Compile to Learn to Compile: There and Back Again

ACACES 2021 — albertcohen@google.com

This reading list was compiled in the context of a course at the ACACES 2021 summer school, organized by HiPEAC and TETRAMAX in Fiuggy, Italy.

Course Abstract

The lurking optimizing compiler dragons are not going away any time soon. Intrepid computer scientists and engineers lured by glittering performance continue to struggle with complex computer architectures. The orchestration of computations, communication and storage on distributed and heterogeneous systems is not getting any simpler. Still, systems have been designed to tame the beast, and some of them made tremendous progress towards hiding all this complexity to end users while delivering excellent performance levels. This is particularly true in the field of Machine Learning (ML), and tensor compilers in particular. Conversely, ML-based compiler techniques have become more widespread, addressing some of the most difficult optimization problems faced by tensor compilers.

We will review some of the interactions between machine learning and compilation in this never-ending quest for performance. Beyond performance, we will also raise research and engineering challenges in compiler construction, semantics and algorithms crossing abstractions and languages.

The course will run over 5 lectures, covering the following topics:

1. Bridging the abstraction gap in ML frameworks: from automatic differentiation to code generation for tensor algebra.
2. Parallelizing compilers, there and back again: from the parallelization of Fortran dusty decks to modern tensor compilers.
3. Good abstractions are like ghosts, which everybody talks about and few have seen: from equational reasoning to polyhedra and structured operations.
4. Heading into the dragon's lair, a.k.a. the construction of ML compilers: comparing XLA, TVM, Tensor Comprehensions and MLIR.

5. The Eagles are coming! To the rescue of the defenseless compiler engineer: ML-based compilation heuristics and autotuning.

Course Material

JAX

- https://jax.readthedocs.io/en/latest/index.html
- https://github.com/google/jax
- https://bit.ly/jax-tpu

Automatic Differentiation

- https://mblondel.org/teaching/autodiff-2020.pdf
- https://sscardapane.github.io/learn-autodiff

XLA

- https://www.tensorflow.org/xla

DNN Fusion

- PLDI 2021

TVM

- https://tvm.apache.org
- OSDI 2018

Tensor Comprehensions

- https://research.fb.com/downloads/tensor-comprehensions
- https://dl.acm.org/doi/10.1145/3355606

FLAME/BLIS

- https://github.com/flame/blis

MLIR

- https://ieeexplore.ieee.org/abstract/document/9370308

Further Reading

Graph optimization

- https://web.stanford.edu/class/cs245/slides/TFGraphOptimizationsStanford.pdf
- https://github.com/jiazhihao/TASO

TensorFlow Lite (embedded/mobile inference)

- https://www.tensorflow.org/lite

XLA

- Phothilimthana et al. A Flexible Approach to Autotuning Multi-Pass Machine Learning Compilers, PACT 2021 (to appear)

Telamon

- CC 2017: Beaugnon et al., Optimization space pruning without regrets
- arXiv preprint: Beaugnon et al., On the Representation of Partially Specified Implementations and its Application to the Optimization of Linear Algebra Kernels

MLIR

- https://llvm.org/devmtg/2020-09/slides/MLIR_Tutorial.pdf
- https://mlir.llvm.org/talks

Polyhedral Compilation

- isl library: https://repo.or.cz/w/isl.git
- Bastoul “Code generation in the polyhedral model is easier than you think”, PACT 2004
- Grosser, Verdoolaege, Cohen “Polyhedral AST generation is more than scanning polyhedra”, TOPLAS 2015
- Feautrier “Some efficient solutions to the affine scheduling problem” (part I and II), Intl. Journal of Parallel Programming 21(5) and 21(6).

- Bondhugula, Hartono, Ramanujam, Sadayappan “A practical automatic polyhedral parallelizer and locality optimizer”, PLDI 2008
- Vasilache, Meister, Baskaran, Lethin “Joint scheduling and layout optimization to enable multi-level vectorization”, IMPACT 2012
- Zinenko, Verdoolaege, Reddy, Shirako, Grosser, Sarkar, Cohen “Modeling the conflicting demands of parallelism and Temporal/Spatial locality in affine scheduling”, CC 2018

ML for Compilers

- Wang and O’Boyle’s survey (U. Lancaster and U. Edinburgh): https://ieeexplore.ieee.org/document/8357388
- Cummins and Leather’s survey (Facebook Research): https://research.fb.com/publications/machine-learning-in-compilers-past-present-and-future

ML for Systems

- Maas’s Taxonomy: https://dl.acm.org/doi/abs/10.1109/MM.2020.3012883
- Fursin’s CK: http://cknowledge.org (supporting MLPerf, ACM Artifact Evaluation)
- https://www.dividiti.com