Outlook on generating optimal HPC code with ML

Emil VATAI, `https://vatai.github.io/talks/2022jlesc.pdf`

September 28, 2022

Outline

Intro

Code Generation

About us

- Emil Vatai, Riken R-CCS, `https://vatai.github.io/talks/2022jlesc.pdf`
- RIKEN R-CCS, High Performance Artificial Intelligence Team
- Team leader: Mohamed WAHIB
- WE ARE HIRING!
  `https://www.riken.jp/en/careers/researchers/20220511_1/index.html`

Other works

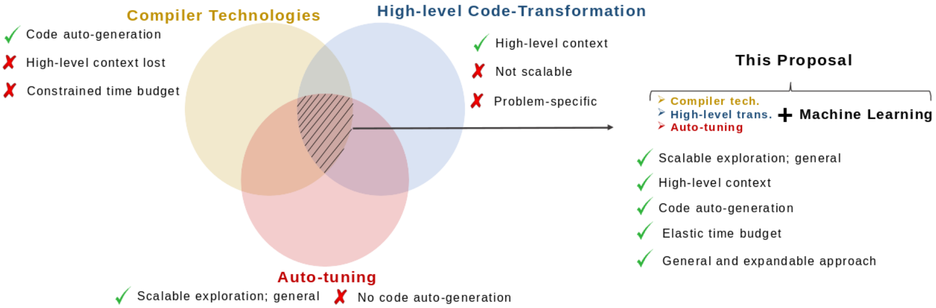
- NLP: Alex Drozd
- Spiking neural networks, Brain simulations: Jun Igarashi
- FugakuNext colaboration Jens Domke from Supercomputing Performance Research Team (SuPeR) (they are also HIRING) and others
- Sparsity in numerical methods and ML (my little project)

Terminology: codegen

1. ML people:
   - generating code from text (e.g. git{hub,lab} commit messages)
2. Compiler people:
   - "lowering" to LLVM-IR, assembly or similar
3. HPC people:
   - "performance portability via source to source compilation"
     - E.g. polly, Pluto, PPCG

Overview

Note: High level transformation – hand tuned by experts



Compiler Technologies

- ✓ Code auto-generation
- ✗ High-level context lost
- ✗ Constrained time budget

High-level Code-Transformation

- ✓ High-level context
- ✗ Not scalable
- ✗ Problem-specific

Auto-tuning

- ✓ Scalable exploration; general
- ✗ No code auto-generation

This Proposal

- ➢ Compiler tech.
- ➢ High-level trans.
- ➢ Auto-tuning

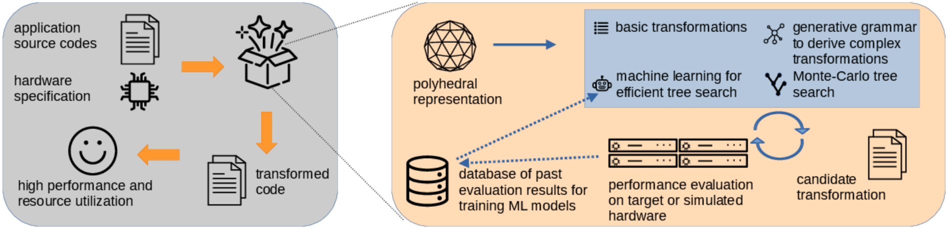
+ **Machine Learning**

- ✓ Scalable exploration; general
- ✓ High-level context
- ✓ Code auto-generation
- ✓ Elastic time budget
- ✓ General and expandable approach

The problem and the goal

- Vast number of experts need to be working on optimizing HPC codes to obtain maximal performance
- The problem: performance portability (for HPC codes)
  - Automatic but not restricted like compilers ("too generic" + playing it safe)
  - ML driven: shrinking the vast search space
  - Focused on scientific codes (e.g. stencils)
- Our goal: Use ML for high-level optimization
  - The ultimate goal: AI produces fastest code for any machine
  - Realistic goal: find a foothold in this field of code generation by learning to generate optimal code for simpler (but still important) problems
  - Key points: representation, ML methods, candidate applications
  - ML for high-level optimizations is mostly left unexplored

A framework to automate the process



application source codes

hardware specification

transformed code

high performance and resource utilization

polyhedral representation

basic transformations

generative grammar to derive complex transformations

machine learning for efficient tree search

Monte-Carlo tree search

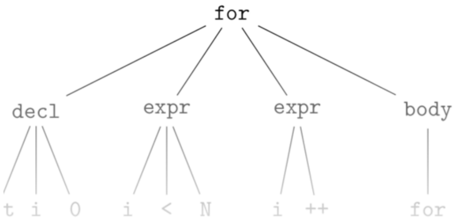
database of past evaluation results for training ML models

performance evaluation on target or simulated hardware

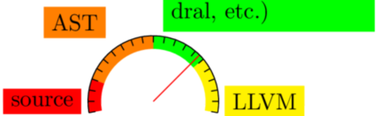
candidate transformation

The main questions

- Representation $\Longrightarrow$ ML methods



Domain: $\{S_1[i] : 0 \leq i < N;\} \cup$
$\{S_2[i,j] : 0 \leq i < N \wedge 0 \leq j < N\}$
Write: $\{S_1[i] \rightarrow v[i]; S_2[i,j] \rightarrow v[i]\}$
Read: $\{S_2[i,j] \rightarrow v[i]; S_2[i,j] \rightarrow A[i,j]\}$



```
void mv(size_t N, double *A, double *v){
 for(size_t i = 0; i < N; i++){
  v[i] = 0.0;
  for(size_t j = 0; j < N; j++){
   v[i] += A[i*N+j];
  }
 }
}
```

```
 %10 = load i64, i64* %7, align 8
 %11 = load i64, i64* %4, align 8
 %12 = icmp ult i64 %10, %11
 br i1 %12, label %13, label %42

13:                    ; preds = %9
 %14 = load double*, double** %6, align
 %15 = load i64, i64* %7, align 8
```

Current efforts

- Polyhedral: ISL, Polly, Polygeist
  - Candidate for representation: It is a symbolic representation of loops which a compact and precise description of the dependencies and the legality of transformations.
- Simple stencil benchmarks, weather codes.