Lecture 10
Loop Transformations for Communication-Free Parallelism

1. Examples of Loop Transformations
2. Affine Partitioning
3. Code Generation

Readings: Chapter 11-11.3, 11.6-11.7.4, 11.9-11.9.6
Shared Memory Machines

Performance on Shared Address Space Multiprocessors: Parallelism & Locality

Interconnect
1. Parallelism and Locality

- Parallelism DOES NOT imply speed up!

- Parallel performance:
  - Improve locality with loop transformations
  - Minimize communication
  - Operations using the same data are executed on the same processor

- Sequential performance:
  - Improve locality with loop transformations
  - Minimize cache misses
  - Operations using the same data are executed close in time.
Loop Permutation (Loop Interchange)

for I = 1 to 4
  for J = 1 to 3
    Z[I,J] = Z[I-1,J]

for J' = 1 to 3
  for I' = 1 to 4
    Z[I',J'] = Z[I'-1,J']

\[
\begin{bmatrix}
  j' \\
i'
\end{bmatrix} = \begin{bmatrix}
  0 & 1 \\
  1 & 0
\end{bmatrix} \begin{bmatrix}
i \\
j
\end{bmatrix}
\]
for I = 1 to 4
    T[I] = A[I] + B[I]  \text{ (s1)}
for I' = 1 to 4
    C[I'] = T[I'] \times T[I']  \text{ (s2)}

for J = 1 to 4
    C[J] = T[J] \times T[J]  \text{ (s2)}
Loop Transformations

• Unimodular transforms on loop nests
  – Interchange
  – Skewing
  – Reversal
• Cross statement transforms
  – Loop fusion
  – Loop fission
  – Re-indexing
• How to combine them to get parallelism and locality?
2. Affine Partitioning:
A Contrived but Illustrative Example

FOR \( j = 1 \) TO \( n \)
FOR \( i = 1 \) TO \( n \)
\[
A[i,j] = A[i,j] + B[i-1,j]; \quad (S_1)
\]
\[
B[i,j] = A[i,j-1] \times B[i,j]; \quad (S_2)
\]
Best Parallelization Scheme

Algorithm finds affine partition mappings for each instruction:

S1: Execute iteration \((i, j)\) on processor \(i-j\).

S2: Execute iteration \((i, j)\) on processor \(i-j+1\).

SPMD code: Let \(p\) be the processor’s ID number

\[
\text{if } (1-n \leq p \leq n) \text{ then } \\
\quad \text{if } (1 \leq p) \text{ then } \\
\quad \quad B[p,1] = A[p,0] \times B[p,1]; \quad (S_2) \\
\quad \text{for } i_1 = \max[1,1+p) \text{ to } \min[n,n-1+p) \text{ do } \\
\quad \quad A[i_1,i_1-p] = A[i_1,i_1-p] + B[i_1-1,i_1-p]; \quad (S_1) \\
\quad \quad B[i_1,i_1-p+1] = A[i_1,i_1-p] \times B[i_1,i_1-p+1]; \quad (S_2) \\
\quad \text{if } (p \leq 0) \text{ then } \\
\quad \quad A[n+p,n] = A[n+p,N] + B[n+p-1,n]; \quad (S_1)
\]
Iteration Space

FOR $i = 0$ to $5$
  FOR $j = i$ to $7$
    ...

• $n$-deep loop nests: $n$-dimensional polytope
• Iterations: coordinates in the iteration space
• Assume: iteration index is incremented in the loop
• Sequential execution order: lexicographic order
  – $[0,0]$, $[0,1]$, …, $[0,6]$, $[0,7]$, $[1,1]$, …, $[1,6]$, $[1,7]$, …
For every pair of data dependent accesses $F_1i_1+f_1$ and $F_2i_2+f_2$

Find $C_1$, $c_1$, $C_2$, $c_2$:

$$\forall i_1, i_2 \quad F_1i_1+f_1 = F_2i_2+f_2 \implies C_1i_1+c_1 = C_2i_2+c_2$$

with the objective of maximizing the rank of $C_1$, $C_2$
Rank of Partitioning = Degree of Parallelism

Affine Mapping

\[
\begin{bmatrix}
0 & 0
\end{bmatrix}
\begin{bmatrix}
i
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & 1
\end{bmatrix}
\begin{bmatrix}
i
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 0
\end{bmatrix}
\begin{bmatrix}
i
\end{bmatrix}
\]

Rank

0

1

2

Mapped to same processor
Solution to the Contrived Example

FOR \( j = 1 \) TO \( n \)
  FOR \( i = 1 \) TO \( n \)
    \( A[i,j] = A[i,j] + B[i-1,j]; \) \hspace{1cm} (S_1)
    \( B[i,j] = A[i,j-1]*B[i,j]; \) \hspace{1cm} (S_2)

\[ S_1 \hspace{1cm} p = [1 \hspace{2mm} -1]^{i\hspace{2mm} j} + 0 \]
\[ S_2 \hspace{1cm} p = [1 \hspace{2mm} -1]^{i\hspace{2mm} j} + 1 \]
Example 1: Loop Transform

Find affine partitioning: $c_1, c_2, c_0$ such that

$$p = \begin{bmatrix} c_1 & c_2 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} + c_0$$

Suppose iteration $i, j$ & $i', j'$ refer to same location

$$i = i' - 1$$
$$j = j'$$

No communication means:

$$c_1 i + c_2 j + c_0 = c_1 i' + c_2 j' + c_0$$

$$c_1(i'-1) + c_2 j' + c_0 = c_1 i' + c_2 j' + c_0$$

$$c_1 = 0$$
$$p = c_2 j + c_0$$

Pick simplest, non-null $c_2, c_0$:

$$c_2 = 1, c_0 = 0$$
$$p = j$$
3. Code Generation

• Naive
  – Each processor visits all the iterations
  – Executes only if it owns that iteration

• Optimization
  – Removes unnecessary looping and condition evaluation
for $I = 1$ to 4  
for $J = 1$ to 3  
$Z[I,J] = Z[I-1,J]$  

for $P = 1$ to 3  
for $I = 1$ to 4  
$Z[I,P] = Z[I-1,P]$

SPMD (single program multiple data) code:  
for $I = 1$ to 4  
$Z[I,P] = Z[I-1,P]$
Loop Permutation (Loop Interchange)

for I = 1 to 4
for J = 1 to 3
Z[I,J] = Z[I-1,J]

for P = 1 to 3
for I = 1 to 4
Z[I,P] = Z[I-1,P]

\[
\begin{bmatrix}
p' \\
i'
\end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix}
\]
Example 2: Loop Fusion

Find affine partitioning: \( c_{1,1}, c_{1,0}, c_{2,1}, c_{1,0} \), such that

\[
\text{s1: } [p] = [c_{1,1}][i] + c_{1,0}
\]
\[
\text{s2: } [p] = [c_{2,1}][i'] + c_{2,0}
\]

Suppose iteration \( i \) & \( i' \) refer to the same location

\( i = i' \)

No communication means:

\( c_{1,1} i + c_{1,0} = c_{2,1} i' + c_{2,0} \)

\( c_{1,1} = c_{2,1} \)
\( c_{1,0} = c_{2,0} \)

Pick simplest, non-null values: \( c_{1,1} = c_{2,1} = 1, c_{1,0} = c_{2,0} = 0 \)

\( p = i; p = i' \)
for $I = 1$ to $4$

\[ T[I] = A[I] + B[I] \quad (s1) \]

for $I' = 1$ to $4$

\[ C[I'] = T[I'] \times T[I'] \quad (s2) \]

\[ p[i] = [1][i] \quad (s1) \]

\[ p[i'] = [1][i'] \quad (s2) \]

for $P = 1$ to $4$


\[ C[P] = T[P] \times T[P] \quad (s2) \]
Example 3: 2 Nested, Parallel Loops

Find affine partitioning: \( c_1, c_2, c_0 \) such that

\[
p = \begin{bmatrix} c_1 & c_2 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} + c_0
\]

Suppose iteration \( i,j \) & \( i', j' \) refer to same location

\[
i = i' \\
j = j'
\]

No communication means:

\[
c_1 i + c_2 j + c_0 = c_1 i' + c_2 j' + c_0
\]

\[
c_1 i' + c_2 j' + c_0 = c_1 i + c_2 j + c_0
\]

No constraints
Two basis vectors: \([c_1 \ c_2] = [1 \ 0]\), or \([c_1 \ c_2] = [0 \ 1]\)
Two answers for \( p \): two degrees of parallelism

\[
\begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix}
\]
Example 3: 2 Nested, Parallel Loops

\begin{align*}
\text{for } I = 1 \text{ to } 4 & \\
\text{for } J = 1 \text{ to } 3 & \\
Z[I,J] &= Z[I,J]+1
\end{align*}

\begin{align*}
\text{for } p_1 = 1 \text{ to } 4 & \\
\text{for } p_2 = 1 \text{ to } 3 & \\
Z[p_1,p_2] &= Z[p_1,p_2]+1
\end{align*}

\begin{align*}
\begin{bmatrix}
p_1 \\
p_2
\end{bmatrix} &=
\begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
i \\
j
\end{bmatrix}
\end{align*}

\begin{align*}
\text{for } p_1 = 1 \text{ to } 4 & \\
\text{for } p_2 = 1 \text{ to } 3 & \\
\text{if } (I==p_1 \& J == p_2) & \\
Z[I,J] &= Z[I,J]+1
\end{align*}

\begin{align*}
\text{for } p_1 = 1 \text{ to } 4 & \\
\text{for } p_2 = 1 \text{ to } 3 & \\
Z[p1,p2] &= Z[p1,p2]+1
\end{align*}
Optimizing Arbitrary Loop Nesting Using Affine Partitions (chotst, NAS)

DO 1 J = 0, N
   IO = MAX ( -M, -J )
   DO 2 I = IO, -1
      DO 3 JJ = IO - I, -1
         DO 3 L = 0, NMAT
      DO 2 L = 0, NMAT
      2 A(L, I, J) = A(L, I, J) * A(L, 0, I+J)
   DO 4 L = 0, NMAT
      EPSS(L) = EPS * A(L, 0, J)
   DO 5 JJ = IO, -1
      DO 5 L = 0, NMAT
         A(L, 0, J) = A(L, 0, J) - A(L, JJ, J) ** 2
      DO 1 L = 0, NMAT
         A(L, 0, J) = 1. / SQRT ( ABS (EPSS(L) + A(L, 0, J)) )
   DO 6 I = 0, NRHS
      DO 7 K = 0, N
         DO 8 L = 0, NMAT
            B(I, L, K) = B(I, L, K) * A(L, 0, K)
         DO 7 JJ = 1, MIN (M, N-K)
            DO 7 L = 0, NMAT
               B(I, L, K+JJ) = B(I, L, K+JJ) - A(L, -JJ, K+JJ) * B(I, L, K)
         DO 6 K = N, 0, -1
         DO 9 L = 0, NMAT
            B(I, L, K) = B(I, L, K) * A(L, 0, K)
         DO 6 JJ = 1, MIN (M, K)
            DO 6 L = 0, NMAT
               B(I, L, K-JJ) = B(I, L, K-JJ) - A(L, -JJ, K) * B(I, L, K)
Chotst: Results with Affine Partitioning + Blocking

(Unimodular: a subset of affine partitioning for perfect loop nests)
Maximum Parallelism & No Communication

For every pair of data dependent accesses $F_{1i_1+f_1}$ and $F_{2i_2+f_2}$

Find $C_1$, $c_1$, $C_2$, $c_2$:

$\forall i_1, i_2 \quad F_{1i_1+f_1} = F_{2i_2+f_2} \rightarrow C_{1i_1+c_1} = C_{2i_2+c_2}$

with the objective of maximizing the rank of $C_1$, $C_2$