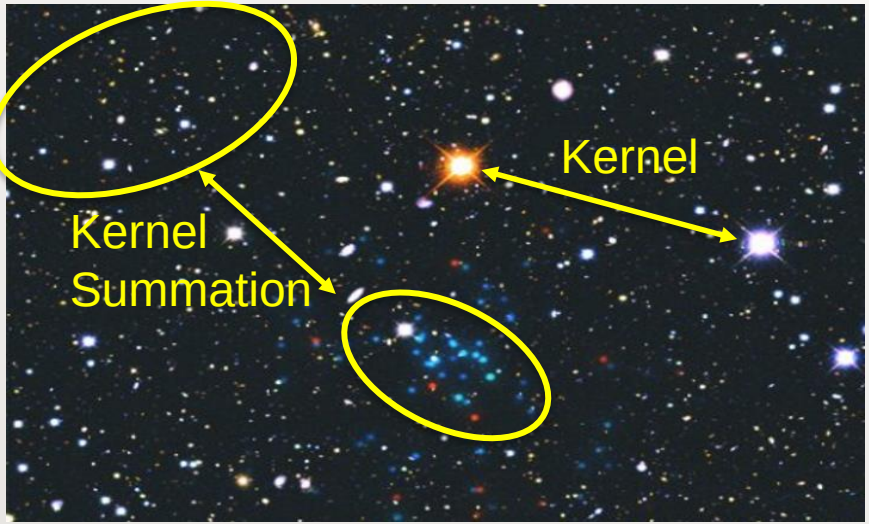
Optimizing GPGPU Kernel Summation for Performance and Energy Efficiency

Jiajun Wang, Ahmed Khawaja, George Biros, Andreas Gerstlauer, Lizy K. John

The University of Texas at Austin

Introduction

- Fundamental problem in computational physics, statistics, machine learning tasks.
- What is Kernel and what is Kernel Summation?



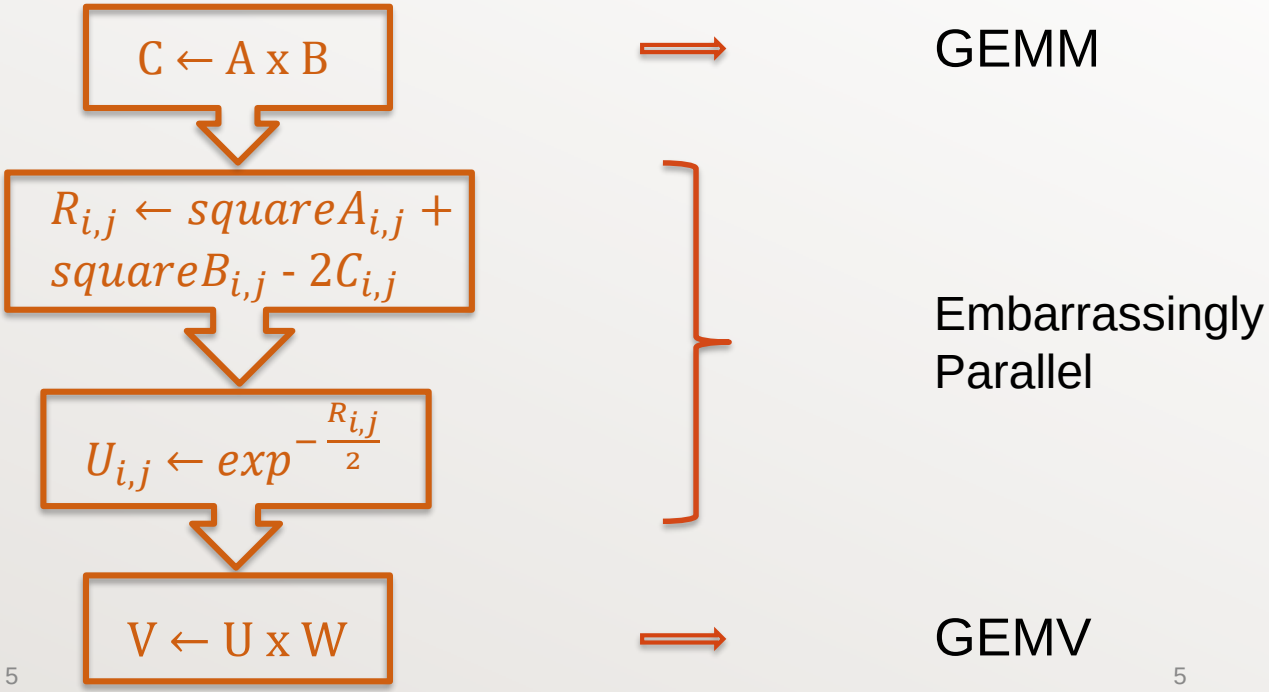
Related Work

- Tree codes, fast multipole methods, Ewald sums
  ++ scale to billions or trillions of points by
  reducing complexity O(N^2) to O(N log$N$)
  -- work for problems in *low dimensions (two or three),*
  not suitable for solving statistics, machine learning task
- Rely on **General Matrix Matrix Multiplication (GEMM)** for high dimensional tasks

Kernel Summation Steps

- Denotation:
  - *M*: number of points in target set
  - *N*: number of points in source set
  - *K*: dimension
- Inputs:
  - ***Target matrix A***: M-by-K
  - ***Source matrix B***: K-by-N
  - ***Weight vector W***: N-by-1
- Gaussian Kernel: $\mathbb{K}(\alpha, \beta) = exp^{-\frac{\|\alpha-\beta\|_2^2}{2}}$
- Output:
  - ***Vector V***: M-by-1

Kernel Summation Steps

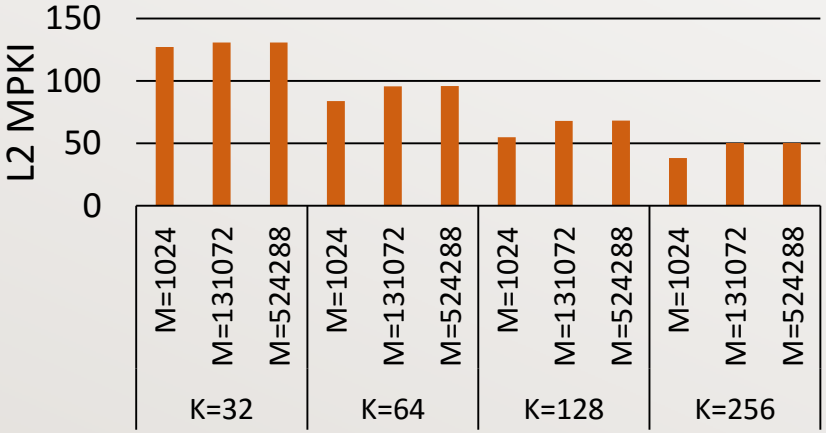


5

Implementation Based on cuBLAS

1. $C \leftarrow A \times B$
2. $R_{i,j} \leftarrow squareA_{i,j} + squareB_{i,j} - 2C_{i,j}$
3. $U_{i,j} \leftarrow exp^{-\frac{R_{i,j}}{2}}$
4. $V \leftarrow U \times W$

Call cuBLAS
++ Fast. Easy to use
-- Sacrifice data locality
-- Waste energy on DRAM accesses



High L2 MPKI indicates opportunity for fusion

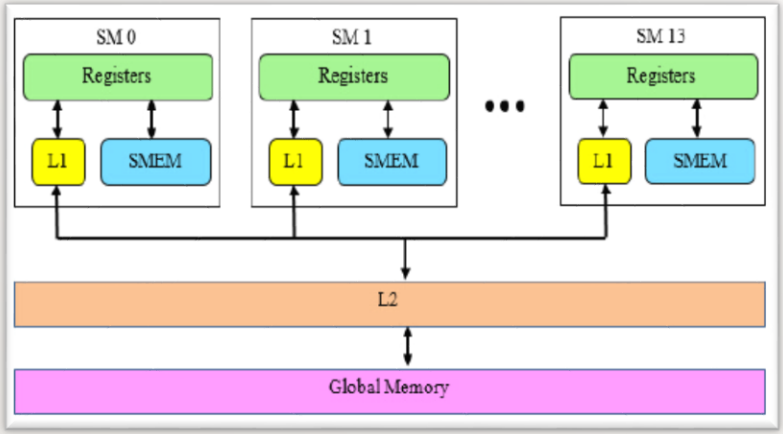
We propose Fused Kernel Summation

FOR each thread DO

1. reg_μC = GEMM(mem_subA, mem_subB)
2. reg_μC = Gaus (mem_subA2, mem_B2, reg_μC)
3. mem_subV = Summation(reg_μC, mem_subW)

GPU Background

- Thread: organized in thread blocks
- Thread block: executed by a SM (Streaming Multiprocessor)
- Warp: basic scheduling unit, 32 threads, execute same instruction in lock-step



Fused Kernel Summation

FOR each thread DO

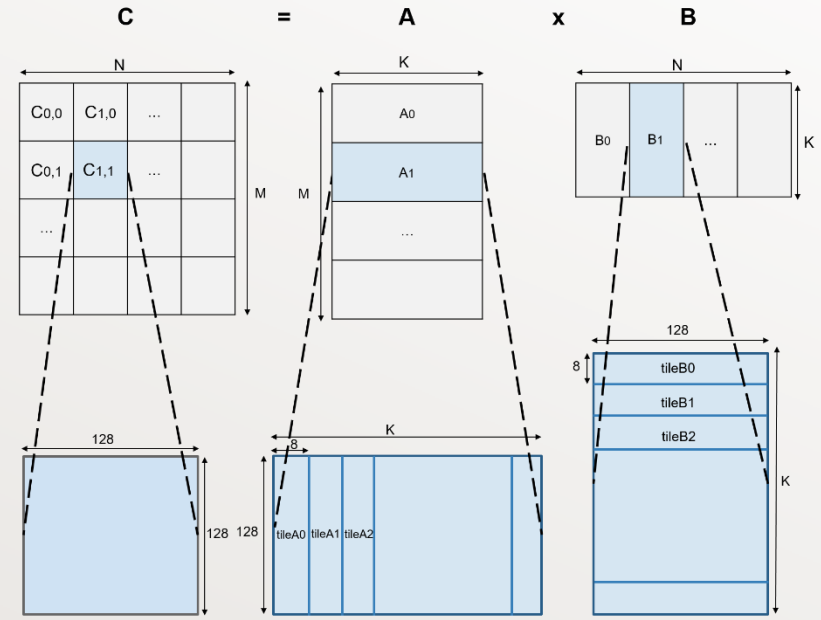
1. reg_µC = GEMM(mem_subA, mem_subB)
2. reg_µC = Gaus (mem_subA2, mem_B2, reg_µC)
3. mem_subV = Summation(reg_µC, mem_subW)

GEMM Algorithm Overview

- A thread block (bx,by) computes

$$submatrixC_{bx,by} = submatrixA_{by} \times submatrixB_{bx}$$

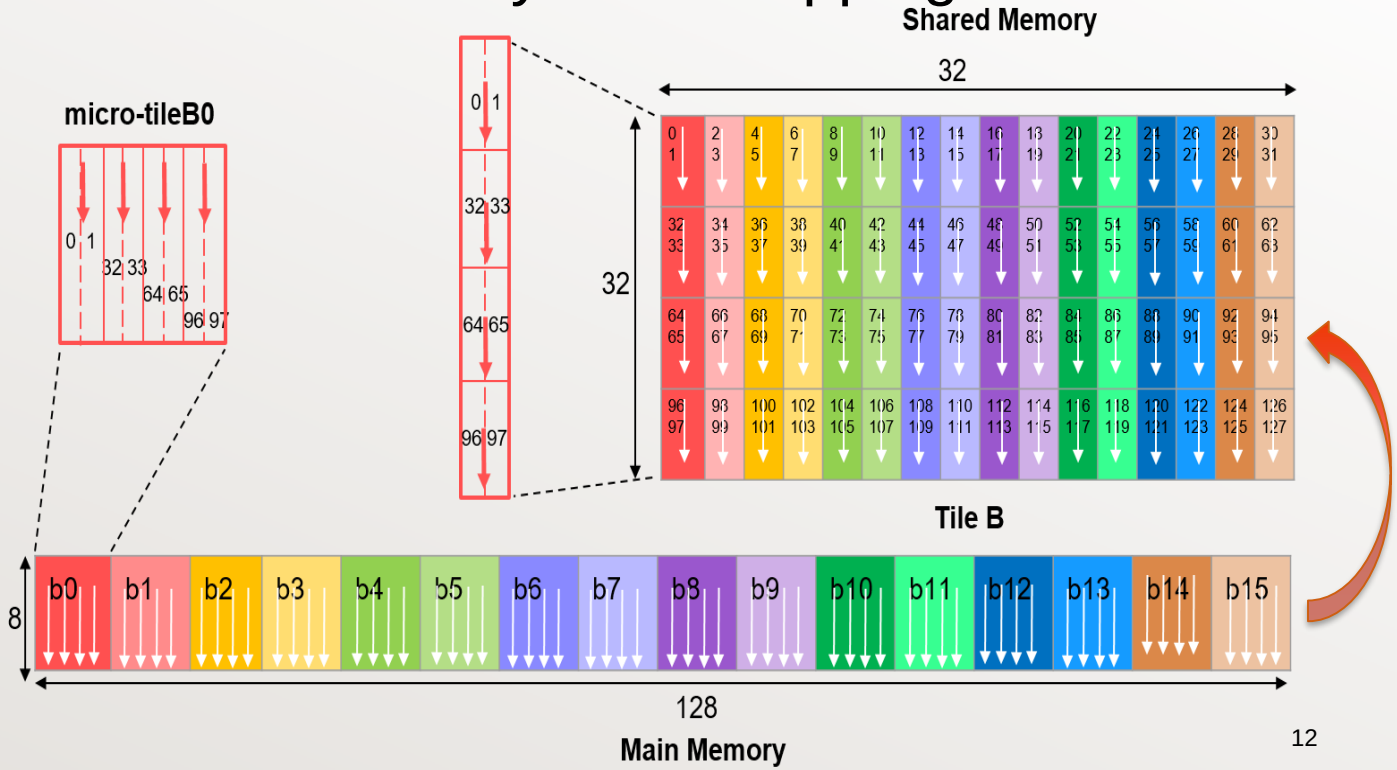
- Carefully select submatrix size and thread block size
- Overlap memory read latency with computation
- Rearrange data location for fast access



Shared memory (SMEM)

- Programmer managed cache
- 32 banks to be accessed simultaneously
- Serialize accesses when there's bank conflict

Shared Memory Data Mapping



Fused Kernel Summation

FOR each thread DO

1. reg_µC = GEMM(mem_subA, mem_subB)
2. reg_µC = Gaus (mem_subA2, mem_B2, reg_µC)
3. mem_subV = Summation(reg_µC, mem_subW)

Fused Kernel Summation

FOR each thread DO

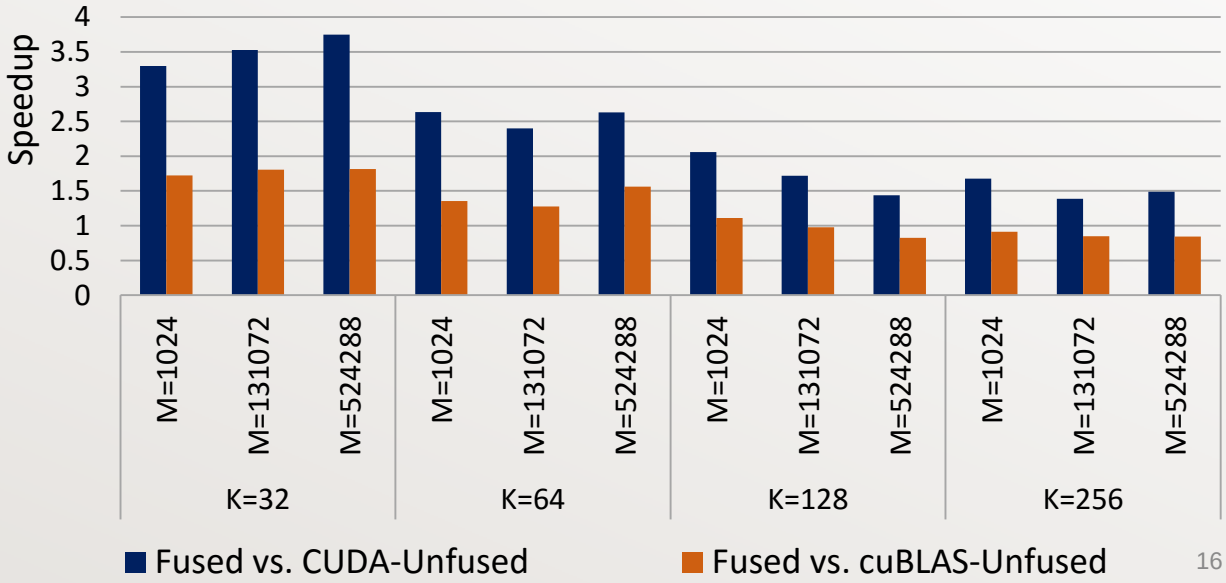
1. reg_μC = GEMM(mem_subA, mem_subB)
2. reg_μC = Gaus (mem_subA2, mem_B2, reg_μC)
3. mem_subV = Summation(reg_μC, mem_subW)
   - Intra thread level
     - Register → Shared memory
   - Intra thread block level
     - Shared memory → DRAM
   - Inter thread block level
     - DRAM → DRAM

Evaluation

- Infrastructure
  - Evaluate on NVIDIA GTX970
  - Profile tool: nvprof
  - cuBLAS library in version 7.0
- Experiments
  - Fused: fuse our own GEMM implementation with the kernel evaluation and the summation routine.
  - CUDA-Unfused: call our own GEMM implementation followed by the kernel evaluation and the summation routine.
  - cuBLAS-Unfused: call cuBLAS GEMM function followed by the kernel evaluation and the summation routine.

Performance Comparison

- Fused beats cuBLAS-Unfused by up to 1.8X speedup when dimension K < 128.



Influence on memory

- Fused optimization reduces memory transactions
  - Fused reduces 50% of L2 accesses in cuBLAS-Unfused
  - Fused reduces 90% of DRAM accesses in cuBLAS-Unfused

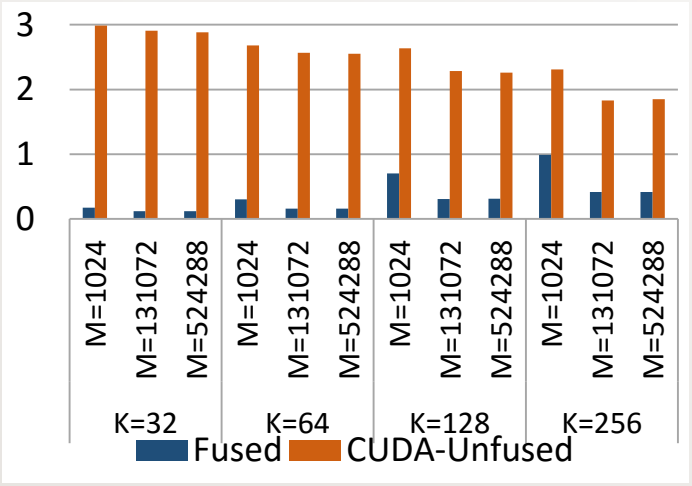


Fig. a: L2 Accesses normalized to cuBLAS-Unfused

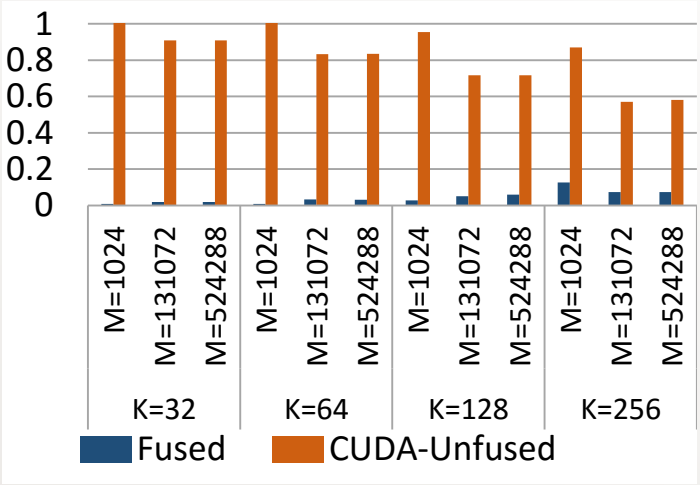


Fig. b: DRAM Accesses normalized to cuBLAS-Unfused

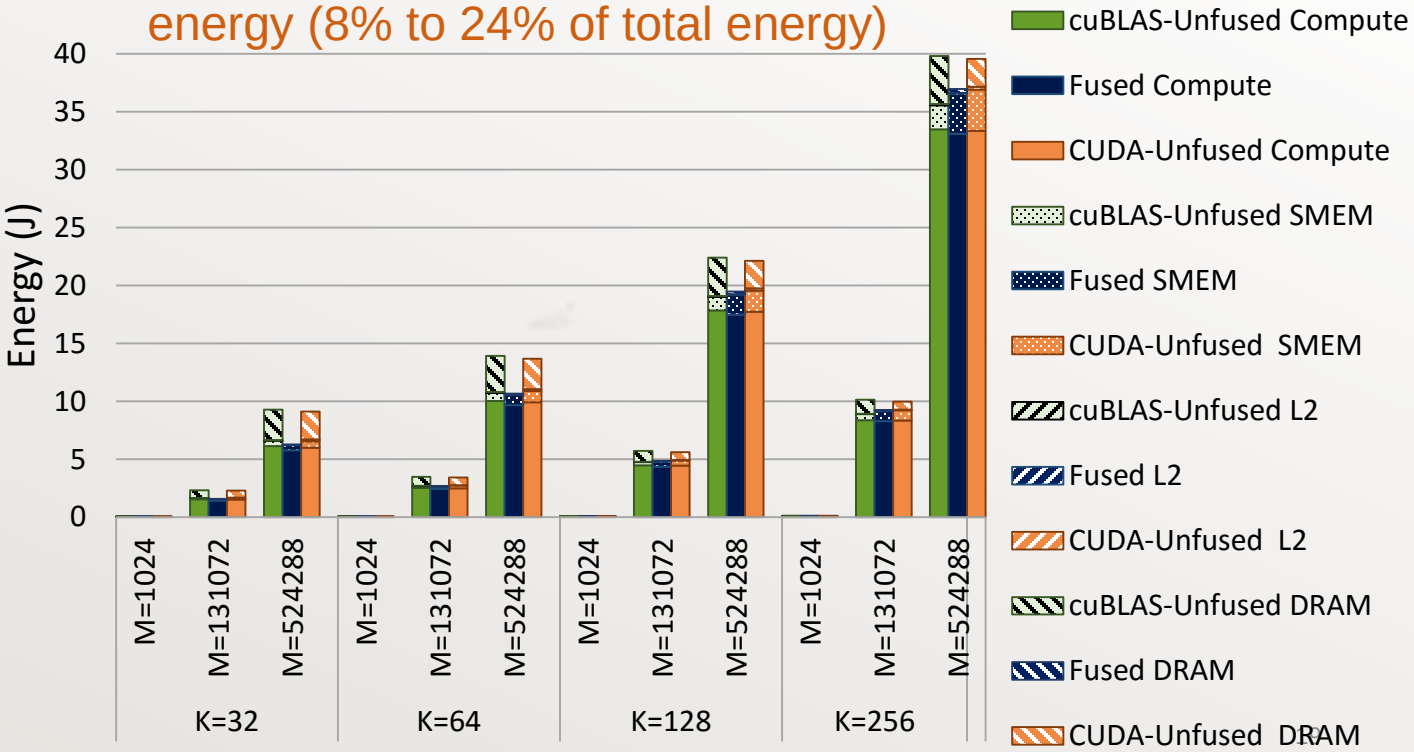
Energy Savings Comparison

- Energy Savings of Fused compared to cuBLAS-Unfused
  - With same K, saving more energy when M increases
  - The amount of energy savings obtained from fusion is greatly affected by the K value

	M=1024	M=131072	M=524288
K=32	31.3%	32.5%	32.5%
K=64	18.7%	23.6%	23.4%
K=128	10.2%	14.8%	13.1%
K=256	3.5%	8.5%	7.2%

Energy Breakdown

- 80% reduction in DRAM access energy (8% to 24% of total energy)



Summary

- Presented a fused approach of implementing the kernel summation on the state of the art GPU.
  - Fusion leads to improvement in locality and reduction of memory accesses.
  - Fusion is seen to improve overall performance of kernel summation up to 1.8X.
  - From the energy perspective, fused kernel summation shows up to 33% of total energy saving across various experimented dimensions.

Thanks!

Lab of Computer Architecture

http://users.ece.utexas.edu/~ljohn/publications.html