High Energy Physics computing on heterogeneous platforms via OpenCL

V. Singhal* and S. Chattopadhyay
Experimental High Energy Physics and Applications Group,
Variable Energy Cyclotron Centre, 1/AF Bidhan Nagar, Kolkata-700 064, INDIA

Introduction

High energy physics experiments produces huge amount of data, therefore needs embarrassingly parallel computing power. In this direction manycore architecture is playing as a pivotal role. With keeping this in mind many hardware developer have been building such dedicated processor other than CPU like NVIDIA GPU (Graphical Processing Unit)[1], Xeon PHI by INTEL, AMD GPU and also in the CPU which can provide manycore, multicore architecture like APU (Accelerated Processing Unit) by AMD (earlier ATI)[2], Cell processor by IBM etc. All these hardware are based on SIMT (Single Instruction Multiple Thread) technology and comes with hundreds of cores therefore architecture wise many threads can run parallel, but all these need different programming paradigm or API for exploiting their power. In this direction Apple developed OpenCL (Open Compute Language) managed by Khronos group [3] API and NVIDIA came with CUDA (Compute Unified Device Architecture) [1]. In this paper we implemented a 3 dimension first level event selection (FLES) process for MUCH (Muon Chamber) at CBM experiment [4] via OpenCL and executed the same on heterogeneous platforms including multiple GPUs and CPUs.

OpenCL programming paradigm

In the beginning OpenCL seems difficult as far as its syntax and programming procedure concerned. Writing a small $HelloWord$ program in OpenCL needs creating $platform, device, context, queue$ then $writebuffer$, creating source or kernel, $enqueue$ kernel and $readbuffer$ etc seems cumbersome compare to CUDA which provide easy terminology for writing programs. OpenCL program can be compiled via available C or C++ compiler unlike CUDA needs another NVIDIA compiler. Once OpenCL program written and compiled then can be executed on any device whether GPU, CPU, APU etc unlike CUDA program which can be executed only on NVIDIA GPU [5]. CUDA treats CPU as host and only NVIDIA GPU as device, whereas OpenCL treats any hardware as computing device, therefore once instruction queue created, that can be executed on all the available computing resources.

FLES process and implementation

Entire process of FLES for MUCH in CBM experiment at FAIR (Facility for Antiproton Ion Research) Darmstadt Germany and implementation of the process on the NVIDIA Tesla C2075 GPU [6] via CUDA[1] have been described in paper [4] and [8]. In this paper the same process has been implemented via OpenCL[3] onto heterogeneous platform comprises of multiple GPUs and CPUs.

We have used a workstation which consists 2 * Intel Xeon 2.8GHz six core processors and 2 Nvidia GPUs one Tesla C2075 [6] and another NVIDIA Quadro 4000 [7] for running the MUCH FLES process implemented in OpenCL. We have generated results for the following:- (a) Using both Intel Xeon six core processors (12 cores of computing) (b) Using Tesla C2075 GPU [6] (14 SMs (Streaming Processor) running parallel in 32 warps comprises 448 cores), via CUDA and (c) OpenCL (d) Using Quadro 4000 GPU [7], via CUDA and (e) OpenCL respectively.

Summary and Conclusion

FIG 1 shows execution time in ms on y-axis and number of events on x-axis for the

*Electronic address: vikas@vecc.gov.in

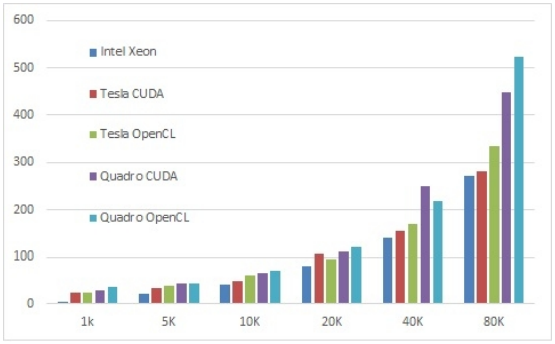

FIG. 1: FLES process execution time comparision

FLES process for multiple event sizes running on multiple computing devices like Intel Xeon CPUs and Tesla Quadro GPUs using CUDA and OpenCL. Used workstation's base CPU is quite powerful and also it has 24G bytes of RAM size therefore computation time for the CPUs is the lowest compare to others. Result shows that OpenCL code execution time is slightly more than CUDA code execution time whether running on Tesla GPU or Quadro GPU, because both GPUs manufactured by NVIDIA and CUDA programing paradigm also developed by NVIDIA therefore it can exploit NVIDIA GPUs efficiently as it is customized for the NVIDIA GPUs only. Tesla GPU is more powerful than Quadro GPU, as many hardware differences between them like processor speed, global memory size, number of computing cores therefore execution time is almost 2 times more for Quadro GPU than Tesla GPU at very high number of events .
We can say that OpenCL programming paradigm is more usable than CUDA paradigm because it is open source multi platform computing paradigm. Our report suggests that once a program is written using OpenCL, then can be run on multiple platforms simultaneously without changing the code. OpenCL can use all available computing units for single problem and simultaneously exploit all. Now these days workstation or server kind of machines have many accelerating computing units like AMD GPUs or NVIDIA GPUs on a single motherboard connected with manycore base CPU via PCIe slots, for this kind of system OpenCL can provide suitable solution to use all the computing units for one application and this will also help us to achieve targeted event rate 10^7 events per second for FLES process of MUCH at CBM experiment.

Future Scope of work

For parallel computation on CPU, MPI and OpenMP can be used. As future work we will perform the similar computation using MPI and compare result with OpenCL result running on the CPU. In this report used FLES process for MUCH is developed using 3 dimensions X, Y, Z only, in near future we will develop FLES process for MUCH using 4 dimensions (time as 4^{th} dimension). After that we will perform computation time analysis using CUDA, OpenCL on both CPUs and GPUs.

References

[1] NVIDIA CUDA, http://www.nvidia.com/cuda
[2] AMD APU, http://www.amd.com/apu
[3] OpenCL, http://www.khronos.org/opencl
[4] V. Singhal *et al.*, Proc. DAE Symp. on Nucl. Phys. **57** (2012) 972
[5] https://devtalk.nvidia.com/default/topic/483047/simple-question-33-can-cuda-code-be-run-on-cpu
[6] Tesla Cards, www.nvidia.in/docs/IO/43395/NV-DS-Tesla-C2075.pdf
[7] Quadro 4000, http://www.nvidia.in/object/product-quadro-4000-in.html
[8] P. P. Bhaduri *et al.*, Proc. DAE Symp. on Nucl. Phys. **55** (2010) 640