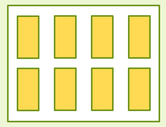
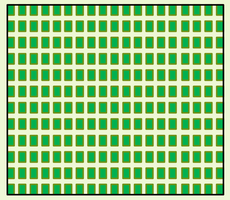
A Framework for External Memory RAM Oblivious GPU Algorithms

Rezaul Chowdhury, Rathish Das, Pramod Ganapathy, Mohammad Mahdi Javanmard and Stephen Tschudi
Stony Brook University

Motivation



CPU: Multiple general purpose cores

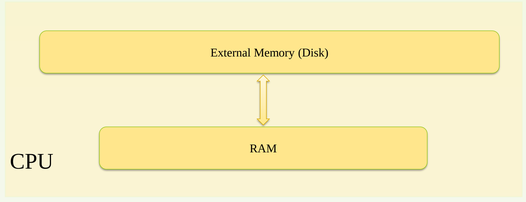


GPU: Thousands of cores good for SIMD operations

GPU gives a significant edge over CPU for many scientific computation as GPU has thousands of cores in comparison to CPU's few cores.

Q. What happens if the data doesn't fit in RAM?

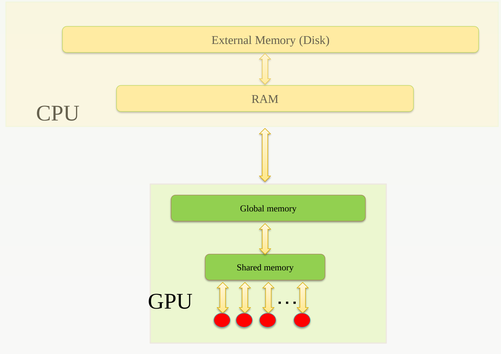
A. Till date there is no framework which use GPU with external memory.



External memory model

Our Contribution

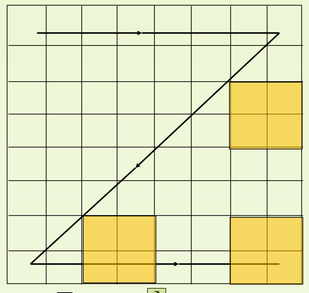
We present the first framework which combines GPU and external memory model.



External memory - GPU model

The Framework in Action

Solving a Dynamic Programming (DP) problem using our framework :

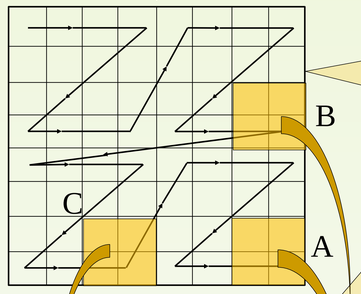


External memory

Input Parameters:
M = Input DP table of size n×n in external memory.
GG = GPU global memory size.
GS = GPU shared memory size.

GPU global memory needs to hold 3 sub-matrices each of size g×g where g = $\sqrt{GG/3}$

Z-Morton layout



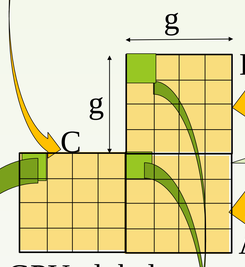
We divide M into 4 sub-matrices of size $\frac{n}{2}\times\frac{n}{2}$

Recursively divide M till the sub-matrices size become g×g. (2-way recursive DP).

External memory

Copy A,B and C (g×g) from RAM to GPU global memory.

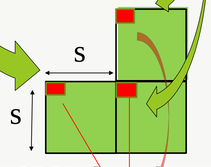
GPU shared memory needs to hold 3 sub-matrices each of size s×s where s = $\sqrt{GS/3}$



GPU global memory

Each of A, B and C is divided into r×r sub matrices of size s×s where r = $\frac{g}{s}$ (r-way recursive DP).

Copy 3 such s×s sub-matrices from GPU global to GPU shared memory.



GPU shared memory

Launch GPU kernels from shared memory.

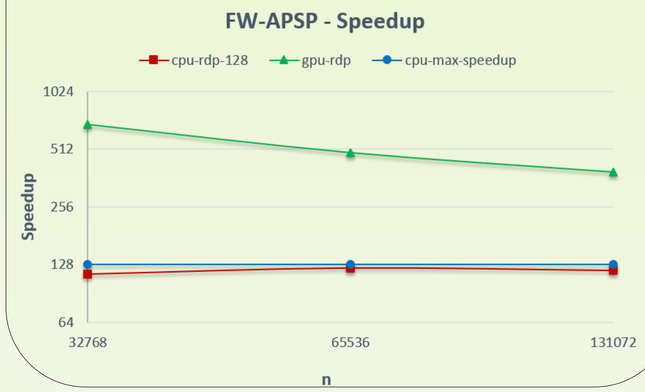
GPU threads

Experimental Result

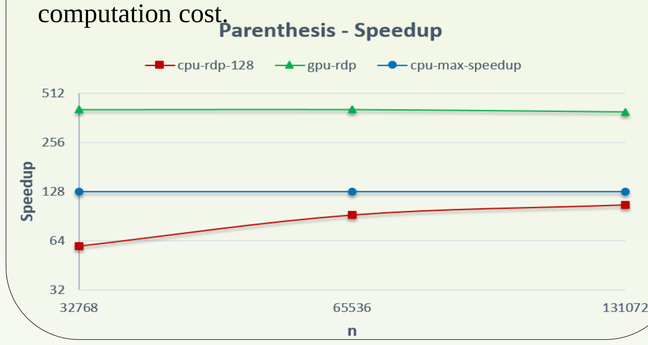
We present two DP problems where the input DP table doesn't fit in RAM.

In both problems GPU performs at least 2 times faster than a 128 core CPU with full parallelism

FW-APSP: Given a graph G, find the shortest path between any pair of vertices.



Parenthesis Problem: Given a sequence of matrices, find the optimum order of multiplication to minimize computation cost.



Q. Why don't we use multiple cores in CPU?
A. Compute bound DP problems similar to grid computation runs faster in GPU compared to multi-core CPU.

Q. Why don't we use iterative algorithm?
A. Solving the problem iteratively will incur more cache misses which will dominate the total computation cost.