Class Information

REMINDEERS

• Homework 8 has been posted; due: Tuesday, December 8 (last lecture); no late submission.

• Final exam: Thursday, December 17, 4:00pm - 7:00pm, BRR 1095; accumulative, closed books, closed notes. [cs211], [cs336]

Make-up (tentative): Thursday, December 17, noon - 3:00pm, Tillett 258 (Livingston Campus)

CONFLICTS?

• Project 3 will be posted today. Deadline: Thursday, December 10, at 11:59pm.
Dependence Analysis

Question

Do two variable references never/maybe/always access the same memory location?

Benefits

• improves alias analysis
• enables loop transformations

Motivation

• classic optimizations
• instruction scheduling
• data locality (register/cache reuse)
• vectorization, parallelization

Obstacles

• array references
• pointer references
Vectorization vs. Parallelization

**vectorization** — Find parallelism in innermost loops; fine–grain parallelism

**parallelization** — Find parallelism in outermost loops; coarse–grain parallelism

- Parallelization is considered more complex than vectorization, since finding coarse–grain parallelism requires more analysis (e.g., interprocedural analysis).

- Automatic vectorizers have been very successful
A **loop-independent** dependence exists regardless of the loop structure. The source and sink of the dependence occur on the same loop iteration.

A **loop-carried** dependence is induced by the iterations of a loop. The source and sink of the dependence occur on different loop iterations.

*Loop-carried dependences can inhibit parallelization and loop transformations*
Dependence Testing

Given

\[
\begin{align*}
\text{do} \ i_1 &= L_1, U_1 \\
\quad &\ldots \\
\text{do} \ i_n &= L_n, U_n \\
S_1 &= A(f_1(i_1, \ldots, i_n), \ldots, f_m(i_1, \ldots, i_n)) = \ldots \\
S_2 &= \ldots = A(g_1(i_1, \ldots, i_n), \ldots, g_m(i_1, \ldots, i_n))
\end{align*}
\]

A dependence between statement \(S_1\) and \(S_2\), denoted \(S_1 \delta S_2\), indicates that \(S_1\), the source, must be executed before \(S_2\), the sink on some iteration of the nest.

Let \(\alpha\) & \(\beta\) be a vector of \(n\) integers within the ranges of the lower and upper bounds of the \(n\) loops.

Does \(\exists \alpha \leq \beta\), s.t.
\[
f_k(\alpha) = g_k(\beta) \quad \forall k, \ 1 \leq k \leq m \ ?
\]
Iteration Space

\begin{align*}
\text{do } & I = 1, 5 \\
\text{do } & J = I, 6 \\
\ldots & \\
\text{enddo} \\
\text{enddo} \\
1 \leq I \leq 5 \\
I \leq J \leq 6
\end{align*}

\begin{itemize}
  \item lexicographical (sequential) order for the above iteration space is
    \begin{align*}
    (1,1), (1,2), \ldots, (1,6) \\
    (2,2), (2,3), \ldots (2,6) \\
    \ldots \\
    (5,5), (5,6)
    \end{align*}
  \item given \( I = (i_1, \ldots i_n) \) and \( I' = (i'_1, \ldots, i'_n) \),
    \( I < I' \) iff 
    \( (i_1, i_2, \ldots i_k) = (i'_1, i'_2, \ldots i'_k) \) \& \( i_{k+1} < i'_{k+1} \)
\end{itemize}
**Distance & Direction Vectors**

<table>
<thead>
<tr>
<th>( S_1 ) ( A(I,J) = A(I,J-1) )</th>
<th>( S_2 ) ( A(I,J) = A(I-1,J-1) )</th>
<th>( S_3 ) ( B(I,J) = B(I-1,J+1) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>do ( I = 1, N )</td>
<td>do ( I = 1, N )</td>
<td>do ( I = 1, N )</td>
</tr>
<tr>
<td>do ( J = 1, N )</td>
<td>do ( J = 1, N )</td>
<td></td>
</tr>
<tr>
<td>( \delta S_1 )</td>
<td>( \delta S_2 )</td>
<td>( \delta S_3 )</td>
</tr>
</tbody>
</table>

**Distance Vector** = number of iterations between accesses to the same location

**Direction Vector** = direction in iteration space (=, <, >)

\[
\begin{align*}
S_1 \delta S_1 \\
S_2 \delta S_2 \\
S_3 \delta S_3 \\
\end{align*}
\]
Which Loops are Parallel?

\[
\begin{align*}
&\text{do } I = 1, N \\
&\quad \text{do } J = 1, N \\
&\quad S_1 \ A(I,J) = A(I,J-1)
\end{align*}
\]

\[
\begin{align*}
&\text{do } I = 1, N \\
&\quad \text{do } J = 1, N \\
&\quad S_2 \ A(I,J) = A(I-1,J-1)
\end{align*}
\]

\[
\begin{align*}
&\text{do } I = 1, N \\
&\quad \text{do } J = 1, N \\
&\quad S_3 \ B(I,J) = B(I-1,J+1)
\end{align*}
\]

- a dependence \( D = (d_1, \ldots, d_k) \) is carried at level \( i \), if \( d_i \) is the first nonzero element of the distance/direction vector

- a loop \( l_i \) is parallel, if \( \forall \) a dependence \( D_j \) carried at level \( i \)

<table>
<thead>
<tr>
<th>distance vector</th>
<th>direction vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \forall D_j ) ( d_1, \ldots, d_{i-1} &gt; 0 )</td>
<td>( d_1, \ldots, d_{i-1} = &quot;&lt;&quot; )</td>
</tr>
<tr>
<td>OR ( d_1, \ldots, d_i = 0 )</td>
<td>( d_1, \ldots, d_i = &quot;=&quot; )</td>
</tr>
</tbody>
</table>
Project and OpenMP

Two important issues while specifying the parallel execution of a `for` loops:

- **safety** – parallel execution has to preserve all dependences
- **profitability** – benefits of parallel execution have to compensate for the overhead penalty
Sample code:

```c
#pragma omp parallel for private(i, hash)
    for (j = 0; j < num_hf; j++) {
        for (i = 0; i < wl_size; i++) {
            hash = hf[j] (get_word(wl, i));
            hash %= bv_size;
            bv[hash] = 1;
        }
    }
```

This specifies:

- outermost (j-loop) is parallel
- each thread will get its own copy of variables i and hash, eliminating loop carried anti and output dependences.
Project and OpenMP

[project and OpenMP]

Sample code:

```c
#define CHUNK_SIZE 2
int chunk = CHUNK_SIZE
#pragma omp parallel for \
    schedule (dynamic, chunk) \
    private(i, hash)
for (j = 0; j < num_hf; j++) {
    for (i = 0; i < wl_size; i++) {
        hash = hf[j] (get_word(wl, i));
        hash %= bv_size;
        bv[hash] = 1; }
}
```

This specifies:

- outermost (j-loop) is parallel, with CHUNK_SIZE iterations scheduled as a group; default chunk size=1
- three basic scheduling strategies: static, dynamic, or guided

There are many more options of specifying how to execute for loops in parallel (see the online OpenMP tutorial)
Next Lecture

Things to do:

• Dependence testing strategies: SIV and ZIV
• More on automatic vectorization
• Work on homework 8
• Work on project 3