A Black-Box Approach to Energy-Aware Scheduling on Integrated CPU-GPU Systems Paper - 5

Ashish Jain & Sarosh Hasan
(MT18052 & MT18084)

The machine used for Project:

Intel i7 8th gen with Intel UHD Graphics 620

Installations Required for Machine

Installation of Intel compute runtime for OpenCL

Info: The Intel(R) Graphics Compute Runtime for OpenCL(TM) is an open source project to converge Intel's development efforts on OpenCL(TM) compute stacks supporting the GEN graphics hardware architecture.

Installation procedure on Ubuntu

1. Create a temporary directory

Example:

```
mkdir neo
```

2. Download all *.deb packages

Example:

```
cd neo
wget
https://github.com/intel/compute-runtime/releases/download/19.10.12542/intel-gmmlib_18
.4.1_amd64.deb
wget
https://github.com/intel/compute-runtime/releases/download/19.10.12542/intel-igc-core_
19.07.1542_amd64.deb
wget
https://github.com/intel/compute-runtime/releases/download/19.10.12542/intel-igc-openc
l_19.07.1542_amd64.deb
wget
https://github.com/intel/compute-runtime/releases/download/19.10.12542/intel-opencl_19
.10.12542_amd64.deb
wget
https://github.com/intel/compute-runtime/releases/download/19.10.12542/intel-ocloc_19.
10.12542_amd64.deb
```

1. Install all packages as root

Example:

```
sudo dpkg -i *.deb
```

Turbostat for Energy measurement on Intel processors:

sudo apt install linux-tools-common

OpenCL installation:

```
sudo apt update
```

```
sudo apt install build-essentials
```

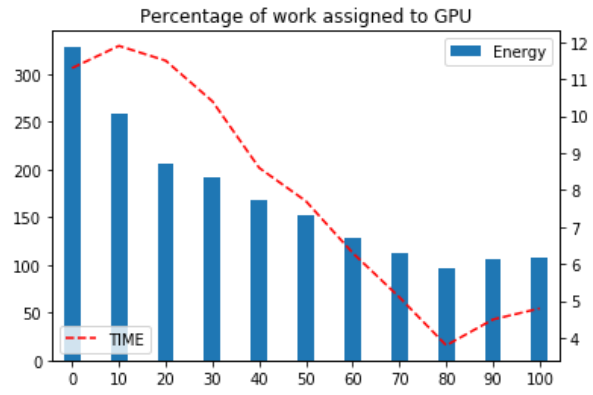
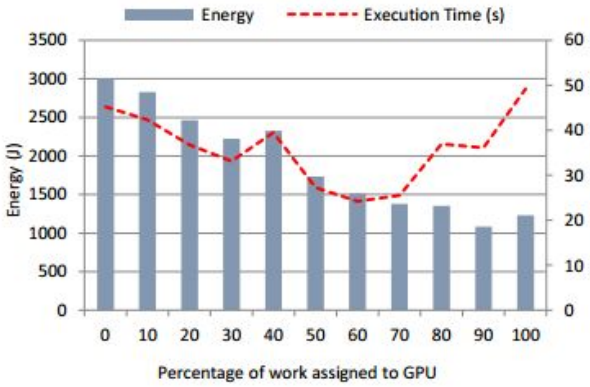
```
sudo apt install ocl-icd-opencl-dev
```

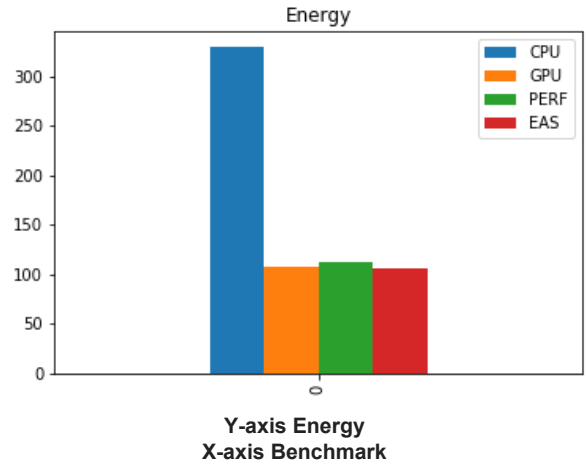
Experimental Results

Benchmark used

Matrix Multiplication:

- **Size: 2000 x 2000**
- **COTTON_WORKERS=4**

Experimental Results	Paper Results
 Y-axis(left Energy) right(Execution Time in seconds) X-axis GPU offload percentage	 Y-axis(left Energy) right(Execution Time in seconds) X-axis GPU offload percentage



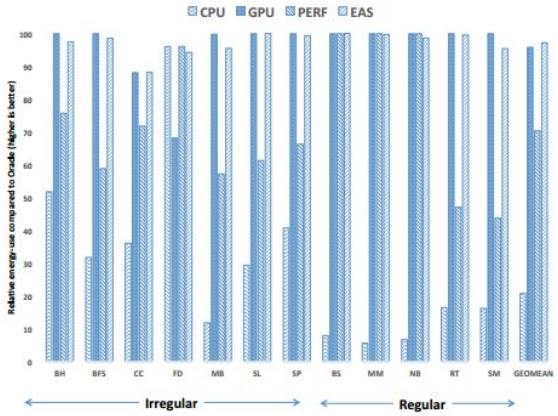
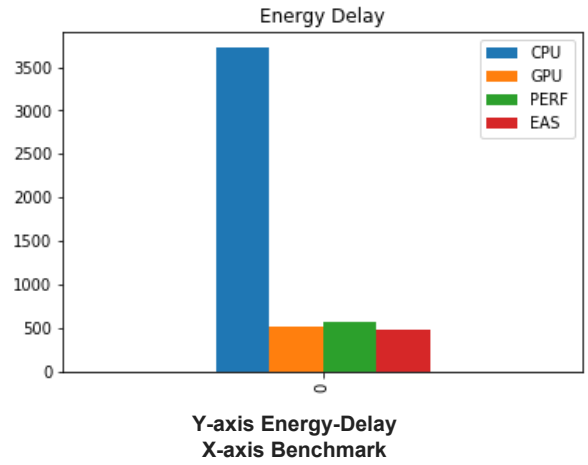


Figure 10: Relative energy-use efficiency compared to Oracle on the desktop. Oracle is at 100% (higher is better).



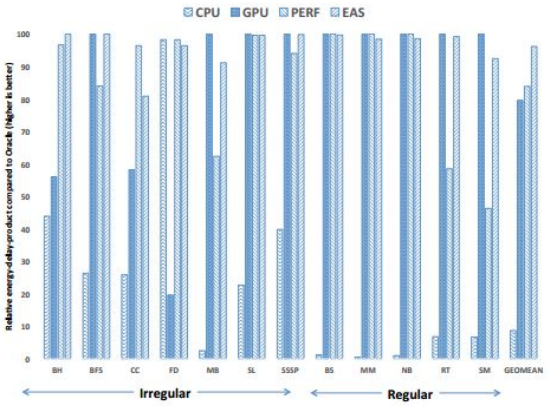


Figure 9: Relative energy-delay product efficiency compared to Oracle on the desktop. Oracle is at 100% (higher is better).

Benchmark 0: Matrix Multiplication

Offloading Factor to GPU	Energy in Joules	Execution Time in sec
0.0	329.02	11.3
.1	258.29	11.9
.2	205.81	11.5
.3	191.81	10.4
.4	168.83	8.6
.5	151.99	7.7

.6	**128.39**	**6.3**
.7	**112.09**	**5.1**
.8	**96.23**	**3.8**
.9	**105.67**	**4.5**
1.0	**107.32**	**4.8**

Alpha perf computed = 0.7
RC (Throughput of CPU)= 176.99
RG (Throughput of GPU)= 416.66

Comparison with Baseline

Implementation:

- We have used cotton runtime implementation for the parallel decomposition of the task associated with the benchmark application.
- Cotton runtime is a work-stealing implementation that is used for offloading task from CPU to GPU.
- We have used turbostat for Linux energy counters
  Example: COTTON_WORKERS=4 turbostat --Joules ./Matrix.e 5 2000
- **We have not used any wrapper/library for OpenCL code offloading.**
- Implementation of various data structures required for the implementation of EAS algorithm.
- OpenCL kernel implementation for benchmark program.
- Method for calculating retired instructions last level cache misses during for online profiling.

Source Code Design -

Directory Structure

- **src/** contains all .cpp and .cl source code files
- **bin/** executable will be generated in this directory
- **inc/** this directory contains all .h files
- **make/** this directory contains make file for the compilation of code.

Lines of code description:

Matrix.cpp

- **Line 14** contains the c++ kernel function that will invoke the offloading of the task to GPU as per the offload provided.
- **Line 56-60** contains code to set parameters according to which GPU tasks will be executed
- **Line 70** contains a call to a parallel GPU offload section.
- **Line 88** contains a decomposition function of a task that will be executed in parallel using cotton runtime.
- **Line 104** contains a function that will distribute CPU load among workers
- **Line 182** invokes GPU kernel using GPU proxy thread.
- **Line 194-200** contains code to verify the results of the benchmark application.

Wrapper.cpp

- **Line 14** calculated energy-delay
- **Line 18** calculates energy-delay product square.
- **Line 63** contains online eas profiling method.
- **Line 114** contains a function to if alpha already existing for a given function.
- **Line 127** contains code to retrieve the value of alpha from the table.
- **Line 161** contains a method that returns a file descriptor that allows measuring performance information.
- **Line 169** method returns the count of retired instructions.

Wrapperinterface.h

- **Line 13** contains a structure for storing the results of profiling.
- **Line 29** contains a global table for values of alpha for a given kernel.
- **Further** lines contain declarations of methods created in wrapper.cpp.

How to Execute the Benchmark:

1. Go to make directory.
2. $ cd make
3. $ make clean matrix
4. $ export COTTON_WORKERS=4
5. $ cd ../bin
6. ./Matrix.e <offload percentage 0-10> <row> <col>