EE 382C: Multicore Computing
Parallel GPU based Algorithms for Image Processing

Wenbo Xu, Wenwen Zhang, Yichi Zhang

I. Abstract

II. Introduction

III. Gaussian Filter

IV. Optimization

A. Pageable vs. Pinned Memory

Host data allocations are pageable by default, which means can be paged in/out between RAM and disk. However, GPU cannot access data directly from pageable memory, but from pinned memory, which means page-locked. Hence, whenever a data transfer is invoked on pageable memory, the CUDA driver has to allocate a temporary pinned memory array to copy host data and then transfer it to the device.

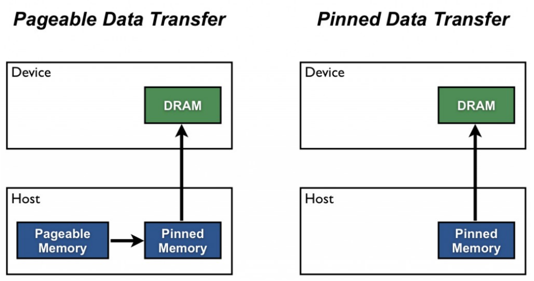

Fig. 1: CUDA data transfer
[1]

We can avoid the cost of this overhead by using pinned memory for host instead of pageable memory. In this case, we use *cudaMallocHost()* and *cudaFreeHost()*. Compare to the *malloc()* and *free()*, *cudaMallocHost()* and *cudaFreeHost()* are more expensive with additional overheads. Then, the question has been raised about how should we made the tradeoff. According to figure below, pinned memory is faster when the size of data to be transfered is larger than 16MB.

This doesn't means we should never use pinned memory when the amount of data to be transfered is less than 16MB. One example is the asynchronous memory copy, *cudaMemcpyAsync()* can be used only with pinned memory. The details of how asynchronous memory copy would be used to improve the efficiency will be discussed in next section.

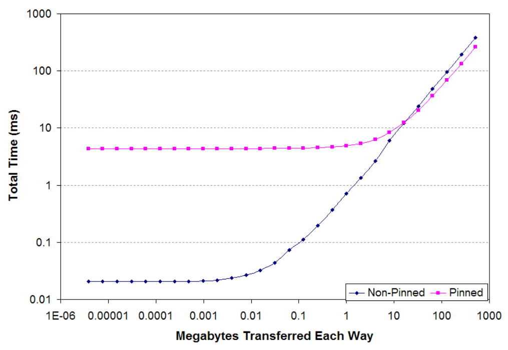

Fig. 2: Time required to allocate, transfer to the GPU, transfer back to the CPU, and deallocate pinned and non-pinned memory.
[2]

B. Streams

References

[1] Harries, M. (2012, December). How to Optimize Data Transfers in CUDA C/C++. Retrieved from https://devblogs.nvidia.com/parallelforall/how-optimize-data-transfers-cuda-cc/

[2] Boyer, M. Choosing Between Pinned and Non-Pinned Memory. Retrieved from https://www.cs.virginia.edu/ mwb7w/cuda˙support/pinned˙tradeoff.html