A general purpose automatic overlapped tiling technique in polyhedral frameworks

Jie Zhao
INRIA & Ecole Normale Supérieure, 45 rue d’Ulm, 75005 Paris, France
jie.zhao@inria.fr

Abstract

Loop tiling [1] is an essential transformation to exploit data locality and parallelism, being implemented as a general technique in numerous existing polyhedral frameworks including Plato [2], PPG [7], etc., and domain specific automatic code generators like PolyMage [7]. Typically, automatic polyhedral frameworks either choose to implement the parallelogram tiling avoiding the difficulty in performing code generation but leading to pipelined startup, or resort to sophisticated tiling shapes, e.g., diamond tiling [1], enabling concurrent startup but complicating both scheduling and code generation.

Overlapped tiling [6] is a technique designed to eliminate pipelined startup by modifying tile shapes obtained from existing frameworks, but no implementations for general purpose multicores have been reported in a polyhedral framework except those domain specific code generators [7], preventing its application in general purpose languages and missing a comparison with other state-of-the-art techniques. We design and implement an automatic overlapped tiling technique for both general purpose and heterogeneous multicores in a polyhedral framework, neither being restricted to domain specific languages nor introducing sophisticated rescheduling in polyhedral frameworks, but neither being restricted to domain specific languages nor introducing sophisticated rescheduling in polyhedral frameworks. Overlapped tiling [6] is a technique designed to eliminate pipelined startup, or resort to sophisticated tiling shapes, e.g., diamond tiling [1], enabling concurrent startup but complicating both scheduling and code generation.

Our work is comparable with the state-of-the-art diamond tiling on stencil PolyMage benchmarks but written in general purpose languages, validating both stencil computations and image processing pipelines extracted from the languages nor introducing sophisticated rescheduling in polyhedral frameworks, but no implementations for general purpose multicores have been reported.

Introduction

• Being easy to implement, simple tiling shapes, e.g., parallelogram tiling, mass parallel startup.
• Enabling concurrent startup, complex tiling shapes complicate scheduling and code generation.

![Figure 1: Tiling shapes](image)

Table 3: Comparison of different tiling techniques

| Code generator: PPG (version ppgr-0.07-26-g236d559) |
| Architecture: 32-core Intel Xeon(R) E5-2683 v4 @ 2.10GHz |
| Compilation: ICC 11.0 (O3 - host -openmp -ipo) |
| Baseline: sequential PolyMage naive code (without tiling) [5] |
| Comparison: Halide manual and automatic scheduling [6], PolyMage naive and optimized scheduling [5] |
| Parallelogram tiling [1] and diamond tiling [1] |

Other Transformations

- Alignment and scaling of stages.
- Grouping/Pinning.
- Storage mapping.
- Hybrid tiling.

Experimental Results

- Code generator: PPG (version ppgr-0.07-26-g236d559).
- Architecture: 32-core Intel Xeon(R) E5-2683 v4 @ 2.10GHz.
- Compilation: ICC 11.0 (O3 - host -openmp -ipo).
- Baseline: sequential PolyMage naive code (without tiling) [5].
- Comparison: Halide manual and automatic scheduling [6], PolyMage naive and optimized scheduling [5].

Overlapped Tiling

For (i = 1; i <= N; i++)

Figure 2: (a) Iteration domain; (b) overlapped tiling with shifting; (c) overlapped tiling without shifting.

Figure 3: Implementing overlapped with expansion nodes (a) with shifting, and in (b) schedule tree; (c) without shifting, and in (d) schedule tree.

Other Transformations

- Alignment and scaling of stages.
- Grouping/Pinning.
- Storage mapping.
- Hybrid tiling.

Figure 4: Performance Comparison of (a) Bilateral Grid; (b) Camera Pipelines; (c) Multiscale Interpolation; (d) Local Laplacian Filter.

Note: Halide automatic scheduling for Multiscale Interpolation is not publicly available.

Conclusions

The polyhedral framework of compilation unifies a wide variety of loop and array transformations using affine (linear) transformations. The availability of a general purpose method to generate imperative code after the application of such affine transformations brought polyhedral compilers to the front scene. As a concurrent-startup-enabled tiling shape, overlapped tiling [1] is implemented in the domain specific code generator PolyMage [5]. In this work, we designed and implemented a general-purpose automatic overlapped tiling technique in a polyhedral framework by showing two implementations with/without shifting/dewing, followed by a series of experimental results on both image processing pipelines and stencil computations, validating the effectiveness of our method.

Forthcoming Research

We will be looking at all the remaining PolyMage benchmarks. Also, we have finished the CUDA code generation of these benchmarks. An in-depth comparison of all the PolyMage benchmarks by conducting experiments on both CPU and GPU architectures is coming soon.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant No. 61702546.

References