SIT315- Seminar 8 - Distributed Computing – OpenCL

Activity 1

1. Host:
   Host is the CPU of the computer and it holds the **host memory**. The host processor has full control of this memory space and can read and write from this space without any restrictions.
2. Device:
   Any hardware that has the OpenCL driver is called a device. For example- GPU-graphics processing unit. Each device has their own **Global memory** which is separate from the host memory.
3. Compute Unit:
   Each device can have **one or more compute units** and each compute unit have their own **Local memory, multiple processing element (core) and private memory.**
4. Processing Elements:
   Processing Elements are part of Compute units. Each processing element do a part of the task in parallel. For example- in case of addition of 100 vectors, and 5 processing units, the task can be divided between the processing units as such each PU gets 20 vectors.
5. Context:
   A context is a platform with a set of available devices for that platform. For example- with an NVIDIA platform, a working context will include a NVIDIA GPU device. In the context environment, the kernels execute, and synchronization and memory management is defined.
6. Command Queue:
   Command Queue as the name suggests contains all the commands for a device. Kernel execution, synchronization, and memory transfer operations are all submitted through a command-queue. Each command queue points to a single device within a context.

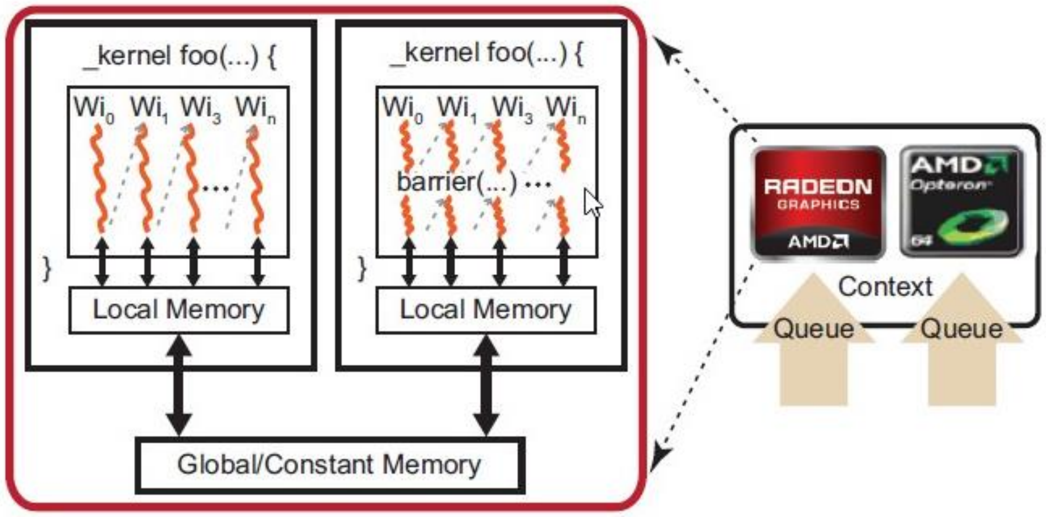
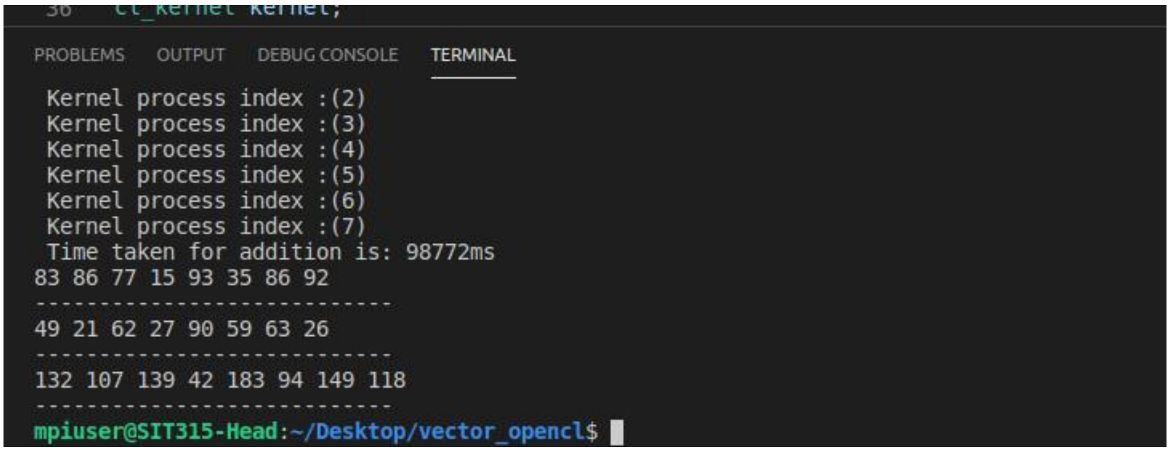


***Image taken from here

7. Host program:
   Host program is where the implementation of OpenCL environment is done by writing the code on the host. Here, we define the platform we are using, create and build the program, setup memory objects, define the kernel and execute the kernels.
8. Program Kernel:
   Kernel executes the part of the code that we want to execute in parallel on our Device- for instance GPU.

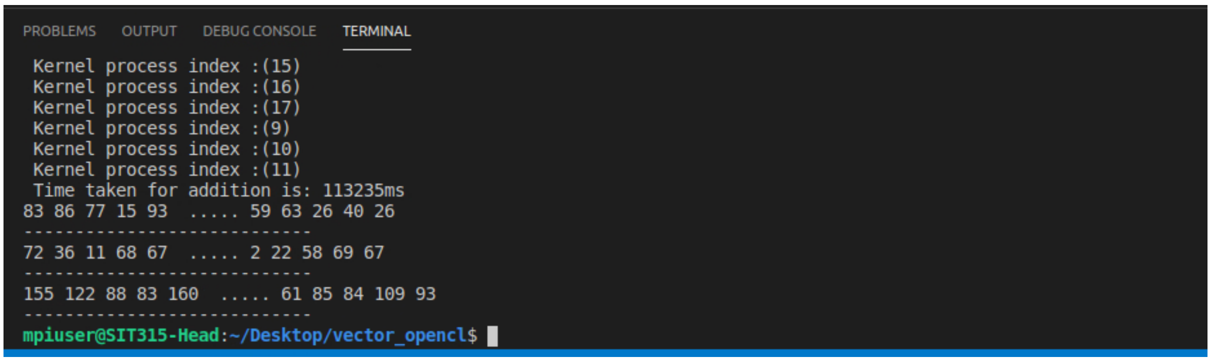
Activity 2



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
 Kernel process index :(2)
 Kernel process index :(3)
 Kernel process index :(4)
 Kernel process index :(5)
 Kernel process index :(6)
 Kernel process index :(7)
 Time taken for addition is: 98772ms
83 86 77 15 93 35 86 92
----------------------------
49 21 62 27 90 59 63 26
----------------------------
132 107 139 42 183 94 149 118
----------------------------
mpiuser@SIT315-Head:~/Desktop/vector_opencl$
```

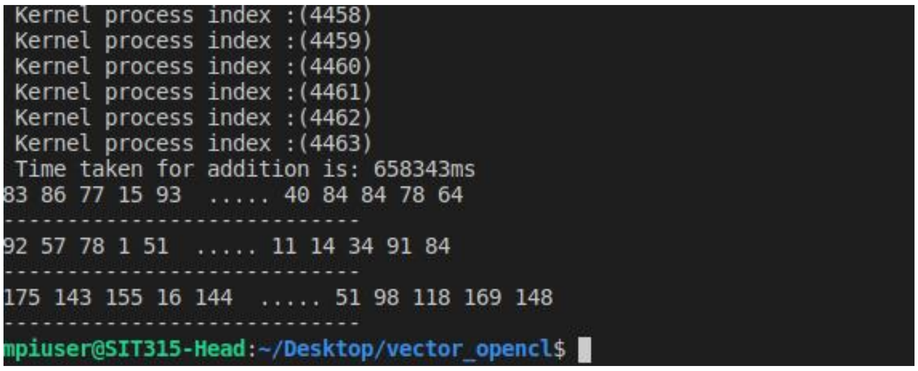
For size=8, time taken 98,772 milliseconds



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
 Kernel process index :(15)
 Kernel process index :(16)
 Kernel process index :(17)
 Kernel process index :(9)
 Kernel process index :(10)
 Kernel process index :(11)
 Time taken for addition is: 113235ms
83 86 77 15 93  ..... 59 63 26 40 26
----------------------------
72 36 11 68 67  ..... 2 22 58 69 67
----------------------------
155 122 88 83 160  ..... 61 85 84 109 93
----------------------------
mpiuser@SIT315-Head:~/Desktop/vector_opencl$
```

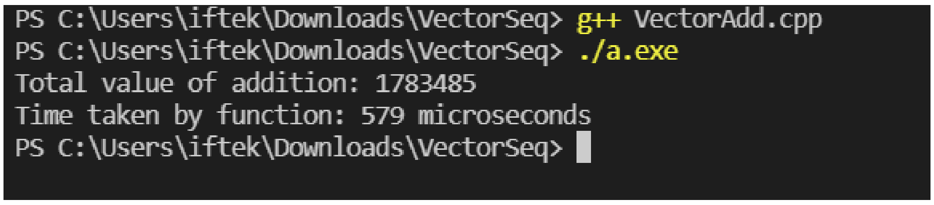
For size=18, time taken 113,235 milliseconds



```
 Kernel process index :(4458)
 Kernel process index :(4459)
 Kernel process index :(4460)
 Kernel process index :(4461)
 Kernel process index :(4462)
 Kernel process index :(4463)
 Time taken for addition is: 658343ms
83 86 77 15 93  ..... 40 84 84 78 64
----------------------------
92 57 78 1 51  ..... 11 14 34 91 84
----------------------------
175 143 155 16 144  ..... 51 98 118 169 148
----------------------------
mpiuser@SIT315-Head:~/Desktop/vector_opencl$
```

For size=18000, time taken 658,343 milliseconds

Therefore, it is evident that with increasing size of the vectors, the time taken for vector addition is increasing.



```
PS C:\Users\iftek\Downloads\VectorSeq> g++ VectorAdd.cpp
PS C:\Users\iftek\Downloads\VectorSeq> ./a.exe
Total value of addition: 1783485
Time taken by function: 579 microseconds
PS C:\Users\iftek\Downloads\VectorSeq>
```

For the sequential program of size 18000, its working faster than the OpenCl implementation.