Driving Cache Replacement with ML-based LeCaR

Giuseppe Vietri†, Liana V. Rodriguez†, Wendy A. Martinez†, Steven Lyons†, Jason Liu†, Raju Rangaswami†, Ming Zhao‡, Giri Narasimhan†

† Florida International University
‡ Arizona State University
Reinforcement Learning On Cache Replacement Algorithms
## Hit-rate Performance vs. ARC

<table>
<thead>
<tr>
<th>Cache Size</th>
<th>Non-Parameterized</th>
<th>Adaptive</th>
<th>Fixed-Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LRU</td>
<td>LFU</td>
<td>FBR</td>
</tr>
<tr>
<td>1000</td>
<td>-6.1</td>
<td>-10.95</td>
<td>-1.97</td>
</tr>
<tr>
<td>2000</td>
<td>-3.61</td>
<td>-10.87</td>
<td>-2.1</td>
</tr>
<tr>
<td>5000</td>
<td>-1.6</td>
<td>-10.49</td>
<td>-1.72</td>
</tr>
<tr>
<td>10000</td>
<td>-1.17</td>
<td>-9.72</td>
<td>0.43</td>
</tr>
<tr>
<td>15,000</td>
<td>-0.77</td>
<td>-9.18</td>
<td>0.26</td>
</tr>
</tbody>
</table>

Worse than ARC  
Better than ARC

Adaptive Replacement Cache (ARC)

Strengths of ARC

- Manages both recent items as well as frequent items
- Dynamically adapts – Self-tuning
- Low overhead
- Competitive Hit-Rate performance
Conventional Online Learning Systems

1. Choose Expert
2. Follow Advice
3. Adjust Weights
4. Repeat

Disadvantages of current ML-based methods

- Not efficient
- Not competitive
ML-based LeCaR

- LRU and LFU
- Efficiency
- Learning the optimal mix.
Input: requested page q
If q in C then
  C.Update(q)
else
  if q is in $H_{LRU}$ then
    $H_{LRU}$.Delete(q)
  if q is in $H_{LFU}$ then
    $H_{LFU}$.Delete(q)
  UpdateWeights(q)
if C is full then
  action = (LRU, LFU)$\sim$($w_{LRU}$,$w_{LFU}$)
  if act == LRU then
    if $H_{LRU}$ is full then
      $H_{LRU}$.Delete(LRU($H_{LRU}$))
      $H_{LRU}$.Add(LRU(C))
      C.Delete(LRU(C))
    else
      if $H_{LFU}$ is full then
        $H_{LFU}$.Delete(LFU($H_{LFU}$))
        $H_{LFU}$.Add(LFU(C))
        C.Delete(LRU(C))
      else
        C.Add(q)
LeCaR (Miss not in History)

Input: requested page q

If q in C then
   C.Update(q)
else
   if q is in \(H_{\text{LRU}}\) then
      \(H_{\text{LRU}}\).Delete(q)
   if q is in \(H_{\text{LFU}}\) then
      \(H_{\text{LFU}}\).Delete(q)
   UpdateWeights(q)
if C is full then
   action = (LRU, LFU)\(\sim(\omega_{\text{LRU}}, \omega_{\text{LFU}})\)
   if act == LRU then
      if \(H_{\text{LRU}}\) is full then
         \(H_{\text{LRU}}\).Delete(LRU(\(H_{\text{LRU}}\)))
         \(H_{\text{LRU}}\).Add(LRU(C))
         C.Delete(LRU(C))
      else
         if \(H_{\text{LFU}}\) is full then
            \(H_{\text{LFU}}\).Delete(LFU(\(H_{\text{LFU}}\)))
            \(H_{\text{LFU}}\).Add(LFU(C))
            C.Delete(LFU(C))
      else
         C.Add(q)

Cache

History
### Weight Update

**LRU Regret**

- **Update**
  
  \[ w_{\text{LRU}} := \alpha \times w_{\text{LRU}} \]

**LFU Regret**

- **Update**
  
  \[ w_{\text{LFU}} := \alpha \times w_{\text{LFU}} \]

\(\alpha = \text{Learning rate}\)
Input: requested page q
If q in C then
    C.Update(q)
else
    if q is in H_{LRU} then
        H_{LRU}.Delete(q)
    if q is in H_{LFU} then
        H_{LFU}.Delete(q)
    UpdateWeights(q)
if C is full then
    action = (LRU, LFU) \sim (w_{LRU}, w_{LFU})
    if act == LRU then
        if H_{LRU} is full then
            H_{LRU}.Delete(LRU(H_{LRU}))
            H_{LRU}.Add(LRU(C))
            C.Delete(LRU(C))
        else
            if H_{LFU} is full then
                H_{LFU}.Delete(LFU(H_{LFU}))
                H_{LFU}.Add(LFU(C))
                C.Delete(LRU(C))
            else
                C.Add(q)
else
    if H_{LFU} is full then
        H_{LFU}.Delete(LFU(H_{LFU}))
        H_{LFU}.Add(LFU(C))
        C.Delete(LRU(C))
    else
        C.Add(q)
Experiments

- 8 Workloads
- 3 days of data [FIU datasets]
- Small, Medium and Large Cache Sizes
- Fixed Learning Rate
### Data used for the Experiments

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Casa, Ikki, Madmax and Topgun</td>
<td>Four different end-user/developer home directories</td>
</tr>
<tr>
<td>Online</td>
<td>Departments online course management system</td>
</tr>
<tr>
<td>Webresearch</td>
<td>Document store for research projects</td>
</tr>
<tr>
<td>Webmail</td>
<td>Mail server of the FIU Computer and Engineering department using Postfix</td>
</tr>
<tr>
<td>Webuser</td>
<td>Web server hosting faculty, staff, and graduate student web sites</td>
</tr>
</tbody>
</table>

*Collected at the School of Computing and Information Sciences at FIU.*

Results

“Online”

<table>
<thead>
<tr>
<th>%</th>
<th>LRU</th>
<th>LFU</th>
<th>ARC</th>
<th>LeCaR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.50%</td>
<td>75</td>
<td>75</td>
<td>75</td>
<td>75</td>
</tr>
<tr>
<td>1%</td>
<td>75</td>
<td>75</td>
<td>75</td>
<td>75</td>
</tr>
<tr>
<td>5%</td>
<td>75</td>
<td>75</td>
<td>75</td>
<td>75</td>
</tr>
<tr>
<td>10%</td>
<td>75</td>
<td>75</td>
<td>75</td>
<td>75</td>
</tr>
</tbody>
</table>
Results

LRU
LFU
ARC
LeCaR

Hit Rate (%) vs Cache Size for different workloads:
- casa
- online
- ikko
- madmax
Results

- LRU
- LFU
- ARC
- LeCaR

Graphs showing results for 'topgun', 'webmail', 'webresearch', and 'webusers'. Each graph compares performance with different metrics on a y-axis ranging from 0 to 100 and a x-axis ranging from 0.1 to 10.
Synthetic Data
Synthetic Data
Hoarding Rate

Definition

A: Stable Period
B: LFU gets penalized

Hoarded Page:
- Accessed at least twice
- Not among the last 2N unique pages
Conclusion

- ML-based LeCaR
- Small cache size
- Real + Synthetic experiments