Automatic Source-to-Source Optimizations for Heterogeneous Architectures using Machine Learning

Maksim Berezov

Adviser: Corinne Ancourt

CRI MINES ParisTech (France), PSL Research University

J2A ISMME

What is our goal?

Our goal is to generate a fast invariant version of a given code by applying a set of legal source-to-source transformations (e.g. tiling, unrolling, interchange). These transformations require input parameters to be applied. Our objective is to find the best transformation parametersand sequence to minimize execution time.

Original code

```
int i, j, k;
for (i = 0; i < N; i++)
l0:
    for (j = 0; j < N; j++)
        for (k = 0; k < N; k++)
            res[i][j] += mat1[i][k]*mat2[k][j];
```

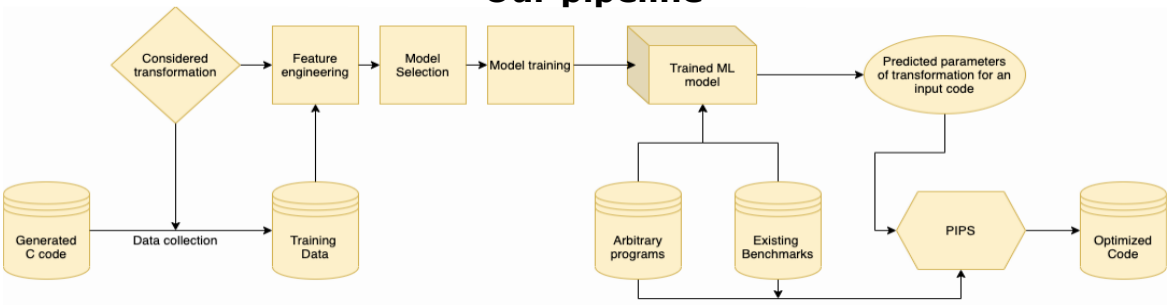
Generated code (after 2D tiling transformation)

```
int i, j, k;
 //PIPS generated variable
 int j_t, k_t, j_l, k_l;
 {
    //PIPS generated variable
    int I_0, I_1;
    I_1 = 999;
    I_0 = 0;

#pragma omp parallel for private(j_l,j_t,k_l,k_t)
      for(i = I_0; i <= I_1; i += 1)
         for(j_t = 0; j_t <= 124; j_t += 1)
            for(k_t = (((-j_t)>((-62))) ? (-j_t): ((-62))); k_t <= -j_t+62; k_t += 1)
               for(j_l = (((0)>((-16*k_t))) ? (0): ((-16*k_t))); j_l <= ((((999)<((-16*k_t+15))) ?
(999): ((-16*k_t+15)))); j_l += 1)
                  for(k_l = 16*j_t+16*k_t; k_l <= ((((16*j_t+16*k_t+15)<((999))) ? (16*j_t+16*k_t+15):
((999)))); k_l += 1)
                     res[i][j_l] += mat1[i][k_l]*mat2[k_l][j_l];
  }
```

This code is semantically equivalent to the original code but may execute faster on a parallel machine.

Our pipeline



Optimization complexity

Search space of possible transformations and their parameters is very large

m = #transformations, **s** = optimization sequence size

#number of transformation sequences = $\mathbf{m^s}$

#number of permutations = **s!**

Moreover, some transformations are parameterized. For example, 2D tiling has **432** combinations of parameters for 2x2 partitioning matrix, 6 scanning directions, 6 tile sizes per dimension and 2 shapes.

Challenges

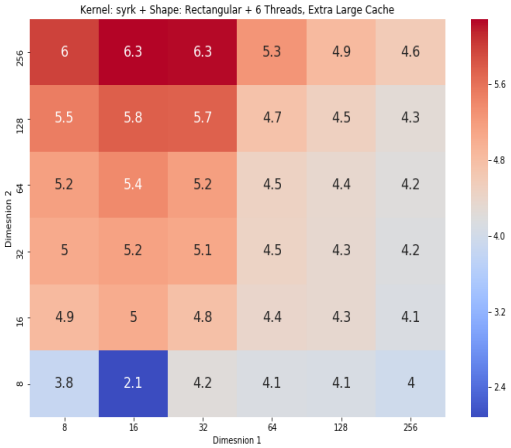
Feature space design. Feature space must introduce the code itself and must not be created artificially for a single transformation.

Code generator: Automatic code generation of C benchmarks is required in order to obtain enough training data for ML models.

Phase-ordering of transformations: Phase ordering matters a lot and this problem should be modeled from the point of view of Machine Learning algorithms.

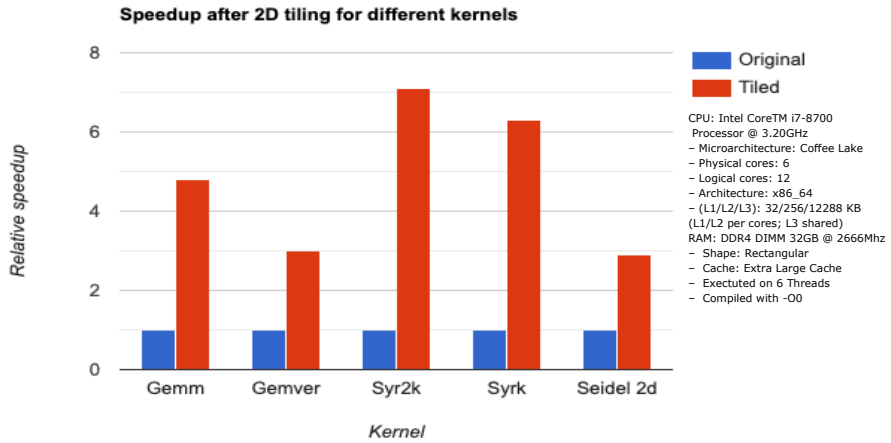
Generalization for different architectures: The type of architecture should be a hyper-parameter.

Parameter selection for 2D tiling



Syrk. Speedup after 2D tiling for different partitioning matrices

Experimental results for 2D tiling transformation



References

- Cummins, Chris, et al. "End-to-end deep learning of optimization heuristics."2017 26th International Conference on Parallel Architectures and Compilation Techniques (PACT).IEEE, 2017.
- Teixeira, SFX Thiago, et al. "Locus: a system and a language for program optimization."2019 IEEE/ACM International Symposium on Code Generation and Optimization (CGO).IEEE, 2019.
- Irigoin, François, and Rémi Triolet. "Supernode partitioning."Proceedings of the 15th ACM SIGPLAN-SIGACT symposium on Principles of programming languages. 1988.
- Chen, Tianqi, et al. "Learning to optimize tensor programs."Advances in Neural Information Processing Systems. 2018.
- Malik, Abid M. "Optimal tile size selection problem using machine learning."2012 11th International Conference on Machine Learning and Applications. Vol. 2.IEEE, 2012.