SALTAR Project Overview

Russell Bentley

Stony Brook

2024

Table of Contents

Stony Brook University

Deliverables

- SALTAR
  - DSL
  - Compiler
  - Autotuner
- New approximation algorithms
- Benchmark Suite

Prior Work

- Pachoir compiler [17]
- PLUTO [7], [6]
- Fourst compiler (FFT based) [4]
- Halide (C++ DSL) [16]
- Devito (FD focus, useful resource) [14]

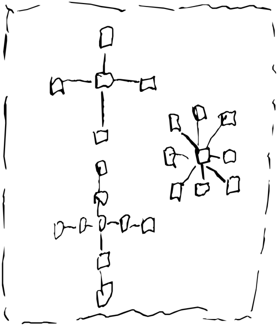
.

Stony Brook University

Table of Contents

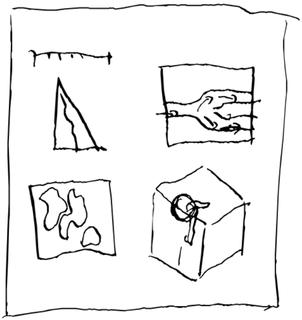
Stencil System

- 1 or more stencil operation
- Each stencil is fully programmable
  - Like a shader, SPMD
  - Limited write access
- Can vary
  - Spatially
  - Temporally
  - By domain composition



Domain

- Node graph / connectivity
- May be geometry driven
- Should be programmable (?)
  - Condensation problems
- Lots of axis aligned boxes in practice
- Might include “coloring” to map different stencils
- Where do want to sample?
- Durating of runtime
  - How often do we sample?



Boundary Conditions

- Fully programmable
  - Highly dependent on domain and stencil system
- May require sampling
  - Outflow boundary conditions
  - Coupling to other simulations

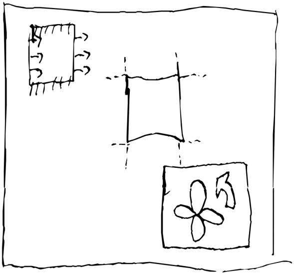


Table of Contents

FFT

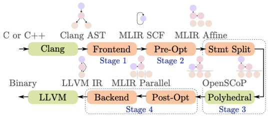
- FFT algorithms [3] [1] [1]

Guassian Approximation

- Guassian Approximation [2]

Polyhedral Compilers

- PLUTO [7], [6]
  - C source to source compiler
  - Access optimized with tiling
- LLVM Polygeist [15]
  - C++ / C interface to MLIR [13]
  - MLIR polyhedral optimization passes
- LLVM Polly [10]
  - Polyhedral optimization of LLVM IR



Polygeist workflow

Misc Techniques

- Branch Removal
  - If statements may have arithmetic based alternative
- Vectorizing
- Data layout / Access patterns (AoS ¡-¿ SoA)

Table of Contents

- Profiling?
- Plan Composition
- Algorithm Tradeoffs
- Other hyper parameters?

Table of Contents

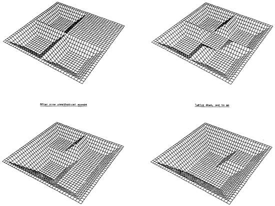
Stony Brook University

Related Projects

- FFTW [8]
  - Fast fourier transform compiler
  - A dependency(?) for SALTAR
  - Similiar architectural concerns
- Taichi [12]
  - JIT compiler parallel numerical code
  - Optimizes computation over sparse data
  - DSL is based on python.
- Eigen [11]
  - Runtime vs compile time configuration
- FEnics [5]
  - Compiler framework for FEA

Related Algorithms

- HashLife [9]
  - Memoized Algorithm for cellular automata
  - Sensitive to entropy
- Quicklife (Open Source with Golly)
  - Tree based evaluation
  - No hashing



References I

[1] Zafar Ahmad et al. "A fast algorithm for aperiodic linear stencil computation using Fast Fourier Transforms". In: *ACM Transactions on Parallel Computing* 10.4 (2023), pp. 1–34.

[2] Zafar Ahmad et al. "Brief announcement: Faster stencil computations using gaussian approximations". In: *Proceedings of the 34th ACM Symposium on Parallelism in Algorithms and Architectures*. 2022, pp. 291–293.

[3] Zafar Ahmad et al. "Fast stencil computations using fast Fourier transforms". In: *Proceedings of the 33rd ACM Symposium on Parallelism in Algorithms and Architectures*. 2021, pp. 8–21.

[4] Zafar Ahmad et al. "Fourst: A code generator for FFT-based fast stencil computations". In: *2022 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE. 2022, pp. 99–108.

Stony Brook University

References II

[5] Igor A Barrata et al. "DOLFINx: The next generation FEniCS problem solving environment". In: (2023).

[6] Uday Bondhugula et al. "A practical automatic polyhedral parallelizer and locality optimizer". In: *Proceedings of the 29th ACM SIGPLAN Conference on Programming Language Design and Implementation*. 2008, pp. 101–113.

[7] Uday Bondhugula et al. "Automatic transformations for communication-minimized parallelization and locality optimization in the polyhedral model". In: *Compiler Construction: 17th International Conference, CC 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings 17*. Springer. 2008, pp. 132–146.

References III

[8] Matteo Frigo and Steven G Johnson. "The design and implementation of FFTW3". In: *Proceedings of the IEEE* 93.2 (2005), pp. 216–231.

[9] R Wm Gosper. "Exploiting regularities in large cellular spaces". In: *Physica D: Nonlinear Phenomena* 10.1-2 (1984), pp. 75–80.

[10] Tobias Grosser, Armin Groesslinger, and Christian Lengauer. "Polly—performing polyhedral optimizations on a low-level intermediate representation". In: *Parallel Processing Letters* 22.04 (2012), p. 1250010.

[11] Gaël Guennebaud, Benoît Jacob, et al. *Eigen v3*. http://eigen.tuxfamily.org. 2010.

References IV

[12] Yuanming Hu et al. "Taichi: a language for high-performance computation on spatially sparse data structures". In: *ACM Transactions on Graphics (TOG)* 38.6 (2019), p. 201.

[13] Chris Lattner et al. "MLIR: Scaling compiler infrastructure for domain specific computation". In: *2021 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*. IEEE. 2021, pp. 2–14.

[14] Fabio Luporini et al. "Architecture and performance of Devito, a system for automated stencil computation". In: *ACM Transactions on Mathematical Software (TOMS)* 46.1 (2020), pp. 1–28.

[15] William S Moses et al. "Polygeist: Raising C to polyhedral MLIR". In: *2021 30th International Conference on Parallel Architectures and Compilation Techniques (PACT)*. IEEE. 2021, pp. 45–59.

References V

[16] Jonathan Ragan-Kelley et al. "Halide: a language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines". In: *Acm Sigplan Notices* 48.6 (2013), pp. 519–530.

[17] Yuan Tang et al. "The Pochoir Stencil Compiler". In: *Proceedings of the twenty-third annual ACM symposium on Parallelism in algorithms and architectures*. 2011, pp. 117–128.

Stony Brook University