The Scope for Online Social Network Aided Caching in Web CDNs

Amit Ruhela
IIT Delhi & C-DOT Delhi
India
aruhela@cse.iitd.ac.in

Sipat Triukose
NICTA
Australia
sipat.triukose@nicta.com.au

Sebastien Ardon
NICTA
Australia
sebastien.ardon@nicta.com.au

Amitabha Bagchi
IIT Delhi
India
bagchi@cse.iitd.ac.in

Anirban Mahanti
NICTA
Australia
anirban.mahanti@nicta.com.au

Aaditeshwar Seth
IIT Delhi
India
aseth@cse.iitd.ac.in

1. ABSTRACT

With Online Social Networking (OSN) services coming to prominence as a major platform for the popularization of Web content, recent research has focused on using information gleaned from these services to improve the performance of Content Distribution Networks (CDN). In this paper we find that contrary to popular belief, the simple cache replacement policy LRU outperforms methods based on OSN input. To establish this we propose an OSN-assisted caching strategy which we evaluate using a large Twitter dataset and through simulation. Our results show that vanilla LRU mostly outperforms the OSN-aided mechanisms. We study the reasons for this and find that if at any time there is a large enough percentage of highly popular content, which becomes popular within a short enough time span, CDN can do well by just following an LRU principle. While we do not claim generalizability to other workloads, or other caching architectures on the Internet, our results are important as they apply to the common use-case and show that simple algorithms do quite well.

Categories and Subject Descriptors

C.4 [**PERFORMANCE OF SYSTEMS**]: Performance attributes

Keywords

Caching Algorithms; LRU; Events Detection; Online Social Network

2. INTRODUCTION

Content Delivery Networks (CDN) are known to carry a large portion of Internet traffic. As a data point, Akamai, a leading CDN provider carries almost 15-30% of the global web traffic [3]. CDN work by returning content from a network of surrogate servers. This reduces the load on the origin servers, while often lowering the perceived network latencies for end-users. Intelligent surrogate server selection algorithms are often used to select the server closest to the client, in terms of latency or throughput. Within the CDN surrogate server network, cache replacement strategies are used to identify which content object to cache or evict from storage. Clearly a good cache replacement strategy is critical to a CDN's performance since fetching content from origin servers incurs a latency cost that negates a major benefit of CDN deployment.

It is therefore natural to think that a better understanding of content consumption patterns will lead to the design of better cache replacement policies. Since Online Social Networks (OSNs) have become a key platform for content popularization and spread, it is not surprising that several researchers have suggested that cache replacement schemes in CDN should take input from the so-called social signal i.e. the trends visible on OSNs [29]. Common intuitions are that (a) OSN activity can give hints on which topics are trending and should consequently be cached, and (b) simple schemes like LRU may not do well because the large amount of long-tailed content prevalent on the Internet can impair LRU-friendly content request patterns, thus reducing the cache hit rate.

In an effort to validate the aforementioned intuitions, we take the following approach. We developed a simulator for a global network of surrogate servers that respond to content requests from users across the world. The workload used to drive the simulator is based on previously collected and publicly available Twitter datasets. In this work, we use Twitter data as a proxy to content popularity; a tweet by a user is considered to trigger content requests by a fraction of followers of the user. The geographical locations of Twitter users help us generate the traffic and direct content requests towards the closest surrogate server. We focus on the relative performance of OSN-assisted caching algorithm vis a vis a vanilla LRU caching approach, where performance is measured by cache hit rates on the surrogate servers to benchmark the performance of different caching algorithms against each other. The counter-intuitive result we present in this paper is that LRU outperforms OSN event-based algorithm in most of the workload scenarios.

We find that event-based caching performs marginally better than LRU only when cache sizes are very small; with larger cache sizes, LRU does much better. This finding is consistent with what has been reported in the literature. Several studies [26, 29] in the past have also proposed complex caching algorithms and found that their improvements over LRU are very small.

978-1-4799-1640-5/13/$31.00

Our major contribution here lies in explaining the reasons behind vanilla LRU's good performance relative to OSN-assited algorithms. We show that despite the long-tailed distribution of content popularity [9, 21, 30], at any point in time the frequency of popular content requests is so high that caches are well off with simply using LRU. This is an important result with two kinds of implications. First, it provides further insight into content consumption patterns on OSNs. We study the frequency of content requests for popular and non-popular content, and the distribution of time between successive request for similar content, finding that these distributions allow LRU to fend off the long tail and keep popular content in its cache for requisitely long periods of time. Second, from the CDN side we show that simple caching strategies work well, even for the long-tailed content popularity distribution currently observed in Internet traffic traces. This will help guide future development of caching strategies for web CDN.

The remainder of this paper is organized as follows. We next present a brief overview of related work. Section IV presents details of the datasets used, the workload generation procedure, and the caching algorithms used. Finally, Section V evaluates the performance of different caching algorithms against each other.

3. RELATED WORK

Cache replacement strategies in the context of content delivery networks have been studied in the past by several researchers [5, 31]. Podlipnig et al. [26] classified cache replacement strategies according to recency, frequency, size, cost of fetching the object, modification time and expiration time. An analysis of cache replacement policies with respect to various access pattern at different points in the Internet is also covered in [14]. These studies are relevant to understand the workload dependent performance of different caching strategies.

Jelenković and and Radovanović [15] have performed an average-case analysis of LRU and shown that LRU fault probability is asymptotically invariant to the underlying dependency structure of the modulating process, i.e., for large cache sizes, the LRU fault probability behaves exactly the same as in the case of independent request sequences. Panagakis et.al. [24] have pointed out that when the memory of the request generation process is less than the capacity of the cache($h < C$), the hit ratio of the LRU replacement policy is approximately independent of the exact length of the memory h. In our work, we have also studied the performance of LRU with different cache sizes relative to the memory and shown that other than very small cache sizes, LRU performance is high in other cases. Our work thus validates their prior findings [15, 24] using real-world workloads.

Scellato et.al. [29] used geographic information extracted from social cascades to improve the caching of multimedia files in a Content Delivery Network. The authors compared LRU, LFU and mixed cache replacements by using 48 hours as the time threshold between consecutive steps of a cascade. The delivery performance implications of centralized and distributed CDNs was studied closely by Triukose et.al. [32,33]. Their study shows that a CDN could vary the number of cache servers' locations in a wide range without compromising the delivery performance of the entire platform. Both these studies give important context to the wide range of caching strategies proposed in recent literature, including the use of OSNs to aid caching. As a generic representative strategy, we use an event detection approach to detect viral topics to aid in cache selection. Here, most researchers [2, 4, 28] have worked on identifying events related to specific keywords occurring in a stream of tweets. We use a similar approach to detect both the start and end of events, and compare the performance of this representative OSN-aided caching strategy with LRU.

Our paper is part of a project studying various aspects of topic diffusion on Twitter that began with creating the data set [27] and continued on to a characterization of popular topics versus non-popular topics in [4]. We take this project forward in this paper by studying the implications of topic diffusion for content placement on CDNs.

4. METHODOLOGY

We design a simulation environment to mimic a global network of surrogate servers that responds to content requests from users across the world. A combination of Twitter datasets, described in the following paragraphs, is used as an input to generate the content request workload. We use Twitter data as a proxy for content popularity; a tweet by a user is considered to trigger content requests by a fraction of followers of the user. Knowledge of the geographical locations of Twitter users helps us generate the traffic and direct content requests towards the closest surrogate server. Surrogate servers are positioned around the world using a weighted algorithm based on the distribution of Twitter users. We then measure cache hit rates on the surrogate servers to benchmark the performance of different caching algorithms against each other.

4.1 Datasets

Our dataset is a combination of several available datasets.

- The twitter7 [18] dataset was collected from June 11, 2009 to September 1, 2009 and contains 196 million tweets of 9.8 million unique Twitter users.
- The KAIST [17] dataset contains social relationships of users in twitter7 dataset during July 6, 2009 to July 31, 2009 with 1.47 billion follower relations among 41.7 million Twitter users.

We merged the twitter7 and KAIST datasets, which results in a comprehensive dataset containing tweets, users, and following/follower relationships among users. This represents 23.5% of all Twitter users and 20-30% of all tweets posted during that period. We used Twitter APIs to find the location information of Twitter users. We used the Yahoo! BOSS Geo Service [1] to retrieve the latitude/longitude coordinates of the user locations. In addition, we used the OpenCalais [23] semantic analysis service, to gather semantic metadata and cluster tweets into aggregate topics. Overall, we found 39 million URLs and 7.5 million unique topics using OpenCalais from 141 million tweets. Out of topics given by OpenCalais and hashtags, we worked with a manageable set of 6 million topics belonging to the first 34 days of the dataset. We classified all topics in three categories namely popular topics, medium-popular topics and non-popular topics on basis of tweets count on each topic. The complete details about the dataset preparation can be referred from [27].

4.2 Surrogate Server Positioning

We vary the number of surrogate servers in our simulations from 1 to 15. To place the servers, we chose from the known geographical locations of the leading CDN provider's points of presence (POP): eg. MaxCDN(12 POPs), BitGravity(19 POPs), Edgecast(24 POPs), Amazon(28 POPs), Telefonica(34 POPs) and Limelight(80 POPs) [25]. From these locations, we picked a common subset of the top 15 locations having the greatest number of Twitter users around them, according to our dataset. Although there can be more than 15 servers in the CDN (e.g. 1000 in case of Akamai), we focus our study only to calculate the relative change in caching performance by changing the count of surrogate locations.

4.3 Cache Sizes

In all our simulations, we assume that the objects are of unit size. We evaluate caching performance by varying the cache size from 1 to 1200 objects. Since in our system, nearly 0.25 million objects are requested by end-users on each day, a cache size of 1200 objects roughly equals 0.48% of the total objects accessed on each day. The cache sizes of individual surrogate servers is proportionally allocated according to the count of users served by the surrogate server. We exclude results of the initial and final two days from the dataset to avoid any biases caused due to cache warm-up in generation of the consumption workloads.

4.4 Workload Generation

In our model, we assume that each time a user posts a tweet, a fraction of that user's followers will request content on the topic of that tweet. There are several parameters which controls how this process operates in details:

4.4.1 Number of Objects per Topic

In previous work such as [29], each URL embedded in a tweet is mapped to a content object. However, as only a small fraction of tweets contain URLs, this approach has the disadvantage of generating small, sparse workloads. To address this problem, we instead derive a synthetic workload from the tweet's text, where we assume that content requests on the Internet have similar spatio-temporal popularity characteristics to that of the topics discussed on Twitter. To generate this workload, we start by performing topic analysis on the tweet content using OpenCalais as mentioned in the previous section. The output of this topic analysis along with hashtags and URLs is then used to cluster tweets together within topics. Each topic is then assigned either one content object (which we refer to as *one object per topic model* or multiple content objects (*multiple objects per topic model*). In the case of the multiple topic model, the number of objects is chosen according to a Zipf distribution, which we observe when plotting the number of URLs per topic in our dataset.

4.4.2 Object accessed by Followers

In case of multiple objects per topic model, two further models are possible. (a) Followers may access the same object as the reference they got from the tweeting user. We call this a *Social Network (SN)* based workload. (b) Alternatively, followers may randomly choose from among all objects for that topic. We call this the *Non Social Network (Non-SN)* based workload.

4.4.3 Time-delay Distribution of Content Requests

Another parameter controlling the workload is the delay distribution from the time when a tweet is made, to when the followers raise content requests. To determine the Time-delay distribution of content requests, we collect a small dataset for 21 days from Twitter, consisting in a random sample of all tweets from Twitter. We measure the time difference between tweets and their retweets, a CDF of which is plotted in Figure 1. We find that 98.5% of retweets are posted within 2 days, with an exponential distribution of mean 9.05. This indicates that the inter-arrival time between requests for a content of a topic are Poisson distributed. Prior measurement studies [6, 11–13, 28] have also shown that the consumption requests for web content objects decreases exponentially with time.

For our time-delay distribution, we then consider three options: first, an exponential distribution in which 98.5% of consumption happens within 2 days, second, as a benchmark, an exponential distribution with mean 25.28, in which 98.5% of consumption happens within 7 days, and finally a uniform distribution of requests within these days rather than an exponential distribution.

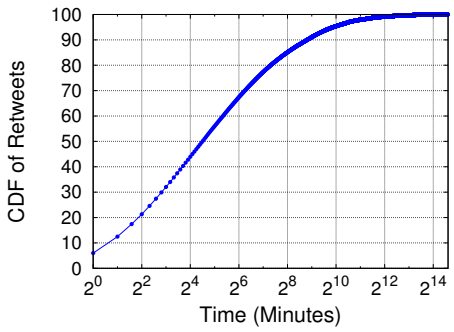


Figure 1: Time-Delay Distribution of Retweets

4.5 Events based Caching Algorithm

Event detection is a well-studied topic in Seismology [7] [34], where a short-term average (STA) is maintained to be sensitive to seismic events while a long-term average (LTA) provides information about the temporal amplitude of seismic noise at the site. When the ratio of both exceeds a certain value, an event is declared. We use a similar approach to find events in topic streams.

We work with the observation that during an event, tweets for a topic are made frequently and therefore the average inter-arrival time between tweets is low. We define α as the ratio of the average inter-arrival time between tweets within a short term window over a long term window.

Setting the window of the STA is a crucial parameter for events detection algorithm to work as setting STA very large makes the detector insensitive to events occurring on shorter time scales. On the other hand, setting the window too small makes the detector highly sensitive to noise and therefore becomes prone to false positives. In our dataset, nearly 95% of consumption requests are generated by topics that have been accessed 20 times or more. We therefore set STA window of size 20. The long term average (LTA) for each topic is maintained as a moving average of inter-arrival time of all requests so far. An event on a topic occurs when α drops be-

low an Event Trigger Threshold (ETT), and continues until it stays below an Event De-Trigger Threshold (EDT).

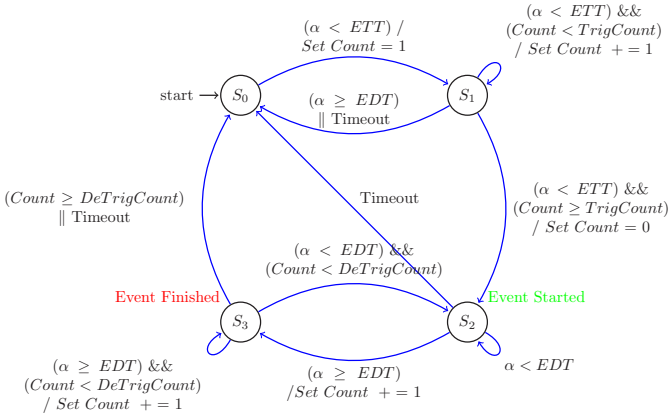


Figure 2: State diagram of the events detection

Figure 2 shows a state diagram of our event detection algorithm. The description about the states of events detection algorithm is as follows.

S_0 **- Default State :** For each topic, the detector starts in the default state S_0 and waits until there are enough requests to fill the short term window before the α is computed. Then it checks whether α satisfies the state transition conditions. On an arrival of each content request, the detector re-computes α and re-checks the state transition conditions.

S_1 **- Event Trigger :** To avoid a spike in the STA value, which could lead to a false event trigger, the detector makes sure that the α stays below the Event Trigger Threshold for a long enough time. It waits in the state S_1 for $TrigCount$ requests before it declares the start of an event and transits to the state S_2. If α goes above EDT or the detector stays idle for more than idle Timeout time(4 Hrs), the state is changed back to the default S_0 state.

S_2 **- Event Started :** The detector stays in state S_2 as long as α remains smaller than EDT. When α goes above EDT, the state is changed to S_3. If the detector stays idle for more than idle Timeout time, the state is changed to S_0 and the current event is said to be finished.

S_3 **- Event Completed :** Similar to the false trigger of the event start, there can be false triggers of the event stop as well. In this case, the detector waits in S_3 state for $DeTrigCount$ requests to confirm that the event is actually expired. If the trigger is false, the state is changed back to S_2. If the count of violations becomes greater than $DeTrigCount$ or the detector stays idle for more than idle Timeout time, the state is changed to S_0 and the current event is said to be finished.

Table 1 shows the values of parameters used in events detection algorithm. The values are determined on basis of samples chosen from each topic class(Popular, Medium-Popular and Non-Popular topics) and comparing the events identified by events detection algorithm with real events that happened on those sample topics.

Figure 3 shows an example of how the event detection algorithm works in practice. Tweets from the topic "Michael Jackson" are fed into the detector and values of α are plotted over time. The horizontal black arrows indicate the duration in which the detector finds the events. The first event corresponds to the *Death of Michael Jackson* on *25 June 2009* and the second event corresponds to the public memorial on *July 7, 2009* which was broadcast worldwide with a reported audience of 2.5 billion people.

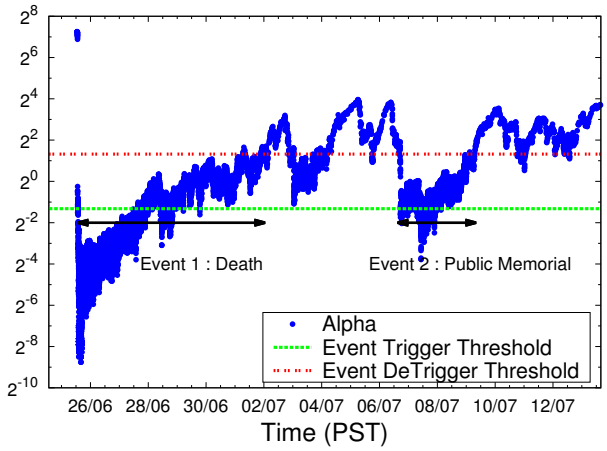


Figure 3: Events Detection on topic "Michael Jackson"

4.6 System Architecture for Events based Simulations

We assume an architecture (Figure 4) where the CDN provider may have access to a Twitter stream to build in real-time OSN-aided indicators that can guide cache replacement policies at the surrogate servers. For simplicity, we assume a centralized model where the CDN provider can track the topics being discussed on Twitter. We use a simple event detection algorithm to find the start and end time of topic events, which become available to surrogate servers to know the top active topics whose contents can be cached. The same Twitter dataset which is used to generate the workload is used by the event detection algorithm to maintain this list of top topics. We experiment with variations such as maintaining a globally common top topics list and surrogate server specific topic lists, but our results remain unchanged.

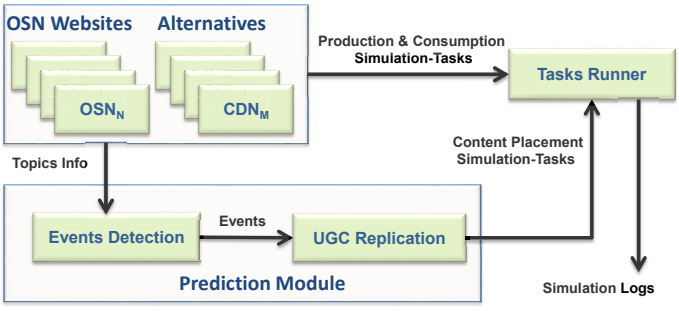


Figure 4: Architecture for Events based Simulations

Table 1: Parameters used in events detection algorithm

Parameter	Description	Value
Events Trigger Threshold(ETT)	An event occurs when $\alpha(= STA/LTA)$ drops below an ETT	0.4
Events De-Trigger Threshold(EDT)	An event ends when $\alpha(= STA/LTA)$ goes above EDT	2
STA Window Size	Size of short term window that contains inter-arrival time between successive tweets	20
TrigCount	Number of additional tweets which must sustain $\alpha < ETT$ to confirm start of an event	5
DeTrigCount	Number of additional tweets that must sustain $\alpha > EDT$ to confirm end of an event	15

5. SIMULATION RESULTS

We next present simulation results showing how the mix of topics influences caching performance.

5.1 Influence of Popular Topics

We assume a simple workload model for this section, with multiple surrogate servers and one object per topic. Variations with changing the delay distribution for request generation from exponential with a mean of 9.05 minutes to uniform with a mean of 3.5 days, are also presented.

The rank-frequency plot of the consumption workload is shown in Figure 5. The figure indicates that the consumption workload is Zipfian with a heavy-tail and raises a concern that the large fraction of niche topics may interfere with LRU's simple operation by causing extensive thrashing of cached content. Figure 6 however shows that the events-based algorithm does better than LRU only with very small cache sizes. We begin to explore the reasons behind this.

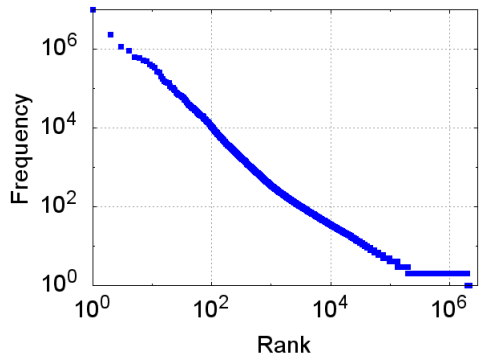


Figure 5: Rank-Frequency plot of Consumption Workload

5.1.1 Separation between Consecutive Requests for the Same Content

For ease of exposition, we choose one popular topic ("IRANELECTION"), one medium popular topic ("NECC09"), and one non-popular topic ("ICANN"). The average inter-arrival time for requests of the same contents for all topics and the sampled topics is shows in Figure 7. The average number of content objects requested between consecutive requests for the same content object are similarly shown in Figure 8.

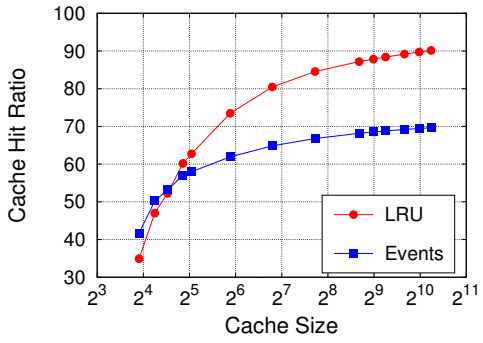


Figure 6: Exponential within 2 Days: 15 Servers

The figures indicate that on average consecutive requests for the same content are very closely spaced, and of the order as the cache size. This is an ideal situation for LRU to do well since the objects for popular content may never get evicted from the cache.

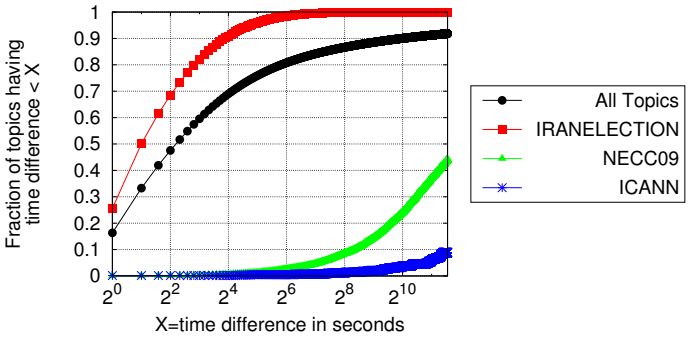


Figure 7: Average time difference between requests on same the contents

5.1.2 Residence Time of Contents in the Cache

We again sample one popular topic "IRANELECTION", one medium popular topic "NECC09" and one non-popular topic "ICANN", to determine the time spent by an object within the cache while running LRU. Table 2 shows that even with larger cache sizes medium and non-popular topics are not able to survive inside the cache for a long duration,

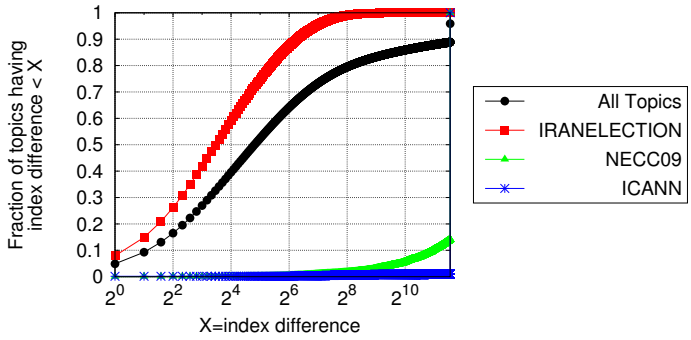


Figure 8: Average index difference between requests on the same contents

but popular topics, despite being challenged by a heavy tail, are able to sustain themselves inside the cache for long durations.

Table 2: Time spent in the cache by the topics

Topic	Cache Size	Time spent in seconds
IRANELECTION	2	687 817 (7.96 days)
IRANELECTION	59	2 652 572 (30.7 days)
NECC09	2	133
NECC09	59	7 238
ICANN	2	6
ICANN	59	408

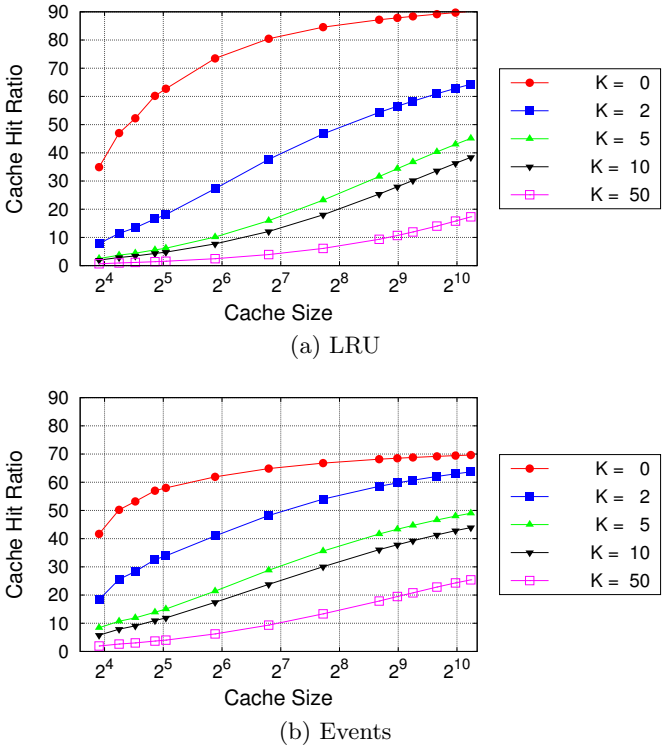
5.1.3 Effect of the removal of top 'K' Topics

To further attribute the reason behind LRU's good performance to the high incidence of popular topics, we removed the top 'K' topics from the dataset. The performance of both caching algorithms is shown in Figure 9. Clearly the performance falls in both cases, as expected. To understand the relative impact of top topics removal for both algorithms, in Table 3 we state the crossover point for cache size when LRU starts performing better than the events-based algorithm. We can see that with removing more and more top topics, the crossover point also increases, and eventually the events-based algorithm begins to completely outperform LRU. In fact, with a more relaxed delay distribution of content requests generated within a 7-day period than a 2-day period, the events-based algorithm begins to entirely outperform LRU with removing only top two topics.

These observations strongly point to the fact that the incidence of highly popular topics in the dataset results in LRU performing quite well, with more complicated event-based algorithms as not particularly useful from the CDN point of view.

5.2 Other Workloads

In this section, we study other kinds of workloads to see if the same trend is preserved. The results are shown in Table 4. Each entry in Table 4 has the format (X/Y/Z) where X corresponds to the case when there is only 1 surrogate



(a) LRU

(b) Events

Figure 9: Removal of Top k Topics

server, Y corresponds to the case when there are 5 surrogate servers, and Z corresponds to the case when there are 15 surrogate servers. The X, Y and Z values are the crossover points when LRU starts doing better than the events-based caching algorithm.

5.2.1 Changing Count of Surrogate Servers

Figure 10 shows the spread of topics across multiple surrogate servers; as expected, popular topics seem to be in demand at more surrogate servers. When the number of servers is increased, the cache size per server decreases and hence causes thrashing for LRU as can be seen in Figure 11. The events-based algorithm also loses performance but is more selective about which objects to cache, and therefore does not perform as worse as LRU.

5.2.2 Changing Count of Objects per Topic

With multiple objects per topic, since the number of objects is higher therefore consumption requests are dispersed, leading to poor caching performance for both the algorithms. Figure 12 shows that the gap in caching performance for LRU between the single-object and multiple objects per topic workloads is larger than the gap for the events-based algorithm. This again shows the merit of events-based caching when cache sizes are very small, but as the sizes increase then LRU begins to perform much better.

5.2.3 Changing the Contents Selection Behavior

Recall that we work with two models for consumption requests: an SN-based model where followers of a user access the same object assumed to be referred by a tweet, and a non-SN model where the followers may request for

Table 4: Relative Cache Performance

Scenario	Cache Size with 1/5/15 Surrogate Servers		
	OneObj per Topic	Multiple Obj per Topic in SN Workload	Multiple Obj per Topic in NonSN Workload
Negative exponential consumption within 2 Days	2/9/24	11/59/144	12/69/182
Negative exponential consumption within 7 Days	5/24/70	60/298/694	63/310/728
Uniform consumption within 2 days	3/16/47	40/166/433	42/179/461
Uniform consumption within 7 days	5/22/61	57/280/644	59/287/665

Table 3: Removal of top K topics in SN based workload with 5 servers

K	Cache Size (Exponential consumption in 2 days)	Cache Size (Uniform consumption in 7 days)
0	24	61
1	176	406
2	1042	Events Better
2	Events Better	Events Better

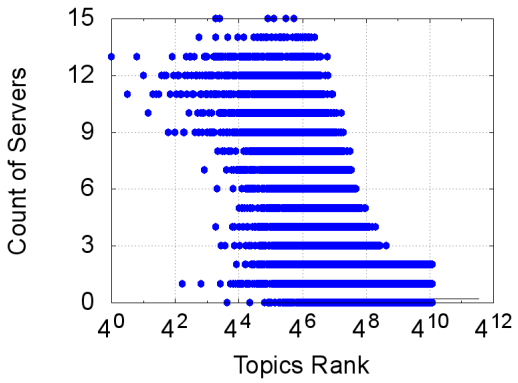


Figure 10: Each topic is discussed on how many servers

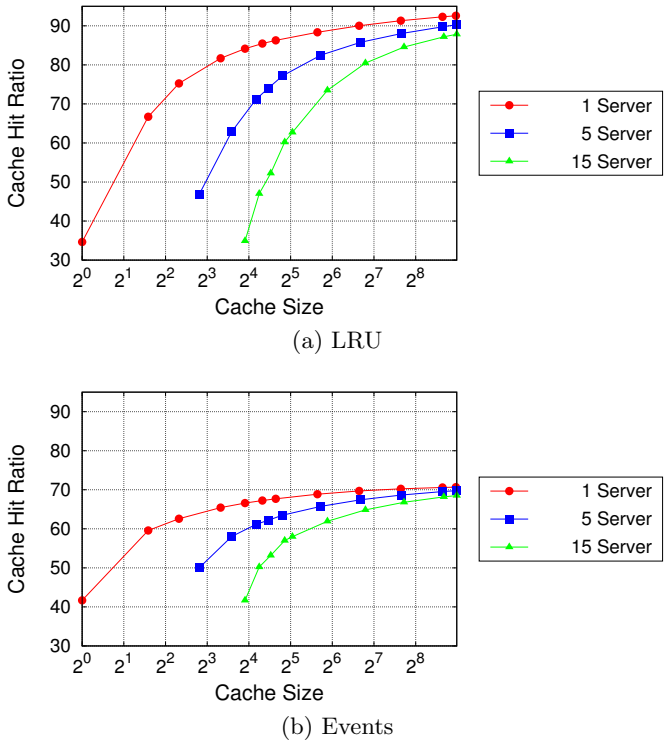


Figure 11: Cache performance by changing count of Surrogate Servers : SN based workload with Exponential Consumption in 7 Days

any object belonging to the topic of the tweet. As expected, LRU performs slight better with an SN-based workload since this results in correlated content requests. The events-based algorithm however does not show much difference in performance between SN and non-SN workloads as shown in Figure 13. This results in an overall relative improvement of the events-based algorithm over LRU, as can be seen in Table 4.

We therefore find that even with other workloads, the same trend of LRU winning over more complicated algorithms persists, because of the high incidence of topic popularity in the workloads. Since we expect web workloads to be similar to the Twitter workloads, this indicates that CDN providers may be well off by simply using LRU instead of more complicated algorithms.

6. DISCUSSION

We have made several assumptions in our experiments, for ease of analysis. First, our study is done only on the Twitter OSN. However, since social networks are known to be representative of the web workload [19], and also similar to each other in terms of popularity distributions [8, 10, 11, 20, 22], we feel our analysis will extend to other social networking workloads as well. Second, we have assumed an exponential distribution of consumption requests for content objects. This is supported by other studies [11–13] and our own experiments as well. Further, we have also experimented with uniform distribution of consumption requests as well. Third, we have shown the validity of our analysis on a simulated network of 5-15 surrogate servers, and consumption request generation within 2 to 7 days post production. Although the ranges are limited for ease of simulations, the trend

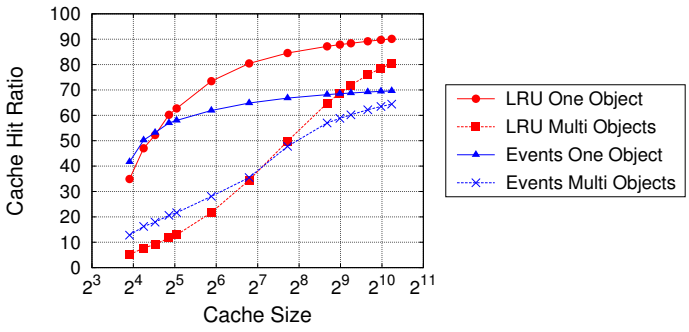


Figure 12: One Object per topic Vs Multiple Objects per Topic

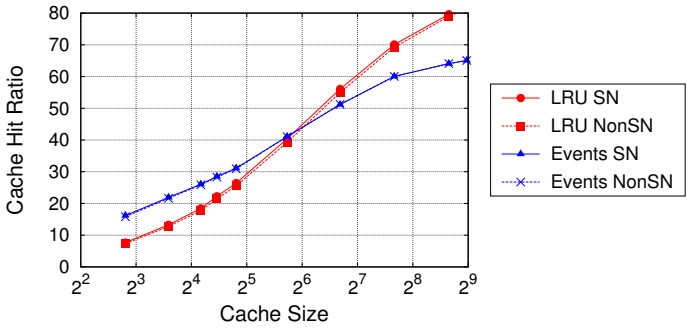


Figure 13: SN vs Non-SN Model

of improved LRU performance with more surrogate servers (implying larger aggregate cache sizes) and larger spread of requests, illustrate that even with higher ranges our conclusions will hold.

7. CONCLUSIONS AND FUTURE WORK

Several researchers have tried to use OSN signals to improve the caching performance of web CDNs. We demonstrate a negative result explaining why even the simple LRU algorithm does well on common workloads; we show that due to the high incidence of popular topics, even heavy tailed workloads are not able to disrupt LRU's performance. This finding is consistent with the result of Jelenković et.al. [16] who have shown that LRU performs as well on a dependent sequence of random requests as it does on an independent sequence of requests with the same request frequencies, as long as the cache size is large enough. Our work can be seen as an empirical validation of this result for the case where the request correlations are defined by object consumption patterns in OSNs. We also experiment with variations of the workload and show which specific variations can indeed benefit from complicated caching algorithms aided by OSN knowledge.

While our work indicates that CDN providers may be well off with just using LRU, it points towards the role that OSN knowledge can play in hybrid CDN-P2P architectures. OSN aided event detection methods can be used, for example, to determine which content to serve from the P2P network and how to control its spread in the network. We plan to look into such issues in the future and build caching strategies for different classes of topic popularity.

Acknowledgments

The authors would like to thank V V R Sastry, Vipin Tyagi and Ravi Gupta from C-DOT India for their valuable support and suggestions. This work was supported by the Commonwealth of Australia and the Department of Science and Technology, India, under the Australia-India Strategic Research Fund, and the Department of Information Technology, India under research project #RP02442 and #RP02355.

8. REFERENCES

[1] Yahoo! placefinder guide. `http://developer.yahoo.com/geo/placefinder/guide/`, June 2011. [Online].

[2] F. Abel, C. Hauff, G.-J. Houben, R. Stronkman, and K. Tao. Twitcident: fighting fire with information from social web streams. In *Proceedings of the 21st international conference companion on World Wide Web*, WWW '12 Companion, pages 305–308, New York, NY, USA, 2012. ACM.

[3] Akmai. Facts and figures. `http://www.akamai.com/html/about/facts_figures.html`, May 2013. [Online; accessed 25 May-2013].

[4] S. Ardon, A. Bagchi, A. Mahanti, A. Ruhela, A. Seth, R. M. Tripathy, and S. Triukose. Spatio-temporal and events based analysis of topic popularity in twitter. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management (CIKM 2013)*, San Francisco, CA, USA, 2013.

[5] S. Bakiras and T. Loukopoulos. Combining replica placement and caching techniques in content distribution networks. *Comput. Commun.*, 28:1062–1073, June 2005.

[6] Y. Borghol, S. Mitra, S. Ardon, N. Carlsson, D. Eager, and A. Mahanti. Characterizing and modelling popularity of user-generated videos. *Perform. Eval.*, 68(11):1037–1055, Nov. 2011.

[7] F. Botella, J. Rosa-Herranz, J. Giner, S. Molina, and J. Galiana-Merino. A real-time earthquake detector with prefiltering by wavelets. *Computers and Geosciences*, 29(7):911–919, 2003.

[8] M. Cha, H. Haddadi, F. Benevenuto, and P. K. Gummadi. Measuring user influence in twitter: The million follower fallacy. In *ICWSM*, 2010.

[9] M. Cha, H. Kwak, P. Rodriguez, Y. Ahn, and S. Moon. Analyzing the video popularity characteristics of Large-Scale user generated content systems. *IEEE/ACM Transactions on Networking*, 17(5):1357–1370, 2009.

[10] M. Cha, A. Mislove, and K. P. Gummadi. A measurement-driven analysis of information propagation in the flickr social network. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 721–730, New York, NY, USA, 2009. ACM.

[11] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. Youtube traffic characterization: a view from the edge. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, pages 15–28, New York, NY, USA, 2007. ACM.

[12] L. Guo, E. Tan, S. Chen, Z. Xiao, and X. Zhang. The stretched exponential distribution of internet media access patterns. In *Proceedings of the twenty-seventh ACM symposium on Principles of distributed computing*, PODC '08, pages 283–294, New York, NY, USA, 2008. ACM.

[13] L. Guo, E. Tan, S. Chen, X. Zhang, and Y. E. Zhao. Analyzing patterns of user content generation in online social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 369–378, New York, NY, USA, 2009. ACM.

[14] G. Hasslinger and O. Hohlfeld. Efficiency of caches for content distribution on the internet. In *Teletraffic Congress (ITC), 2010 22nd International*, pages 1–8, 2010.

[15] P. R. Jelenković and A. Radovanović. Least-recently-used caching with dependent requests. *Theor. Comput. Sci.*, 326(1-3):293–327, Oct. 2004.

[16] P. R. Jelenković, A. Radovanović, and M. S. Squillante. Critical sizing of LRU caches with dependent requests. *Journal of Applied Probability*, 43:1013–1027, 2006.

[17] H. Kwak. What is twitter, a social network or a news media? `http://an.kaist.ac.kr/traces/WWW2010.html`, June 2011.

[18] J. Leskovec. Snap: Network datasets: 476 million twitter tweets:. `http://snap.stanford.edu/data/twitter7.html`, Aug. 2011.

[19] M. Marcon, B. Viswanath, M. Cha, and P. K. Gummadi. Sharing social content from home: a measurement-driven feasibility study. In *NOSSDAV*, pages 45–50, 2011.

[20] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, pages 29–42, New York, NY, USA, 2007. ACM.

[21] S. Mitra, M. Agrawal, A. Yadav, N. Carlsson, D. Eager, and A. Mahanti. Characterizing web-based video sharing workloads. *ACM Trans. Web*, 5(2):8:1–8:27, May 2011.

[22] A. Nazir, S. Raza, and C.-N. Chuah. Unveiling facebook: a measurement study of social network based applications. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, IMC '08, pages 43–56, New York, NY, USA, 2008. ACM.

[23] OpenCalais. How Does Calais Work? `http://www.opencalais.com/about`, May 2011.

[24] A. Panagakis, A. Vaios, and I. Stavrakakis. Approximate analysis of lru in the case of short term correlations. *Comput. Netw.*, 52(6):1142–1152, Apr. 2008.

[25] C. Planet. Overview of content delivery networks - cdn planet. `http://www.cdnplanet.com/cdns/`, Jan. 2013. [Online; accessed 10 Jan-2013].

[26] S. Podlipnig and L. Böszörmenyi. A survey of web cache replacement strategies. *ACM Comput. Surv.*, 35(4):374–398, Dec. 2003.

[27] A. Ruhela, R. Tripathy, S. Triukose, S. Ardon, A. Bagchi, and A. Seth. Towards the use of online social networks for efficient internet content distribution. In *Advanced Networks and Telecommunication Systems (ANTS), 2011 IEEE 5th International Conference on*, pages 1–6, 2011.

[28] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 851–860, New York, NY, USA, 2010. ACM.

[29] S. Scellato, C. Mascolo, M. Musolesi, and J. Crowcroft. Track globally, deliver locally: improving content delivery networks by tracking geographic social cascades. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 457–466, New York, NY, USA, 2011. ACM.

[30] V. K. Singh and R. Jain. Structural analysis of the emerging event-web. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 1183–1184, New York, NY, USA, 2010. ACM.

[31] K. Stamos, G. Pallis, and A. Vakali. Integrating caching techniques on a content distribution network. In *Proceedings of the 10th East European conference on Advances in Databases and Information Systems*, ADBIS'06, pages 200–215, Berlin, Heidelberg, 2006. Springer-Verlag.

[32] S. Triukose, Z. Wen, and M. Rabinovich. Content delivery networks: how big is big enough? *SIGMETRICS Perform. Eval. Rev.*, 37(2):59–60, Oct. 2009.

[33] S. Triukose, Z. Wen, and M. Rabinovich. Measuring a commercial content delivery network. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 467–476, New York, NY, USA, 2011. ACM.

[34] A. Trnkoczy. Understanding and parameter setting of sta/lta trigger algorithm. In P. Bormann, editor, *New Manual of Seismological Observatory Practice (NMSOP)*, pages 1 – 20. Deutsches GeoForschungsZentrum GFZ, Potsdam, 2009.