The Polyhedral Model (Dependences and Transformations)

Announcements
– HW3 is due Wednesday February 15th, tomorrow
– Project proposal is due NEXT Friday, extension so that example with tool is possible

Today
– CUDA programming, key concepts for HW3
– Automatic parallelization
– Transformation models/frameworks
– Polyhedral model
  – Iteration space representation
  – Data dependence problem and representation
  – Transformation representation and legality check

CUDA programming

Some key concepts
– Kernel call parameters, slide 37 in http://www.nvidia.com/content/cudazone/download/Getting_Started_w_CUDA_Training_NVISION08.pdf
– The kernel call is asynchronous
  – Have to call cudaThreadSynchronize() before calling cutStopTimer() or gettimeofday()
**Automatic Parallelization**

Input program has a set of operations $E$ with a strict order

Find a partial order on $E$ that is deterministic and results in the same output as the original strict total order.

**Overall process**
- Translate the code to a model
- Select a transformation/schedule
  - Determination of partial order on $E$, data dependence analysis
  - Ensure that the loop transformation/schedule is legal
- Transform the model and generate the transformed code

---

**What should the model be?**

**Bernstein conditions**
- Let $u$ and $v$ be operations, $M(u)/M(v)$ be the set memory locations written by $u/v$, $R(u)/R(v)$ be the set of memory locations read by $u/v$.
- If $u$ precedes $v$ and the intersection of $M(u)$ and $R(v)$ is non-empty, then there is a **flow** dependence.
- If $u$ precedes $v$ and the intersection of $R(u)$ and $M(v)$ is non-empty, then there is an **anti** dependence.
- If $u$ precedes $v$ and the intersection of $M(u)$ and $M(v)$ is non-empty, then there is an **output** dependence.

**Problematic Example**

$$M(u) = \{a^n + b^n | n > 2 \land a, b, n \in \mathbb{Z}\}$$

$$R(v) = \{c^n | n > 2 \land c, n \in \mathbb{Z}\}$$
Dependence Testing in General

General code

do  i_1 = l_1, h_1
...  
do  i_n = l_n, h_n
...  A(f(i_1, ..., i_n))
...  A(g(i_1, ..., i_n))
enddo
... 
enddo

There exists a dependence between iterations I=(i_1, ..., i_n) and J=(j_1, ..., j_n) when at least one of the accesses is a write and

- f(I) = g(J)
- (l_1, ..., l_n) < I, J < (h_1, ..., h_n)
- I << J or J << I, where << is lexicographically less

Polyhedron
(source: http://www.cse.ohio-state.edu/~pouchet/lectures/888.11.lect1.html)

Affine functions
- A function \( f : \mathbb{K}^m \rightarrow \mathbb{K}^n \) is affine if there exists a vector \( \vec{b} \in \mathbb{K}^n \) and a matrix \( A \in \mathbb{K}^{n \times m} \) such that \( \forall \vec{x} \in \mathbb{K}^m, f(\vec{x}) = A\vec{x} + \vec{b} \)

Affine half spaces
- An affine half-space of \( \mathbb{K}^m \) (affine constraint) is defined as a set of points \( \{ \vec{x} \in \mathbb{K}^m | \vec{a} \cdot \vec{x} \leq \vec{b} \} \)

Polyhedron
- A set \( S \subseteq \mathbb{K}^m \) is a polyhedron if there exists a system of finite inequalities \( A\vec{x} \leq \vec{b} \) such that \( P = \{ \vec{x} \in \mathbb{K}^m | A\vec{x} \leq \vec{b} \} \)
- Equivalently it is the intersection of finitely many half-spaces.

Intersection between polyhedral sets
- When you intersect two polyhedral sets the results is a polyhedral set.
- Many questions we need to automate check whether a polyhedral set or sets are empty or not.
  - Is there a dependence at a certain loop level?
  - Is a transformation legal?
Dependence Testing in General

General code

\[
\begin{align*}
\text{do } i_1 &= l_1, h_1 \\
... \\
\text{do } i_n &= l_n, h_n \\
... & A(f(i_1, ..., i_n)) \\
... & A(g(i_1, ..., i_n)) \\
\text{enddo} \\
... \\
\text{enddo}
\end{align*}
\]

There exists a dependence between iterations \( I=(i_1, ..., i_n) \) and \( J=(j_1, ..., j_n) \) when at least one of the accesses is a write and

- \( f(I) = g(J) \)
- \( (l_1, ..., l_n) < I, J < (h_1, ..., h_n) \)
- \( I \ll J \) or \( J \ll I \), where \( \ll \) is lexicographically less

Algorithms for Solving the Dependence Problem

Heuristics can say NO or MAYBE

- GCD test (Banerjee76,Towle76): determines whether integer solution is possible, no bounds checking
- Banerjee test (Banerjee 79): checks real bounds
- I-Test (Kong et al. 90): integer solution in real bounds
- Lambda test (Li et al. 90): all dimensions simultaneously
- Delta test (Goff et al. 91): pattern matches for efficiency
- Power test (Wolfe et al. 92): extended GCD and Fourier-Motzkin combination

Exact solutions, exponential worst-case since integer linear programming is NP-complete

- Parametric Integer Programming (Feautrier91), based on the Simplex algorithm
- Omega test (Pugh92), based on the Fourier-Motzkin elimination algorithm
**Dependence Testing**

Consider the following code...

```verbatim
do i = 1,5  
   A(3*i+2) = A(2*i+1)+1  
enddo
```

**Question**
- How do we determine whether one array reference depends on another across iterations of an iteration space?

---

**Dependence Testing: Simple Case**

**Sample code**

```verbatim
do i = l,h  
   A(a*i+c_1) = ... A(a*i+c_2)  
enddo
```

**Dependence?**
- \( a*i_1+c_1 = a*i_2+c_2 \), or
- \( a*i_1 - a*i_2 = c_2-c_1 \)
- Solution may exist if \( a \) divides \( c_2-c_1 \)
GCD Test

Idea
– Generalize test to linear functions of iterators/induction variables

Code

do i = l_i, h_i
  do j = l_j, h_j
    A(a_1*i + a_2*j + a_0) = ... A(b_1*i + b_2*j + b_0) ...
  enddo
enddo

Again
– a_1*i_1 - b_1*i_2 + a_2*j_1 - b_2*j_2 = b_0 - a_0
– Solution exists if gcd(a_1, a_2, b_1, b_2) divides b_0 - a_0

Example

Code

do i = l_i, h_i
  do j = l_j, h_j
    A(4*i + 2*j + 1) = ... A(6*i + 2*j + 4) ...
  enddo
enddo

gcd(4, -6, 2, -2) = 2

Does 2 divide 4-1?
**Banerjee Test**

```c
for (i=L; i<=U; i++) {
    x[a0 + a1*i] = ...
    ... = x[b0 + b1*i]
}
```

Does \(a_0 + a_1i = b_0 + b_1i'\) for some real \(i\) and \(i'\)?

If so then \((a_1i - b_1i') = (b_0 - a_0)\)

Determine upper and lower bounds on \((a_1i - b_1i')\)

```c
for (i=1; i<=5; i++) {
    x[i+5] = x[i];
}
```

upper bound = \(a_1\max(i) - b_1 \min(i') = 4\)
lower bound = \(a_1\min(i) - b_1\max(i') = -4\)

\(b_0 - a_0 = \)

---

**Next Time**

**Reading**
- No reading assignment this week, should be reading about CUDA

**Homework**
- HW3 is due Wednesday 2/15/12, tomorrow
- Project proposal pushed to next Friday, 2/24/12

**Lecture**
- Basics of code generation