Relational Edge Caching for Edge-Aware Web Applications
Hemant Saxena, Kenneth Salem

Introduction

- With the global expansion of web applications like eBay, Craigslist and Groupon, data centres have become more distant from the users.
- Many applications use edge networks, like Akamai, for caching static content, such as images, CSS and javascript files, to hide this network latency.
- However, other than the static content, web applications still communicate heavily with the distant database core for all their dynamic content.

Placement Advisor

- Task: To determine what tables of the database and which requests are “edgeable”.
- Each table has following two possible placement policy, we identify them by labels:
  - L1: The table is completely replicated statically at all edges, core site owns the mastership.
  - L2: The table is partially replicated at the edges. Partitions are horizontal slices across tables, identified by the primary-key. Edges can own the mastership of individual partitions.
- The tool also takes consistency requirements of each request into consideration.
- Consider the following cases:

  CASE 1: This is the base case where everything is at the core, therefore request also executes at the core.
  CASE 2: The request needs fresh data from the partition whose master stays at the core, and a stale copy of partition P2 also exists at core. Therefore, the request executes at the core.
  CASE 3: The request demands fresh data from a partition whose master stays at one of the edge, and a stale copy of partition P2 also exists at all the edge. Therefore, this request is edgeable.
  CASE 4: The request accesses two partitions which are edgeable, in this case it is hard to check at runtime if both the partitions are present at a single site, because partitions are frequently replicated and deleted from edge sites. Therefore the request runs at core.

- The tool does the above described analysis for each request, and assigns the possible execution site.
- It can be observed that each partition can get different labels from each request that access it.
- Therefore to handle the ties, we generate a Boolean expression that combines the possible placement decisions gathered for each request and uses a SAT solver to come up with a solution that fits all.

Evaluation

- Benchmark: RUBiS, an eBay like auction application. 26 predefined web operations and 7 tables.
- MySQL database at backend, application server runs Apache and accepts PHP requests.
- Experimental setup: we use SAVI Testbed to perform experiments. Consists of 1 core site located near Toronto and 3 edge sites, each located at Vancouver (VC), Montreal (MG) and Ottawa (CT).

<table>
<thead>
<tr>
<th>CORE</th>
<th>L1</th>
<th>L2</th>
<th>VC</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>7.5</td>
<td>8.7</td>
<td>62.8</td>
</tr>
<tr>
<td>CT</td>
<td>7.3</td>
<td>9.6</td>
<td>77.2</td>
</tr>
<tr>
<td>MG</td>
<td>8.7</td>
<td>9.6</td>
<td>30.0</td>
</tr>
<tr>
<td>VC</td>
<td>62.9</td>
<td>78.0</td>
<td>71.3</td>
</tr>
</tbody>
</table>

Conclusion

- Global web applications still face a problem of high network latency for dynamic content, such as query results over database, the reason being that the database is hosted at a distant data center, and Edge networks only cache static web content.
- A Core and Edge architecture utilizes the already existing Edge infrastructure to provide database features at Edge level. This can result in low network latency for dynamic content as well.
- In an Edge infrastructure, the placement of database between Core and Edge becomes crucial for the system performance.

Acknowledgement

This work is funded in part or completely by the Smart Applications on Virtual Infrastructure (SAVI) project funded under the National Sciences and Engineering Research Council of Canada (NSERC) Strategic Networks grant number NETG594424-10.