Flexible Learning-based CDN Caching

Innovations and Challenges in Modern Content Distribution Network

Waleed Ahmad
Department of Computer Science
University of Saskatchewan
Saskatoon
waa392@usask.ca

Dhruv Dalwadi
Department of Computer Science
University of Saskatchewan
Saskatoon
pzd905@usask.ca

ABSTRACT

This research project, inspired by the dynamic cache management system "Darwin", [1] centers on the networking implications of adaptive CDN cache policies. In the current digital landscape, CDNs play a pivotal role in delivering content efficiently across the globe. However, static caching policies traditionally used by CDNs are increasingly inadequate due to the unpredictable nature of internet traffic, which can fluctuate due to various factors such as time of day, geographic location, content popularity, and specific events. These fluctuations challenge the network's ability to maintain high performance, minimize latency, and optimize bandwidth usage. Consequently, there's a need for CDNs that can dynamically adapt their caching strategies to changing traffic patterns, ensuring efficient resource allocation, and improving overall network performance.

1 INTRODUCTION

Content Delivery Networks (CDNs) are foundational to the architecture of the internet, designed to efficiently deliver content to users across the globe. By caching content at strategically distributed network locations, CDNs minimize the distance between the content and the end user, significantly reducing latency and enhancing the user experience. This network of servers, deployed across various geographical locations, serves as a backbone for modern digital content distribution, supporting everything from video streaming services and social media platforms to cloud computing and software downloads. The efficacy of a CDN is largely dependent on its caching strategies—how it decides which content to store, where to store it, and when to update or discard it. Effective CDN caching reduces the need to retrieve content from the original source, thereby decreasing bandwidth costs, improving response times, and scaling up to meet demands during peak traffic periods [1].

However, managing the cache in a CDN is fraught with challenges, primarily due to the dynamic nature of web content and user requests. Traditional CDN caching mechanisms often rely on static rules for cache management—deciding beforehand which content is deemed cacheable, setting fixed time-to-live (TTL) values, and employing simple eviction policies like Least Recently Used (LRU). While these methods are straightforward to implement, they do not account for the unpredictable variations in internet traffic, such as sudden spikes in demand for certain content or changes in content popularity over time. As a result, static caching policies can lead to inefficient cache utilization, where valuable cache space is wasted on infrequently accessed content, while frequently requested content is not readily available, leading to increased latency and decreased user satisfaction.

In their groundbreaking work, Jiayi Chen, Nihal Sharma, Tarannum Khan, Shu Liu, Brian Chang, Aditya Akella, Sanjay Shakkottai, and Ramesh K. Sitaraman address these challenges by introducing Darwin, a flexible, learning-based caching solution tailored for CDNs. Unlike traditional approaches, Darwin employs machine learning algorithms to dynamically adjust caching policies in response to real-time traffic patterns and content popularity changes. This adaptive strategy allows for more nuanced decision-making, enabling the CDN to prioritize the caching of content that is most likely to be requested in the near future. Darwin's methodology comprises a three-stage pipeline: initially collecting traffic pattern features, followed by unsupervised clustering for classification, and finally, neural bandit expert selection to optimize the caching policy. This sophisticated approach allows Darwin to achieve significant improvements in key performance metrics, such as hit rates and disk write operations, thereby enhancing the overall efficiency and cost-effectiveness of CDN operations.

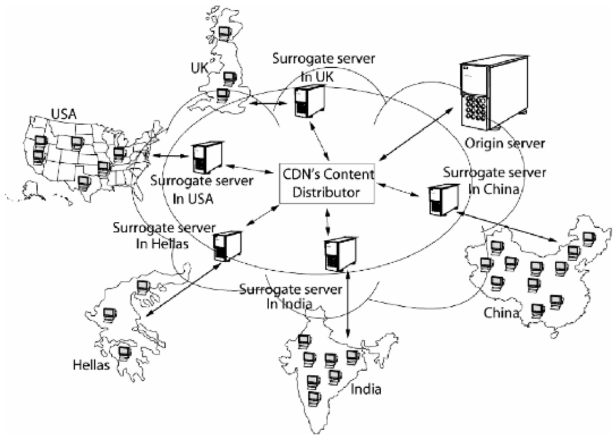
Darwin represents a significant leap forward in CDN cache management, moving beyond the limitations of static policies to embrace the complexity and dynamism of internet traffic. By leveraging the latest advancements in machine learning, Darwin offers a scalable, adaptive solution that can continuously evolve with the changing patterns of content consumption. This not only optimizes CDN performance but also sets a new standard for how modern CDNs can leverage artificial intelligence to meet the demands of the digital age.

2 BACKGROUND

1.1 Content Delivery Network (CDN)

Content Delivery Networks (CDNs) are integral to the infrastructure of the internet, designed to optimize the delivery and accessibility of web content across the globe. The efficiency of a CDN is largely reliant on its caching strategies, which are essential for managing internet traffic and ensuring the swift delivery of content.

When a user requests content, the nearest CDN server evaluates its cache for the requested data. If present, a cache hit occurs, allowing for immediate delivery. Conversely, a cache miss requires fetching the content from the origin server, increasing latency and bandwidth usage. The dichotomy of cache hits and misses underscores the critical nature of caching within CDNs, driving efforts to maximize cache hit rates through advanced management strategies.



UK
Surrogate server In UK
USA
Origin server
Surrogate server In USA
CDN's Content Distributor
Surrogate server In China
Surrogate server In Hellas
Surrogate server In India
China
Hellas
India

Figure 1: Model of a CDN

Cache Management System: The Cache Management System within CDNs employs a hierarchical structure crucial for efficient web content delivery, incorporating both Hot Object Cache (HOC) and Disk Cache (DC) layers to balance speed and capacity. The HOC, situated in main memory, is designed for rapid access to frequently requested objects, facilitating swift content delivery to end-users. Conversely, the DC offers higher storage capacity at the cost of slower access times, serving as a secondary layer for content not stored in the HOC. This bifurcated caching strategy is pivotal for managing the diverse demands placed on CDNs, ensuring that popular content is readily available while also maintaining a broader library of web content. The decision-making process regarding content admission into either cache layer or its eviction is governed by a set of policies that consider various factors, including object access frequency and size, to optimize CDN performance and resource utilization.

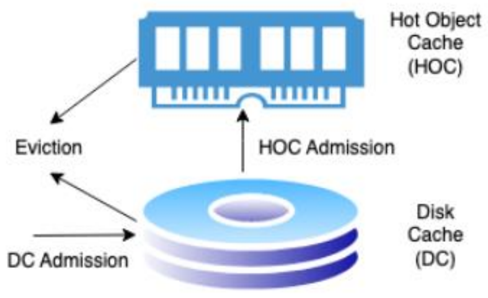


Figure 2: A two-level CDN cache serves requests with Hot Object Cache (HOC)

As CDNs evolve to accommodate the dynamic nature of internet content delivery, innovative caching strategies, such as the use of Edge Side Includes (ESI) for managing dynamic content, highlight the ongoing advancements in CDN technology. By allowing dynamic content to be delivered directly from surrogate servers through the caching of independently cacheable web page fragments, CDNs can significantly reduce bandwidth requirements for dynamic content. This approach underscores the sophisticated methodologies employed by CDNs to optimize web content delivery, ensuring they can meet the growing demand for fast, reliable, and efficient web services while managing the complexities inherent in internet content distribution. [2]

1.2 CDN Caching Algorithms Evolution.

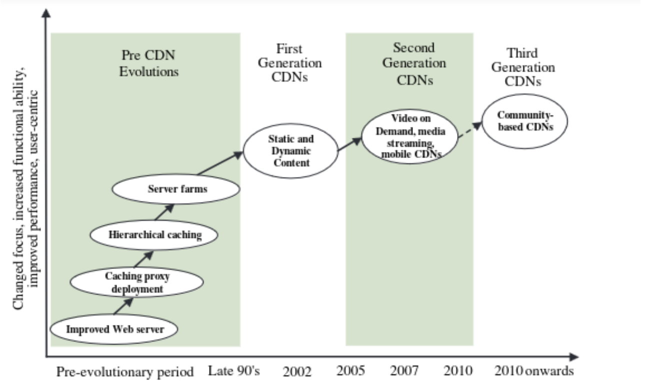


Figure 3 CDN Evolution

1.2.1 The Beginning 1990s. During the 1990s, foundational caching strategies began to shape the early development of CDNs. Among these initial strategies were the Least Recently Used (LRU) and Random Replacement (RR) algorithms. [3] The LRU approach, one of the simplest yet most effective caching algorithms, operates on the principle that the items most recently accessed are likely to be requested again in the near future. When the cache reaches its capacity, LRU identifies and removes the item that has gone the longest without access. Complementing LRU, the Least Frequently Used (LFU) method tracks the frequency of access for each item, purging those with the least hits under the assumption that items frequently accessed continue to be in high demand. In contrast, Random Replacement (RR) employs a more haphazard tactic, indiscriminately selecting and evicting items from the cache. While RR's unpredictability makes it less popular than LRU and LFU, its simplicity has merits in certain applications where complexity management is a priority.

1.2.2 Early 2000s, Expansion and Sophistication: As the internet underwent global expansion in the early 2000s, CDNs evolved to meet the increasing demands for speed and efficiency, leading to the adoption of more sophisticated strategies. A significant development of this era was the implementation of geographical caching strategies. These strategies aimed to minimize latency by storing content in cache servers that were geographically closer to the user's location, thereby reducing the distance data needed to travel [4]. Concurrently, Time-To-Live

(TTL) became a widely adopted standard for cache management. TTL provided a straightforward mechanism to maintain cache freshness, by specifying a period after which content would be considered stale and thus, be updated or removed to ensure users had access to the most current content. This period-based approach allowed CDNs to deliver content that was both fast and reliable, further enhancing the user's experience on the rapidly growing internet.

1.2.3 Mid to Late 2000s: Advancements and Diversification: In the mid to late 2000s, the landscape of online content and traffic experienced a significant surge, prompting CDNs to further refine their caching strategies. Around the mid-2000s, the concept of content popularity began to play a pivotal role in caching decisions. CDNs started optimizing their resources for frequently accessed items, ensuring that popular content was readily available, thereby enhancing user experience and reducing latency. As the decade progressed, the late 2000s witnessed the advent of segmented caching. This technique, particularly useful for video content, became crucial in addressing the unique demands of streaming media. Segmented caching allowed for more efficient delivery of video by caching individual segments of a larger file, thereby improving the performance and scalability of video streaming services to accommodate an ever-growing audience. This diversification in caching techniques showcased the CDNs' commitment to advancing technology and adapting to the evolving needs of internet content distribution. [5]

1.2.4 Late 2010s to Today: From the late 2010s up until now, the CDN landscape experienced further refinement and specialization to keep pace with the evolving internet ecosystem. One of the key advancements during this period was the integration of Adaptive Bitrate Streaming (ABR) caching. With video content dominating the online space, ABR caching emerged as an essential component for providing a high-quality streaming experience. It dynamically adjusted the quality of the video stream in real-time to match the user's current bandwidth, ensuring smooth playback under varying network conditions.[6] In parallel, predictive caching is also beginning to harness the power of big data and predictive analytics to make informed caching decisions. This approach aims to pre-emptively cache content by predicting user requests before they occurred, thereby significantly reducing latency, and improving content delivery speeds. Additionally, the era is seeing a growing emphasis on customizable content delivery policies. These policies considered a multitude of factors, such as content type, user engagement levels, and even the time of day, to manage and prioritize caching more effectively. This approach allowed CDNs to cater to the unique requirements of different content types and user demographics, providing a more tailored and efficient content delivery service.

3 Issues with Static CDN Implementations

Traditional CDN caching mechanisms, designed primarily for static content, struggle to keep pace with dynamic nature of web content, often leading to outdated content being served to users or the inability to cache content that is frequently updated. One such example is the flooding of popular news sites with requests in the wake of the September 2001 terrorist attack in the US caused serious caching problems resulting in sites becoming temporarily unavailable. [7]

Another significant challenge is the prediction of content popularity. CDN caching strategies rely heavily on accurately forecasting which content will be in high demand to pre-emptively cache it closer to the user base. However, predicting internet traffic and content popularity can be highly complex due to the unpredictable nature of user behavior and trends.

The scalability of CDNs also presents a considerable challenge. As the volume of data and the number of internet users continue to grow exponentially, CDNs must scale their infrastructure accordingly. However, scaling a CDN involves not just adding more servers or increasing storage capacity but also optimizing the geographic distribution of those resources to match user demand. This scalability issue is compounded by the need to maintain low latency and high availability, requiring sophisticated algorithms for load balancing and traffic management to ensure that the addition of resources translates into tangible performance improvements.

4 Adaptive Caching

Adaptive Caching tries to mitigate all major shortcomings of static caching techniques. The core idea behind adaptive CDN caching is to dynamically adjust caching mechanisms to optimize content delivery based on varying network conditions, content types, and user demands. This approach ensures that end-users experience reduced latency, higher bandwidth efficiency, and improved overall service quality. Recent studies and developments in this field have introduced innovative methods to enhance adaptive CDN caching.

One significant advancement is the introduction of an Adaptive Size-aware Cache Insertion Policy (ASC-IP) presented by Wang et al. [8], it aims to improve the Object Hit Ratio (OHR) in CDNs. This approach focuses on dynamically adjusting cache decisions based on the size and popularity of the content. The goal is to optimize cache space utilization, ensuring that popular and size-appropriate content is readily available to users, thus improving the efficiency of content delivery.

Another approach that further explores the role of edge computing in adaptive CDN caching presented by Aguilar-Armijo et al. is to focus on Segment Prefetching and Caching at the Edge (SPACE) for Adaptive Video Streaming[9]. By leveraging edge computing capabilities, this strategy aims to prefetch and cache video segments closer to the end-users, significantly reducing latency and buffering times for video streaming services. This approach highlights the importance of geographical proximity in CDN caching strategies, ensuring content is stored as close to the user as possible to expedite delivery and enhance the viewing experience.

The concept of a Software-Defined Content Delivery Network (SD-CDN) also plays a crucial role in adaptive CDN caching. Kumar et al. [10] proposed a model where CDN configurations and routing decisions are managed through software, allowing for more dynamic and flexible content delivery mechanisms. This model enables CDNs to quickly adapt to changes in traffic patterns, content popularity, and network conditions, ensuring optimal performance and resource utilization across the network.

Lastly, the introduction of SLAP, an Adaptive, Learned Admission Policy for CDN caching by Liu et al. [11], underscores the potential of machine learning in enhancing CDN performance. SLAP utilizes a learned model to make informed decisions on content caching, prioritizing content based on predicted popularity and caching efficiency. This approach moves beyond traditional caching strategies by incorporating adaptive learning mechanisms that continually refine caching decisions based on observed data and trends.

Another noteworthy development is Darwin, a learning-based caching strategy that employs a practical and effective approach to adaptive learning-based High-Order Caching (HOC) admission. Chen et al. [2] presented Darwin to address the limitations of traditional caching strategies by using machine learning to adaptively select which content to cache, based on real-time analysis of content popularity and network traffic patterns. This method represents a shift towards more intelligent, data driven CDN caching strategies, where decisions are made based on predictive analytics rather than static rules.

These advancements in adaptive CDN caching highlight the field's move towards more intelligent, efficient, and user-centric content delivery strategies. By leveraging machine learning, edge computing, and software-defined networking, researchers and practitioners are developing innovative solutions to meet the demands of an increasingly content-driven internet.

4.1 Exploring Darwin in Detail

Darwin's design offers an innovative approach to CDN caching by integrating a machine learning framework that is split into two distinct phases: offline training and online selection. The goal is to optimize the High-Order Caching (HOC) hit rate adaptively, with the ability to extend the methodology to other CDN objectives. Here is an in-depth explanation of each phase, as detailed in the paper 'Darwin' by Chen et al[1]:

4.1.1 Offline Training in Darwin's CDN Caching System: Offline training is the foundational pillar of Darwin's approach to optimizing CDN caching policies. This phase is methodically designed to leverage historical traffic data to create a predictive model that informs real-time CDN cache management.

4.1.2 Data Collection and Preprocessing: The initial step in the offline training phase is the collection of large datasets encompassing historical traffic traces from the operation of CDN servers. This data is rich in variety, representing a wide spectrum of request patterns, object sizes, frequencies, and temporal distributions. To facilitate effective analysis, this data undergoes preprocessing to extract meaningful features that are indicative of caching performance. Typical features include the average size of requested objects, the frequency of requests, inter-arrival times between requests, and stack distances that signify how often objects are accessed in succession.

4.1.3 Clustering of Traffic Traces: Utilizing the extracted features, Darwin employs clustering algorithms to categorize the traffic data into distinct groups. Each cluster is formed based on the similarity of traffic patterns, encapsulating specific characteristics of CDN requests. For instance, one cluster may represent traffic with frequent requests for small-sized objects, while another might represent infrequent requests for large objects. These clusters are pivotal in that they enable Darwin to predict and adapt to similar traffic patterns in the future, ensuring that CDN caching policies are tailored to the actual observed demand.

4.1.4 Expert Policy: Identification Within each traffic cluster, Darwin identifies what are referred to as "experts." These experts are essentially sets of rules or policies, characterized by frequency and size thresholds — (f, s). The thresholds dictate the conditions under which objects are promoted to HOC: an object must be requested more frequently than f and be smaller than s. These criteria are designed to optimize the cache's hit rate, which is a critical performance metric for CDNs. Experts are selected based on their historical performance, with a focus on those whose hit rates come within a specified percentage (denoted by θ%) of the optimal hit rate. This selection criterion is crucial as it ensures that only the most effective policies, as proven by historical data, are included in Darwin's repertoire.

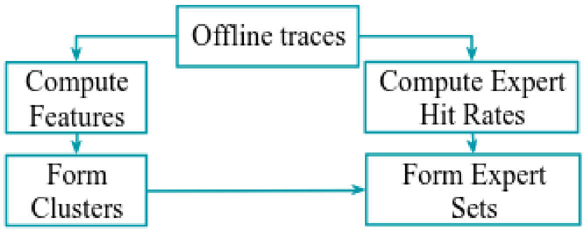


Figure 4 Darwin's Offline Clustering and Expert Sets.

4.1.5 Cross-Expert Performance: Predictors A novel aspect of Darwin's offline training is the development of cross-expert performance predictors. Recognizing that the performance of different experts on the same traffic trace is interrelated, Darwin trains neural networks to predict the performance of one expert based on the observed outcomes of another. To achieve this, Darwin extends the feature set associated with each trace with additional metrics, such as a bucketed version of its size distribution. This enriched feature set serves as input to the neural network, which then outputs conditional probabilities, such as P(Ej hit|Ei miss) and P(Ej hit|Ei hit), over the trace. The predictions are fine-tuned to the point where the neural network can reliably estimate the expected hit rate variance for each expert policy.

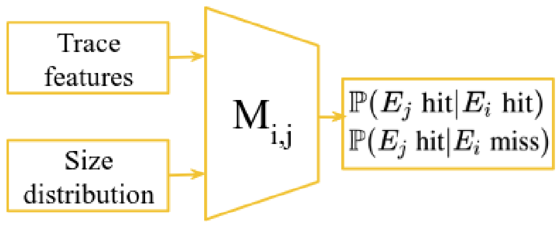

Trace features
Size distribution
$M_{i,j}$
$\mathbb{P}(E_j \text{ hit}|E_i \text{ hit})$
$\mathbb{P}(E_j \text{ hit}|E_i \text{ miss})$

Figure 5 Darwin's Cross Expert Prediction Network

4.1.6 Periodic Refinement: The offline training phase is not static; it is periodically revisited as more data becomes available. New traffic traces can lead to the identification of new patterns, the formation of additional clusters, or the adjustment of existing expert policies. This iterative process ensures that Darwin’s model remains up-to-date and reflective of the latest traffic trends, sustaining its effectiveness over time. The outcome of the offline training phase is a sophisticated mapping from traffic features to a curated set of expert policies. This mapping is instrumental during the online selection phase, where real-time traffic is analyzed, and the most suitable expert policy is deployed to manage the CDN’s cache. By preparing this predictive framework in advance, Darwin significantly reduces the complexity and computational load required during the system's live operation. In essence, the offline training phase equips Darwin with a robust and flexible knowledge base, allowing for informed and dynamic CDN caching decisions. It encapsulates a deep understanding of traffic behaviors and establishes a mechanism for predicting future demands, ensuring that CDN resources are utilized effectively and efficiently.

4.1.7 Online Expert Identification: The Online Expert Identification phase is where Darwin applies its offline training to real-time CDN operations. This phase dynamically adapts caching decisions based on live traffic, utilizing the predictive groundwork laid during offline training.

4.1.8 *Deployment and Warm-Up:* Upon entering the online phase, Darwin faces the immediate challenge of understanding and categorizing live traffic. To tackle this, Darwin deploys a warm-up strategy, using an arbitrarily chosen expert or one from a previous epoch to manage High-Order Caching (HOC) admissions temporarily. During this warm-up period, key features of the incoming traffic are estimated based on the observed requests. The empirical features gleaned from this initial set of requests enable Darwin to perform a critical matching process—aligning live traffic with the most similar cluster identified during offline training.

4.1.9 Expert Set Selection: With the traffic cluster identified, Darwin then draws from its pool of expert policies associated with that cluster. Each expert represents a potential strategy for CDN caching, and the goal is to find the one that best aligns with the current traffic's unique demands. Darwin’s selection is not random but informed by the rich dataset and neural network predictions cultivated during the offline phase. The accurate identification of the best expert is vital, as it directly impacts the CDN's caching efficiency and overall performance.

4.1.10 Best-Expert Identification Algorithm The identification of the best expert employs a multi-armed bandit-like algorithm, which, in the context of Darwin, involves sequential deployment of different experts and the collection of rewards—in this case, the HOC hit rates. Unlike traditional bandit problems where only the rewards of the deployed experts are observable, Darwin benefits from the cross-expert predictors developed offline. These predictors allow the system to generate fictitious reward samples for experts not currently deployed, providing a comprehensive understanding of potential performances across the entire expert set.

4.1.11 Reward Collection and Fictitious: Sampling As each expert is deployed over a series of requests, actual hit rates are observed and recorded. Concurrently, Darwin uses its cross-expert predictors to generate estimates (fictitious samples) of what the hit rates would have been had other experts been deployed. These fictitious samples enrich Darwin's dataset, ensuring that decisions are made on the fullest possible set of information.

4.1.12 Adaptive Selection Process: The selection of the best expert is not static; it is an adaptive process that unfolds in real time. Darwin’s algorithm must determine the best expert with a high probability, ensuring that the selected expert truly represents the optimal choice for the current conditions. This process involves a calculated balance between exploring the performance of various experts (exploration) and leveraging the known high-performance strategies (exploitation).

4.1.13 Deployment Rounds and Cache Decoupling: During online operation, expert deployment is segmented into rounds. Each round involves deploying an expert for a series of requests, known as $Nround$. The length of these rounds is carefully determined to ensure that the influence of previously deployed experts on the cache's state diminishes over time, allowing each expert's performance to be assessed on its own merit.

4.1.14 Probability-Guided: Expert Selection The expert selection during each round is governed by probability—specifically, the likelihood that deploying a particular expert will yield the best possible caching outcomes given the current traffic. This selection is not a simple matter of choosing the expert with the highest historical performance; it must consider the live traffic's unique characteristics and the predictions of the cross-expert neural networks.

4.1.15 Termination and Deployment: The online phase continues until Darwin's algorithm is sufficiently confident in the identification of the best expert. This confidence is quantified using a failure probability threshold, δ, beyond which the search for the best expert is terminated. The selected expert is then deployed for the remainder of the epoch to direct HOC admissions.

4.1.16 From Experts to Bandit: Solution At its core, the problem of expert selection in Darwin is closely akin to the Multi-armed Bandit problem but enriched with the additional structure provided by the cross-expert predictors. This additional structure allows Darwin to identify the best expert in a finite expected time, which, critically, does not scale with the number of experts, setting it apart from classical bandit solutions where learning time typically increases with the number of options.

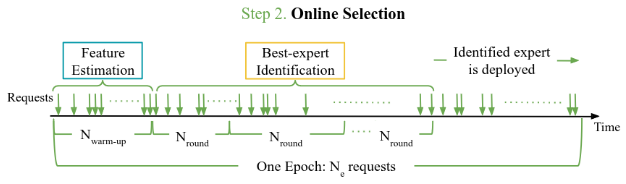


Figure 6 Darwin's Online Selection

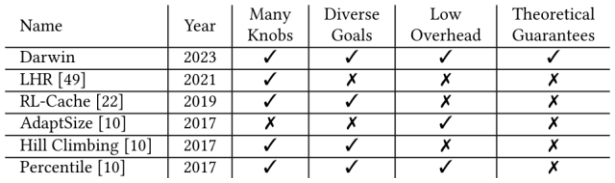
4.2 Comparison of Adaptive Caching Techniques

4.2.1 Adaptation and Flexibility: Darwin is engineered to rapidly adjust to variable traffic patterns; a capability not fully realized in static caching methods. Static models can be effective under uniform traffic but often underperform when traffic patterns change. Conversely, Darwin consistently performs well across different traffic scenarios by employing suboptimal experts during the learning phase for only a fraction of requests. This strategy allows Darwin to achieve superior performance in environments with diverse and changing traffic conditions.

4.2.2 Multiple Decision Knobs: Darwin's design includes multiple decision knobs, such as object frequency and size thresholds, enabling a multi-dimensional approach to caching decisions. In comparison, approaches like AdaptSize, [12] consider only one dimension, such as object size, for cache admissions. This singular focus can lead to suboptimal performance in scenarios where traffic is mixed with popular and unpopular objects. Darwin's consideration of multiple factors allows for a more comprehensive and effective caching strategy.

4.2.3 Diverse Goals and Theoretical Guarantees: Darwin aligns with a variety of CDN optimization goals, from maximizing Object Hit Ratio (OHR) to reducing hardware wear, accommodating diverse operational objectives without the need for substantial redesign. This contrasts with earlier methods that typically target a single, hardware-independent goal. Moreover, Darwin is underpinned by theoretical guarantees that affirm its adaptability and performance.

4.2.4 Low Overhead: A critical advantage of Darwin is its low operational overhead, essential for CDN efficiency. Other machine learning-based methods like RL-Cache [13] and LHR [14] require substantial computational resources for object-based learning. Darwin minimizes overhead by activating its learning mechanisms at specific intervals or after a set number of requests, rather than on a per-object basis. This reduces the system load and avoids the memory overhead associated with the shadow caches used in other models.



Name	Year	Many Knobs	Diverse Goals	Low Overhead	Theoretical Guarantees
Darwin	2023	✓	✓	✓	✓
LHR [49]	2021	✓	✗	✗	✗
RL-Cache [22]	2019	✓	✓	✗	✗
AdaptSize [10]	2017	✗	✗	✓	✗
Hill Climbing [10]	2017	✓	✓	✗	✗
Percentile [10]	2017	✓	✓	✓	✗

Figure 7: Cache Admission Schemes

Darwin's advanced learning-based caching methodology significantly enhances CDN performance, offering adaptability to traffic changes, a multi-dimensional decision-making framework, diverse optimization goals, and reduced system overhead. These improvements make Darwin a more precise and efficient system compared to earlier techniques, advancing the state of CDN caching technology.

4.3 Disadvantages of Darwin

4.3.1 Complexity of Model Training and Maintenance: One of the principal disadvantages of advanced machine learning systems like Darwin for CDN caching is the inherent complexity of their training and maintenance processes. The sophistication that allows Darwin to adaptively predict and adjust to varying traffic patterns also means that it is a highly complex system with numerous parameters that require fine-tuning. This complexity stems from the need to process and analyze vast amounts of data, extract meaningful features, and train predictive models that can accurately map these features to effective caching policies.

Training such models is computationally demanding; it involves running extensive simulations or processing historical traffic data to identify patterns and associations that can inform future caching decisions. The task is further complicated by the fact that CDNs operate at a massive scale, handling an enormous volume and diversity of requests. As a result, the models need to be trained on large datasets that are representative of the broad range of conditions the CDN will encounter. This training requires substantial computational resources, not just in terms of raw processing power but also in terms of data storage and management capacity to handle the input data and store the resulting models.

Moreover, the CDN environment is dynamic, with traffic patterns that evolve over time due to changes in user behavior, the introduction of new content, and shifts in popular trends. To remain effective, the machine learning models must be periodically retrained or updated to reflect these changes, which can be a resource-intensive process requiring ongoing investment in computational infrastructure and skilled personnel to manage and execute these updates.

4.3.2 Risk of Overfitting: Another critical disadvantage is the risk of overfitting. Machine learning models, by their very nature, are designed to identify and leverage patterns in the data they are trained on. While this can be highly effective when past conditions are a good predictor of future events, it can also lead to models that are too finely tuned to the specific idiosyncrasies of the training data, failing to generalize to new conditions. For CDN caching, where the goal is to anticipate and respond to future requests, a model that is overfitted to historical data may not perform well when user behavior changes or when new types of content become popular. The challenge is exacerbated by the high dimensionality of CDN data and the subtle interdependencies between data features, which can lead to models that capture noise as if it were a signal, mistaking random fluctuations for meaningful trends.

Combatting overfitting requires careful model design, including the use of regularization techniques, cross-validation, and other statistical methods to ensure that the model's complexity is appropriate for the patterns in the data. It also requires rigorous testing against independent datasets to ensure that the model's predictive power holds up under conditions it hasn't been directly trained on. This can be a delicate balance to strike, as models that are too simple may fail to capture important patterns, while those that are too complex may become mired in the minutiae of the training data.

4.3.3 Dependency on Quality of Data: The third significant disadvantage is the dependency on the quality of the data. Machine learning models are only as good as the data they are trained on. In the context of CDN caching, the data must accurately reflect the diversity and variability of real-world CDN requests. If the training data is incomplete, biased, or otherwise unrepresentative, the models may develop skewed or inaccurate perceptions of traffic patterns, leading to caching policies that do not serve the actual needs of end-users.

For example, if the data is predominantly from periods of low traffic, the model may not learn how to handle high-traffic events effectively. Similarly, if the data over-represents certain types of content or user behaviors, the model's caching policies may be biased towards those cases, at the expense of others. Ensuring data quality involves not just collecting large quantities of data but also ensuring that the data collection process is designed to capture a broad and representative sample of CDN traffic. This can involve technical challenges, such as how to efficiently sample traffic at scale, and methodological challenges, such as how to account for potential sources of bias in data collection and preprocessing.

5 Conclusion

While machine learning based CDN caching systems like Darwin present a leap forward in adaptive caching technology, they also bring with them challenges in training complexity, the risk of overfitting, and data dependency. Addressing these challenges requires careful consideration of the model design and training process, ongoing investment in computational resources, and a commitment to maintaining high data quality standards.

ACKNOWLEDGMENTS

We extend our sincere gratitude to the University of Saskatchewan (USask) for their support and resources. A special thank you to Professor Dwight Makaroff for his invaluable mentorship and guidance throughout the development of this project.

REFERENCES

[1] Chen, J., Sharma, N., Khan, T., et al. 2023. Darwin: Flexible learning-based CDN caching. Proceedings of the ACM SIGCOMM 2023 Conference.

[2] Wang, Y., Dai, H., Han, X., Wang, P., Zhang, Y., and Xu, C.-Z. 2021. Cost-driven data caching in edge-based content delivery networks. IEEE Transactions on Mobile Computing, 1–1.

[3] Pallis, G. and Vakali, A. 2006. Insight and perspectives for Content Delivery Networks. Communications of the ACM 49, 1, 101–106.

[4] Pathan, M. and Buyya, R. A taxonomy of cdns. Content Delivery Networks, 33–77.

[5] La, H.-L., Tran, A.-T.N., Le, Q.-T., Yoshimi, M., Nakajima, T., and Thoai, N. 2020. A use case of content delivery network raw log file analysis. 2020 International Conference on Advanced Computing and Applications (ACOMP).

[6] Ghabashneh, E. and Rao, S. 2020. Exploring the interplay between CDN caching and video streaming performance. IEEE INFOCOM 2020 - IEEE Conference on Computer Communications.

[7] Vakali, A. and Pallis, G. 2003. Content delivery networks: Status and trends. IEEE Internet Computing 7, 6, 68–74.

[8] Wang, P., Liu, Y., Zhao, Z., Zhou, K., Huang, Z., and Chen, Y. 2022. Adaptive size-aware cache insertion policy for content delivery networks. 2022 IEEE 40th International Conference on Computer Design (ICCD).

[9] Aguilar-Armijo, J., Timmerer, C., and Hellwagner, H. 2023. Space: Segment prefetching and caching at the edge for adaptive video streaming. IEEE Access 11, 21783–21798.

[10] Kumar, A., Banerjea, S., Jain, R., and Pandey, M. 2022. Software-defined content delivery network at the edge for adaptive video streaming. International Journal of Network Management 32, 6.

[11] Liu, K., Wu, K., Wang, H., Zhou, K., Zhang, J., and Li, C. 2023. Slap: An adaptive, learned admission policy for content delivery network caching. 2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS).

[12] Berger, D. S., Sitaraman, R. K., and Harchol-Balter, M. Adaptsize: Orchestrating the hot object memory cache in a content delivery network. In 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)
(2017), pp. 483–498.

[13] Kirilin, V., Sundarrajan, A., Gorinsky, S., and Sitaraman, R. K. Rl-cache: Learning-based cache admission for content delivery. In Proceedings of the 2019 Workshop on Network Meets AI & ML (2019), pp. 57–63.

[14] Yan, G., Li, J., and Towsley, D. Learning from optimal caching for content delivery. In CoNEXT ’21: The 17th International Conference on emerging Networking Experiments and Technologies, Virtual Event, Munich, Germany, December 7 - 10, 2021 (2021), G. Carle and J. Ott, Eds., ACM, pp. 344–358.

A Appendix

Dhruv researched existing static CDN caching techniques, delving into the literature to gather relevant resources and insights. He reviewed academic papers and industry reports to compile a comprehensive understanding of traditional CDN caching methodologies. Additionally, Dhruv actively participated in the collaborative process of compiling the report, contributing valuable insights and suggestions during discussions.

Waleed took charge of researching dynamic CDN caching, exploring innovative approaches and emerging trends in the field. Did an in-depth investigation into dynamic caching techniques provided crucial insights into the evolving landscape of content delivery networks. Waleed actively engaged in the process of compiling the report, synthesizing research findings, and integrating them into the overall narrative.