Improving Locality Using Loop and Data Transformations in an Integrated Framework

Mahmut Kandemir
Alok Choudhary
Jagannathan Ramanujam
Prith Banerjee

---

1The work of Kandemir and Choudhary was supported in part by NSF Young Investigator Award CCR-9357840, NSF grant CCR-9509143 and Air Force Materials Command under contract F30602-97-C-0026. Ramanujam was supported in part by NSF Young Investigator Award CCR-9457768. Banerjee was supported in part by DARPA under contract F30602-98-2-0144 and by the NSF grant CCR-9526325.

Center for Parallel and Distributed Computing
Outline

- Hardware platform
- Previous work on data locality
- Integrated approach
- Performance results
- Extensions and ongoing work
Center for Parallel and Distributed Computing
R10000 Processor

- Clocked at 195 MHz
- MIPS IV Instruction set
- 5 independent fully-pipelined functional units
- 4 way super-scalar
  - Up to 2 integer instruction per cycle
  - Up to 2 floating-point instruction per cycle
  - Up to 1 load/store per cycle
- Dynamic scheduling (out-of-order execution)
- Peak performance 390 MFLOPS
Data and Instruction Caches

- L1 on-chip
  - 32 KB FP cache, 32 KB Instruction/Integer cache
  - 2-way set-associative
  - 32-byte cache line

- L2 off-chip, unified
  - 4 MB secondary cache, Write-back
  - 2-way set-associative
  - Cache line 128-bytes
  - 3.2 GB/sec possible
Memory Accesses

- Levels in the memory hierarchy
  - L1
  - L2
  - Local memory
  - Non-local memory
- Latency: 1:11:61:150
- Similar in HP Convex Exemplar: 3:90:281
- Latency increases from level to level, data locality is vital
Data Reuse and Data Locality #1

- Data reuse
  - Temporal reuse: the same data item is referenced
  - Spatial reuse: a nearby data item is referenced
- Data locality
  - Capturing data reuse in cache
Data Reuse and Data Locality #2

\[
\begin{align*}
\text{do } & \quad i = li, ui \\
& \quad \text{do } j = lj, uj \\
\ldots & = U(i) + V(j) + W(i,j) + X(j,i) + Y(i+j,j)
\end{align*}
\]

- Column-major layouts
- \(U\) has temporal locality
- \(V, X\) have spatial locality
- \(W, Y\) have no locality
Previous work on Data Locality and Drawbacks

- Loop transformations (e.g., [Wolf and Lam 91]):
  - Constrained by data dependences
  - Not very successful for imperfectly nested loops
  - Need satisfy all arrays in the nest
- Data transformations (e.g., [O’Boyle and Knijnenburg 97]):
  - Cannot improve temporal locality
  - Impact is global
- Unified transformations (e.g., [Cierniak and Li 95]):
  - Loop + Data transforms
  - Search space is very large
  - Limitations on types of transforms
Loop and Data Transformations

\[ \text{do } i = li, ui \]
\[ \text{do } j = lj, uj \]
\[ \ldots = U(i,j) + V(i,j) \quad \text{loop transform is OK} \]

\[ \text{do } i = li, ui \]
\[ \text{do } j = lj, uj \]
\[ \ldots = U(i,j) + V(j,i) \quad \text{data transform is required} \]

\[ \text{do } i = li, ui \]
\[ \text{do } j = lj, uj \]
\[ \ldots = U(i,j) + U(j,i) \quad \text{loop+data transform is required} \]
Integrated Approach #1

• Questions
  • How will loop and data transforms be integrated?
  • What types of transforms are allowed?

• Observations
  • Loop transforms can improve spatial and temporal locality
  • Data transforms can improve only spatial reuse

• Idea
  • Use loop transforms for temporal locality
  • Use data transforms for spatial locality
Integrated Approach #2

- A loop nest accessing arrays \( A_1, A_2, \ldots, A_k \)
- Find a loop transform that exploit temporal locality for max # of arrays
- Divide arrays into two groups:
  - \( A_1, \ldots, A_s \) have temporal locality
  - \( A_{s+1}, \ldots, A_k \) have no temporal locality
- Use data transforms to optimize spatial locality for \( A_{s+1}, \ldots, A_k \)
- Result
  - \( A_1, \ldots, A_s \) have temporal locality
  - \( A_{s+1}, \ldots, A_k \) have spatial locality
Integrated Approach #3

What about the remaining nests?

- A loop nest accessing arrays $A_1, A_2, ..., A_k$
- Let $F$ be a subset of arrays whose layouts are fixed
- Find a loop transform such that
  - Temporal locality for max # of arrays
  - The layouts for $A_j \in F$ are satisfied
- Use data transforms to optimize spatial locality for the remaining arrays
Integrated Approach #4

What type of transforms?

- Loop transforms: General non-singular transformations ([Li 93])
- Data transforms: General non-singular transformations ([O’Boyle and Knijnenburg 97])

What else?

- Apply tiling to further improve locality
- Apply register optimizations
Experimental Setup

- Single node of SGI Origin 2000
- 16 loop nests
- cc compiler, -O2 optimization level
- 5 different versions
  - CM Column-major layouts for all arrays
  - RM Row-major layouts for all arrays
  - LP Loop transforms only
  - DT Data transforms only
  - UN Integrated (Unified) approach
# Experimental Suite

<table>
<thead>
<tr>
<th>Nest #</th>
<th>Code</th>
<th>Source</th>
<th>Total Ref</th>
<th>Tempref</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>hydro2d/T1.1</td>
<td>Spec92</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>hydro2d/fct.7</td>
<td>Spec92</td>
<td>14</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>vpenta.6</td>
<td>Spec92</td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>emit.4</td>
<td>Spec92</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>btrix.4</td>
<td>Spec92</td>
<td>12</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>bakvec.2</td>
<td>Eispack</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>htribk.2</td>
<td>Eispack</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>qzhes.7</td>
<td>Eispack</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>fnorm.1</td>
<td>Odepack</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>adi.2</td>
<td>Livermore</td>
<td>33</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>mxm.2</td>
<td>Spec92</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>rhf_hessian.1</td>
<td>Nwchem</td>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>13</td>
<td>transpose.1</td>
<td>Nwchem</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>gfunp.4</td>
<td>Hompack</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>r1mpyq.1</td>
<td>Minpack</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>16</td>
<td>rhf_hessian.2</td>
<td>Nwchem</td>
<td>12</td>
<td>4</td>
</tr>
</tbody>
</table>
Experimental Results #1

Center for Parallel and Distributed Computing
Center for Parallel and Distributed Computing
## Summary of Results

<table>
<thead>
<tr>
<th>Version</th>
<th>Nest #</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16</td>
</tr>
<tr>
<td>CM</td>
<td>×</td>
</tr>
<tr>
<td>RM</td>
<td></td>
</tr>
<tr>
<td>LP</td>
<td>×</td>
</tr>
<tr>
<td>DT</td>
<td>× × ×</td>
</tr>
<tr>
<td>UN</td>
<td>× × × × × × × × × × × × ×</td>
</tr>
</tbody>
</table>

Center for Parallel and Distributed Computing
Summary of Results

<table>
<thead>
<tr>
<th>Version</th>
<th>Program #</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>CM</td>
<td>×</td>
</tr>
<tr>
<td>RM</td>
<td></td>
</tr>
<tr>
<td>LP</td>
<td>×</td>
</tr>
<tr>
<td>DT</td>
<td>×</td>
</tr>
<tr>
<td>UN</td>
<td>×</td>
</tr>
</tbody>
</table>

Center for Parallel and Distributed Computing
Extensions and Ongoing Work

- Dynamic layout transforms
- Inter-procedural analysis - reshaped arrays
- Extensions to irregular array accesses
- Using ILP for determining optimal memory layouts