SAFE MEMORY RECLAMATION

EPOCH RECLAMATION
BACKGROUND

• As reachability and liveness are delineated, more complex memory management mechanisms are necessary for the safe destruction of objects.

• Object reference counting is the most common solution to this problem where blocking synchronization is permitted.
BACKGROUND
REFERENCE COUNTING

\[ T_0 \] LOGICAL DELETE

\[ T_1 \] READ

\[ T_2 \] READ PHYSICAL DELETE

Time
Reference counting has performance limitations that can affect both scale and fast path latency.

Object reference counting is generally insufficient for concurrent synchronization with linearization guarantees.
INTRODUCTION

- Safe memory reclamation mechanisms have been developed for performance and for correctness.

  Pass-The-Buck

  Hazard Pointers

  Read-Copy-Update

  Proxy Collection

  Time-Based Deferral
INTRODUCTION

• Passive interfaces such as the one provided by RCU and Concurrency Kit allow for versatility across all types of workloads and algorithms.

• Central to passive mechanisms is grace period detection.
GRACE PERIOD DETECTION

- **T₀**: LOGICAL DELETE \(\rightarrow\) SYNCHRONIZE \(\rightarrow\) PHYSICAL DELETE
- **T₁**: READ
- **T₂**: READ

Time
GRACE PERIOD DETECTION

- $T_0$: Logical Delete
- $T_1$: Read followed by Physical Delete
- $T_2$: Read

Time
**GRACE PERIOD DETECTION**

```c
void reader(void)
{
    object_t *object;
    struct node *n;

    for (;;;;) {
        smr_begin();
        object = lookup(something);
        CK_LISTFOREACH(n, my_list, linkage)
            function(n, *object);
        smr_end();
    }
}
```

```c
void single_writer(void)
{
    object_t *object;
    struct node *n;

    for (;;;;) {
        object = remove(something);
        n = CK_LIST_HEAD(my_list);
        CK_LIST_REMOVE(n, linkage);
        synchronize();
        free(object);
        free(n);
    }
}
```

- After **synchronize**, it is guaranteed no thread executing **reader** could have a reference to any of the logically deleted objects.
EPOCH RECLAMATION

• Upon entry into a read-side protected section, readers set an active bit and take a snapshot of a global epoch counter. **A memory barrier is required to avoid store to load re-ordering.**

• Synchronize operations increment the epoch counter only if all active threads have a snapshot of the latest value of the global counter.

• If the epoch counter is successfully incremented twice from the time synchronize was called, then no references could exist to objects logically deleted before the synchronize call.
EPOCH RECLAMATION

\[ E_g \]

\[ T_0 \]

LOGICAL DELETE  SYNCHRONIZE  PHYSICAL DELETE

\[ T_1 \]

\[ T_2 \]

Time
SYNCHRONIZE COST (WRITE-MOSTLY WORKLOAD/ RDTSCP)
## Epoch Reclamation

<table>
<thead>
<tr>
<th></th>
<th>Min</th>
<th>1st Qu.</th>
<th>Median</th>
<th>Mean</th>
<th>3rd Qu.</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epoch</td>
<td>188</td>
<td>772</td>
<td>1408</td>
<td>1781</td>
<td>2348</td>
<td>1888000</td>
</tr>
<tr>
<td>RCU</td>
<td>720</td>
<td>9260</td>
<td>26610</td>
<td>44080</td>
<td>60670</td>
<td>2181000</td>
</tr>
<tr>
<td>QSBR</td>
<td>400</td>
<td>10130</td>
<td>25420</td>
<td>38250</td>
<td>53590</td>
<td>4526000</td>
</tr>
</tbody>
</table>

**Synchronize Cost (Write-Mostly Workload/ RDTSCP)**
EPOCH RECLAMATION

ENTER COST (READ-ONLY WORKLOAD / RTSCP)
# EPOCH RECLAMATION

<table>
<thead>
<tr>
<th></th>
<th>Min</th>
<th>1st Qu.</th>
<th>Median</th>
<th>Mean</th>
<th>3rd Qu.</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epoch</td>
<td>72</td>
<td>76</td>
<td>76</td>
<td>204</td>
<td>236</td>
<td>53350</td>
</tr>
<tr>
<td>RCU</td>
<td>136</td>
<td>144</td>
<td>144</td>
<td>375</td>
<td>572</td>
<td>264600</td>
</tr>
<tr>
<td>QSBR</td>
<td>128</td>
<td>136</td>
<td>136</td>
<td>248</td>
<td>224</td>
<td>54330</td>
</tr>
<tr>
<td>Signal</td>
<td>44</td>
<td>44</td>
<td>44</td>
<td>62.35</td>
<td>48</td>
<td>63930</td>
</tr>
</tbody>
</table>

**ENTER COST (READ-ONLY WORKLOAD / RTSCP)**
EPOCH RECLAMATION

ENTER COST (WRITE-MOSTLY WORKLOAD / RDTSCP)
<table>
<thead>
<tr>
<th></th>
<th>Min</th>
<th>1st Qu.</th>
<th>Median</th>
<th>Mean</th>
<th>3rd Qu.</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epoch</td>
<td>72</td>
<td>100</td>
<td>484</td>
<td>619</td>
<td>940</td>
<td>551300</td>
</tr>
<tr>
<td>RCU</td>
<td>136</td>
<td>144</td>
<td>296</td>
<td>414</td>
<td>636</td>
<td>412800</td>
</tr>
<tr>
<td>QSBR</td>
<td>128</td>
<td>136</td>
<td>220</td>
<td>287</td>
<td>380</td>
<td>43050</td>
</tr>
</tbody>
</table>
THE END

HTTP://CONCURRENCYKIT.ORG