Understanding cache dynamics in the network: a software approach

Ian Marsh
SICS, Swedish-ICT
Sweden
ianm@sics.se

ABSTRACT

Complex time-varying *load* dynamics can be found at a network cache. This is because data requests carried by thousands of connections over diverse network paths load the cache by many read (and sometimes) write operations. User populations with a heterogeneity of request patterns and delay requirements induce *hitrate* dynamics at the network and cache interface. *Content* dynamics are further compounded by the existence of local expurgation policies and dimensioning decisions for each cache on a network path. Multi-level caches on the same network path may cause *inter-cache* dynamics of their contents. Finally, since the cache is inevitably part of a networked *system*, there is a *complex* dynamic relation between the cache *hitrate* and the network.

Categories and Subject Descriptors

Networks [**Network performance evaluation**]: [Network simulations]; Computing Methodology [**Visualization**]: [Visualization application domains]

Keywords

Caching, dynamics, hitrate, visualization.

1. THE CACHING PREMISE

The simple idea of caching is, if one can store data "closer" to the consumers, the time to access and retrieve the data can be lowered. This is because the data is closer in terms of distance (actual km) and probably less router hops, thereby avoiding potential congestion. Loads on the network and servers between the cache and server will also be reduced, assuming the cache has non-zero efficacy. All that said, buffering events during media replay are still commonplace and annoying, caching solutions may be at the root of problem or be part of a solution such as in a CDN. Either way, we present a joint simulation-visualization tool to give some **insight** into the dynamics of network caching.

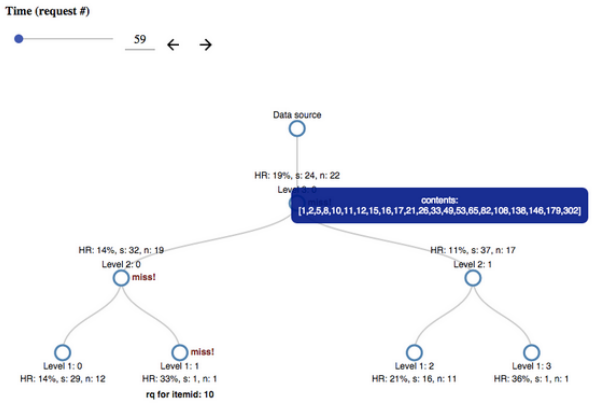

Figure 1: Inspection of a caching simulation.

2. THEORY, SIMULATION, VISUALIZATION

Within this section we elucidate where network caching can be better understood by using a *combination* of theory, simulation and visualization. Our role is not to dismiss or advocate existing approaches, but rather to show where approach differences can give rise to inaccuracies, e.g. in the predicted hitrate. We motivate a joint approach within the context of the original theoretical works, often expressed, or not, as a mathematical assumption. The independent reference model (below) being an example of what we could point to as an (accepted) difference between theory and practice. We believe a simulation-visualization tool, which can parse available datasets, typically compromising long streams of time-series data, to be valuable. In caching, where size is an important entity, a visualization can show where a 100×1U object is preferable to store, rather than 100×1U objects, or vice versa, by stepwise inspection. A single steppable, time-reversible tool has proven invaluable in our own work, i.e. assessing the effect of object size and tradeoffs between edge and pervasive caching, see figure 1.

3. HITRATE FROM THEORY

The seminal [3] introduced a unified framework for the analysis of a class of random allocation processes. In the paper Flajolet et al. analyze a Least Recently Used management scheme, using a probabilistic (independent) reference model. The framework, which has persisted for some years in the research community, means references to the cache objects are considered independent (from the content)

VALUETOOLS 2015, December 14-16, Berlin, Germany

DOI 10.4108/eai.14-12-2015.2262567

and stationary (constant mean and variance). Requests for content n arrive according to a time homogeneous Poisson process with intensity λ_n. Typically n is taken from a Zipf distribution with popularity parameter α from $1/r^{\alpha}$, where r is the rank of the object. Che et al. showed that the hitrate for each object present in an LRU cache can be derived *directly* [2]. They showed that the probability this object is present in the cache, $h(n)$, is approximated from the exponent of the popularity as given in the left approximation in Equation 1, where the unique root of t_C is given below,

$$h(n) \approx 1 - e^{q(n)t_C}, \qquad \sum_{n=1}^{N}(1 - e^{q(n)t}) = C. \qquad (1)$$

A user requests an object from a population of N objects, in a cache of capacity C. The probability a request is for object n, for $1 \leq n \leq N$, is proportional to popularity $q(n)$, independently of all past requests. Just to be clear, the cache eviction time T_C is assumed to be a deterministic constant, see [4]. An shortcoming in the original approximation is that the LRU hitrate can be variable in a real setting, it changes over time as new objects are introduced to the catalog. Within our environment, a dynamic catalog can be catered for by separating the input dataset into time-dependent portions, e.g. per hour, day or week.

4. A WORKFLOW APPROACH

Network simulators are commonplace, some which support caching (ndnSIM, Icarus). Producing a tool which allows user interaction, state examination, run-at-near native binary speed within a browser supporting multiple users, requires some software engineering. Furthermore, the demonstration abilities of the tool should be at least pedagogical; for students, industrial partners as well as network engineers in the inspection and analysis of complex dynamics.

We chose to use the Emscripten C++ to Javascript framework to compile the simulator, enabling us to verify if, the JS code can execute inside a virtual machine, and thus inside a web browser. However, just running the simulation inside a Javascript VM does not improve the simulate-plus-visualize workflow. Therefore, some work was done to run the Emscripten as a web worker, which avoids the browser stalling when executing the simulation. As a worker it can send its messages with the input parameters for the simulation, and receive the resulting JSON output to be visualized. Being runnable within a browser allows remote experimentation. Tracking the input sequence of dataset provides insight, particularly in workload characterizations, as shown in figure 2. In this case the workload was used to parameterize the time constants in a control theoretic approach to cache hitrate stabilization. Another simulation view is shown in figure 3. In this case, we have run multiple simulations, varying the size of the caches. This was done in order to see how large caches need to be to achieve a reasonable hitrate, and to see at what cache size (if any) we start to observe diminishing returns from increasing the cache size further. The returns seem to diminish after around 6% of our real catalog, which was from Orange's 3G network in France. It comprises `http` requests taken over the complete country between the access network and backhaul network. We should point out that the simulation can be seen from multiple views simultaneously and indeed writing additional views is straightforward,

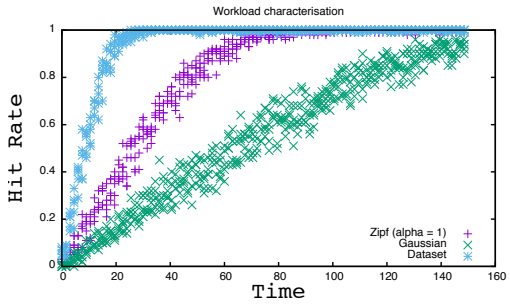

Figure 2: Input data to a workload characterization.

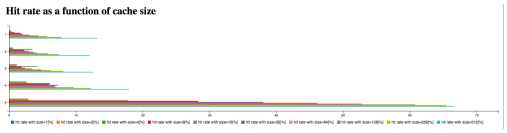

Figure 3: Example hitrate performance of a networked cache.

a CSS stylesheet and small snippets of PicoJSON in the rare case where instrumentation is not already available.

5. DISCUSSION

There are at least 20 papers on the analysis of LRU eviction policy spanning 15 years up to 2015. All that work done has been done for LRU only, whereas LFU is an alternative expiration policy, where few papers exist [1]. The point being, a pragmatic and arguably more accurate eviction policy can be coded into 15 lines of code within an existing simulator. This abstract, and accompanying tool is intended to bring theory and simulation closer together, as there are clearly advantages to both approaches. Given a *highly* dynamical system such as network caching, we propose, there is a role for good visualization tools as well. The simulator is publicly available, with documentation from[1].

6. REFERENCES

[1] Ardelius, J., Grönvall, B., Westberg, L., and Arvidsson, A. On the effects of caching in access aggregation networks. In *Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking* (New York, NY, USA, 2012), ICN '12, ACM, pp. 67–72.

[2] Che, H., Tung, Y., and Wang, Z. Hierarchical web caching systems: modeling, design and experimental results. *Selected Areas in Communications, IEEE Journal on 20*, 7 (Sep 2002), 1305–1314.

[3] Flajolet, P., Gardy, D., and Thimonier, L. Birthday paradox, coupon collectors, caching algorithms and self-organizing search. *Discrete Applied Mathematics 39*, 3 (1992), 207 – 229.

[4] Fricker, C., Robert, P., and Roberts, J. A versatile and accurate approximation for LRU cache performance. *24th International Teletraffic Congress, ITC '12* (2012).

[1] `https://bitbucket.org/redstrom/fmdcache`