From AAPC to Random Permuters and Sorters

Thomas M. Stricker and Jonathan C. Hardwick

School of Computer Science
Carnegie Mellon University
Pittsburgh PA 15213

Talk presented by
Thomas Gross

SPAA96, Padova, Veneto, Italia
June 25, 1996
Outline

AAPC: All to All Personalized Communication

• Congestion controlled AAPC routing
  • Quantify benefits and practical aspects (T3D)
• Memory operations in element permutations
  • Quantify difference between AAPC of blocks and true permutation of elements
• Sorting performance
  • Derive performance from an architectural model instead of measured implementation
AAPC Algorithms

• Conventional AAPC
  • Independent loop through all destinations
  • Router resolves congestion dynamically
  • Performance of AAPC: average over routes

• Phased AAPC
  • Synchronized loop through $k$ phases/patterns
  • Congestion resolved statically
  • Performance of AAPC: slowest route determines phase; after that average over phases
Target Architecture

Cray T3D with 512 nodes at the PSC:

• Topology 8x4x8x2 torus

• Non adaptive, dimension order routing on a 3-dimensional torus

• Short messages: 32 bytes pay load

• Fast links (2*75 MB/s per link raw speed)
AAPC in Hell
(default routing - no phases)

Router: plain msg passing

average
31 MB/s
AAPC in Purgatory:
(phased routing with simple 3-dimensional patterns)

Throughput distribution over 262k routes

Throughput distribution over 512 phases

average 41 MB/s
AAPC in Heaven:
(congestion free - after modified pattern and router)

Throughput distribution
over 262k routes

Throughput distribution
over 512 phases

Router: final

55.5MB/s
## Performance Impact of Congestion Controlled AAPC Routing

### Balanced AAPC on 512 node T3D

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>MBytes/s per node</th>
<th>MByte/s aggregate</th>
<th>Percent of Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>No phases</td>
<td>31.1</td>
<td>15,945</td>
<td>41%</td>
</tr>
<tr>
<td>Phased</td>
<td>41.4</td>
<td>21,200</td>
<td>55%</td>
</tr>
<tr>
<td>Pattern fixed</td>
<td>46.7</td>
<td>23,910</td>
<td>62%</td>
</tr>
<tr>
<td>Router fixed</td>
<td>54.4</td>
<td>27,852</td>
<td>73%</td>
</tr>
<tr>
<td>Final</td>
<td>55.5</td>
<td>28,416</td>
<td>74%</td>
</tr>
<tr>
<td>Hardware</td>
<td>75.0</td>
<td>38,400</td>
<td>100%</td>
</tr>
</tbody>
</table>
Options for Element Permutations

- **Buffer-packing transfers**
  - Pack buffer (gather)
  - Transfer data across network, AAPC of blocks
  - Unpack buffer (scatter)

- **Chained transfers**
  - Pack - transfer - unpack

---

*On a Cray T3D:*

- Buffer packing: 14 MB/s - Chained: 32 MB/s

*Trade-offs described in Stricker/Gross ISCA95*
Element Permutations (B=Perm[A])
(routed on a message passing machine)
## Performance of Element Permutations

(including memory operations)

<table>
<thead>
<tr>
<th>AAPC type</th>
<th>Model</th>
<th>Measured on 512 node T3D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MBytes/s per proc.</td>
<td>MBytes/s per proc.</td>
</tr>
<tr>
<td>Blocks</td>
<td>69</td>
<td>55.5</td>
</tr>
<tr>
<td>Transpose</td>
<td>38</td>
<td>29.5</td>
</tr>
<tr>
<td>Random Permute</td>
<td>32</td>
<td>22.0</td>
</tr>
</tbody>
</table>
Motivation for Modelling Steps of Sorters

• Recent algorithmic work:
  • Bucket work vs. key work [Blelloch, Zagha]
  • Balanced transposes vs. unbalanced permutes [Bader, Helman, Jájá]

• Previous implementation work:
  • Measured implementations
  • Indirect connection to node architecture

• Copy transfer model:
  • Analytic model
  • Direct connection to architectural features
## Modelling Radix Sort Performance

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Model work</th>
<th>Model MByte/s</th>
<th>Measured MByte/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bucket Counting</td>
<td>key</td>
<td>21</td>
<td>14</td>
</tr>
<tr>
<td>Local Permutation</td>
<td>key</td>
<td>21</td>
<td>8</td>
</tr>
<tr>
<td>Global Permutation</td>
<td>key</td>
<td>32</td>
<td>20</td>
</tr>
<tr>
<td>Bucket Transpose</td>
<td>bucket</td>
<td>38</td>
<td>20</td>
</tr>
<tr>
<td>Bucket Reduction</td>
<td>bucket</td>
<td>220</td>
<td>125</td>
</tr>
<tr>
<td>Bucket Scan</td>
<td>bucket</td>
<td>90</td>
<td>60</td>
</tr>
<tr>
<td>Bucket Untranspose</td>
<td>bucket</td>
<td>38</td>
<td>20</td>
</tr>
</tbody>
</table>

**Model:** 7.88 MB/s per processor  
**Measured:** 3.99 MB/s per processor, 2042 MB/s total, 255 million keys/s (16bit key, 48bit data)
Conclusions

• Difficult to achieve hardware performance on AAPC routing of blocks.
  • Increase from 41% of nominal peak to 74%
  • Hiding net topology details is not a good idea!

• Cost of memory operations in permutations.
  • Element permutations at 29% of peak
  • Memory operations are significant!

• Estimated sorting performance with copy transfer model
  • Local memory operations are dominant costs!
Simple Radix Sorter

• for (i=0; i<radix_steps; i++)
  1: B=bucket_counts(A)
  2: transpose(B)
  3: B’=prefix_scan(B)
  4: transpose(B’)
  5: A’=presort_copy(A,B’)
  6: A=grouped_permute(A’)

Bucket work: steps 2,3,4
Key work: steps 1,5,6

Analysis of sample sort:
Future work!
Index counting:

Bucket[Radix(a)]++

Local memory operation
Patterns

Ref: Hinrichs et al. SPAA94
Grouped Permutation (B=Perm*[A])
(prearranged for a message passing machine)
AAPC: Algorithm - Router Interaction

- Service node
- Conflict link
- Unused link