Akshit Singh
TIFAC CORE in Cybersecurity
Amrita Vishwa Vidyapeetham

OpenCL

19CSE100
Problem Solving AND Algorithmic Thinking
PROGRAMMING LANGUAGE Survey Assignment

Project Objectives

01 To broaden our horizon by learning about a new language

02 To learn about the features of OpenCL

03 Relevance of OpenCL today

History of OpenCL

KHRONOS™
GROUP
CONNECTING SOFTWARE TO SILICON

OPENCL WAS DEVELOPED BY THE KHRONOS GROUP, AN INDUSTRY CONSORTIUM OF LEADING COMPANIES IN THE COMPUTER GRAPHICS, PARALLEL COMPUTING, AND CONSUMER ELECTRONICS INDUSTRIES.

THE KHRONOS GROUP IS RESPONSIBLE FOR THE CREATION OF OPEN STANDARDS FOR DYNAMIC MEDIA AND PARALLEL COMPUTING, AND OPENCL WAS CREATED AS PART OF THEIR EFFORTS IN THIS AREA. IT WAS RELEASED IN 2008 AND HAS SINCE BECOME A WIDELY ADOPTED STANDARD FOR PARALLEL PROGRAMMING ON HETEROGENEOUS SYSTEMS. WHILE OPENCL WAS CREATED BY THE KHRONOS GROUP, ITS DEVELOPMENT INVOLVED INPUT AND CONTRIBUTIONS FROM A WIDE RANGE OF INDUSTRY PARTNERS AND INDIVIDUALS.

Paradigm

01 OpenCL follows the data parallel programming paradigm, which means that the same program is executed on multiple data elements in parallel.

02 In OpenCL, this is achieved by dividing the computation into smaller tasks and executing these tasks on different processing units.

03 The processing units can be CPU cores, GPU cores, or specialized processing units like DSPs or FPGAs.

Language Support

OpenCL is a cross-platform standard for parallel programming, and as such, it is supported by a wide range of operating systems. Some of the most popular operating systems that support OpenCL include:

01 Windows

OpenCL is supported on Windows 7 and later versions, including Windows 10.

02 Linux

OpenCL is supported on a wide range of Linux distributions, including Ubuntu, Fedora, and Red Hat.

03 macOS

OpenCL is supported on macOS 10.7 and later versions.

04 Android

OpenCL is supported on some Android devices, though support varies depending on the device and vendor.

Hello World

```c
#include <CL/cl.h>
#include <stdio.h>
```
} necessary header files

```c
int main()
{
    cl_platform_id platform_id = NULL;
    cl_device_id device_id = NULL;
    cl_uint ret_num_devices;
    cl_uint ret_num_platforms;
    cl_int ret = clGetPlatformIDs(1, &platform_id, &ret_num_platforms);
    ret = clGetDeviceIDs(platform_id, CL_DEVICE_TYPE_DEFAULT, 1, &device_id, &ret_num_devices);
    cl_context context = clCreateContext(NULL, 1, &device_id, NULL, NULL, &ret);
    cl_command_queue command_queue = clCreateCommandQueue(context, device_id, 0, &ret);
    cl_mem a_mem_obj = clCreateBuffer(context, CL_MEM_READ_WRITE, sizeof(char) * 20, NULL, &ret);
    const char *hello = "Hello, World!";
    ret = clEnqueueWriteBuffer(command_queue, a_mem_obj, CL_TRUE, 0, sizeof(char) * 20, hello, 0, NULL, NULL);
    char result[20];
    ret = clEnqueueReadBuffer(command_queue, a_mem_obj, CL_TRUE, 0, sizeof(char) * 20, result, 0, NULL, NULL);
    result[19] = '\0';
    printf("%s\n", result);
    ret = clFlush(command_queue);
    ret = clFinish(command_queue);
    ret = clReleaseMemObject(a_mem_obj);
    ret = clReleaseCommandQueue(command_queue);
    ret = clReleaseContext(context);
    return 0;
}
```

} declaring variables for the OpenCL platform and device ID

→ create an OpenCL context

→ create an command queue

→ create an buffer to store "Hello World"

} these functions are use to write and read the string in the buffer

→ printing the result

RELEVANCE TODAY

- OpenCL is still relevant today, especially in the field of parallel computing and general-purpose GPU (GPGPU) programming.
- With the increasing demand for high-performance computing and the growth of machine learning and AI applications, OpenCL has seen a resurgence in popularity in recent years. This is because OpenCL provides a flexible and portable way to harness the parallel processing power of GPUs and other accelerators, which can greatly improve the performance of computationally-intensive applications.
- Despite the availability of other GPGPU programming frameworks, such as CUDA and OpenMP, OpenCL remains a popular choice due to its open standard and cross-platform compatibility. This allows developers to write applications that can run on a wide range of devices, without having to rewrite the code for each platform.

References

- https://en.wikipedia.org/wiki/OpenCL
- https://www.youtube.com/playlist?list=PLiwt1iVUib9s6vyEqdpcgAq7NBRlp9mAY - Introduction to OpenCL
- https://gist.github.com/ddemidov/2925717
- https://ulhpc-tutorials.readthedocs.io/en/latest/gpu/opencl/