Malaysian Journal of Computing, 11 (1): 2447-2468, 2026

eISSN: 2600-8238

LEAST RECENTLY USED (LRU) CACHING POLICY BASED ON NAÏVE BAYES ML ALGORITHM IN COLLABORATIVE PEER-TO-PEER SYSTEMS FOR NETWORK BANDWIDTH UTILIZATION

Waheed Yasin[1*]

[1*]*Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, 40450 Shah Alam, Selangor Darul Ehsan, Malaysia*

[1*]waheedos80@yahoo.com

(indicates the corresponding author)*

ABSTRACT

Web caching offers several advantages, such as increasing cache hit rates, lowering the workload on origin servers, and minimizing network traffic. Nevertheless, limited cache capacity poses a major challenge in web caching systems. Moreover, repeatedly fetching same media objects from origin servers leads to unnecessary bandwidth consumption. Furthermore, traditional caching policies, including Least Recently Used (LRU), are vulnerable to cache pollution. This study introduces a collaborative caching policy based on the Naïve Bayes (NB) Machine Learning (ML) algorithm. The proposed policy exploits structured peer-to-peer architectures, allowing cache contents to be shared among peers to improve the efficiency of LRU web caching policy. Performance evaluation is conducted through simulations using two real-world datasets obtained from YemenNet Internet Service Provider (ISP) and the IRCache network. The results show that the proposed policy outperforms the traditional LRU policy in terms of Hit Ratio (HR), Byte Hit Ratio (BHR), and Cost Throughput (CT). For example, in some cases the Improvement Ratio *(IR) of is more than 12% for YemeNet dataset; while it is* more than 24% for *IRCache dataset.*

Keywords*: Least Recently Used, Machine Learning, Media Objects Caching, Naïve Bayes, Peer-to-Peer Systems*

Received for review: 15-01-2026; Accepted: 31-03-2025; Published: 01-04-2026

DOI: 10.24191/mjoc.vo11i1.10055

1. Introduction

A cache replacement policy defines the action taken when the cache reaches its capacity and new web objects need to be stored. Cache replacement plays a key role in web caching systems. Collaborative caching enables more effective use of cache space by allowing neighboring peers within a network to share cached web objects. In addition, collaborative caching reduces response latency, as user requests can be served from nearby caches instead of being retrieved from the original server. Moreover, this approach improves the Hit Ratio (HR), Byte Hit Ratio (BHR), and Cost Throughput (*CT*) (Ibrahim et al., 2016; Yasin, 2018; Pires et al., 2021; Jialu et al., 2022; Li & Haichao, 2022).

The cache replacement decision, which is based on the entry time, location, frequency, last access, and expiration time of the objects in peers' caches, is one of many difficulties that must be taken into account when collaborative caching is used in the system. Other issues of collaborative caching include network topology, varied cache sizes, and user request patterns. However, there are numerous other issues that need to be considered when implementing a peer-to-peer system, including peer-to-peer communication overhead, searching mechanisms, and the effects of peers joining or departing the excessively large network. Additionally, the Quality of Service (QoS) in terms of delays for cached objects must be considered. The time it takes for peers' caches to respond to a query is known as the delay. Furthermore, one of the issues of caching in peer-to-peer systems is updating and maintaining cached objects. The media objects that are far larger than text objects are the subject of this study. A thesis that included portions of this study was already published at: http://psasir.upm.edu.my/id/eprint/76955/1/FSKTM%202018%2065%20-%20IR.pdf.

Recent research indicates that media items are the most common type of Internet traffic, hence cache size must be taken into account when caching is necessary. For instance, between 65% and 80% of Internet bandwidth is consumed by media streaming (Esakki et al., 2021). However, standard web caching strategies like LRU are hampered by caching pollution, which occurs when media objects stored int the storage of the cache are not regularly accessed, hence reducing web caching performance. Additionally, repeatedly downloading the same video content from the original server wastes network capacity (Ali et al., 2012a, 2012b).

Machine Learning (ML) improves the performance of algorithms in computer science (Ibrahim et al., 2024; Ramli et al., 2025; Saxena et al.,2026). However, several variations of caching policies have addressed cache pollution issue, they did not consider the contents of neighbors' caches in the network (Pernabas et al., 2019). Collaborative approach adopted in this study is based on the benefits of Naïve Bayes (NB) ML algorithm. The proposed approach considers peer-to-peer systems' advantages.

The motivation of this study is to reduce the request response time by improving the HR, BHR and *CT* of LRU web caching policy. This study focuses on LRU because it is the most popular web caching policy in practice which has many desirable characteristics such as high HR, low complexity, and flexibility (Kushwah et al. 2022). On the other hand, HR, BHR, and *CT* are most important metrics to measure the performance of any caching policy (Yasin, 2018). The novelty of this study can be summarized in more performance analysis of the proposed approach which contributes to web caching based on LRU using NB ML algorithm in collaborative peer-to-peer systems. This policy focuses on media objects because, as it has been mentioned earlier, the dominant traffic is media.

The structure of this document is as follows: Related works are shown in Section 2. The proposed approach is presented in Section 3. Section 4 presents the experiment. Section 5 presents the results and discussions. The computation performance is presented in Section 6. Section 7 presents the conclusion and future works.

2. Related Works

In this section, recent related works of collaborative caching are presented. In addition, recent related works for caching based on ML are presented.

Yasin et al. (2018) proposed web caching policies using C4.5 ML algorithm for peer-to-peer systems to alleviate caching pollution, such as IC-J48-LRU. The performance of proposed policies has been evaluated by conducting trace-driven simulations. Furthermore, the results have been compared to LRU, LFU, GDS and state-of-the-art policies; however, computation performance has not been studied. On the other hand, IC-NB-LRU was presented in (Ibrahim et al., 2016) which was evaluated using only HR and BHR in general as

performance metrics and compared to LRU and NB-LRU which might be considered as an unfair comparison. In this study, IC-NB-LRU is improved to C-NB-LRU where it is evaluated using Local Hit Ratio (*LHR*), Local Bit Hit Ratio (*LBHR*), Global Hit Ratio (*GHR*), Global Bit Hit Ratio (*GBHR*), and *CT*. Furthermore, C-NB-LRU is compared to C-C4.5-LRU which is an improved version of IC-J48-LRU. Moreover, computation performance is studied.

In (Lanying et al., 2022), a web caching approach based on minimum frequency policy and flow load algorithm has been presented to solve the problem of network congestion. A combination of two systems, including network caching and traffic identification have been used. The proposed approach has improved HR in peer-to-peer systems. Also, it utilizes network bandwidth.

An approach which is called Peng-Robinson-Stryjek-Vera (PRSV) Multi-Criteria Group Decision-Making (MCGDM) was proposed by Peng et al. (2022) to achieve an optimal cache replacement policy which is based on Pythagorean fuzzy set. The results have demonstrated that PRSRV MCGDM can reduce the cache redundancy.

Sharif et al. (2022) proposed a sliding window-based adaptive cache replacement policy, which was capable to change its behavior according to changes in data popularity over time. The proposed policy was based on online learning using a neural network. The results demonstrated that the proposed approach was robust to data popularity changes.

In (Xu et al., 2022), a caching strategy which is called Learning based Cooperative Caching Strategy (LECS) has been presented for reducing the content delivery latency based on cooperative edge servers. A popularity prediction approach has been used to improve the HR according to the simulation results.

An edge caching approach based on ML for video streaming in 5G networks has been presented in (BenMimoune, 2023). The proposed approach has combined edge caching and ML to optimize video streaming in terms of HR and Request Latency (RL); however, no details of the ML algorithm that has been implemented in the approach were presented.

An approach called KNN-LRU has been presented in (Negara et al., 2024) which implemented kNN ML algorithm on routers in Named Data Networking (NDN) for cache replacement by predicting popular contents. NDN router stores content; however, the limited cache capacity is considered as a limitation of NDN. KNN-LRU is better than traditional LRU, FIFO, and LFU using HR, RL, and bandwidth utilization performance metrics.

In (Saxena et al., 2024), a caching approach called Latency Aware Dynamic Caching (LAD-Caching) has been presented. LAD-Caching has applied Long Short-Term Memory (LSTM) (Li et al., 2022) policy with deep learning to manage cache contents. Furthermore, Access Time (AT), RL, HR, and cache utilization have been used as performance metrics; however, it is limited for datalakes cloud services.

A caching approach which called Catcher+ has been presented in (Zhou et al., 2024), which improved HR. Furthermore, it reduced RL and traffic to servers. Catcher+ used Deep Reinforcement Learning and LSTM algorithm; however, it was limited to cache replacement in block storage clouds and did not consider prefetching policies. Also, it has only been compared to traditional caching polices such as LRU and LFU.

A caching scheme called Learning Machine-based Mobility and Popularity Aware Proactive (E-MAPP) has been presented in (Ahmad et al., 2025). It considered movement of the user and popularity of contents in 5G and 6G Multi-Tier Cellular Networks (MTCNs). E-MAPP has applied Extreme Learning Machine (ELM) to address the problem of cache pollution. On the other hand, RL and HR have been used as performance metrics to measure the effectiveness of E-MAPP in enhancing QoS and user satisfaction.

In (Deepakraj et al., 2025), an approach for caching has been presented, which utilized ML and Reinforcement Learning to optimize cache replacement by predicting popular contents. The proposed approach increased HR by using deep learning and LSTM algorithm. It also

reduced the load on Content Delivery Networks (CDNs) and servers. Table 1 presents a summary of the proposed web caching policies using ML algorithms.

In this study, a novel web caching approach based on NB supervised ML algorithm for structured peer-to-peer systems is proposed to alleviate caching pollution.

3. Proposed Approach

In this section, peer-to-peer systems' media object caching infrastructure based on Chord protocol (Stoica et al., 2001) is presented. After that the implementation of NB ML is presented.

Table 1. A summary of proposed web caching policies using ML algorithms

Approach Author(s) / Year	Machine Learning Algorithm	Performance Metrics	Advantages	Disadvantages
Lanying et al. (2022)	Flow Load	HR	- It utilized servers and network bandwidth	- It depended on the number of users
Peng et al. (2022)	Pythagorean fuzzy set	HR, Distance	- It reduced the cache redundancy.	- It is dedicated for Content- Centric Networks (CCN)
Sharif et al. (2022)	Neural Network	HR	- It was robust to data popularity changes.	- It was dedicated for IoT-CCN
Xu et al. (2022)	LECS	HR	- It reduced delay	- It was dedicated for wireless network traffic
BenMimoune (2023)	Not mentioned	HR, RL	- It combined edge caching and ML to optimize video streaming	- It was dedicated for 5G Networks.
Negara et al. (2024)	kNN	HR, RL	-It predicted popular contents	-It was dedicated for NDN.
Saxena et al. (2024)	Deep Learning	AT, RL	-It used LSTM to manage cache contents. -It reduced AT & LR.	-It was limited for data lakes cloud services.
Zhou et al. (2024)	Reinfo--rcement Deep Learning	HR, RL	- It used LSTM algorithm to improve HR. - It reduced RL and traffic to servers.	-It was limited to cache replacement in block storage clouds. - It did not consider prefetching policies.
Ahmad et al. (2025)	ELM	HR, RL	- It considered movement of the user and the popularity of contents.	-It was dedicated for 5G and 6G MTCNs.
Deepakraj et al. (2025)	Reinfo--rcement Deep Learning	HR	- It used LSTM algorithm to optimize cache replacement by predicting popular contents.	- It was dedicated for CDNs.

3.1 Media Object Caching Model

In media object caching system, peers collaborate and coordinate with a proxy to provide a scalable cache space and contribute to providing a media object caching service to end-users. The proxy and all peers are the main components of media object caching system, where it employs a proxy for fetching requested media objects from the origin server if they are requested for the first time.

Each peer in media object works in the same way as in other systems (i.e. as a client when it requests media object, or as a server when it streams media object to other peers). The system model consists of a number of peers $N= n_1, n_2, ...n_N$ where each peer has three components that are: (i) Cache Manager; this component has two tasks including answering the requests and executing the replacement policy. In the first task, when a user requests a media object, the Cache Manager checks the availability of the media object in the local cache. If it is available, the Cache Manager responds to the user with the media object. If it is not available in the local cache, the Cache Manager checks the neighbors' caches. Finally, if the media object is not available in the neighbors' caches, it is fetched from the original server. In the second task, the Cache Manager is responsible for cache replacement policy when the cache is full. (ii) Distributed Hash Table. (DHT); this component is responsible to keep the peer's and media object's identifiers using Chord ring (Stoica et al., 2001). (iii) Local Cache; this component is responsible to cache media objects and provides Cache Manager with the list of cached items.

The proposed media object caching system model employs Chord as the overlay structure because of its flexibility. Furthermore, it allows the best selection of serving peer according to its capacity and workload. Also, its distributed indexing is scalable and does not form a single point of failure. As a result of that, the workload is distributed over the media object caching system. Thus, the media object caching system is considered as a cost-effective system (Yasin, 2018).

3.2 NB Supervised ML Algorithm Implementation

The proposed approach is based on NB-ML algorithm because it is one of the fastest and the easiest algorithms for predicting a class of datasets (Mohanty et al., 2022). All attributes in NB classifier are assumed to be conditionally independent, given the class label. The term naïve originates from the idea of independence among attributes, disregarding any correlations between them.

An observed pattern is to be assigned to a class where the NB classifier relies on a probability calculation known as posterior probability. The evaluation of class posterior probability obtained from a test sample d is what is known as the classification decision, as demonstrated in Equation (1). All equations pertaining to NB classifier are presented in Friedman et al. (1997). The sample *d* is allocated to the most probable class.

$$P\,(C = c_j|d) \tag{1}$$

In this study, the attributes for YemenNet dataset are *Time, Client ID, URL ID, Length, Delta Time, Freq*. On the other hand, *C* is the class attribute where the values of |*C*| are 0 and 1. Assume a test sample $d = < F_1 = Time = f_1, F_2 = Client\_ID = f_2, F_3 = URL\_ID = f_3, F_4 = Length = f_4, F_5 = Delta\ Time = f_5, F_6 = Freq = f_6 >$, where f_i is a possible value.

In IRCache dataset, the attributes are Timestamp, Client ID, URL ID, Size, Elapsed Time, Freq. On the other hand, *C* is the class attribute where the values of |*C*| are 0 and 1. Asume a test sample $d = < F_1 = Timestamp = f_1, F_2 = Client\_ID = f_2, F_3 = URL\_ID = f_3, F_4 = Size = f_4, F_5 = Elapsed\_Time = f_5, F_6 = Freq = f_6 >$, where fi is a possible value as shown in Equation (2) and Equation (3).

$$P(C = c_j|d) = \frac{P(d|C = c_j)}{P(d)}$$

$$= \frac{P(d|C = c_j)P(C=c_j)}{\sum_{k=1}^{2} P(d|C = c_k)P(C=c_j)} \quad (2)$$

$$P(d|c_j) = \prod_{i=1}^{6} P(F_i = f_i|C = c_j) \quad (3)$$

From Equation (3) and Equation (4), k = [1,2] because we have two cases 0 or 1; while *i*= [1,6] because we have 6 attributes for both datasets. All attributes are assumed conditionally independent that given the class $C = c_j$ as demonstrated in Equation (4). The decision function is obtained as shown in Equation (5).

$$P(C = c_j|F_1 = f_1, F_2 = f_2, \dots, F_6 = f_6) = \frac{P(F_1 = f_1, F_2 = f_2, \dots, F_6 = f_6|c_j)}{P(F_1 = f_1, F_2 = f_2, \dots, F_6 = f_6)}$$

$$= \frac{P(F_1 = f_1, F_2 = f_2, \dots, F_6 = f_6|C = c_j)}{\sum_{k=1}^{2}(P(F_1 = f_1, F_2 = f_2, \dots, F_6 = f_6|C = c_k)P(C = c_j))} \quad (4)$$

$$P(C = c_j|d) = \frac{P(C = c_j)\prod_{i=1}^{6} P(F_i = f_i|C = c_j)}{\sum_{k=1}^{2} P(C = c_k)\prod_{i=1}^{6} P(F_i = f_i|C = c_k)} \quad (5)$$

Since the denominator is the same for every class in classification tasks, the numerator of Equation (6) is only required to determine which class has the highest probability for each sample. Thus, we can use Equation (6) to determine the most likely class for a given test sample. The $arg \max_{x \in S} f(x)$ can be defined as the set of elements in *S* that achieve the global maximum in *S*, which is defined as the largest overall value of a set or absolute maximum as shown in Equation (7) (Alan, 1979).

$$arg \max_{x \in S} f(x) = \{x \in S: f(x) \max_{y \in S} f(y)\} \quad (6)$$

$$c = arg \max_{c_j} P(C = c_j)\prod_{i=1}^{6} P(F_i = f_i|C = c_j) \quad (7)$$

Only the prior probabilities $P(C = c_j)$ and the conditional probabilities $P(F_i = f_i | C = c_j)$ must be estimated from the training datasets during the NB classifier's training phase using Equations (8) and (9).

$$P(C = c_j) = \frac{No.of\ samples\ of\ class\ c_j}{Total\ No.of\ samples} \quad (8)$$

$$P(F_i = f_i|C = c_j) = \frac{No.of\ samples\ with\ F_i = f_i\ and\ class\ c_j}{No.of\ samples\ of class\ c_j} \quad (9)$$

Then, using Equation (10), the class of the requested media object *c* can be predicted by the NB classifier, assigning it to the most likely class, either 1 or 0.

$$c = \begin{cases} 1, & if\ the\ object\ will\ be\ revisited \\ 0, & otherwise \end{cases} \quad (10)$$

3.3 Collaborative LRU based on NB

Of all the caching policies, the LRU policy is the most often used (Yasin, 2018). LRU's primary strategy is to start by deleting the web items that have been used the least from the cache. LRU policy must therefore track these items by creating a recency index for each item in order to accomplish that. This index shows how long it has been since the item was last used. The new item will take the place of the least used one; however, cache pollution negatively affects the efficacy of LRU web caching policy. In this study, a web caching policy called Collaborative-Naïve Bayes-Least Recently Used (C-NB-LRU) is proposed, which combines the Collaborative-LRU with the NB classifier.

C-NB-LRU operates as follows: Let's say we have two media objects, g and k, and we have m peers. Let x_i be object i's recency index, where N is the number of media objects in the cache and $i \in [1,N]$. Additionally, suppose that g is a media object that a user has requested. The recency index x_g is applied to the Timestamp in the IRCache dataset if g is in the local cache, whereas it is assigned to the Time in the YemenNet dataset, and the NB classifier assigns the class of g. Furthermore, Local Cache Hit (LCH) also happens. On the other hand, Local Cache Miss (LCM) happens if g is not in the local cache, and the cache manager verifies that it is present in the caches of its peers. The Cache Manager retrieves g from the peer's cache if it is available, assigns the class of g to the NB classifier, and assigns the x_g to the Time in YemenNet dataset and the Timestamp in IRCache dataset; while, Peer Cache Hit (PCH) happens in the event that it is unavailable; where, the Cache Manager redirects the request to the peer that serves as a proxy in order to retrieve g from the origin server. This is known as Peer Cache Miss (PCM). In the IRCache dataset, the recency index x_g is assigned to the Timestamp if there is sufficient space for it; however, in YemenNet, it is assigned to the Time dataset, and the NB classifier assigns the class of g.

In the case, if there is no enough space, let n is considered as the least recently used object with x_n recency index, and the class of object n is 1 according to Equation (10). It implies that the object n will be revisited in the future. Conversely, let's say that object k has the same value as n's recency index x_n, but its class is 0, meaning that it won't be revisited. Thus, k will be selected by the Cache Manager as a victim to be replaced. Consequently, by minimizing cache pollution and making efficient use of the available cache space, the proposed C-NB-LRU approach can successfully eliminate undesirable objects to make space for new media objects.

The proposed C-NB-LRU approach's general flow chart is shown in Figure 1, and its pseudo code is shown in Figure 2.

4. Experiment

In order to evaluate the performance of the proposed approach, an experiment is run which includes many stages that are discussed below.

Two datasets are gathered in the first stage from YemenNet, the Internet Service Provider (ISP) of Yemen that is owned by Yemen Telecom (Yasin et al., 2023), and IRCache network. In IRCache dataset, when a user sends a request to a web server, this event is recorded into a log file. The logs are collected for users from many cache servers that are: BO2, NY, SD, SV, and UC as shown in Table 2.

In YemenNet dataset, a user sends a request to ISP servers that record his request. The dataset is collected on 19th and 20th of December 2016 for normal Internet users where the number of records is 309,684A network tool that is called Wireshark version 2.2.6 is used to analyze the data, where the total number of bytes is 228.51294 GB. The raw data format is not appropriate for direct simulation use. Because they have no bearing on the caching policy, unnecessary fields are eliminated. The second stage, data pre-processing, involves organizing the data in an appropriate way.

Table 2. Explanation of the IRCache dataset

Cache server	**Cache location**	**Duration of collection**	**Total No. of records**	**Total No. of Bytes (GB)**
BO2	Boulder, Colorado	8th to 14th of March 2013	571,919	20.60076
NY	New York	8th to 14th of March 2013	1,209,561	27.44352
SD	San Diego, California	8th to 14th of March 2013	20,514,737	446.41998
SV	Silicon Valley, California	22nd to 28th of August 2012	304,007	14.17165
UC	Urbana-Champaign, Illinois	21st to 27th of November 2010	7,874,706	663.97497

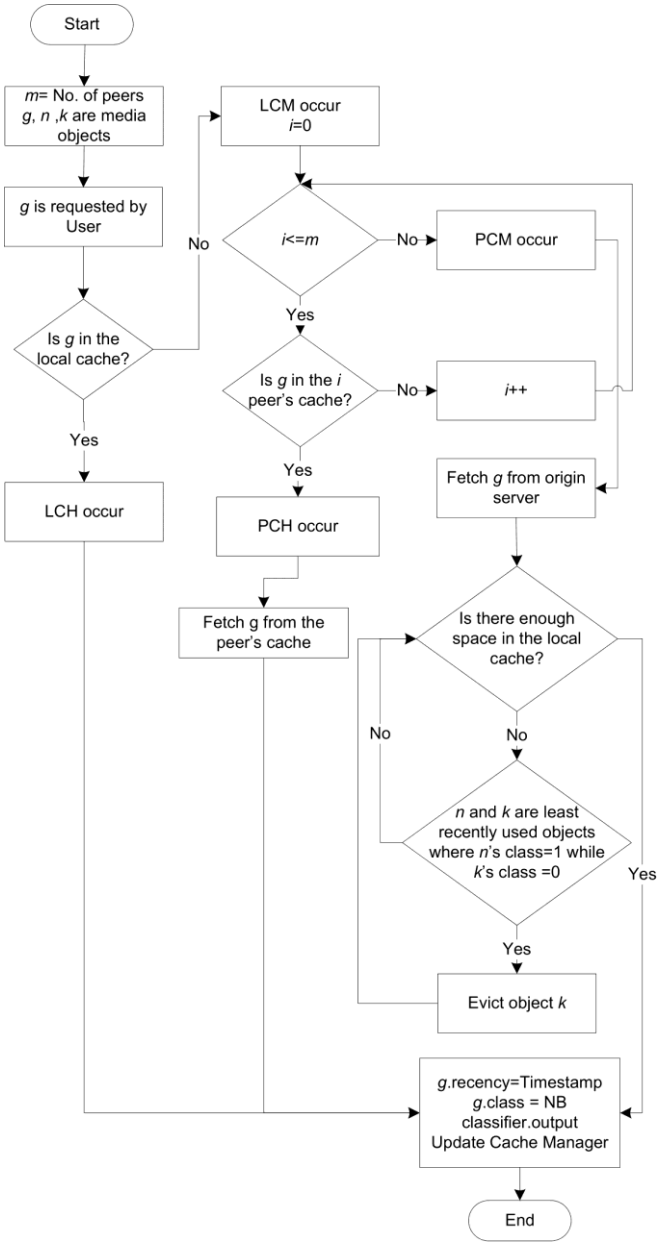

Figure 1. A general flow chart of C-NB-LRU approach.

In the third stage, a field which is called class is added to 70% of both datasets after the data pre-processing stage for training purpose to represent whether the media object will be revisited in the future or not. Once datasets are prepared, NB classifier is trained using Version 3.6.9 of Waikato Environment for Knowledge Analysis (WEKA) The aim of training is to train NB classifier in order to get the classification model which is tested using 30% of both datasets. NB classifier uses the attributes of media objects that are extracted from datasets as inputs while *y* is the output of the classifier as shown in Table 3.

4.1 Performance Evaluation

While *LBHR* is the total number of bytes found in the caches divided by the total number of bytes requested by the user during an observation time, *LHR* is calculated as a percentage of the total number of LCH divided by the total number of requests. Let *N* be the total number of media object requests. Equation (11), which defines *LHR*, and Equation (12), which defines

LBHR, where S_i is the size of the ith request, and $x_i = 1$ in the case of LCH occurrence and 0 otherwise.

```
1: Begin
2:   m= No. of Peers; g, n, k are media objects
3:  For each media object g requested by user
4:    Begin
5:     If g in cache
6:      Begin
7:        LCH occurs
8:        g.recency = Timestamp for IRCache | Time for YemenNet
9:        g.class = NB classifier.output
10:       Update Cache Manager
11:     End
12:   Else If g is not in the local cache
13:     Begin
14:       LCM occurs
15:       for (i = 1; i <= m; i++)
16:         Begin
17:           Check peers' caches //using Chord
18:           If g in a peer's cache
19:             Begin
20:               PCH occurs
21:               Fetch g from the peer's cache
22:               g.recency=Timestamp for IRCache | Time for YemenNet
23:               g.class = NB classifier.output
24:               Update Cache Manager
25:               Break
26:             End
27:           Else
28:             Begin
29:               PCM occurs
30:               Fetch g from origin server
31:               While no enough space in the local cache for g
32:                   if n and k are least recently used objects where n's class=1 while k's class =0
33:                       Evict object k
34:               g.recency=Timestamp for IRCache | Time for YemenNet
35:               g.class = NB classifier.output
36:               Update Cache Manager
37:           End
38:       End
39:     End
40:   End
41: End
```

Figure 2. The pseudocode of C-NB-LRU approach

Table 3. NB classifier's inputs and output and their description

Input	Description	IRCache	YemenNet
x_1	The recency of the media object	Timestamp	Time
x_2	The Client ID	Client_ID	Client_ID
x_3	The URL ID of the media object	URL_ID	URL_ID
x_4	The size of the media object in bytes	Size	Length
x_5	The access latency or retrieval time of the media object in milliseconds, which means the cost	Elapsed_Time	Delta_Time
x_6	Media object frequency	Freq	Freq
Y	Classifier output	Class	Class

(11)

$$LHR = \frac{Total\ No.of\ LCH}{N}$$

$$LBHR = \frac{\sum_{i=1}^{N} S_i x_i}{\sum_{i=1}^{N} S_i} \tag{12}$$

The Global Byte Hit Ratio (*GBHR*) is calculated by dividing the total number of bytes found in the caches by the total number of bytes requested by the user during an observation period, while *GHR* is defined as a percentage of the total number of PCH divided by the total number of requests. Whereas *GHR* is defined by Equation (13), *GBHR* is defined by Equation (14), where $y_i = 1$ if the request i is in the peer's cache; while $y_i = 0$ otherwise.

$$GHR = \frac{Total\ No.of\ PCH}{N} \tag{13}$$

$$GBHR = \frac{\sum_{i=1}^{N} S_i y_i}{\sum_{i=1}^{N} S_i} \tag{14}$$

SavedCost is the saved cost by retrieving the media object from local or peers' cache. In order to calculate *SavedCost*, two costs are defined as: (i) Out-Cost: the cost to retrieve the media object from the original server in milliseconds. The Out-Cost for IRCache dataset is gathered from Elapsed Time field, while it is gathered from Delta Time field for YemenNet dataset. (ii) In-Cost: the cost to retrieve the media object from the peers' cache in milliseconds. Let *OC* be the Out-Cost and *IC* be the In-Cost of a media object x_i then the *SavedCost* is defined as Equation (15), and the *CT* is defined as Equation (16); where $i, j \in [1,N]$ and x_i is retrieved from the local cache; while, x_j is retrieved from the peers' cache. If x_i is retrieved from the cache either the local cache or peers' cache, $y_i = 0$; while $y_i = 1$ otherwise.

$$SavedCost = \sum_{i=1}^{N} OC_{x_i} + \sum_{j=1}^{N} (OC_{x_j} - IC_{x_j}) \tag{15}$$

$$CT = \frac{SavedCost}{\sum_{j=1}^{N} (OC_{x_i} * y_i)} \tag{16}$$

The *IC* is generated randomly for each record, where the range is between [180,285] ms (Wan et al., 2010). The uniform discrete distribution, which is characterized by the probability formula shown in Equation (17), that is used by the random function to generate integer numbers.

$$P(i|a,b) = \frac{1}{(b-a+1)}\ , \qquad a \le i \le b \tag{17}$$

5. Results and Discussion

The C-NB-LRU approach is compared to LRU policy and Collaborative-LRU (C-LRU). The HR, BHR, and *CT* performance of LRU, C-LRU, and C-NB-LRU on the IRCache dataset are displayed in Table 4, Table 5, and Table 6, respectively.

Table 4. A comparative analysis of the HR performance among LRU, C-LRU, and C-NB-LRU using IRCache dataset.

Cache size (No. of items)	**LRU**	**C-LRU**			**C-NB-LRU**		
	LHR (%)	***LHR (%)***	***GHR (%)***	**Total (%)**	***LHR (%)***	***GHR (%)***	**Total (%)**

1	0.05727	0.03818	52.32913	52.36732	0.09546	52.72054	52.81596
10	0.05727	0.03818	52.33391	52.37209	0.12887	52.84464	52.97346
100	0.09069	0.06205	52.36254	52.42459	0.20046	52.88278	53.08324
1000	0.53933	0.50115	52.77778	53.27892	0.56319	53.16915	53.73234
10000	1.05001	0.51546	52.80164	53.31711	0.56797	53.16915	53.73711
100000	1.05001	0.51546	52.80164	53.31711	0.56797	53.16915	53.73711

Table 5. A comparative analysis of the BHR performance among LRU, C-LRU, and C-NB-LRU using IRCache dataset.

Cache size (No. of items)	**LRU**	**C-LRU**			**C-NB-LRU**		
	LBHR (%)	***LBHR (%)***	***GHR (%)***	**Total (%)**	***LBHR (%)***	***GBHR (%)***	**Total (%)**
1	0.00289	0.00089	55.55016	55.55105	0.00143	55.56851	55.56993
10	0.00289	0.00089	55.55056	55.55146	0.00188	55.57462	55.57650
100	0.00499	0.00167	55.55297	55.55463	0.00278	55.57474	55.57753
1000	0.00822	0.00312	55.55918	55.56230	0.00391	55.57802	55.58193
10000	0.00949	0.00316	55.56220	55.56536	0.00392	55.57802	55.58194
100000	0.00949	0.00316	55.56220	55.56536	0.00392	55.57802	55.58194

Table 6. A comparative analysis of the *CT* performance among C-LRU and C-NB-LRU using IRCache dataset.

Cache size (No. of items)	***CT* of C-LRU (%)**	***CT* of C-NB-LRU (%)**
1	48.12429	48.11814
10	48.12493	48.11710
100	48.13022	48.11543
1000	48.09803	48.09110
10000	48.09530	48.09116
100000	48.09530	48.09116

Referring to Table 4, Table 5, and Table 6, the following is observed: (i) C-NB-LRU outperforms LRU regarding HR and BHR. This is because C-NBLRU considers the contents of peers when it responds to a user's request based on intelligent collaborative caching approach using NB classifier. For example, the average *LHR* of C-NB-LRU is 0.354%; while the average *LHR* of LRU is 0.474% and the average *LHR* of C-LRU is 0.278%. On the other hand, the *GHR* of C-NB-LRU is 52.992%; while the *GHR* of C-LRU is 52.568%. That means C-NB-LRU increases the performance of C-LRU local cache 27.34%; while it increases the performance of C-LRU peers' caches 0.81%. Furthermore, the average *LHR* of C-NBLRU is 0.00297%; while the average *LBHR* of LRU is 0.00633% and the average *LBHR* of C-LRU 0.00215%. That means C-NB-LRU increases the performance of C-LRU local cache 38.14%. The average *GBHR* of C-NB-LRU is 55.57532%; while the average *GBHR* of C-LRU is 55.55621%. Thus, C-NB-LRU increases the performance of C-LRU peers' caches 0.034%. In contrast, the average *CT* of C-NB-LRU is 48.10401%; while the average *CT* of C-LRU is 48.11134%. That means C-LRU is better than C-NB-LRU 0.015% in terms of *CT* because of using additional process which is classification using NB ML. (ii) The primary factor positively influencing C-NB-LRU's HR and BHR performance is the caches of its peers. For example, peers' caches contribute with 99.34% in terms of HR, while local cache only contributed 0.66%. Also, the contribution of peers' caches is 99.99% in terms of BHR; however local caches contribute with 0.01%. (iii) Small cache sizes such as 1item and 10 items have the same HR and BHR performance of local cache of LRU and C-LRU, while C-NB LRU is affected by small cache sizes. (iv) The *LHR* of LRU, *LHR*, *LBHR*, *GHR*, *GBHR*, *CT* performances of C-LRU and C-NB-LRU reach a steady state at large caches' sizes. This is because the cache size is enough for caching the media objects. (v) The *CT* performance of LRU and C-NB-LRU decreases when the cache size increases.

On the other hand, the HR, BHR, and *CT* performance of LRU, C-LRU, and C-NB-LRU for YemenNet dataset are shown in Table 7, Table 8, and Table 9 respectively.

Table 7. A comparative analysis of the HR performance among LRU, C-LRU, and C-NB-LRU using YemenNet dataset

Cache size (No. of items)	**LRU**	**C-LRU**			**C-NB-LRU**		
	LHR (%)	***LHR (%)***	***GHR (%)***	**Total (%)**	***LHR (%)***	***GHR (%)***	**Total (%)**
1	0.34451	0.21819	63.16031	63.37854	0.28709	63.29812	63.58521
10	0.44786	0.27561	63.16031	63.43592	0.36747	63.34405	63.71153
100	0.83830	0.52825	63.34405	63.87234	0.72347	63.56224	64.28571
1000	2.23932	1.02205	63.53927	64.56132	1.05654	63.67708	64.73358
10000	2.23932	1.02205	63.53927	64.56132	1.05654	63.67708	64.73358
100000	2.23932	1.02205	63.53927	64.56132	1.05654	63.67708	64.73358

Table 8. A comparative analysis of the BHR performance among LRU, C-LRU, and C-NB-LRU using YemenNet dataset

Cache size (No. of items)	**LRU**	**C-LRU**			**C-NB-LRU**		
	LBHR (%)	**LBHR (%)**	**GHR (%)**	**Total (%)**	**LBHR (%)**	**GBHR (%)**	**Total (%)**
1	0.35899	0.21429	63.35358	63.56787	0.38334	63.49594	63.87928
10	0.47133	0.27334	63.35358	63.62693	0.49875	63.54339	64.04214
100	0.81217	0.54257	63.55102	64.09358	0.92756	63.79205	64.71960
1000	2.54900	1.17570	63.73664	64.91234	1.31494	63.87944	65.19393
10000	2.54900	1.17570	63.73664	64.91234	1.31494	63.87944	65.19393
100000	2.54900	1.17104	63.78409	64.95514	1.31494	63.87944	65.19393

Table 9. A comparative analysis of the *CT* performance among C-LRU and C-NB-LRU using YemenNet dataset

Cache size (No. of items)	**CT of C-LRU (%)**	**CT of C-NB-LRU (%)**
1	61.80390	62.13419
10	62.01561	62.34590
100	62.06522	62.72689
1000	62.82312	63.03268
10000	62.82312	63.03268
100000	62.82312	63.03268

Referring to Table 7, Table 8, and Table 9, the following is observed: (i) C-NB-LRU outperforms LRU regarding HR, BHR, and *CT*. On average, C-NB-LRU increases the LHR performance of C-LRU local cache 11.24%; while it increases the *GHR* performance of LRU peers' caches 0.25%. Furthermore, CNB-LRU increases the *LBHR* performance of C-LRU 26.40%. Also, C-NB-LRU increases the GBHR performance of C-LRU peers' caches 0.25%. Moreover, the performance of C-NB-LRU is better than the performance of C-LRU 0.52% in terms of *CT*. (ii) On average, peers' caches contribute with 98.82% in terms of HR, while local cache only contributes with 1.18%. Also, the contribution of peers' caches is 98.52% in terms of BHR, while local cache only contributes with 98.52%. (iii) Small cache sizes such as 1 item and 10 items have the same HR and BHR performances of local cache of LRU, while the C-NB-LRU is affected by small cache sizes. (iv) The HR, BHR, and *CT* performances of LRU and C-NB-LRU reach a steady state at large caches' sizes. This is because the cache size is enough for caching media objects. (v) The *CT* performance of C-LRU and C-NB-LRU decreases when the cache size increases.

In this study, the performance of C-NB-LRU is compared to C4.5 ML based approach because of its accuracy compared to other ML algorithms, such as kNN and Logistic Regression (LR) as shown in Figure 3 (Negara et al., 2024).

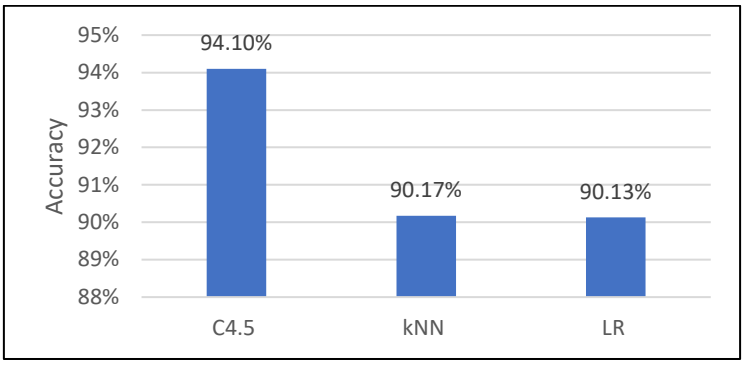

Figure 3. Accuracy results of different ML algorithms (Negara et al., 2024).

Table 10. C-C4.5-LRU and C-NB-LRU comparison for the IRCache dataset

Cache size (No. of items)	HR of C-C4.5-LRU (%)	HR of C-NB-LRU (%)	BHR of C-C4.5-LRU (%)	BHR of C-NB-LRU (%)	CT of C-C4.5-LRU (%)	CT of C-NB-LRU (%)
1	52.8159	52.8165	55.5699	55.5699	48.1181	48.1181
10	52.9734	52.9735	55.5764	55.5764	48.1171	48.1171
100	53.0832	53.0832	55.5775	55.5775	48.1154	48.1154
1000	53.7323	53.7323	55.5819	55.5819	48.0911	48.0911
10000	53.7371	53.7371	55.5819	55.5819	48.0911	48.0911
100000	53.7371	53.7371	55.5819	55.5819	48.0911	48.0911

Table 11. C-C4.5-LRU and C-NB-LRU comparison for the YemenNet dataset

Cache size (No. of items)	HR of C-C4.5-LRU (%)	HR of C-NB-LRU (%)	BHR of C-C4.5-LRU (%)	BHR of C-NB-LRU (%)	CT of C-C4.5-LRU (%)	CT of C-NB-LRU (%)
1	63.5852	63.5852	63.8792	63.8792	62.1341	62.1341
10	63.7115	63.7115	64.0421	64.0421	62.3459	62.3459
100	64.2857	64.2857	64.7196	64.7196	62.7268	62.7268
1000	64.7335	64.7335	65.1939	65.1939	63.0326	63.0326
10000	64.7335	64.7335	65.1939	65.1939	63.0326	63.0326
100000	64.7335	64.7335	65.1939	65.1939	63.0326	63.0326

Figure 3 illustrates that the accuracy of C4.5 ML is approximately 94.10%; while it is 90.17% for kNN. On other hand, Table 10 shows a comparison between C-C4.5-LRU and C-NB-LRU for IRCache dataset; where C-C4.5-LRU is collaborative least recently used caching policy based on C4.5 ML algorithm, while Table 11 shows the same comparison for YemenNet dataset.

Table 10 illustrates that HR performance of C-NB-LRU is 52.8165% when the cache size is 1 item; while for the same cache size the HR of C-C4.5-LRU is 52.8159%. It means that C-NB-LRU is 0.001136% better than C-C4.5-LRU. On the other hand, Table 11 illustrates that HR performance of C-NB-LRU and C-C4.5-LRU is the same.

Table 12 shows the HR Improvement Ratio (*IR*) of C-NB-LRU compared to C-LRU in IRCache dataset; while Table 13 shows the BHR IR of C-NB-LRU compared to C-LRU.

With reference to Table 12, it is evident that C-NBLRU outperforms C-LRU. For example, when the cache size is 1 item, the *IR* of *LHR* is more the 150%; while the *IR* of *LHR* is 10.187017% for 10,000 and 100,000 items of cache sizes, which means C-NBLRU reaches the steady state of *LHR IR.* Moreover, the *GHR IR* of C-NB-LRU compared to C-LRU is 0.74797% when the cache size is 1 item, while the *IR* of *GHR* is 0.69602% for 10,000 and 100,000 items of cache sizes.

Table 12. The HR *IR* of C-NB-LRU in IRCache dataset

Cache size (No. of items)	**The *LHR IR* of C-NB-LRU compared to C-LRU (%)**	**The *GHR IR* of C-NB-LRU compared to C-LRU (%)**	**The *IR* of C-NB-LRU compared to C-LRU (%)**
1	150.02619	0.74797	0.85671
10	63.7115	0.97590	1.14826
100	223.06204	0.99353	1.25637
1000	12.379527	0.74154	0.85103
10000	10.187017	0.69602	0.78773
100000	10.187017	0.69602	0.78773

Table 13. The BHR IR of C-NB-LRU in IRCache dataset

Cache size (No. of items)	**The LBHR IR of C-NB-LRU compared to C-LRU (%)**	**The GBHR IR of C-NB-LRU compared to C-LRU (%)**	**The IR of C-NB-LRU compared to C-LRU (%)**
1	60.67415	0.03303	0.03398
10	111.23595	0.04331	0.04507
100	66.46706	0.03918	0.04122
1000	25.32051	0.03390	0.03532
10000	24.05063	0.02847	0.02983
100000	24.05063	0.02847	0.02983

Table 14. The HR IR of C-NB-LRU in YemenNet dataset

Cache size (No. of items)	**The *LHR IR* of C-NB-LRU compared to C-LRU (%)**	**The *GHR IR* of C-NB-LRU compared to C-LRU (%)**	**The *IR* of C-NB-LRU compared to C-LRU (%)**
1	31.57798	0.21819	0.326088
10	33.32970	0.29091	0.434469
100	36.95598	0.34445	0.647181
1000	3.37459	0.216889	0.266816
10000	3.37459	0.216889	0.266816
100000	3.37459	0.216889	0.266816

As can be seen from Table 13, C-NBLRU outperforms C-LRU in terms of *LBHR*. For example, when the cache size is 1 item, the *IR* of *LBHR* is more than 60%; while the *IR* of *LBHR* is more than 24% for 10,000 and 100,000 items of cache sizes, which means C-NB-LRU reaches the steady state of *LBHR IR*. Also, The *GBHR IR* of C-NB-LRU compared to C-LRU is 0.03303%. while the *IR* of *GBHR* is 0.02847% for 10,000 and 100,000 items of cache sizes.

Table 14 shows the HR *IR* of C-NB-LRU compared to C-LRU in YemenNet dataset; while Table 15 shows the *BHR IR* of C-NB-LRU compared to C-LRU.

As can be seen from Table 14, it can be observed that C-NBLRU performs better than C-LRU. For example, when the cache size is 1 item, the *LHR IR* is more than 31%; while the *IR* of *LHR* is 3.37459% for 10,000 and 100,000 items of cache sizes, which means C-NB-LRU reaches the steady state of *LHR IR*. Also, The *GHR IR* of C-NB-LRU compared to C-LRU is

0.326088% when the cache size is 1 item; while the *IR* of *GHR* is 0.266816% for 10,000 and 100,000 items of cache sizes.

Table 15. The IR of C-NB-LRU of BHR in YemenNet dataset

Cache size (No. of items)	The *LBHR IR* of C-NB-LRU compared to C-LRU (%)	The *GBHR IR* of C-NB-LRU compared to C-LRU (%)	The *IR* of C-NB-LRU compared to C-LRU (%)
1	78.88842	0.22470	0.48988
10	82.46506	0.29960	0.65256
100	70.95674	0.37927	0.97672
1000	11.84315	0.22404	0.43380
10000	11.84315	0.22404	0.43380
100000	12.28822	0.149488	0.36762

Referring to Table 15, it can be observed that C-NBLRU performs better than C-LRU in terms of *LBHR*. For example, when the cache size is 1 item, the *IR* of *LBHR* is more than 78%; while the *IR* of *LBHR* is more than 12% for 100,000 cache size, which means C-NB-LRU outperforms C-LRU. Also, the *GBHR IR* of C-NB-LRU compared C-LRU is more than 0.22% when the cache size is 1 item; while the *IR* of *GBHR* is more than 0.14% when the cache size is 100,000 items.

6. Computation Performance Study

Computation performance is defined as the duration of time required to perform a computation task. The computation study is carried out in this section to investigate the performance of the proposed approach for both datasets.

6.1 Computation Performance Study for IRCache Dataset

Figure 4 illustrates the computational performance of LRU, NB-LRU, and C-NB-LRU. Referring to Figure 4, the following is observed: on average, the computational performance of C-NB-LRU is 7216.41 ms; while it is 6750.71 ms for NB-LRU and 7061.25 ms for LRU. Thus, C-NB-LRU consumes time 6.45% more than NBLRU and 2.15% more than LRU. This is due to the consideration of peers' caches by C-NB-LRU when there is a need for a cache replacement. Moreover, they consume time for checking media object's class.

A comparison of the computational performance between C-NB-LRU and C-C4.5-LRU using the IRCache dataset is shown in Figure 5. Referring to Figure 5, the following is observed: C-NB-LRU consumes more time than C-C4.5-LRU. For example, when the cache size is 1000 items C-NB-LRU consumes 7325.5 ms; while C-C4.5-LRU consumes 6873.5 ms for the same cache size. On average, C-NB-LRU consumes 3% more than C-C4.5-LRU in terms of time.

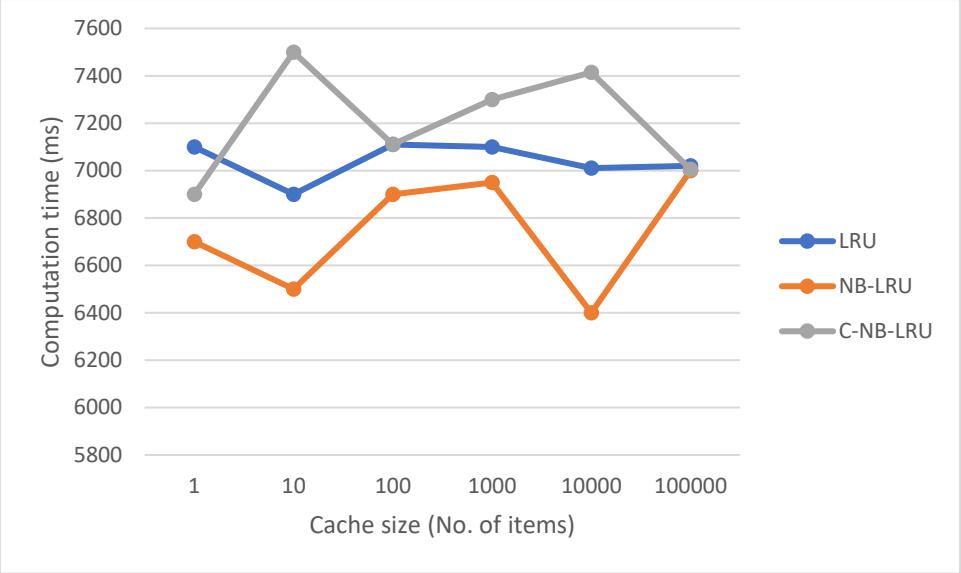

Figure 4. The computation performance of C-NB-LRU, NB-LRU, and LRU for IRCache dataset.

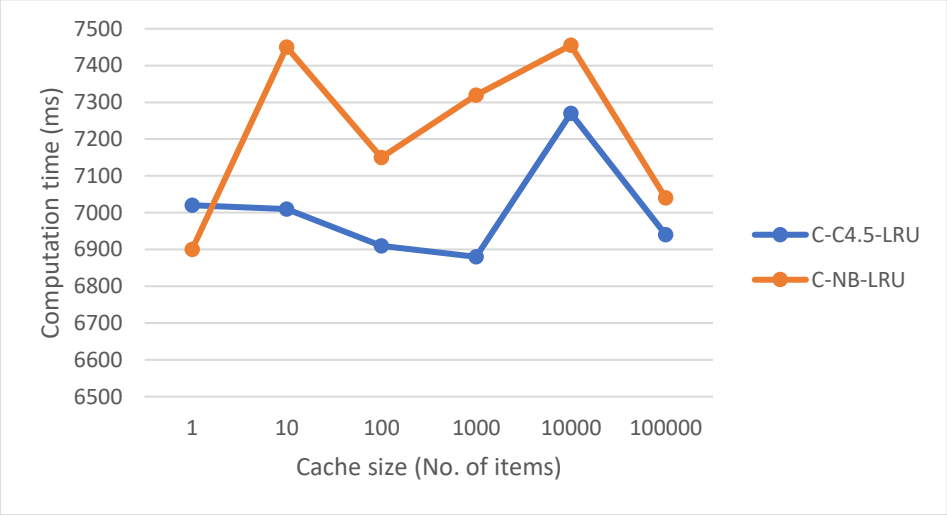

Figure 5. A comparison of the computational performance between C-C4.5-LRU and C-NB-LRU using the IRCache dataset.

6.2 Computation Performance Study for YemenNet dataset

The computational performance of LRU, NB-LRU, and C-NB-LRU is shown in Figure 6. Referring to Figure 6, on average, the computational performance of C-NB-LRU is 2311.13 ms; while it is 2341 ms for NBLRU and 2564.29 ms for LRU. Thus, C-NB-LRU consumes time 1.28% less than NB-LRU and 9.87% less than LRU. This is because C-NB-LRU uses recency index.

A comparison of the computational performance between C-NB-LRU and C-C4.5-LRU using YemenNet dataset is shown in Figure 7. Referring to Figure 7, the following is observed: C-C4.5-LRU consumes time more than C-NB-LRU. For example, C-C4.5-LRU consumes 3031.5 ms when the cache size is 100 items; while CNB-LRU consumes 2230.5 ms for the same cache size.On the average, C-C4.5-LRU consumes 16% more than C-NB-LRU in terms of time.

7. Conclusion and Future Works

In this study, collaborative media object caching approach based on NB supervised ML algorithm has been presented, which is called C-NB-LRU. The proposed approach has considered the benefits of sharing the contents of peers' caches in structured peer-to-peer systems to outperforms LRU web caching policy.

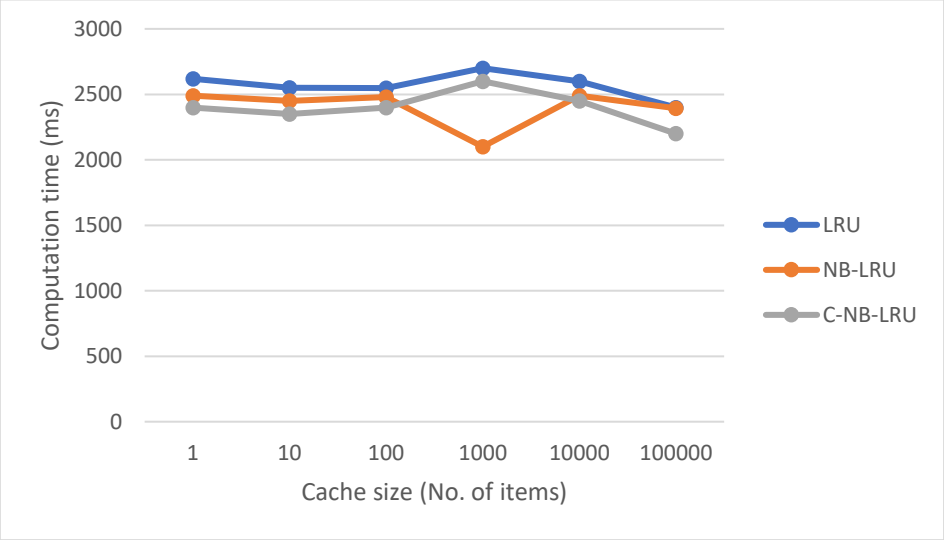

Figure 6. The computation performance of C-NB-LRU, NB-LRU, and LRU for YemenNet dataset.

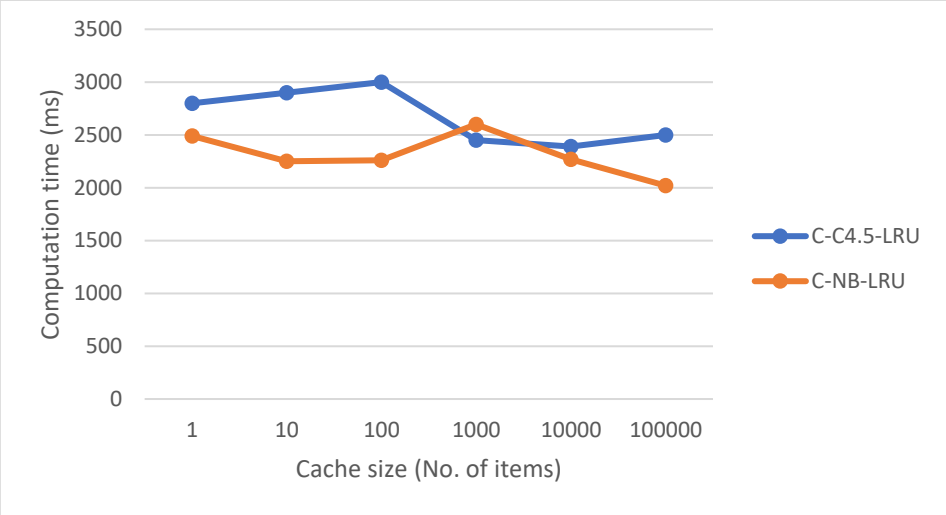

Figure 7. A comparison of the computational performance between C-NB-LRU and C-C4.5-LRU using YemenNet dataset.

The efficacy of C-NB-LRU has been evaluated using simulations conducted on two datasets obtained from the YemenNet ISP network and the IRCache network. The findings indicate that C-NB-LRU enhanced the efficacy of the LRU web caching policy regarding HR, BHR, and *CT*. Additionally, the *IR* metric has been analyzed for the proposed approach in comparison to LRU, C-LRU, and C-C4.5-LRU.

The proposed approach found that network bandwidth has been utilized based on collaborative web caching using NB ML algorithm for media web objects based on HR, BHR, and *CT* performance metrics. A web page may encompass, among other elements: data, text,

photos, e-services, audio, video, animations, and applications. Web pages are often updated, leading to data consistency concerns; nevertheless, media objects adhere to the Write Once Read Many (WORM) principles, rendering cache consistency challenges more robust than those associated with caching other web objects. This study focusses exclusively on caching media objects within peer-to-peer systems.

Some future work for further improvements are as follows: (i) Web objects that are normally updated are considered as a future work for further improvement with taking into account the issues of consistency. (ii) This research has proposed collaborative web caching approach based on NB supervised ML algorithm. Thus, adopting other ML algorithms is also considered as a future work. (iii) The proposed approach leverages the benefits of structured peer-to-peer systems by sharing the cached objects of peers to improve the efficacy of web caching policies. Thus, considering cloud computing in creating peer-to-peer system on cloud should be considered in future work.

On the other hand, it is noted that Internet traffic has changed in the last ten years. Today, the social networks, streaming services, e-commerce, among others, are the main traffic. Although, the approach is independent from datasets, it is encouraged to validate the proposed approach with recent datasets.

Acknowledgement

The IRCache dataset is provided by the National Science Foundation (grants NCR 9616602 and NCR-9521745), and the National Laboratory for Applied Network Research (NLANR); while the YemenNet dataset used is provided by YemenNet ISP. On the other hand, the author gratefully acknowledges publication support provided by Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Shah Alam, Selangor D. E., Malaysia.

Funding

This work is partly sponsored by Yemen Telecom, Yemen Mobile, and Yemen Net, Yemeni Ministry of Higher Education, and the Malaysian Ministry of Higher Education.

Conflict of Interest

The author declares that he has no conflict of interest.

References

Ahmad, A., Ahmad, F., Atif, S., & Aldalbahi, A. (2025). Extreme learning machine-driven joint user mobility and content popularity-based proactive caching in multi-tier wireless networks. *Ad Hoc Networks, 178*(1), 103920. https://doi.org/10.1016/j.adhoc.2025.103920

Alan, B. (1979). *Complex analysis: The argument principle in analysis and topology*. Wiley.

Ali, W., Shamsuddin, S. M., & Ismail, A. S. (2012a). Intelligent naïve Bayes-based approaches for web proxy caching. *Knowledge-Based Systems, 31*(1), 162–175. https://doi.org/10.1016/j.knosys.2012.02.015

Ali, W., Shamsuddin, S. M., & Ismail, A. S. (2012b). Intelligent web proxy caching approaches based on machine learning techniques. *Decision Support Systems, 53*(3), 565–579. https://doi.org/10.1016/j.dss.2012.04.011

BenMimoune, A. (2023). Machine learning-based edge caching for video streaming in 5G networks. In *the IEEE 8th International Conference on Recent Advances and Innovations in Engineering (ICRAIE)* (pp. 1–5). IEEE. https://doi.org/10.1109/ICRAIE59459.2023.10468338

Deepakraj, J., Thangavel, P., & Deepa, D. (2025). AI-driven smart caching for optimized web performance and reduced latency. In *the 4th International Conference on Smart Technologies, Communication and Robotics (STCR)* (pp. 1–5). IEEE. https://doi.org/10.1109/STCR62650.2025.11018927

Esakki, G., Panayides, A. S., Jalta, V., & Pattichis, M. S. (2021). Adaptive Video Encoding for Different Video Codecs. *IEEE Access, 9*(1), 68720-68736. https://doi.org/10.1109/ACCESS.2021.3077313

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning, 29*(2–3), 131–163. https://doi.org/10.1023/A:1007465528199

Ibrahim, H., Yasin, W., Udzir, N. I., & Hamid, N. A. W. A. (2016). Intelligent cooperative web caching policies for media objects based on J48 decision tree and naïve Bayes supervised machine learning algorithms in structured peer-to-peer systems. *Journal of Information and Communication Technology, 15*(2), 85–116.

Ibrahim, N., Ishak, U. M., Ali, N. N. A., & Shaadan, N. (2024). Machine learning-based approaches for credit card debt prediction. *Malaysian Journal of Computing*, *9* (1), 1722-1751. https://doi.org/10.24191/mjoc.v9i1.25656

Jialu, N., Quan, Z., Feng, Y., Zhenghuan, X., & Qianbao, S. (2022). Cache pricing mechanism for ICN on ISP peering. In *the 5th International Conference on Hot Information-Centric Networking (HotICN)* (pp. 13–18). IEEE. https://doi.org/10.1109/HotICN57539.2022.10036171

Kushwah, J. S., Gupta, D., Shrivastava, A., Pramitha, P. A., Abraham, J. T., & Lunagaria, M. (2022). Analysis and visualization of proxy caching using LRU, AVL tree and BST with supervised machine learning. *Materials Today: Proceedings, 51*(1), 750–755. https://doi.org/10.1016/j.matpr.2021.06.224

Lanying, S., Wensheng, Y., Kai, W., Huan, L., Yong, C., Hongming, Q., Jiangang, T., Man, L., Mengxia, C., Chengwei, Y., & Yiquan, J. (2022). Research on network proxy cache system for P2P flow. In *the 7th International Conference on Intelligent Computing and Signal Processing (ICSP)* (pp. 1522–1525). IEEE. https://doi.org/10.1109/ICSP54964.2022.9778533

Li, L., & Haichao, D. (2022). Research and application on distributed multi-level cache architecture. In *the 11th International Conference of Information and Communication Technology (ICTech)* (pp. 138–143). IEEE. https://doi.org/10.1109/HotICN57539.2022.10036171

Li, P., Guo, Y., & Gu, Y. (2022). Predicting Reuse Interval for Optimized Web Caching: An LSTM-Based Machine Learning Approach. In *SC22: International Conference for High*

Performance Computing, Networking, Storage and Analysis (pp. 1-15). IEEE. https://doi.org/10.1109/SC41404.2022.00091

Mohanty, S., Sahoo, M., & Acharya, A. A. (2022). Predicting phishing URLs using filter-based univariate feature selection technique. In *the Second International Conference on Computer Science, Engineering and Applications (ICCSEA)* (pp. 1–5). IEEE. https://doi.org/10.1109/ICCSEA54677.2022.9936298

Negara, R. M., Syambas, N. R., Mulyana, E., & Wasesa, N. P. (2024). Maximizing router efficiency in named data networking with machine learning-driven caching placement strategy. In *the 6th International Conference on Computer Communication and the Internet (ICCCI)* (pp. 118–123). IEEE. https://doi.org/10.1109/ICCCI62159.2024.10674657

Peng, X., Huang, H., & Luo, Z. (2022). When CCN meets MCGDM: Optimal cache replacement policy achieved by PRSRV with Pythagorean fuzzy set pair analysis. *Artificial Intelligence Review, 55*(7), 5621–5671. https://doi.org/10.1007/s10462-022-10139-y

Pernabas, J. B., Fidele, S. F., & Vaithinathan, K. K. (2019). Enhancing greedy web proxy caching using weighted random indexing-based data mining classifier. *Knowledge-Based Systems, 167*, 117–130. https://doi.org/10.1016/j.eij.2019.01.001

Pires, S., Ziviani, A., & Sampaio, N. L. (2021). Contextual dimensions for cache replacement schemes in information-centric networks: A systematic review. *PeerJ Computer Science, 7*, e418. https://doi.org/10.7717/peerj-cs.418

Ramli, A., Darus, M. Y., Yussoff, Y. M., Azni, B., & Xie, K. (2025). Integrated cybersecurity framework for enhanced threat detection and incident response in the digital era. *Malaysian Journal of Computing, 10* (1), 2099-2116. https://doi.org/10.24191/mjoc.v10i1.4520

Saxena, D., Singh, A. K., & Lindenstruth, V. (2024). A latency-aware and dynamic caching model for heterogeneous datalake environments. In *the IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)* (pp. 2302–2307). IEEE. https://doi.org/10.1109/COMPSAC61105.2024.00370

Saxena, D., Singh, A. K., & Lindenstruth, V. (2026). QuAd-caching management model for heterogeneous data lake environments. *Expert Systems with Applications*, *296*(1), 129133. https://doi.org/10.1016/j.eswa.2025.129133

Sharif, S., Moghaddam, Y. H., & Seno, S. A. H. (2022). Adaptive cache content placement for software-defined Internet of Things. *Future Generation Computer Systems, 136*, 34–48. https://doi.org/10.1016/j.future.2022.05.019

Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., & Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup service for Internet applications. *ACM SIGCOMM Computer Communication Review, 31*(4), 149–160. https://doi.org/10.1145/964723.383071

Wan, M., Liu, Y., Zhou, H., & Zhang, H. (2010). A Chord-based handoff authentication scheme under ID/locator separation architecture. In *the International Conference on Advanced Intelligence and Awareness Internet (AIAI 2010)* (pp. 309–314). https://doi.org/10.1049/cp.2010.0776

Xu, Z., Zhengnan, Q., Geyong, M., Wang, M., Qilin, F., & Zhan, M. (2022). Cooperative edge caching based on temporal convolutional networks. *IEEE Transactions on Parallel and Distributed Systems, 33*(9), 2093–2105. https://doi.org/10.1109/TPDS.2021.3135257

Yasin, W. (2018). *Network bandwidth utilization based on collaborative web caching using machine learning algorithms in peer-to-peer systems for media web objects* (Doctoral dissertation, Universiti Putra Malaysia).

Yasin, W., Algunied, A., Mustafa, N., Alhanshali, R., Omiran, S., & Al-Jahafi, M. (2023). A case study on optical fiber network current situation and future vision in Sana'a city. In *the 3rd International Conference on Emerging Smart Technologies and Applications (eSmarTA)* (pp. 1–8). IEEE. https://doi.org/10.1109/eSmarTA59349.2023.10293577

Zhou, Y., Wang, F., Shi, Z., & Feng, D. (2024). An efficient deep reinforcement learning-based automatic cache replacement policy in cloud block storage systems. *IEEE Transactions on Computers, 73*(1), 164–177. https://doi.org/10.1109/TC.2023.3325625