Lecture 11
Loop Transformations
for Parallelism and Locality

1. Examples
2. Affine Partitioning: Do-all
3. Affine Partitioning: Pipelining

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
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)

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

for \(I = 1 \text{ to } 4\)
for \(J = 1 \text{ to } 3\)
\(Z[I,J] = Z[I-1,J]\)

for \(I' = 1 \text{ to } 3\)
for \(J' = 1 \text{ to } 4\)
\(Z[I',J'] = Z[I'-1,J']\)
Carnegie Mellon

Loop Fusion

\[ \text{for } I = 1 \text{ to } 4 \]
\[ T[I] = A[I] + B[I] \quad (s1) \]
\[ \text{for } I' = 1 \text{ to } 4 \]
\[ C[I'] = T[I'] \times T[I'] \quad (s2) \]

\[ \text{s1: } [j] = [i] [i'] \]
\[ \text{s2: } [j] = [i] [i'] \]

M. Lam

CS243: Loop Transformations

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?
Affine Partitioning:
An 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]*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

if (1-n <= p <= n) then
    if [1 <= p) then
        B[p,1] = A[p,0] * B[p,1]; \quad (S_2)
    for i_1 = max[1,1+p) to min[n,n-1+p) do
        A[i_1,i_1-p] = A[i_1,i_1-p] + B[i_1-1,i_1-p]; \quad (S_1)
        B[i_1,i_1-p+1] = A[i_1,i_1-p] * B[i_1,i_1-p+1]; \quad (S_2)
    if (p <= 0) then
        A[n+p,n] = A[n+p,N] + B[n+p-1,n]; \quad (S_1)
2. 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
  - $[0,0], [0,1], ..., [0,6], [0,7], [1,1], ..., [1,6], [1,7], ...$

Maximum Parallelism & No Communication

For every pair of data dependent accesses $F_{i_1} + f_1$ and $F_{j_2} + f_2$

Find $C_{i}, C_{j}, C_{i'}, C_{i''}$:

$\forall i, i' \quad F_{i_1} + f_1 = F_{j_2} + f_2 \rightarrow C_{i_1} + C_{i} = C_{j_2} + C_{i}$

with the objective of maximizing the rank of $C_{i}, C_{i''}$
Rank of Partitioning = Degree of Parallelism

Affine Mapping

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

Rank

0
1
2

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 \(c_2, c_0\): \(c_2 = 1, c_0 = 0\)
\[p = j\]
Code Generation

- **Naive**
  - Each processor visits all the iterations
  - Executes only if it owns that iteration
- **Optimization**
  - Removes unnecessary looping and condition evaluation

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

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 $4$
$Z[I,P] = Z[I-1,P]$

$[p'] = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} [j]$

**Example 2: Loop Fusion**

for $I = 1$ to $4$
$T[I]= A[I]+B[I]$  (s1)
for $I' = 1$ to $4$
$C[I']= T[I'] \times T[I']$  (s2)

Find affine partitioning: $c_{1,1}, c_{1,0}, c_{2,1}, c_{2,0}$, such that

s1: $[p] = \begin{bmatrix} c_{1,1} \\ c_{1,0} \end{bmatrix} [i] + c_{1,0}$
s2: $[p] = \begin{bmatrix} c_{2,1} \\ c_{2,0} \end{bmatrix} [i'] + c_{2,0}$

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 values: $c_{1,1} = c_{2,1} = 1$, $c_{1,0} = c_{2,0} = 0$
$p = i; p = i'$
Loop Fusion

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)

$S1: \begin{bmatrix} p \end{bmatrix} = \begin{bmatrix} 1 \\ i \end{bmatrix}$
$S2: \begin{bmatrix} p \end{bmatrix} = \begin{bmatrix} 1 \\ i' \end{bmatrix}$

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:

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

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**

For $I = 1$ to $4$

For $J = 1$ to $3$

$Z[I,J] = Z[I,J] + 1$

For $p1 = 1$ to $4$

For $p2 = 1$ to $3$

$Z[p1,p2] = Z[p1,p2] + 1$

\[
[p_1] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}
\]

For $I = 1$ to $4$

For $J = 1$ to $3$

If $(I == p1 & J == p2)$

$Z[I,J] = Z[I,J] + 1$

For $p1 = 1$ to $4$

For $p2 = 1$ to $3$

$Z[p1,p2] = Z[p1,p2] + 1$

---

**Optimizing Arbitrary Loop Nesting Using Affine Partitions (chotst, NAS)**

DO $I = 0$, N

DO $J = 0$, N

ID = MAX ($-I$, $-J$)

DO $I = I0$, $-1$

DO $J = J0$, $-1$

DO $L = 0$, NMAT


DO $L = 0$, NMAT

2 $A(L,I,J) = A(L,I,J) * A(L,0,I+J)$

DO $I = 0$, NRHS

DO $K = 0$, N

DO $L = 0$, NMAT

8 $B(I,L,K) = B(I,L,K) * A(L,0,K)$

DO $J = J0$, $-1$

DO $L = 0$, NMAT

7 $B(I,L,K+JJ) = B(I,L,K+JJ) - A(L,-JJ,K+JJ) * B(I,L,K)$

DO $K = K$, $0$, $-1$

DO $L = 0$, NMAT

9 $B(I,L,K) = B(I,L,K) * A(L,0,K)$

DO $J = J0$, $-1$

DO $L = 0$, NMAT

6 $B(I,L,K-JJ) = B(I,L,K-JJ) - A(L,-JJ,K) * B(I,L,K)$

---
### Chotst: Results with Affine Partitioning + Blocking

(Unimodular: a subset of affine partitioning for perfect loop nests)

<table>
<thead>
<tr>
<th>Number of Processors</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

#### Summary of Affine Partitioning

**Communication-Free**

- Loops: \( F_{i_1} + f_1 \), \( F_{i_2} + f_2 \), \( C_{i_1} + c_1 \), \( C_{i_2} + c_2 \)
- Array
- Processor ID
Advanced topic: Pipelining

SOR (Successive Over-Relaxation): An Example

for i = 1 TO m
  for j = 1 to n

Finding the Maximum Degree of Pipelining

For every pair of data dependent accesses \( F_{i_1} + f_1 \) and \( F_{i_2} + f_2 \)
Let \( B_{i_1} + b_1 \geq 0, B_{i_2} + b_2 \geq 0 \) be the corresponding loop bound constraints,

Find \( C_{i_1}, c_{i_1}, C_{i_2}, c_{i_2} \):

\[ \forall i_1, i_2 \quad B_{i_1} + b_1 \geq 0, B_{i_2} + b_2 \geq 0 \]

\( (i_1, i_2) \land (F_{i_1} + f_1 = F_{i_2} + f_2) \rightarrow C_{i_1} + c_{i_1} \leq C_{i_2} + c_{i_2} \)

with the objective of maximizing the rank of \( C_{i_1}, C_{i_2} \)
Key Insight

- Choice in time mapping => (pipelined) parallelism
- Rank(C) - 1 degree of parallelism with 1 degree of synchronization
- Can create blocks with Rank(C) dimensions

- Find time partitions is not as straightforward as space partitions
  - Need to deal with linear inequalities
  - Solved using Farkas Lemma - no simple intuitive proof

Summary of Affine Partitioning

Communication-Free

Loops

Array

Processor ID

Pipelining

i_1 \leq i_2

Loops

Array

Time Stage

C_i_{i_1} + C_1

F_{i_1} + f_1

F_{i_2} + f_2

C_{j_1} + C_2