Cairo University
Faculty of Engineering
Spring 2025

Credit Hours System

Parallel Processing

Parallel Processing Report

Prefix Sum

Submitted to: Eng Mohamed Qotb
Submitted on: 25/04/2025

Submitted by:

Basim Sherief Zeenelabdeen	CCEC	1210

Compare work efficient and work inefficient implementations.

Note for all comparisons I have used dataset of sizes 1 mil and 50 thousand elements

	Kernel	Cuda mem copy D2H
EfficientPrefixSumUlt	2807461	2275038.0
InefficientPrefixSumUlt	2992744	1682007

The work efficient kernel is normally more desirable since its work efficient so its better energy efficiency and require less execution resource but

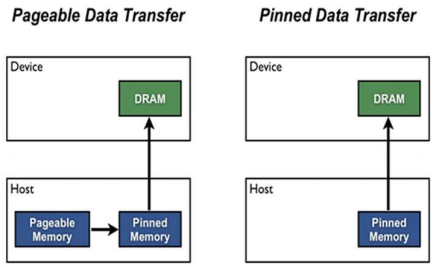
The work inefficient kernel could be better when I tested with small sizes but the requirement was to calculate accumulated sum of input array more than 1 mil

Compare work efficient But using different memories

Speedup relative to pageable memory pinned memory had 36.58x while unified memory 1.54 Zero mapped memory was the worst, I know pinned memory is much faster

	Total time from start of allocating memory executing the kernel and memcpy back to write in file **in ms**
Pageable memory	419.235
Pinned memory	11.460
Unified memory	271.782
Zero mapped memory	428.450

this is the expected because we remove overhead of additional copying of data instead from pageable memory to pinned we copy directly to pinned memory. When I searched there was method called prefetch that could further increase the unified memory performance but I commented it because we didn't take it in lecture



also note there is also memcpy in unified memory and this does not make sense but I do it because I send the same pointer to writing output function so don't penalize me

Compare between thread coarsening and using streams

To output correct prefix sum In case of streams I needed to add another kernel called adjustStreamValues which takes the last sum from the stream and then add it to other streams to produce correct output but its needless to do it in case of thread Coarsening

	Efficient prefix sum Kernel	AdjustStreamValues	cudaStreamCreate	cudaMemcpy
Using Streams	2792388	11809	626830698	3719053
Thread Coarsening	891531			4785854

My conclusion to the results is that we should not use streams unless there are huge amount of data at least to worth the overhead of creating streams as we see it takes ages to be created that's why thread coarsening win against streams in that amout of data 1 mil and 50 thousands

Thank you