Class Information

• Homework 8 is released.
• One extra-credit project on Scheme has been released.
• Extended TA office hours this week:
  Qiaoying:  5/1 Tuesday 7pm-9pm, HW 1, HW4, HW6, HW8.
  Qin         :  5/3 Thursday 2pm - 4pm, for HW2 and HW7.
  Ari          :  5/7 Monday 5:45pm-7:45pm, for HW 3 and HW5.
             (Ari’s office is CoRE 334).
• My office hour before exam:
  5/5 Saturday 3pm — 5pm.
• All questions on Sakai before 5/8 Tuesday 10pm are guaranteed to be answered, within 3 days or before 5/8 midnight, whichever is sooner.
Given

\[
\begin{align*}
\text{do } & i_1 = L_1, U_1 \\
& \quad \ldots \\
\text{do } & i_n = L_n, U_n \\
& S_1 : \quad A[ f_1(i_1, \ldots, i_n), \ldots, f_m(i_1, \ldots, i_n) ] = \ldots \\
& S_2 : \quad \ldots = A[ g_1(i_1, \ldots, i_n), \ldots, g_m(i_1, \ldots, i_n) ]
\end{align*}
\]

Let \( \alpha & \beta \) be a vector of \( n \) integers within the ranges of the lower and upper bounds of the \( n \) loops.

Does \( \exists \alpha, \beta \) in the loop iteration space, s.t.

\[
f_k(\alpha) = g_k(\beta) \quad \forall k, 1 \leq k \leq m?
\]
Does $\exists \alpha, \beta$ in the loop iteration space, s.t.
\[
 f_k(\alpha) = g_k(\beta) \quad \forall k, 1 \leq k \leq m?
\]

Consider the two memory references:

$S1(\alpha): X[i_1, j_1], S2(\beta): X[i_2, j_2-1]$

$\alpha$: $(i_1, j_1)$
$\beta$: $(i_2, j_2)$

Access the same memory location

Does there exist a solution to this integer linear programming (ILP) problem?

Loop bounds constraint

- $i_1 = i_2$
- $j_1 = j_2 - 1$
- $1 \leq i_1 \leq 100$
- $1 \leq j_1 \leq 100$
- $1 \leq i_2 \leq 100$
- $1 \leq j_2 \leq 100$
If we use the matrix vector notation \(<F_1, f_1, B_1, b_1>\) and \(<F_2, f_2, B_2, b_2>\) for two references at two iterations \(\alpha\): \((i_1, j_1)\) and \(\beta\): \((i_2, j_2)\)

\[
\begin{align*}
  i_1 &= i_2 \\
  j_1 &= j_2 - 1 \\
  1 &\leq i_1 \leq 100 \\
  1 &\leq j_1 \leq 100 \\
  1 &\leq i_2 \leq 100 \\
  1 &\leq j_2 \leq 100
\end{align*}
\]

\[
F_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad f_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}
\]

\[
F_1 \begin{bmatrix} i_1 \\ j_1 \end{bmatrix} + f_1 = F_2 \begin{bmatrix} i_2 \\ j_2 \end{bmatrix} + f_2
\]

\[
\begin{bmatrix} i_1 \\ j_1 \end{bmatrix} = \begin{bmatrix} i_2 \\ j_2 - 1 \end{bmatrix}
\]

Memory reference \(X[i_1, j_1]\)

\[
\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} i_1 \\ j_1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} i_1 \\ j_1 \end{bmatrix}
\]

Memory reference \(X[i_2, j_2-1]\)

\[
F_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad f_2 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}
\]

\[
F_2 \begin{bmatrix} i_2 \\ j_2 \end{bmatrix} + f_2 = \begin{bmatrix} i_2 \\ j_2 - 1 \end{bmatrix}
\]
If we use the matrix vector notation $<F_1, f_1, B_1, b_1>$ and $<F_2, f_2, B_2, b_2>$ for two references at two iterations $\alpha$: $(i_1, j_1)$ and $\beta$: $(i_2, j_2)$

Loop bounds for $(i_1, j_1)$

$$B_1 \begin{bmatrix} i_1 \\ j_1 \end{bmatrix} + b_1 \geq 0$$

$$B_2 \begin{bmatrix} i_2 \\ j_2 \end{bmatrix} + b_2 \geq 0$$
If we use the matrix vector notation \(<F_1, f_1, B_1, b_1>\) and \(<F_2, f_2, B_2, b_2>\) for two references at two iterations \(\alpha: (i_1, j_1)\) and \(\beta: (i_2, j_2)\)

\[
\begin{align*}
F_1 &= \begin{bmatrix} i_1 \\ j_1 \end{bmatrix} + f_1 = F_2 \begin{bmatrix} i_2 \\ j_2 \end{bmatrix} + f_2 \\
B_1 &= \begin{bmatrix} i_1 \\ j_1 \end{bmatrix} + b_1 \geq 0 \\
B_2 &= \begin{bmatrix} i_2 \\ j_2 \end{bmatrix} + b_2 \geq 0
\end{align*}
\]

\[
\begin{align*}
F_1 &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & f_1 &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} & F_2 &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & f_2 &= \begin{bmatrix} 0 \\ -1 \end{bmatrix}
\end{align*}
\]

\(B_1, b_1, B_2, b_2\) see previous slides
Three spaces

• Iteration space
  - The set of dynamic execution instances
    For instance, the set of value vectors taken by loop indices
  - A \textit{k-dimensional} space for a \textit{k-level} loop nest

• Data space
  - The set of array elements accessed
  - An \textit{n-dimensional} space for an \textit{n-dimensional} array

• Processor space
  - The set of processors in the system
  - In analysis, we may pretend there are unbounded \# of virtual processors
Affine Half Space

Definition

An affine half-space of $\mathbb{Z}^d$ is defined as the set of points

$$\{ \bar{x} \in \mathbb{Z}^d \mid \bar{a} \ast \bar{x} \leq b \}$$

$\bar{a} \ast \bar{x} = b$ is the hyperplane that divides the $d$-dimensional space into two half-spaces.
Iteration Space

- Bounded by multiple hyperplanes

\[
\begin{align*}
\text{for } (i=0; i \leq 5; i++) \\
\text{for } (j=i; j \leq 7; j++) \\
Z[j, i] &= 0;
\end{align*}
\]
Three Spaces

- Iteration space, data space, and processor space

```c
float Z[100];
for (i=0; i<10; i++)
    S: Z[i+10] = Z[i];
```
Synchronization-free Parallelism

Parallelize an application **without** allowing any *communication* or *synchronization* among (logical) processors.

**Example:**

```
  do i = 1, N
    do j = 1, N
```

*Write in* $S_1(i, 1)$ *to Read in* $S_1(i, 2)$

*Write in* $S_1(i, j)$ *to Read in* $S_1(i, j+1)$

**Dependence from** $S_1$ **to** $S_1$
Synchronization-free Parallelism

Parallelize an application **without** allowing any *communication* or *synchronization* among (logical) processors.

**Example:**

```plaintext
do i = 1, N
    do j = 1, N
    end do
end do
```

*Write in* $S_1(1, 1)$ *to Read in* $S_1(1, 2)$

*Write in* $S_1(i, j)$ *to Read in* $S_1(i, j+1)$

Dependence from $S_1$ to $S_1$

Communication is limited to the iterations on one processor.
Synchronization-free Parallelism

Parallelize an application **without** allowing any *communication* or *synchronization* among (logical) processors.

**Example:**

```plaintext
do i = 1, N
    do j = 1, N
    end do
end do
```

*Write in* $S_1(1,1)$ *to Read in* $S_1(1,2)$

*Write in* $S_1(i, j)$ *to Read in* $S_1(i, j+1)$

Which loop can be parallelized? The “i” loop or the “j” loop?

**Answer:** the “i” loop
Synchronization-free Parallelism

Parallelize an application **without** allowing any *communication* or *synchronization* among (logical) processors.

**Example 1:**

```plaintext
doall i = 1, N
    do j = 1, N
    end do
end do
```

*Write in* $S_1(1, 1)$ *to Read in* $S_1(1, 2)$

*Write in* $S_1(i, j)$ *to Read in* $S_1(i, j+1)$

The dependence chain is characterized by a **hyperplane**. In this case it is “$i = \text{constant}$”.
Parallelize an application **without** allowing any *communication* or *synchronization* among (logical) processors.

Example 2:

```plaintext
do i = 1, N
   do j = 1, N
   end do
end do
```

Dependence from $S1(i,j)$ to $S1(i+1,j+1)$

The dependence chain is characterized by a **hyperplane**. In this case it is “$j - i + \text{constant} = 0$”.
Synchronization-free Parallelism

Parallelize an application **without** allowing any *communication* or *synchronization* among (logical) processors.

**Example 3:**

```c
for (i=1; i<=100; i++)
    for (j=1; j<=100; j++){
        S1: X[i,j] = X[i,j] + Y[i-1, j];
        S2: Y[i,j] = Y[i,j] + X[i, j-1];
    }
```

True, i loop, for Y

True, j loop, for X

Dependence from **S1**(1,1) to **S2**(1,2)

Dependence from **S2**(1,1) to **S1**(2,1)
Dependence and Parallelization

- Dependence chain in affine loops modeled as a hyperplane.
- Iterations along the same hyperplane must execute sequentially.
- **Iterations on different hyperplanes can execute in parallel.**

**Example:**

```plaintext
do i = 1, N
  do j = 1, N
```

Dependence from $S₁(i,j)$ to $S₁(i+1,j+1)$

The hyperplane is $j - i + \text{constant} = 0$.
Processing Space: Affine Partition Schedule

• Map an iteration to a processor using $< C, d >$
  
  $C$ is a $n \times m$ matrix
  
  • $m = d$ (the loop level)
  
  • $n$ is the dimension of the processor grid
  
  $d$ is a $n$-element constant vector
  
  $\vec{p} = C \vec{x} + \vec{d}$, where $\vec{x}$ is an iteration vector
Map an iteration to a processor using $<C, d>$

$$\vec{p} = C \vec{x} + \vec{d}, \text{ where } \vec{x} \text{ is an iteration vector}$$

Example

```plaintext
for (i=1; i<=N; i++)
    S: Y[i] = Z[i];
```

$C = [1], d = [0]$

$$\vec{p}(S(i)) = 1*i + 0$$

$$= i$$

Map iteration $i$ to Processor $i$
Synchronization-free Parallelism

- Two memory references as \(<F_1, f_1, B_1, b_1>\) and \(<F_2, f_2, B_2, b_2>\) such that \(<F_2, f_2, B_2, b_2>\) at iteration \(\alpha\): \((i_1, j_1)\) depends on \(<F_1, f_1, B_1, b_1>\) at iteration \(\beta\): \((i_2, j_2)\)
  
  - \(F_1\) is a matrix and \(f_1\) is a vector. 
    The affine memory access index is \(F_1^* \alpha + f_1\).
  
  - \(B_1\) is a matrix and \(b_1\) is a vector. 
    The affine loop bounds can be expressed as \(B_1^* \alpha + b_1 \geq 0\)

- Let \(<C_1, d_1>\) and \(<C_2, d_2>\) represent the respective processor schedule, to have synchronization-free parallelism,
  \[ C_1^* \alpha + d_1 = C_2^* \beta + d_2 \]

These two memory references must execute on the same processor (sequentially).
Synchronization-free Parallelism

• Two memory references as \(<F_1, f_1, B_1, b_1>\) and \(<F_2, f_2, B_2, b_2>\) such that \(<F_2, f_2, B_2, b_2>\) at iteration \((i_1, j_1)\) depends on \(<F_1, f_1, B_1, b_1>\) at iteration \((i_2, j_2)\)

```
for (i=1; i<=100; i++)
  for (j=1; j<=100; j++){
    S1: \(X[i, j] = X[i, j] + Y[i-1, j]\);
    S2: \(Y[i, j] = Y[i, j] + X[i, j-1]\);
  }
```

• We want to find processor schedule \(<C_1, d_1>\) and \(<C_2, d_2>\) such that

\[
\begin{bmatrix}
C_{11} & C_{12}
\end{bmatrix}
\begin{bmatrix}
i_1 \\
j_1
\end{bmatrix}
+ \begin{bmatrix}
d_1 \\
j_1
\end{bmatrix}
= \begin{bmatrix}
C_{21} & C_{22}
\end{bmatrix}
\begin{bmatrix}
i_2 \\
j_2
\end{bmatrix}
+ \begin{bmatrix}
d_2 \\
j_2
\end{bmatrix}
\]

\[
F_1 \begin{bmatrix} i_1 \\ j_1 \end{bmatrix} + f_1 = F_2 \begin{bmatrix} i_2 \\ j_2 \end{bmatrix} + f_2
\]

\[
B_1 \begin{bmatrix} i_1 \\ j_1 \end{bmatrix} + b_1 >= 0
\]

\[
B_2 \begin{bmatrix} i_2 \\ j_2 \end{bmatrix} + b_2 >= 0
\]
for (i=1; i<=100; i++)
    for (j=1; j<=100; j++){
        S1: \(X[i, j] = X[i, j] + Y[i-1, j];\)
        S2: \(Y[i, j] = Y[i, j] + X[i, j-1];\)
    }

\[
\begin{bmatrix}
C_{11} & C_{12} \\
j_1 & \\
i_1
\end{bmatrix} + [d_1] = \begin{bmatrix}
C_{11} & C_{12} \\
j_2 & \\
i_2
\end{bmatrix} + [d_2]
\]

\[
\begin{bmatrix}
C_{11} - C_{21} & C_{12} - C_{22} \\
j_1 & \\
i_1
\end{bmatrix} + [d_1 - d_2 - C_{22}] = 0
\]

\(C_{11} - C_{21} = 0\)
\(C_{12} - C_{22} = 0\)
\(d_1 - d_2 - C_{22} = 0\)

\textbf{S1 to S2 dependence}
Synchronization-free Parallelism

for (i=1; i<=100; i++)
    for (j=1; j<=100; j++){
        S1: X[i,j] = X[i,j] + Y[i-1, j];
        S2: Y[i,j] = Y[i,j] + X[i, j-1];
    }

True, i loop, for Y

True, j loop, for X

\[
\begin{bmatrix}
C_{11} & C_{12}
\end{bmatrix}
\begin{bmatrix}
i_3 \\
j_3
\end{bmatrix} + [d_1] = \begin{bmatrix}
C_{11} & C_{12}
\end{bmatrix}
\begin{bmatrix}
i_4 \\
j_4
\end{bmatrix} + [d_2]
\]

C_{11} - C_{21} = 0
C_{12} - C_{22} = 0
\(d_1 - d_2 + C_{21} = 0\)

[S2 to S1 dependence]

\[
\begin{bmatrix}
C_{11} - C_{21}, C_{12} - C_{22}
\end{bmatrix}
\begin{bmatrix}
i_3 \\
j_3
\end{bmatrix} + [d_1 - d_2 + C_{21}] = 0
\]
Synchronization-free Parallelism

for (i=1; i<=100; i++)
    for (j=1; j<=100; j++){
        S1:  \( X[i,j] = X[i,j] + Y[i-1,j]; \)
        S2:  \( Y[i,j] = Y[i,j] + X[i,j-1]; \)
    }

True, i loop, for Y

S1
True, j loop, for X
S2

S1 to S2 dependence

\[
\begin{align*}
C_{11} - C_{21} &= 0 \\
C_{12} - C_{22} &= 0 \\
d_1 - d_2 - C_{22} &= 0
\end{align*}
\]

S2 to S1 dependence

\[
\begin{align*}
C_{11} - C_{21} &= 0 \\
C_{12} - C_{22} &= 0 \\
d_1 - d_2 + C_{21} &= 0
\end{align*}
\]

\[
C_{11} = C_{21} = -C_{22} = -C_{12} = d_2 - d_1
\]
Synchronization-free Parallelism

Example:

```c
for (i=1; i<=100; i++)
    for (j=1; j<=100; j++){
        S1: X[i,j] = X[i,j] + Y[i-1, j];
        S2: Y[i,j] = Y[i,j] + X[i, j-1];
    }
```

$$C_{11} = C_{21} = -C_{22} = -C_{12} = d_2 - d_1$$

One Potential Solution:
Affine schedule for S1, p(S1):  
$$[C_{11} \ C_{12}] = [1 \ -1], \quad d_1 = -1$$ 
  i.e.  (i, j) iteration of S1 to processor p = i - j - 1;

Affine schedule for S2, p(S2):  
$$[C_{21} \ C_{22}] = [1 \ -1], \quad d_2 = 0$$ 
  i.e.  (i, j) iteration of S2 to processor p = i - j.
Affine partition schedule

\begin{align*}
do I = 1, N \\
do J = 1, N \\
\end{align*}

Affine schedule for \( S_1 \), \( p(S_1) \):
\[
C = \begin{bmatrix} C_{11} & C_{12} \end{bmatrix} = \begin{bmatrix} 1 & -1 \end{bmatrix}, \quad d = 0
\]

i.e. \( (i, j) \) iteration of \( S_1 \) to processor \( p = i - j \);
Affine partition schedule

\[
\text{do } I = 1, N \\
\text{do } J = 1, N \\
\]

Write After Read

Read in \(S_1(1,2)\) to Write in \(S_1(2,1)\)

\(S_1(i, i)\) to \(S_1(i+1, i-1)\)

The hyperplane is \(j + i = \text{“a constant”}\)

Affine schedule for \(S_1\), \(p(S1)\):

\[
C = [C_{11} \ C_{12}] = [1 \ 1], \quad d = 0
\]

i.e. \((i, j)\) iteration of \(S1\) to processor \(p = i + j\) ;