



INRIA

INVENTEURS DU MONDE NUMÉRIQUE



caMus



APOLLO

Automatic speculative POLyhedral Loop Optimizer

*Juan Manuel Martinez Caamaño, Aravind Sukumaran-Rajam,
Artiom Baloian, Manuel Selva, Philippe Claus*

INRIA CAMUS, ICube lab., CNRS
University of Strasbourg, France

Summary

The Compilation Process

The Polyhedral Model

What it is?

Limits

APOLLO

Overview

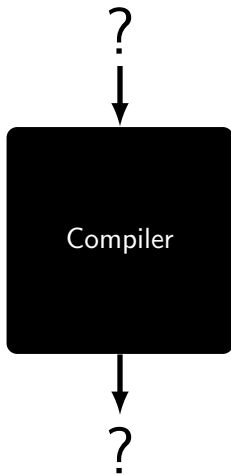
Compiler

Runtime

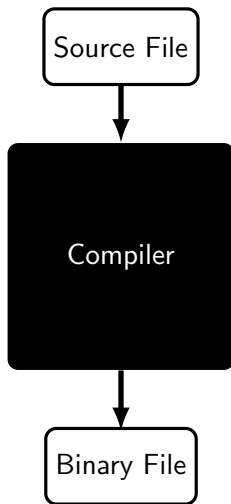
Conclusion



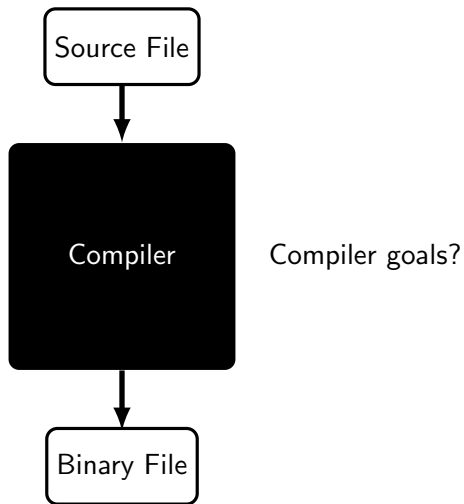
What is a Compiler?



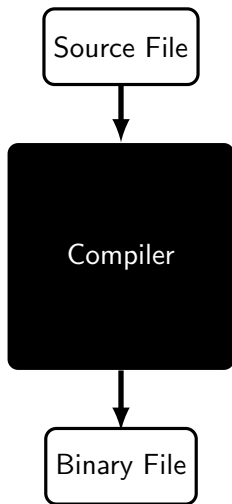
What is a Compiler?



What is a Compiler?



What is a Compiler?



Compiler goals?

Correctness

Performance of the code

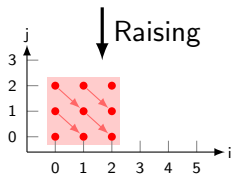
Size of the code

Polyhedral Compilation

```
for (int i = 0; i < 3; i++)  
  for (int j = 0; j < 3; j++)  
    z[i+j] += x[i] * y[j];
```

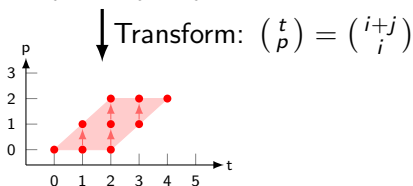
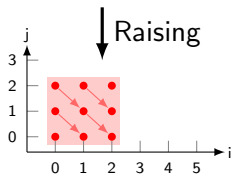
Polyhedral Compilation

```
for (int i = 0; i < 3; i++)  
  for (int j = 0; j < 3; j++)  
    z[i+j] += x[i] * y[j];
```



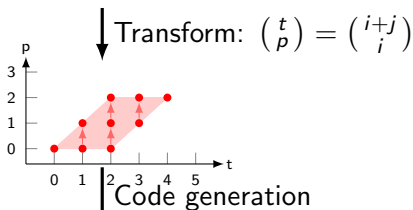
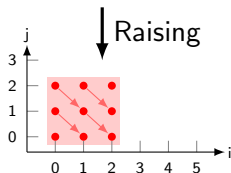
Polyhedral Compilation

```
for (int i = 0; i < 3; i++)  
  for (int j = 0; j < 3; j++)  
    z[i+j] += x[i] * y[j];
```



Polyhedral Compilation

```
for (int i = 0; i < 3; i++)  
  for (int j = 0; j < 3; j++)  
    z[i+j] += x[i] * y[j];
```

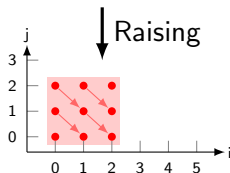


↓ Code generation

```
#pragma omp parallel for  
for (int t = 0; t < 5; t++)  
  for (int p = max(0,t-2); p <= min(2,t); p++)  
    z[t] += x[p] * y[t-p];
```

Polyhedral Compilation

```
for (int i = 0; i < 3; i++)  
  for (int j = 0; j < 3; j++)  
    z[i+j] += x[i] * y[j];
```

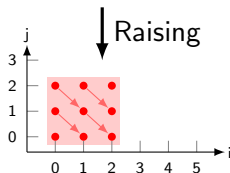


Raising on Static Control Parts (SCoPs) only

- ▶ Affine loop bounds
- ▶ Affine conditionals
- ▶ Affine memory accesses

Polyhedral Compilation

```
for (int i = 0; i < 3; i++)  
  for (int j = 0; j < 3; j++)  
    z[i+j] += x[i] * y[j];
```



Raising on Static Control Parts (SCoPs) only

- ▶ Affine loop bounds
- ▶ Affine conditionals
- ▶ Affine memory accesses

SCoPs cannot always be detected statically

- ▶ Need for runtime mechanisms
- ▶ APOLLO

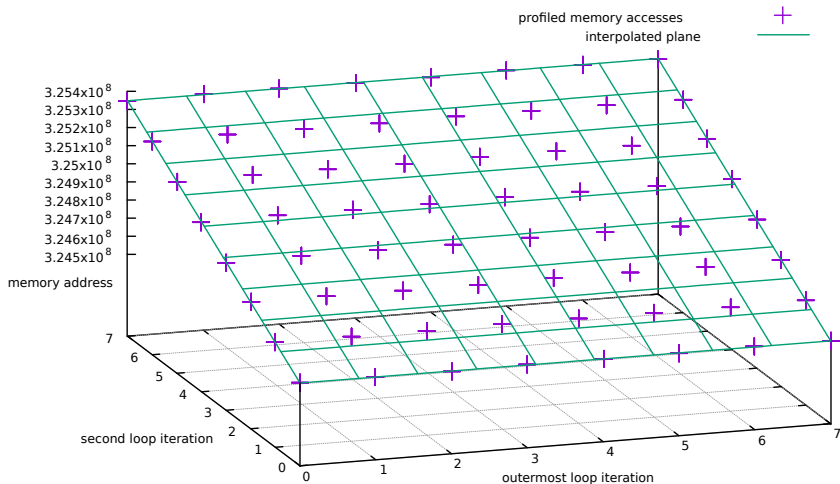
Sparse Matrix * Dense Matrix

```
// Iterate over rows of sparse matrix
for (i = 1; i <= left->size; i++) {
    elem = left->firstInRow[i];

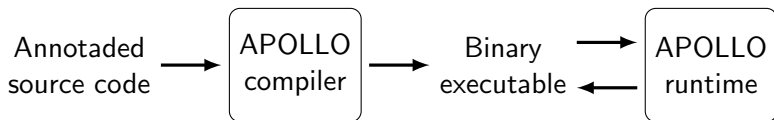
    // Iterate elements in row of spare matrix
    while (elem) {

        // Iterate over columns in dense matrix
        for (col = 1; col <= cols; col++) {
            res[row][col] +=
                elem->value * right[elem->col][col];
        }
        elem = elem->next;
    }
}
```

Sparse Matrix * Dense Matrix - At Runtime



APOLLO - Automatic Speculative POLyhedral Loop Optimizer



APOLLO Compiler

Virtual Iterators - Handling any kind of loop consistently

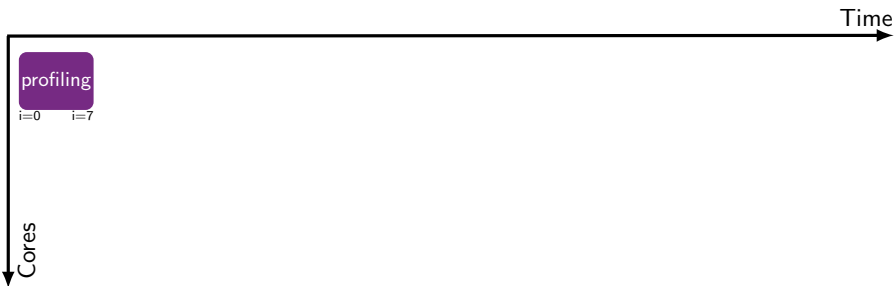
- ▶ Inserted at each level of the target loop nest
- ▶ Starting at zero with step one

Extraction of Code-Bones

- ▶ Assembled at runtime for optimization

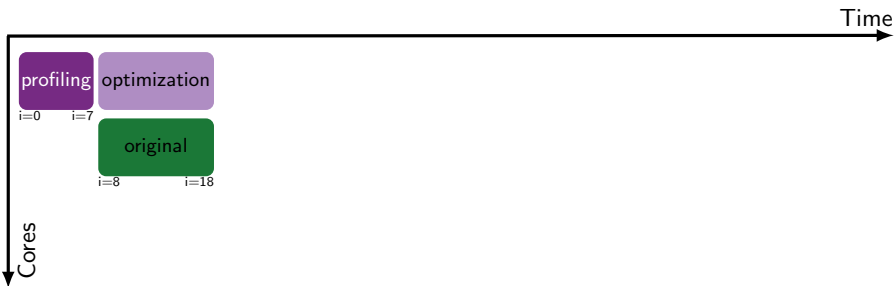


APOLLO Runtime



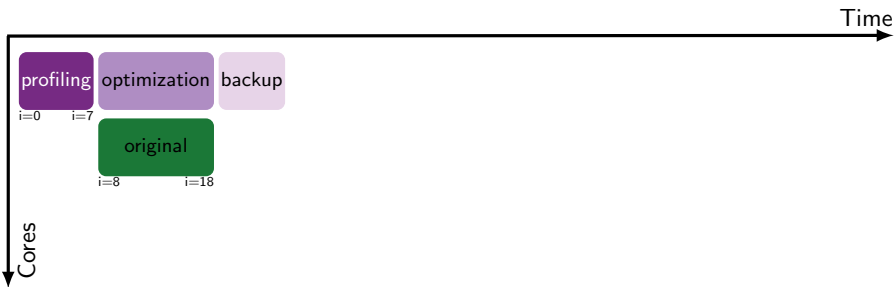
```
for (i = 1; i <= left->size; i++) {
    elem = left->firstInRow[i];
    while (elem) {
        for (col = 1; col <= cols; col++) {
            res[row][col] +=
                elem->value * right[elem->col][col];
        }
        elem = elem->next ;
    }
}
```

APOLLO Runtime



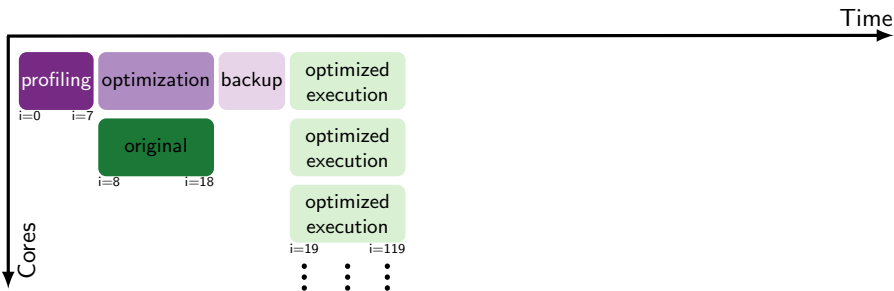
```
for (i = 1; i <= left->size; i++) {  
    elem = left->firstInRow[i];  
    while (elem) {  
        for (col = 1; col <= cols; col++) {  
            res[row][col] +=  
                elem->value * right[elem->col][col];  
        }  
        elem = elem->next ;  
    }  
}
```

APOLLO Runtime



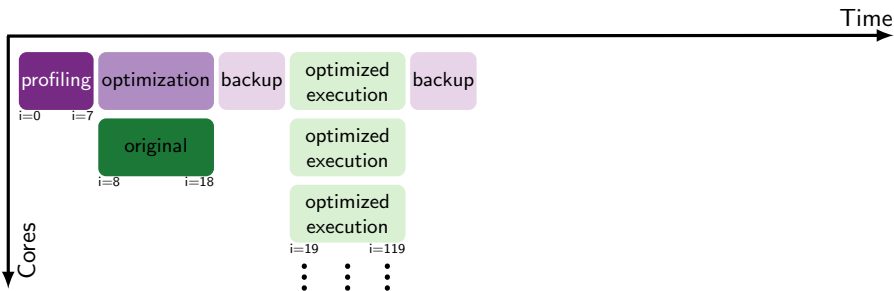
```
for (i = 1; i <= left->size; i++) {
    elem = left->firstInRow[i];
    while (elem) {
        for (col = 1; col <= cols; col++) {
            res[row][col] +=
                elem->value * right[elem->col][col];
        }
        elem = elem->next ;
    }
}
```

APOLLO Runtime



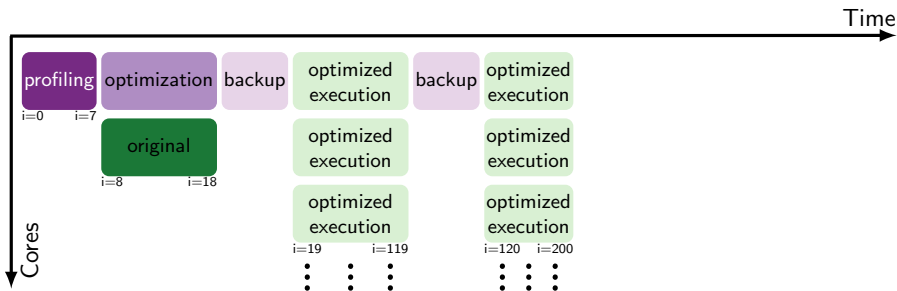
```
for (i = 1; i <= left->size; i++) {  
    elem = left->firstInRow[i];  
    while (elem) {  
        for (col = 1; col <= cols; col++) {  
            res[row][col] +=  
                elem->value * right[elem->col][col];  
        }  
        elem = elem->next ;  
    }  
}
```

APOLLO Runtime



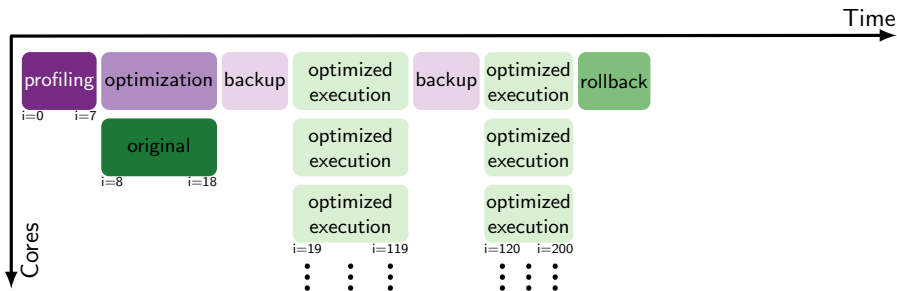
```
for (i = 1; i <= left->size; i++) {  
    elem = left->firstInRow[i];  
    while (elem) {  
        for (col = 1; col <= cols; col++) {  
            res[row][col] +=  
                elem->value * right[elem->col][col];  
        }  
        elem = elem->next ;  
    }  
}
```

APOLLO Runtime



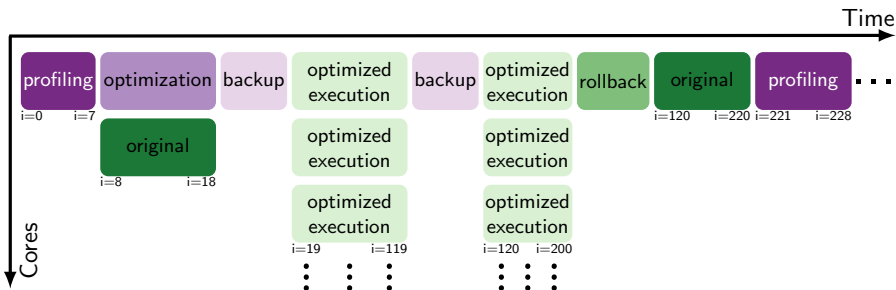
```
for (i = 1; i <= left->size; i++) {  
    elem = left->firstInRow[i];  
    while (elem) {  
        for (col = 1; col <= cols; col++) {  
            res[row][col] +=  
                elem->value * right[elem->col][col];  
        }  
        elem = elem->next ;  
    }  
}
```

APOLLO Runtime



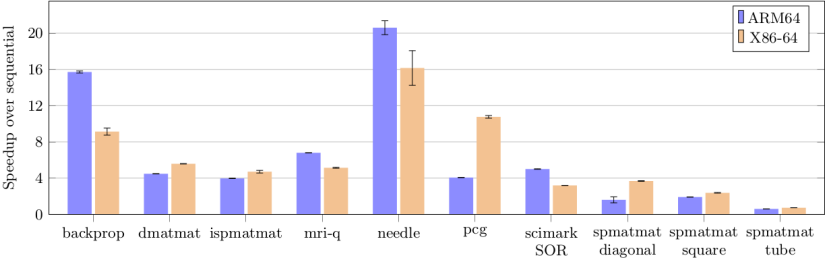
```
for (i = 1; i <= left->size; i++) {  
    elem = left->firstInRow[i];  
    while (elem) {  
        for (col = 1; col <= cols; col++) {  
            res[row][col] +=  
                elem->value * right[elem->col][col];  
        }  
        elem = elem->next ;  
    }  
}
```

APOLLO Runtime



```
for (i = 1; i <= left->size; i++) {  
    elem = left->firstInRow[i];  
    while (elem) {  
        for (col = 1; col <= cols; col++) {  
            res[row][col] +=  
                elem->value * right[elem->col][col];  
        }  
        elem = elem->next ;  
    }  
}
```


Results



- ▶ 8 threads
- ▶ Speedups over clang -O3



Conclusion

Open source software

- ▶ APOLLO 1.1.0¹
- ▶ APOLLO Benchmarks 1.0.0²

On going work

- ▶ Polyhedral profiler based on APOLLO
- ▶ Memory reallocation (virtually) to fit SCoPs
- ▶ Polyhedral optimization for dynamic languages (JavaScript)



¹<http://apollo.gforge.inria.fr>

²<https://scm.gforge.inria.fr/anonscm/git/apollo-benchs>

THANK YOU

The Inria logo is displayed in a white rounded square with a red border. The word "Inria" is written in a red, cursive script font.

Inria

University of Strasbourg
INRIA, ICube lab., CNRS
<http://team.inria.fr/camus>