Efficient Miss Ratio Curve Computation for Heterogeneous Content Popularity

Damiano Carra
Computer Science Dept.
University of Verona
Verona, Italy

Giovanni Neglia
Université Côte d’Azur
Inria
Sophia Antipolis, France
Context

- Caches are fundamental components of computing architectures
  - Fast access to the most used items
- Shared resource
  - Used by different processes or applications, with different requirements and access patterns
- Main issue
  - How to divide and assign dynamically cache portions to applications?
Miss Ratio Curves - MRCs

- **Profiling**
  - Hit ratio vs amount of cache space for each application

- **Use**
  - Compute the MRCs for each application for a given interval
  - Assign cache space that maximize some objective function

- **Main concern: Computational complexity**
Current approaches

- MRC are computed from the trace
- Exact computation requires:
  - $O(M)$ memory
  - $O(\log M)$ computational complexity
- Approximate computation
  - Trade precision with space and speed
    - $O(1)$ memory and $O(1)$ operation per request
  - Most of the solutions are based on sampling
Spatial sampling

Approach:

- Observe a randomly selected fraction $R$ of the items
- Build the MRC and scale the X-axis by $1/R$

R can be adaptive

- This allows to obtain $O(1)$ memory and $O(1)$ computational complexity
Sampling bias

- Spatial sampling biased if requests rates are highly heterogenous across objects
  - The sample may or may not include some objects that are crucial for the MRC

- Solutions to such bias focus on the MRC tail
  - A correction factor accounts for the difference between the expected and actual average observed requests
Experiments on synthetic traces

IRM traces, with Zipf-distributed object popularity

- Various $\alpha$ (Zipf parameter)

![Graph showing miss ratio vs cache size for different $\alpha$ values](image.png)
[Detour] How to measure the accuracy?

- Mean Absolute Error (MAE)
  - Average of the absolute difference between the exact and approximate MRC

- Main issue → All sizes have the same weight
[Detour] MAE per Quantile $\rightarrow$ MAEQ

- Divide the Y-axis into intervals
  - And determine the corresponding intervals on the X-axis
- Compute the $MAE_i$ within each interval $i$
- $MAEQ = \text{average } MAE \text{ over all intervals}$
- In this example:
  - $MAE = 0.025$
  - $MAEQ = 0.09$
Accuracy with synthetic traces

- As $\alpha$ increases (high variability in the popularities) $\rightarrow$ the error increases
- Does the error depend on the tail of the popularity distribution?
The role of popular objects

- Case with $\alpha = 0.6 \rightarrow$ We add 20 popular objects

- The results are confirmed by a model (see the paper)
Our solution: Key idea

- Small cache sizes depend mostly on the highly popular items.
- Large cache sizes can be built from sample.
- Our approach → Combine exact and approximate MRCs.
  - Can adopt a “constant sampling rate” or “constant complexity” approach.
Experimental methodology

- Comparison with the state-of-the-art solutions
  - We set the parameters aiming at fair comparison
    - Use the same amount of memory

- CPU overheads
  - With our scheme, the CPU usage is on average 10% higher than state-of-the-art solutions
Results: IRM

- Accurate for any size
  - MAEQ always below 1%
Results: IRM, sensitivity to B

- Error mainly where the curves join
Real-world traces

- Traces from different sources, with different characteristics

<table>
<thead>
<tr>
<th>name</th>
<th>year</th>
<th># items</th>
<th># req</th>
</tr>
</thead>
<tbody>
<tr>
<td>fiu</td>
<td>2008</td>
<td>6.1 M</td>
<td>14.3 M</td>
</tr>
<tr>
<td>ms-ex</td>
<td>2007</td>
<td>2.6 M</td>
<td>8.9 M</td>
</tr>
<tr>
<td>ms-dev</td>
<td>2007</td>
<td>6.3 M</td>
<td>18.2 M</td>
</tr>
<tr>
<td>systor</td>
<td>2016</td>
<td>12.7 M</td>
<td>34.3 M</td>
</tr>
<tr>
<td>CDN</td>
<td>2015</td>
<td>1.6 M</td>
<td>11.2 M</td>
</tr>
</tbody>
</table>

Graphs showing request frequency vs Item ID for different traces.
Real-world traces: results

![Graphs showing Miss ratio vs Cache size for MRC and samples, comparing systor and CDN.](image)

![Bar chart showing Average error (MAEQ) for different Trace IDs, comparing SHARDS_{adj} and our mixed approach.](image)
Extension (1/2)

- “Non-stack” algorithms
  - Eviction policies that do not satisfy the inclusion property
  - Need to compute the MRC by points
  - Our approach:
    - use high R with small cache sizes, decrease R as the cache size increases
Heterogenous item size

- Web caches deal with items with heterogenous size
- Can we build the MRC in such a case?
  → Order statistics tree
- Does sampling work in this case?
Conclusions and perspectives

- Build a MRC from samples requires a careful design
  - Highly popular items have a significant impact
  - Instead of the tail of the distribution, the head is important

- In our approach we combine an exact MRC with an approximate one
  - Improving the accuracy of the final result

- Future works
  - Online adaptation of the scheme parameters
Thanks!

For any question, you can reach me at

damiano.carra@univr.it