Table of Contents

[Individual paper summaries 2](#_Toc348958527)

[2010, A GPGPU Compiler for Memory Optimization and Parallelism Management, Y. Yang et al 3](#_Toc348958528)

# Individual paper summaries

2010, A GPGPU Compiler for Memory Optimization and Parallelism Management, Y. Yang et al

Their optimization process includes vectorization and memory coalescing for memory bandwidth enhancement, tiling and unrolling for data reuse and parallelism management, and thread block remapping or ad-dress-offset insertion for partition-camping elimination.



Optimized code achieves very high performance, either superior or very close to the highly fine-tuned library, NVIDIA CUBLAS 2.2, and up to 128 times speedups over the naive versions.

Another distinguishing feature of their compiler is the understandability of the optimized code, which is useful for performance analysis and algorithm refinement. As a result, it is relatively easy to reason about the optimized code generated ; which facilitates algorithm-level exploration

Their compiler works as follows. The input is a naïve GPU kernel function, which is functionally correct but does not include any device-specific performance optimizations. The compiler analyzes the naïve kernel, checks the off-chip memory access patterns, and optimizes the memory accesses through vectorization and coalescing. Then the compiler analyzes data dependencies and identifies possible data sharing across threads and thread blocks. Based on data sharing patterns, the compiler intelligently merges threads and/or thread-blocks. These merges provide a novel way to achieve loop tiling and unrolling. Additionally, the compiler schedules the code to enable data prefetching. Thread blocks are checked for their memory accesses and depending on the thread block dimensions, either an address offset is inserted or the block identifiers (ids) are remapped, if necessary. The compiler also performs hardware-specific tuning based on hardware parameters such as the register file size, the shared memory size, and the number of cores in the target GPU.