Caching Strategies for Storage-Shared Content Delivery Networks

Ning Wang[1], Sanjay Kumar Bose[2], Gangxiang Shen[1,*],
[1]*School of Electronic and Information Engineering, Soochow University, Suzhou, Jiangsu Province, P. R. China*
[2]*Department of Electrical and Electronic Engineering, IIT Guwahati, Guwahati, INDIA*
**Correspondence email: shengx@suda.edu.cn*

Abstract: We propose three different caching strategies and develop analytical models for each strategy for a storage-shared Content Delivery Network. Simulation results show that the strategy of renewing caching lifetime on a user's request can achieve the best performance.
OCIS codes: (060.4250) Networks; (060.4256) Networks, network optimization

1. Introduction

Content distribution via networks like the Internet has been explored in recent years and both content providers and users wish to deliver and receive contents as quickly as possible [1] [2] through a content delivery network (CDN) dedicated for this. A CDN places many local servers close to users and caches contents there whenever these are downloaded from an original server (e.g., a large data center). The content delivery time can then be significantly reduced by providing content cached in a closer local server, whenever that is available, rather than getting it from the original server. Fig. 1 illustrates an example of a one-level caching structure, with one original server, multiple local servers, and many end users served by each of the local servers. The local servers are connected with the original server through a communication network, which is suitably optimized. When a user requests a specific content, its local server will provide the content if it does have it in its own cache. Otherwise, the local server will obtain the item from the original server and then give that to the requesting user, and will also cache a copy locally. When the content is available locally, then the delivery latency will be significantly shortened for the next local request. However, if the content needs to be downloaded from the original server, then the delivery latency will be much longer.

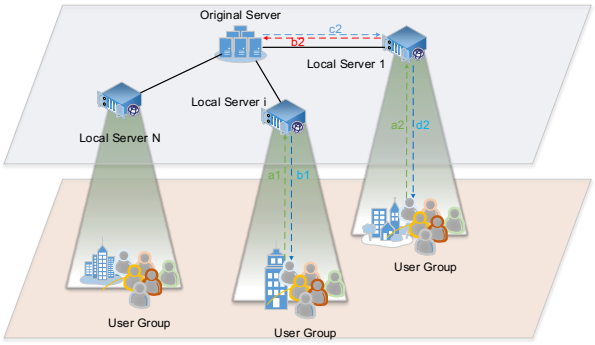


Fig. 1. CDN system model

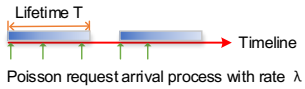


Fig. 2. Content lifetime based expiry strategy (Strategy A)

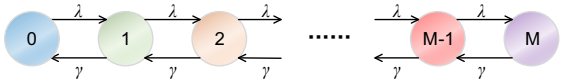


Fig. 3. The state transition diagram of a content counter (Strategy B)

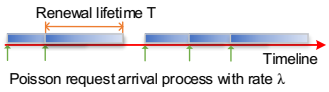


Fig. 4. Lifetime renewal based expiry strategy (Strategy C)

In designing such a network, the important issues include the storage capacity in the local server and its caching policy, and the local server's processing and bandwidth capacity. These will affect the performance of the caching strategy in terms of the delivery latency and average storage cost, and will also impact the load on the network. A commercial CDN provider providing such services will optimize the performance so that the limited infrastructure resources may be used to serve more content providers and end users. This paper mainly focuses on the caching strategy, proposing three different caching strategies based on a shared storage CDN infrastructure that can be beneficial to both the content and CDN providers. For each of the strategies, we develop analytical models to estimate the resources needed and its performance. Simulation results show that renewing the caching lifetime based on user's requests gives the best performance.

2. Caching strategies and analytical models

In this section, we present three different caching strategies. It may be noted that, in each of the following, we assume that the users of a local server generate requests for a content at rate λ, following a Poisson process.

A) Constant lifetime based expiry strategy: This is a simple strategy (Fig. 2) where a local server sets the caching lifetime to be T (constant) whenever it loads the item into an empty cache. The content would be cleared from the local server after time T. If there is a new user request for the item after it has been cleared from the local server, then it is downloaded from the original server once again and cached in the local server for another lifetime T. The probability P_{cache} of finding the item in the cache is given by (1).

$$P_{cache} = \frac{T}{T+1/\lambda} = \frac{\lambda T}{1+\lambda T} \quad (1) \qquad L = L_1 \cdot P_{cache} + L_2 \cdot (1 - P_{cache}) \quad (2)$$

Here the lifetime can be random with its mean value as T and P_{cache} can be taken as the overall mean cost of the storage in the local server since the actual storage cost can be obtained by multiplying this by a suitable scaling factor. Using (1), the average delivery latency in providing a content to a user is given by (2) where L_1 is the delivery latency from the local server to the user and L_2 is the delivery latency from the original server to the user.

B) Counter based expiry strategy: In this strategy, the local server uses a counter to record a content's state. This is initialized to 0. When the item is first downloaded to the local server upon a user request, it is incremented to 1. Each time the content is requested by a user, the counter further increments by 1 (up to a maximum value of M). The counter also simultaneously randomly decrements by 1 at rate γ (Note that the decay intervals are assumed to be exponentially distributed) until it reaches 0; at that time, the content is removed from the local server's cache. If a fresh request for this content is received after that, then it is downloaded again from the original server, cached once more and then provided to the requesting user. The state transition diagram for the counter corresponding to a particular content at a local server is shown in Fig. 3 which gives the equilibrium equations of (3), where P_j represents the probability that the content is in state j and $\sum_{j=0}^{M} P_j = 1$ from the *normalization condition*.

$$\begin{bmatrix} -\lambda & \gamma & 0 & \cdots & & 0 \\ \lambda & -(\lambda+\gamma) & \gamma & & & \\ \vdots & & \ddots & & \vdots & \\ & & & \lambda & -(\lambda+\gamma) & \gamma \\ 0 & & \cdots & 0 & \lambda & -\gamma \end{bmatrix} \cdot \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ \vdots \\ P_M \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (3)$$

$$P_j = \left(\frac{1-\rho}{1-\rho^{M+1}}\right) \cdot \rho^j \quad j = 0,1,\dots,M \quad (4)$$

$$P_{cache} = 1 - P_0 = \rho \cdot \frac{1-\rho^M}{1-\rho^{M+1}} \quad (5)$$

Using $\rho = \lambda/\gamma$ as the system load, (3) along with the *normalization condition* gives the equilibrium probability P_j of finding the counter in state j as (4) for a particular content in a local server. The probability of finding the content in the local cache is given by (5) which may be used in (2) to calculate its average delivery latency.

C) Lifetime renewal based expiry strategy: This strategy is similar to Strategy A except that if a content is cached at a local server and subsequently there is a new request from a user, then the lifetime of the content is further renewed by T (fixed) as shown in Fig. 4. We model the cache as having a **Busy-Idle** cycle where the cache is **Busy** when it has the content and is **Idle** when the cache is empty. Let B be the random variable denoting the length of the busy period and I be the random variable denoting the length of the idle period. Since the user requests follow a Poisson process with rate λ, the mean length of the idle period is given by $\bar{I} = 1/\lambda$. To find the mean $\bar{B}$, we consider the first time when the cache becomes busy (i.e., downloading the content from the original server) after an idle period. We can then obtain equation (6) and (7). The probability of finding the content in the local cache is then given by (8). We use (8) for P_{cache} in (2) to find the average delivery latency for the content.

$$B = \begin{cases} T & t > T \quad probability\ e^{-\lambda T} \\ t + B & 0 < t \le T \quad probability\ \lambda(dt)e^{-\lambda(T-t)} \end{cases} \quad (6)$$

$$\bar{B} = Te^{-\lambda T} + \int_{t=0}^{T} (t + \bar{B})\lambda e^{-\lambda(T-t)} dt = \left(\frac{1}{\lambda} + T\right) + \left(T - \frac{1}{\lambda}\right) e^{\lambda T} \quad (7)$$

$$P_{cache} = \frac{\bar{B}}{\bar{B}+\bar{I}} = 1 - \frac{1}{\lambda T + 2 + (\lambda T - 1)e^{\lambda T}} \quad (8)$$

Note that all the above models are given for caching one item in one local server. In an actual system, there can be multiple contents and many local servers. Thus, from the point of view of the whole system, we can define Average Delivery Latency (ADL) and Mean Storage Cost (MSC) for the system as follows.

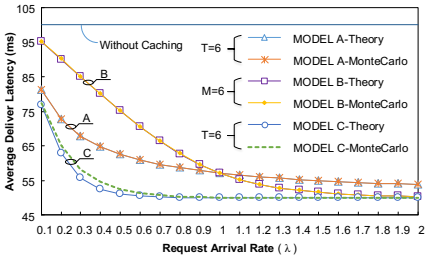
$$ADL = \frac{\sum_{a \in \boldsymbol{A}, c \in \boldsymbol{C}} \lambda_{ac} \cdot L_{ac}}{\sum_{a \in \boldsymbol{A}, c \in \boldsymbol{C}} \lambda_{ac}} \quad (9) \qquad MSC = \kappa \cdot \sum_{a \in \boldsymbol{A}, c \in \boldsymbol{C}} \phi_c \cdot P_{ac} \quad (10)$$

Here $\boldsymbol{A}$ represents the set of local servers deployed in the network. $\boldsymbol{C}$ represents the set of contents which are always available at the original server. ϕ_c represents the storage space required for content c. κ represents the storage cost per unit time. λ_{ac}, L_{ac}, and P_{ac} represent the arrival rate, the average delivery latency, and the caching probability of content c in local server a, respectively.

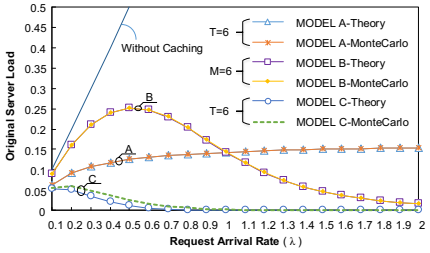
3. Simulation and discussion

Without losing generality, we set the unit time to be one hour, and the delivery latencies to be L_1=50 ms and L_2=100 ms. Each content occupies one unit of storage space. To compare the performance of the different strategies, we first consider the case with one content provider with varying Poisson request arrival rates (i.e., Case A). Then we extend to consider a more general case with multiple content providers (i.e., Case B and without losing generality, here we assume two content providers sharing the CDN infrastructure). In case A, the content request arrival rate ranges from 0 to 2.0 requests per unit time. Since the storage cost holds a linear relationship with the delivery latency, we mainly focus on the delivery latency performance here for simplicity. In addition, we also want to know how many requests will be back to the original server. For this, we define a term called the average Original Server Load (OSL) as $\boldsymbol{OSL = \lambda \cdot \left(1 - P_{cache}(\lambda)\right) \overset{\text{def}}{=} Z/Simulation\_Time}$, where Z is a counter that records the number of requests for a particular content to the local server, for which the local server has to fetch the content from the original server.

Case A: Under the assumption that there is only one content provider in the CDN, we consider the delivery latency and OSL for each strategy. Fig. 5 shows both the analytical and Monte Carlo simulation results for varying request arrival rates. The lifetimes T for strategies A and C were chosen to be 6 units. For strategy B, the maximum counter value is $M = 6$ and the counter decrements at rate $\gamma = 1.0$ per unit time.



(a) Average delivery latency



(b) Original server load (OSL)

Fig. 5. Results without original server updating (T: lifetime; M: state counter)

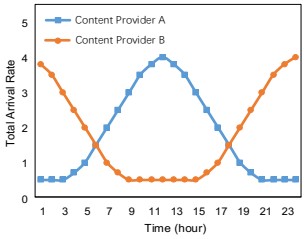


Fig. 6. Total arrival rate in different times

As shown in Fig. 5, we can see that Monte Carlo simulations almost exactly match the analytical results, thereby confirming the accuracies of the analytical models. We also observe that the mean delivery latency reduces with increasing arrival rates for all the three strategies. This means that all the three strategies can automatically cache the content with high popularity or arrival rate for a longer time at the local server to provide a lower delivery latency to the end users. In strategy A, OSL increases with increasing arrival rates and is approximately inversely related to the content lifetime T when the arrival rate is very high. In strategy B, for low arrival rates, the counter is likely to be at 0 when the next request comes and the content will then need to be downloaded again. Therefore, an increasing arrival rate when the arrival rate is small would tend to increase OSL. In contrast, when the arrival rate is high enough, the OSL tends to decrease with increasing arrival rates. Finally, in strategy C, a high arrival rate keeps on renewing the content's lifetime T before it expires, so the content will seldom be cleared from the local server and OSL tends to zero when the arrival rate is high. Moreover, strategy C is the most efficient among all the three strategies.

Case B: In this case, we consider the situation that two content providers share a common CDN infrastructure. Each content provider has 10 different items and the content popularity distribution is subject to a Zipf law (i.e., content i's $\text{ArrivalRate}(i) = \text{TotalArrivalRate} * (1/i^{\alpha})/(\sum_{i=1}^{10} 1/i^{\alpha})$, α =0.88). We also assume that the two content providers have opposite total arrival rates in different time slots as shown in Fig. 6 (This is possible under the situation that provider B is a video website with frequent visits from users after office time and provider A is a government website with frequent visits from users during office time). We also assume that each local server has a limited 10-unit storage space. We compare the performance of the proposed storage-shared strategies with the classic Least Recently Used (LRU) strategy [4] as shown in Fig. 7. The LRU strategy assigns a fixed storage space (i.e., 5 units of space for each content provider) and discards the least recently used items at a higher priority if the cache is full when needing to cache a new item. In contrast, in strategies A, B, and C, the two content providers are allowed to share the whole 10 units of storage space and the caching strategies will dynamically adjust the storage spaces to the two content providers based on their request arrival rates. We can see that strategy C achieves the best performance with both the lowest ADL and OSL, which therefore further verifies the efficiency of strategy C.

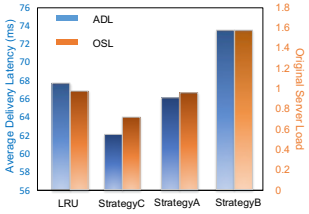


Fig. 7. LRU vs. Strategy A, B, C

4. Conclusion

We proposed three caching strategies for the storage-shared CDN. We developed analytical models to estimate the performance from the perspectives of the storage space at each local server, the average delivery latency of contents, and the original server load for the three strategies. The accuracy of those models were verified through Monte Carlo simulations and the comparison between the different caching strategies in the context of multiple content providers sharing a common CDN shows that the lifetime renewal strategy (i.e., strategy C) can achieve the best performance.

Acknowledgement: This work was jointly supported by NSFC (61671313) and Sci&Tech Achievement Trans. Proj. of Jiangsu Prov. (BA2016123).

[1] IOT connected devices to triple to 38 bn by 2020, Juniper [oneline] https://www.juniperresearch.com/press/press-releases/iot-connected-devices-to-triple-to-38-bn-by-2020.
[2] Cisco visual networking index traffic and service adoption forecasts, 2015-2020, Jun. 2017. [online] https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/qa_c67-482177.html.
[3] A. Passarella, "Review: A survey on content-centric technologies for the current Internet…," *Computer Communications*, 35(1), Jan. 2012.
[4] C. Imbrenda, *etl*, "Analyzing cacheable traffic in isp access networks for micro cdn applications via…," in Proc. ACM 2014.
[5] X. Li and G. Shen, "Optimal content caching based on content popularity for content delivery networks," in Proc. *ACP 2015*.