Is the Segment Store on Disk Fast Yet?

Progress Report Meeting
December 12, 2016
Loïc Prieur-Drevon    Michel Dagenais
Table of Contents

1 Publications

2 State System Queries

3 SegmentStore on disk
Publications

- **Enhanced State History Tree (eSHT) : a Stateful Data Structure for Analysis of Highly Parallel System Traces**
  Presented at IEEE Big Data Congress in San Francisco

- **R-SHT: a State History Tree with R-Tree Properties for Analysis and Visualization of Highly Parallel System Traces**
  Submitted to ACM TOMPECS
R-SHT results: Single Query Scalability

![Graph showing the relationship between Single Query Time (μs) and Attributes. The x-axis represents the number of attributes ranging from $10^4$ to $10^6$, and the y-axis represents the single query time ranging from $10^1$ to $10^5$ μs. The graph compares different methods: SHT, oSHT, eSHT, R2, R3, R4, and νR. Each method is represented by different markers and line styles. The SHT method shows the highest single query time, while the νR method shows the lowest.](image-url)
Extract data faster

Before: Single and Full query API.

- **Full Query**
  - returns *all* stati at time $t$
  - requires reading many nodes – $\propto$ attributes

- **Single Query**
  - return status of one attribute at time $t$
  - requires reading half as many nodes
Extract data faster

After: 2D query API.

- `@param` range or set of attributes
- `@param` range or set of time stamps
- `@return` `Iterable<StateInterval>`
  - lazy
  - low memory footprint
  - unordered
R-SHT results: 2D Query Scalability

Is the Segment Store on Disk Fast Yet? – Loïc Prieur-Drevon, Michel Dagenais

www.dorsal.polymtl.ca
<table>
<thead>
<tr>
<th>Query Type</th>
<th>PsTree (ms)</th>
<th>Zoom (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>full</td>
<td>5360</td>
<td>29880</td>
</tr>
<tr>
<td>eSHT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>full</td>
<td>6168</td>
<td>30310</td>
</tr>
<tr>
<td>2D</td>
<td>226.8</td>
<td>4408</td>
</tr>
<tr>
<td>vR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>full</td>
<td>17100</td>
<td>40970</td>
</tr>
<tr>
<td>2D</td>
<td>135.7</td>
<td>4041</td>
</tr>
</tbody>
</table>
Previous implementations were based upon `TreeMap` and `ArrayList`. 

*IN MEMORY*

Didn’t scale.
Quark-less oSHT

The State History Tree stores Intervals:
Interval : < key, start, end, value >
Segment : < start, end, value >
Why not reuse the work from the State History Tree instead of starting from scratch?
Issue: Sorting

Many views, analysis required a sorted list of segments. Easy when the Segment Store can fit into memory. But when they don’t?

`ISegmentStore.getIntersectingElements(start, end, order)`
class iterator
\[ \text{time, order} \]

\[ \begin{align*}
\text{nodes} & \leftarrow \text{PriorityQueue(order, rootNode)}; \\
\text{segments} & \leftarrow \text{PriorityQueue(order)};
\end{align*} \]

function next()
\[ \begin{align*}
\text{return segments.remove}();
\end{align*} \]

function hasNext()
\[ \begin{align*}
\text{while segments} & = \emptyset \lor \text{segments.head()} > \text{nodes.head()} \land \\
\text{nodes} & \neq \emptyset \text{ do}
\end{align*} \]

\[ \begin{align*}
\text{node} & \leftarrow \text{tree.readNode(nodes.remove())}; \\
\text{nodes.add(node.intersectingChildren(time))}; \\
\text{segments.add(node.intersectingSegments(time))};
\end{align*} \]

\[ \begin{align*}
\text{return segments} & \neq \emptyset;
\end{align*} \]

end
Is the Segment Store on Disk Fast Yet? – Loïc Prieur-Drevon, Michel Dagenais
Is the Segment Store on Disk Fast Yet? – Loïc Prieur-Drevon, Michel Dagenais
What about custom aspects?

Segments can carry a number of aspects:

- a name
- a value
- both
- anything else...

We cannot index all possible aspects. Build an index on the side?
Before and After on the side Name index

Is the Segment Store on Disk Fast Yet? – Loïc Prieur-Drevon, Michel Dagenais

www.dorsal.polymtl.ca
## Summary of Master’s results

<table>
<thead>
<tr>
<th>Scale</th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size on Disk</td>
<td>$O(\text{Max}(A^2, I))$</td>
<td>$O(I)$</td>
</tr>
<tr>
<td>Depth (D)</td>
<td>$O(\text{Max}(A, \log(I)))$</td>
<td>$O(\log(I))$</td>
</tr>
<tr>
<td>CFE-entryList</td>
<td>$O(W \times D)$</td>
<td>$O(I)$</td>
</tr>
<tr>
<td>CFE-zoom</td>
<td>$O(W \times D)$</td>
<td>$O(\log(W \times I))$</td>
</tr>
<tr>
<td>Mem CFE-SS</td>
<td>$O(W \times D)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>Max SegStore</td>
<td>RAM segments</td>
<td>Disk</td>
</tr>
<tr>
<td>SegStore – RAM</td>
<td></td>
<td>log(segments)</td>
</tr>
</tbody>
</table>

$W = \text{screen width}$  
$I = \text{nb Intervals}$
What’s next?

- Finish Thesis
- Get patches into Trace Compass
- Get a job ∨ Do a PhD

loic.prieur-drevon@polymtl.ca