Speeding up Trace Compass

Loïc Prieur-Drevon
December 10, 2015

École Polytechnique de Montréal
Laboratoire DORSAL
Agenda

Context
- State System
- State History Tree (SHT)

Approach
- Tools
- Known Bugs

In-progress
- Overlapping SHT

Future Work
State System

Source: State History Tree : an Incremental Disk-based Data Structure for Very Large Interval Data, Alexandre Montplaisir
State System: State History Tree (SHT)

Source: State History Tree: an Incremental Disk-based Data Structure for Very Large Interval Data, Alexandre Montplaisir
Approach

1. State System Tools
2. Know State System Issues
3. Literature Review
4. Work upwards

   From Data Structure
   To Trace Compass UI
SHT Benchmark

- Trace Size
- SHT Size
- SHT Build Time
- Number of Attributes
- Number of Nodes
- Number of Intervals
- Depth

- Node Fill
- Number of NullStateValues
- Quark Locality
- Query :
  - Time
  - Nodes Searched
Visualization

Library: GraphStream

Demo: http://secrataire.dorsal.polymtl.ca/~lprieur/
So that didn't work out...
StdOut

Node #2776, Core Node, 1 children, 0 intervals (0% used), [1424271996923817644 - 1424272000935664941]
Node #2777, Core Node, 2 children, 831 intervals (33% used), [1424271996923817644 - 1424272000935664941]
Node #2742, Core Node, 4 children, 2579 intervals (99% used), [1424271996923817644 - 1424272000935664941]
Node #2748, Core Node, 2 children, 2479 intervals (99% used), [1424271996923817644 - 1424272000935664941]
Node #2751, Core Node, 2 children, 2469 intervals (99% used), [1424271996923817644 - 1424272000935664941]
Node #2743, Core Node, 4 children, 2482 intervals (99% used), [1424271996923817644 - 1424272000935664941]
Node #2754, Core Node, 3 children, 1149 intervals (48% used), [1424271998217152870 - 1424272000935664941]
Node #2641, Core Node, 22 children, 1920 intervals (99% used), [1424271999327073230 - 1424272000921791689]
Node #2593, Core Node, 23 children, 1916 intervals (99% used), [1424271999258453073 - 1424272000851540109]
Node #2571, Core Node, 21 children, 1910 intervals (99% used), [1424271999223964073 - 1424272000813805565]
Node #2095, Core Node, 2 children, 1859 intervals (99% used), [1424271998607523322 - 1424272000217321922]
Node #166, Core Node, 17 children, 2020 intervals (99% used), [1424271996923817644 - 1424272000686438261]
Node #2121, Core Node, 24 children, 333 intervals (14% used), [1424272000154220690 - 1424272000217321922]
Node #2096, Core Node, 24 children, 1921 intervals (99% used), [1424271998607523322 - 1424272000177163707]
Node #167, Core Node, 21 children, 2568 intervals (99% used), [1424271996923817644 - 14242721997920778168]
Node #2468, Core Node, 2 children, 2513 intervals (99% used), [1424271996923817644 - 1424272000935664941]
Node #2221, Core Node, 11 children, 17 intervals (0% used), [1424271998775907724 - 1424272000686438261]
Node #2461, Core Node, 6 children, 482 intervals (25% used), [1424271999125924051 - 1424272000686438261]
Traces with many threads : SHT Structure

Tree depth is proportional to the number of threads!
Traces with many threads : Explanation

```
Start           End
  1
Start           End
  0             2
  Start
  End
```

Start

End

Start

End
Traces with many threads : Modification
# Traces with many threads : Fix

<table>
<thead>
<tr>
<th></th>
<th>Baseline 1.1 (Before)</th>
<th>Overlapping Nodes (After)</th>
<th>Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Build Time (ms)</strong></td>
<td>28 845</td>
<td>12 222</td>
<td>57.6%</td>
</tr>
<tr>
<td><strong>Nodes</strong></td>
<td>23 716</td>
<td>2 788</td>
<td>88.2%</td>
</tr>
<tr>
<td><strong>SHT Size (MiB)</strong></td>
<td>1 486</td>
<td>178</td>
<td>88.0%</td>
</tr>
<tr>
<td><strong>Average Fill</strong></td>
<td>9%</td>
<td>95%</td>
<td>x10.5</td>
</tr>
<tr>
<td><strong>Depth</strong></td>
<td>206</td>
<td>7</td>
<td>96.6%</td>
</tr>
<tr>
<td><strong>Single Query</strong></td>
<td>84 Nodes Searched</td>
<td>41 Nodes Searched</td>
<td>51.2%</td>
</tr>
</tbody>
</table>
Overlapping Nodes - Benefits

- Denser State History Tree
- Faster Build Times
- Faster Queries
- Multithreaded Queries (future work)
- Shallower Tree → Shorter Branches:

Bug 430050 - [LTTng] OutOfMemoryError when opening a trace with a lot of streams
Alternative : Splitting Intervals (Future Work)

Reasoning: long intervals go higher up in tree: depth penalty

- When to split them?
- Which intervals to split?
- How to distinguish split intervals?
- Advantages/Disadvantages
Future Work

- **Removal:**

<table>
<thead>
<tr>
<th></th>
<th>Intervals</th>
<th>Size</th>
<th>Depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant Values</td>
<td>4%</td>
<td>50%</td>
<td>30%</td>
</tr>
<tr>
<td>Null Values</td>
<td>16%</td>
<td>60%</td>
<td>45%</td>
</tr>
</tbody>
</table>

- **Queries:**

Source: https://github.com/giraldeau/inf8601-scratchpad/tree/master/03-pthread-recursive

- **Quark Locality**
- **Attribute Tree**
Questions?
Use Cases?

loic.prieur-drevon@polymtl.ca