15-745
Optimizing For Data Locality - 1

Seth Copen Goldstein
Seth@cs.cmu.edu
CMU

Based on "A Data Locality Optimizing Algorithm, Wolf & Lam, PLDI '91

Outline

- The Problem
- Loop Transformations
  - dependence vectors
  - Transformations
  - Unimodular transformations
- Some Linear Algebra

The Issue

- Improve cache reuse in nested loops
- Canonical simple case: Matrix Multiply

for I₁ := 1 to n
  for I₂ := 1 to n
    for I₃ := 1 to n

Tiling solves problem

for I₁ := 1 to n
  for I₂ := 1 to n by s
    for I₃ := 1 to n by s

for II₂ := 1 to n by s
  for II₃ := 1 to n by s
    for I₁ := 1 to n
      for I₂ := II₂ to min(II₂ + s - 1, n)
        for I₃ := II₃ to min(II₃ + s - 1, n)
          C[I₁, I₃] += A[I₁, I₂] * B[I₂, I₃];

In next iteration of I₂ previous data that could be reused has been replaced in cache.
The Problem

• How to increase reuse by transforming loop nest
• Matrix Mult is simple as it is both
  – legal to tile
  – advantageous to tile
• Can we determine both legality and benefit? (reuse vector space)
• Can we transform loop to make it legal? (unimodular transformations)

Loop Transformation Theory

• Iteration Space
• Dependence vectors
• Unimodular transformations

Loop Nests and the Iter space

• General form of tightly nested loop

  for $I_1 := \text{low}_1$ to $\text{high}_1$ by $\text{step}_1$
  for $I_2 := \text{low}_2$ to $\text{high}_2$ by $\text{step}_2$
  ...
  for $I_i := \text{low}_i$ to $\text{high}_i$ by $\text{step}_i$
  ...
  for $I_n := \text{low}_n$ to $\text{high}_n$ by $\text{step}_n$

  Stmts

• The iteration space is a convex polyhedron
  in $\mathbb{Z}^n$ bounded by the loop bounds.
• Each iteration is a node in the polyhedron
  identified by its vector: $p=(p_1, p_2, \ldots, p_n)$

Lexicographical Ordering

• Iterations are executed in
  lexicographic order.

• for $p=(p_1, p_2, \ldots, p_n)$ and $q=(q_1, q_2, \ldots, q_n)$
  if $p \succ_k q$ iff for $1 \leq k \leq n$,

  \[ \forall 1 \leq i < k, (p_i = q_i) \text{ and } p_k > q_k \]

• For MM:
  - $(1,1,1), (1,1,2), (1,1,3), \ldots$,
    $(1,2,1), (1,2,2), (1,2,3), \ldots$,
    \ldots,
    $(2,1,1), (2,1,2), (2,1,3), \ldots$,
  - $(1,2,1) \succ_2 (1,1,2), (2,1,1) \succ_1 (1,4,2), \text{ etc.}$
### Iteration Space

Every iteration generates a point in an n-dimensional space, where n is the depth of the loop nest.

```plaintext
for (i=0; i<n; i++) {
    ...
}
for (i=0; i<n; i++)
    for (j=0; j<4; j++) {
        ...
    }
```

### Dependence Vectors

- Dependence vector in an n-nested loop is denoted as a vector: \(d=(d_1, d_2, \ldots, d_n)\).
- Each \(d_i\) is a possibly infinite range of ints in \([d_i^{\min}, d_i^{\max}]\), where \(d_i^{\min} \in \mathbb{Z} \cup \{-\infty\}, d_i^{\max} \in \mathbb{Z} \cup \{\infty\}\) and \(d_i^{\min} \leq d_i^{\max}\)
- So, a single dep vector represents a set of distance vectors.
- A distance vector defines a distance in the iteration space.
- A dependence vector is a distance vector if each \(d_i\) is a singleton.

### Other defs

- Common ranges in dependence vectors
  - \([1, \infty]\) as \(+\) or >
  - \([-\infty, -1]\) as - or <
  - \([-\infty, \infty]\) as \(\pm\) or *

- A distance vector is the difference between the target and source iterations (for a dependent ref), e.g., \(d = I_t - I_s\)

### Examples

```plaintext
for I_1 := 1 to n
    for I_2 := 1 to n
        for I_3 := 1 to n
            C[I_1, I_3] += A[I_1, I_2] * B[I_2, I_3]

for I_1 := 0 to 5
    for I_2 := 0 to 6
```

\[ D = \{(0,1), (1,0), (1,-1)\} \]
Loop indep and dep data deps

- A loop independent data dependence has a dependence vector of $?$
- A loop dependent data dependence has a dependence vector of $?$

Plausible Dependence vectors

- A dependence vector is plausible iff it is lexicographically non-negative.
- All sequential programs have plausible dependence vectors. Why?
  - Plausible: $(1,-1)$
  - implausible $(-1,0)$

Loop Transforms

- A loop transformation changes the order in which iterations in the iteration space are visited.
- For example, Loop Interchange

Unimodular Transforms

- Interchange
  permute nesting order
- Reversal
  reverse order of iterations
- Skewing
  scale iterations by an outer loop index
Interchange

- Change order of loops
- For some permutation \( p \) of \( 1 \ldots n \)
  
  
  \[
  \text{for } I_1 := \ldots \\
  \text{for } I_2 := \ldots \\
  \vdots \\
  \text{for } I_n := \ldots \\
  \text{body}
  \]
  
  \[
  \text{for } I_{p(1)} := \ldots \\
  \text{for } I_{p(2)} := \ldots \\
  \vdots \\
  \text{for } I_{p(n)} := \ldots \\
  \text{body}
  \]

- When is this legal?

Transform and matrix notation

- If dependences are vectors in iteration space, then transforms can be represented as matrix transforms
- E.g., for a 2-deep loop, interchange is:

\[
T = \begin{bmatrix}
0 & 1 \\
1 & 0
\end{bmatrix} \begin{bmatrix}
p_1 \\
p_2
\end{bmatrix} = \begin{bmatrix}
p_2 \\
p_1
\end{bmatrix}
\]

- Since, \( T \) is a linear transform, \( Td \) is transformed dependence:

\[
\begin{bmatrix}
0 & 1 \\
1 & 0
\end{bmatrix} \begin{bmatrix}
d_1 \\
d_2
\end{bmatrix} = \begin{bmatrix}
d_2 \\
d_1
\end{bmatrix}
\]

Reversal

- Reversal of \( i \)th loop reverses its traversal, so it can be represented as:

- For 2 deep loop, reversal of outermost is:

\[
T = \begin{bmatrix}
-1 & 0 \\
0 & 1
\end{bmatrix} \begin{bmatrix}
p_1 \\
p_2
\end{bmatrix} = \begin{bmatrix}
-p_1 \\
p_2
\end{bmatrix}
\]
Skewing

- Skew loop $I_j$ by a factor $f$ w.r.t. loop $I_i$ maps
  $$(p_1, ..., p_i, ..., p_j, ...) \rightarrow (p_1, ..., p_i, ..., p_j + fp_i, ...)$$

- Example for 2D
  $$T = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$
  $$= \begin{bmatrix} p_1 \\ p_2 + p_1 \end{bmatrix}$$

Loop Skewing Example

- For $I_1 := 0$ to 5
  - for $I_2 := 0$ to 6

Vector Spaces

- $n$ is a point in $n$-space
- $V = \{ v_1, v_2, ..., v_m \}$ is a finite set of $n$-vectors over $m \ \mathbb{R}^n$.
- Linear combination of vectors of $V$ is a vector $x$ as defined by
  $$x = \alpha_1 v_1 + \alpha_2 v_2 + ... + \alpha_m v_m$$
  where $\alpha_i$ are real numbers.
- $V$ is linearly dependent if a combination results in the $0$ vector, otherwise it is linearly independent.
Dim and Basis

• dimensionality of V is dim(V)
  the number of independent vectors in V
• A basis for an m-dimensional vector space is a set of linearly independent vectors such that every point in V can be expressed as a linear comb of the vectors in the basis.
  - The vectors in the basis are called basis vectors

Subspaces and span

• Let V be a set of vectors
• The subspace spanned by V, span(V), is a subset of \( \mathbb{R}^n \) such that
  - \( V \subseteq \text{span}(V) \)
  - \( x, y \in \text{span}(V) \Rightarrow x + y \in \text{span}(V) \)
  - \( x \in \text{span}(V) \) and \( \alpha \in \mathbb{R} \Rightarrow \alpha x \in \text{span}(V) \)

Range, Span, Kernel

• A matrix A can be viewed as a set of column vectors.
• Range \( A^{nxm} \) is \( \{Ax | x \in \mathbb{R}^m\} \)
• \( \text{span}(A) = \text{Range } A^{nxm} \)
• \( \text{nullspace}(A) = \ker(A) = \ker(A^{nxm}) = \{x^m | Ax \in 0\} \)
• \( \text{rank}(A) = \dim(\text{span}(A)) \)
• \( \text{nullity}(A) = \dim(\ker(A)) \)
• \( \text{rank}(A) + \text{nullity}(A) = n, \text{ for } A^{nxm} \)

For next Thur

• Read wolf & Lam
• Tiling
• localized vector space
• reuse and locality
• SRP algorithm
• This Thur: No class
• Next Tue: Project checkpoints