Improving the Locality of the Sparse Matrix-Vector Product on Shared Memory Multiprocessors

J. C. Pichel  D. B. Heras  J. C. Cabaleiro  F. F. Rivera

Grupo de Arquitectura de Computadores
Universidade de Santiago de Compostela
Introduction

Sparse Matrix-Vector Product (SpMxV)

- Poor data locality in irregular accesses
- Specially important on SMPs

Several algorithms for evaluating and optimizing data locality

Code restructuring techniques: blocking, strip-mining, …

Reorganization of the data

- Our technique is effective for matrices with any structure
- Other reordering techniques: Cuthill-McKee Algorithm, Minimum-Degree based heuristics (AMD,…)

© 2004 J. C. Pichel
Data Locality for the SpMxV (1)

**Locality properties**

High spatial or temporal locality in the accesses → Reuse of data in a particular level of the memory hierarchy

\[
\begin{align*}
\text{DO } & I=0, \ldots, N-1 \\
& \text{REG} = 0.0 \\
& \text{DO } J=\text{PTR}(I), \text{PTR}(I+1)-1 \\
& \quad \text{REG} = \text{REG} + DA(J) \times X(\text{Index}(J)) \\
& \text{ENDDO} \\
& Y(I) = \text{REG} \\
& \text{ENDDO} \\
& \text{ENDDO}
\end{align*}
\]

DA, Index, PTR

Spatial Locality

Spatial and Temporal Locality

Spatial and Temporal Locality are not ensured → Matrix pattern dependent

© 2004 J. C. Pichel
Data Locality for the SpMxV (II)

- SMP case:
  - Add some directives to specify data partitioning
  - Each processor computes the product of only some consecutive rows of the original matrix

  - Closer grouping of nonzero elements within a particular row  \( \rightarrow \)  \textit{Spatial locality}
  - Closer grouping of nonzero elements between two or more consecutive rows  \( \rightarrow \)  \textit{Temporal locality}

\( Y \) updated by only one processor  \( \rightarrow \)  No invalidations
Modelling the locality

Briefly outline of the model:

- Locality is measured over consecutive pairs of rows and columns
- Model based on two locality parameters:
  - Entry matches ($a_{\text{elems}}$)
  - Block matches ($a_{\text{block}}$)

We have defined: **Distance between rows $x$ and $y$**

\[
\begin{align*}
  d_1(x,y) &= \max(n_{\text{elems}}(x), n_{\text{elems}}(y)) - a_{\text{elems}}(x,y) \\
  d_2(x,y) &= n_{\text{blocks}}(x) + n_{\text{blocks}}(y) - 2 \times a_{\text{blocks}}(x,y)
\end{align*}
\]

\[
D_i = \sum d_j(i, i+1), \quad j = 1, 2
\]
Data locality Improvement (I)

- Search for the permutation of rows and columns that will produce an increase in the locality properties in the corresponding processor.
- Decrease in the value of $D_1$ or $D_2$

- NP-complete problem (analogy with the TSP)
  - Minimum Spanning tree (Prim’s algorithm).
  - Depth first search
  - Greedy heuristic – Nearest Neighbour algorithm
Data locality Improvement (II)

Prim’s algorithm and Nearest Neighbour

Strong dependence with the pattern of the matrix

New heuristic based on Lin-Kernighan algorithm

- Minimum Spanning tree (Prim’s algorithm). Depth first search
- Greedy heuristic – Nearest Neighbour algorithm
- Lin-Kernighan heuristic
Reordering matrices for SMPs

The process of ordering data is performed in two steps:

1. Reordering over the whole matrix $\rightarrow$ Good load balance

2. Particular locality improvement is applied over the portion of the matrix accessed by each processor $\rightarrow$ Locality increase
Test set Matrices

Matrices

<table>
<thead>
<tr>
<th>Matrices</th>
<th>Name</th>
<th>N</th>
<th>N_z</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>MSC10848</td>
<td>10848</td>
<td>1229778</td>
</tr>
<tr>
<td>$M_2$</td>
<td>NC5</td>
<td>19652</td>
<td>1499816</td>
</tr>
<tr>
<td>$M_3$</td>
<td>BCSSTK30</td>
<td>23000</td>
<td>1608696</td>
</tr>
<tr>
<td>$M_4$</td>
<td>LI</td>
<td>22695</td>
<td>1350309</td>
</tr>
<tr>
<td>$M_5$</td>
<td>Synthetic Matrix</td>
<td>12000</td>
<td>1436806</td>
</tr>
</tbody>
</table>
Results (I)

- Simulated cache model:
  - Two levels:  
    - L1 → 32 KBytes, line size 4 words
    - L2 → 256 KBytes, line size 8 words
  - Least Recently Used (LRU) replacement algorithm
  - Two way associative cache configuration
  - No prefetching

- Heuristics applied:

  $P_i \rightarrow$ Solution obtained using *Prim's algorithm*

  $P_in \rightarrow$ *Nearest Neighbour algorithm*

  $LK_i \rightarrow$ *Lin-Kernighan algorithm*

  $CM \rightarrow$ *Cuthill-McKee*

  $AMD \rightarrow$ *Approximate Minimum Degree*

  $ND \rightarrow$ *Nested Dissection*
Comparison between different orderings using three threads

L1 Cache

Cache misses improvement for the most costly thread with respect to the original matrix
## Results (III)

*Load balance using three threads*

<table>
<thead>
<tr>
<th>Matrices</th>
<th>Original</th>
<th>LK$_1$</th>
<th>LK$_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>0.58</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td>$M_2$</td>
<td>0.96</td>
<td>0.99</td>
<td>0.99</td>
</tr>
<tr>
<td>$M_3$</td>
<td>0.85</td>
<td>0.99</td>
<td>0.99</td>
</tr>
<tr>
<td>$M_4$</td>
<td>0.57</td>
<td>0.75</td>
<td>0.77</td>
</tr>
<tr>
<td>$M_5$</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
</tr>
</tbody>
</table>

Ratio between the number of entries assigned to the processor with the lowest load and the number of entries of the most loaded one

© 2004 J. C. Pichel
Validating the model on a real SMP:

**SGI Origin 200 system**

- **4 MIPS R10000** at 225 MHz
- Two-level cache hierarchy:
  - L1 \( \rightarrow \) 32 KBytes, line size 32 bytes
  - L2 \( \rightarrow \) 2 MBytes, line size 128 bytes
- Two-way set associative
- LRU replacement policy
Results

(V)

L1 cache  L2 cache

M1

M2

M3

M4

M5

© 2004 J. C. Pichel
Conclusions

- A methodology for characterizing and increasing locality of sparse algebra codes on SMPs was introduced.
- The problem of locality improvement has been solved using an analogy with the TSP.
- On applying this methodology significant decreases in the number of cache misses is obtained (specially using $LK$).
- The algorithms developed are competitive comparing to standard ordering algorithms, and also present more stability.
Improving the Locality of the Sparse Matrix-Vector Product on Shared Memory Multiprocessors

J. C. Pichel  D. B. Heras  J. C. Cabaleiro  F. F. Rivera

Grupo de Arquitectura de Computadores
Universidade de Santiago de Compostela