Polyhedral Compilation Foundations

Louis-Noël Pouchet

pouchet@cse.ohio-state.edu

Dept. of Computer Science and Engineering, the Ohio State University

Feb 22, 2010

888.11, Class #5

Overview of Today's Lecture

Outline:

- The Space of all (bounded) multidimensional affine schedules
- Overview of the tiling hyperplane method
- Polyhedral compilation toolbox
- State-of-the-art techniques

Mathematical concepts:

- No new concept today!

Reminder From Last Week

- How to select a good one-dimensional schedule
- Greedy algorithm to build a multidimensional schedule
  - Feautrier's: minization of latency

But different objectives are needed:

- multi-core requires coarse-grain parallelism
- Minimal latency is not always the best criterion for performance

Two possible approaches to find a good schedule:

- Proceed in one shot (all coefficients all at a time)
  - Less scalable on large programs
  - But optimal point is in the solution set
- Proceed level-by-level (greedy approach)
  - Usually more scalable
  - But easy to be sub-optimal: no global view of the schedule

Space of All Affine Schedules

Objective:

- Design an ILP which operates on all scheduling coefficients
- Optimality guaranteed since the space contains all schedules (hence necesarily the optimal one)
- Examples: maximal fusion, maximal coarse-grain parallelism, best locality, etc.

idea:

- Combine all coefficients of all rows of the scheduling function into a single solution set
- Find a convex encoding for the lexicopositivity of dependence satisfaction
  - A dependence must be weakly satisfied until it is strongly satisfied
  - Once it is strongly satisfied, it must not constrain subsequent levels

Reminder on Dependence Satisfaction

Definition (Strong dependence satisfaction)

Given $\mathcal{D}_{R,S}$, the dependence is strongly satisfied at schedule level k if

$$\forall \langle \vec{x}_R, \vec{x}_S \rangle \in \mathcal{D}_{R,S}, \quad \Theta_k^S(\vec{x}_S) - \Theta_k^R(\vec{x}_R) \geq 1$$

Definition (Weak dependence satisfaction)

Given $\mathcal{D}_{R,S}$, the dependence is weakly satisfied at dimension k if

$$\begin{array}{ll} \forall \langle \vec{x}_R, \vec{x}_S \rangle \in \mathcal{D}_{R,S}, & \Theta_k^S(\vec{x}_S) - \Theta_k^R(\vec{x}_R) \geq 0 \\ \exists \langle \vec{x}_R, \vec{x}_S \rangle \in \mathcal{D}_{R,S}, & \Theta_k^S(\vec{x}_S) = \Theta_k^R(\vec{x}_R) \end{array}$$

Reminder on Lexicopositivity of Dependence Satisfaction

Lemma (Semantics-preserving affine schedules)

Given a set of affine schedules $\Theta^R, \Theta^S \dots$ of dimension m, the program semantics is preserved if:

$$\begin{array}{rl} & \forall \mathcal{D}_{R,S},\ \exists p \in \{1,\dots,m\},\ \delta_p^{\mathcal{D}_{R,S}} = 1 \\ \wedge & \forall j < p,\ \delta_j^{\mathcal{D}_{R,S}} = 0 \\ \wedge & \forall j \leq p, \forall \langle \vec{x}_R, \vec{x}_S \rangle \in \mathcal{D}_{R,S},\ \Theta_p^S(\vec{x}_S) - \Theta_p^R(\vec{x}_R) \geq \delta_j^{\mathcal{D}_{R,S}} \end{array}$$

Schedule Lower Bound

Idea:

- Bound the schedule latency with a lower bound which does not prevent to find all solutions
- Intuitively:
  - $\Theta^S(\vec{x}_S) - \Theta^R(\vec{x}_R) \geq \delta$ if the dependence has not been strongly satisfied
  - $\Theta^S(\vec{x}_S) - \Theta^R(\vec{x}_R) \geq -\infty$ if it has

Lemma (Schedule lower bound)

Given Θ_k^R, Θ_k^S such that each coefficient value is bounded in $[x,y]$. Then there exists $K \in \mathbb{Z}$ such that:

$$\min\left(\Theta_k^S(\vec{x}_S) - \Theta_k^R(\vec{x}_R)\right) > -K.\vec{n} - K$$

Convex Form of All Bounded Affine Schedules

Lemma (Convex form of semantics-preserving affine schedules)

Given a set of affine schedules $\Theta^R, \Theta^S \ldots$ of dimension m, the program semantics is preserved if the three following conditions hold:

(i) $\forall \mathcal{D}_{R,S},\ \delta_p^{\mathcal{D}_{R,S}} \in \{0,1\}$

(ii) $\forall \mathcal{D}_{R,S},\ \sum_{p=1}^{m} \delta_p^{\mathcal{D}_{R,S}} = 1$

(iii) $\forall \mathcal{D}_{R,S},\ \forall p \in \{1,\ldots,m\},\ \forall \langle \vec{x}_R, \vec{x}_S \rangle \in \mathcal{D}_{R,S},$

$$\Theta_p^S(\vec{x}_S) - \Theta_p^R(\vec{x}_R) \geq -\sum_{k=1}^{p-1} \delta_k^{\mathcal{D}_{R,S}}.(K.\vec{n}+K) + \delta_p^{\mathcal{D}_{R,S}}$$

- $\longrightarrow$ Note: schedule coefficients must be bounded for Lemma to hold
- $\longrightarrow$ Scalability challenge for large programs

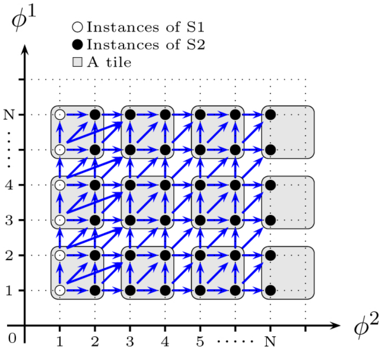
Key Ideas of the Tiling Hyperplane Algorithm

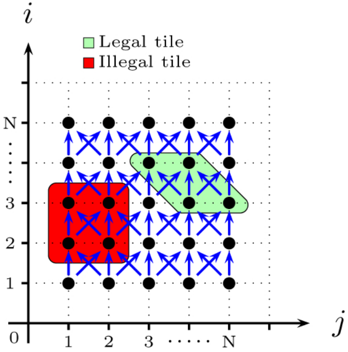
Affine transformations for communication minimal parallelization and locality optimization of arbitrarily nested loop sequences
[Bondhugula et al, CC'08 & PLDI'08]

- Compute a set of transformations to make loops tilable
  - Try to minimize synchronizations
  - Try to maximize locality (maximal fusion)
- Result is a set of *permutable* loops, if possible
  - Strip-mining / tiling can be applied
  - Tiles may be sync-free parallel or pipeline parallel
- Algorithm always terminates (possibly by splitting loops/statements)

Tiling in the Polyhedral Model

- Tiling partition the computation into blocks
- Note we consider only rectangular tiling here
- For tiling to be legal, such a partitioning must be legal





Legality of Tiling

Theorem (Legality of Tiling)

Given ϕ^R, ϕ^S two one-dimensional schedules. They are valid tiling hyperplanes if

$$\forall \mathcal{D}_{R,S},\ \forall \langle \vec{x}_R, \vec{x}_S \rangle \in \mathcal{D}_{R,S},\ \phi^S(\vec{x}_S) - \phi^R(\vec{x}_R) \geq 0$$

- For a schedule to be a legal tiling hyperplane, all communications must go forward: Forward Communication Only [Griebl]
- All dependences must be considered at each level, **including the previously strongly satisfied**
- **Equivalence between loop permutability and loop tilability**

Greedy Algorithm for Tiling Hyperplane Computation

1. Start from the outer-most level, find the set of FCO schedules
2. Select one by minimizing its latency
3. Mark dependences strongly satisfied by this schedule, but do not remove them
4. Formulate the problem for the next level (FCO), adding orthogonality constraints (linear independence)
5. solve again, etc.

<u>Special treatment when no permutable band can be found: splitting</u>

A few properties:

- Result is a set of permutable/tilable outer loops, when possible
- It exhibits coarse-grain parallelism
- Maximal fusion achieved to improve locality

Summary of the Past Lectures

Polyhedral compilation: a 3 stage process

1. Program analysis
   - Extract polyhedral representation of iteration domains: **lecture 1**
   - Extract dependence polyhedra: **lecture 2**
2. Optimization
   - Construct a legal one-dimensional schedule: **lecture 3**
   - Construct a legal multidimensional schedule: **lecture 4-5**
   - Compute a set of candidate legal schedules: **lecture 3-5**
   - Select a good one in this set: **lecture 3-5**
3. Code generation (not treated)
   - "Apply" the transformation in the polyhedral representation
   - Regenerate transformed syntactic code

Polyhedral Software Toolbox

- Analysis:
  - Extracting the polyhedral representation of a program: Clan, PolyRose
  - Computing the dependence polyhedra: Candl
- Mathematical operations:
  - Doing polyhedral operations on $\mathbb{Q}$-, $\mathbb{Z}$- and $\mathcal{Z}$-polyhedral: PolyLib, ISL
  - Solving ILP/PIP problems: PIPLib
  - Computing the number of points in a (parametric) polyhedron: Barvinok
  - Projection on $\mathbb{Q}$-polyhedra: FM, the Fourier-Motzkin Library
- Scheduling:
  - Tiling hyperplane method: PLuTo
  - Iterative selection of affine schedules: LetSee
- Code generation:
  - Generating C code from a polyhedral representation: CLooG
  - Parametric tiling from a polyhedral representation: PrimeTile

Polyhedral Compilers

Available polyhedral compilers:

- Non-Free:
  - IBM XL/Poly
  - Reservoir Labs RStream
- Free:
  - GCC (see the GRAPHITE effort)
- Prototypes:
  - **PoCC, the POlyhedral Compiler Collection**
    http://pocc.sourceforge.net
    Contains Clan, Candl, Pluto, LetSee, PIPLib, PolyLib, FM, ISL, Barvinok, CLooG, ...
  - **PolyRose** from OSU (DARPA PACE project), a polyhedral compiler using parts of PoCC and the Rose infrastructure

Polyhedral Methodology Toolbox

- Semantics-preserving schedules:
  - Dependence relation finely characterized with dependence polyhedra
  - Algorithms should harness the power of this representation (ex: legality testing, parallelism testing, etc.)
- Scheduling:
  - Scheduling algorithm can be greedy (level-by-level) or global
  - Beware of scalability
  - Special properties can be embedded in the schedule via an ILP (ex: fusion, tiling, parallelism)
- Mathematics:
  - Beware of the distinction between $\mathbb{Q}$-, $\mathbb{Z}$- and $\mathcal{Z}$-polyhedra: always choose the most relaxed one that fits the problem
  - Farkas Lemma is useful to characterize a solution set
  - Farkas Lemma is also useful to linearize constraints

State-of-the-art in Polyhedral Compilation

- Analysis
    - Array Dataflow Analysis [Feautrier,IJPP91]
    - Dependence polyhedra [Feautrier,IJPP91] (Candl)
    - Non-static control flow support [Benabderrahmane,CC10]
- Program transformations:
    - Tiling hyperplane method [Bondhugula,CC08/PLDI08]
    - Convex space of all affine schedules [Vasilache,07]
    - Iterative search [Pouchet,CGO07/PLDI08]
    - Vectorization [Trifunovic,PACT09]
- Code generation
    - Arbitrary affine scheduling functions [Bastoul,PACT04]
    - Scalable code generation [Vasilache,CC06/PhD07]
    - Parametric Tiling [Hartono et al,ICS09/CGO10]

Some Ongoing Research

- Scalability: provide more scalable algorithms, operating on hundreds of statements
  - Trade-off between optimality and scalability
  - Redesigning the framework: introducing approximations
- Vectorization: pre- and post- transformations for vectorization
  - Select the appropriate transformations for vectorization
  - Generate efficient SIMD code
- Scheduling: get (very) good performance on a wide variety of machines
  - Using machine learning to characterize the machine/compiler/program
  - Using more complex scheduling heuristics

Potential MS projects [1/2]

I am currently looking for students to work on the following topics:

- Cost models for vectorization
  - Used to select which loop should be vectorized
  - Current models must be generalized and improved
  - Interaction with other optimization objectives (tiling)
- Cost models for array contraction
  - Used to select which dimension of an array should be reduced to a scalar
  - Huge interaction with tiling, parallelism and vectorization

Potential MS projects [2/2]

Related to Machine Learning:

- Computing the similarities between two polyhedral programs
  - Used for machine learning techniques
  - The goal is to have a systematic way to characterize two programs that may need the same transformation, leveraging the polyhedral representation
- Evaluation of machine learning for the selection of affine schedules
  - Hot topic! ask me if you're interested

We have many other projects not in this list! Ask Prof. Sadayappan

Credits

Several figures and examples in this series of lectures were borrowed from:

- Uday Bondhugula's PhD thesis
- Cedric Bastoul's PhD thesis
- Wikipedia

They are gratefully acknowledged for their support!