Polyhedral compilation and loop counting

Marcin Copik

July 3, 2021

Loop complexity

```
for(int i = 0; i < 10; ++i) {
  for(int j = 0; j < 100; ++j) {
    sum += 1.5 * i;
  }
}
```

How many times is the inner loop executed?

Loop complexity

```
for(int i = 0; i < 10; ++i) {
  for(int j = 0; j < 100; ++j) {
    sum += 1.5 * i;
  }
}
```

How many times is the inner loop executed?
Easy, **1000**

Loop complexity

```
for(int i = 0; i < matrix_size; ++i) {
  for(int j = i; j < matrix_size; ++j) {
    sum += 1.5 * i;
  }
}
```

How about now?

Loop complexity

```
for(int i = 0; i < matrix_size; ++i) {
  for(int j = i; j < matrix_size; ++j) {
    sum += 1.5 * i;
  }
}
```

How about now?
Slightly more complex,

$$matrix\_size + matrix\_size - 1 + \cdots + 1 = \frac{matrix\_size^2 + matrix\_size}{2}$$

Let's solve it!

First approach - LLVM's Scalar Evolution

```
for(int i = 0; i < matrix_size; ++i) {
    for(int j = i; j < matrix_size; ++j) {
        sum += 1.5 * i;
    }
}
```

Outer loop: `(0 smax %matrix_size)`
Inner loop: `{%matrix_size,+,-1}<%for.i>` i.e. $\%matrix\_size - i$.

+ fast, simple, available in LLVM
− limited to non-nested loops with constant stride

Let's solve it!

Second approach - symbolic solver

1. extract loop information from source code or IR
2. create count functions n_i for all d nested loops
3. compute the sum $\sum_{i_0=0}^{n_1} \sum_{i_1}^{n_1} \cdots \sum_{i_{d-1}}^{n_{d-1}} n_d$

How to solve this?

Let's solve it!

Second approach - symbolic solver

1. extract loop information from source code or IR
2. create count functions n_i for all d nested loops
3. compute the sum $\sum_{i_0=0}^{n_1} \sum_{i_1}^{n_1} \cdots \sum_{i_{d-1}}^{n_{d-1}} n_d$

How to solve this?

+ can process every loop as long as we can determine parameters
− slow, requires external solver, quality of results depends entirely on the solver

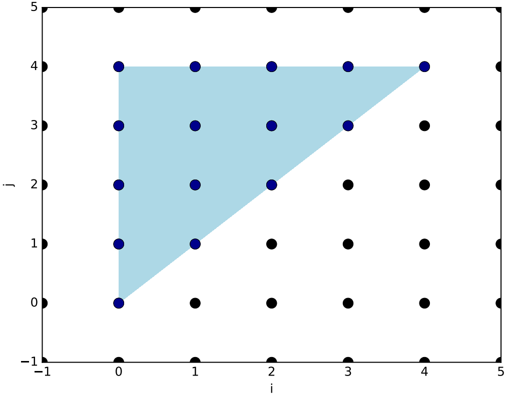
Let's solve it!

```
for(int i = 0; i < matrix_size; ++i) {
  for(int j = i; j < matrix_size; ++j) {
    sum += 1.5 * i;
  }
}
```

For the inner loop, we have two constraints:

- $0 \leq i < matrix\_size$
- $i \leq j < matrix\_size$

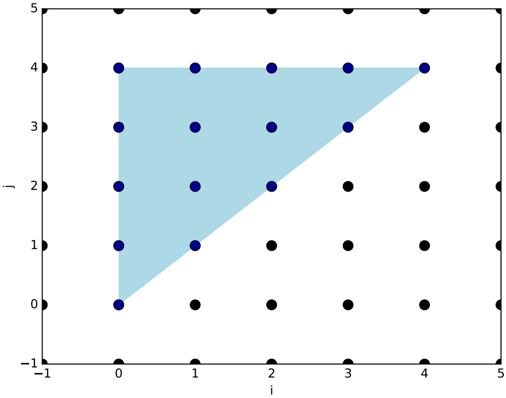
Let's solve it!



How many integer points can we find in this set?

$$\frac{matrix\_size^2 + matrix\_size}{2}$$

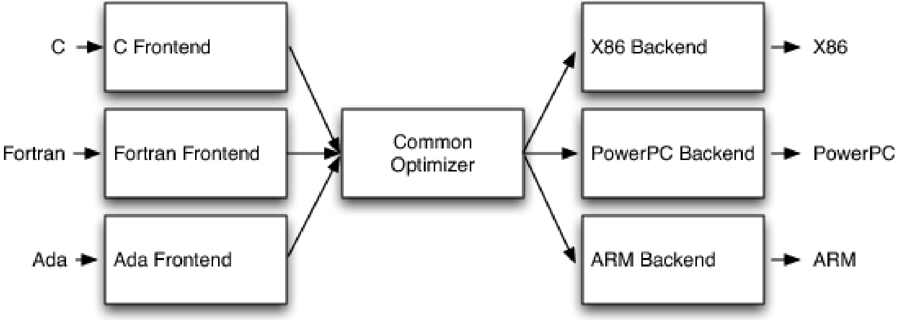
Let's solve it!



That's nice, but can we automatically...

- map loops to such sets?
- count them?

Loop optimizations



Optimizations can be tricky

- loop tiling
- loop fusion
- loop unrolling
- loop parallelization
- loop vectorization
- memory access patterns
- mapping iterations to cores on heterogeneous hardware

Image source: *The Architecture of Open Source Applications*

Integer polyhedra

Polyhedra

The set of solutions to a system of m affine inequalities $Ax \leq b$

$$P = \{x \in \mathbb{Z}^n \mid Ax \leq b\}$$

where $A \in \mathbb{Z}^{m \times n}$, $x \in \mathbb{Z}^n$ a $b \in \mathbb{Z}^m$.

Polyhedral modeling of code

```
for(int i = 0; i <= N; ++i) {
  if(i <= N - 50)
S1:   A[5*i] = 1;
  else
S2:   A[3*i] = 2;

%  for(int j = 0;j <= N; ++j)
%S3:   B[i][2*j] = 3;
}
```

- Static Control Part (SCoP) - single entry node and single exit
- static control flow and memory access - **for** loops and **if** statements
- extensions to this model are possible

Polyhedral modeling of code

```
for(int i = 0; i <= N; ++i) {
  if(i <= N - 50)
S1:   A[5*i] = 1;
  else
S2:   A[3*i] = 2;

%  for(int j = 0;j <= N; ++j)
%S3:   B[i][2*j] = 3;
}
```

$D_{S1} = \{S1[i] : i \geq 0 \wedge i \leq N \wedge i \leq N - 50\}$
$S_{S1} = \{S1[i] \rightarrow [0, i, 0, 0, 0]\}$

$D_{S2} = \{S2[i] : i \geq 0 \wedge i \leq N \wedge i > N - 50\}$
$S_{S2} = \{S2[i] \rightarrow [0, i, 1, 0, 0]\}$

Transformations - loop blocking

$$D_f = \{f[i,j] : 0 \leq i \leq N \wedge 0 \geq j < M\}$$
$$S_f = \{f[i,j] \rightarrow [i,j]\}$$

What is S_f?

```
for(int i = 0; i <= M; ++i)
  for(int j = 0;j <= N; ++j)
    f(i, j);
```

Transformations - loop blocking

$$D_f = \{f[i,j] : 0 \leq i \leq N \wedge 0 \geq j < M\}$$
$$S_f = \{f[i,j] \rightarrow [i,j]\}$$

- $\mathcal{T}_{StripMineOuter} = \{[s_0, s_1] \rightarrow [t, s_0, s_1] : t \mod 4 = 0 \wedge t \leq s_0 < t + 4\}$

What is $S_f \circ \mathcal{T}_{StripMineOuter}$?

```
for(int ti = 0; ti <= M; ti += 4)
  for(int i = ti; i < min(M, ti + 4); ++i)
    for(int j = 0; j <= N; ++j)
       f(i, j);
```

Transformations - loop blocking

$$D_f = \{f[i,j] : 0 \leq i \leq N \wedge 0 \geq j < M\}$$
$$S_f = \{f[i,j] \rightarrow [i,j]\}$$

- $\mathcal{T}_{StripMineOuter} = \{[s_0, s_1] \rightarrow [t, s_0, s_1] : t \mod 4 = 0 \wedge t \leq s_0 < t + 4\}$
- $\mathcal{T}_{StripMineInner} = \{[s_0, s_1, s_2] \rightarrow [s_0, s_1, t, s_2] : t \mod 4 = 0 \wedge t \leq s_2 < t + 4\}$

What is $S_f \circ \mathcal{T}_{StripMineInner} \circ \mathcal{T}_{StripMineOuter}$?

```
for(int ti = 0; ti <= M; ti += 4)
  for(int i = ti; i < min(M, ti + 4); ++i)
    for(int tj = 0; tj <= N; tj += 4)
      for(int j = tj; j <= min(N, tj + 4); ++j)
        f(i, j);
```

Transformations - loop blocking

$$D_f = \{f[i,j] : 0 \leq i \leq N \wedge 0 \geq j < M\}$$
$$S_f = \{f[i,j] \to [i,j]\}$$

- $\mathcal{T}_{StripMineOuter} = \{[s_0, s_1] \to [t, s_0, s_1] : t \mod 4 = 0 \wedge t \leq s_0 < t + 4\}$
- $\mathcal{T}_{StripMineInner} = \{[s_0, s_1, s_2] \to [s_0, s_1, t, s_2] : t \mod 4 = 0 \wedge t \leq s_2 < t + 4\}$
- $\mathcal{T}_{Interchange} = \{[s_0, s_1, s_2, s_3] \to [s_0, s_2, s_1, s_3]\}$

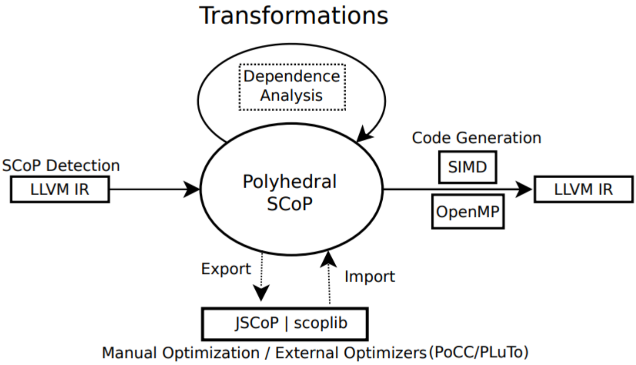
What is $S_f \circ \mathcal{T}_{Interchange} \circ \mathcal{T}_{StripMineInner} \circ \mathcal{T}_{StripMineOuter}$?

```
for(int ti = 0; ti <= M; ti += 4)
  for(int tj = 0; tj <= N; tj += 4)
    for(int i = ti; i < min(M, ti + 4); ++i)
      for(int j = tj; j <= min(N, tj + 4); ++j)
        f(i, j);
```

Polly

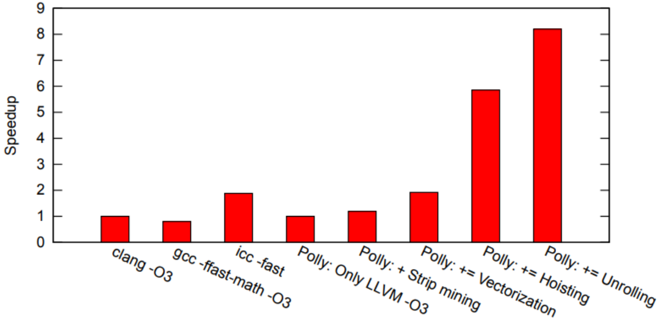
- we know already that transformations can be **composed** - main idea behind successful polyhedra optimizations since GRAPHITE in GCC.
- a powerful way to optimize performance on an abstract model of program
- can we apply language-agnostic mathematical transformations to a polyhedral model of language-agnostic representation?

Polly



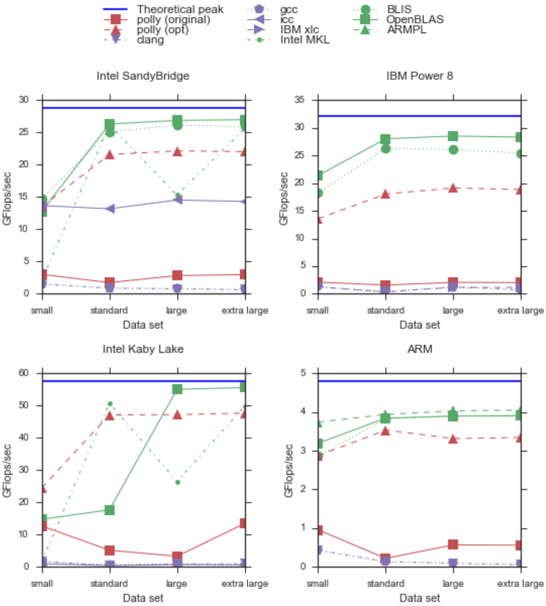
- polyhedral framework for optimizing LLVM IR
- code generation: sequential, OpenMP, GPUs

Polly - does it work?



32x32 GEMM, serial on Intel CPU.

Polly - does it work?



Let's go back to the counting

```
for(int i = 0; i < matrix_size; ++i) {
  for(int j = i; j < matrix_size; ++j) {
    sum += 1.5 * i;
  }
}
```

- Extraction with Polly-based Scalar Evolution
  `[matrix_size] -> [i0, i1] : i0 >= 0 and 0 <= i1 < matrix_size - i0`
- Use Barvinok library to count points in integer sets
  `[matrix_size] -> (1/2 * matrix_size + 1/2 * matrix_size^2) : matrix_size > 0`

Summary

The Good

- polyhedra capable of modeling huge parts of code
- accurate information about control flow
- composable code transformations
- yet another practical application of a mature field of mathematics

Summary

The Good

- polyhedra capable of modeling huge parts of code
- accurate information about control flow
- composable code transformations
- yet another practical application of a mature field of mathematics

The Bad

- NP-completeness of many problems on integer polyhedra
- exponential complexity in dimension for many algorithms
- overhead and accuracy of analysis is hard to predict

Summary

The Good

- polyhedra capable of modeling huge parts of code
- accurate information about control flow
- composable code transformations
- yet another practical application of a mature field of mathematics

The Bad

- NP-completeness of many problems on integer polyhedra
- exponential complexity in dimension for many algorithms
- overhead and accuracy of analysis is hard to predict

and the Ugly

- does not encode non-affine relations
- code with dynamic control flow is hard to process

Try it!

- polyhedral.info
- https://polly.llvm.org/
- *The Many Aspects of Counting Lattice Points in Polytopes*, JA De Loera

Why do we care about that?

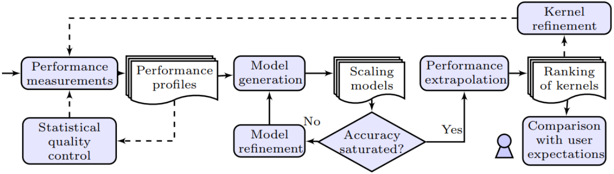
- automatic parallel complexity analysis

  My loop has $\frac{n \log_2 n}{T}$ *iterations on* T *threads, can it scale efficiently?*
- feedback for developers to prevent performance bugs

```
void vec_normalize(double * vec, int len
for(int i = 0; i < len; ++i)
vec[i] /= norm(vec, len);
}
```

Is the complexity of this loop really linear?

- performance modeling



Pluto

- polyhedral source-to-source compiler for parallelism and data locality
- introduced at PLDI in 2008
- automatically find best tiling configuration for independent loops