

Assignment 1 Report: Parallel Vector Addition with PyCUDA and PyOpenCL

Weijie Wang

ww2739@columbia.edu

<https://forums.developer.nvidia.com/t/info-finished-with-code-256-error-install-of-driver-component-failed/107661>

<https://medium.com/@abhishekjainindore24/gpus-vs-cpus-threads-core-speed-75b5732ce389>

In this assignment, the introduction to PyCUDA and PyOpenCL was introduced, and the theory of the Parallel Vector Addition was learned. Compare and contrast different methods of host-to-device memory allocation for simple elementwise operations on vectors. Also, profiling is introduced.

First, modify the CudaModule.py to change the kernel and complete the methods in the file. In PyCUDA.py, run the program by setting the vector size up to 10^6 . As a result, the CPU_Loop_Add method is slower than the CPU_numpy_add method. Thus, it is removed, and re-run the program with the vector size up to 10^8 .

The output of the PyCuda.py is attached below. Execution time including and excluding memory transfer for add_gpuarray_kernel and add_gpuarray_no_kernel was included.

```
SSH-in-browser
cudamodule.py  PyOpenCL.py  clModule.py  cuda1.png  cuda_12.2.0_535.54.03_linux.run  cuda.cl  openc11.png
pyCUDA.py      pycache      cuda.png  cuda2.png  cuda_12.6.1_560.35.03_linux.run  openc1.png  openc12.png
(cuda.cl) ww2739@vm:~$ rm PyCUDA.py
(cuda.cl) ww2739@vm:~$ python PyCUDA.py
vectorlength
10
vectorlength
100
vectorlength
1000
vectorlength
10000
vectorlength
100000
vectorlength
1000000
Pass Vector and Number The CPU times are
[2.86102235e-06 2.75397787e-06 3.11403858e-06 4.87541666e-06
 2.33842149e-05 3.70302579e-04]
Pass Vector and Number The CPU Loop times are
[1.33855002e-05 3.94753047e-05 3.12634877e-04 3.07322035e-03
 3.01421905e-02 3.03554530e-01]
Pass Vector and Number The add_device_mem_gpu times including memory transfer are
[0.00021545 0.0002015 0.00021385 0.0002399 0.00042481 0.00219737]
Pass Vector and Number The add_device_mem_gpu times excluding memory transfer are
[4.38177960e-05 3.59203267e-05 3.65577146e-05 3.61782862e-05
 2.65259593e-05 8.37054697e-05]
Pass Vector and Number The add_host_mem_gpu times are
[0.00025987 0.0002555 0.00026327 0.00029112 0.00053121 0.00240284]
Pass Vector and Number The add_gpuarray_kernel times including memory transfer are
[0.00031314 0.00030597 0.00031329 0.00034029 0.00054803 0.00222962]
Pass Vector and Number The add_gpuarray_kernel times excluding memory transfer are
[3.24989387e-05 3.30442448e-05 3.28117550e-05 3.24995917e-05
 3.91353468e-05 8.13126532e-05]
Pass Vector and Number The add_gpuarray_no_kernel times including memory transfer are
[0.00250048 0.00030523 0.000315 0.00034189 0.00054639 0.00215003]
Pass Vector and Number The add_gpuarray_no_kernel times excluding memory transfer are
[2.25054626e-03 6.04538778e-05 6.05289797e-05 6.07020411e-05
 5.06435917e-05 1.95843266e-04]
vectorlength
10
vectorlength
100
vectorlength
1000
vectorlength
```

```
3.68478347e-05 5.38154524e-04]
Pass Two Vectors The CPU Loop times are
[1.49035940e-05 5.07199034e-05 4.28146246e-04 4.13263087e-03
 4.14749554e-02 4.12721395e-01]
Pass Two Vectors The add_device_mem_gpu times including memory transfer are
[0.00021862 0.00029854 0.00021257 0.00024742 0.00054292 0.00321491]
Pass Two Vectors The add_device_mem_gpu times excluding memory transfer are
[4.00698778e-05 3.65335510e-05 3.65479182e-05 3.64114288e-05
 2.40300406e-05 8.27611426e-05]
Pass Two Vectors The add_host_mem_gpu times are
[0.00028001 0.00026539 0.00026545 0.00030767 0.00064694 0.00359104]
Pass Two Vectors The add_gpuarray_kernel times including memory transfer are
[0.00035519 0.00033488 0.00033743 0.00036968 0.00067062 0.00323237]
Pass Two Vectors The add_gpuarray_kernel times excluding memory transfer are
[3.41485714e-05 3.08669387e-05 3.29201632e-05 3.18373878e-05
 3.46226940e-05 9.17622859e-05]
Pass Two Vectors The add_gpuarray_no_kernel times including memory transfer are
[0.00253435 0.00033068 0.00033545 0.00036806 0.00064256 0.00311585]
Pass Two Vectors The add_gpuarray_no_kernel times excluding memory transfer are
[2.24507232e-03 5.70115918e-05 5.77730612e-05 5.66909389e-05
 4.63183672e-05 1.97585632e-04]
vectorlength
10
vectorlength
100
vectorlength
1000
vectorlength
10000
vectorlength
100000
vectorlength
1000000
vectorlength
10000000
vectorlength
100000000
Pass Vector and Number The CPU times are
[3.68818945e-06 2.86102295e-06 3.59574143e-06 5.86801646e-06
 2.44062774e-05 3.54153769e-04 1.17091257e-02 1.35648319e-01]
Pass Vector and Number The add_device_mem_gpu times including memory transfer are
[2.19471601e-04 2.03229943e-04 2.16921982e-04 2.55900986e-04
 4.38062512e-04 1.9684318e-03 2.73350696e-02 2.54960011e-01]
Pass Vector and Number The add_device_mem_gpu times excluding memory transfer are
[4.08000001e-05 3.46710204e-05 3.84163266e-05 3.96244901e-05
 2.70602448e-05 4.28564899e-05 4.09242122e-04 3.44104815e-03]
[4.08000001e-05 3.46710204e-05 3.84163266e-05 3.96244901e-05
 2.70602448e-05 4.28564899e-05 4.09242122e-04 3.44104815e-03]
```

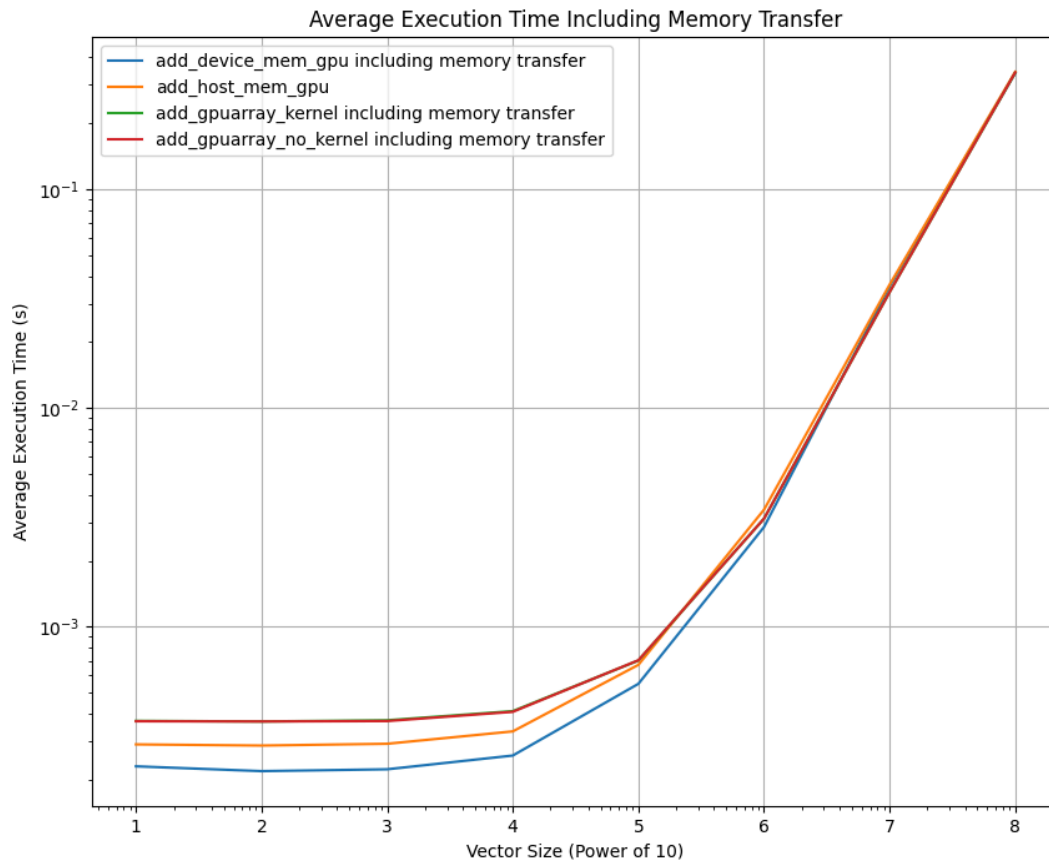
```
vectorlength
100000000
vectorlength
100000000
vectorlength
100000000
Pass Vector and Number The CPU times are
[3.68818945e-06 2.86102295e-06 3.59574143e-06 5.86801646e-06
 2.44062774e-05 3.54153769e-04 1.17091257e-02 1.35648319e-01]
Pass Vector and Number The add_device_mem_gpu times including memory transfer are
[2.19471601e-04 2.03229943e-04 2.16921982e-04 2.55900986e-04
 4.38062512e-04 1.9684318e-03 2.73350696e-02 2.54960011e-01]
Pass Vector and Number The add_device_mem_gpu times excluding memory transfer are
[4.08000001e-05 3.46710204e-05 3.84163266e-05 3.96244901e-05
 2.70602448e-05 4.28564899e-05 4.09242122e-04 3.44104815e-03]
Pass Vector and Number The add_host_mem_gpu times are
[0.00028793 0.00026129 0.0002687 0.00032401 0.00054361 0.00227012
 0.02632864 0.25584536]
Pass Vector and Number The add_gpuarray_kernel times including memory transfer are
[0.00033505 0.00030861 0.00032142 0.00035736 0.00056186 0.00210797
 0.02481667 0.25467244]
Pass Vector and Number The add_gpuarray_kernel times excluding memory transfer are
[3.72231837e-05 3.26380407e-05 3.34066939e-05 3.63232652e-05
 4.03768166e-05 7.46102861e-05 4.15410938e-04 3.43790237e-03]
Pass Vector and Number The add_gpuarray_no_kernel times including memory transfer are
[0.00033799 0.00030454 0.00032391 0.00035744 0.00056259 0.00211086
 0.02485947 0.25566942]
Pass Vector and Number The add_gpuarray_no_kernel times excluding memory transfer are
[7.00368985e-05 6.07471025e-05 6.30125714e-05 6.75892240e-05
 5.51118373e-05 1.88247511e-04 7.44867264e-04 4.60115534e-03]
vectorlength
10
vectorlength
100
vectorlength
1000
vectorlength
10000
vectorlength
100000
vectorlength
1000000
vectorlength
10000000
vectorlength
100000000
Pass Two Vectors The CPU times are
[1.80030356e-06 2.07278193e-06 2.43284264e-06 5.02138722e-06
```

```
SSH-in-browser
[7.0036895e-05 6.07471025e-05 6.30125714e-05 6.75892240e-05
5.51118373e-05 1.88247511e-04 7.44867264e-04 4.60115594e-03]
vectorlength
10
vectorlength
100
vectorlength
1000
vectorlength
10000
vectorlength
100000
vectorlength
1000000
vectorlength
10000000
vectorlength
100000000
Pass Two Vectors The CPU times are
[1.80030356e-06 2.07278193e-06 2.43284264e-06 5.02138722e-06
3.90957813e-05 5.04123921e-04 1.43322945e-02 1.65382818e-01]
Pass Two Vectors The add_device_mem_gpu times including memory transfer are
[2.30146914e-04 2.18766076e-04 2.23086805e-04 2.57798604e-04
5.49469777e-04 2.85591398e-03 3.59914011e-02 3.39715082e-01]
Pass Two Vectors The add_device_mem_gpu times excluding memory transfer are
[4.01338774e-05 3.95826937e-05 4.09776325e-05 4.15758368e-05
2.68310204e-05 5.20620409e-05 5.28542042e-04 4.66641367e-03]
Pass Two Vectors The add_host_mem_gpu times are
[2.89936455e-04 2.86423430e-04 2.91790281e-04 3.32409022e-04
6.69688595e-04 3.42806018e-03 3.66120338e-02 3.43867093e-01]
Pass Two Vectors The add_gpuarray kernel times including memory transfer are
[0.00037125 0.00036886 0.00037357 0.00041132 0.0007009 0.00312066
0.03388042 0.34073543]
Pass Two Vectors The add_gpuarray kernel times excluding memory transfer are
[3.62102859e-05 3.60927351e-05 3.43973877e-05 3.59582043e-05
3.92450613e-05 0.69153469e-05 5.34075427e-04 4.65507918e-03]
Pass Two Vectors The add_gpuarray_no_kernel times including memory transfer are
[0.00036981 0.00036894 0.00037056 0.00040842 0.00070291 0.00311925
0.03384898 0.34182631]
Pass Two Vectors The add_gpuarray_no_kernel times excluding memory transfer are
[6.4895346e-05 6.53231020e-05 6.29263668e-05 6.44101221e-05
5.06586124e-05 2.01225144e-04 8.45788737e-04 6.05096945e-03]
(cuda_cl) ww2739@vm:~$ ls
CudaModule.py  PyOpenCL.py  clModule.py  cuda1.png  cuda_12.2.0_535.54.03_linux.run  cuda.cl  opencl1.png
pyCUDA.py      pycuda.py    cuda.png     cuda2.png  cuda_12.6.1_560.35.03_linux.run  opencl.png  opencl2.png
(cuda_cl) ww2739@vm:~$
```

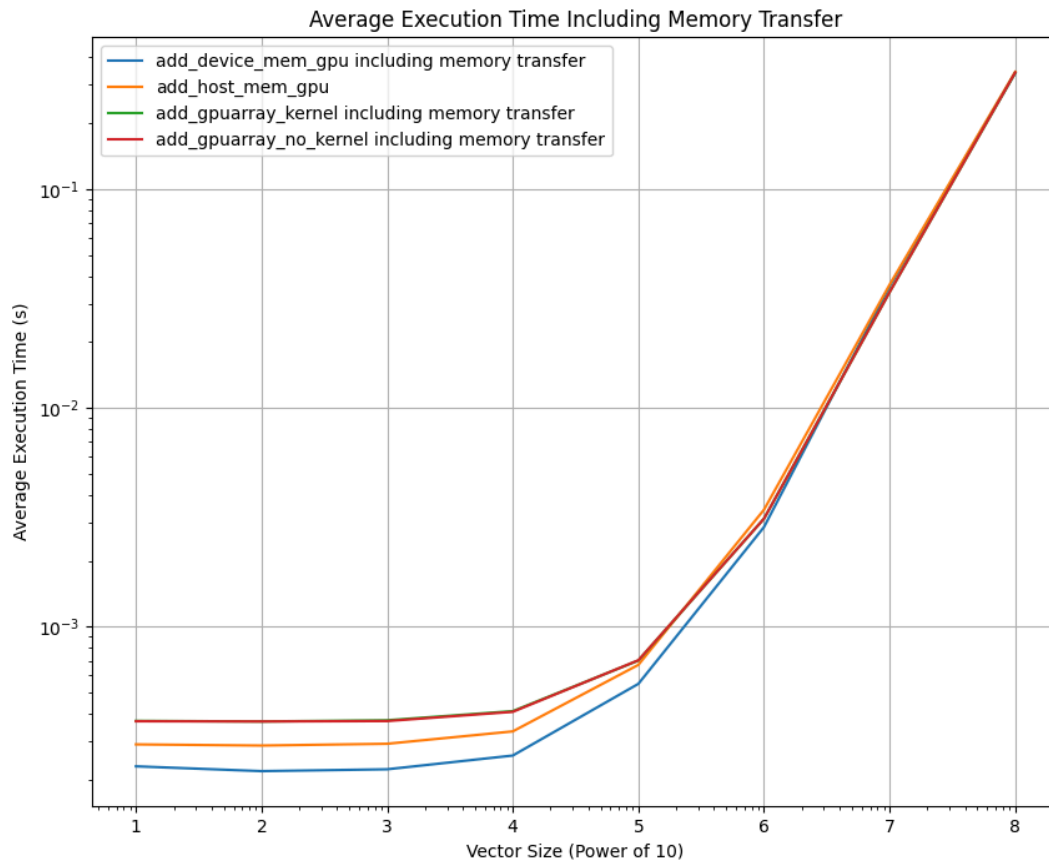
PyCUDA code output

For the `add_host_mem_gpu` method, the reason why there is no execution time without memory transfer is that we can only measure the total execution time because the data transfers and kernel execution are inseparable in this method. The `add_gpuarray_no_kernel` method is slower than `CPU_numpy_add` with small vector sizes. But with the increase of vector sizes, the `add_gpuarray_no_kernel` method is faster than the `CPU_numpy_add` method.

The plots for the average execution times including and excluding memory transfer for GPU operations against vector size are attached below.



Plot of the average execution times, including memory transfer for GPU operations, against vector size



Plot of the average execution times, excluding memory transfer for GPU operations, against vector size

Then I profile `add_device_mem_gpu` method for both `Add_to_each_element_GPU` and `Add_two_vectors_GPU` kernels. The profiling file is uploaded to GitHub and the screenshot of the Nsight Compute UI is attached to the report.

SSH-in-browser

UPLOAD FILE

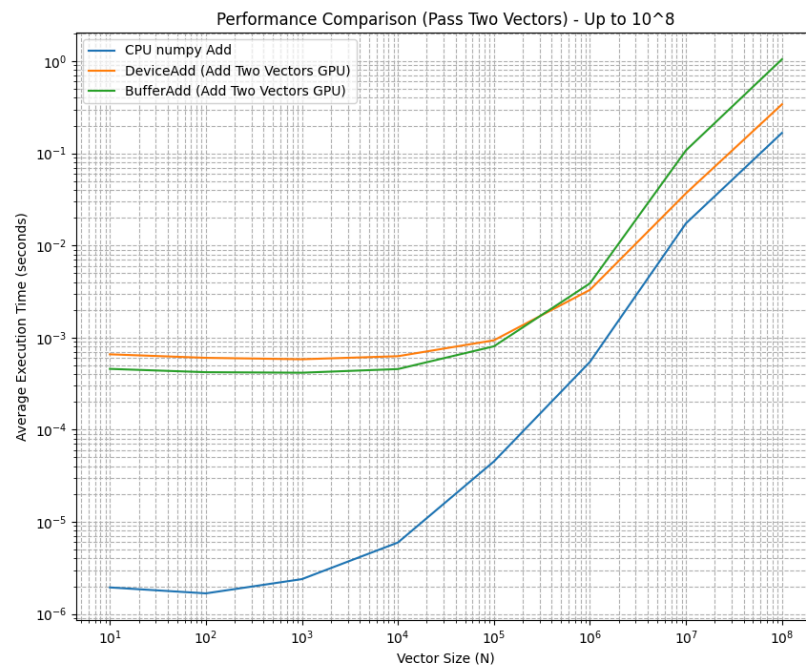
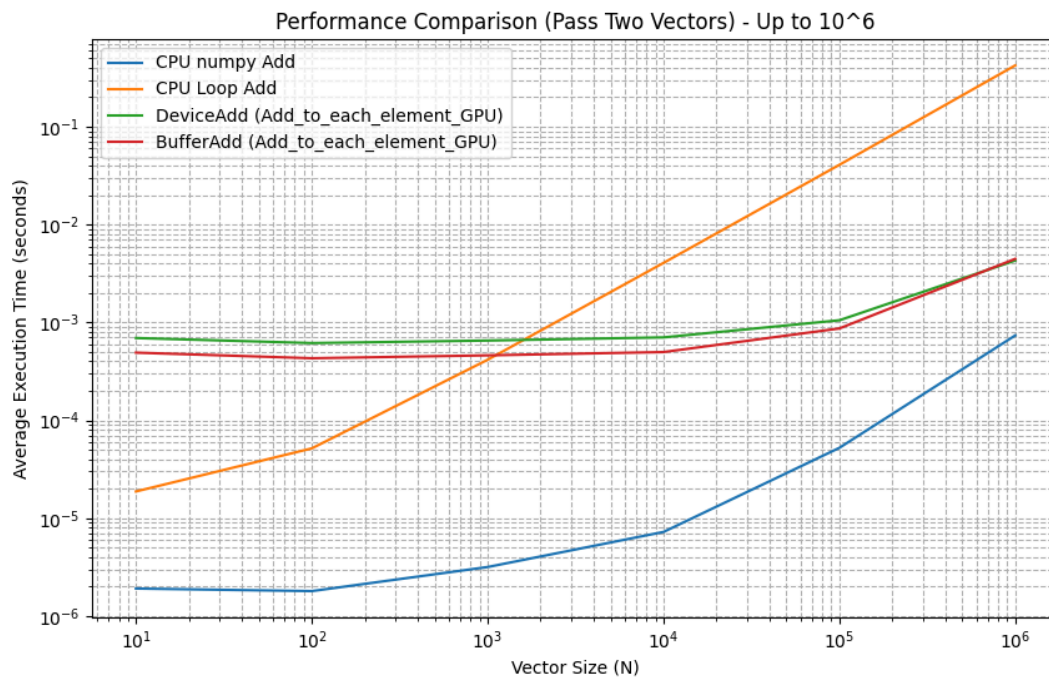
DOWNLOAD FILE

SSH-in-browser

[cuda_cli] **wz739@wzm:**\$ rm PyOpenCL.py
[cuda_cli] **wz739@wzm:**\$ python PyOpenCL.py
Vector Length: 10
Vector Length: 100
Vector Length: 1000
Vector Length: 10000
Vector Length: 100000
Vector Length: 1000000
Pass Vector and Number - CPU numpy Add times:
[3.0848447e-06 3.57141300e-06 4.64672945e-06 7.84835037e-06 4.29986069e-05 4.57111670e-04]
Pass Vector and Number - CPU Loop Add times:
[1.55993870e-05 4.05749496e-05 3.13306341e-04 2.94777812e-03 2.9334932e-02 2.99596299e-01]
Pass Vector and Number - DeviceAdd times:
[0.0006372 0.00052089 0.0005731 0.0006065 0.00078855 0.00303874]
Pass Vector and Number - BufferAdd times:
[0.00043584 0.0042034 0.00045826 0.00048 0.00069922 0.00273079]
Vector Length: 10
Vector Length: 100
Vector Length: 1000
Vector Length: 10000
Vector Length: 100000
Vector Length: 1000000
Pass Two Vectors - CPU numpy Add times:
[1.91214142e-06 1.80030356e-06 3.17242681e-06 7.24013971e-06 5.23645031e-05 7.37316754e-04]
Pass Two Vectors - CPU Loop Add times:
[1.87669482e-05 5.17125032e-05 4.16001495e-04 4.06034139e-03 4.07955306e-02 4.23085532e-01]
Pass Two Vectors - DeviceAdd times:
[0.00069337 0.00061586 0.00065351 0.00070394 0.00105269 0.0042931]
Pass Two Vectors - BufferAdd times:
[0.00049165 0.00043101 0.00046136 0.00049856 0.00086848 0.00446633]
Vector Length: 10
Vector Length: 100
Vector Length: 1000
Vector Length: 10000
Vector Length: 100000
Vector Length: 1000000
Pass Vector and Number The CPU Times are:
[3.0848447e-06 3.06538173e-06 4.11636975e-06 6.56380945e-06 3.61812358e-05 3.81459995e-04 1.44417821e-02 1.38209742e-01]
Pass Vector and Number The Device Add Times are:

[1.87669482e-05 5.17125032e-05 4.16001495e-04 4.06034139e-03 4.07955306e-02 4.23085532e-01]
Pass Two Vectors - DeviceAdd times:
[0.00069337 0.00061586 0.00065351 0.00070394 0.00105269 0.0042931]
Pass Two Vectors - BufferAdd times:
[0.00049165 0.00043101 0.00046136 0.00049856 0.00086848 0.00446633]
Vector Length: 10
Vector Length: 100
Vector Length: 1000
Vector Length: 10000
Vector Length: 100000
Vector Length: 1000000
Pass Vector and Number The CPU Times are:
[3.0848447e-06 3.06538173e-06 4.11636975e-06 6.56380945e-06 3.61812358e-05 3.81459995e-04 1.44417821e-02 1.38209742e-01]
Pass Vector and Number The Device Add Times are:
[0.00055945 0.00054788 0.00055278 0.00056915 0.00078003 0.00243015 0.02767219 0.2547967]
Pass Vector and Number The Buffer Add Times are:
[4.50470010e-04 4.35848625e-04 4.43959723e-04 4.69718661e-04 7.03714032e-04 2.53846207e-03 6.26552932e-02 6.07721674e-01]
Vector Length: 10
Vector Length: 100
Vector Length: 1000
Vector Length: 10000
Vector Length: 100000
Vector Length: 1000000
Pass Two Vectors The CPU Times are:
[1.94140843e-06 1.67379574e-06 2.38905148e-06 5.96533016e-06 4.52460075e-05 5.44951887e-04 1.73759266e-02 1.66550602e-01]
Pass Two Vectors The Device Add Times are:
[0.0006581 0.00060281 0.00058172 0.00062758 0.00093448 0.00329504 0.03679529 0.3402311]
Pass Two Vectors The Buffer Add Times are:
[4.57238178e-04 4.20998554e-04 4.15573315e-04 4.55145933e-04 8.04000971e-04 3.87593678e-03 1.07683021e-01 1.04869595e+00]
[cuda_cli] **wz739@wzm:**\$ ls
CUDADoutput.txt PyOpenCL.py cudal.png opencl1.png
Documents CModule.py _pycache_ cudal2.png cudaprofile.py opencl2.png
PyCUDA.py cModule.py cudal_12.2.0_535.54.03_linux.run metrics.ncu-rep output.txt
[cuda_cli] **wz739@wzm:**\$ cd
cudal_12.6.1_560.35.03_linux.run opencl.png profile.ncu-rep

The Plot of the average execution times for the GPU cases and the faster CPU case against vector size is attached below.



The average execution times for the four GPU cases and the faster CPU case against vector size

Theory

In a CPU, a thread is the smallest unit that shares memory and resources of the process which enable parallelism in multi-core CPUs. In GPUs, thread is a register per work item which is ideally low and a nice divisor of the number of hardware registers per computer unit. A task is a unit of work that is scheduled for execution. In a CPU, each core is capable of handling complex tasks independently, with a focus on general-purpose computing. In a GPU, it comprises thousands of smaller cores designed to handle multiple tasks simultaneously. A process is an independent execution unit that has its own memory space. In a CPU, processes rely on context switching for parallelism while in a GPU, a process can spawn multiple threads to run different computations in parallel.

The `CPU_numpy_Add` method is proved to be faster for CPU methods. It comes with a smaller processing time because it takes advantage of NumPy's optimized, low-level implementations for array operations which maximize the performance of operations. Meanwhile, `CPU_loop_Add` is slower due to the inefficiency inherent in executing explicit loops.

These parallel approaches in PyCUDA and PyOpenCL can perform thousands of operations simultaneously while the CPU methods require multi-threading and multi-processing, which limit the number of operations that happen simultaneously. All in all, parallel approaches in PyCUDA and PyOpenCL bring more computations at the same time and are more efficient in dealing with large workloads.

In PyOpenCL, the `deviceAdd` method is faster as indicated in the previous results output with the increase of vector sizes. It takes advantage by making all operations performed on the GPU, while host-to-device and device-to-host memory transfers are much slower than directly on the device.

In PyCUDA, the `add_gpuarray_kernel` is the fastest as indicated in the previous results. The reason it is faster than the other methods is that it takes advantage of the overhead of custom kernel compilation and launches, utilizes efficient memory management provided by `gpuarray`, and has fewer layers of abstraction.

I would prefer PyCUDA. Since PyCUDA targets NVIDIA GPUs and we're using NVIDIA T4 GPUs, it may be more compatible with the platform while we're coding. PyCUDA is optimized for NVIDIA GPUs which brings high performance for specific tasks. PyCUDA also seems to be easier to ease than PyOpenCL with easier access to highly optimized libraries.