Loop Transformations for Data Parallelism

November 10, 1994

CS551

Marina Chen

Computer Science Department
Boston University

Transformations in an Optimizing Compiler

- Example: Gaussian Elimination on Multiple Tridiagonal Systems

- Iterative Spatial Loops (ISL)

- ISL After Data Partitioning

- Transformations

- Benefit

Pipelined Gaussian Elimination

```
      subroutine tridiag_2d(a,b,c,y,m,n)
      integer m,n
      real a(m,n),b(m,n),c(m,n),y(m,n)

c     forward elimination

      do i=1,m-1
         b(i,:)= 1.0/b(i,:)
         a(i+1,:)=a(i+1,:)*b(i,:)
         b(i+1,:)=b(i+1,:)-a(i+1,:)*c(i,:)
         y(i+1,:) = y(i+1,:)-a(i+1,:)*y(i,:)
         enddo
      end do
      b(m,:)=1.0/b(m,:)

c     back substitution

      y(m,:) = y(m,:)*b(m,:)
      do i=m-1,1,-1
         y(i,:) = (y(i,:)-c(i,:)*y(i+1,:))*b(i,:)
      end do

      return
      end
```

Multiple Instances

Almost all scientific library routines provide the capability of solving multiple systems because the need arises often in real applications.

Example: Alternating Direction Method (ADM) for 2-D domain, 1-D central difference approximations of partial derivatives result in multiple tridiagonal systems, one for each row (or column).

Pipelined Gaussian Elimination: Problem

1. Embarrassingly parallel over multiple instances (rows).

2. However, the equation axis (columns) may be distributed over processors due to layout inherited from a different scoping unit.

   In other words, this subroutine prefers column distribution but may inherit block distributed arrays as its input.

Pipelined Gaussian Elimination: Solution

Possible Solutions:

1. Convert the layout which induces high communication overhead

2. Retain the same layout but use a sophisticated pipelined algorithm, which is difficult to express in Fortran-90.

Iterative Spatial Loops (ISL)

DO-T $(K = 0, M-1)\,\{$
  DO-S $(I = 0, N_1 - 1)\,\{$
    DOALL-S $(J = 0, N_2 - 1)\,\{$
      $A(I,J) = \eta[A(I,J), A(I+c_1, J+c_2)]$

- Loop Structure:
  - **iterative** loops : DO-T
    loop index does not range over array index domain, 0 or more levels
  - **dependent spatial** loops : DO-S
    loop index ranges over array index domain, with dependence, 0 or more levels
  - **parallel spatial** loops : DOALL-S
    loop index ranges over array index domain, no dependence, 1 or more levels

- References are static or spatial-loop invariant
  c_i is a constant, or a function of K_j, e.g. K_j, 2^{K_j}

Consider ISL after Data Distribution

HPF$ DISTRIBUTE A (BLOCK,BLOCK)

The loop nest becomes

$$
\begin{array}{l}
\text{DO-T } (K = 0, M-1)\,\{ \\
\quad \text{DO-P } (I_1 = 0, P_1 - 1)\,\{ \\
\quad \text{DO-V } (I_2 = 0, V_1 - 1)\,\{ \\
\qquad \text{DOALL-P } (J_1 = 0, P_2 - 1)\,\{ \\
\qquad \text{DOALL-V } (J_2 = 0, V_2 - 1)\,\{ \\
\qquad\quad A(I_1, I_2, J_1, J_2) = \eta[A(I_1, I_2, J_1, J_2), \\
\qquad\qquad\qquad\qquad\qquad\qquad A(I_1 + (I_2 + c_1) \text{ DIV } V_1, \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad (I_2 + c_1) \text{ MOD } V_1, J_1, J_2)]
\end{array}
$$

where

$$
\begin{array}{rcl}
\text{DO-S} & \rightarrow & \left\{ \begin{array}{l} \text{DO-P (serial processor loop)} \\ \text{DO-V} \end{array} \right. \\
\text{DOALL-S} & \rightarrow & \left\{ \begin{array}{l} \text{DOALL-P (parallel processor loop)} \\ \text{DOALL-V (can potentially become parallel)} \end{array} \right.
\end{array}
$$

Class I Loops: DO-P is absent

- DO-S is absent before block distribution

$$
\begin{array}{l}
\text{DO-T } (K = 0, M-1)\,\{ \\
\quad \text{DOALL-S } (I = 0, N_1 - 1)\,\{ \\
\quad \text{DOALL-S } (J = 0, N_2 - 1)\,\{ \\
\qquad A(I,J) = \eta[A(I + 2^K, J)]
\end{array}
$$

$\Rightarrow$

$$
\begin{array}{l}
\text{DO-T } (K = 0, M-1)\,\{ \\
\quad \text{DOALL-P } (I_1 = 0, P_1 - 1)\,\{ \\
\quad \text{DOALL-V } (I_2 = 0, V_1 - 1)\,\{ \\
\qquad \text{DOALL-P } (J_1 = 0, P_2 - 1)\,\{ \\
\qquad \text{DOALL-V } (J_2 = 0, V_2 - 1)\,\{ \\
\qquad\quad A(I_1, I_2, J_1, J_2) = \eta[A(I_1 + (I_2 + 2^K) \text{ DIV } V_1, \\
\qquad\qquad\qquad\qquad\qquad\qquad (I_2 + 2^K) \text{ MOD } V_1, J_1, J_2)]
\end{array}
$$

Or

- DO-S loops are serialized

$$
\begin{array}{l}
\text{DO-T } (K = 0, M-1)\,\{ \\
\quad \text{DO-S } (i = 0, N_1 - 1)\,\{ \\
\quad\quad \text{DOALL-S } (j = 0, N_2 - 1)\,\{ \\
\quad\quad\quad A(I,J) = \eta[A(I + c_1, J)]
\end{array}
$$

$\Rightarrow$

$$
\begin{array}{l}
\text{DO-T } (K = 0, M-1)\,\{ \\
\quad \text{DO-V } (I_2 = 0, N_1 - 1)\,\{ \\
\quad\quad \text{DOALL-P } (J_1 = 0, P_2 - 1)\,\{ \\
\quad\quad \text{DOALL-V } (J_2 = 0, V_2 - 1)\,\{ \\
\quad\quad\quad A(I_2, J_1, J_2) = \eta[A(I_2 + c_1, J_1, J_2)]
\end{array}
$$

Class II Loops: DO-P present

A DO-S loop is distributed over processors:

$$
\begin{array}{l}
\text{DO-T } (K = 0, M-1)\,\{ \\
\quad \text{DO-S } (I = 0, N_1 - 1)\,\{ \\
\quad\quad \text{DOALL-S } (J = 0, N_2 - 1)\,\{ \\
\quad\quad\quad A(I, J) = \eta[A(I + c_1, J)]
\end{array}
$$

$\Rightarrow$

$$
\begin{array}{l}
\text{DO-T } (K = 0, M-1)\,\{ \\
\quad \text{DO-P } (I_1 = 0, P_1 - 1)\,\{ \\
\quad \text{DO-V } (I_2 = 0, V_1 - 1)\,\{ \\
\quad\quad \text{DOALL-P } (J_1 = 0, P_2 - 1)\,\{ \\
\quad\quad \text{DOALL-V } (J_2 = 0, V_2 - 1)\,\{ \\
\quad\quad\quad A(I_1, I_2, J_1, J_2) = \eta[A(I_1 + (I_2 + c_1) \text{ DIV } V_1, \\
\qquad\qquad\qquad\qquad\qquad (I_2 + c_1) \text{ MOD } V_1, J_1, J_2)]
\end{array}
$$

(1) Processor-Memory Skewing

- Increase processor parallelism

Loop Nest 1 (Input loop nest)

$$
\begin{array}{l}
\text{DO-P } (I_1 = 0, P_1 - 1)\,\{ \\
\quad \text{DO-V } (I_2 = 0, V_1 - 1)\,\{ \\
\quad\quad \text{DOALL-P } (J_1 = 0, P_2 - 1)\,\{ \\
\quad\quad \text{DOALL-V } (J_2 = 0, V_2 - 1)\,\{ \\
\quad\quad\quad A(I_1, I_2, J_1, J_2) = \eta[A(I_1 + (I_2 + c_1) \text{ DIV } V_1, \\
\quad\quad\quad\quad\quad\quad (I_2 + c_1) \text{ MOD } V_1, J_1, J_2)]
\end{array}
$$

Loop Nest 2 (Loop interchanging)

$$
\begin{array}{l}
\text{DO-P } (I_1 = 0, P_1 - 1)\,\{ \\
\quad \text{DOALL-V } (J_2 = 0, V_2 - 1)\,\{ \\
\quad\quad \text{DO-V } (I_2 = 0, V_1 - 1)\,\{ \\
\quad\quad\quad \text{DOALL-P } (J_1 = 0, P_2 - 1)\,\{ \\
\quad\quad\quad\quad A(I_1, I_2, J_1, J_2) = \eta[A(I_1 + (I_2 + c_1) \text{ DIV } V_1, \\
\quad\quad\quad\quad\quad\quad\quad (I_2 + c_1) \text{ MOD } V_1, J_1, J_2)]
\end{array}
$$

Loop Nest 3 (Loop skewing)

$$
\begin{array}{l}
\text{DO-V } (\hat{J}_2 = 0, P_1 + V_2 - 2)\,\{ \\
\quad \text{DOALL-P } (\hat{I}_1 = \max(0, \hat{J}_2 - V_2 - 1), \min(\hat{J}_2, P_1 - 1))\,\{ \\
\quad\quad \text{DO-V } (I_2 = 0, V_1 - 1)\,\{ \\
\quad\quad\quad \text{DOALL-P } (J_1 = 0, P_2 - 1)\,\{ \\
\quad\quad\quad\quad A(\hat{I}_1, I_2, J_1, \hat{J}_2 - \hat{I}_1) = \eta[A(\hat{I}_1 + (I_2 + c) \text{ DIV } V, \\
\quad\quad\quad\quad\quad\quad\quad (I_2 + c) \text{ MOD } V, J_1, \hat{J}_2 - \hat{I}_1)]
\end{array}
$$

(2) Hierarchy Split

- Identify iterations of an ISL containing only inter-processor communication vs. those containing both local memory access and communication

Loop Nest 4

DO-T $(K = 0, M - 1)\,\{$
  DOALL-P $(I_1 = 0, P_1 - 1)\,\{$
  DOALL-V $(I_2 = 0, V_1 - 1)\,\{$
    $A(I_1, I_2) =$
    $\eta[A(I_1 + (I_2 + 2^K) \text{ DIV } V_1, (I_2 + 2^K) \text{ MOD } V_1)]$

Loop Nest 5 (After Hierarchy Split)

DO-T $(K = 0, \log(V_1) - 1)\,\{$
  DOALL-P $(I_1 = 0, P_1 - 1)\,\{$
  DOALL-V $(I_2 = 0, V_1 - 1)\,\{$
    $A(I_1, I_2) =$
    $\eta[A(I_1 + (I_2 + 2^K) \text{ DIV } V_1, (I_2 + 2^K) \text{ MOD } V_1)]$

DO-T $(K = \log(V_1), M - 1)\,\{$
  DOALL-P $(I_1 = 0, P_1 - 1)\,\{$
  DOALL-V $(I_2 = 0, V_1 - 1)\,\{$
    $A(I_1, I_2) = \eta[A(I_1 + 2^{K - \log(V_1)}, I_2)]$

(3) Multi-port Communication

- Pipeline communication and exploit multi-port communication

Loop Nest 6 (Input loop nest)

$$
\begin{array}{l}
\text{DO-T } (K = 0, M-1)\,\{ \\
\quad \text{DOALL-P } (I_1 = 0, P-1)\,\{ \\
\quad \text{DOALL-V } (I_2 = 0, V-1)\,\{ \\
\qquad A(I_1, I_2) = \eta[A(I_1 + 2^K, I_2)]
\end{array}
$$

Loop Nest 7 (Loop interchanging)

$$
\begin{array}{l}
\text{DO-T } (K = 0, M-1)\,\{ \\
\quad \text{DOALL-V } (I_2 = 0, V-1)\,\{ \\
\quad \text{DOALL-P } (I_1 = 0, P-1)\,\{ \\
\qquad A(I_1, I_2) = \eta[A(I_1 + 2^K, I_2)]
\end{array}
$$

Loop Nest 8 (Loop skewing)

$$
\begin{array}{l}
\text{DO-T } (\hat{I}_2 = 0, M+V-2)\,\{ \\
\quad \text{DOALL-V } (\hat{K} = \max(0, \hat{I}_2 - V_1 + 1), \min(\hat{I}_2, t-1))\,\{ \\
\quad \text{DOALL-P } (I_1 = 0, P-1)\,\{ \\
\qquad A(I_1, \hat{I}_2 - \hat{K}) = \eta[A(I_1 + 2^{\hat{K} - \log(V)}, \hat{I}_2 - \hat{K})]
\end{array}
$$

(4) Boundary Separation

- Distinguish local memory access and inter-processor communication

Loop Nest 9 (Local stages of Loop Nest 5)

DO-T $(K = 0, \log(V_1) - 1)\,\{$
  DOALL-P $(I_1 = 0, P_1 - 1)\,\{$
  DOALL-V $(I_2 = 0, V_1 - 1)\,\{$
    $A(I_1, I_2) =$
    $\eta[A(I_1 + (I_2 + 2^K)\ \text{DIV}\ V_1, (I_2 + 2^K)\ \text{MOD}\ V_1)]$

Loop Nest 10 (After Boundary Separation)

DO-T $(K = 0, \log(V_1) - 1)\,\{$
  DOALL-P $(I_1 = 0, P_1 - 1)\,\{$
    DOALL-V $(I_2 = 0, V_1 - 2^K - 1)\,\{$
      $A(I_1, I_2) = \eta[A(I_1, I_2 + 2^K)]$
    DOALL-V $(I_2 = V_1 - 2^K, V_1 - 1)\,\{$
      $A(I_1, I_2) =$
      $\eta[A(I_1 + (I_2 + 2^K)\ \text{DIV}\ V_1, (I_2 + 2^K)\ \text{MOD}\ V_1)]$

(5) Increasing Granularity

- Minimize global synchronization overhead

Loop Nest 8

DO-T ($\hat{I}_2 = 0, M + V - 2$) {
  DOALL-V ($\hat{K} = \max(0, \hat{I}_2 - V_1 + 1), \min(\hat{I}_2, t - 1)$) {
  DOALL-P ($I_1 = 0, P - 1$) {
    $A(I_1, \hat{I}_2 - \hat{K}) = \eta[A(I_1 + 2^{\hat{K} - \log(V)}, \hat{I}_2 - \hat{K})]$

Loop Nest 11 (DOALL-P moved outward)

DO-T ($\hat{I}_2 = 0, M + V - 2$) {
  DOALL-P ($I_1 = 0, P - 1$) {
  DOALL-V ($\hat{K} = \max(0, \hat{I}_2 - V_1 + 1), \min(\hat{I}_2, t - 1)$) {
    $A(I_1, \hat{I}_2 - \hat{K}) = \eta[A(I_1 + 2^{\hat{K} - \log(V)}, \hat{I}_2 - \hat{K})]$

Performance Improvement

GE for tridiagonal systems (Class II loop),
CM time in seconds on a 4K CM-2

Eqns × Insts	CMF total (comp+comm)	F90-Y total (comp+comm)	CMSSL
64×64	0.062 (0.004+0.058)	0.020 (0.016+0.004)	0.014
128×128	0.197 (0.008+0.189)	0.029 (0.022+0.007)	0.020
256×256	0.690 (0.016+0.674)	0.055 (0.042+0.014)	0.036
512×512	2.479 (0.032+2.447)	0.130 (0.095+0.035)	0.088
1024×1024	10.97 (0.093+10.88)	0.373 (0.267+0.106)	0.226

Computation worsened by a factor of 3 due to indirect addressing. from pipelined computation (non-uniform starting addresses)

Communication decreased dramatically – only adjacent communication of boundary elements is needed.

Performance gain from avoiding layout conversion and

processor-memory skewing