Stream Scheduling: A Framework to Manage Bulk Operations in a Memory Hierarchy

Abhishek Das
Computer Systems Laboratory
Stanford University
abhishek@cva.stanford.edu

William J. Dally
Computer Systems Laboratory
Stanford University
billd@cva.stanford.edu

Abstract

Recently, Streaming architectures such as Imagine, Merrimac and Cell were demonstrated to achieve significantly higher performance and efficiency over traditional architectures by introducing an explicitly managed on-chip storage in the memory hierarchy. This software managed memory serves as a staging area for bulk amounts of data, making all functional unit references short and predictable, while data is asynchronously transferred from external memory. The decoupling of computation from memory accesses allows the software to statically optimize the execution pipeline, transferring the onus of latency tolerance from hardware to software. The Stream programming model captures this by making computation and communication explicit in a 2-level storage hierarchy. This paradigm of structuring algorithms and explicitly managing data so that they are serviced by levels of the memory hierarchy as close to the processors as possible, however, applies to modern systems of all scales. The levels of memory hierarchy could include on-die storage such as caches, local DRAM, or even remote memory accessed over high-speed interconnect. Sequoia, a recently proposed programming language, extends the stream programming model to describe array blocking and communication for machines that can be abstracted as a tree of distinct memory modules.

It has become essential to optimize performance in such machines by managing bulk operations, such that the latency of data transfers are hidden. Bulk operations are coarse-grained in nature: either bulk data transfers or kernels (bulk computations) on bulk amounts of data. We present Stream Scheduling, a static optimization framework to efficiently schedule bulk computations and data transfers, and allocate storage at multiple levels of a memory hierarchy. The Stream Scheduling framework was introduced for performance optimization on Imagine and Merrimac, using one dimensional streams in a 2-level storage hierarchy. In this work, we show that Stream Scheduling can be extended easily to schedule bulk operations on any machine with multiple levels of memory hierarchy, adding optimizations to enhance parallelism and manage more generic multi-dimensional arrays.

Adopting Sequoia’s abstraction of a machine as a tree of memories, the intermediate representation expresses a program as a hierarchy of bulk operations with explicit parallelism. Using this hierarchy of bulk operations and estimates of their execution times, Stream Scheduling maximizes performance by overlapping bulk data transfers with useful computation. Software pipelining, applied at the level of coarse-grained bulk operations, is critical to maximize this overlapping. We introduce optimizations to enhance this by exploiting slack in bulk operations, as well as managing copies of bulk data using modulo variable expansion. Efficient allocation of scarce on-chip memory minimizes use of expensive external bandwidth, and is essential for high performance. Stream Scheduling optimizes the use of available memory at each level of the hierarchy by varying a collection of parameters (tunables) in the Sequoia program, used with blocking primitives for task decomposition. A steepest gradient search algorithm is adopted to search for the best performing working set that fits in each memory level.

We evaluate each optimization by plugging Stream Scheduling into the Sequoia compiler and running on a suite of compute-intensive and bandwidth-hungry applications. We demonstrate the scalability of our framework by targeting machines with different memory hierarchies: a Cell blade, a distributed-memory cluster, and the Cell blade attached to a disk. Overall, our efforts realize 90% and more utilization of the critical resources, while matching the best known (hand-tuned) performance.