# Assignment 4 CUDA HOST AND GLOBAL MEMORY

# Overview

There are two main files

|  |  |
| --- | --- |
| File Name | Purpose |
| **pageableHostmem\_arithmetics\_fake2D.cu** | Run math operations defined in assignment3. Data copy from host pageable memory into device memory. The array is dynamically allocated by malloc which is different from assignment 3. It is called fake2D because we allocated memory for 1D but pretend that we are dealing with a 2-D memory. See method 1 in [How to dynamically allocate a 2D array in C? - GeeksforGeeks](https://www.geeksforgeeks.org/dynamically-allocate-2d-array-c/) |
| **pinnedHostmem\_arithmetics\_fake2D.cu** | Run math operations defined in assignment3. Data copy from host pinned memory into device memory. |

There are bunch of other .cu files. pinnedHostmem\_arithmetics\_2D.cu is very similar to pinnedHostmem\_arithmetics\_fake2D.cu. The only difference is that it truly allocate memory for 2D and enable the standard 2D array accessing arr[i][j]. See method 2 and 3 in [How to dynamically allocate a 2D array in C? - GeeksforGeeks](https://www.geeksforgeeks.org/dynamically-allocate-2d-array-c/). The assignment.cu is pretty similar to pinnedHostmem\_arithmetics\_2D except it is just using the pageable memory. And it is just to make Vocareum happy.

The main analysis and execution will be using the **pageableHostmem\_arithmetics\_fake2D.cu and pinnedHostmem\_arithmetics\_fake2D.cu.**

|  |
| --- |
| Diagram  Description automatically generated |
| Figure 1 pinned memory concept illustration |

As shown in Figure 1, even if we do not use the cudaMallocHost(), what end up happening is that before transferring data from cpu to gpu, it will create a pinned memory, copy data from pageable memory into that pinned memory first, then transfer from pinned memory (buffer) to the GPU. This buffer behavior can be avoided if we use cudaMallocHost() to create the source memory itself as the pinned memory.

Methods for memory management in Cuda runtime.

* **Malloc()/free() for pageable host memory**
* **cudaMallocHost()/cudaFreeHost() for host pinned memory**
* **cudaMalloc()/cudaFree() for memory on GPU**

A run.sh has been provided as a sequence of commandline commands to

* Build appropriate executable
* run to couple of tests.

# Tests and Result Discussion

## Test 1: four previous math operations using pageable host memory and device memory

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Table 1 Summary of execution time with different total number of threads&data (fixed block size=256) using host pageable memory | | | | |
| Threads\*  Operation | 256 | 1024 | 32768 | 1048576 |
| Add | 25259 ns | 25220 ns | 25975ns | 27974ns |
| subtract | 9507ns | 13694 ns | 9481ns | 11149ns |
| multiply | 7885ns | 10903 ns | 7596ns | 9234ns |
| mod | 7162 ns | 9895 ns | 6483ns | 8387ns |
| Host mem allocation | 1418 ns | 11761 ns | 10791 ns | 17708ns |
| GPU mem allocation | 291250531 ns | 243632257 ns | 265683185 ns | 270172914 ns |
| Mem copy from cpu to gpu | 38505 ns | 44204ns | 108535 ns | 153053 ns |

\*The total number of threads does not necessarily needs to be equal to the total number of data. In the real world applications. Data size can be much larger than the total number of threads. Just for the sake of simplicity, in my code in this assignment, I assume the array size will be same as the total num of threads.

From this result, it is quite impressive that the actual computation time did not increase with the increase of the data size at all. However, we should notice that there is overhead in terms of moving data between GPU and CPU.

## Test 2: four previous math operations using pinned host memory and device memory

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Table 2 Summary of execution time with different total number of threads&data (fixed block size=256) using host pinned memory | | | | |
| Threads\*  Operation | 256 | 1024 | 32768 | 1048576 |
| Add | 25238 ns | 26090 ns | 30960ns | 27119 ns |
| subtract | 7902 ns | 7423 ns | 10122ns | 9031ns |
| multiply | 7341 ns | 7449 ns | 11119ns | 8885ns |
| mod | 5942 ns | 6449 ns | 8019 ns | 6537ns |
| Host mem allocation | 2918305859 | 2659140658 | 2930064568 | 2638803269 |
| GPU mem allocation | 150277 | 145044 | 153998 | 367959 |
| Mem copy from cpu to gpu | 43318 ns | 49789 ns | 82374 ns | 726492 ns |

One observation is that the time spent on Host mem allocation becomes much longer than that of the pageable case. However, we can notice that the time of device mem allocation is greatly reduced compared to that of the pageable case. This proves that even by using the pageable case, when we are doing the device memory allocation, it will try to create the buffer which might take the same amount of time.

Another thing is that memory copy from cpu to gpu seems faster, especially when the data size to be copied is larger.

I did not observe performance increase in terms of the actual computing.

# Summary of future improvements

1. Looks like the timing profiling tool is not that accurate. Maybe we should try to learn how to use nvprof
2. It looks like that if we need to copy data on and off gpu frequently, we should use pinned host memory. Needs to confirm this during office hour.