UNIVERSITÉ PARIS SUD

# Contributions to High-Level Program Optimization

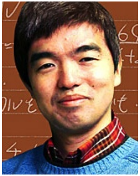
Cédric Bastoul

LRI, University of Paris-Sud / CNRS UMR 8623 / INRIA Saclay Île-de-France

December 12, 2012

# Outstanding Optimization Heroes

- John Carmack
  Game developer, id Software
  innovations in 3D Graphics, fast inverse square root



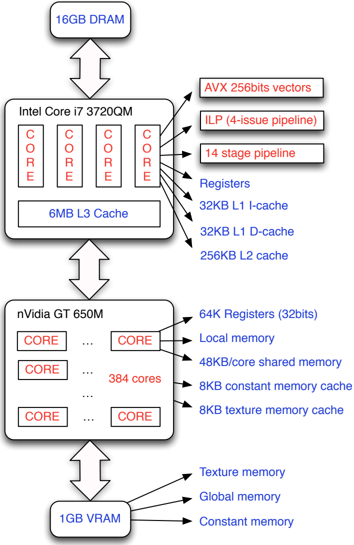
- Kazushige Goto
  Engineer, Intel
  hand-tuned programs for supercomputers
  "Goto BLAS"

- Vasily Volkov
  PhD student, Berkeley
  3x faster FFT than nVidia's proprietary implementation

# "Heroes" Is Not Strong Enough

This laptop:

- 2.7 Billion transistors
- 5 types of parallelism
- At least 4 programming models + APIs
- 15 types of memory



# Ways to Optimized Applications

1. Parallel Languages (Cilk, Chapel, $C^n$, CUDA, Fortress, HPF, UPC, X10...)
   - High-level abstraction (mathematical objects, trees...)
   - Parallel programming idioms (forall, reductions, tasks...)
   - High-level control (data distribution, alignment...)
2. Libraries (IPP, LAPACK, MKL, Parallel VSIPL++...)
   - High-performance routines (BLAS, FFT...)
   - Auto-tuning capabilities (ATLAS, FFTW, SPIRAL...)
3. Compilers (GCC, XL, ICC...)
   - Language extensions (OpenMP, OpenACC)
   - *Automatic parallelization and optimization* (loop level, SIMD...)

Demo

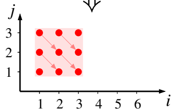
# Outline

1. Introduction
2. **The Polyhedral Framework**
3. More Useful: Optimization-Centric Compilation
4. More Efficient: Looking for Optimizing Transformations
5. In More Situations: Polyhedral Model Extensions
6. Conclusion & Perspectives

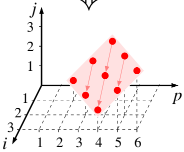
# Polyhedral Framework Overview

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

1. Raising




2. Transformation




3. Code generation

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

# Polyhedral Relation

## Polyhedral Relation Component

$$\mathcal{R}(\vec{p}) = \left\{ \vec{x}_{in} \to \vec{x}_{out} \in \mathbb{Z}^{dim(\vec{x}_{in})} \times \mathbb{Z}^{dim(\vec{x}_{out})} \;\middle|\; \exists \vec{l} \in \mathbb{Z}^{dim(\vec{l})} : \begin{bmatrix} A_{out} & A_{in} & L & P & \vec{c} \end{bmatrix} \begin{pmatrix} \vec{x}_{out} \\ \vec{x}_{in} \\ \vec{l} \\ \vec{p} \\ 1 \end{pmatrix} \geq \vec{0} \right\}$$

- $\vec{p}$ is the vector of *parameters*,
- $\vec{x}_{in}$ is an input coordinate,
- $\vec{x}_{out}$ is an output coordinate,
- $\vec{l}$ is the vector of *local variables*,
- $A_{out}$, $A_{in}$, $L$ and $P$ are integer matrices,
- $\vec{c}$ is an integer vector,

- Originally proposed by Kelly and Pugh [Kelly & Pugh, TR 1993]
- Research on $\mathbb{Z}$-polyhedra, Generalized Change of Basis, etc. made it applicable only recently [Verdoolaege, isl 2010]

# Thinking Instancewise

## Polynomial Multiply

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

Program execution:

```
1: z[0] += x[0] * y[0];
2: z[1] += x[0] * y[1];
3: z[2] += x[0] * y[2];
4: z[1] += x[1] * y[0];
5: z[2] += x[1] * y[1];
6: z[3] += x[1] * y[2];
7: z[2] += x[2] * y[0];
8: z[3] += x[2] * y[1];
9: z[4] += x[2] * y[2];
```

A few observations:

- The statement is executed 9 times
- A unique value of $i,j$ is associated to each of these 9 instances
- Each instance can access different data
- The 9 instances are ordered

# Raising Step

**Program**

Static Control Code

**Iteration Domain (instances)**

$$\mathcal{D}_S(\vec{p}) = \left\{ () \rightarrow \vec{\imath}_S \in \mathbb{Z}^{dim(\vec{\imath}_S)} \;\middle|\; [D_S]\begin{pmatrix}\vec{\imath}_S\\ \vec{p}\\ 1\end{pmatrix} \geq \vec{0}\right\}$$

**Access Relation (data)**

$$\mathcal{A}_{S,r}(\vec{p}) = \left\{ \vec{\imath}_S \rightarrow \vec{a}_{S,r} \in \mathbb{Z}^{dim(\vec{\imath}_S)} \times \mathbb{Z}^{dim(\vec{a}_{S,r})} \;\middle|\; [A_{S,r}]\begin{pmatrix}\vec{a}_{S,r}\\ \vec{\imath}_S\\ \vec{p}\\ 1\end{pmatrix} \geq \vec{0}\right\}$$

**Mapping Relation (orders)**

$$\theta_S(\vec{p}) = \left\{ \vec{\imath}_S \rightarrow \vec{t}_S \in \mathbb{Z}^{dim(\vec{\imath}_S)} \times \mathbb{Z}^{dim(\vec{t}_S)} \;\middle|\; [T_S]\begin{pmatrix}\vec{t}_S\\ \vec{\imath}_S\\ \vec{p}\\ 1\end{pmatrix} \geq \vec{0}\right\}$$

- Low-level compilers (GCC GRAPHITE, IBM XL, LLVM Polly, ORC WRAP-IT...)
  [Bastoul et al. LCPC'03], [Girbal et al. IJPP'06], [Pop et al. GCC'06]
- High-level compilers (R-Stream, PoCC, Pluto, Rose/Bee, CHiLL...)
  [Bastoul et al. PMEA'09], [Bastoul and Pouchet, Clan 2008-2012]

# Raising Step

## Polynomial Multiply

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

## Iteration Domain (instances)

$$\mathcal{D}_S(N) = \left\{ () \rightarrow \begin{pmatrix} i \\ j \end{pmatrix} \in \mathbb{Z}^2 \middle| \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & -1 \end{bmatrix} \begin{pmatrix} i \\ j \\ N \\ 1 \end{pmatrix} \geq \vec{0} \right\}$$

## Access Relation of `z[i+j]` (data)

$$\mathcal{A}_{S,1}(N) = \left\{ \begin{pmatrix} i \\ j \end{pmatrix} \rightarrow \left( a_{S,1} \right) \in \mathbb{Z}^2 \times \mathbb{Z} \middle| \begin{bmatrix} -1 & 1 & 1 & 0 & 0 \end{bmatrix} \begin{pmatrix} a_{S,1} \\ i \\ j \\ N \\ 1 \end{pmatrix} = \vec{0} \right\}$$

## Mapping Relation (orders)

$$\theta_S(N) = \left\{ \begin{pmatrix} i \\ j \end{pmatrix} \rightarrow \begin{pmatrix} t_S^1 \\ t_S^2 \\ t_S^3 \\ t_S^4 \\ t_S^5 \end{pmatrix} \in \mathbb{Z}^2 \times \mathbb{Z}^5 \middle| \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{pmatrix} t_S^1 \\ t_S^2 \\ t_S^3 \\ t_S^4 \\ t_S^5 \\ i \\ j \\ N \\ 1 \end{pmatrix} = \vec{0} \right\}$$

- **Low-level compilers** (GCC GRAPHITE, IBM XL, LLVM Polly, ORC WRAP-IT...)
  [Bastoul et al. LCPC'03], [Girbal et al. IJPP'06], [Pop et al. GCC'06]
- **High-level compilers** (R-Stream, PoCC, Pluto, Rose/Bee, CHiLL...)
  [Bastoul et al. PMEA'09], [Bastoul and Pouchet, Clan 2008-2012]

# Transformation Step

## Iteration Domain

$$\mathcal{D}_S(\vec{p}) = \left\{ () \to \vec{\imath}_S \in \mathbb{Z}^{dim(\vec{\imath}_S)} \;\middle|\; [D_S] \begin{pmatrix} \vec{\imath}_S \\ \vec{p} \\ 1 \end{pmatrix} \geq \vec{0} \right\}$$

## Mapping Relation

$$\theta_S(\vec{p}) = \left\{ \vec{\imath}_S \to \vec{t}_S \in \mathbb{Z}^{dim(\vec{t}_S)} \times \mathbb{Z}^{dim(\vec{\imath}_S)} \;\middle|\; [T_S] \begin{pmatrix} \vec{t}_S \\ \vec{\imath}_S \\ \vec{p} \\ 1 \end{pmatrix} \geq \vec{0} \right\}$$

## Generalized Change of Basis

$$\mathcal{T}_S(\vec{p}) = \left\{ () \to \begin{pmatrix} \vec{t}_S \\ \vec{\imath}_S \end{pmatrix} \in \mathbb{Z}^{dim(\vec{t}_S)+dim(\vec{\imath}_S)} \;\middle|\; \left[ \begin{array}{c|c} \multicolumn{2}{c}{T_S} \\ \hline 0 & D_S \end{array} \right] \begin{pmatrix} \vec{t}_S \\ \hline \vec{\imath}_S \\ \vec{p} \\ 1 \end{pmatrix} \geq \vec{0} \right\}$$

- Complex transformations applied as a single trivial step
  [Le Verge, TR 1995] [Bastoul PACT'04]
- GCB can be mixed with other transformation schemes

# Transformation Step

## Iteration Domain

$$\mathcal{D}_S(N) = \left\{ () \rightarrow \begin{pmatrix} i \\ j \end{pmatrix} \in \mathbb{Z}^2 \;\middle|\; \left[\begin{array}{rr|r|r} 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & -1 \end{array}\right] \begin{pmatrix} i \\ j \\ \hline N \\ \hline 1 \end{pmatrix} \geq \vec{0} \right\}$$

## Mapping Relation

$$\theta_S(N) = \left\{ \begin{pmatrix} i \\ j \end{pmatrix} \rightarrow \begin{pmatrix} p \end{pmatrix} \in \mathbb{Z}^2 \times \mathbb{Z}^2 \;\middle|\; \left[\begin{array}{r|rr|r|r} -1 & 1 & 1 & 0 & 0 \end{array}\right] \begin{pmatrix} p \\ \hline i \\ j \\ \hline N \\ \hline 1 \end{pmatrix} = \vec{0} \right\}$$

## Generalized Change of Basis

$$\mathcal{T}_S(N) = \left\{ () \rightarrow \begin{pmatrix} p \\ \hline i \\ j \end{pmatrix} \in \mathbb{Z}^4 \;\middle|\; \left[\begin{array}{r|rr|r|r} -1 & 1 & 1 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & -1 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & -1 & 1 & 0 \end{array}\right] \begin{pmatrix} p \\ \hline i \\ j \\ \hline N \\ \hline 1 \end{pmatrix} \begin{array}{c} = \\ \hline \geq \end{array} \vec{0} \right\}$$

- Complex transformations applied as a single trivial step

  [Le Verge, TR 1995] [Bastoul PACT'04]

- GCB can be mixed with other transformation schemes

# Code Generation Step

**Transformed Domain(s)**

$$\mathcal{T}_S(\vec{p}) = \left\{ () \to \begin{pmatrix}\vec{t}_S\\ \vec{\imath}_S\end{pmatrix} \in \mathbb{Z}^{dim(\vec{t}_S)+dim(\vec{\imath}_S)} \;\middle|\; \left[\begin{array}{c|c} \multicolumn{2}{c}{T_S} \\ \hline 0 & D_S\end{array}\right] \begin{pmatrix}\vec{t}_S\\ \hline \vec{\imath}_S\\ \vec{p}\\ 1\end{pmatrix} \ge \vec{0}\right\}$$

↓

**Target Program**

Generated Code

- Considered as the weak spot of the framework until CLooG
  [Bastoul, Verdoolaege et al., CLooG 2001-2012], [Bastoul ISPDC'03]
  [Bastoul PACT'04], [Vasilache et al., CC'06], [Bastoul et al., PMEA'09]
- Extensions to the original QRW's algorithm
  [Quilleré, Rajopadhye, Wilde, IJPP'00]

# Code Generation Step

## Transformed Domain(s)

$$\mathcal{T}_S(N) = \left\{ () \rightarrow \begin{pmatrix} p \\ \hline i \\ j \end{pmatrix} \in \mathbb{Z}^4 \;\middle|\; \left[\begin{array}{c|cc|c|c} -1 & 1 & 1 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & -1 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & -1 & 1 & 0 \end{array}\right] \begin{pmatrix} p \\ \hline i \\ j \\ \hline N \\ \hline 1 \end{pmatrix} \begin{array}{c} = \\ \hline \geq \end{array} \vec{0} \right\}$$

## Target Polynomial Multiply

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

- Considered as the weak spot of the framework until CLooG

  [Bastoul, Verdoolaege et al., CLooG 2001-2012], [Bastoul ISPDC'03]

  [Bastoul PACT'04], [Vasilache et al., CC'06], [Bastoul et al., PMEA'09]

- Extensions to the original QRW's algorithm

  [Quilleré, Rajopadhye, Wilde, IJPP'00]

# Conclusion on the Polyhedral Framework

- Maturation to polyhedral relations
  - Generalized Change of Basis / Scattering
  - OpenScop standardization effort
- Code Generation
  - Extensions to the QRW's code generation algorithm
  - Robust and Scalable code generation
  - Limited-overhead code generation
- Raising
  - Technical participation to various frameworks
  - Static Control as a programming model view
- Raising from compiler IR is a complex task
- Manipulation involves high-complexity techniques

Main contributing references: [Bastoul, PACT'04], [Vasilache et al., CC'06]

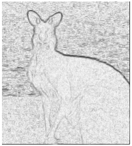
# Outline

# On the Hardship to Apply Optimizations

## Edge Detection For Noisy Images

```
/* Ring Blur Filter */
for (i = 1; i < length - 1; i++)
  for (j = 1; i < width - 1; j++)
    Ring[i][j] = (Img[i-1][j-1] + Img[i-1][j] + Img[i-1][j+1]+
                  Img[i][j-1]    +                Img[i][j+1]   +
                  Img[i+1][j-1] + Img[i+1][j] + Img[i+1][j+1])/8;

/* Roberts Edge Detection Filter */
for (i = 1; i < length - 2; i++)
  for (j = 2; i < width - 1; j++)
    Img[i][j] = abs(Ring[i][j]   - Ring[i+1][j-1])+
                abs(Ring[i+1][j] - Ring[i][j-1]);
```

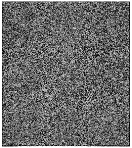


Experts or compilers may wish to fuse loop for locality, but...

# On the Hardship to Apply Optimizations

## Edge Detection For Noisy Images

```
...
/* Ring and Roberts Filters Trivially Fused */
for (i = 1; i < length - 2; i++) {
  for (j = 2; j < width - 1; j++) {
    Ring[i][j] = (Img[i-1][j-1] + Img[i-1][j] + Img[i-1][j+1]+
                  Img[i][j-1]   +               Img[i][j+1]   +
                  Img[i+1][j-1] + Img[i+1][j] + Img[i+1][j+1])/8;
    Img[i][j] = abs(Ring[i][j]   - Ring[i+1][j-1]) +
                abs(Ring[i+1][j] - Ring[i][j-1]);
  }
}
...
```



Experts or compilers may wish to fuse loop for locality, but...

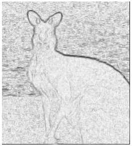
- Data dependences prevent trivial loop fusion

# On the Hardship to Apply Optimizations

## Edge Detection For Noisy Images

```
/* Ring Blur Filter */
for (i = 1; i < length - 1; i++)
  for (j = 1; i < width - 1; j++)
    Ring[i][j] = (Img[i-1][j-1] + Img[i-1][j] + Img[i-1][j+1]+
                  Img[i][j-1]   +               Img[i][j+1]   +
                  Img[i+1][j-1] + Img[i+1][j] + Img[i+1][j+1])/8;

/* Roberts Edge Detection Filter */
for (i = 1; i < length - 2; i++)
  for (j = 2; i < width - 1; j++)
    Img[i][j] = abs(Ring[i][j]   - Ring[i+1][j-1])+
                abs(Ring[i+1][j] - Ring[i][j-1]);
```



Experts or compilers may wish to fuse loop for locality, but...

- Data dependences prevent trivial loop fusion

We need analyses and techniques to overcome this issue

# Violated Dependence Analysis

Given a transformation, the exact set of illegaly ordered pair amongst instances in dependence may be represented using relations [Vasilache et al. ICS'06]

## Violation Relation (see manuscript for details)

$$\upsilon_{S,r_S \xrightarrow{d,v} T,r_T}(\vec{p}) = \left\{ \begin{pmatrix} \vec{\imath}_S \\ \vec{a}_{S,r_S} \\ \vec{t}_S \end{pmatrix} \rightarrow \begin{pmatrix} \vec{\imath}_T \\ \vec{a}_{T,r_T} \\ \vec{t}_T \end{pmatrix} \in \begin{matrix} \mathbb{Z}^{dim(\vec{\imath}_S)+dim(\vec{a}_{S,r_S})+dim(\vec{t}_S)} \\ \times \\ \mathbb{Z}^{dim(\vec{\imath}_R)+dim(\vec{a}_{T,r_T})+dim(\vec{t}_T)} \end{matrix} \;\middle|\; \Upsilon_{S,r_S \xrightarrow{d,v} T,r_T} \right\}$$

Applications:

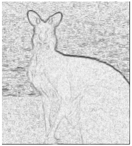
- Apply expansion or privatization only when necessary [Leung et al. R-Stream 2010]
- Find limited deviation to the transformation to make it legal [Vasilache et al. PACT'07], [Bastoul, HDR 2012]

# Correcting an Illegal Transformation

## Edge Detection For Noisy Images

```
/* Ring Blur Filter */
for (i = 1; i < length - 1; i++)
  for (j = 1; i < width - 1; j++)
    Ring[i][j] = (Img[i-1][j-1] + Img[i-1][j] + Img[i-1][j+1]+
                  Img[i][j-1]   +               Img[i][j+1]   +
                  Img[i+1][j-1] + Img[i+1][j] + Img[i+1][j+1])/8;

/* Roberts Edge Detection Filter */
for (i = 1; i < length - 2; i++)
  for (j = 2; i < width - 1; j++)
    Img[i][j] = abs(Ring[i][j]   - Ring[i+1][j-1])+
                abs(Ring[i+1][j] - Ring[i][j-1]);
```

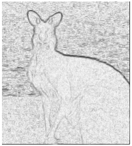


Loop fusion is **not** legal, but...

# Correcting an Illegal Transformation

## Edge Detection For Noisy Images

```
...
for (i = 3; i < length - 1; i++) {
   Ring[i][1] = (Img[i-1][0] + Img[i-1][1] + Img[i-1][2]+
                 Img[i][0]    +                Img[i][2]   +
                 Img[i+1][0] + Img[i+1][1] + Img[i+1][2])/8;
  for (j = 2; j < width - 1; j++) {
    Ring[i-2][j] = (Img[i-3][j-1] + Img[i-3][j] + Img[i-3][j+1]+
                    Img[i-2][j+1] +                Img[i-2][j-1]+
                    Img[i-1][j-1] + Img[i-1][j] + Img[i-1][j+1])/8;
    Img[i][j] = abs(Ring[i][j]    - Ring[i+1][j-1])+
                abs(Ring[i+1][j] - Ring[i][j-1]);
...
```



Loop fusion is **not** legal, but... It can be corrected by *shifting*

$\sim$ 50% cache miss reduction!

# Conclusion on Optimization-Centric Compilation

- Instancewise Data Dependence Analysis
  - Fast legality check
  - Use dependence regularity to achieve scalability
  - Demonstrated scalability on large SCoPs (SPEC2000 FP)
- User Accessibility
  - Violated dependence analysis
  - Clay semi-automatic transformation framework
  - New transformation correction technique
- Scalability on non human-written codes?
- Low interest from users for transformation scripts

Main contributing references: [Vasilache et al., ICS'06], [Bastoul, HDR 2012]

# Outline

1. Introduction
2. The Polyhedral Framework
3. Optimization-Centric Compilation
4. **Looking for Optimizing Transformations**
5. Polyhedral Model Extensions
6. Conclusion & Perspectives

# Compiler Optimizations for Performance

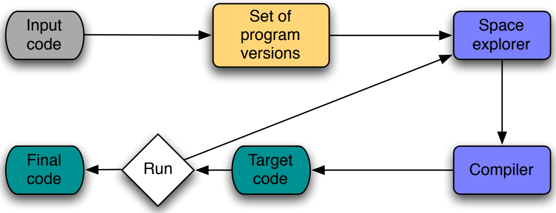
- **High-level transformations are critical...**
  - Coarse-grain parallelism (OpenMP)
  - Fine-grain parallelism (SIMD)
  - Data locality (reduce cache misses)

# Compiler Optimizations for Performance

- **High-level transformations are critical...**
  - Coarse-grain parallelism (OpenMP)
  - Fine-grain parallelism (SIMD)
  - Data locality (reduce cache misses)
- **...But what is the best sequence of transformations?**
  - Conflicting objectives: threads vs SIMD vs locality vs...
  - Architecture, compiler and program-dependent!

# Compiler Optimizations for Performance

- **High-level transformations are critical...**
  - Coarse-grain parallelism (OpenMP)
  - Fine-grain parallelism (SIMD)
  - Data locality (reduce cache misses)
- **...But what is the best sequence of transformations?**
  - Conflicting objectives: threads vs SIMD vs locality vs...
  - Architecture, compiler and program-dependent!
- **Empirical approach: iterative compilation**



# Iterative Compilation Frameworks

- Focus usually on composing existing compiler flags/passes
  - Optimization flags [Bodin et al., PFDC98] [Fursin et al., CGO06]
  - Phase ordering [Kulkarni et al., TACO05]
  - Auto-tuning libraries/platforms (ATLAS, FFTW, SPIRAL...)
- Others attempt to select a transformation sequence
  - Within UTF [Long and Fursin, ICPPW05], GAPS [Nisbet, HPCN98]
  - CHiLL [Chen et al., USCRR08], POET [Yi et al., LCPC07], etc.
  - URUK [Girbal et al., IJPP06]

# Iterative Compilation Frameworks

- Focus usually on composing existing compiler flags/passes
  - Optimization flags [Bodin et al., PFDC98] [Fursin et al., CGO06]
  - Phase ordering [Kulkarni et al., TACO05]
  - Auto-tuning libraries/platforms (ATLAS, FFTW, SPIRAL...)
- Others attempt to select a transformation sequence
  - Within UTF [Long and Fursin, ICPPW05], GAPS [Nisbet, HPCN98]
  - CHiLL [Chen et al., USCRR08], POET [Yi et al., LCPC07], etc.
  - URUK [Girbal et al., IJPP06]

- **Existing high-level approaches**
  - Capability proven for efficient optimization
  - Limited in applicability (legality)
  - Limited in expressiveness (mostly simple sequences)
  - Traversal efficiency compromised (uniqueness)

# Iterative Compilation Frameworks

- Focus usually on composing existing compiler flags/passes
  - Optimization flags [Bodin et al., PFDC98] [Fursin et al., CGO06]
  - Phase ordering [Kulkarni et al., TACO05]
  - Auto-tuning libraries/platforms (ATLAS, FFTW, SPIRAL...)
- Others attempt to select a transformation sequence
  - Within UTF [Long and Fursin, ICPPW05], GAPS [Nisbet, HPCN98]
  - CHiLL [Chen et al., USCRR08], POET [Yi et al., LCPC07], etc.
  - URUK [Girbal et al., IJPP06]

▶ **Our approach**

- ▶ Iterative compilation in the polyhedral model
- ▶ Legality enforced by construction
- ▶ Each optimization is unique and may be very complex
- ▶ Heuristics for fast convergence to near-optimal solutions

# Space of Affine Mappings (1/2)

## Iteration Domain

$$\mathcal{D}_S(N) = \left\{ () \rightarrow \begin{pmatrix} i \\ j \end{pmatrix} \in \mathbb{Z}^2 \;\middle|\; \left[\begin{array}{cc|c|c} 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & -1 \end{array}\right] \begin{pmatrix} i \\ j \\ N \\ 1 \end{pmatrix} \geq \vec{0} \right\}$$

## Mapping Relation

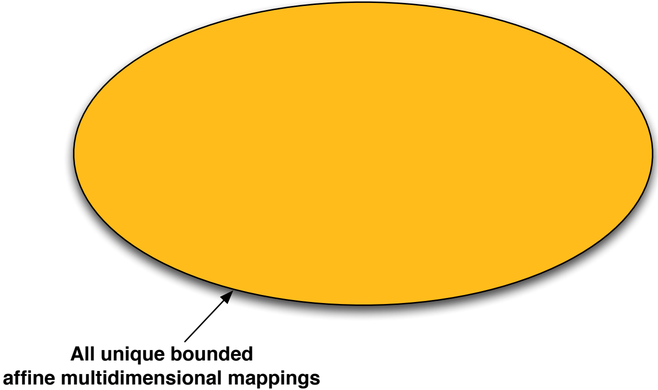
$$\theta_S(N) = \left\{ \begin{pmatrix} i \\ j \end{pmatrix} \rightarrow (\, p \,) \in \mathbb{Z}^2 \times \mathbb{Z}^2 \;\middle|\; \left[\begin{array}{c|cc|c|c} -1 & ? & ? & ? & ? \end{array}\right] \begin{pmatrix} p \\ i \\ j \\ N \\ 1 \end{pmatrix} = \vec{0} \right\}$$

## Generalized Change of Basis

$$\mathcal{T}_S(N) = \left\{ () \rightarrow \begin{pmatrix} p \\ i \\ j \end{pmatrix} \in \mathbb{Z}^4 \;\middle|\; \left[\begin{array}{c|cc|c|c} -1 & ? & ? & ? & ? \\ \hline 0 & 1 & 0 & 0 & -1 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & -1 & 1 & 0 \end{array}\right] \begin{pmatrix} p \\ i \\ j \\ N \\ 1 \end{pmatrix} \begin{array}{c} = \\ \geq \end{array} \vec{0} \right\}$$

- A mapping corresponds to a sequence of transformations
- We look for mapping coefficients
- Bounding them is necessary

# Space of Affine Mappings (2/2)




Radar – Bounded: $10^{200}$

# Semantics-Preserving Mappings (1/2)

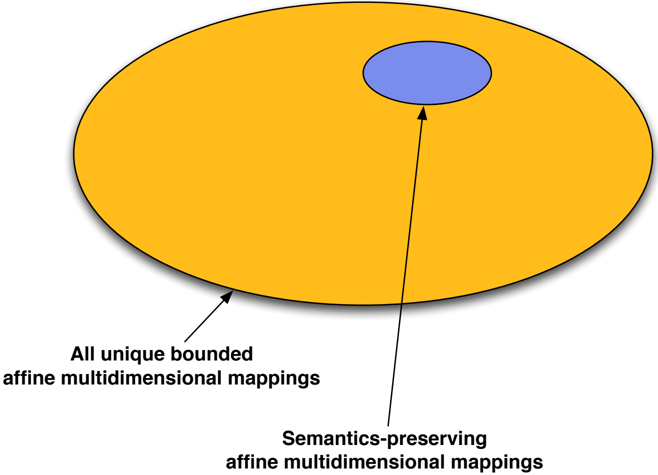
**Definition (Causality Constraint)**

If $\vec{x_R}$ depends on $\vec{x_S}$, their mapping must satisfy:

$$\theta_S(\vec{x_S}) < \theta_R(\vec{x_R})$$

- Simple intuition: if an instance $\vec{x_R}$ depends on another instance $\vec{x_S}$, its target logical date must be at least the one of $\vec{x_S}$ plus 1
- Using the Farkas Lemma, we can write the problem as a set of linear constraints [Feautrier, IJPP'92]
- Dimension per dimension construction of a multidimensional legal space [Pouchet et al., PLDI'08]
- Convex form of a multidimensional legal space [Pouchet et al., PoPL'11]

# Semantics-Preserving Mappings (2/2)




Radar – Bounded: $10^{200}$ Legal: $10^{50}$

# Traversal Heuristics

Extensive performance distribution study conclusions:

- Distribution is not random
- Low proportion of good mappings
- Some coefficients are more impactful
- Large performance deviations shows the way

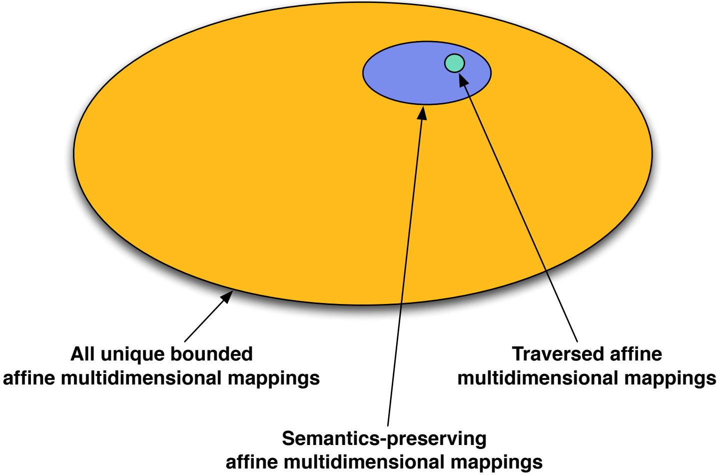
## Decoupling Heuristic

- Traverse iterator coefficients first
- Focus on subspaces where performance variation is high
- Completion algorithm to instantiate the full mapping

## Genetic Algorithm

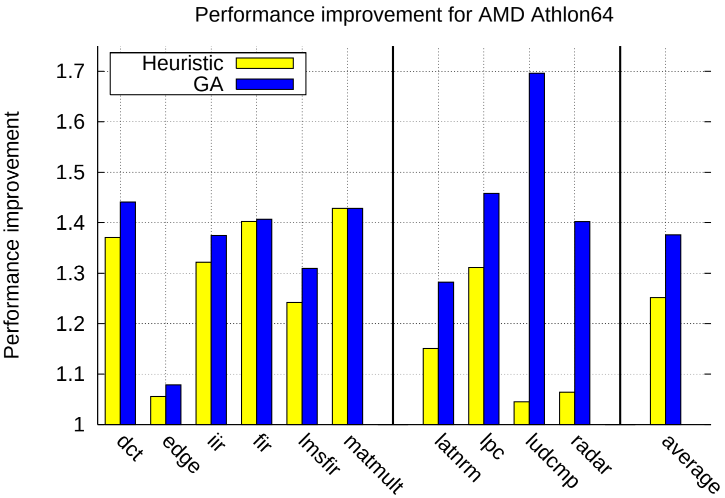
- Special legality-preserving mutation operators
- Well adapted since performance distribution is non uniform
- Tailored to focus on the most promising subspaces

# Semantics-Preserving Mappings (2/2)




Radar – Bounded: $10^{200}$ Legal: $10^{50}$ Empirical search: 50

# Experimental Results




**Baseline: gcc -O3 -ftree-vectorize -msse2**

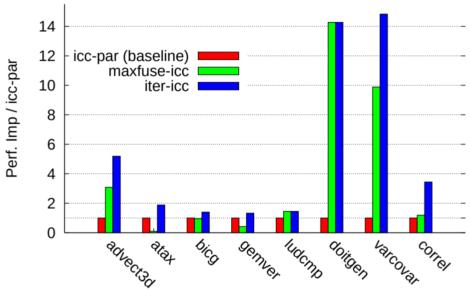
# Coupling Iterative and Model-Based Techniques

Models are not always bad:

- Decide when to tile [Bondhugula et al. PLDI'08]
- Choose the best vectorizable loop [Trifunovic et al. PACT'09]

Iterative techniques for hard-to-model parts: fusion-distribution

Performance Improvement - AMD Opteron 8380 (16 threads)




**Baseline: icc -fast -parallel -openmp**

# Conclusion on Optimizing Transformations

Iterative Compilation in the Polyhedral Model

- Original approach to face compiler and architecture complexity
- Solid theory to encode legality
- Practical solutions for efficient search space traversal
- Strong performance improvement reported
- Possible coupling with existing techniques
- Several runs are necessary
- Sensitivity to dataset size to be considered

Main contributing references: [Pouchet et al., CGO'07, PLDI'08, SC'10, PoPL'11]

# Outline

# Polyhedral Model Constraints

Strict control constraints to be eligible: *static control*

- Affine bounds (`for`)
- Affine conditions (`if`)

*Does it mean that more general codes cannot benefit from a polyhedral compilation framework?*

# Motivating *Transformation*: Loop Fusion

```
// 2strings: count occurences of two words in the same string
nb1 = 0;
for(i=0; i < size_string - size_word1; i++){
   match1 = 0;
   while(word1[match1] == string[i+match1] && match1 <= size_word1)
       match1++;
   if (match1 == size_word1)
       nb1++;
}
nb2 = 0;
for(i=0; i < size_string - size_word2; i++) {
   match2 = 0;
   while(word2[match2] == string[i+match2] && match2 <= size_word2)
       match2++;
   if (match2 == size_word2)
       nb2++;
}
```

- Loop fusion would improve data locality
- Tough by hand
- Trivial transformation if expressed in the polyhedral domain
- But `while` loops and non-static `if` conditions here...

# Extension to `while` Loops

- Extend iteration domain to support predication tags
- (Virtually) Convert `while` loops into infinite `for` loops
- Tag statement iteration domains with *exit predicates*

```
while (condition)
  S();
```

(a) Original Code

```
for (i = 0;; i++) {
  ep = condition;
  if (ep)
    S();
  else
    break;
}
```

(b) Equivalent Code

$$\begin{cases} i \geq 0 \\ (ep = condition) \end{cases}$$

(c) Iteration Domain of S

**Exit-Predicate Extented Iteration Domain**

$$\mathcal{D}_S() = \left\{ () \to (\ i\ ) \in \mathbb{Z} \,\middle|\, ep \in \mathcal{E}_S, [\ 1 \mid 0\ ] \begin{pmatrix} i \\ 1 \end{pmatrix} \geq \vec{0} \wedge ep \right\}$$

# Extension to Non-Static `if` Conditionals

- Extend iteration domain to support predication tags
- Tag statement iteration domains with *control predicates*

```
for (i = 0; i < N; i++)
  if (condition(i))
    S();
```

(a) Original Code

```
for (i = 0; i < N; i++){
  cp = condition(i);
  if (cp)
    S();
}
```

(b) Equivalent Code

$$\begin{cases} i \geq 0 \\ i < N \\ (cp = condition(i)) \end{cases}$$

(c) Iteration Domain of S

## Control-Predicate Extented Iteration Domain

$$\mathcal{D}_S(N) = \left\{ () \to \left( \, i \, \right) \in \mathbb{Z} \,\middle|\, cp(i) \in \mathcal{C}_S, \left[ \begin{array}{c|c|c} 1 & 0 & 0 \\ -1 & 1 & -1 \end{array} \right] \begin{pmatrix} i \\ \hline N \\ \hline 1 \end{pmatrix} \geq \vec{0} \wedge cp(i) \right\}$$

# Revisiting the Polyhedral Framework

- Conservative analysis
  - We consider extra dependences
  - Non-static control is *over-approximated*
  - Non-static references are *over-approximated*
  - Predicate evaluations are considered as plain statements
  - Predicated statements depend on their predicate definitions
- Transformation expressiveness recovery
  - Manipulating unbounded domains is not easy
  - Introduction of an artificial parameter to solve the problem
- Predicate post-processing
  - Usual QRW code generation for predicated domains
  - Exit and control predicates are post-processed
  - Predicate evaluation hoisting and variable privatization

# Experimental Results

State-of-the-art polyhedral optimization techniques applied to (partially) irregular programs

- LeTSeE [Pouchet et al., PLDI'08]
- Pluto [Bondhugula et al., PLDI'08]

| | Speedup regular | | Speedup extended | | Compilation time penalty | |
|---|---|---|---|---|---|---|
| | LetSee | Pluto | LetSee | Pluto | LetSee | Pluto |
| 2strings | N/A | N/A | 1.18× | 1× | N/A | N/A |
| Sat-add | 1× | 1.08× | 1.51× | 1.61× | 1.22× | 1.35× |
| QR | 1.04× | 1.09× | 1.04× | 8.66× | 9.56× | 2.10× |
| ShortPath | N/A | N/A | 1.53× | 5.88× | N/A | N/A |
| TransClos | N/A | N/A | 1.43× | 2.27× | N/A | N/A |
| Givens | 1× | 1× | 1.03× | 7.02× | 21.23× | 15.39× |
| Dither | N/A | N/A | 1× | 5.42× | N/A | N/A |
| Svdvar | 1× | 3.54× | 1× | 3.82× | 1.93× | 1.33× |
| Svbksb | 1× | 1× | 1× | 1.96× | 2× | 1.66× |
| Gauss-J | 1× | 1.46× | 1× | 1.77× | 2.51× | 1.22× |
| PtIncluded | 1× | 1× | 1× | 1.44× | 10.12× | 1.44× |

Setup: Intel Core 2 Quad Q6600

Backend compiler (and baseline): ICC 11.0 `icc -fast -parallel -openmp`

# Conclusion on Polyhedral Model Extensions

**The limitation to static control programs is mostly artificial**

- *Slight and natural extension* to consider irregular codes
  - Infinite loops plus exit and control predication
  - Special parameter to preserve mapping expressiveness
  - Code generation with predicate support
- Benefit from *unmodified* existing techniques for both analysis and optimization

**New extensions should be investigated**

- Minimizing the conservative aspects (inspection and speculation for control dependences)
- Designing optimizations in the context of full functions (algorithmic complexity issues)

Main contributing reference: [Benabderrahmane et al., CC'10]

# Outline

# Conclusion

**The last decade has been *fantastic* for high-level compilation in the polyhedral model!**

- Maturation of the representation to unions of relations
- Maturation of the framework, from theory to practice
- Polyhedral frameworks in most production compilers
- Efficient scheduling techniques (Letsee, Pluto...)
- Growing community, success of the IMPACT workshops

**However many problems are still open or need better solutions**

- Low-level compiler raising may be disappointing
- Complexity is our Sword of Damocles
- Complete static analysis with dynamic or domain-specific knowledge

# Perspectives

- **White-box compiler**
  - Translation of mappings to optimization scripts
  - Non-syntactic optimization directives
  - Human-compiler interface
- **Dynamic optimization**
  - Combine static and dynamic analyses
  - Switcheable versions to adapt to the system load
  - Speculative parallelization
- **Hierarchical optimization**
  - Mimic the strategy of optimization heroes
  - Adapt the optimization technique to each application layer
  - Leverage our experience on iterative, or machine learning, or model-based optimization

# Thanks to

**Co-authors:**
Eduard Ayguadé, Riyadh Baghdadi, Muthu Baskaran, Uday Bondhugula, Paul Carpenter, John Cavazos, Zbigniew Chamski, Albert Cohen, Marco Cornero, Clément Delamare, Philippe Dumont, Marc Duranton, Paul Feautrier, Mohammed Fellahi, Roger Ferrer, Sylvain Girbal, Tobias Großer, Albert Hartono, Pierre Jouvelot, Sriram Krishnamoorthy, Razya Ladelsky, Richard Lethin, Allen Leung, Menno Lindwer, Xavier Martorell, Benoît Meister, Cupertino Miranda, Harm Munk, Boyana Norris, Dorit Nuzman, Rea Ornstein, David Parello, Eunjung Park, Antoniu Pop, Sebastian Pop, J. Ramanujam, Alex Ramírez, Lawrence Rauchwerger, David Ródenas, Erven Rohou, Ira Rosen, P. Sadayappan, Sébastien Salva, Saurabh Sharma, Uzi Shvadron, Marc Sigler, Georges-André Silber, Peter Szilagyi, Olivier Temam, Konrad Trifunović, Sven Verdoolaege, Peter Vouras, David Wohlford, Ayal Zaks

**PhD students:**
Lénaïc Bagnères, Mohamed-Walid Benabderrahmane, Louis-Noël Pouchet, Nicolas Vasilache