Hình 3.1. Mã Code bài toán thực hiện phép cộng mảng 2 vector số nguyên

A screen shot of a computer program

Description automatically generated

Hình 3.2. Mã Code bài toán thực hiện phép cộng mảng 2 vector số nguyên

A computer screen shot of a program code

Description automatically generated

Hình 3.3. Mã Code bài toán thực hiện phép cộng mảng 2 vector số nguyên

* **Đồ thị biểu diễn:**

**A graph with blue and orange lines

Description automatically generated**

Đồ thị 3.1. Đồ thị biểu thị thời gian thực thi dựa trên kích cỡ vector

A graph with a line going up

Description automatically generated

Đồ thị 3.2. Đồ thị thể hiện tốc độ Speedup của chương trình song song so với tuần tự

# CHƯƠNG IV: KẾT QUẢ

## KẾT LUẬN

Bộ môn Tính toán hiệu năng cao trong Công nghệ Thông tin đã và đang chứng minh vai trò quan trọng của mình trong việc giải quyết các bài toán phức tạp, đòi hỏi hiệu năng tính toán lớn. Với sự phát triển của các công nghệ như OpenCL, việc khai thác sức mạnh của các kiến trúc phần cứng khác nhau để thực hiện tính toán song song đã trở nên khả thi và hiệu quả hơn.

Thông qua quá trình nghiên cứu, tìm tòi và triển khai đề tài “Giới thiệu về OpenCL”, chúng em đã đạt được những kết quả nhất định trong việc tiếp cận công nghệ này, hiểu rõ hơn về cách xây dựng và tối ưu hóa các ứng dụng tính toán song song trên nền tảng OpenCL. Việc sử dụng OpenCL giúp phát triển tư duy lập trình và kỹ năng xử lý bài toán thực tiễn.

Chúng em xin gửi lời cảm ơn chân thành đến Giảng viên Hà Mạnh Đào, người đã tận tình hướng dẫn và hỗ trợ trong suốt quá trình thực hiện đề tài. Sự chỉ bảo và động viên từ thầy đã giúp chúng em hoàn thiện bài nghiên cứu và tích lũy được nhiều kinh nghiệm quý báu cho những dự án sau này.

## TÀI LIỆU THAM KHẢO

## PHỤ LỤC