Lecture 3: Polyhedral Model

CR11 – Hardware Compilation and Simulation
ENS-Lyon – M2IF

Christophe Alias

Polyhedral Model

\[
\begin{aligned}
&\text{for } i := 0 \text{ to } 2N \\
&\text{S: } c[i] := 0; \\
&\text{for } i := 0 \text{ to } N \\
&\text{for } j := 0 \text{ to } N \\
&\text{T: } c[i+j] := c[i+j] + a[i]*b[j];
\end{aligned}
\]

- **Control**: Loop kernels with for loops and tests if
- **Data**: arrays and scalar variables.
- **Affine** loop bounds, conditions and array indices.
- Iteration domains are integer polyhedra
  \[
  D_T := \{ i \in \mathbb{Z} | 0 \leq i \leq 2N \} \\
  D_S := \{ (i, j) \in \mathbb{Z}^2 | 0 \leq i, j \leq N \}
  \]
- **Tools**: ILP, geometric operations, etc

Quizz

\[
\begin{aligned}
&\text{for } i := 0 \text{ to } 2N \\
&\text{S: } c[i] := 0; \\
&\text{for } i := 0 \text{ to } N \\
&\text{for } j := 0 \text{ to } N \\
&\text{T: } c[i+j] := c[i+j] + a[i]*b[j];
\end{aligned}
\]

- **Locally**: lexicographic order
  \[
  (T, i, j) < (T', i', j') \iff (i, j) \ll (i', j')
  \]
- **Globally**, we consider the common iterations
- **Tools**: min\_\_<P / max\_\_<P

Find the first instance of \( T \) which reads \( c[2] \)
Data Dependences

\[\text{for } i := 0 \text{ to } 2N S: \quad c[i] := 0;\]
\[\text{for } i := 0 \text{ to } N\]
\[\text{for } j := 0 \text{ to } N\]
\[T: \quad c[i+j] := c[i+j] + a[i]*b[j];\]

- \(x \rightarrow y\) iff \(x \prec y\) and both access the same data slot:
  - \(\rightarrow_{\text{flow}} y\): \(x\) writes; \(y\) reads
  - \(\rightarrow_{\text{anti}} y\): \(x\) reads; \(y\) writes
  - \(\rightarrow_{\text{output}} y\): \(x\) writes; \(y\) writes

- \(\rightarrow_{\text{flow}}\) captures the computation
  \(\rightarrow_{\text{anti}}\) and \(\rightarrow_{\text{output}}\) can be eliminated.
- **Tool:** affine relation \(\{i \rightarrow j \mid (i,j) \models \Phi\}\)

Quizz

\[\text{for } i := 0 \text{ to } 2N S: \quad c[i] := 0;\]
\[\text{for } i := 0 \text{ to } N\]
\[\text{for } j := 0 \text{ to } N\]
\[T: \quad c[i+j] := c[i+j] + a[i]*b[j];\]

Find \(\rightarrow_{\text{flow}}\) between instances of \(T\)

Direct Dependences

\[\text{for } i := 0 \text{ to } 2N S: \quad c[i] := 0;\]
\[\text{for } i := 0 \text{ to } N\]
\[\text{for } j := 0 \text{ to } N\]
\[T: \quad c[i+j] := c[i+j] + a[i]*b[j];\]

Flow dependences **producer/consumer**

\(\rightarrow_{pc}\) For each read \(y\), we keep the last write \(x_{\text{max}}\) defining \(y\):

\(\rightarrow_{pc} = \{x_{\text{max}} \rightarrow_{\text{flow}} y \mid x_{\text{max}} = \max_{\prec} \{x \mid x \rightarrow_{\text{flow}} y\}\}\)

\(\rightarrow_{pc}\) Smallest \(\rightarrow_{pc} \subseteq \rightarrow_{\text{flow}}\) whose \(\rightarrow_{\text{flow}}\) is the transitive closure

Perfect Loop Nests

\[\text{for } i := 1 \text{ to } N\]
\[\text{for } j := 1 \text{ to } N\]
\[B: \quad a[i,j] := a[i-1,j-1] + a[i-1,j];\]

- **Single iteration domain** \(D_B\)
- **Dependence vectors:** \(\Delta D = \{\overline{j} - \overline{i} \mid (B, \overline{i}) \rightarrow_{pc} (B, \overline{j})\}\)
- **Uniform dependences:** \(\Delta D\) is finite
- **Dependence cone:** \(C_D = \text{cone}(\Delta D)\)
- **Tools:** dual \(C^* = \{\overline{x} \mid \overline{x} \cdot \overline{y} \geq 0 \forall \overline{y} \in C\}\)
Quizz

for $i := 1$ to $N$
for $j := 1$ to $N$

B: $a[j] := a[j-1] + a[j] + a[j+1];$

Find $\Delta D$ and the dependence cone $C_D$

Schedules

for $i := 0$ to $2N$
S: $c[i] := 0;$

for $i := 0$ to $N$
for $j := 0$ to $N$
T: $c[i+j] := c[i+j] + a[i]b[j];$

$\theta_S(i) = 0, \theta_T(i,j) = i + 1$

- A schedule assigns an execution date $\vec{t} \in (\mathbb{Z}^d, \ll)$ to each execution instance $\langle R, \vec{x} \rangle$.
- Affine (per statement) schedule: $\theta_R(\vec{x}) = A_R \vec{x} + \vec{b}_R$
- Correctness: $\langle S, \vec{x} \rangle \rightarrow \langle T, \vec{y} \rangle \Rightarrow \theta_S(\vec{x}) \ll \theta_T(\vec{y})$

Quizz

Expressivity: loop permutation

for $i := 0$ to $2N$
S: $c[i] := 0;$

for $i := 0$ to $N$
for $j := 0$ to $N$
T: $c[i+j] := c[i+j] + a[i]b[j];$

Assuming $\theta_S(i) = i$, find a correct $\theta_T$

Quizz: prove the correctness of $\theta_B$
Expressivity: loop skewing

for \( i := 1 \) to \( N \)
for \( j := 1 \) to \( N \)
B: \( a[i,j] := a[i-1,j] + a[i,j-1] \);

\( \theta_B(i,j) = U(i,j) \) e.g. \((i+j,j)\)

Expressivity: loop fusion

for \( i := 1 \) to \( N \)
for \( j := 1 \) to \( N \)
S: \( c[j] := a[i] + b[j] \);
for \( j := 1 \) to \( N \)
T: \( d[i] := d[i] + c[j] \);

\( \theta_S(i, j) = (i, 2j) \)
\( \theta_T(i, j) = (i, 2j + 1) \)
\( \theta_S(i, j) = (i, j, 0) \)
\( \theta_T(i, j) = (i, j, 1) \)

Expressivity: loop fission

for \( i := 1 \) to \( N \)
for \( j := 1 \) to \( N \)
S: \( c[j] := a[i] + b[j] \);
for \( j := 1 \) to \( N \)
T: \( d[i] := d[i] + c[j] \);

\( \theta_S(i, j) = (i, j) \)
\( \theta_T(i, j) = (i, j + N) \)
\( \theta_S(i, j) = (i, 0, j) \)
\( \theta_T(i, j) = (i, 1, j) \)

Expressivity: loop shifting

for \( i := 1 \) to \( N \)
for \( j := 1 \) to \( N \)
S: \( b[i] := a[i] \);
for \( i := 1 \) to \( N \)
if \( i \geq 2 \) then
S\(_1\): \( b[i-1] := a[i-1] \);
T: \( c[i] := b[i-1] \);
S\(_2\): \( b[N] := a[N] \);

\( \theta_S(i) = S(i) \)
\( \theta_T(i) = T(i) \)

Draw the resulting iteration domains. Give \( \theta_S \) and \( \theta_T \).
Expressivity: loop pipelining

\[
\begin{align*}
\text{for } & i := 1 \text{ to } N; \\
S: & \text{ read}[i] := x[i]; \\
T: & \text{ result}[i] := 5*\text{read}[i]+3; \\
U: & \text{ y}[i] := \text{result}[i];
\end{align*}
\]

- \(\theta_S(i) = i\)
- \(\theta_T(i) = i + 1\)
- \(\theta_U(i) = i + 2\)
- In general: \(\theta_S(i) = \lambda i + \delta_S\)

Lessons

- Affine schedules express any composition of:
  - skewing, permutation, fusion/fission, shifting
  - and parallelism!
- Perfect loop nest \(\leftrightarrow\) Linear schedule
  \(\theta_B(\vec{i}) = A\vec{i}\)
- Non-perfect loop nest \(\leftrightarrow\) Affine schedule
  \(\theta_S(\vec{i}) = A_S\vec{i} + b_S\)

(Naive) polyhedral compilation flow

... and composition thereof

\[
\begin{align*}
\text{// } y & := Ax; z := By; \\
\text{for } & i := 1 \text{ to } N \\
& \text{for } j := 1 \text{ to } N \\
S: & \text{ y}[i] += A[i,j]*x[j]; \\
T: & \text{ z}[i] += B[i,j]*y[j];
\end{align*}
\]

Give \(\theta_S\) and \(\theta_T\) minimizing the latency
Lab

1. Get iscc.tgz and code-template.cc from the course webpage.

2. Enter the script skewing.iscc:
   Domain := [N] -> {S[i,j]: 1 <= i, j <= N};
   Schedule := {S[i,j] -> [i+j, j]};
   Scheduled_domain := Schedule * Domain;
   codegen Scheduled_domain;

3. Evaluate with ./iscc < skewing.iscc

4. Modify the code to print the iterations (with cout << ...).

5. Try with the composition of matrix-vector product (slide 19).