Ulterior Reference Counting: Fast Garbage Collection without a Long Wait

Stephen M Blackburn and Kathryn S Mckinley

Present by Qi Chen
March 6, 2012

Slides adapted from presentation by Dimitris Prountzos
Outline

- Problem Statement
- Background
  - Reference Counting
- Ulterior Reference Counting (URC)
- URC Implementation
- Evaluation
- Conclusion
Problem Statement

- Throughput/Responsiveness trade-off
  - High throughput: mark-sweep (MS)
  - Short pause time: reference counting (RC)
Problem Statement

- Throughput/Responsiveness trade-off
  - High throughput: mark-sweep (MS)
  - Short pause time: reference counting (RC)

- Any collector can achieve the two goals?
Problem Statement

• Throughput/Responsiveness trade-off
  ◦ High throughput: mark-sweep (MS)
  ◦ Short pause time: reference counting (RC)

• Any collector can achieve the two goals?
  ◦ Ulterior Reference Counting (URC)
Ulterior RC Approach

- Match mechanisms to object demographics
  - **Copying nursery space**
    - Highly mutated, high mortality young objects
    - Ignore nursery pointer mutations
    - GC time proportional to survivors
  - **RC mature space**
    - Low mutation, low mortality old objects
    - GC time proportional to dead objects and pointer mutations

- Generalize deferred RC to heap objects
  - Defer fields of highly mutated objects and enumerate them quickly
  - Reference count only infrequently mutated fields
Background

- Reference Counting
  - Advantage
    - Incremental: the work of garbage detection is spread out over every mutation
  - Disadvantage
    - Unable to reclaim cycles
      - Solution: additional algorithm
    - Tracking every pointer mutation is expensive
      - Solution: Deferral, Buffering, Coalescing
Background

- **RC Formal Definitions**
  - **Mutation event**: $\text{RCM}(p)$
    - $\text{RC}(P_{\text{before}}) --$, $\text{RC}(P_{\text{after}}) ++$
    - May be buffered or performed immediately
  - **Retain event**: $\text{RCR}(p)$
    - Zero count table (ZCT)
    - Generate a temporary increment for $p$
  - **Deferral**
    - No mutation event generates $\text{RCM}(p)$
    - Need a $\text{RCR}(p)$ to preserve objects
Background

- **RC Optimization Mechanism**: to reduce computation overhead
  - **Buffering**
    - apply \( RC(p) \)--, \( RC(p) \)++ later
  - **Coalescing**
    - apply \( RCM(p) \) only for the initial and final values of \( p \)
      (coalesce intermediate values)
      \{\( RCM(p) \), \( RCM(p_1) \), ... \( RCM(p_n) \)\} \( \rightarrow \) \( RC(p_{initial}) \)--, \( RC(p_{final}) \)++
  - **Deferral**
    - Defer RC events.
Ulterior Reference Counting

- Idea: Extends deferral to select heap pointers
  - e.g. pointers from nursery space to mature space
- Deferral is not a fixed property of a pointer
  - e.g. an object can be moved between nursery and mature spaces.
- Integrate Event: RCI(p)
  - Change p from deferred to not-deferred.
Ulterior Reference Counting

- Generalizing Deferral
Ulterior Reference Counting

- A Generational RC Hybrid Collector (BG-RC)
  - Combine a bounded copying nursery with RC.
  - For young objects
    - Bump-pointer allocation
    - Copying collection
  - For old objects
    - Free-list allocation
    - Reference counting collection
Ulterior Reference Counting

- **Nursery phase**
  - Scan roots
  - Process the modified object buffer
  - Reclaim nursery

- **RC phase**
  - Process decrement buffer, recursively decrement
  - Reclaim old objects
  - Cycle detection if needed
Ulterior Reference Counting

- **Write Barrier**
  - Remember pointers into the nursery from the non-nursery spaces. (RC, immortal and boot image spaces)
  - Generate $RCM(p)$ for mutations to pointer fields within the non-nursery spaces.
  - An object remembering coalescing barrier.
private void writeBarrier(VM_Address srcObj, VM_Address srcSlot, VM_Address tgtObj)
    throws VM_PragmaInline {
    if (getLogState(srcObj) != LOGGED)
        writeBarrierSlow(srcObj);
    VM_Magic.setMemoryAddress(srcSlot, tgtObj);
}

private void writeBarrierSlow(VM_Address srcObj)
    throws VM_PragmaNoInline {
    if (attemptToLog(srcObj)) {
        modifiedBuffer.push(srcObj);
        enumeratePointersToDecBuffer(srcObj);  // trade-off for sparsely
        setLogState(srcObj, LOGGED);          // modified objects
    }
}
Ulterior Reference Counting

- Mutation Phase

![Diagram showing RC space and non-RC space with nodes a, b, d, and e connected by arrows, and stacks and registers labeled.obj buf, dec buf, root buf are also labeled.]
Ulterior Reference Counting

- Mutation Phase

![Diagram of Ulterior Reference Counting]

- RC space
- non-RC space
- Stacks
- Regs

- obj buf
- dec buf
- root buf
Ulterior Reference Counting

- Mutation Phase

![Diagram showing RC space and non-RC space with nodes a, b, d, e and connections]

- Stacks Regs

- RC space

- non-RC space

- obj buf

- dec buf

- root buf
Ulterior Reference Counting

- Mutation Phase

![Diagram](image)
Ulterior Reference Counting

- Mutation Phase

![Diagram showing RC space and non-RC space with nodes a, b, d, e, r, s, and their connections.](Image)
Ulterior Reference Counting

- Mutation Phase

![Diagram showing RC and non-RC spaces with connected objects and buffers]

- Stacks
- Regs
- RC space
- non-RC space
- b
- obj buf
- d
- e
- dec buf
- root buf
- a
- b
- d
- e
Ulterior Reference Counting

- Mutation Phase

![Diagram showing RC space and non-RC space with nodes a, b, d, e, r, s, t and connections between them, with labels for obj buf, dec buf, and root buf.]
Ulterior Reference Counting

- Nursery Collection: Scan Roots

Diagram showing the relationships between nodes labeled a, b, d, e, r, s, and t, indicating the RC space and non-RC space with pointers and buffers labeled obj buf, dec buf, and root buf.
Ulterior Reference Counting

- Nursery Collection: Scan Roots
Ulterior Reference Counting

- Nursery Collection: Scan Roots

Diagram:
- RC space
  - Node a
  - Node b
  - Node d
  - Node e
  - Node t
- non-RC space
  - Node r
  - Node s

Buffers:
- obj buf
- dec buf
- root buf
Ulterior Reference Counting

- Nursery Collection: Process Object Buffer
Ulterior Reference Counting

- Nursery Collection: Reclaim Nursery

- RC space

- non-RC space

- obj buf

- dec buf

- root buf
Ulterior Reference Counting

- RC Collection: Process Decrement Buffer
Ulterior Reference Counting

- RC Collection: Recursive Decrement

![Diagram of RC space and non-RC space with nodes labeled a, b, r, s, e, t tied by arrows and free state indicated by an x]

- **RC space**
  - Stack: 1
  - RC Collection: Recursive Decrement
  - Stack: 1
  - RC Collection: Recursive Decrement
  - Stack: 1
  - RC Collection: Recursive Decrement
  - Stack: 1

- **non-RC space**
  - free state

- **Buffers**
  - obj buf
  - dec buf
  - root buf
Ulterior Reference Counting

- RC Collection: Process Decrement Buffer

![Diagram showing RC Collection and Process Decrement Buffer]
Ulterior Reference Counting

- Collection Complete.

![Diagram of Ulterior Reference Counting]

- Stacks
- Regs
- RC space
- non-RC space

- a
- b
- e
- r
- t
- s

- obj buf
- dec buf
- root buf

- ✓
- ✗
Ulterior Reference Counting

- Controlling Pause Times: nursery collection & reference counting times
  - Modest bounded nursery size
  - Limit the growth of meta data
    - Decrement and modified object buffers
    - Trigger a collection if too big
  - RC time cap
    - Limit time recursively decrementing RC obj & in cycle detection

- Cycle detection
  - Use Bacon/Rajan trial deletion algorithm
  - Add a trigger to invoke cycle detection
Evaluation

- Jikes RVM and JMTK
- 4 Collectors
  - MS, RC, BG-MS, BG-RC
- Benchmarks
  - SPEC JVM & pseudojbb
- Collection triggers
  - Each 4MB of allocation for BG-RC (1MB for RC)
  - Time cap of 60ms
  - Cycle detection at 512KB
## Throughput/Pause time

<table>
<thead>
<tr>
<th>benchmark</th>
<th>heap used MB</th>
<th>BG-MS time sec</th>
<th>MS norm time</th>
<th>MS max pause</th>
<th>BG-MS norm time</th>
<th>BG-MS max pause</th>
<th>BG-RC norm time</th>
<th>BG-RC max pause</th>
<th>RC norm time</th>
<th>RC max pause</th>
</tr>
</thead>
<tbody>
<tr>
<td>_202_jess</td>
<td>24</td>
<td>6.2</td>
<td>1.91</td>
<td>182</td>
<td>1.00</td>
<td>181</td>
<td>0.99</td>
<td>44</td>
<td>2.36</td>
<td>131</td>
</tr>
<tr>
<td>_213_javac</td>
<td>68</td>
<td>13.4</td>
<td>1.01</td>
<td>268</td>
<td>1.00</td>
<td>285</td>
<td>1.00</td>
<td>68</td>
<td>1.78</td>
<td>580</td>
</tr>
<tr>
<td>_228_jack</td>
<td>21</td>
<td>7.7</td>
<td>1.52</td>
<td>184</td>
<td>1.00</td>
<td>185</td>
<td>0.94</td>
<td>44</td>
<td>1.66</td>
<td>72</td>
</tr>
<tr>
<td>_205_raytrace</td>
<td>27</td>
<td>7.5</td>
<td>1.31</td>
<td>203</td>
<td>1.00</td>
<td>184</td>
<td>1.03</td>
<td>49</td>
<td>1.71</td>
<td>133</td>
</tr>
<tr>
<td>_227_mtrt</td>
<td>32</td>
<td>8.3</td>
<td>1.29</td>
<td>241</td>
<td>1.00</td>
<td>180</td>
<td>1.04</td>
<td>49</td>
<td>1.75</td>
<td>130</td>
</tr>
<tr>
<td>_201_compress</td>
<td>25</td>
<td>11.6</td>
<td>0.98</td>
<td>160</td>
<td>1.00</td>
<td>175</td>
<td>0.88</td>
<td>68</td>
<td>0.93</td>
<td>72</td>
</tr>
<tr>
<td>pseudojbb</td>
<td>74</td>
<td>20.0</td>
<td>1.00</td>
<td>264</td>
<td>1.00</td>
<td>281</td>
<td>1.00</td>
<td>53</td>
<td>1.33</td>
<td>297</td>
</tr>
<tr>
<td>_209_db</td>
<td>30</td>
<td>19.2</td>
<td>1.01</td>
<td>238</td>
<td>1.00</td>
<td>244</td>
<td>1.01</td>
<td>59</td>
<td>1.11</td>
<td>43</td>
</tr>
<tr>
<td>_222_mpegaudio</td>
<td>18</td>
<td>10.3</td>
<td>1.05</td>
<td>185</td>
<td>1.00</td>
<td>178</td>
<td>0.96</td>
<td>43</td>
<td>1.14</td>
<td>121</td>
</tr>
<tr>
<td><strong>mean</strong></td>
<td><strong>35</strong></td>
<td><strong>11.3</strong></td>
<td><strong>1.23</strong></td>
<td><strong>214</strong></td>
<td><strong>1.00</strong></td>
<td><strong>210</strong></td>
<td><strong>0.98</strong></td>
<td><strong>53</strong></td>
<td><strong>1.53</strong></td>
<td><strong>175</strong></td>
</tr>
<tr>
<td><strong>geometric mean</strong></td>
<td><strong>31</strong></td>
<td><strong>10.4</strong></td>
<td><strong>1.20</strong></td>
<td><strong>211</strong></td>
<td><strong>1.00</strong></td>
<td><strong>206</strong></td>
<td><strong>0.98</strong></td>
<td><strong>52</strong></td>
<td><strong>1.47</strong></td>
<td><strong>130</strong></td>
</tr>
</tbody>
</table>

Table 3: Throughput and Responsiveness of MS, BG-MS, BG-RC, and RC at a Moderate Heap Size
### Throughput/Pause time

<table>
<thead>
<tr>
<th>benchmark</th>
<th>heap used MB</th>
<th>BG-MS time sec</th>
<th>MS norm time</th>
<th>max pause</th>
<th>BG-MS norm time</th>
<th>max pause</th>
<th>BG-RC norm time</th>
<th>max pause</th>
<th>RC norm time</th>
<th>max pause</th>
</tr>
</thead>
<tbody>
<tr>
<td>.202_jess</td>
<td>24</td>
<td>6.2</td>
<td>1.91</td>
<td>182</td>
<td>1.00</td>
<td>181</td>
<td>0.99</td>
<td>44</td>
<td>2.36</td>
<td>131</td>
</tr>
<tr>
<td>.213_javac</td>
<td>68</td>
<td>13.4</td>
<td>1.01</td>
<td>268</td>
<td>1.00</td>
<td>285</td>
<td>1.00</td>
<td>68</td>
<td>1.78</td>
<td>580</td>
</tr>
<tr>
<td>.228_jack</td>
<td>21</td>
<td>7.7</td>
<td>1.52</td>
<td>184</td>
<td>1.00</td>
<td>185</td>
<td>0.94</td>
<td>44</td>
<td>1.66</td>
<td>72</td>
</tr>
<tr>
<td>.205_raytrace</td>
<td>27</td>
<td>7.5</td>
<td>1.31</td>
<td>203</td>
<td>1.00</td>
<td>184</td>
<td>1.03</td>
<td>49</td>
<td>1.71</td>
<td>133</td>
</tr>
<tr>
<td>.227_mtrt</td>
<td>32</td>
<td>8.3</td>
<td>1.29</td>
<td>241</td>
<td>1.00</td>
<td>180</td>
<td>1.04</td>
<td>49</td>
<td>1.75</td>
<td>130</td>
</tr>
<tr>
<td>.201_compress</td>
<td>25</td>
<td>11.6</td>
<td>0.98</td>
<td>160</td>
<td>1.00</td>
<td>175</td>
<td>0.88</td>
<td>68</td>
<td>0.93</td>
<td>72</td>
</tr>
<tr>
<td>pseudojbb</td>
<td>74</td>
<td>20.0</td>
<td>1.00</td>
<td>264</td>
<td>1.00</td>
<td>281</td>
<td>1.00</td>
<td>53</td>
<td>1.33</td>
<td>297</td>
</tr>
<tr>
<td>.209_db</td>
<td>30</td>
<td>19.2</td>
<td>1.01</td>
<td>238</td>
<td>1.00</td>
<td>244</td>
<td>1.01</td>
<td>59</td>
<td>1.11</td>
<td>43</td>
</tr>
<tr>
<td>.222_mpegaudio</td>
<td>18</td>
<td>10.3</td>
<td>1.05</td>
<td>185</td>
<td>1.00</td>
<td>178</td>
<td>0.96</td>
<td>43</td>
<td>1.14</td>
<td>121</td>
</tr>
<tr>
<td><strong>mean</strong></td>
<td><strong>35</strong></td>
<td><strong>11.3</strong></td>
<td><strong>1.23</strong></td>
<td><strong>214</strong></td>
<td><strong>1.00</strong></td>
<td><strong>210</strong></td>
<td><strong>0.98</strong></td>
<td><strong>53</strong></td>
<td><strong>1.53</strong></td>
<td><strong>175</strong></td>
</tr>
<tr>
<td><strong>geometric mean</strong></td>
<td><strong>31</strong></td>
<td><strong>10.4</strong></td>
<td><strong>1.20</strong></td>
<td><strong>211</strong></td>
<td><strong>1.00</strong></td>
<td><strong>206</strong></td>
<td><strong>0.98</strong></td>
<td><strong>52</strong></td>
<td><strong>1.47</strong></td>
<td><strong>130</strong></td>
</tr>
</tbody>
</table>

Table 3: Throughput and Responsiveness of MS, BG-MS, BG-RC, and RC at a Moderate Heap Size
Throughput & Responsiveness

Utilization vs. Window (msec)

- BG-MS
- RC
- BG-RC
Conclusion

- Match allocation and collection policies to the behaviors of older and younger object demographics
- Extend deferral to select heap objects
- Achieve good throughput performance and good responsiveness