Lecture 4: Linear Scheduling

CR11 – Hardware Compilation and Simulation
ENS-Lyon – M2IF

Christophe Alias
Linear Schedules

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

Scope: perfect loop nests with uniform dependences

\(\theta : D_B \rightarrow (\mathbb{Z}^d, \ll) \quad \vec{i} \mapsto A\vec{i}\)
for $i := 1$ to $N$
  for $j := 1$ to $N$
    $B$: $a[j] := a[j] + 1$;

Draw the execution order for:

- $\theta_1(i, j) = i$
- $\theta_2(i, j) = i + j$
- $\theta_3(i, j) = (i + j, i)$
Latency

for $i := 1$ to $N$
for $j := 1$ to $N$
B: $a[j] := a[j] + 1;$

- Latency: “Number of steps”
- $\lambda = \text{card}\{\theta(\vec{i}), \vec{i} \in D_B\}$
for \( i := 1 \) to \( N \)
for \( j := 1 \) to \( N \)
B: \( a[j] := a[j] + 1; \)

Give the latency for:
- \( \theta_1(i, j) = i \)
- \( \theta_2(i, j) = i + j \)
- \( \theta_3(i, j) = (i + j, i) \)

Theorem 1

*If \( d = 1 \), then the latency is a affine form of program parameters*
for $i := 1$ to $N$
for $j := 1$ to $N$
B: $a[i,j] := a[i-1,j] + a[i,j-1]$;

\[ \theta(\vec{i}) = A\vec{i} = \vec{\tau} \cdot \vec{i} \]

Correctness: $\vec{\tau} \cdot \vec{d} > 0 \quad \forall \vec{d} \in \Delta D$
for $i := 1$ to $N$
for $j := 1$ to $N$
B: $a[j] := a[j] + 1$;

Characterize all the correct linear schedules
\begin{align*}
\text{for } i & := 1 \text{ to } N \\
\text{for } j & := 1 \text{ to } N \\
B: & \quad s := s + a[i,j];
\end{align*}

Prove that no monodimensional linear schedule exists
Multidimensional case ($d \geq 2$)

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

\[
\theta(\vec{i}) = \begin{pmatrix}
\vec{\tau}_1 \\
\vdots \\
\vec{\tau}_d
\end{pmatrix} \vec{i}
\]
for $i := 1$ to $N$
    for $j := 1$ to $N$
        $B$: $s := s + a[i,j]$;

Give $\vec{\tau}_1$ and $\vec{\tau}_2$ for $\theta(i, j) = (i, j)$
for $i := 1$ to $N$
for $j := 1$ to $N$
\[ B: \quad s := s + a[i,j]; \]

Give $\vec{\tau}_1$ and $\vec{\tau}_2$ for $\theta(i,j) = (i,j)$

<table>
<thead>
<tr>
<th></th>
<th>$\vec{d}_1$</th>
<th>$\vec{d}_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\vec{\tau}_1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\vec{\tau}_2$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Finding $\vec{\tau}_1$

for $i := 1$ to $N$    
for $j := 1$ to $N$    
B: $s := s + a[i,j]$;

$\forall \vec{d} \in \Delta D \enspace \vec{\tau} \cdot \vec{d} > 0$ or $\vec{\tau} \cdot \vec{d} = 0$
for $i := 1$ to $N$
for $j := 1$ to $N$

B: \[ s := s + a[i,j]; \]

Give $C_D$ and $C_T$
Finding $\vec{\tau}_2$

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

- Focus on respected dependences $\Delta D'$
- Pick $\tau_2 \in \text{cone}\langle \Delta D'\rangle^*$
Finding $\vec{\tau}_2$

\begin{verbatim}
for $i := 1$ to $N$
    for $j := 1$ to $N$
        B: $s := s + a[i,j]$;

$\Delta D' = \{(0, 1)\}$
\end{verbatim}

Focus on respected dependences $\Delta D'$

Pick $\tau_2 \in \text{cone}(\Delta D')^*$
Algorithm

for $i := 1$ to $N$
  for $j := 1$ to $N$
    $B$: $s := s + a[i,j]$;

$i := 1$
while $\Delta D \neq \emptyset$
  $C_T := \text{cone}(\Delta D)^*$
  Pick $\vec{\tau}_i \in C_T \setminus \{0\}$
  Remove from $\Delta D$ the satisfied dependences ($\vec{\tau}_i \cdot \vec{d} > 0$)
  $i := i + 1$
Algorithm

for $i := 1$ to $N$
for $j := 1$ to $N$
B: $s := s + a[i,j]$;

$i := 1$
while $\Delta D \neq \emptyset$
    $C_T := \text{cone} \langle \Delta D \rangle^*$
    Pick $\vec{\tau}_i \in C_T \setminus \{0\}$
    Remove from $\Delta D$ the satisfied dependences $(\vec{\tau}_i \cdot \vec{d} > 0)$
    $i := i + 1$
for $i := 1$ to $N$
    for $j := 1$ to $N$
        $B$: $s := s + a[i,j]$;

$i := 1$

while $\Delta D \neq \emptyset$
    $C_T := \text{cone}(\Delta D)^*$
    Pick $\vec{\tau}_i \in C_T \setminus \{0\}$
    Remove from $\Delta D$ the satisfied dependences ($\vec{\tau}_i \cdot \vec{d} > 0$)
    $i := i + 1$
Algorithm

for $i := 1$ to $N$
for $j := 1$ to $N$
\[ B: \quad s := s + a[i,j]; \]

$i := 1$
while $\Delta D \neq \emptyset$
\[ C_T := \text{cone}(\Delta D)^* \]
Pick $\vec{\tau}_i \in C_T \setminus \{0\}$
Remove from $\Delta D$ the satisfied dependences ($\vec{\tau}_i \cdot \vec{d} > 0$)
$i := i + 1$
Algorithm

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

\[
i := 1 \\
\text{while } \Delta D \neq \emptyset \\
\quad C_T := \text{cone}(\Delta D)^* \\
\quad \text{Pick } \vec{\tau}_i \in C_T \setminus \{0\} \\
\quad \text{Remove from } \Delta D \text{ the satisfied dependences } (\vec{\tau}_i \cdot \vec{d} > 0) \\
i := i + 1
\]
Lemma 2

\( \theta \) is correct over \( \Delta D \) iff \( \theta(\vec{d}) \gg \vec{0} \ \forall \vec{d} \in \Delta D \)

Quizz: Prove it

Theorem 3

The algorithm finds a correct schedule

Proof.

Consider \( \vec{d} \in \Delta D \). If the algorithm terminates,

\[ \exists k : \left( \vec{\tau}_k \text{ satisfies } \vec{d} \right) \land \left( \vec{\tau}_\ell \text{ respects } \vec{d} \ \forall \ell < k \right) \]

Hence, \( \theta(\vec{d}) \gg \vec{0} \).
How to “pick” $\vec{\tau}_i$?

**Greedy:** satisfy as much dependences as possible
How to “pick” $\vec{\tau}_i$?

\begin{align*}
\text{for } i &:= 1 \text{ to } N \\
&\text{Greedy: satisfy as much dependences as possible}
\end{align*}
How to “pick” $\vec{\tau}_i$?

for $i := 1$ to $N$

$B$

$N = 4$

$\vec{\tau}_1$

Greedy: satisfy as much dependences as possible
How to “pick” \( \tau_i \)?

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

Greedy: satisfy as much dependences as possible
for $i := 1$ to $N$
for $j := 1$ to $N$
B: $a[j] := a[j] + 1$;

Greedy: satisfy as much dependences as possible
How to “pick” $\vec{τ}_i$?

Lazy: satisfy as less dependences as possible
How to “pick” $\vec{\tau}_i$?

forpar $j := 1$ to $N$

Lazy: satisfy as less dependences as possible
How to “pick” $\vec{\tau}_i$?

```plaintext
forpar j := 1 to N
   for i := 1 to N
     B: a[j] := a[j] + 1;
```

Lazy: satisfy as less dependences as possible
How to “pick” $\vec{r}_i$?

\begin{verbatim}
forpar j := 1 to N
  for i := 1 to N
    B: a[j] := a[j] + 1;
\end{verbatim}

Lazy: satisfy as less dependences as possible
How to “pick” $\vec{τ}_i$?

```
for par j := 1 to N
  for i := 1 to N
    B: a[j] := a[j] + 1;
```

Lazy: satisfy as less dependences as possible
Lessons

- **Greedy approach**: minimize the dimension (sequentiality degree)
  → maximize the parallelism degree

- **Lazy approach**: push dependence resolution into inner loops
  → improve the data locality

- **The lazy approach is the best**
  → parallel outer loops + inner loops with good locality
Consider the Seidel 1D kernel:

\[
\text{for } t := 1 \text{ to } T \\
\text{for } i := 1 \text{ to } N \\
a_i = a_{i-1} + a_i + a_{i+1}
\]

1. Give $\Delta D$, the dependence cone and the time cone.
2. Apply the linear scheduling algorithm with the greedy approach.
3. Draw the iteration domain (for small values of $T$ and $N$) and the obtained execution order.