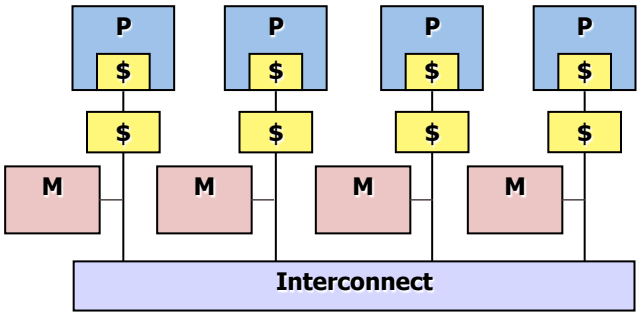
Lecture 11
Loop Transformations for Parallelism and Locality

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

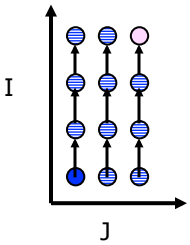


Parallelism and Locality

- 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.

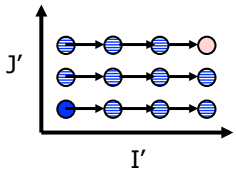
Loop Permutation (Loop Interchange)

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



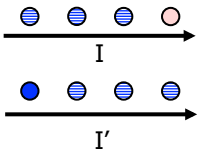
$$\begin{bmatrix} j' \\ i' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix}$$

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



Loop Fusion

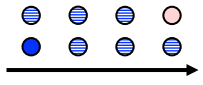
```
for I = 1 to 4
    T[I]= A[I]+B[I]     (s1)
for I' = 1 to 4
    C[I']= T[I'] x T[I'] (s2)
```



s1: $\begin{bmatrix} j \end{bmatrix} = \begin{bmatrix} 1 \end{bmatrix} \begin{bmatrix} i \end{bmatrix}$

s2: $\begin{bmatrix} j \end{bmatrix} = \begin{bmatrix} 1 \end{bmatrix} \begin{bmatrix} i' \end{bmatrix}$

```
for J = 1 to 4
    T[J]= A[J]+B[J]  (s1)
    C[J]= T[J] x T[J] (s2)
```

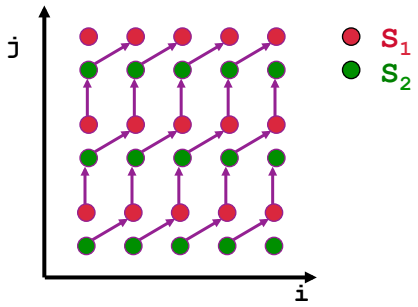


Loop Transformations

- 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?

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];          (S1)
    B[i,j] = A[i,j-1]*B[i,j];          (S2)
```



Best Parallelization Scheme

Algorithm finds affine partition mappings for each instruction:

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

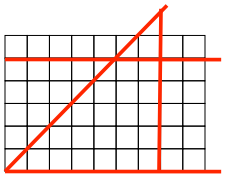
S2: Execute iteration (i, j) on processor i-j+1.

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];                    (S2)
  for i1 = max[1,1+p) to min[n,n-1+p) do
      A[i1,i1-p] = A[i1,i1-p] + B[i1-1,i1-p];      (S1)
      B[i1,i1-p+1] = A[i1,i1-p] * B[i1,i1-p+1];    (S2)
  if (p <= 0) then
      A[n+p,n] = A[n+p,N] + B[n+p-1,n];            (S1)
```

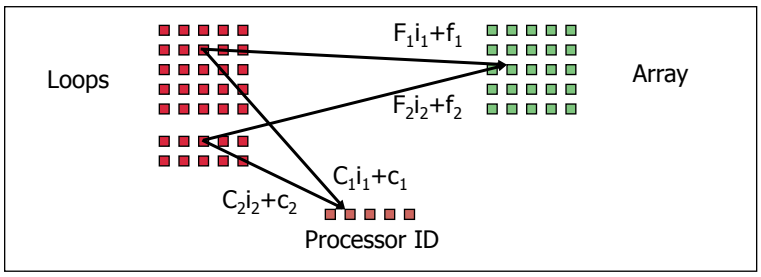
2. Iteration Space

```
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 $F_1i_1+f_1$ and $F_2i_2+f_2$

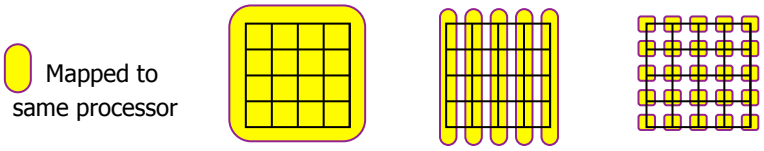
Find C_1, c_1, C_2, c_2:

$\forall\ i_1, i_2 \quad F_1 i_1 + f_1 = F_2 i_2 + f_2 \rightarrow C_1 i_1 + c_1 = C_2 i_2 + c_2$

with the objective of maximizing the rank of C_1, C_2

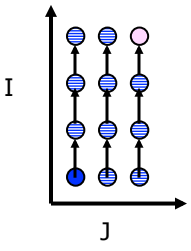
Rank of Partitioning = Degree of Parallelism

Affine Mapping	$\begin{bmatrix}0 & 0\end{bmatrix}\begin{bmatrix}i\\j\end{bmatrix}$	$\begin{bmatrix}0 & 1\end{bmatrix}\begin{bmatrix}i\\j\end{bmatrix}$	$\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\begin{bmatrix}i\\j\end{bmatrix}$
Rank	0	1	2



Example 1: Loop Transform

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



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: $c_2 = 1$, $c_0 = 0$

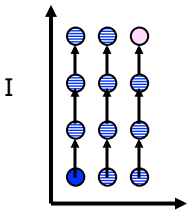
$p = j$

Code Generation

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

Code Generation

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



p = j

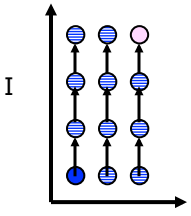
```
for P = 1 to 3
  for I = 1 to 4
    for J = 1 to 3
      if (j == P)
        Z[I,J] = Z[I-1,J]

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]
```

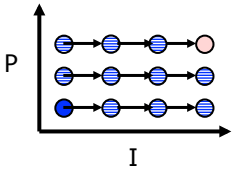
Loop Permutation (Loop Interchange)

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



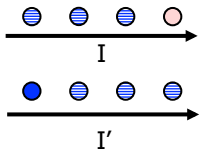
$$\begin{bmatrix} p' \\ i' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix}$$

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



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'] x T[I'] (s2)
```



Find affine partitioning: $c_{1,1}, c_{1,0}, c_{2.1}, c_{1,0},$ such that

s1: $\left[p\right] = \left[c_{1,1}\right]\left[i\right] + c_{1,0}$

s2: $\left[p\right] = \left[c_{2,1}\right]\left[i'\right] + c_{2,0}$

Suppose iteration i & i' refer to the same location

$i = i'$

No communication means:

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

$c_{1,1} = c_{2,1}$

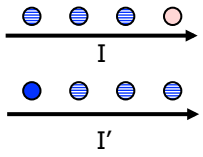
$c_{1,0} = c_{2,0}$

Pick simplest values: $c_{1,1} = c_{2,1} = 1$, $c_{1,0} = c_{2,0} = 0$

$p = i;\ p = i'$

Loop Fusion

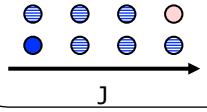
```
for I = 1 to 4
    T[I]= A[I]+B[I]     (s1)
for I' = 1 to 4
    C[I']= T[I'] x T[I'] (s2)
```



s1: $[p] = [1]\,[i]$

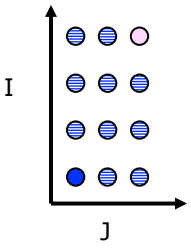
s2: $[p] = [1]\,[i']$

```
for P = 1 to 4
  for I = 1 to 4
    if (I == P)
      T[I]= A[I]+B[I]      (s1)
  for I' = 1 to 4
    if (I' == P)
      C[I']= T[I'] x T[I']  (s2)

for P = 1 to 4
    T[P]= A[P]+B[P]   (s1)
    C[P]= T[P] x T[P]  (s2)
```



Example 3: 2 Nested, Parallel Loops

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



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

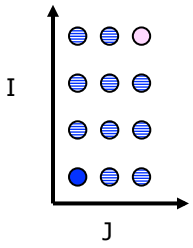
Two basis vectors: $[c_1\ c_2]=[1\ 0]$, or $[c_1\ c_2] = [0\ 1]$

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

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

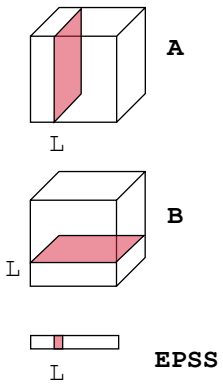


$$\begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix}$$

```
for p1 = 1 to 4
  for p2 = 1 to 3
    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
```

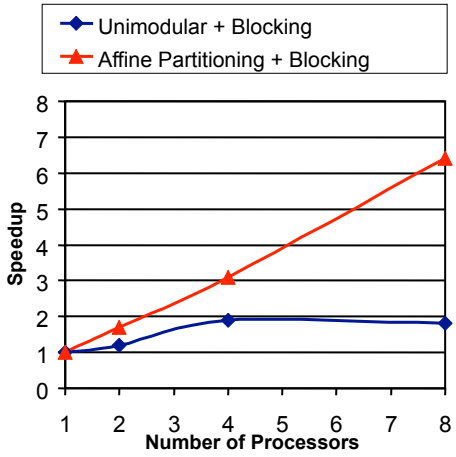

Optimizing Arbitrary Loop Nesting Using Affine Partitions (chotst, NAS)

```
      DO 1 J = 0, N
       I0 = MAX ( -M, -J )
         DO 2 I = I0, -1
           DO 3 JJ = I0 - I, -1
             DO 3 L = 0, NMAT
3              A(L,I,J) = A(L,I,J) - A(L,JJ,I+J) * A(L,I+JJ,J)
           DO 2 L = 0, NMAT
2            A(L,I,J) = A(L,I,J) * A(L,0,I+J)
         DO 4 L = 0, NMAT
4          EPSS(L) = EPS * A(L,0,J)
         DO 5 JJ = I0, -1
           DO 5 L = 0, NMAT
5            A(L,0,J) = A(L,0,J) - A(L,JJ,J) ** 2
         DO 1 L = 0, NMAT
1          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
8            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
7              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
9            B(I,L,K) = B(I,L,K) * A(L,0,K)
           DO 6 JJ = 1, MIN (M, K)
             DO 6 L = 0, NMAT
6              B(I,L,K-JJ) = B(I,L,K-JJ) - A(L,-JJ,K) * B(I,L,K)
```

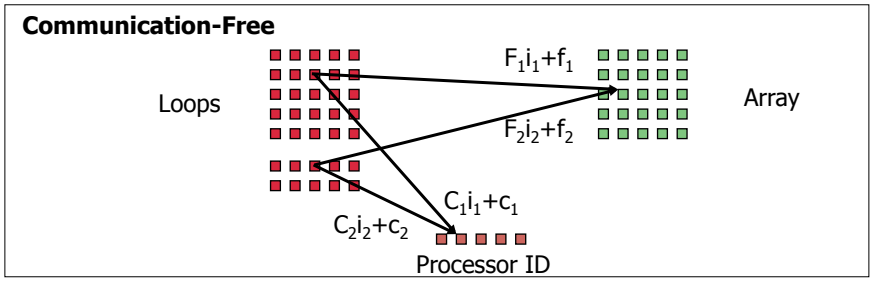


Chotst: Results with Affine Partitioning + Blocking

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

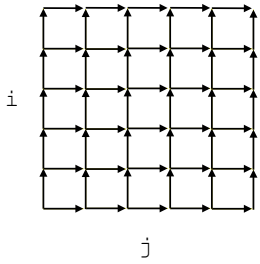


Summary of Affine Partitioning

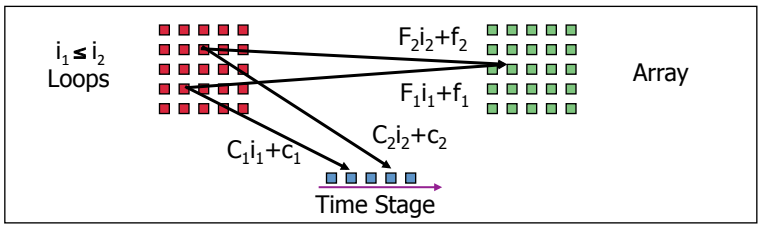


Advanced topic: Pipelining
SOR (Successive Over-Relaxation): An Example

```
for i = 1 TO m
   for j = 1 to n
     A[i,j] = c * (A[i-1,j] + A[i,j-1])
```



Finding the Maximum Degree of Pipelining



For every pair of data dependent accesses $F_1 i_1 + f_1$ and $F_2 i_2 + f_2$
Let $B_1 i_1 + b_1 \ge 0$, $B_2 i_2 + b_2 \ge 0$ be the corresponding loop bound constraints,

Find C_1, c_1, C_2, c_2:

$\forall\, i_1, i_2 \quad B_1 i_1 + b_1 \ge 0, \; B_2 i_2 + b_2 \ge 0$

$(i_1 \le i_2) \wedge (F_1 i_1 + f_1 = F_2 i_2 + f_2) \rightarrow C_1 i_1 + c_1 \le C_2 i_2 + c_2$

with the objective of maximizing the rank of C_1, C_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