Accelerating Rule-matching Systems with Learned Ranker
Rule Engine Matching Process

Input

Rule 1
Rule 2
...
Rule Match
...
Rule N

Filter*

Speed up the engine by removing certain un-matched rule(s)

E.g. a String, HTTP Request...

Matched Rule ID

Ranker

Prioritize matching rule as a top candidate to achieve early termination

Rule Match

Rule A
...
Rule 1
...
Rule X

Challenges of Ranker Design

- Ranker should estimate input features, instead of assuming data stream distribution.
- LRU or LFU based ranker orders ruleset for current input from historical data stream.
- LRU or LFU is good at long-tailed data stream but bad in uniform distribution.
Challenges of Ranker Design

- Learned ranker should consider the trade-off between inference cost and accuracy.
- Learned ranker should consider training data quality.
  - Artificial datasets might not provide sufficient insights to learn decision boundaries.
  - Logged real-world system workloads might not cover all cases.
Performance Gains from Learned Ranker

- Average reduction in the number of rules that the rule engine needs to process.

<table>
<thead>
<tr>
<th>Rule set</th>
<th>No rule ranker</th>
<th>Rule ranker</th>
<th>Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRS</td>
<td>22.38</td>
<td>1.68</td>
<td>92.49%</td>
</tr>
<tr>
<td>SNORT</td>
<td>91.56</td>
<td>1.34</td>
<td>98.54%</td>
</tr>
</tbody>
</table>

- Average reduction in latency for matching one input on different rule engines for CRS.

<table>
<thead>
<tr>
<th>Rule engine(regex)</th>
<th>No rule ranker</th>
<th>Rule ranker</th>
<th>Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCRE</td>
<td>1878.79 μsec</td>
<td>404.36 μsec</td>
<td>78.47%</td>
</tr>
<tr>
<td>PCRE with JIT</td>
<td>773.82 μsec</td>
<td>185.65 μsec</td>
<td>78.81%</td>
</tr>
<tr>
<td>RE2</td>
<td>206.01 μsec</td>
<td>55.15 μsec</td>
<td>73.22%</td>
</tr>
</tbody>
</table>
Thanks for watching!
USENIX 2019, July 12th 11: 50 am-1:10 pm
Track II : Machine Learning Applications & System Aspects