OpenCL Using Amazon Web Services

Kouame H. Kouassi[†] and Kamogelo Mphela[‡]
EEE4084F Class of 2017
University of Cape Town
South Africa
[†]KSSKOU001 [‡]MPHKAM003

***Abstract*—Open Computing Language (OpenCL) is a framework for writing programs that execute across heterogeneous platforms consisting of central processing units (CPUs), graphics processing units (GPUs), digital signal processors (DSPs), field-programmable gate arrays (FPGAs) and other processors or hardware accelerators.**

I. Introduction

Cloud computing refers to storing and processing data on a cluster of servers on the internet. This introduces a greater degree of flexibility, convenience and ease of use.This paper focuses on developing OpenCL code and doing performance testing.

We will be comparing the performance of a CPU and a GPU provided by AWS

II. Methodology

This section describes the methods used in these experiments.

A. Device Information

1) CPU:

- Device Name: Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz
- Available Compute Units: 4
- Available work Units: [8192, 8192, 8192]
- Clockrate: 2300
- Available Alloc Memory: 16097245184

2) GPU:

- Device Name: Tesla K80
- Available Compute Units: 13
- Available work Units: [1024, 1024, 64]
- Clockrate: 823
- Available Alloc Memory: 2998894592

B. Implementation

Also mention the implementation source code:

```
name_properties = {
        "Device Name":pyopencl.device_info.NAME,
}

processing_properties = {
        "Available Compute Units": pyopencl.device_info.MAX_COMPUTE_UNITS,
        "Available work Units": pyopencl.device_info.MAX_WORK_ITEM_SIZES,
        "Clockrate": pyopencl.device_info.MAX_CLOCK_FREQUENCY
}

memory_properties = {
        "Available Alloc Memory": pyopencl.device_info.MAX_MEM_ALLOC_SIZE
}

for device in (nvidia_device,intel_device):
        #print out all of the device name properties, except the device type
        for property_name in sorted(name_properties.keys() - {"Device Type"}):
                property_string_args =\\
                (property_name,device.get_info(name_properties[property_name]))
                print("%s: %s"%property_string_args)

        #print out all of the processing properties
        for property_name in sorted(processing_properties.keys()):
                property_string_args =\\
                (property_name,device.get_info(processing_properties[property_name]))
                print("%s: %s"%property_string_args)

        #print out all of the memory properties
        for property_name in sorted(memory_properties.keys()):
                property_string_args =\\
                (property_name,device.get_info(memory_properties[property_name]))
                print("%s: %s"%property_string_args)

        print("\n")
```

Listing 1. OpenCL code to obtain device information

```
def run_cpu_program():
        #copying data onto CPU
        pyopencl.enqueue_copy(intel_queue,
        src=a,
        dest=a_intel_buffer)
        pyopencl.enqueue_copy(intel_queue,
        src=b,
        dest=b_intel_buffer)

        #running program
        kernel_arguments = (a_intel_buffer,b_intel_buffer,c_intel_buffer)
        intel_program.sum(intel_queue,
        a.shape, #global size
        None, #local size
        *kernel_arguments)

        #copying data off CPU
        copy_off_event = pyopencl.enqueue_copy(intel_queue,
        src=c_intel_buffer,
        dest=c)
        copy_off_event.wait()
```

```
def run_gpu_program():
        #copying data onto GPU
        pyopencl.enqueue_copy(nvidia_queue,
        src=a,
        dest=a_nvidia_buffer)
        pyopencl.enqueue_copy(nvidia_queue,
        src=b,
        dest=b_nvidia_buffer)

        #running program
        kernel_arguments = (a_nvidia_buffer,b_nvidia_buffer,c_nvidia_buffer)
        nvidia_program.sum(nvidia_queue,
        a.shape, #global size
        None, #local size
        *kernel_arguments)

        #copying data off GPU
        copy_off_event = pyopencl.enqueue_copy(nvidia_queue,
        src=c_nvidia_buffer,
        dest=c)
        copy_off_event.wait()
```

```
#define data_t int

// Factor Count
// X - N x M matrix
// factors - N x M --
kernel void factor_count(global data_t *X,
global data_t *factors,
int N,
int M)
{
        int row = get_global_id(0);
        int col = get_global_id(1);

        int offset = row*M; //start of row of X and factor

        //Calculation loop
        data_t sum = 0; //private variable used to cache result
        for(int i=0;i<N;++i) {
                for(int j=0;j<M;++j) {
                        int nf = 0;
                        for(int k=2;k<101;++k) {
                                if( (X[offset + j + i*M] % k) == 0)
                                ++nf;
                        }
                        factors[offset + j + i*M] = nf;
                }
        }
}
```

```
def opencl_factor_count(data_arrays,buffers,queue,kernel,N=100,M=100):
        a = data_arrays
        a_buffer,c_buffer = buffers
        N,M = numpy.int32(M),numpy.int32(N) #converting to the exepcted type for OpenCL

        #copying data onto device
        copyon_events = []

        copyon_events += [pyopencl.enqueue_copy(queue,
        src=a,
        dest=a_buffer)]

        #running program
        kernel_event = kernel(queue,
        (N,N), #global size
        None, #local size
        a_buffer,c_buffer,N,M, #Kernel Arguments
        wait_for = copyon_events)


        #copying data off device
        c = numpy.empty((N,N),dtype=dt)
        copyoff_event = pyopencl.enqueue_copy(queue,
        src = c_buffer,
        dest = c,
        wait_for = [kernel_event])
        copyoff_event.wait()

        return c

data_arrays = create_arrays()

get_ipython().magic('timeit opencl_factor_count(data_arrays,nvidia_buffers,\\
nvidia_queue,nvidia_program.factor_count)')
get_ipython().magic('timeit opencl_factor_count(data_arrays,intel_buffers,\\
intel_queue,intel_program.factor_count)')
```

Only list what is relevant. Don't give too much detail - just enough to show what you've done. This template supports the following languages:

C. Experiment Procedure

Furthermore, include detail relating to the experiment itself: what did you do, in what order was this done, why was this done, etc. What are you trying to prove / disprove?

III. Results

The results section is for presenting and discussing your findings. You can split it into subsections if the experiment has multiple sections or stages.

A. Figures

Include good quality graphs (see Fig. **??**). These were produced by the Octave code presented in listings **??** and **??**. You can play around with the `PaperSize` and `PaperPosition` variables to change the aspect ratio. An easy way to obtain more space on a paper is to use wide, flat figures, such as Fig. **??**.

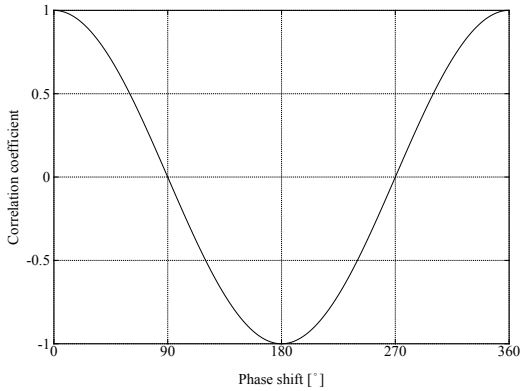

Fig. 1. The CPU vs. GPU speedup on sum as a function of vector size.

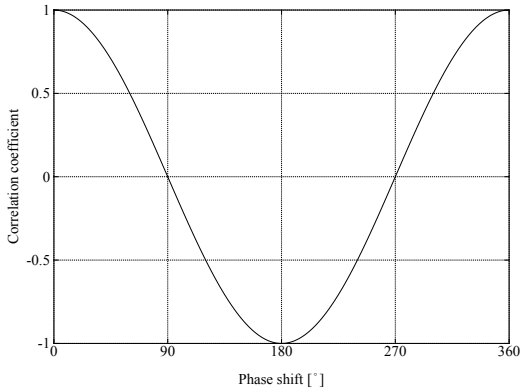

Fig. 2. The CPU vs. GPU speedup on factor count as a function of matrix size.

B. Tables

CPU vs GPU on Sum

Size(Nx1)	Golden Time(ms)	CPU Time(ms)	GPU Time (ms)	CPU Speedup
1e2	00	00	00	00
1e4	00	00	00	00
1e8	00	00	00	00
1e9	00	00	00	00

CPU vs GPU on Factor Count

Size(NxM)	Octave Time(s)	CPU Time(s)	GPU Time (s)	CPU Speedup	G
10x10	00	00	00	00	
100x100	17.44	5.14	0.17	3.39	
1000x1000	00	00	00	00	
10000x10000	00	00	00	00	

IV. Conclusion

The conclusion should provide a summary of your findings. Many people only read the introduction and conclusion of a paper. They sometimes scan the tables and figures. If the

conclusion hints at interesting findings, only then will they bother to read the whole paper.

You can also include work that you intend to do in future, such as ideas for further improvements, or to make the solution more accessible to the general user-base, etc.

Publishers often charge "overlength article charges" [7], so keep within the page limit. In EEE4084F we will simulate overlength fees by means of a mark reduction at 10% per page. Late submissions will be charged at 10% per day, or part thereof.

REFERENCES

[1] T. Oetiker, H. Partl, I. Hyna, and E. Schlegl, "The Not So Short Introduction to LaTeX 2_ε," https://tobi.oetiker.ch/lshort/lshort.pdf, Jul. 2015, version 5.05.

[2] "IEEE Conference Paper Templates," http://www.ieee.org/conferences_events/conferences/publishing/templates.html.

[3] A. Baboon, B. Charles, D. Ester, and F. Generalson, "An Amazing Title," Their Not-so-awesome University, Technical Report, Apr. 1492.

[4] B. van der Zander, J. Sundermeyer, and T. Hoffmann, "TeXstudio – A LaTeX Editor," https://sourceforge.net/projects/texstudio/.

[5] "InkScape Website," http://www.inkscape.org/.

[6] J. Taylor and J. G. Hoole, "Robust Protocol for Sending Synchronisation Pulse and RS-232 Communication over Single Low Quality Twisted Pair Cable," in *Proceeding of ICIT*. Taiwan: IEEE, Mar. 2016.

[7] "Voluntary Page and Overlength Article Charges," http://www.ieee.org/advertisement/2012vpcopc.pdf, 2014.