Content Delivery Networks – Q-Learning Approach for Optimization of the Network Cost and the Cache Hit Ratio

Conference Paper · August 2020

DOI: 10.1109/CCECE47787.2020.9255813

3 authors:

Diego Felix de Almeida
British Columbia Institute of Technology
6 PUBLICATIONS **1** CITATION
SEE PROFILE

Jason Yen
British Columbia Institute of Technology
2 PUBLICATIONS **1** CITATION
SEE PROFILE

Michal Aibin
British Columbia Institute of Technology
60 PUBLICATIONS **303** CITATIONS
SEE PROFILE

Content Delivery Networks - Q-Learning Approach for Optimization of the Network Cost and the Cache Hit Ratio

Diego Felix de Almeida, Jason Yen, Michal Aibin
Department of Computing, British Columbia Institute of Technology, Vancouver, Canada
Email: maibin@bcit.ca

***Abstract*—With an increasing demand for web content delivery, it is necessary to optimize the CAPEX and OPEX costs of the Content Delivery Networks. Ideally, all web content to be requested should be stored in local cache nodes at all times. However, the content demand varies across space and time. In this paper, we propose a Content Delivery Network model that allows us to choose the best trade-off between costs and cache hit ratio.**

***Index Terms*—content delivery networks, q-learning, caching**

I. Introduction

With the rise of content streaming services like Netflix, Spotify and Amazon Prime, the need to provide secure and cheap content is an increasing concern [1]–[3]. Although cloud services like Microsoft Azure and Amazon Web Services (AWS) make it much easier to implement a Content Delivery Network (CDN) strategy and to manage the CDN infrastructure, the CDN overall cost is still obviously relevant [4].

Ideally, in terms of response time, all content to be delivered should be in CDN local cache nodes that are close to consumers. Thus, any commercially viable CDN strategy has to take CDN costs as a limit. Furthermore, it makes sense to auto-scale the CDN infrastructure to minimize human intervention and, thus, its cost [5].

In order to reduce the OPEX costs related to CDN storage costs, the implementation of one CDN manager that can predict content requests is needed. For example, it is intuitively a good idea to distribute a new episode of a popular series right before its release; and to remove it from the CDN at some point when it is no longer requested [6]. Several different parameters can interfere with the CDN strategy: user profile, content delivery strategy, and content regional blockage rates [7], [8].

The usage of Machine Learning (ML) algorithms is a common trend to auto-scale services and seems to be an excellent approach to predict content demand [9], [10].

A. Key Insight

In this paper, we design and implement a Q-Learning approach to simultaneously optimize the network cost and the cache hit ratio. Network cost is defined as $ value per 100000 requests and we want to minimize it. On the other hand, the cache hit ratio is the ratio of the number of cache hits to the number of lookups, expressed as a percentage, and we want to maximize it as much as possible. As a secondary goal, we want to reduce the time required to find the optimal solution.

The rest of the paper is divided as follows. In Section II, we discuss related works. Furthermore, we introduce the problem, our network model and algorithms, in Section III, IV and V, respectively. We then discuss the simulation setup in Section VI. Finally, Section VII contains results, and it is followed by final remarks.

B. Contributions

The following are the main contributions of this paper.

- **Approach:** We study Content Delivery Networks using the traffic model and content based on the Netflix platform. The model is similar to the one introduced in more detail in [11].
- **Implementation:** We implement two algorithms to find the best trade-off between the network cost and cache hit ratio - an optimal approach using Brute Force and machine learning-based approach, namely Q-Learning. Using reinforcement learning, we can reduce the time of the execution and achieve results close to optimal. These two techniques are described in more detail in Section V. Our implementation has fast computation time, which means it can be easily deployed to real-time optical networks' operations.
- **Evaluation:** We evaluate our algorithms by utilizing CEONS simulator [12].

II. Related works

Usually, the content provider tends to choose the CDN that has a more significant point of presence (PoP). Arguably, the smaller, cheaper CDNs do not have comparable performance to the big ones. However, by comparing the number of PoP is not sufficient to benefit the content providers on selecting the suitable CDN [8].

Quoting the authors of [13], the CDN servers are expensive to deploy and maintain because the server capacity that can be allocated to the distribution of one media file is limited, and it incurs a non-trivial cost to the provider and/or edge users of this media file. Machine Learning algorithms can be used to produce a predictive model for every tuple area-content-format, based on monitoring data available in the Data Repository [14]. ML-based models can be estimated to predict

the evolution of the relative popularity of a group of content. Advanced monitoring concepts and machine learning tools enable local control loops allowing re-configuration to adapt resources to changing conditions [6].

Some network monitor tools can help us to collect the data, such as Real User Monitoring (RUM) and application monitoring (APM) tools. RUM tools have been used to extract several types of content and structure that are available on Facebook pages [15]. RUM also has a well-balanced and robust capability of preserving lower-order proximity while discovering and capturing higher-order network structures [16].

Several papers attempt to measure the performance of the CDNs. Authors of [17] use the DNS-based approach to evaluate CDNs. Others are employing advanced frameworks for network management [18], heuristics [19] or even decision trees for caching strategies [20].

In our approach, though, we try to minimize the cost by analyzing the interrelations of CDN total costs and cache hit ratio of the contents stored on a local cache node using the Q-Learning machine learning technique.

III. PROBLEM STATEMENT

The web content providers deploy their websites, web applications, or live streaming media on the CDN servers. Ideally, all the components, such as JavaScript files, images, and videos, should be stored in the CDN server at all times. However, the operating, storage costs and users' demand vary across space and time. It is vital to have auto-scale services to minimize the cost while maintaining the best possible performance of CDN. Our primary goal is to reduce OPEX costs of CDN by increasing the TTL of content stored on the server. By increasing TTL, we want to see an increase in the Cache Hit Ratio metric.

IV. CONTENT DELIVERY NETWORK MODEL

Our simulator emulates CDN in which the web contents are sent to the local servers to facilitate the edge users on requesting the contents with lower latency and costs. In our simulation tests, the model allows us to record the total cost of CDN as well as the cache hit ratio of the stored web content on the local cache nodes. As shown in Figure 1, the model contains a primary server and many edge users in different regions. The central server has the ability to send web content to the local cache nodes upon request. Each region has several edge users and its cache node that stores the cached version of the web content. The size and design of the network is based on Akamai network [21].

For us to abstract useful and relevant information from monitoring the data, we also identify the KPIs (Key Performance Indicators) to collect as follow:

- Cache hit ratio (CHR)
- Cost of Cache node (in $)

In the model we implemented, content is characterized using two values: size and popularity level. Each instance of content is split into several content parts, emulating different episodes of a TV show or movies with different language settings. The

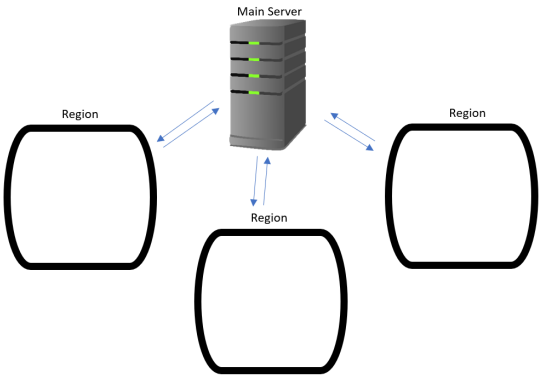

Fig. 1: Main Server and Regions

content parts have different sizes and can be stored in the cache node and requested by the users. Whenever users request them, we keep the information about which content was requested and from which region.

We assume that all requests go through the local cache node. The cache nodes can only request content from the central server, and the primary server has the ability to delete or update the content on cache nodes. The cost of each cache node is different in various regions in the network. It is also affected by the number of stored content parts on the local storage.

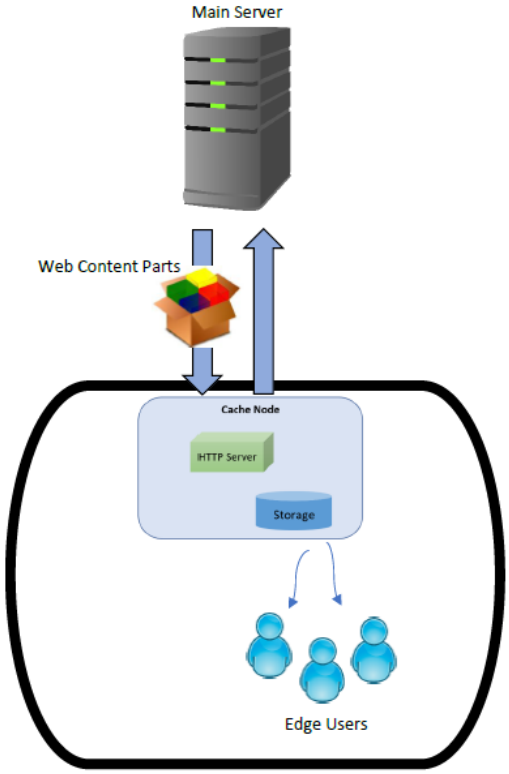

Fig. 2: Cache Node

The cache node in each region stores the content parts and has the ability to record the number of requests on the stored web content parts. The cache node sets a TTL based on the popularity of the content requested by the edge users. When a content that is not already stored on the cache node gets requested, the cache node sends a request to the main server and stores the content part on its storage with a TTL assigned to it (as shown in Figure 2). The traffic model is based on the Netflix usage with statistics about the active edge users based on the daily and weekly usage demand over time [6], similar as in [22].

V. Algorithms

As a baseline solution, we use a brute force approach to find the optimal solution [23]. This approach finds the best cost and CHR, but it is not scalable to real-life networks, due to its execution time. To overcome the problem of lengthy simulations, we implement a Q-learning algorithm. We then compare the execution time and the optimality gap.

The steps for our Q-learning approach are as follows. When the simulation starts, we assign a random TTL starting value and calculate its cost recursively. Then, depends on the reward value, we exercise the following possible actions:

- add 1 day to current TTL,
- deduct 1 day from current TTL,
- add 7 days to current TTL,
- deduct 7 days from current TTL,
- (current TTL + maximum TTL) / 2,
- (current TTL + minimum TTL) / 2.

For each simulation cycle, if the cache hit ratio with the currently assigned TTL is less than the minimum acceptable cache hit ratio (25%) or the cost is higher than the best cost we recorded so far, we assign a negative number to its reward. If the cost improvement is less than 0.05 percent, we assign 0 to its reward. Only when the reward of the new TTL is greater than 0, we proceed to exercise new actions.

VI. Simulation Setup

In our simulation, there are 1000 instances of web content in total. For each web content, we split them into several content parts in a range from 10 to 1000. Every web content has its popularity levels randomly assigned. Based on different popularity level, we assign the following chances of being requested in percentage:

- very popular - 80%
- popular - 50%
- regular - 20%
- obsolete - 5%

We perform a one-year-long simulation of a CDN (52 weeks). Once a content part reaches zero on its TTL and is not requested on that day, we remove it from the cache node and update the storage cost. For every hour in the simulation, we generate the requests and calculate the total number of requests sent by the edge users, as well as the cache hit ratio. If the content is not already in the cache node, the cache node sends a request to the main server. If a stored content gets requested, we renew its TTL.

Algorithm 1: TTL Algorithm

$minCHR = minCacheHitRatioAcceptable$;
$TTLdomain = [minTTL, maxTTL]$;
$InitialTTL = random$;

run simulation

if $CHR < minCHR$ **then**
  $reward = -1000$;
else
  if $newCost > lastCost$ **then**
    $reward = -10$;
  end if
  if $newCost > (1 - minImprovement) * lastCost$ **then**
    $reward = 0$;
  end if
  $reward = lastCost - newCost$;
end if

if $reward > 0$ **then**
  randomly choose the next action and repeat the reward calculation
end if

For the data that local cache node transfers to the edge users, we calculate the cost by the size of the content parts. Since Netflix is hosted entirely on Amazon Web Services, we use their pricing to calculate the cost of servers [24]. The following table summarizes the average cost of storage of content in various locations.

TABLE I: Average content storage price (1 GB of data per month)

	North America	Europe	Asia
First 10 TB	$0.081	$0.085	$0.114
Next 40 TB	$0.076	$0.080	$0.089
Next 100 TB	$0.054	$0.060	$0.086
Next 350 TB	$0.036	$0.040	$0.084
Next 500 TB	$0.028	$0.030	$0.080
>1000 TB	$0.023	$0.025	$0.070

VII. Results

First, we validate our Q-Learning approach against the brute force approach, checking for the CAPEX cost of the network (as seen in Figure 3), and then we see the resulting CHR (as shown in Figure 4). In our simulations we try to find the optimal TTL for every content stored in the cache nodes, as well we investigate how TTL affects the CHR. We collect the total number of requests, total cost, cache hit ratio, and calculate the cost per 100000 requests for each scenario of TTL within the 52 weeks cycle. The first observation is that

the cost drops if we increase TTL up to the level of 3-4 days, and then it just continues to increase. To understand this pattern, we look into the Figure 4, where we can observe that the CHR has a significant increase in the very first few instances of TTL increase, and then the change is not that visible. The increased CHR improves the cost, but since we store our content much longer, at a certain point, the storage costs are higher then benefits from improved CHR. It is worth noticing that Q-Learning is a fast-execution and very effective method, close to optimal.

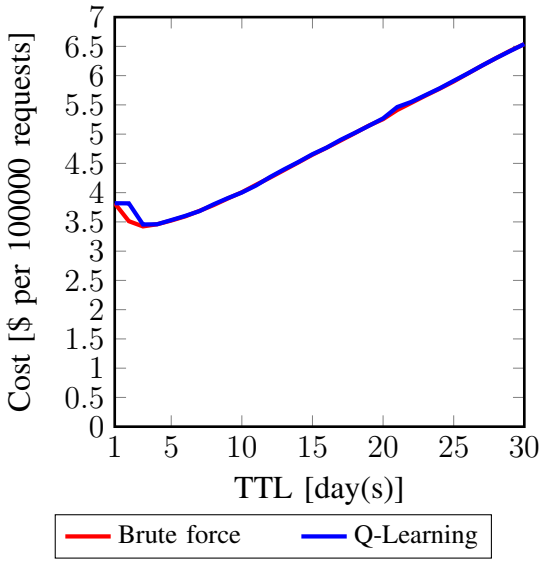

Fig. 3: Cost analysis

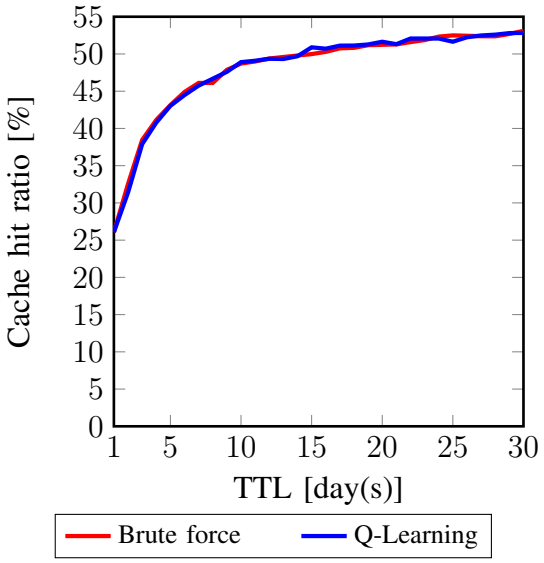

Fig. 4: CHR analysis

We then present Table II with the optimality gap analysis and Table III with the execution time analysis. Based on the aggregated data, we can state that the optimality gap was only 1.42%, and it resulted in an increased CHR of 0.64%. While the CHR improvement is not very large, the execution time is reduced 100 times (see Table III), which with a similar cost, allows to design and change the configuration of CDNs in a real-time, what is not possible for brute force

TABLE II: Q-Learning analysis

TTL	Optimality Gap - Cost	CHR Improvement
1-30	1.42%	0.64%

TABLE III: Execution time analysis

Simulation Time	Brute force	Q-Learning
In seconds	52 723	517

VIII. Conclusion

In this paper, we analyze the pros and cons of the process of increasing the Cache Hit Ratio and TTL in Content Delivery Networks. We also design and implement a Q-Learning algorithm to increase the speed of the simulation. Our simulation leads to a result that the minimum cost can be reached when the TTL is set to around 3-5 days. As the TTL increase, there is no significant growth in the cache hit ratio while the cost still goes up. For the Content Delivery Networks to be cost-effective, an auto-scaling cache node should be about to determine the best TTL to set to the stored contents where the cost is optimized and the minimum acceptable cache hit ratio is achieved. The solution presented in this paper, can be applied to different network topologies used by different network operators.

References

[1] R. Tripathi, S. Vignesh, and V. Tamarapalli, "Minimizing cost of provisioning in fault-tolerant distributed data centers with durability constraints," in *2016 IEEE International Conference on Communications, ICC 2016*, 2016.

[2] B. M. Maggs and R. K. Sitaraman, "Algorithmic nuggets in content delivery," in *Computer Communication Review*, 2015.

[3] M. Furdek, L. Wosinska, R. Goscien, K. Manousakis, M. Aibin, K. Walkowiak, S. Ristov, M. Gushev, and J. Marzo, "An overview of security challenges in communication networks," in *8th International Workshop on Resilient Networks Design and Modeling*, Halmstad, Sweden, 2016.

[4] M. Wang, P. P. Jayaraman, R. Ranjan, K. Mitra, M. Zhang, E. Li, S. Khan, M. Pathan, and D. Georgeakopoulos, "An overview of cloud based content delivery networks: Research dimensions and state-of-the-Art," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015.

[5] H. Ibn-Khedher, M. Hadji, E. Abd-Elrahman, H. Afifi, and A. E. Kamal, "Scalable and Cost Efficient Algorithms for Virtual CDN Migration," in *Proceedings - Conference on Local Computer Networks, LCN*, 2016.

[6] L. Velasco, L. Gifre, and M. Ruiz, "Autonomic Content Delivery Network Service," in *International Conference on Transparent Optical Networks*, no. 1, 2019, pp. 8–11.

[7] C. Huang, A. Wang, J. Li, and K. W. Ross, "Measuring and evaluating large-scale CDNs," *Networks*, 2008.

[8] G. Tang, H. Wang, K. Wu, D. Guo, and C. Zhang, "When more may not be better: Toward cost-efficient CDN selection," in *INFOCOM 2018 - IEEE Conference on Computer Communications Workshops*, 2018.

[9] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang, "Machine Learning for Networking: Workflow, Advances and Opportunities," 2018.
[10] M. Aibin, "Traffic prediction based on machine learning for elastic optical networks," *Optical Switching and Networking*, vol. 30, pp. 33–39, 11 2018.
[11] ——, "Dynamic Routing Algorithms for Cloud-Ready Elastic Optical Networks," Ph.D. dissertation, Wroclaw University of Science and Technology, 2017.
[12] M. Aibin and M. Blazejewski, "Complex Elastic Optical Network Simulator (CEONS)," in *17th International Conference on Transparent Optical Networks (ICTON)*, Budapest, Hungary, 2015, pp. 1–4.
[13] D. Xu, S. S. Kulkarni, C. Rosenberg, and H. K. Chai, "Analysis of a CDN-P2P hybrid architecture for cost-effective streaming media distribution," *Multimedia Systems*, 2006.
[14] Y. Bengio, "Learning Deep Architectures for AI," *Foundations and Trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.
[15] R. M. Duwairi and M. Alfaqeeh, "RUM Extractor: A Facebook Extractor for Data Analysis," in *Proceedings - 2015 International Conference on Future Internet of Things and Cloud, FiCloud 2015 and 2015 International Conference on Open and Big Data, OBD 2015*, 2015.
[16] Y. Yu, Z. Lu, J. Liu, G. Zhao, and J. R. Wen, "RUM: Network representation learning using motifs," in *Proceedings - International Conference on Data Engineering*, 2019.
[17] K. L. Johnson, J. F. Carr, M. S. Day, and M. F. Kaashoek, "Measured performance of content distribution networks," *Computer Communications*, 2001.
[18] D. Ilie and V. V. K. S. Datta, "On designing a cost-aware virtual CDN for the federated cloud," in *IEEE International Conference on Communications*, 2016.
[19] H. Khedher, E. Abd-Elrahman, H. Afifi, and M. Marot, "Optimal and Cost Efficient Algorithm for Virtual CDN Orchestration," in *Proceedings - Conference on Local Computer Networks, LCN*, 2017.
[20] D. S. Berger, "Towards lightweight and robust machine learning for CDN caching," in *HotNets 2018 - Proceedings of the 2018 ACM Workshop on Hot Topics in Networks*, 2018.
[21] Akamai, "Media Delivery Network Map," 2019. [Online]. Available: https://www.akamai.com/us/en/resources/visualizing-akamai/media-delivery-map.jsp
[22] M. Aibin, K. Walkowiak, S. Haeri, and L. Trajkovic, "Traffic Prediction for Inter-Data Center Cross-Stratum Optimization Problems," in *IEEE Internation Conference on Computing, Networks and Communication*, Maui, Hawaii, USA, 2018.
[23] T. Anantharaman, M. S. Campbell, and F. h. Hsu, "Singular extensions. Adding selectivity to brute-force searching," *Artificial Intelligence*, 1990.
[24] Amazon Web Services, "Elastic Compute Cloud (EC2) Cloud Server & Hosting – AWS," 2016. [Online]. Available: https://aws.amazon.com/ec2