Relational Edge Caching for Edge-Aware Web Applications

Hemant Saxena, Kenneth Salem

Introduction

Fig 1. Amazon data centres and Akamai CDNs around the world. Courtesy: Akamai

- 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 data centre for all their dynamic content.

URL	Status	Domain	Size	Remote IP	Timeline
GET 2912274419247hash=item43ce821b	200 OK	ebay.ca	43.1 KB	66.211.181.182:80	674ms
GET odfntwifcuyqjklol4qctmkjl.css	200 OK	ir.ebaystatic.com	36.7 KB	216.191.211.211:80	7ms
GET 2nrzdlke4m3wvmnd4zcp1sssf.css	200 OK	ir.ebaystatic.com	812 B	216.191.211.211:80	3ms
GET imgLoading_30x30.gif	200 OK	pics.ebaystatic.com	3.1 KB	216.191.211.201:80	7ms
GET $_12.JPG	200 OK	i.ebayimg.com	26.4 KB	23.72.82.193:80	33ms
GET s.gif	200 OK	p.ebaystatic.com	49 B	216.191.211.201:80	7ms
GET $_14.JPG	200 OK	i.ebayimg.com	1000 B	23.72.82.193:80	34ms
GET $_14.JPG	200 OK	i.ebayimg.com	971 B	23.72.82.193:80	49ms
GET $_14.JPG	200 OK	i.ebayimg.com	1006 B	23.72.82.193:80	51ms
GET $_14.JPG	200 OK	i.ebayimg.com	1.3 KB	23.72.82.193:80	35ms

Fig 2. Latency of individual requests while viewing an item at eBay.ca

Overview

- **Problem**: Design an architecture that allow edge networks to host dynamic content as well.
- We present preliminary work towards an edge-aware data partitioning and dynamic data replication architecture for relational database systems supporting OLTP applications.

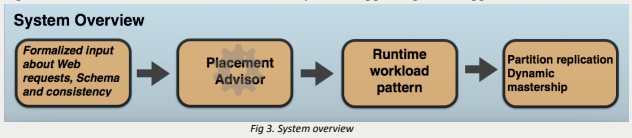
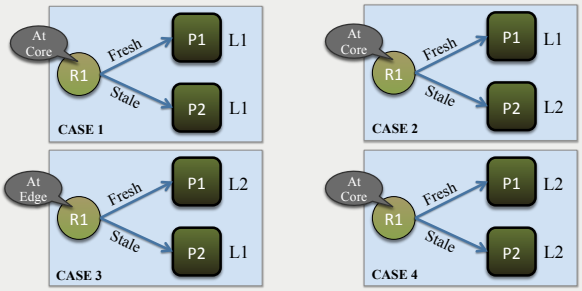


Fig 3. System overview

- Topology: A two-tier infrastructure consists of few **core** sites and numerous **edge** sites.
- Data:
  - Relational database at the core is partially replicated to the edges.
  - Partitions are horizontal slices of tables related via foreign-key relationship.
  - System maintains partition-level dynamic mastership.
- Application:
  - Application has short SQL queries embedded into predefined web operations.
  - Each web request can run either at an edge or at the core site, but not both.
  - Overall application execution is split between edge and core, each request is either executed at edge or forwarded to core.

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 edges. 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 assign 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 satisfiability expression that combine the possible placement decisions gathered for each request and uses a SAT solver to come up with a solution that fits all.

Runtime System

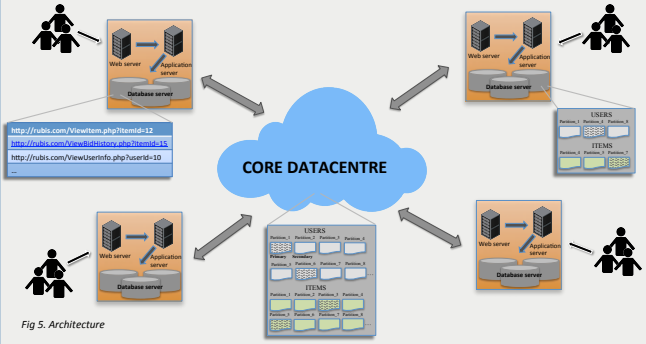


Fig 5. Architecture

- Each web request runs either at an edge or at the core, but not on both.
- Partitions may be replicated and each partition has a designated master site.
- Updates are propagated lazily to the secondary replicas.
- Any edge site responds to the request only if it has the required partition, else the request is forwarded to the core site.
- Each edge keeps a metadata about which requests it can handle locally.
- A background process at core monitors the access patterns at each edge, and triggers the data replication and mastership changes.

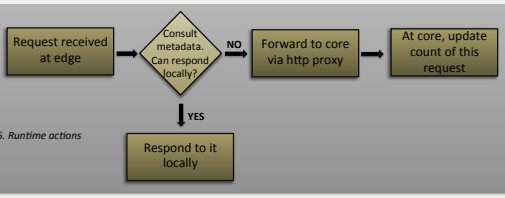
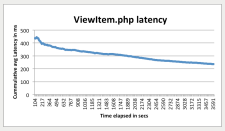


Fig 6. Runtime actions

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).**

	CORE	CT	MG	VC
CORE	X	7.5	8.7	62.8
CT	7.3	X	9.6	77.2
MG	8.7	9.4	X	70.9
VC	62.9	78.0	71.5	X



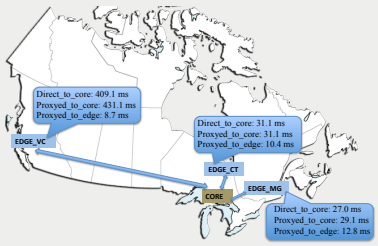


Fig 7. Micro-benchmark demonstrating latencies between the edges and core when ViewItem.php request is issued.

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 NETGP394424-10.

NSERC CRSNG