Implementing Faster Associative Containers for GIDs in HPX
John Bell¹, Parsa Amini², Dr. Hartmut Kaiser²
1-Department of Computer Science, Louisiana State University – Shreveport, 2-Center for Computational Technology, Louisiana State University

Introduction
• HPX is a general purpose C++ runtime system for parallel and distributed applications and can run on multiple localities.
• HPX uses associative containers extensively for its Global Identification addresses (GIDs).
• Associative containers hold data by pairs in a key, value relationship.
• Multiple associative containers need to be compared using the most common database operations.
• Narrowed the search to binary searches trees that are or can be implemented and Standard Template Library compatible.

Objective
• Topic: Work with various associative containers and either determine or refine one or several to optimally work with HPX.
• Goal: To select the best performing data structure for AGAS used to lookup information about GIDs.
• Task: Testing data bases along with benchmarking.

Experimental Process
• Created benchmark that concentrates on the main operations preformed on databases.
• Examined our current database and analyzing what operations are being performed the most.
• In the last phase – Identify the best case scenario of databases (which may mean the combining of certain aspects of various databases) and then maybe implement it into HPX.

Results

Benchmark Program
Key Features:
• Uniform data sets (1million – 10 million elements).
• Measured time for each tasks:
  • Insertions
  • Searches
  • Deletions

Search Trees

<table>
<thead>
<tr>
<th>Tree Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-3</td>
<td>Bx</td>
</tr>
<tr>
<td>2-3-4</td>
<td>(Optimal) Binary Search</td>
</tr>
<tr>
<td>AA</td>
<td>Dancing</td>
</tr>
<tr>
<td>(a,b)</td>
<td>H Tree</td>
</tr>
<tr>
<td>AVL</td>
<td>Interval</td>
</tr>
<tr>
<td>B+</td>
<td>Order Statistic</td>
</tr>
<tr>
<td>B</td>
<td>(Left-leaning) Red Black</td>
</tr>
<tr>
<td>B*</td>
<td>Scapegoat</td>
</tr>
</tbody>
</table>

Table displaying identified search trees

Revised Search Trees

<table>
<thead>
<tr>
<th>Tree Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-3</td>
<td>AVL</td>
</tr>
<tr>
<td>B</td>
<td>(Optimal) Binary Search</td>
</tr>
</tbody>
</table>

Table displaying search trees implemented into C++

Analysis & Conclusion
• Examined several data structures using a benchmark program. Concentrating on the main operations preformed on databases.
• Six were implemented six and still being evaluated in each category.
• In the last phase – Identify the best case scenario of databases (which may mean the combining of certain aspects of various databases) and then maybe implement it into HPX.

Next Steps
• Revisit the list and maybe implement more search trees.
• Use the benchmark program to evaluate other programs.
• Create new data sets for better evaluation.

Acknowledgments
• This material is based upon work supported by the National Science Foundation with additional support from the Center for Computation & Technology at Louisiana State University.
• National Science Foundation under award OCI-1560410.