Brief Announcement

Hyaline: Fast and Transparent Memory Reclamation

Ruslan Nikolaev and Binoy Ravindran
rnikola@vt.edu, binoy@vt.edu
Systems Software Research Group
Virginia Tech, USA
Memory Reclamation Problem

- Concurrent programming is hard
  - Non-blocking (lock-free) data structures require special treatment of deleted memory objects
  - Garbage collectors are often impractical in C/C++
- Desirable properties for memory reclamation
  - *Non-blocking*: protecting non-blocking data structures
  - *Robust*: bound memory usage even when threads are stalled or preempted
  - *Transparent*: avoid implicit assumptions about threads; they can be created/deleted dynamically
Hyaline

- General idea
  - Distributed reference counting, triggered only when deleting objects
  - Maintains multiple global lists of deleted objects
  - Each list is used by a subset of threads
## Comparison

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Performance</th>
<th>Robust</th>
<th>Transparent</th>
<th>Extra Memory</th>
<th>API complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference Counting</td>
<td>Very Slow</td>
<td>Yes</td>
<td>Partially (swap)</td>
<td>Double each pointer</td>
<td>Intrusive</td>
</tr>
<tr>
<td>Hazard Pointers</td>
<td>Slow</td>
<td>Yes</td>
<td>No (deletion)</td>
<td>1 word</td>
<td>Hard</td>
</tr>
<tr>
<td>Epoch Based Reclamation</td>
<td>Fast</td>
<td>No</td>
<td>No (deletion)</td>
<td>1 word</td>
<td>Easy</td>
</tr>
<tr>
<td>Hazard Eras</td>
<td>Fast</td>
<td>Yes</td>
<td>No (deletion)</td>
<td>3 words</td>
<td>Hard</td>
</tr>
<tr>
<td>Interval Based Reclamation</td>
<td>Fast</td>
<td>Yes</td>
<td>No (deletion)</td>
<td>3 words</td>
<td>Medium</td>
</tr>
<tr>
<td>(2GEIBR)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hyaline</td>
<td>Very Fast</td>
<td>No</td>
<td>Yes</td>
<td>3 words</td>
<td>Easy</td>
</tr>
<tr>
<td>Hyaline-1</td>
<td>Very Fast</td>
<td>No</td>
<td>Almost</td>
<td>3 words</td>
<td>Easy</td>
</tr>
<tr>
<td>Hyaline-S</td>
<td>Fast</td>
<td>Yes</td>
<td>Yes</td>
<td>3 words</td>
<td>Medium</td>
</tr>
<tr>
<td>Hyaline-1S</td>
<td>Fast</td>
<td>Yes</td>
<td>Almost</td>
<td>3 words</td>
<td>Medium</td>
</tr>
</tbody>
</table>
**Evaluation**

**Xeon E7-8880 v3 2.30 GHz, 72 cores**

---

**Bonsai Tree**

- Throughput (M ops/sec)

---

**Hash Map**

- Retired Objects per Operation
More details

- Code is open-source and available at:
  - https://github.com/rusnikola/lfsmr
- Full paper is available as an arXiv report:

Thank you!