NUMA-Aware Data Structure Design & Benchmarking

Presented by Yifan Kang
Mentored by Shangdi Yu and Prof. Julian Shun
What is NUMA

- Non-Uniform Memory Access
- What is Uniform Memory Access?
Uniform Memory Access

All the processors have the same access to all memory.
Non-Uniform Memory Access

Each processor has its own memory controller.
Parallel Programming Backgrounds

- Split a problem into smaller tasks
- Execute them in different processors *concurrently*
- Perform tasks more *efficiently*
Parallel Programming Backgrounds

\[ T_p = \text{runtime with } p \text{ processors} \]
\[ T_1 = \text{work} \]
\[ T_\infty = \text{span} \]

Brent’s Law:

\[ T_p \leq T_\infty + \frac{T_1 - T_\infty}{p} \]
Example: Parallel prefix sum

Given an array $A_0, A_1, \ldots, A_{n-1}$, the prefix sum array $S$ defined as:

$$S_i = \sum_{k=0}^{i} A_k$$

<table>
<thead>
<tr>
<th>input</th>
<th>6</th>
<th>4</th>
<th>16</th>
<th>10</th>
<th>16</th>
<th>14</th>
<th>2</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>output</td>
<td>6</td>
<td>10</td>
<td>26</td>
<td>36</td>
<td>52</td>
<td>66</td>
<td>68</td>
<td>76</td>
</tr>
</tbody>
</table>
Example: Parallel prefix sum

Core idea: Divide & Conquer

Step 1: Builds a tree of sums bottom-up
Example: Parallel prefix sum

Core idea: Divide & Conquer

Step 1: Builds a tree of sums bottom-up
Example: Parallel prefix sum

Step 2: Traverses the tree top-down to compute prefixes
kd-tree

data set

kd-tree
Nearest neighbor search

Search for the nearest neighbor of a.

data set

kd-tree
Nearest neighbor search

Step 1: Find the leaf node from the root.
Nearest neighbor search

Step 2: Backtrack to find candidates.
Nearest neighbor search

Step 2: Backtrack to find candidates.
Nearest neighbor search

Step 2: Backtrack to find candidates.

Bounding box of a node
NUMA-aware
kd-tree

How to make kd-tree NUMA-aware?
NUMA-aware kd-tree

1. Split two parts into different NUMA nodes
2. Copy some subtrees in both nodes
Experiment Setup

- We performed our experiments in a virtual NUMA machine, c5.metal (96 vCPUs and 196 GiB), via Amazon Elastic Compute Cloud (Amazon EC2).
- Implemented with Parlaylib
- Random generated datasets
Runtime Comparison
Future Work

- Optimize dynamic kd-trees on NUMA machines.
- Real-world datasets
- Develop more NUMA data-structures such as interval trees and range trees.
Acknowledgements

- Shangdi Yu
- Prof. Julian Shun
- MIT PRIMES
Questions?
Image Sources
