High Performance Computing

National University of Computer and Emerging Sciences

Deliverable # 02

Complex Computing Problem

By

Hamza Kaleem — 23i-0783
Umaima — 23i-0790
Warisha Shaukat — 23i-0809

Contents

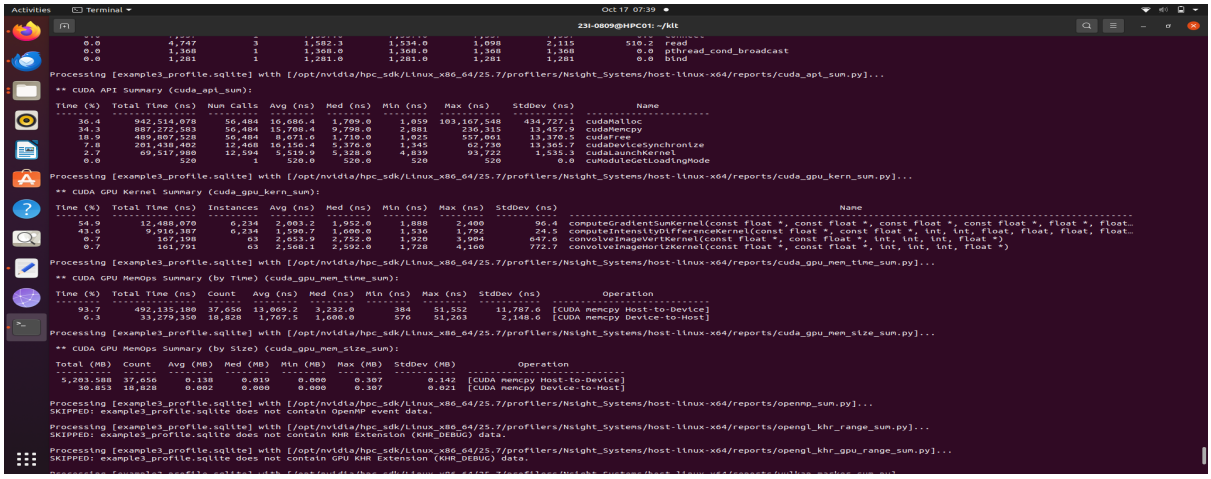
1 Converted Functions

- ConvolveImageHoriz — Offloaded wrapper function call.
- ConvolveImageVert — Offloaded wrapper function call.
- Interpolate — Implemented as both host and device function with core logic intact.
- ComputeGradientSum — Converted to kernel with wrapper function.
- ComputeIntensityDifference — Converetd to kernel with wrapper function.

2 Program Execution

```
(KLT) Tracking 150 features in a 320 by 240 image...
        140 features successfully tracked.
(KLT) Writing 140 features to PPM file: 'feat1.ppm'
(KLT) Tracking 140 features in a 320 by 240 image...
        136 features successfully tracked.
(KLT) Writing 136 features to PPM file: 'feat2.ppm'
(KLT) Tracking 136 features in a 320 by 240 image...
        136 features successfully tracked.
(KLT) Writing 136 features to PPM file: 'feat3.ppm'
(KLT) Tracking 136 features in a 320 by 240 image...
        134 features successfully tracked.
(KLT) Writing 134 features to PPM file: 'feat4.ppm'
(KLT) Tracking 134 features in a 320 by 240 image...
        132 features successfully tracked.
(KLT) Writing 132 features to PPM file: 'feat5.ppm'
(KLT) Tracking 132 features in a 320 by 240 image...
        129 features successfully tracked.
(KLT) Writing 129 features to PPM file: 'feat6.ppm'
(KLT) Tracking 129 features in a 320 by 240 image...
        125 features successfully tracked.
(KLT) Writing 125 features to PPM file: 'feat7.ppm'
(KLT) Tracking 125 features in a 320 by 240 image...
        123 features successfully tracked.
(KLT) Writing 123 features to PPM file: 'feat8.ppm'
(KLT) Tracking 123 features in a 320 by 240 image...
        120 features successfully tracked.
(KLT) Writing 120 features to PPM file: 'feat9.ppm'
(KLT) Writing feature table to text file: 'features.txt'
(KLT) Writing feature table to binary file: 'features.ft'

C:\Users\MrLaptop\source\repos\HPC-Deliverable-2\x64\Debug\HPC-Deliverable-2.exe (process 25080) exited with code 0 (0x0).
Press any key to close this window . . .
```



The program was tested on various frame sizes with screenshots for nframes = 5, and nframes = 10 present in the output directory. Additionally, multiple thread configurations were tested specified in the aforementioned directory.

3 Performance Results

3.1 CPU Times

% Time	Cumulative seconds	Self seconds	Calls	Total ms/call	Self ms/call	Name
42.86	0.03	0.03	2069270	0.00	0.00	_interpolate
28.57	0.05	0.02	63	0.32	0.32	_convolveImageVert
14.29	0.06	0.01	63	0.16	0.16	_convolveImageHoriz
0.00	0.07	0.00	8645	0.00	0.00	_computeIntensityDifference
0.00	0.07	0.00	6235	0.00	0.00	_computeGradientSum

Table 1: Profiling summary of CPU execution times.

3.2 GPU Times

3.2.1 CUDA API Summary

Time (%)	Total Time (ns)	Num Calls	Avg (ns)	Med (ns)	Min (ns)	Name
36.4	942,514,078	56,468	16,684.8	4,179.5	520	cudaMalloc
34.9	902,487,528	56,464	15,984.4	1,710.9	520	cudaMemcpy
18.3	201,438,420	56,468	3,568.5	1,654.6	520	cudaFree
7.7	69,517,980	12,520	5,551.9	4,530.3	520	cudaDeviceSynchronize
2.7	69,517,980	12,520	5,519.0	5,328.0	520	cudaLaunchKernel
0.0	520	1	520.0	520.0	520.0	cuModuleGetLoadingMode

Table 2: CUDA API Summary (cuda_api_sum).

3.2.2 CUDA GPU Kernel Summary

Time (%)	Total Time (ns)	Instances	Avg (ns)	Med (ns)	Min (ns)	Name
54.9	12,488,070	6,234	1,952.1	1,888.0	1,820	computeGradientSumKernel
43.6	9,916,387	6,234	1,590.7	1,600.0	1,536	computeIntensityDifferenceKe
0.7	161,791	63	2,567.9	2,592.0	1,728	convolveImageVertKernel
0.7	161,791	63	2,568.1	2,592.0	1,728	convolveImageHorizKernel

Table 3: CUDA GPU Kernel Summary (cuda_gpu_kern_sum).

3.2.3 CUDA GPU MemOps Summary (by Time)

Time (%)	Total Time (ns)	Count	Avg (ns)	Med (ns)	Min (ns)	Operation
93.7	492,135,180	37,656	13,069.2	3,232.0	576	[CUDA memcpy Host-to-Device]
6.3	33,279,350	18,828	1,767.5	1,600.0	576	[CUDA memcpy Device-to-Host]

Table 4: CUDA GPU Memory Operations Summary (by Time).

3.2.4 CUDA GPU MemOps Summary (by Size)

Total (MB)	Count	Avg (MB)	Med (MB)	Min (MB)	Max (MB)	Operation
5,203.583	37,656	0.138	0.000	0.000	51.552	[CUDA memcpy Host-to-Device]
30.853	18,828	0.002	0.000	0.000	0.307	[CUDA memcpy Device-to-Host]

Table 5: CUDA GPU Memory Operations Summary (by Size).

Based on this data, the GPU kernels are SIGNIFICANTLY faster than their CPU equivalents. The highest speedup occurs in _convolveImageVert which is a 120x speedup when comparing the CPU function to the GPU Kernel.
However, factoring in overhead from CUDA API calls shows that the overall program runs slower than the CPU program. With the overall runtime for GPU APIs being around 2.5 seconds and overall runtime for CPU execution being 0.07 seconds.
The overall speedup $\frac{2.5}{0.07} \approx 0.3$x indicating a slower execution for the naive GPU implementation.

4 Conclusion

A naive implementation of the KLT algorithm using CUDA slows down overall execution due to API overhead and inefficient memory accesses. The kernels themselves execute code faster but inefficient cudaMallocs and cudaMemcpys drastically slow down the program.
For a measurable increase in speedup memory optimizations are crucial.