Loop Fusion and Tiling

B629

11/17/2010

Temporal & Spatial Locality Recap

Loop Fusion

- Consider following F90 example:

```
A(1:N) = C(1:N) + D(1:N)
B(1:N) = C(1:N) - D(1:N)
```

Loop Fusion

- Consider following F90 example:

```
A(1:N) = C(1:N) + D(1:N)
B(1:N) = C(1:N) - D(1:N)
```

- Each statement uses identical sections of C and D

Loop Fusion

- Consider following F90 example:

```
A(1:N) = C(1:N) + D(1:N)
B(1:N) = C(1:N) - D(1:N)
```

- Each statement uses identical sections of C and D
- What happens after scalarization?

Loop Fusion

- What happens after scalarization?

```
DO I = 1, N
     A(I) = C(I) + D(I)
ENDDO
DO I = 1, N
     B(I) = C(I) - D(I)
ENDDO
```

Loop Fusion

- What happens after scalarization?

```
DO I = 1, N
     A(I) = C(I) + D(I)
ENDDO
DO I = 1, N
     B(I) = C(I) - D(I)
ENDDO
```

- No temporal locality if N is large!

Loop Fusion

- *Fusing* loops together will bring references together, enabling reuse:

```
DO I = 1, N
     A(I) = C(I) + D(I)
     B(I) = C(I) - D(I)
ENDDO
```

Loop Fusion

- When is Loop Fusion legal?

Loop Fusion

- When is Loop Fusion legal?
- **Definition:**
  - An loop-independent dependence between statements in two different loops (i.e., from `S1` to `S2`) is ***fusion-preventing*** if fusing the two loops causes the dependence to be carried by the combined loop in the reverse direction (from `S2` to `S1`).

Loop Fusion

- When is Loop Fusion legal?
- **Definition:**
  - An loop-independent dependence between statements in two different loops (i.e., from `S1` to `S2`) is ***fusion-preventing*** if fusing the two loops causes the dependence to be carried by the combined loop in the reverse direction (from `S2` to `S1`).
- Can fuse two loops when no fusion-preventing dependencies between them

Loop Fusion

- When is Loop Fusion legal?
- **Definition:**
  - An loop-independent dependence between statements in two different loops (i.e., from `S1` to `S2`) is ***fusion-preventing*** if fusing the two loops causes the dependence to be carried by the combined loop in the reverse direction (from `S2` to `S1`).
- Can fuse two loops when no fusion-preventing dependencies between them
  - Also desirable to have same bounds

Loop Fusion

- A more complicated example:

```
DO J = 1, N
    DO I = 1, M
       A(I,J) = C(I,J) + D(I,J)
    ENDDO
    DO I = 1, M
       B(I,J) = A(I,J-1) - E(I,J)
    ENDDO
ENDDO
```

Loop Fusion

- First, fuse loops:

```
DO J = 1, N
     DO I = 1, M
        A(I,J) = C(I,J) + D(I,J)
        B(I,J) = A(I,J-1) - E(I,J)
     ENDDO
ENDDO
```

Loop Fusion

- First, fuse loops:

```
DO J = 1, N
     DO I = 1, M
         A(I,J) = C(I,J) + D(I,J)
         B(I,J) = A(I,J-1) - E(I,J)
     ENDDO
ENDDO
```

- Still no reuse if `M` is large!

Loop Fusion

- First, fuse loops:

```
DO J = 1, N
     DO I = 1, M
         A(I,J) = C(I,J) + D(I,J)
         B(I,J) = A(I,J-1) - E(I,J)
     ENDDO
ENDDO
```

- Still no reuse if `M` is large!
- Can we do better?

Loop Fusion

- Yes, by performing loop interchange:

```
DO I = 1, M
     DO J = 1, N
         A(I,J) = C(I,J) + D(I,J)
         B(I,J) = A(I,J-1) - E(I,J)
     ENDDO
ENDDO
```

Loop Fusion

- Yes, by performing loop interchange:

```
DO I = 1, M
    DO J = 1, N
       A(I,J) = C(I,J) + D(I,J)
       B(I,J) = A(I,J-1) - E(I,J)
    ENDDO
ENDDO
```

- Still not optimal
  - `A(I,J)` used after `A(I,J+1)` defined
  - Requires additional register

Loop Fusion

- No loop independent dependencies
  - Can re-order statements in loop body:

```
DO I = 1, M
    DO J = 1, N
        B(I,J) = A(I,J-1) - E(I,J)
        A(I,J) = C(I,J) + D(I,J)
    ENDDO
ENDDO
```

- Now `A(I,J)` can be saved in register for use in next iteration without additional register

Loop Fusion

- Fusion-preventing dependencies cause problems, however:

```
      DO I = 1, M
            DO J = 1, N
S1                   A(J,I) = B(J,I) + 1.0
            ENDDO
            DO J = 1, N
S2                   C(J,I) = A(J+1,I) + 2.0
            ENDDO
      ENDDO
```

Loop Fusion

- Fusion-preventing dependencies cause problems, however:

```
      DO I = 1, M
            DO J = 1, N
S1                   A(J,I) = B(J,I) + 1.0
            ENDDO
            DO J = 1, N
S2                   C(J,I) = A(J+1,I) + 2.0
            ENDDO
      ENDDO
```

- Cannot fuse inner loops directly, due to backward carried anti-dependence

Loop Fusion

- Solution?

Loop Fusion

- Solution?
- Loop *alignment*:

```
        DO I = 1, M
              DO J = 0, N-1
S1                     A(J+1,I) = B(J+1,I) + 1.0
              ENDDO
              DO J = 1, N
S2                     C(J,I) = A(J+1,I) + 2.0
              ENDDO
        ENDDO
```

Loop Fusion

```
      DO I = 1, M
            DO J = 0, N-1
S1                   A(J+1,I) = B(J+1,I) + 1.0
            ENDDO
            DO J = 1, N
S2                   C(J,I) = A(J+1,I) + 2.0
            ENDDO
      ENDDO
```

- But now iteration ranges are no longer aligned

Loop Fusion

- However, can peel single iteration from start of first loop and end of second:

```
     DO I = 1, M
S0      A(1,I) = B(1,I) + 1.0
        DO J = 1, N-1
S1         A(J+1,I) = B(J+1,I) + 1.0
        ENDDO
        DO J = 1, N-1
S2         C(J,I) = A(J+1,I) + 2.0
        ENDDO
S3      C(N,I) = A(N+1,I) + 2.0
     ENDDO
```

Loop Fusion

- Now resulting loops can be fused:

```
       DO I = 1, M
S0         A(1,I) = B(1,I) + 1.0
           DO J = 1, N-1
S1                 A(J+1,I) = B(J+1,I) + 1.0
S2                 C(J,I) = A(J+1,I) + 2.0
           ENDDO
S3         C(N,I) = A(N+1,I) + 2.0
       ENDDO
```

Loop Fusion

- Formalizing loop alignment
  - **Definition** Given a dependence δ that has a source in one loop and a sink in another loop, the *alignment threshold* of the dependence is defined as follows:
    a. If the dependence would be loop independent after the two loops were fused, the alignment threshold is 0.
    b. If the dependence would be forward loop carried after fusion of the loops, the alignment threshold is the negative of the threshold of the resulting carried dependence.
    c. If the dependence is fusion-preventing—that is, the dependence would be backward carried after fusion—the alignment threshold is defined as the threshold of the backward carried dependence.

Loop Fusion

- Alignment threshold example

```
   DO I = 1, N
S1   A(I) = B(I) + 1.0
   ENDDO
   DO I = 1, N
S2   C(I) = A(I+1) + A(I-1)
   ENDDO
```

- 2 Forward dependencies from S1 to S2
- If fused without concern for dependencies, they would become:
  - A forward carried dependence with threshold 1 from S1 to S2, due to ref `A(I-1)` in S2. Thus, corresponding dependence before fusion has alignment threshold of -1.
  - A backward carried anti-dependence from S2 to S1, involving reference `A(I+1)` with threshold 1. Thus, corresponding dependence before fusion has alignment threshold 1.

Loop Fusion

- Once alignment thresholds are known, alignment is straightforward
- Simply align each loop by largest threshold
- Adjust iteration range of source by:
  - Adding amount equal to alignment threshold to each instance of loop index
  - Subtracting amount equal to alignment threshold from upper and lower bounds of iteration range

```
   DO I = 0, N-1
S1   A(I+1) = B(I+1) + 1.0
   ENDDO
   DO I = 1, N
S2   C(I) = A(I+1) + A(I-1)
   ENDDO
```

Loop Fusion

- Once loops are aligned, easy to peel iterations that are not common to all loops

Loop Tiling

- Another technique to improve temporal locality
- Basic idea: *strip-mine-and-interchange*
  - First, strip-mine a loop into two loops:
    - Inner loop that iterates within contiguous strips
    - Outer loop that iterates strip-by-strip
  - Then, interchange by-strip loop to outside of containing loops

Loop Tiling

- Matrix multiply example:

```
DO J = 1, N
    DO K = 1, N
        DO I = 1, N
            C(I,J) = C(I,J) + A(I,K) * B(K,J)
        ENDDO
    ENDDO
ENDDO
```

Loop Tiling

- Matrix multiply example:
  - Strip-mine step

```
DO J = 1, N
    DO K = 1, N
        DO I = 1, N, S
            DO ii = I, MIN(I+S-1,N)
                C(ii,J) = C(ii,J) + A(ii,K) * B
(K,J)
            ENDDO
        ENDDO
    ENDDO
ENDDO
```

Loop Tiling

- Matrix multiply example:
  - Interchange step

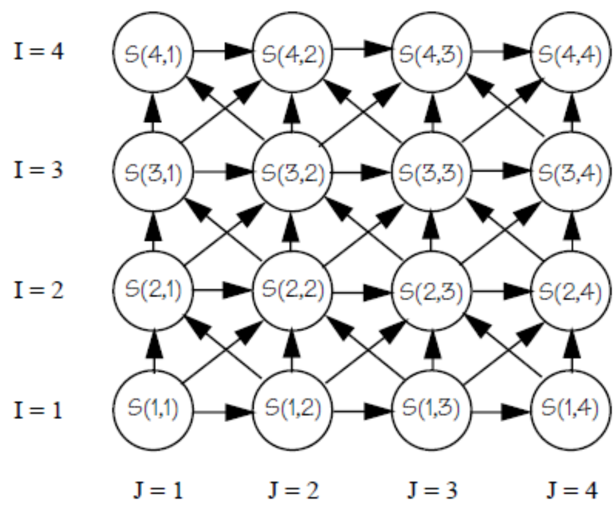
```
DO I = 1, N, S
    DO J = 1, N
        DO K = 1, N
            DO ii = I, MIN(I+S-1,N)
                C(ii,J) = C(ii,J) + A(ii,K) * B
(K,J)
            ENDDO
        ENDDO
    ENDDO
ENDDO
```

Loop Tiling

- Sometimes, simple tiling is not enough:

```
DO I = 1, N
  DO J = 1, M
    A(J+1) = (A(J) + A(J+1))/2
  ENDDO
ENDDO
```

- Dependence pattern:



Loop Tiling

- Dependencies prevent loop interchange after strip-mining

Loop Tiling

- Dependencies prevent loop interchange after strip-mining
- Solution?

Loop Tiling

- Dependencies prevent loop interchange after strip-mining
- Solution?
- *Skew* inner loop, making loop interchange possible:

```
DO I = 1, N
  DO j = I, M+I-1
    A(j-I+2) = (A(j-I+1) + A(j-I+2))/2
  ENDDO
ENDDO
```

Loop Tiling

- Now, strip-mine inner loop:

```
DO I = 1, N
  DO j = I, M+I-1, S
    DO jj = j, MIN(j+S-1,M+I-1)
      A(jj-I+2) = (A(jj-I+1) + A(jj-I+2))/2
    ENDDO
  ENDDO
ENDDO
```

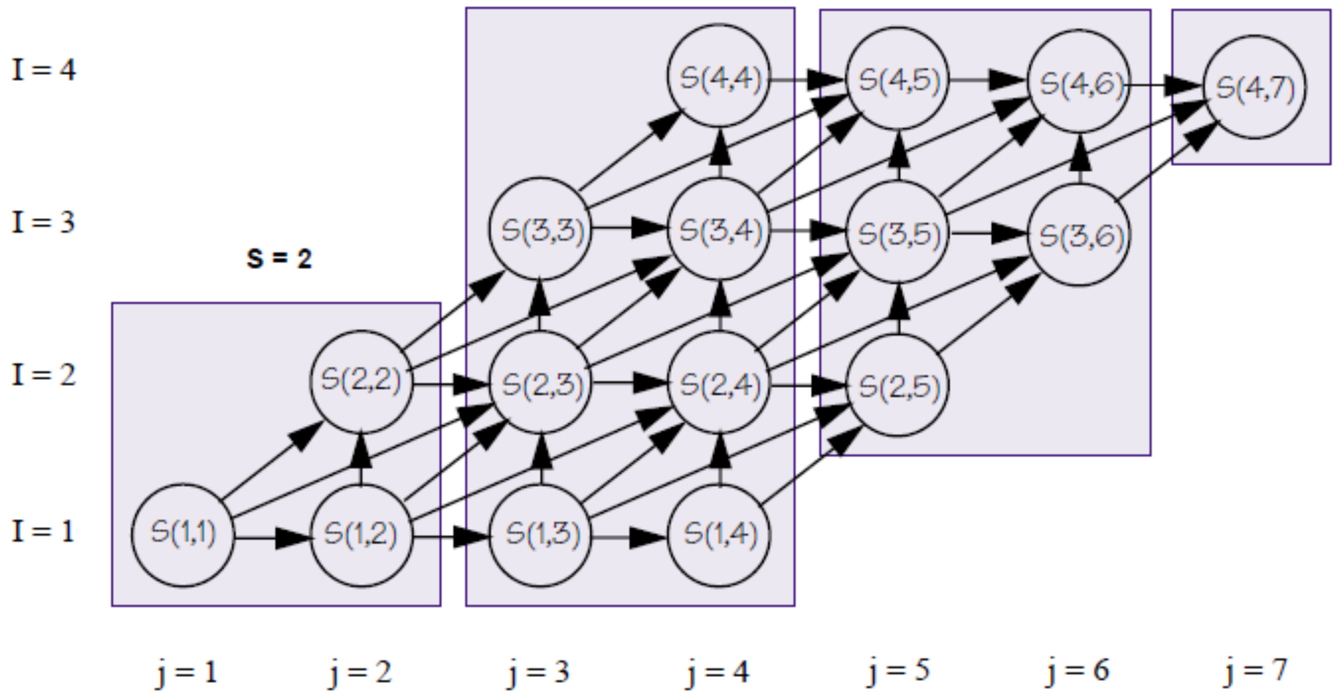
Loop Tiling

- Then interchange by-strip loop outwards:

```
DO j = 1, M+N-1, S
  DO I = MAX(1,j-M+1), MIN(j,N)
    DO jj = j, MIN(j+S-1,M+I-1)
      A(jj-I+2) = (A(jj-I+1) + A(jj-I+2))/2
    ENDDO
  ENDDO
ENDDO
```

Loop Tiling

- Dependence pattern after skewing and tiling:



Loop Tiling

- A real matrix multiply in C++