

Metamorph - Runtime System Optimizations for GPU Architectures

Motivation

Moore's Law is **DEAD!!!!**

Can't rely anymore on new CPU generations for Performance Improvement.

Leverage accelerators such as GPUs

Attention to runtime behaviors and optimizations (e.g., memory hierarchy, concurrency, and data movement).



Bhargavi
kurukunda

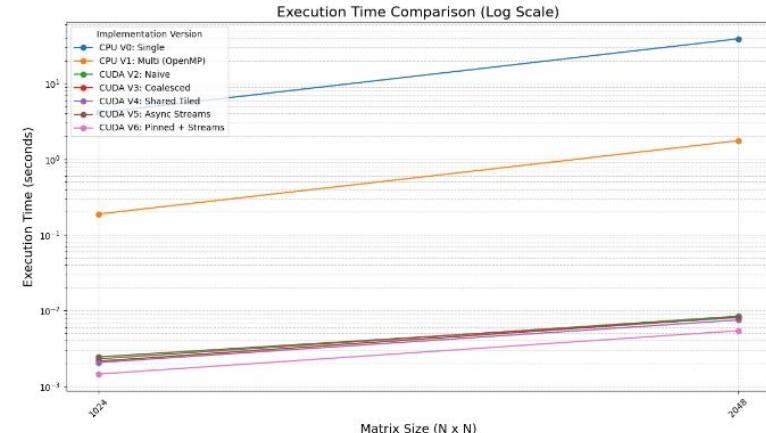
Vikas Kalagi

Project Overview

MetaMorph evolves Matrix Multiplication across several runtime strategies:

CPU baseline → multithreaded CPU → naive GPU → coalesced GPU → shared-memory → async streams → pinned-memory overlap.

We measure performance, transfer costs, and execution behavior, showing how the algorithm "morphs" as it adapts to deeper hardware-aware optimizations.



Metamorph - Runtime System Optimizations for GPU Architectures

Challenges

- Bandwidth stalls due to Non-Coalesced memory accesses.
- Shared memory required careful synchronization.
- Asynchronous streams introduced overhead for smaller matrices.
- PCI-e Data transfer latency became a major bottleneck.
- Balancing thread-block geometry, occupancy, and memory hierarchy demanded iterative profiling and tuning.

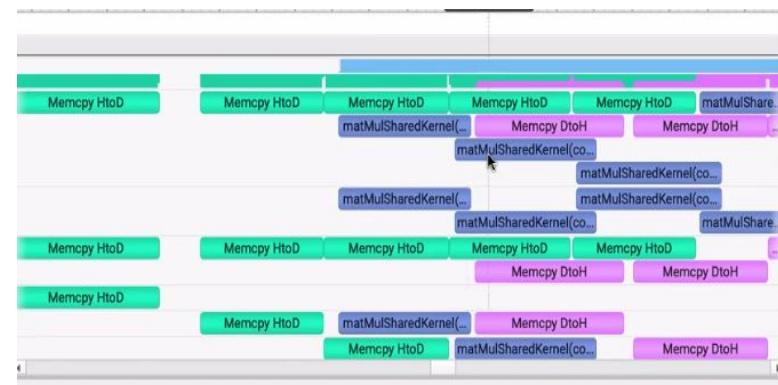


Table 1: Execution Time (ms) for Matrix Multiplication Versions ($C = A \cdot B$)

Version	Optimization Type	Time (N=1024)	Time (N=2048)
CPU V0: Single	Sequential Baseline	4097.54 ms	38921.20 ms
CPU V1: Multi	Thread Parallelism (OpenMP)	187.937 ms	1753.23 ms
CUDA V2: Naive	GPU Parallelism (Initial)	2.297 ms	8.425 ms
CUDA V3: Coalesced	Launch Geometry	2.138 ms	8.146 ms
CUDA V4: Shared Tiled	Memory Hierarchy (Kernel Opt)	2.055 ms	7.496 ms
CUDA V5: Async Streams	Concurrency (Overlap)	2.442 ms	8.095 ms
CUDA V6: Pinned + Async	Transfer Bandwidth (I/O Opt)	1.446 ms	5.415 ms