Implementation of a Binary Tree Driver (OAKc) in Cactus

Jeff DeReus
Center for Computation & Technology
Louisiana State University
Introduction

- Applications
  - N-body problems
  - Computational Fluid Dynamics

- Difficulties
  - The use of parallel processors

- Galaxy formation modeling
  - $10^{11}$ stars $== 10^9$ years for one iteration using $N^2$ algorithms
  - 1 year using $N \log_2 N$ algorithms
Binary Trees

- Ordered binary trees vs. binary heaps
- Efficiency due to ability to cut half of the tree with each comparison
- Gains in parallel performance
  - Several independent trees of lesser height (degree)
  - Decreased RAM requirements
Cactus Code Framework

- http://www.cactuscode.org
- Open source development designed primarily for scientists and engineers
- The “Flesh” - no true functionality
- The “Thorns” - actual computation
- How does it work?
  - Configuration files passed through Perl interpreter
# Standard Thorns

<table>
<thead>
<tr>
<th>Computational Toolkit</th>
<th>Standard capabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDF5 Toolkit</td>
<td>I/O using HDF5 data format</td>
</tr>
<tr>
<td>PETSc Toolkit</td>
<td>Elliptic equation solvers</td>
</tr>
<tr>
<td>Web Browser Toolkit</td>
<td>Web steering of computations</td>
</tr>
<tr>
<td>CFD Toolkit</td>
<td>Solving problems in CFD</td>
</tr>
</tbody>
</table>
Design Methodology

Considerations

• User specifies ranges and types of data to be analyzed
  – Modification of algorithms to accommodate

• Added complexity
  – Unknown number of data points until runtime
  – Different processors might require different search criteria
Prospective Applications

• Adaptive Mesh Refinement
  – Equations as function parameters
  – Recursive crawling over extracted subtrees
  – Dynamic memory allows for node storage of results of computations
  – Nodes as distinct individual 3-dimensional meshes

• Galaxy formation modeling
  – Extracted trees ported to new processors
  – “Real” vs. “Virtual” trees
Prospective Applications

• Causal set problems
  – Requires implementation of posets
    • Knowledge of a nodes’ descendants and ancestors
    • As the number of ancestors increases the amount of data stored grows rapidly

• Computational Fluid Dynamics
  – Fast multipole methods require a binary tree structure
Future Directions

• Dynamic division of the tree based on initial organization and heirarchy
  – No longer need arrays for initial input
  – Individual data structures populated directly into tree structure

• Extension to graph structures
  – No single “root” to anchor tree
  – Allows for analysis of unstructured mesh problems

• Results in a dynamic data structure able to model real time changes
Future Directions

• Detailed benchmarking
  – Reflect parallel vs. serial performance differences
  – Performance differences of “Real” vs. “Virtual” trees

• Optimization of code structure
  – Search functions modified to allow for one function to do all types of searches

• Modification of stream parsing functions
  – Missing values
  – Auto detection of data types
Acknowledgements

• First and foremost, Yaakoub Y El Khamra
  – supervisor and mentor, Frameworks at CCT
• Gabrielle Allen
  – Assistant Director for Computing Applications
• Tom Goodale
  – Chief Architect of the Cactus Code Framework
• The Computer Science Department along with The Center for Computation & Technology
  – Assistance with funding