A Review of: Generalized Cache Tiling for Dataflow Programs

Zachary Sisco

April 12, 2017

Overview

"Generalized Cache Tiling for Dataflow Programs" by Łukasz Domagała, Duco van Amstel, and Fabrice Rastello. LCTES 2016.

- What is dataflow programming? Why?
- Extending tiling to dataflow programs
- Cache-tiling optimization solutions
- Experimental Results & Analysis
- Conclusions & Future Work

Dataflow Programming

Controlflow Program

- Stream of instructions operate on external data.
- Conditional execution change the instruction-execution path.
- Data is “static” unless instruction stream moves it.

Dataflow Program

- Stream of data passing from instruction to instruction.
- Conditional execution routes data to different instructions.
- Data flowing through “static” instructions.

Dataflow Programming

Applications

- Data streaming and processing (e.g., video processing)
- Cryptographic encoding and decoding
- Embedded systems

Loop Tiling

Group together operations within an iteration space so they execute atomically.

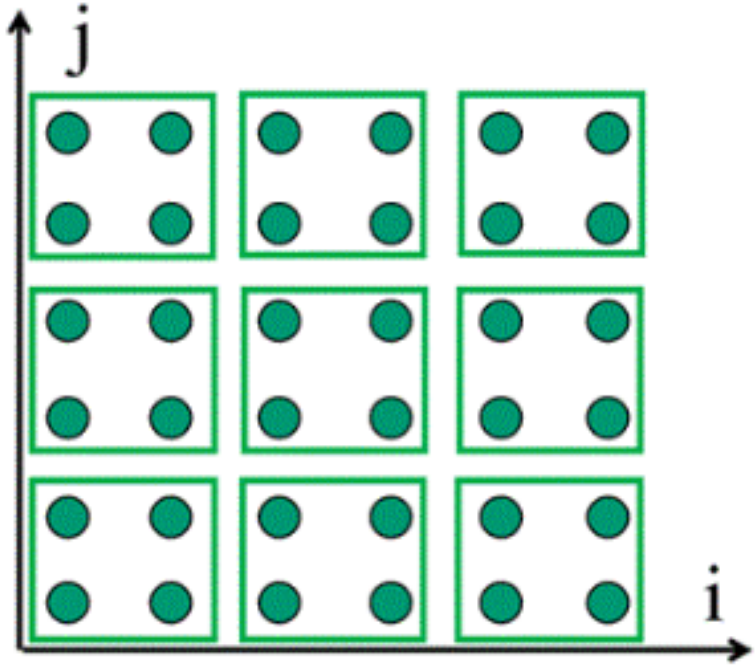

Figure 1: Reduce cache misses with loop-nest tiling. Figure from (Liu, 2017)

Dataflow Iteration Space

- Actors — basic programmable units
- Channels — connect actors, pass data elements

Dataflow Iteration Space

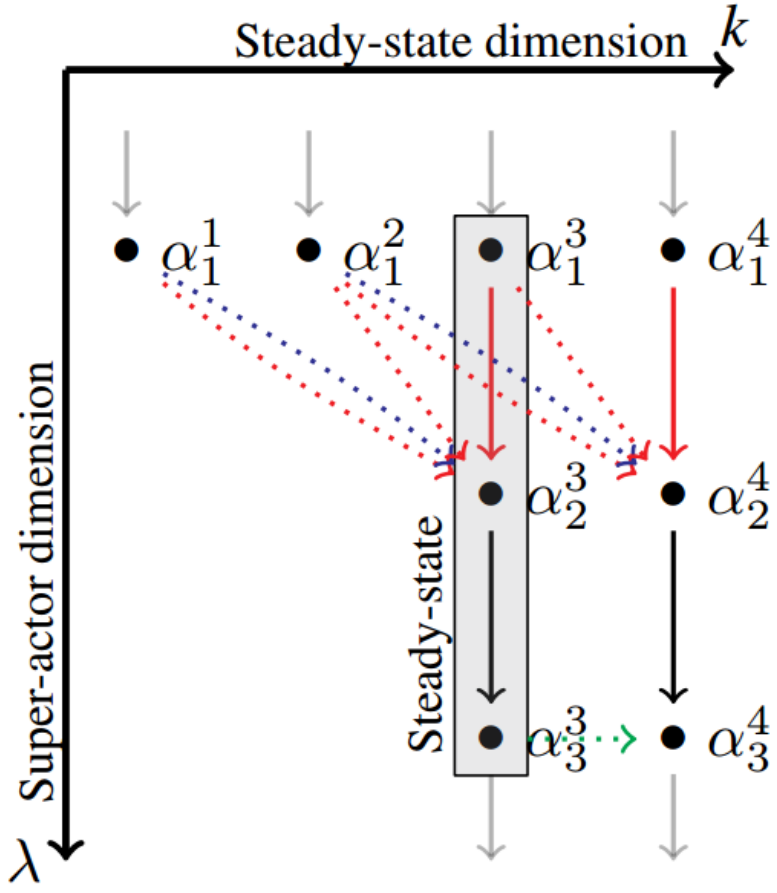

Figure 2: Iteration space with dependencies. From (Domagała et al., 2016)

Dataflow Iteration Space

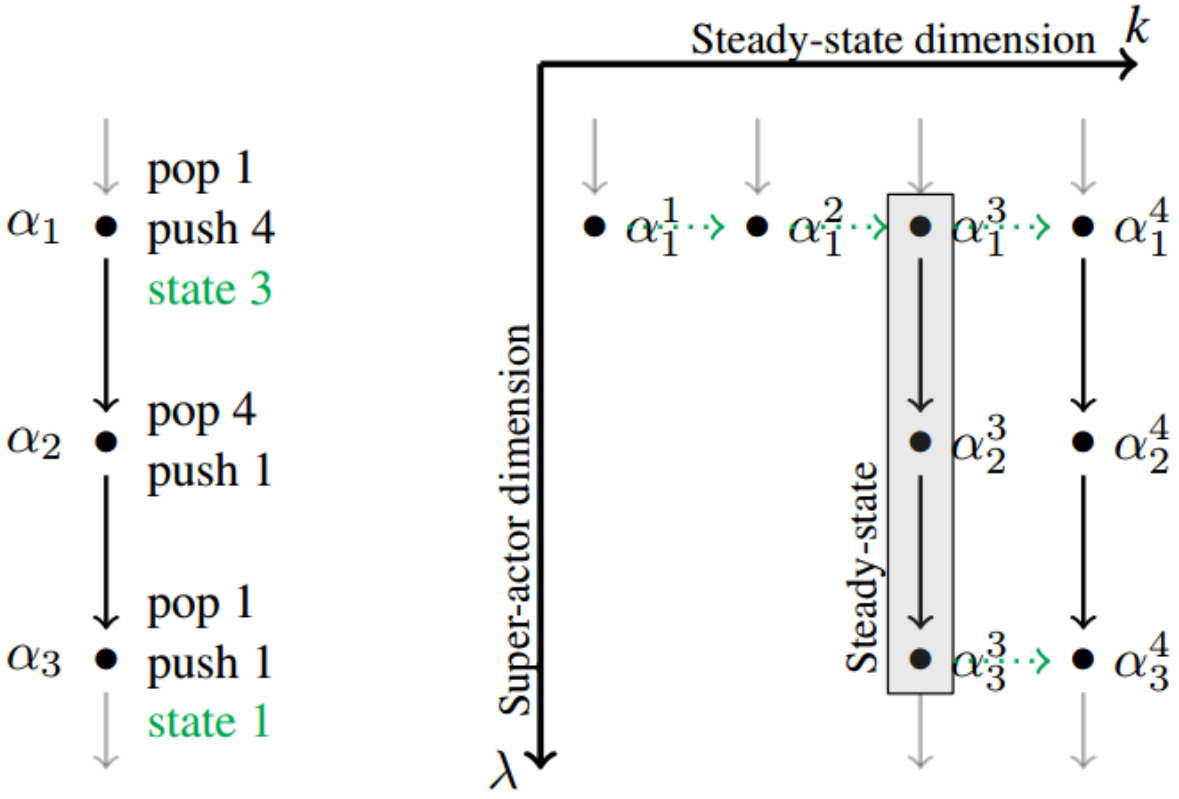

Figure 3: Iteration space after transformation. From (Domagała et al., 2016)

Optimization Problem

Minimize cache misses

- Find the largest possible width for each tile that satisfies its cache-size requirement and minimizes a cost function.
- Cost function is defined with respect to minimizing the variable *spill*.

Constraint Programming

Constraints

- Ensure the ordering of nodes respects the original dataflow dependencies
- Ensuring the unique indexing and ordering of nodes
- Ensuring the cache size is not exceeded
- And more...

Heuristics-based Solutions

Near-optimal solution in linear-time complexity

- Greedy tiling
- Schedule actors based on descending order of edge weights (size of data transfer)

Experimental Evaluation

- StreamIt dataflow language
- Benchmark of 12 dataflow programs
- Simulated ARM architecture with 16kB L1 instruction cache and 8kB L1 data cache
- Least-Recently Used cache replacement policy

Experimental Evaluation

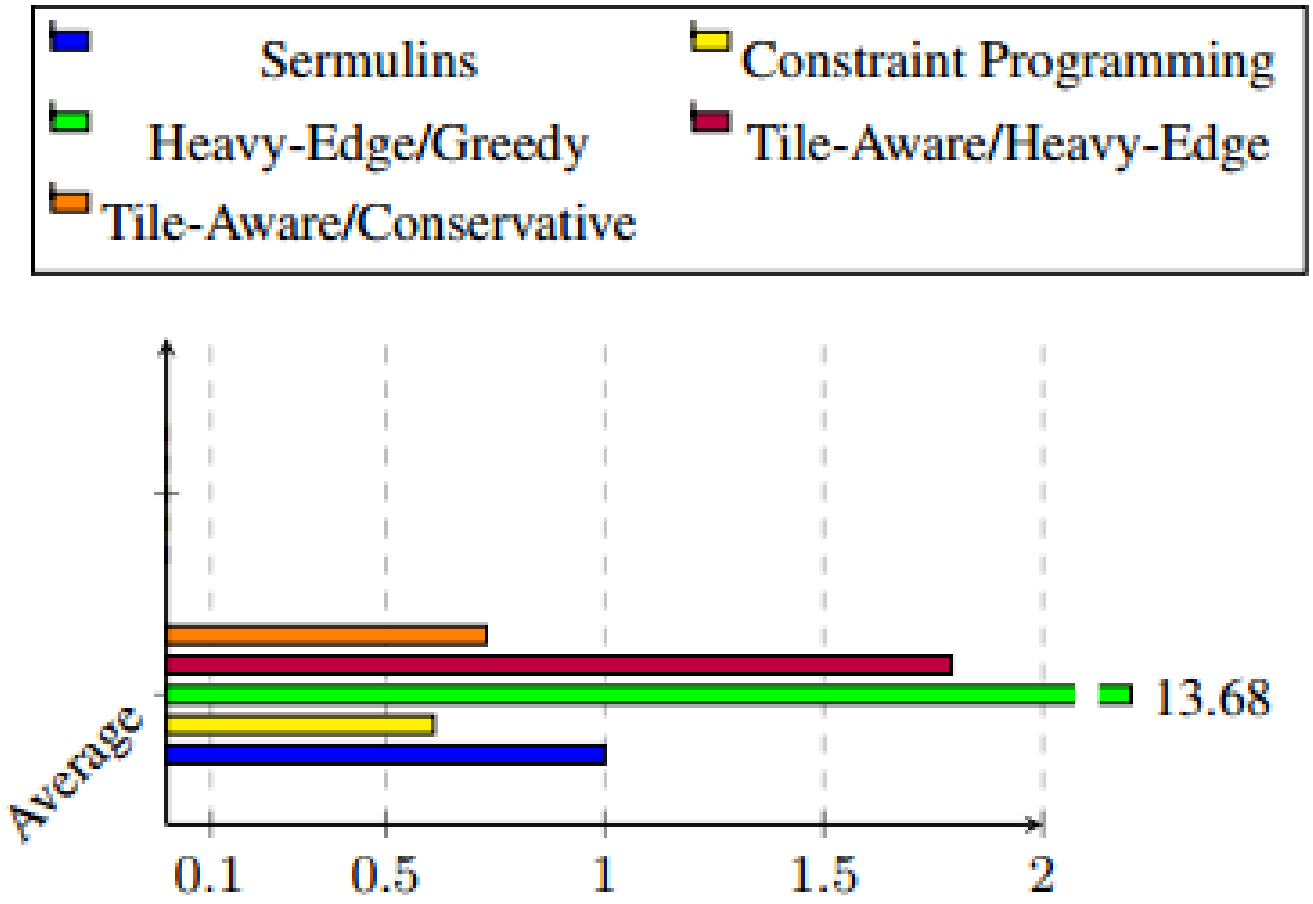

Figure 4: Average cache-miss amounts. Adapted from (Domagała et al., 2016)

Conclusions & Future Work

- Reduce cache misses in dataflow programs through cache-tiling optimization
- Improves over existing optimizations
- Extension to parallel or distributed architectures
- Improve model with information about cache layout/associativity

Thank You

Questions?