The effect of video caching on network resource planning – a real-case study.

Luca Brilli, Romano Fantacci, Tommaso Pecorella
Dpt. of Information Engineering, Università di Firenze
Firenze, Italia
luca.brilli@unifi.it, romano.fantacci@unifi.it, tommaso.pecorella@unifi.it

Tiziano Ionta
Telecom Italia S.p.A.
Roma, Italia
tiziano.ionta@telecomitalia.it

***Abstract*—Traffic Engineering is one of the building blocks for a correct network planning. Internet Service Providers are always trying to fulfill the user Quality of Experience (QoE). However, each technological advance brings new services to the user, with new challenges to be solved to maintain the QoE.**

An example of this feedback system is the increased access speed in the user's premises. The increased bandwidth allows the Internet-based video services, but the increased load also requires new techniques to prevent congestion in the core network. A quite common system to lower the network load and increase the system availability is is represented by Content Delivery Networks (CDNs). However, the optimization of the CDN servers optimization is still an open problem.

In this paper we will analyze the performance of a deployed Content Delivery Network caching system. We will derive its performance metrics and we will show how a properly chosen metric can indicate if the cache is underutilized. We will also demonstrate that some 'logical' assumptions are not true in the vast majority of cases, and that moving the cache close to the users is negatively impacting the network performances in most cases.

***Index Terms*—Content Delivery Network, Cache, Video on Demand, Traffic Analysis**

I. Introduction

The types of data traffic carried by Internet did vary over time. One of the main success reasons of Internet is, indeed, that it was not designed to carry any specific traffic. As a consequence, it has been able to adapt itself to the ever-changing traffic patterns generated by the users.

The traffic characterization is very important to design and manage a network. Its correct forecast enables a network provider to expand the network and to foresee if and how it can offer new services to the users.

In the old telephone networks, the traffic was mainly characterized though Erlang B or C formulas, and a common assumption was that the traffic follows a Poisson distribution. The Poisson model, and in general the Markov models, are very interesting because they often allow closed-form solutions. However, in a milestone paper Paxson and Floyd [1] demonstrated that Internet traffic is not Poisson-like. On the opposite, it is Long Range Dependent (LRD).

However, Internet traffic characterization needs to be always re-validated. The changes in the users behaviour, services, link capacity, or even a small change in a protocol (e.g., HTTP/1.1 to HTTP/2) can have a dramatic effect on the traffic pattern (see for example [2], [3]).

The traffic pattern problem is of particular interest when new services are rolled out. As an example, when an Internet Service Provider (ISP) wants to offer a video streaming service to its users, it needs to evaluate the amount of traffic that this service will generate. Due to the Internet horizontal business model, the traffic could also be modified as a reaction to new service offered by third party, e.g., an Internet-based television on demand. It is fairly obvious that the ISP goal is to provide an active support to the new traffic types, in order to maximize the user's Quality of Experience (QoE). Failing to match the expected QoE could be detrimental for the third party provider, but most probably it will be more detrimental for the ISP.

In order to enhance the user's QoE, ISPs and service providers use a number of techniques, often completely transparent to the user. The most common one is the use of CDNs [4]. A CDN enhance the service quality and availability by replicating the content in multiple servers, possibly closer to the final user. As a matter of fact, a local server of a CDN is similar to a transparent cache system: the user should not know that it is there, and it should dynamically update the locally stored resources to save internal memorization space.

The main problems in a CDN are: 1) where to place the servers, 2) how to manage the content in the servers and 3) how to choose the 'best' server for a given resource request.

It must be noted that CDN performance is dependent on the traffic pattern, but it can change the traffic pattern as well. In other terms, the optimization process must be dynamic. Moreover, CDN servers hardware (i.e., memory, SSD disks, etc.) is expensive. As a consequence, the cache optimization is an important element to consider in the network design phase.

In this paper we will analyze some real traces derived from an ISP content delivery network caching system. The outcomes will be useful to have a better understanding of the cache load type and variation, and can drive decisions about their position in terms of number of served users.

The rest of the paper is structured as follows. In Section II we will describe our scenario along with the relevant state of the art. In Section III and IV we will outline respectively our reference architecture and the kind of data available for the study. Section V will analyze the results obtained from the data. and Section VI will summarize the important elements we found and the future research ideas.

II. State of the art for Content Delivery Networks

Multimedia content delivery is quite old problem. The first approaches to efficient multimedia dissemination relied on IP multicast. However, this solution is only feasible for live contents. Moreover, IP multicast still faces many hurdles in its deployment, preventing its effective use.

Peer-to-peer systems have been studied as a possible alternative for efficient video streaming [5], [6]. However, this technology did not gain much attention, probably due to the security and accounting/billing issues associated with P2P content.

Content Delivery Networks (CDNs) [7]–[10] have gained more attention by Content Service Providers and Internet Service Providers (ISPs) because they do not require any special software and the necessary network infrastructure can be deployed with a relatively small effort.

As a matter of fact, a CDN represents an overlay network that implements a transparent, distributed caching system across different servers. The user requests are redirected automatically to the 'best' server. The CDN local servers do not store all the possible resources; instead, internal algorithms are used to dynamically update the local caches with the most requested contents. Internal CDN algorithms are responsible for maintaining the local caches updated.

There are mainly two types of traffic that can benefit from a CDN: live streams and Video on Demand (VoD).

Theoretically, a live streaming system should use multicast transport at IP level. By using multicast, the streaming optimal performance is a matter of finding the best resource allocation for the multicast tree, which is a routing problem. Alas, multicast is not (yet) a widespread transport system. As a consequence, application-level caching system have to replicate the networking functionalities. The cache size is, in these system, proportional to the number of served streams, as it is useless to keep data for more than a few seconds or less. Live streaming CDN systems are used to implement a multicast architecture at the application level, implementing efficient routing schemes, link failure recovery, path redundancy, etc. Although interesting, these topics will not be discussed in the present paper.

The CDNs used to deliver VoD services, on the opposite, heavily rely on local caches to minimize the network load and increase the user's QoE. In this case there is no real-time constraint, and the resource can be stored locally: like the most common web caches, the most frequently requested resources are stored locally, and the cache space is freed when a resource is too old or not used. Summarizing, the goal is to increase the user's QoE by providing a server with a more stable connection (lower load on the server, lower load on the connection between the user and the server, etc), to decrease the core network load and to minimize the resource required by the CDN system (mainly the cache size).

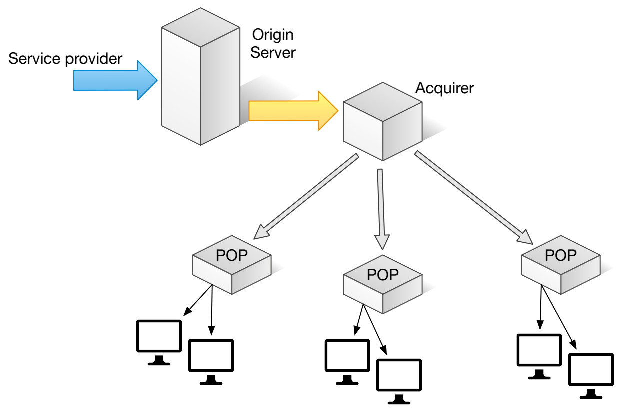


Fig. 1: Content Delivery Network architecture

III. VoD CDN architecture

The general architecture of a VoD CDN is shown in Figure 1. The *Service provider* loads the contents in the *Origin Server*. When the user tries to load the content from the *Origin Server*, the CDN automatically downloads the content through an *Acquirer* to the *Point of Presence* (POP) closer to the user. The content is delivered to the user by the POP, which acts as a transparent cache from the user's point of view (i.e., the user is not aware of its presence).

It is worth noticing that the actual video streaming system models are based on TCP rather than UDP. Moreover, streams are segmented in multiple parts, and each one is delivered to the user separately. As an example, a film is usually split in the audio and video parts, Each one is further split in chunks of different time length, and each chunk is stored with different coding rates. This seemingly complex storing system allows to serve users with different bandwidth and quality requirements (e.g., fixed users, mobile users, etc.) and to start the video playback from any point without too much performance hit.

Thanks to the above mentioned resource splitting architecture, VoD caching becomes almost similar to a 'normal' web caching problem, with the notable difference that there should be a strong correlation between the resources served to a single user.

Request-routing [11] is the technique used to redirect a resource request to the 'optimal' local server. It is worth noticing that the 'optimal' concept is subject to a number of parameters, and could include the content availability, the server load, the delay/jitter from the server to the user, etc. The discussion of the optimal choice is out of scope in this context. The interested reader can, for example, see [12].

In the present work it is important to note that, due to the service goals, the server locality is a key feature. Even though the core network is very fast, the design goal is to maintain a geographic binding between the user and the serving node. As a consequence, the network planning phase must take into account the requests dynamics and deploy the servers where they are the most effective.

Once the servers are deployed, the request routing mechanisms are used to redirect the requests to the geographically closer one, or to a backup server in case of failure (in this case the geographical location is partially dismissed).

According to [11], the request routing can be performed at DNS, transport or application level.

The most effective is the application-level redirect, which is based on an inspection of the HTTP request and redirect the user to the appropriate server through HTTP response (typically HTTP REDIRECT 302). The HTTP header inspection allows to fine-tune the redirects, taking into account the user's location (its IP is known), the resource requested, and the servers load. This is the preferred redirect system, and it is used for all the HTTP-based requests.

The DNS-based redirection the server choice is based on the DNS-proxy location, rather than on the user's location. As a matter of fact, the user does not interact directly with the authoritative DNS servers. As a consequence, the redirection is only based on a very rough geographical location (we can assume that all the users of a given area use the same DNS proxy). Moreover, in this redirect mode, the exact requested resource is unknown. As a consequence, it is also difficult to micro-manage the cache contents.

The transport-layer redirection does not add much to the DNS-based one, being only able to add the port numbers to the selection criteria. As a consequence, it is not used in the cache system we studied.

IV. Analysis methodology

The analysis has been carried out thanks to the caching system logs. These logs contains the full report of the (anonymized) requests from each user, along with the result of the caching operation and the time needed to fulfill a particular request.

The logs are simple text files, where each request is stored in a single line. The cache logs we worked with follows a format that is almost identical to the one used by the popular SQUID web cache system [13]. For each request, the following data are recorded:

1) time-received (millisecond)
2) time-to-serve (microsecond)
3) client-IP (anonymized)
4) request-description / response-status
5) bytes-sent
6) request-method
7) request-url
8) mime-type
9) request-header (User-Agent)

We will now analyze the peculiarities of the most important fields.

The first field (time-received) represents the time a request has been recorded. It is saved in milliseconds, i.e., a low time resolution. This somewhat hinder the analysis on the inter-arrival time between successive requests. As a matter of fact, in the logs there are many requests apparently recorded at the very same time, which is obviously an artifact.

The time-to-serve is the time needed by the system to process a request, i.e., to process the request, find it (locally or remotely), and build the response. It could be logical to guess that requests served by the local cache are processed faster than the ones needing a remote retrieval. However, this is not always true, as we will demonstrate in the following.

The client IP is anonymized, and we have not found evidences of multiple clients types recorded with the same IP (multiple clients types can be found by looking at the request agent). This could either means that the anonymization process also takes into account the possible NATs, or that the the users are not yet using more than one device in their household (e.g., SmartTV and tablets).

The request description and response status indicate the request kind and if the cache was able to find the content locally (e.g., TCP_HIT, TCP_MISS, etc.). Codes are available to indicate the conditional requests and other less frequent cases. Even though we have no indication that the cache is based on SQUID [13], the codes are practically identical. The hit/miss ratio is usually chosen as an indicator for the cache efficiency.

The request method is relevant for the VoD resources retrieved through HTTP. In our analysis we found that almost all the VoD requests came from HTTP clients (but not necessarily web browsers).

The request url is almost always an URL coded according to the Unified Streaming format [14]. An example is: `http://mycdn.it/prefix/video/video.ism/QualityLevels(128000)/Fragments(video=8000)`
In some cases the resource is a 'normal' one, usually pointing to programs listing pages.

The URL, minus the QualityLevels and Fragments parts, can be useful to correlate different requests incoming from the same IP (even if anonymized). We will call these requests a *flow*, as they represent the chunks of an audio/video stream that is being played by the user.

The mime-type is the most useless parameter, as it does not really reflect the resource content, only its encoding. We did not consider it in our analysis.

The request-header (User-Agent) is relevant only for statistical purposes (i.e., to find out the diffusion of a particular browser or set top box. In our case, we found that a given version of a set top box has a bad behavior, which helped fixing it. In a general context, it is possible to analyze the logs and correlate the user agents behaviours to find out software anomalies. Even though this is an interesting topic, it have not been considered in the present study and it has been left for future works.

V. Results

The log data have been processed using custom log analysis tools and the R statistical tool for distribution analysis and fitting [15].

The first check was about the QoE perceived by the users. We can use the time to serve a request as a very rough indicator. As a matter of fact, this is a simplification, because

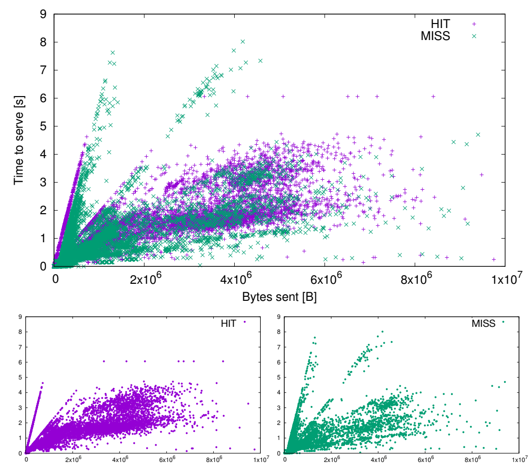


Fig. 2: Resource time to serve vs size

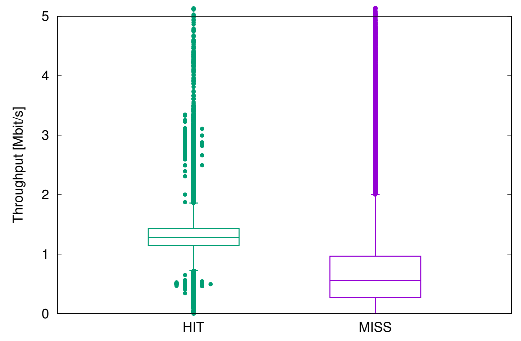


Fig. 3: Throughput for the hit and miss requests

in the VoD systems the requirement is that the next block is transferred before the previous one playout is concluded. As a consequence, the QoE is optimal if the time to serve is bounded, not if it is the lowest possible one.

We could expect that the time to serve a request from the local cache is lower than the time to serve a request that have to be retrieved from the acquirer. In Figure 2 this is not evident at all. Moreover, there's not even a straight proportional law between the time to serve and the resource size.

This result is counterintuitive, but can be explained. POPs are connected through high speed links (10 to 40 Gbps), while local caches are in SSD disks. Despite this, the SSD disks have an access time that is limited by the disk BUS, which as a limited speed, and it is slower than 40 Gbps. As a consequence, if the core network (POP, Acquirer, service provider) is not congested, the difference between local or remote retrieving is minimal.

Note that the difference is small, but still relevant. Figure 3 shows that hit requests have a slightly higher throughput than the missed ones. The low throughput could be an indication that the users are still largely using a small fraction of the CDN potentiality, and that a traffic increase is most likely.

The hit ratio is a very common method used to verify the cache effectiveness. Intuitively, the more requests are served by the cache (hit), the better. The problem is that not every resource is the same, and that the 'pure' hit ratio might be misleading in judging the cache effectiveness and the user's experience.

In order to evaluate the cache performances, we collected three kind of data: 1) the raw hit ratio, 2) the per-flow, per-user hit ratio, and 3) the per-flow hit ratio. The raw hit ratio is evaluated simply by counting the number of hits vs the number of requests. The per-flow, per-user is the average hit ratio of every flow requested by a single user. Finally, the per-flow is the average of he hit ratio of a flow regardless of the requesting user.

The three data roughly represent the overall cache efficiency, the probability that a user request of an active flow is served by the cache, and the probability that the cache efficiency is boosted by multiple users using the same content. We believe that the real parameters to be considered are the two flow-based curves. As a matter of fact, the cache system can easily forecast if a user will request a chunk of a resource according to the previous chunks it requested. Of course the user can stop watching a given content and switch to another, but if he/she does not, then the next chunk is highly predictable.

The result is shown in Figures 4a and 5a for two load cases: low and high.

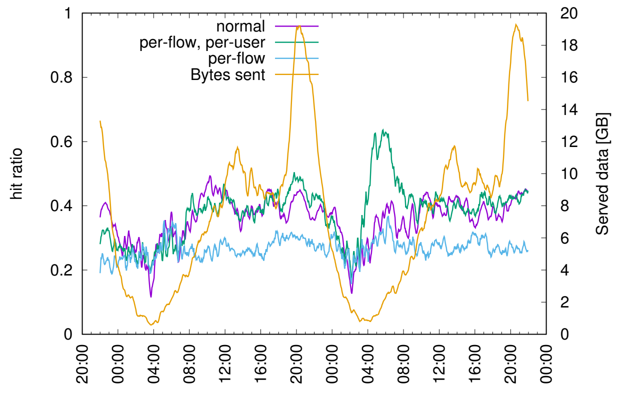
Finally, in order to evaluate the cache efficiency with respect to the resource size, all the above quantities have been also averaged according to the resource size (Figures 4b and 5b).

As a reference, the number of bytes served by the cache is drawn as well (its axis is on the right).

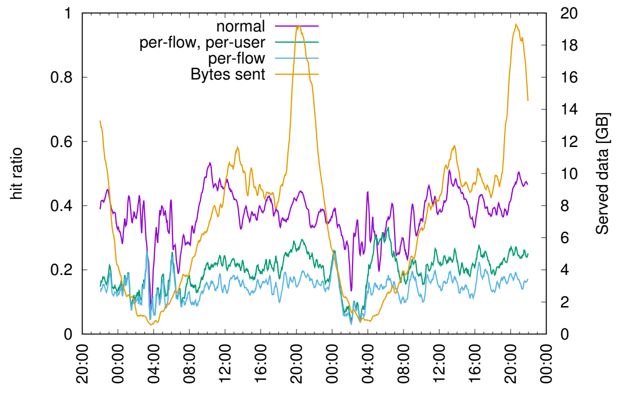
The comparison between the four graphs is very interesting. The first observation is that in the low traffic case the per-flow hit ratio is lower than the per-flow, per-user hit ratio (Figure 4). This means that the cache is not taking advantage of the users multiplexing or, in other terms, that the probability that the cache is serving the same content to multiple users is very low. On the opposite, with a high load the cache performances are boosted by multiple users, with the per-flow values often greater than the per-flow, per-user (Figure 5). Practically, this is a clear evidence that the cache in the first case is underutilized and misplaced (i.e., it is serving too few users).

Another result that can be drawn from Figures 4-5 comparison is the fact that the normalization to the resource size makes the graphs easier to read and, in general, more meaningful. Counting each resource request without considering its size does not take into account the frequent requests for the list of active channels and such. The cache effectiveness over these resources is important for the user's QoE, but it does not give a strong hint of the overall cache effectiveness. On the opposite, by normalizing over the resource size (Figure 5b), it is clear that the real cache performance is below 40%. In other terms, between 60% and 80% of the local memory (i.e., SSD disks) used by the cache is used to store useful content.

The analysis of the high load case shows a completely different scenario. Here the 'pure' hit ratio is around 60%, and the flow-based ones above 40%. Moreover, in the normalized graph, both of them are almost identical, with only two local

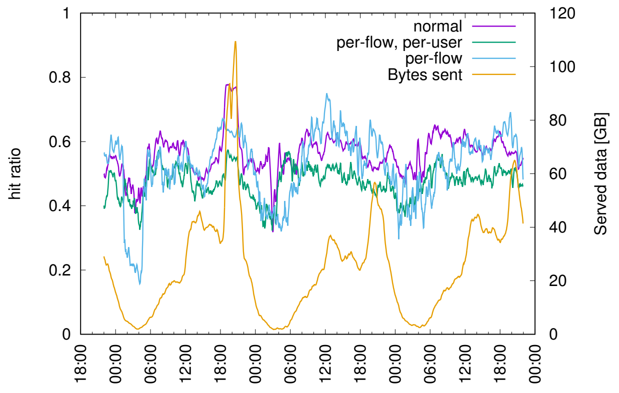


(a) Resource hit count

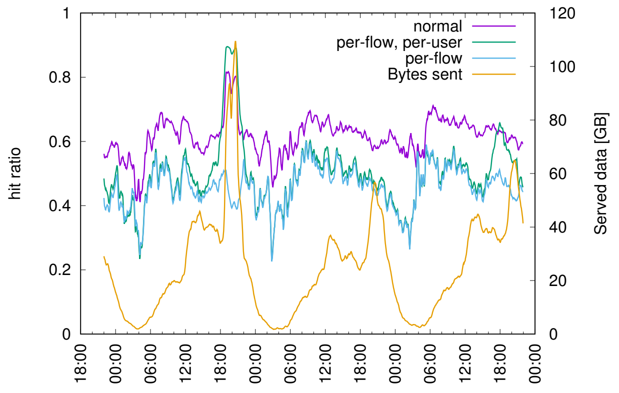


(b) Byte hit count

Fig. 4: Hit ratio - low load



(a) Resource hit count



(b) Byte hit count

Fig. 5: Hit ratio - high load

cases where they differ (perhaps due to particular events being broadcasted).

The above analysis leads to a practical, even if empirical, way to evaluate the cache effectiveness, i.e., if it is serving an optimal number of users: if the per-flow, per-user and the per-flow hit ratio, both normalized to the resource size, are almost identical, then the cache is well performing, and adding more users will not lead to further benefits.

The problem now shifts to another question, i.e., if it is possible to improve the cache performances even further. Obviously, the main element responsible for the cache optimization is the cache pruning decision algorithm and, due to the kind of cache, the cache prefetch algorithm. The first tags the resources that have to be removed in case of memory shortage, the second can preload in the cache a resource if it is going to be requested with an high probability (e.g., the next chunk of a flow).

Unfortunately, in the present study it was not possible to analyze in detail the two sub-systems (the cache system uses proprietary algorithms). Nevertheless, we have no doubts that the cache memory usage is highly optimized.

About the possible improvements, the resource pruning algorithm is (usually) quite simple, and involves a timer and, possibly, a weight to keep in memory important resources, even if their use is sporadic. The resource prefetch algorithm, on the opposite, can leverage the correlations between the chunks of the same flow.

It is well known [1] that the Internet traffic is LRD. This effect is due, in the web browsing case, to the resource size and to the request interarrival time. The main effect of the LRD behaviour is that by combining multiple flows, the overall variance does not decreases or, in other terms, the traffic have “peaks” at every aggregation size.

We can already observe in Figures 4-5 that there is a strong fluctuation in the served resource size, but this could be only a periodic effect (i.e., time of the day).

In order to further analyze the problem, we have isolated the interarrival time of the requests belonging to different flows. I.e., we did not consider the chunks of the same flow, and we measured the equivalent of a ‘reading time’. This time serie have been analyzed thanks to R.

The resource request density (Figure 6) already indicates that the interarrival time is LRD (note that the ordinate is in logarithmic scale). In order to further confirm this clue, we have drawn the Cullen-Frey graph [16] in Figure 7. Also this test leads to the conclusion that the distribution is well approximated by Weibull or Pareto distributions, and it is not short range dependant or limited.

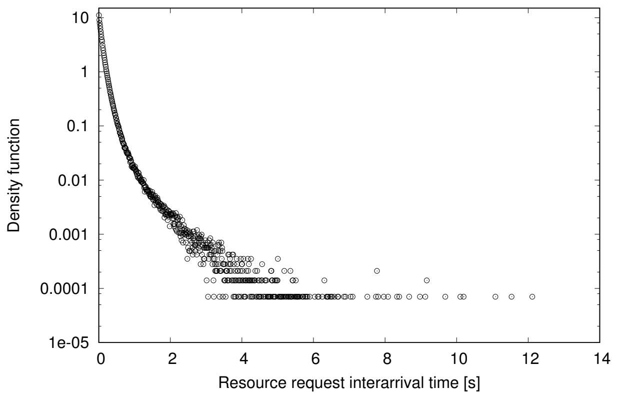


Fig. 6: Density of the resource requests interarrival time

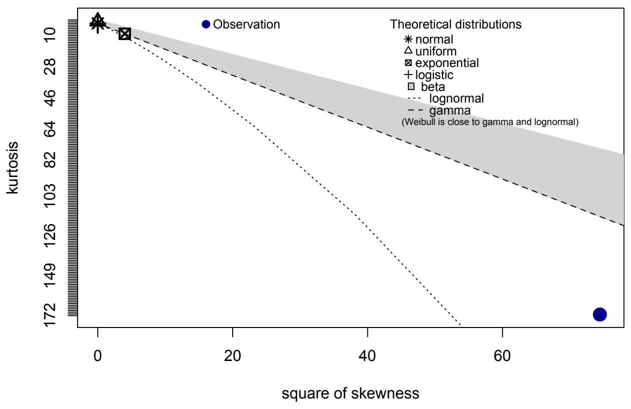


Fig. 7: Cullen-Frey graph for the requests interarrival time

The resource size analysis is a bit more complex, and it is not reported here for space constraints. It can be demonstrated that also in that case the distribution is a mixture between LRD distributions.

The conclusion is that, no matter how smart the prediction and pruning algorithms can be, the LRD effects will fix a strong limitation on the cache effectiveness.

VI. Conclusions

In this paper we analyzed the performance of a real VoD cache system. The results have shown some counterintuitive behaviours. Moreover, we demonstrated that caches serving a large user population are to be preferred, and we found a practical system to highlight if the cache is optimal or it is serving a too small number of users.

We have shown that the user traffic pattern follows an LRD distribution also in the case of VoD requests. This element provides am upper boundary to the performance limits of the CDN caches. Future works should consider these two key elements during the CDN design phase.

Last but not least, we would light to stress that the CDN performances, and even their actual work model, is strongly bound to the application-level requests inspection. Future protocols and secure protocols (e.g., HTTPS) might hinder this possibility. Due to the increasing spread of encrypted communications, we believe that the CDN architectures will have to consider this issue in the near future.

References

[1] V. Paxson and S. Floyd, "Wide area traffic: the failure of Poisson modeling," *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, pp. 226–244, Jun 1995.

[2] M. Zukerman, T. D. Neame, and R. G. Addie, "Internet traffic modeling and future technology implications," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 1, March 2003, pp. 587–596 vol.1.

[3] T. Karagiannis, M. Molle, M. Faloutsos, and A. Broido, "A nonstationary Poisson view of Internet traffic," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 3, March 2004, pp. 1558–1569 vol.3.

[4] R. Buyya, M. Pathan, and A. Vakali, Eds., *Content Delivery Networks*, 1st ed. Springer-Verlag Berlin Heidelberg, 2008.

[5] R. Lo Cigno, T. Pecorella, M. Sereno, and L. Veltri, "Peer-to-peer beyond file sharing: Where are p2p systems going?" in *Sixth International Workshop on Hot Topics in Peer-to-Peer Systems*, 2009.

[6] L. Chisci, F. Papi, T. Pecorella, and R. Fantacci, "An Evolutionary Game Approach to P2P Video Streaming," in *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, nov. 2009, pp. 1 –5.

[7] M. Day, B. Cain, G. Tomlinson, and P. Rzewski, "A Model for Content Internetworking (CDI)," RFC 3466 (Informational), Internet Engineering Task Force, Feb. 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3466.txt

[8] J. Ni and D. H. K. Tsang, "Large-scale cooperative caching and application-level multicast in multimedia content delivery networks," *IEEE Communications Magazine*, vol. 43, no. 5, pp. 98–105, May 2005.

[9] N. Kamiyama, T. Mori, R. Kawahara, S. Harada, and H. Hasegawa, "ISP-Operated CDN," in *INFOCOM Workshops 2009, IEEE*, April 2009, pp. 1–6.

[10] D. Tuncer, M. Charalambides, R. Landa, and G. Pavlou, "More control over network resources: An ISP caching perspective," in *Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013)*, Oct 2013, pp. 26–33.

[11] A. Barbir, B. Cain, R. Nair, and O. Spatscheck, "Known Content Network (CN) Request-Routing Mechanisms," RFC 3568 (Informational), Internet Engineering Task Force, Jul. 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3568.txt

[12] L. Wang, V. Pai, and L. Peterson, "The effectiveness of request redirection on CDN robustness," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 345–360, Dec. 2002. [Online]. Available: http://doi.acm.org/10.1145/844128.844160

[13] Squid-cache wiki - SquidLogs. [Online]. Available: http://wiki.squid-cache.org/SquidFaq/SquidLogs

[14] Unified streaming - player urls. [Online]. Available: http://docs.unified-streaming.com/documentation/vod/player-urls.html

[15] M. Delignette-Muller and C. Dutang, "fitdistrplus: An R Package for Fitting Distributions," *Journal of Statistical Software*, vol. 64, no. 1, pp. 1–34, 2015. [Online]. Available: https://www.jstatsoft.org/index.php/jss/article/view/v064i04

[16] A. C. Cullen and H. C. Frey, *Probabilistic Techniques in Exposure Assessment: A Handbook for Dealing with Variability and Uncertainty in Models and Inputs*. Springer, 1999.