

Effective Automatic Parallelization of Stencil Computations

- Parallelization of stencil codes
- Load balance
- Data locality
- 1-D Jacobi Code
- N-D Jacobi Code

Loop Tiling

- Data locality
- Data reuse
- Inter-tile dependences could inhibit concurrent execution of tiles on different processors, even if it is possible originally
- In general, tiling transforms an n -deep loop nest into a $2n$ -deep loop nest where the inner n loops execute a compiler-determined number of iterations

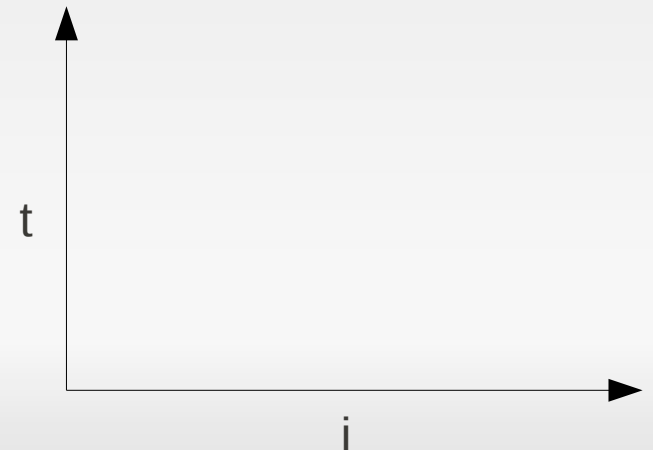
A Data Locality Optimizing Algorithm

- Michael E. Wolf and Monica S. Lam, 1991
- Extract dependence information
- Extract locality information
- Search transformation space
- Skew to make inner loop permutable
- Generate Final code

Iteration space

- A loop nest of depth n corresponds to a finite convex polyhedron of iteration space Z^n , bounded by the loop bounds
- Each iteration in the loop corresponds to a node in the polyhedron, and is identified by its index vector $(p_1, p_2, \dots, p_n)$, p_i is the loop index of the i_{th} loop in the nest

```
for t = 0 to T - 1
  for i = 0 to N - 1
    A[t][i] = (A[t - 1][i - 1] + A[t - 1][i] + A[t - 1][i + 1]) / 3
```



Dependence vectors

- http://en.wikipedia.org/wiki/Loop_dependence_analysis
- The dependence vectors define a partial order on the nodes in the iteration space, and any topological ordering on the graph is a legal execution order, as all dependences in the loop are satisfied

- $(i_1 - j_1, i_2 - j_2, \dots, i_n - j_n)$

$$\begin{aligned}(t - (t - 1), i - i) &= (1, 0) \\(t - (t - 1), i - (i - 1)) &= (1, 1) \\(t - (t - 1), i - (i + 1)) &= (1, -1)\end{aligned}$$

```
for t = 0 to T - 1
  for i = 0 to N - 1
    A[t][i] = (A[t - 1][i - 1] + A[t - 1][i] + A[t - 1][i + 1]) / 3
```

```
foreach i1
  foreach i2
```

Perfectly-nested loop,
all computation is nested
in the innermost loop

```
....
foreach in
  a[i1, i2, ..., in] = a[j1, j2, ..., jn]
```

Unimodular loop transformations

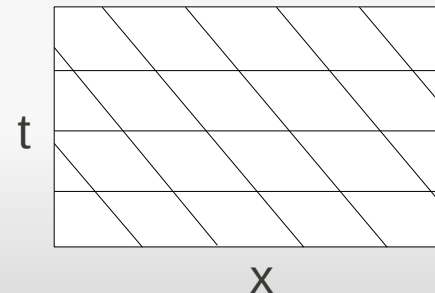
- Due to dependences, the loops might need to be first be skewed before they can be tiled
- With dependences represented as vectors in the iteration space, loop transformations such as interchange, skewing, and reversal, can be represented as matrix transformations
- Non-rectangular tiles are obtained by first applying unimodular transformations to the iteration space and then applying tiling

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

for interchanging

$$\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

for skewing



Loop Tiling and Transformations

- It is not always possible to tile. Loops I_i through I_j in a loop nest can be tiled if they are fully permutable. They are fully permutable if and only if all dependence vectors are lexicographically positive, or, in other words, all elements in any dependence vectors are positive. $\Rightarrow$ There exists a valid total ordering of the tiles (supernode partition)

```
for t = 0 to T - 1  
  for i = 0 to N - 1
```

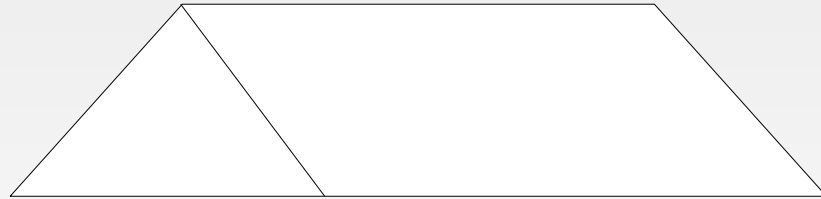
```
    A[t][i] = (A[t - 1][i - 1] + A[t - 1][i] + A[t - 1][i + 1]) / 3
```

$(t - (t - 1), i - i) = (1, 0)$
 $(t - (t - 1), i - (i - 1)) = (1, 1)$
 $(t - (t - 1), i - (i + 1)) = (1, -1)$

not working.
skew it, making $D' = H \cdot D \geq 0$.
get figure 4.

Overlapping Tiling

- Increases computation cost
- Eliminates the dependence between tiles along the horizontal direction
- All processes can start executing in parallel



Split Tiling

- Does all of computations which require only elements in the tile
- Communicates
- Does the rest of computations

