Algorithms for Asymmetric GEMM-like Operations

Vivek Bharadwaj
Advised by: James Demmel, Aydın Buluç

ADEPT End-of-project Presentation
Motivation

• Dense matrix-matrix multiplication (GEMM) is an intensely studied kernel. We have:
  – Shared Memory GEMM algorithms minimizing memory-to-processor communication through cache-aware tiling
  – Distributed Memory GEMM algorithms minimizing processor-to-processor communication (Cannon, SUMMA)
  – Accelerators optimized for GEMM computations (GPU, Google TPU, GEMMINI systolic array...)

• For the past three semesters, we’ve looked at GEMM-like operations where the input arguments have **unequal access costs**, both in distributed and shared memory. We summarize the work here.
Communication Avoiding SDDMM, FusedMM Operators

In review at IPDPS 2022

- Sparse-Dense matrix multiplication (SpMM) and Sampled Dense-Dense Matrix multiplication (SDDMM) are key computational kernels in graph learning, scientific computing operations

- For graphs, SDDMM is key kernel for message generation, SpMM is key for message aggregation

- Extensive existing work on shared memory optimization for both kernels and distributed-memory SpMM operations

\[
\text{SpMMA}(S, B) := S \cdot B \\
\text{SDDMM}(A, B, S) := S \ast (A \cdot B^T)
\]
We provide the first distributed-memory, communication-avoiding implementation for SDDMM using the same data distributions as SpMM Cannon-style algorithms.

Two distinct methods for combining SDDMM / SpMM primitives:
- Reuse replication of input dense matrix
- Overlap SDDMM / SpMM phases

When additional memory is available and input dense matrix does not change, even further communication savings are possible.

In review at IPDPS 2022
Communication Avoiding SDDMM, FusedMM Operators

In review at IPDPS 2022

- Experiments ran on 256 Knights Landing Cores on Cori, a Cray XC40 at LBNL

- Combining the SDDMM and SpMM primitives (FusedMM) yielded significant communication savings. Up to 1.6x faster runtime for the pair of kernels on 256 nodes with kernel overlap.

- Embedded our algorithms in applications. Tested collaborative filtering problem (Netflix Challenge) and Graph Attention Network Training on symmetric sparse matrices with tens of millions of rows, ~200-300 million edges
Now consider computations of the form $C = A \cdot B$, where $A, B$ are $m \times k, k \times n$ dense matrices and $A$ is either:

- Random i.i.d. (e.g. sketching operations in randomized linear algebra)
- Lower precision than $B$ or $C$ (mixed-precision neural network training)

If $A$ is i.i.d. random, suppose that it costs less to regenerate entries of $A$ in registers than to load it from memory (assume a single level of caching for now)

Could generate the random matrix and dispatch a GEMM cal for either problem. Can we take advantage of the unequal access cost of the two inputs?

- In theory: yes!
- In practice: working on it...
Weakened Memory Lower Bounds

Work in Progress

• Let $M$ be the number of data words in the cache that either $B$ or $C$ can hold. Standard GEMM has a data movement lower bound:

\[ \Omega \left( \frac{mnk}{\sqrt{M}} \right) + \Omega(\text{lower order terms...}) \]

• If loading words of $A$ costs $0 < \rho < 1$ time compared to loading words of $B$ or $C$, the lower bound weakens to:

\[ \Omega \left( \frac{mnk\sqrt{\rho}}{\sqrt{M}} \right) + \Omega(\text{lower order terms...}) \]
Tiling to Meet Lower Bounds

Work in Progress

• Specially tuned cache tiling shape for GEMM allows us to meet the lower bound in theory

• **In practice:** Hardware-accelerated RNG required to get performance fast enough to take advantage of tiling, need hardware support for mixed-precision matrix multiplication

• We are continuing to develop theory and experiments for unequal access cost GEMM
Thanks!