Fast Query Processing Using Cooperative Caching for Index Structures

Xiaoqin Ma and Gene Cooperman
Email: xqma, gene@ccs.neu.edu
College of Computer and Information Science
Northeastern University

Cooperative Caching for n-ary Tree Lookup

- The Problem:
  - There is a static n-ary tree and a huge number of key lookups
  - The n-ary tree can fit in memory but not in cache
  - High Throughput and Fast Response Time

Method A
- One By One Search:

Method B: Buffering Accesses
- Requires >>10^6 keys for efficient batch processing
Method C: Cooperative Caching

- Requires \(>> 10^4\) keys for efficient batch processing.

Design Methods

- **A**: dominated by memory latency

- **B**: dominated by memory bandwidth (33%) and CPU speed (67%)
  - need large batches to amortize the cost to load the subtree into the L2 cache.
  - larger batches give long delay and slow response.

- **C**: dominated by network bandwidth (5%) and CPU speed (95%)
  - Moderated batch to efficiently use network bandwidth and amortize the network latency.
  - Faster response.

Today's Hardware:
Boston Univ. Linux cluster (Mariner)

- 4-ary tree size: 3.2MB with 8 levels
- Pentium III
- Myrinet network measured vs. ideal bandwidth: 138MB/s vs. 275MB/s
- PC-133 DRAM measured vs. ideal bandwidth: 647MB/s vs. 1.06GB/s
- Memory latency: 60ns (measured)
- Cache miss penalty: 110ns (measured)
- L2 cache size: 512KB

Experimental Results

4-ary tree size: 3.2MB
L2 cache size: 512KB
Number of CPUs used: 10
Size of Batch/Msg: 40KB storing 10K keys

<table>
<thead>
<tr>
<th>Method</th>
<th>75ns</th>
<th>76ns</th>
<th>75ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Method B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Method C</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Comparison

\[ A \times B + C = D \]

Future Trends

- Memory latency will not change
- Memory Bandwidth will increase moderately
- CPU speed and network bandwidth will increase dramatically

Predicted Future Results

- Memory latency: 50ns
- Memory bandwidth: 4GB/s
- Network bandwidth: 10Gb/s
- CPU speed: 10GHz

<table>
<thead>
<tr>
<th>Method</th>
<th>A Method</th>
<th>B Method</th>
<th>C Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>40ns</td>
<td>10ns</td>
<td>5ns</td>
<td></td>
</tr>
</tbody>
</table>

Experimental Results:

<table>
<thead>
<tr>
<th>Method</th>
<th>A Method</th>
<th>B Method</th>
<th>C Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>95.5ns</td>
<td>76ns</td>
<td>75ns</td>
<td></td>
</tr>
</tbody>
</table>

Predicted Future Results:

<table>
<thead>
<tr>
<th>Method</th>
<th>A Method</th>
<th>B Method</th>
<th>C Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>40ns</td>
<td>10ns</td>
<td>5ns</td>
<td></td>
</tr>
</tbody>
</table>