Tiling for Dynamic Scheduling

Ravi Teja Mullapudi
Uday Bondhugula

CSA, Indian Instiute of Science

January 20, 2014
Table of Contents

1 Tiling
   • Introduction
   • Validity

2 Limitations
   • Where do current approaches fail?

3 Directions
   • Localizing Dependences
   • Iterative Tiling
   • Approximate Cycle Check

4 Applications
Table of Contents

1 Tiling
   • Introduction
   • Validity

2 Limitations
   • Where do current approaches fail?

3 Directions
   • Localizing Dependences
   • Iterative Tiling
   • Approximate Cycle Check

4 Applications
Definition (Computation Graph)
A directed acyclic graph in which the nodes represent operations and edges denote dependences between operations.

Definition (Tiling)
Partitioning a computation graph into atomic units of execution.
Iteration spaces, Dependences and Hyperplanes

for (i = 0; i < N; i++){
    for (j = 0; j < N; j++) {
        A[j] = A[j] + B[i]; \( s_1 \)
        if (i == j)
            B[i+1] = A[j]; \( s_2 \)
    }
}

Iteration spaces

\[ D^s = \left\{ \vec{i}_s \mid \vec{i}_s \in \mathbb{Z}^n, A\vec{i}_s + b \geq 0 \right\} \]
for (i = 0; i < N; i++)
    for (j = 0; j < N; j++) {
        A[j] = A[j] + B[i]; \( s_1 \)
        if (i == j)
            B[i+1] = A[j]; \( s_2 \)
    }

Dependences

\[ P_e = \left\{ \langle \vec{i}_s, \vec{i}_t \rangle \mid \vec{i}_s \in D^s, \vec{i}_t \in D^t, \vec{i}_s = h_e(\vec{i}_t), e : s \rightarrow t \right\} \]
for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++) {
        A[j] = A[j] + B[i]; \textcolor{red}{s_1}
        if (i == j)
            B[i+1] = A[j]; \textcolor{red}{s_2}
    }
}

Hyperplanes

\[ \phi_s(\vec{i}_s) = \vec{h} \cdot \vec{i}_s + h_0 \]
for (i = 0; i < N; i++){
    for ( j = 0; j < N; j++) {
        A[j] = A[j] + B[i]; s_1
        if (i == j)
            B[i+1] = A[j]; s_2
    }
}
Validity Conditions

**Non-Negative Dependence Components**

\[ \phi_t(\vec{i}_t) - \phi_s(\vec{i}_s) \geq 0, \langle \vec{i}_s, \vec{i}_t \rangle \in P_e, e : s \rightarrow t \ (HD \geq 0) \]

- Irigoin and Triolet, Lim and Lam, Griebl, Pluto
Validity Conditions

Lexicographically Non-Negative Tile Dependences

$\lfloor HD \rceil \succeq \vec{0}$

- Xue
1 Tiling
   - Introduction
   - Validity

2 Limitations
   - Where do current approaches fail?

3 Directions
   - Localizing Dependences
   - Iterative Tiling
   - Approximate Cycle Check

4 Applications
Conservative Validity Conditions

- Negative dependence components along $\phi^2$ but tiling is valid
- Inter-tile dependences lexicographically negative
- Tile sizes effect validity

\[
\begin{align*}
&\text{for (i = 0; i < N; i++)} \\
&\quad \text{for (j = 0; j < N; j++)} \\
&\quad \quad \text{A}[j] = A[j] + B[i]; \text{ s}_1 \\
&\quad \quad \text{if (i == j)} \\
&\quad \quad \quad B[i+1] = A[j]; \text{ s}_2 \\
&\quad \}
\end{align*}
\]
Conservative Validity Conditions

- Negative dependence components along $\phi^2$ but tiling is valid
- Inter-tile dependences lexicographically negative
- Tile sizes effect validity
Conservative Validity Conditions

- Negative dependence components along $\phi^2$ but tiling is valid
- Inter-tile dependences lexicographically negative
- Tile sizes effect validity
Conservative Validity Conditions

- Negative dependence components along $\phi^2$ but tiling is valid
- Inter-tile dependences lexicographically negative
- Tile sizes effect validity
Tile dependence graph has cycles

Splitting breaks cycles

```c
for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++) {
        if (j > i)
        if (j < i)
    }
}
```
for (i = 0; i < N; i++)
    for (j = 0; j < N; j++) {
        if (j > i)
        if (j < i)
    }

• Tile dependence graph has cycles
• Splitting breaks cycles
Uniformly Tiling Iteration Spaces

- Tile dependence graph has cycles
- Splitting breaks cycles
Not easy to come up with a static schedule for the tiles

Dynamic task scheduler
1 Tiling
   • Introduction
   • Validity

2 Limitations
   • Where do current approaches fail?

3 Directions
   • Localizing Dependences
   • Iterative Tiling
   • Approximate Cycle Check

4 Applications
General Validity Condition

Definition (Valid Tiling)

A set of hyperplanes $\phi^1$, $\phi^2$, $\ldots$, $\phi^k$ with tile size $\tau_i$ for $\phi^i$, is a valid tiling of an iteration space if the dependence graph of $k$-dimensional tiles formed by the hyperplanes with their respective tile sizes is cycle-free.

- Transitive closure of dependence relations
  - Precise computation might be infeasible
  - Expensive operation even for an approximation
- Conservative cycle detection
Theorem

\[ \{\phi^1, \ldots, \phi^k\} \] with tile size \( \tau_i \) for \( \phi^i \) is a valid tiling of an iteration space, if \( \{\phi^1, \ldots, \phi^{k-1}\} \) is a valid tiling and \( \phi^k \) is a valid one-dimensional tiling of each \( k - 1 \) dimensional tile formed by \( \{\phi^1, \ldots, \phi^{k-1}\} \).
Theorem

\{\phi^1, \ldots, \phi^k\} with tile size \tau_i for \phi^i is a valid tiling of an iteration space, if \{\phi^1, \ldots, \phi^{k-1}\} is a valid tiling and \phi^k is a valid one-dimensional tiling of each \(k - 1\) dimensional tile formed by \{\phi^1, \ldots, \phi^{k-1}\}.
Localizing Dependences

Theorem

\{\phi^1, \ldots, \phi^k\} \text{ with tile size } \tau_i \text{ for } \phi^i \text{ is a valid tiling of an iteration space, if } \\
\{\phi^1, \ldots, \phi^{k-1}\} \text{ is a valid tiling and } \phi^k \text{ is a valid one-dimensional tiling of } \\
each k-1 \text{ dimensional tile formed by } \{\phi^1, \ldots, \phi^{k-1}\}.
Theorem

\{φ^1, \ldots, φ^k\} with tile size τ_i for φ^i is a valid tiling of an iteration space, if
\{φ^1, \ldots, φ^{k-1}\} is a valid tiling and φ^k is a valid one-dimensional tiling of
each k − 1 dimensional tile formed by \{φ^1, \ldots, φ^{k-1}\}.

\begin{center}
\includegraphics[width=\textwidth]{diagram.png}
\end{center}
Theorem

\{\phi^1, \ldots, \phi^k\} with tile size \(\tau_i\) for \(\phi^i\) is a valid tiling of an iteration space, if \(\{\phi^1, \ldots, \phi^{k-1}\}\) is a valid tiling and \(\phi^k\) is a valid one-dimensional tiling of each \(k-1\) dimensional tile formed by \(\{\phi^1, \ldots, \phi^{k-1}\}\).
Localizing Dependences

Dependences in the transformed space

\[
\left\{ \langle \langle T_1^s, \ldots, T_k^s, \vec{i}_s \rangle, \langle T_1^t, \ldots, T_k^t, \vec{i}_t \rangle \rangle \mid \langle \vec{i}_s, \vec{i}_t \rangle \in P_e, e : s \to t, \right.

1 \leq l \leq k, \tau_l \cdot T_s^l \leq \phi_s^l(\vec{i}_s) \leq \tau_l \cdot (T_s^l + 1) - 1,

\left. \tau_l \cdot T_t^l \leq \phi_t^l(\vec{i}_t) \leq \tau_i \cdot (T_t^l + 1) - 1 \right\}
\]

Inter-Tile Dependences

\( P_e \) denotes the inter-tile dependence polyhedron between \( k \)-dimensional tiles formed by \( \langle \phi^1, \ldots, \phi^k \rangle \) due to the dependence edge \( e \).
Localizing Dependences

Dependences in the transformed space

\[
\left\{ \left\langle \langle T^1_s, \ldots, T^k_s, \vec{i}_s \rangle, \langle T^1_t, \ldots, T^k_t, \vec{i}_t \rangle \right\rangle \mid \langle \vec{i}_s, \vec{i}_t \rangle \in P_e, e : s \to t, \right. \\
1 \leq l \leq k, \tau_l \ast T^l_s \leq \phi^l_s(\vec{i}_s) \leq \tau_l \ast (T^l_s + 1) - 1, \\
\tau_l \ast T^l_t \leq \phi^l_t(\vec{i}_t) \leq \tau_l \ast (T^l_t + 1) - 1 \right\}
\]

Inter-Tile Dependences

\(P_e\) denotes the inter-tile dependence polyhedron between \(k\)-dimensional tiles formed by \(\langle \phi^1, \ldots, \phi^k \rangle\) due to the dependence edge \(e\).

- Computed by projecting out dimensions inner to the tiling dimension \(\phi^k\) (for both source and target statements)
Restricted Tile Dependence

$Q^k_e$ is a subset of $P^k_e$ restricted to the same $k - 1$ dimensional tile defined by the $k - 1$ tiling hyperplanes outer to $\phi^k$. 
$Q^k_e$ is a subset of $P^k_e$ restricted to the same $k - 1$ dimensional tile defined by the $k - 1$ tiling hyperplanes outer to $\phi^k$. 
Localizing Dependences

**Restricted Tile Dependence**

\( Q^k_e \) is a subset of \( P^k_e \) restricted to the same \( k - 1 \) dimensional tile defined by the \( k - 1 \) tiling hyperplanes outer to \( \phi^k \).

- \( Q^k_e \) thus captures dependences between only those \( k \)-dimensional tiles which are in the same \( k - 1 \) dimensional tile formed by \( \langle \phi^1, \ldots, \phi^{k-1} \rangle \).

If \( e \) is from statement \( s \) to statement \( t \), then:

\[
Q^k_e = P^k_e \land \left( \bigwedge_{1 \leq l \leq k-1} T^l_s = T^l_t \right)
\]
Improved Iterative Tiling

1. while $\tau_k > 1$ do

   // 1. Check validity of tiling at level $k$.
   
   for each $e \in E$ do
   
   // $Q^k_e$ for dependence $e: s \rightarrow t$.
   
   // $C$ is the set of tiles that might be in a cycle
   
   $C = \text{CycleCheck}(Q^k_e$ for each edge $e \in E)$
   
   if $C = \emptyset$ then
   
   // Tiling is valid, move to next level
   
   break
   
   // 2. Attempt to correct tiling
   
   $\tau_k = \lfloor \tau_k / 2 \rfloor$
while $\tau_k > 1$ do

// 1. Check validity of tiling at level $k$.
for each $e \in E$ do
    Compute $Q^k_e$ for dependence $e: s \rightarrow t$.
// $C$ is the set of tiles that might be in a cycle
$C = \text{CycleCheck}(Q^k_e$ for each edge $e \in E)$
if $C = \emptyset$ then
    // Tiling is valid, move to next level
    break
// 2. Attempt to correct tiling
$\tau_k = \lfloor \tau_k / 2 \rfloor$
while $\tau_k > 1$ do

// 1. Check validity of tiling at level $k$. 
for each $e \in E$ do
    Compute $Q^k_e$ for dependence $e: s \rightarrow t$. 
// $C$ is the set of tiles that might be in a cycle 
$C = \text{CycleCheck}(Q^k_e \text{ for each edge } e \in E)$ 
if $C = \emptyset$ then
    // Tiling is valid, move to next level 
    break 
// 2. Attempt to correct tiling 
$\tau_k = \lfloor \tau_k / 2 \rfloor$
1 while $\tau_k > 1$ do

// 1. Check validity of tiling at level $k$.
for each $e \in E$ do
  Compute $Q^k_e$ for dependence $e: s \rightarrow t$.
// $C$ is the set of tiles that might be in a cycle
$C = \text{CycleCheck}(Q^k_e$ for each edge $e \in E$)
if $C = \emptyset$ then
  // Tiling is valid, move to next level
  break
// 2. Attempt to correct tiling
$\tau_k = \lfloor \tau_k / 2 \rfloor$
while $\tau_k > 1$ do
  // 1. Check validity of tiling at level $k$.
  for each $e \in E$ do
    Compute $Q^{k}_e$ for dependence $e: s \rightarrow t$.
    // $C$ is the set of tiles that might be in a cycle
    $C = \text{CycleCheck}(Q^{k}_e$ for each edge $e \in E)$
    if $C = \emptyset$ then
      // Tiling is valid, move to next level
      break
  // 2. Attempt to correct tiling
  $\tau_k = \lfloor \tau_k / 2 \rfloor$
Approximate Cycle Check

Dependences On a Line

Each hyperplane $\phi^k$ gives a one-dimensional coordinate for a tile which can be used to map a tile to a point on a line.

- Forward Dependences
- Backward Dependences
- When are there no cycles?
Approximate Cycle Check

### Dependences On a Line

Each hyperplane $\phi_k$ gives a one-dimensional coordinate for a tile which can be used to map a tile to a point on a line.

- **Forward Dependences**
- **Backward Dependences**
- When are there no cycles?
Approximate Cycle Check

Dependences On a Line

Each hyperplane $\phi^k$ gives a one-dimensional coordinate for a tile which can be used to map a tile to a point on a line.

- Forward Dependences
- Backward Dependences
- When are there no cycles?
Approximate Cycle Check

Backward Violation

\[ B_{e,e'}^k = \left\{ \langle T_1^s, \ldots, T_k^s \rangle \mid \exists T_t^l, \exists T_s^l, \exists T_t^l', (T_k^s \geq T_t^k + 1) \land \\
( T_k^s' \leq T_t^k' - 1 ) \land _{1 \leq l \leq k} (T_t^l = T_s^l) \land \\
\langle \langle T_1^s, \ldots, T_k^s \rangle, \langle T_1^t, \ldots, T_k^t \rangle \rangle \in Q_{e,e}^k, e : s \rightarrow t \\
\langle \langle T_1^s', \ldots, T_k^s' \rangle, \langle T_1^t', \ldots, T_k^t' \rangle \rangle \in Q_{e,e'}^k, e' : s' \rightarrow t' \right\} \]
Approximate Cycle Check

Forward Violation

\[ F_{e, e'}^k = \left\{ \langle T_s^1, \ldots, T_s^k \rangle | \exists T_t^l, \exists T_s^{l'}, \exists T_{t'}^l, (T_s^k \leq T_t^k - 1) \land \\
(T_s^{l'} \geq T_{t'}^k + 1) \land_{1 \leq l \leq k} (T_{t'}^l = T_s^{l}) \land \\
\langle \langle T_s^1, \ldots, T_s^k \rangle, \langle T_t^1, \ldots, T_t^k \rangle \rangle \in Q_{e, e:s \rightarrow t}^k \\
\langle \langle T_s^{l'}, \ldots, T_s^{l'} \rangle, \langle T_{t'}^1, \ldots, T_{t'}^k \rangle \rangle \in Q_{e', e':s' \rightarrow t'}^k \right\} \]
Approximate Cycle Check

**Input**: Tile dependence polyhedra at level $k$ $Q^k_e$ for all $e \in E$ the set of edges in the GDG.

**Output**: Set of tiles that might be part of a cycle

- $B^k$ set of tiles that satisfy $B$ at level $k$.
- $F^k$ set of tiles that satisfy $F$ at level $k$.

$B^k = \emptyset$, $F^k = \emptyset$

for each pair $\langle Q^k_e, Q^k_{e'} \rangle$ $e, e' \in E$ // $e$ can be equal to $e'$
do

- Compute $B^k_{e,e'}$, $F^k_{e,e'}$ using $Q^k_e$ and $Q^k_{e'}$

- $B^k = B^k_{e,e'} \cup B^k$

- $F^k = F^k_{e,e'} \cup F^k$

return $B^k \cup F^k$
Table of Contents

1. Tiling
   - Introduction
   - Validity

2. Limitations
   - Where do current approaches fail?

3. Directions
   - Localizing Dependences
   - Iterative Tiling
   - Approximate Cycle Check

4. Applications
Using additional arrays $R$ and $C$ to remove false dependences in the All-Pairs Shortest-Paths kernel
- All-Pairs Shortest-Paths kernel after removing spurious writes
for (i = N-1; i >= 0; i--) {
    for (j = i+1; j < N; j++) {
        for (k = 0; k < j-i; k++) {
            S[i][j] = MAX(S[i][k+i] + S[k+i+1][j], S[i][j]);
        }
        S[i][j] = MAX(S[i][j], S[i+1][j-1] +
                      can_pair(RNA[i],RNA[j]));
    }
}
for (i = 0; i < N; i++) {
    for (j = 0; j < i+1; j++) {
        A[i] = A[i] + A[i-j]; s_1  
    }
}

- Merging tiles
- Use commutative properties
Experimental Evaluation

(a) floyd – seq time is 231s

(b) zuker – single thread time is 253s

- Experimental setup is a four socket machine with an AMD Opteron 6136 (2.4 GHz, 128 KB L1, 512 KB L2, 6 MB L3 cache) in each socket.
Conclusions and Future Work

- Using improved validity constraints for hyperplane search
- Integrating splitting techniques
- More accurate cycle detection
Thank You!