A Review on Adaptive Web Caching Technique

Pranay Nanda
Research Scholar
Department of Computer Science and Engineering,
Amity School of Engineering and Technology,
Amity University, Rajasthan
India

Shamsher Singh, PhD
Asst professor
Department of Computer Science,
AB CollegePathankot,India

G. L. Saini
Asst Professor
Department of Computer Science and Engineering,
Amity School of Engineering and Technology,
Amity University, Rajasthan
India

ABSTRACT
Traditional caches see a fixed-size page; a Web cache, on the other hand, sees complete objects (text files, images, or video clips), which vary considerably in size. In addition, a traditional cache deals with addresses, while a Web cache can potentially deduce more contextual information from its objects. Web objects are predominantly read-only, taking implementation of cache coherence easier. The response times of Web accesses are in the order of seconds (versus milliseconds for system access), which allows for more elaborate caching algorithms. Finally, a Web cache encounters more dimensions of dependence than are taken into account by traditional methods. Primary cache replacement algorithms consider arrival time as the only one factor as the basis of their functionality. They disregard parameters such as page size, fetching delay, reference rate, and invalidation cost and invalidation frequency of a web object. Considering these parameters produces better and more apt results with greater efficiency, thereby surpassing traditional and conventional algorithms such as LRU, LIFO and LFU in performance and accuracy. Most of them are favorable to objects with homogenous sizes. Also, many of these algorithms depend on manual interference to find quick cures for symptoms instead of understanding the core issues. Because the cache space is limited and no technology can be as suitable to cater to each user's request separately, we need caching algorithms that are intelligent and adapt to the available resources and utilize them optimally. Systems must evolve towards more scalable, adaptive, efficient and self-configuring web caching systems to effectively support the phenomenal growth in demand for web content on the internet. Adaptive caching views caching problems as a way of optimizing global data dissemination. Studies have shown that adaptive algorithms yield better results than conventional caching algorithms.

Keywords
Web Caching, Caching Cache Algorithm.

1. INTRODUCTION
Primary and significant cache alternative algorithms consider landing time period as the only one factor as the basis of their functionality. They disregard the parameters such as page size, fetching delay; reference rate, invalidation cost and invalidation rate of recurrence of a web object [2]. Thinking About these parameters generates much better and more apt results with greater efficiency, thereby exceeding conventional as well as conventional algorithms such as LRU, FIFO, LIFO and LFU in performance and accuracy [11]. Most of them are favourable to CPU caches instead of web caches [13]. Also, many of these algorithms are dependent on manual intervention to find rapid treats for symptoms instead of understanding the fundamental issues. Because the cache space is limited and no technologies can be as suitable to cater to each user's inquire individually, we need caching algorithms that are well-informed as well as adjust to the available resources and utilize those optimized [12]. Systems must progress towards more scalable, adaptive, efficient and self-configuring web caching systems to effectively support the phenomenal growth in demand for web content on the internet [1]. Adaptive caching views caching difficulties as a way of perfecting global data dissemination [14]. Studies have shown [9] [12] [6] [11] that adaptive algorithms yield much better results than traditional caching algorithms

2. FUZZY INFERENCE SYSTEM
As artificial intelligence is choosing their way in practically every single thing as well as really stands as foundation stone for many technologies, here a Sugeno type Fuzzy Inference System [2] is implemented. The mechanism utilizes three guidelines which are Frequency, Latency and Byte sent. Frequency implies the number of cache hits of the web objects in a time interval. Higher frequency specifically implies higher chances of the object being re-accessed. Therefore the object with greater rate of recurrence would be granted greater priority against those with a lesser frequency. Latency is the delay that was required to get the object from the web server. An object with greater Latency would imply prerequisite of greater priority. Setting Up the web object with higher latency close to the client would lead to performance improvements and lower latency. Byte sent means the size of an object or the bytes sent from the server to the proxy server for the requested for web objects. An object with large Byte sent size would require greater bandwidth and subsequently it could possibly, potentially reduce the performance of the web. Therefore a larger object would be expected to be placed close to the client to cut down on utilization of bandwidth.

The architecture proposes:

1. All objects are initially placed in the cache of the proxy server. The information regarding the latency and byte sent is recorded and saved at proxy server cache. The proxy server records the frequency of hits for each object. When the cache memory has no more space for new objects than the replacement algorithm comes into play.

2. The Sugeno type Fuzzy Inference System assigns ranks to objects based on the three parameters of Frequency, Latency and Byte sent. The object with lower rank is removed from the cache and the object with higher rank is kept in the cache. In case, all objects arbitrarily have

high ranks, then any web object can be removed for the new objects coming into cache

3. LOGISTIC REGRESSION OPTIMAL CACHE ALGORITHM

We embrace logistic simple regression model to anticipate our future accesses based on our access history [3]. By using this model, a web caching agent can acquire knowledge about the elements, it encounters and deduce to an excellent strategy for better web experience and diminished network congestion [10]. A logistic regression model is a simple and easily repeatable method of analysis. Our expectation is the fact that Web access patterns over the short term are not altogether random, nor chaotic (self-similar). Web pages deemed most likely to be re-accessed in the near future will be acknowledged and kept in cache. We pose this as a "learning from example" problem in machine learning theory: the predicate to be learned is Web object re-access; the examples are traces of past Web accesses; and a logistic regression model accomplishes the discovering.

Regression models have been widely used by bio-statisticians to model the risk (or probability) of disease development (the event) as a function of factors suspected to affect the disease. These factors are also referred to as predictors or covariates of the model. Our goal is to express the outcome of the dependent variable Y (of some value g), in terms of its predictors (1, X_1, …, X_k) and their respective coefficient (β_o, β_1,…… β_k).

$P(Y=gI1, X_1, …, X_k) = f(z)$……

Where $z = \beta_o + \beta_1 x_1 +$…… $+ \beta_k x_k$

Given a set of observed data, the coefficients can be estimated by a suitable method (learning phase). Once the coefficients are determined, we can use the function to predict the outcome of an event when the predictors are known (prediction phase)

The event of interest is whether an object is re-accessed at least once in the next WF accesses. Predictors include

- Size and type of object,
- Number of times it has been accessed
- Time since last access

We want to ascertain whether such factors are predictive of Web re-accesses.

Let (Y = 1IX) be the probability of the event Y = 1, given the predictor vector X. This is modelled as a function of linear predictor z, P (Y = 1IX) = f (z). Since P (Y = 1IX) is a probability, the function f (z) must map the real line between 0 and 1. Several so-called link functions are commonly used for this purpose, the most popular being the logistic function taking the form

P (Y=1IX) = $-\infty < z < +\infty$ ……..

The logistic function has several favourable properties:

- It maps probability to 0 and 1
- It lends itself to biologically meaningful interpretation
- Its "S" shape indicates that the effect of z is minimal for low 'zs' until some threshold is reached.

When enough predictors are present, the function rises rapidly and remains relatively constant once z gets large. Part of our investigation is to determine whether such a function can be applied to the Web. This specific model is referred to as the logistic regression model. A non-event is simply modelled as the complement of an event, or

P (Y=0IX) 1- P (Y=1IX)……..

Since Yi is the binary outcome (1 for an event, 0 for a non-event) observed on the access. Let Xi be the associated vector of predictors. Assuming Y_1, Y_2, ……. Y_m are independent, the joint probability of the sequence Y_1, Y_2, …… Y_m is proportional to the

Product Π P ($Y = Y_i$ I X_i)……..

This is the likelihood equation. The coefficient of a predictor variable indicates show it influences the probability of an event. If $\beta = 0$, the variable has no influence. If $\beta > 0$, then the value of variable increases the probability of the event and if $\beta < 0$, then the value of variable decreases the probability of the event. However, it must be noted that the value of β coefficients is obtained. The p-value is commonly used to measure the strength of evidence that the estimates of β are statistically significantly different from zero. For our purposes, we consider any predictors with p-value ≤ 0.1 to be significant.

Our caching algorithm can theoretically be implemented as two background process: continuous learning and predicting. . For simplicity, we performed only one pass of learning and prediction. Because we based our learning on the number of accesses, or events, instead of time, we were able to circumvent the possibility of down-time on the servers.

The ultimate goal of a cache system is to reduce the access times for the end user. The classical textbook formula of average access time of an object in system cache is as follows:

Average access time + t_{cache} + (1-4)*$t_{noncache}$……..

Cache = Time needed to transfer objects from cache

Tnoncache = Time needed to transfer objects from non-cache location,

h= Object hit ratio

This formula assumes that transfer times are independent of object size, network conditions (that is, the availability of bandwidth), and object location. These assumptions are valid for system caches since they deal with fixed-size pages; stable memory/disk connection; and objects residing at similar distances from point of processing.

We introduce an object replacement ratio. Reducing the replacement ratio reduces disk fragmentation and the need for constant cache maintenance. One-time referring behaviour can result in the clattering of one-third of a server cache with useless files. A low replacement ratio, therefore, also indicates the caching efficiency. Other estimates of cache performance use similar metrics since they can be easily computed with an off line simulator.

4. ACASH

ACASH [16] takes care of a cache scope by dividing it dependent on the heterogeneity of web objects. It adaptively contemplates modifications on the reference point characteristics of the object based the time elapsed. Large heterogeneity of the web object stimulates more frequent cache substitution and helps to create large variations in object size. ACASH divides the storage scope of web cache into two kinds of domain in accordance to object size.

- Some considerations recommended would be to store as many web objects as possible in web cache and not to save small sized web objects in the cache with this perspective.
- Also, the replacement of a large object eliminates many small sized objects at a time.

The absence of cache for a large object highly boosts network traffic and decreases the performance of the web system. With this belief, saving a large object in a cache is better to appreciate to network. Based on prior research, we can determine a large number of dissimilarities between your total transmission high quality and the number of rate of recurrence references based on a 10K size object. ACASH takes care of web objects by dividing them in accordance to their size. Objects are classified as LARGE (over 10K in size) and SMALL (below 10K in size) based on their size for easy administration.

ACASH gets flexibility with reference characteristics of web objects in accordance to time changes by controlling the division rate of each presented cache domain. A cache management checks for the existence of the object in the division's scope constituted by the object size when an object is requested by a client. In case of a cache hit, the cache management provides the service object for the client request and enhancements the time used record, which receives high priority in a LRUMIN substitution. In case of a cache miss, the cache manager receives the transmission of the object by requesting service from the URL internet server. Then the transmissible objects are divided to LARGE or SMALL basis of object size. The cache manager then checks the storage scope to save these objects in the cache domain in the proper object level. If there are spaces to save it, the objects are stored in cache scope and if there are no spaces to save it, the objects are allocated to free spaces by LRUMIN and are saved in the cache. The objects can be replaced between same scope level objects and high priority is allocated by saving the time record of the newly stored object. ACASH has a relatively small variation of web object size compared with LRU, LRUMIN and SIZE because it manages the storage scope of the cache by dividing the object's size into LARGE and SMALL

ACASH ADAPTOR is used to get the adaptability of division rate of the cache scope. Each domain is divided into lightly loaded, lightly overloaded and overloaded stated based on load state. The workflow is as follows:

- Initially and the ratio of load state is 5:5 which changes according to object reference on each division domain.
- Periodically, as the load state of a domain changes, if a load state of a domain reaches the overloaded state, the ACASH ADAPTOR checks the load state of the different domain.
- If the load state of other domain is lesser than the overloaded state, then the division rate of overloaded state is increased by 5% on the total cache division scale. Else, if the load state of other domain is slightly overloaded, the cache division operates without adjustment.
- The rate of a domain cannot be over 70% on a division rate adjustment.
- Periodically, as the load state of a domain changes, if a load state of a domain reaches the overloaded state, the ACASH ADAPTOR checks the load state of the different domain.
- If the load state of other domain is lesser than the overloaded state, then the division rate of overloaded state is increased by 5% on the total cache division scale. Else, if the load state of other domain is slightly overloaded, the cache division operates without adjustment.

The rate of a domain cannot be over 70% on a division rate adjustment

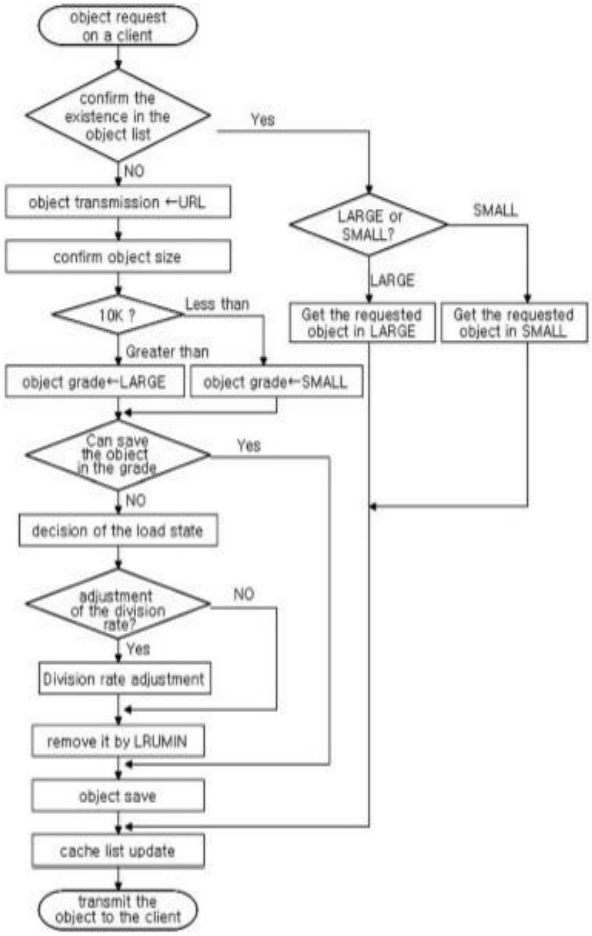

Fig 1: ACASH Algorithm

5. ACME

ACME (Adaptive Caching using Multiple Experts) [9] is inspired by problems existing in existing static caching algorithms such as data access latency due to disparity in between processor and storage I/O speeds, locality, lack of adaption to network topologies, changing workloads and increasing dynamic content. ACME treats existing cache replacements algorithms (such as LRU, RAND, MRU, FIFO, LIFO, LFU, SIZE, GDS, GDSF) as experts and registers them to a pool which has equal weights initially. In case of invention of new algorithms, they are added to the pool to test their proficiency. ACME does not propose new algorithms but uses existing algorithms more successfully. When the requests are manufactured by clients and the intensity improves, the weights are transformed by computationally simple but strong machine learning algorithms based on their achievements on hit rate or byte hit rate. Each node is an independent transformative organization. Because cache content information or synchronization messages are not

shared within peer caches, the clusters of these cache nodes will be scalable and easy to manage. This technique increases machine learning algorithms to improve caching algorithms in resolving insignificant problems [8]. It uses a pool of static cache replacement algorithms to join their relatively weak predictions into one highly-accurate prediction that aims to decide what objects remain in cache. Similar mechanics have been earlier used in resolving non-trivial complications with operating system.

The mechanism identifies a pool of virtual caches each of which simulates a single static cache substitution policy by sustaining an object ordering as if it possessed the complete physical cache. Each virtual cache uses keeps only the object metadata and not the actual object data with a purpose to save space. On each request, each virtual cache reports whether it would have got a hit (scored as 1) or a miss (scored as 0) as if it were real cache. This ranking is used to adjust the weights of the policies by increasing the weight of the objects that would have kept the object and lowering for people who would have dumped the object. Both caching and replacement are done based on votes. Each virtual cache votes on the objects it wishes to keep and assigning higher values to objects that it believes are most worth keeping. The objects with the highest weighted total vote stay in the cache. Over a period, the real cache ordering will probably appear like the ordering of virtual caches with highest weights, but will still be a mixture of a number of policies

One potential restriction that can be anticipated because of this approach is that virtual caches only keep as many objects that fit in the physical cache. In this case, a virtual cache with space for n objects will not contain an object that rank n+1 or n+100, where n is an object with rank of 1, which implies n is the most worthwhile object in the virtual cache. Therefore, it would be advisable to reward caches rather than rank reused objects and use virtual caches larger than the physical cache. By doing so, an object ranked n+1 in all virtual caches might be chosen over an object that is ranked n-1 in one cache and unranked in every other cache. If virtual caches were the same size as physical caches, the object with rank n+1 would be totally unknown and thereby, it will be ineligible for ranking. However, in a most likely scenario, the object with rank n+1 is more appealing and much more likely to be placed in cache than one with n-1 since so many policies rank it relatively extreme.

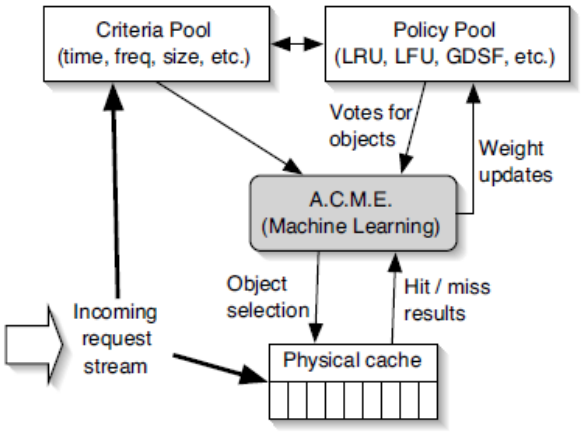

Fig 2 ACME Framework

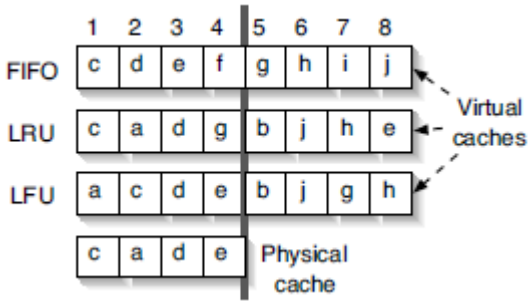

Figure 3 Virtual Caches and Physical Caches

6. NEURO-FUZZY SYSTEM IN PARTITIONED CLIENT SIDE WEB CACHE (ICWCS)

This procedure suggests that client-side caching is more affordable as well as efficient way to improve web overall performance of the web [14] due to the nature of browser cache that is closer to the user. A neuro-fuzzy system is a neural network that is functionally equal to a fuzzy inference model. A prevalent perspective in neuro-fuzzy development is adaptive neuro-fuzzy inference system (ANFIS) which is more compelling when compared with synthetic neural-networks (ANNs) and fuzzy systems. An intelligent client-side web caching scheme (ICWCS) is proposed to enhance the performance of the client side caching by splitting the client side cache into two caches, short-term cache and long-term cache. The splitting mechanism is implemented for storing the ideal web objects and eliminating undesirable objects in the cache for more effective storage. Also, ANFIS is employed to ascertain which objects in the long-term cache should be removed when the long-term cache runs out of capacity.

Neuro-fuzzy systems incorporate the parallel computation and learning abilities of ANNs with human-like knowledge representation and explanation abilities of fuzzy systems. A common technique in neuro-fuzzy development is ANFIS which has shown colossal functionality at binary classification tasks, being comparatively more profitable than other classification methods

The web cache is separated into short-term cache that receives objects immediately from the web and long term cache that receives objects from the short-term cache. In ICWCS, when a web page is viewed by the user, the objects relating to the page are stored in short-term cache originally. If the objects are accessed again, then the objects are transmitted to the long-term cache. This implies that if a web object is reused, it would be transferred to the long-term cache. Other objects will be managed by LRU algorithm. This ensures that objects that are used more often will be cached for a longer duration while lesser preferred objects would be removed to eliminate cache pollution and maximize hit ratio. ANFIS has a Sugeno-type fuzzy model. ANFIS is trained using a hybrid algorithm that uses least-squares estimator and the gradient ancestry method. In case, the long-term cache runs out of capacity, the ANFIS is employed in replacement process that classifies each object each object in long-term cache as cacheable or non-cacheable object. The old non-cacheable objects are removed first form the long term cache to make space for the incoming objects. If all objects are considered as cacheable, then the scheme will work like LRU policy. ICWCS can be converted from client cache to a proxy cache using minimum

effort. The distinction lies mainly in data size which is humongous at server end.

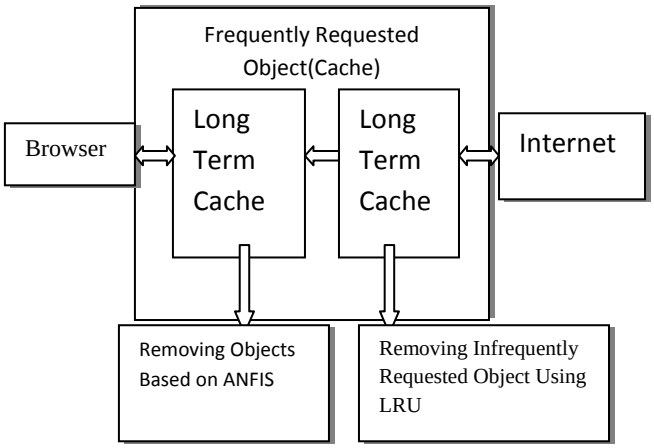

Figure 4 Framework of ICWCS

Some other intelligent web cache replacement policies are:

- **Neural Network Proxy Cache Replacement:** In NNPCR, artificial neural network has been used to make cache substitution decision. And object is selected for caching based on the rating given to it by the ANN. The restriction with this particular policy is that the objects with the same class are removed without any precedence anywhere between those objects.
- **Web cache optimization with non-linear model using object feature:** Artificial neural network has been used depending on syntactical features from HTML structure of the piece of content and the HTTP responses of the server as inputs. The constraints include that the rate of recurrence factor is ignored in web cache replacement decision. Also, this policy takes in some factors that do not manipulate web caching.
- **Adaptive Web cache access predictor using neural network:** The approach exhibits design of an intelligent predicator that used back propagation neural network to augment the performance of web caching by forecasting the most likely re-accessed objects established on history of accesses from the origin server and then keep them in cache. However, it disregards appearance period of the object.

7. SUMMARY OF ADAPTIVE CACHING ALGORITHMS.

Adaptive caching is a remedy to efficiency issues in the dynamic and complex web [4]. Each algorithm optimizes a different parameter to achieve greater efficiency in designated environment [7]. Because most of the results manufactured for caching algorithms are imitative, they may differ at some levels in real implementations due to lack of external limitations during simulation. Real implementations may require to minimize space and computational overheads and make performance trade-offs. Table 2 discusses few caching algorithms, their guidelines, brief and disadvantages

Table 1:Summery of Caching Algorithim

Caching Algorithm	Brief Description	Disadvantage
Fuzzy Inference System	Replacement policy based on Sugeno-type fuzzy inference system is employed that ranks objects using three parameters: Byte sent, Frequency and latency	Extra computation overhead, Implementation can be complex [2]
Logistic Regression Optimal cache algorithm	Logistic Regression function is used to statistically predict future accesses by learning access patterns from previous accesses	The objects with the lowest re-access probability value are replaced first regardless of cost and size of the predicted object [5] [20][3]
ACASH	Adaptively caches objects by dividing cache scope based on heterogeneity of web objects and adaptively reflects changes based in the reference characteristics of the objects according to the time elapsed	Does not take factors such as arrival time in account
ACME	Machine learning algorithms rate and select current best policies or mixture of policies via weight updates based on their recent success, allowing each adaptive cache node to tune itself based on the workload it observes	Virtual caches may only keep objects as much fit in physical caches making eligible caching candidates invalid when the cache fills [9]
Intelligent client side web caching system	Client-side approach that divides cache into two caches: short term and long term; The objects in short term cache are replaced using LRU and if the objects are re-accessed, they are relocated to long term cache where a neuro-fuzzy system is employed in case the long-term cache saturates	Training process requires long-time and extra computation; implemented at client-side [14] [6] [15]

8. CONCLUSION

Due to heterogeneous characteristics of the web, the algorithms suited for CPU cache (Such as LRU, LFU) are not conditioned to support web objects. Algorithms for CPU caches are more familiar to static and consistent objects. However, the web has an ever-changing dynamic and complicated structure and more dynamic algorithms are required. Adaptive web caching techniques need to be built that use machine learning to adapt to the important changes in usage behaviors as well as store appropriate objects in cache while utilizing the cache space thoroughly. In this particular document, we survey various adaptive caching techniques and

algorithms while discussing their pros and cons that may help to reduce the bottleneck in data transmission through web

9. ACKNOWLEDGMENTS

Especially, please allow me to dedicate my acknowledgment of gratitude toward the significant advisors and contributors; I would like to thank Dr. Shamsher Singh for his most support and encouragement. He kindly read my paper and offered invaluable detailed advices on grammar, organization, and the theme of the paper. Second, I would like to thank GL Saini to reproof the paper, Finally, I sincerely thank to my parents, family, Department of Computer Science and Engineering, Amity School of Engineering and Technology, Amity University, Rajasthan India and friends, who provide the advice and financial support. The product of this research paper would not be possible without all of them.

10. REFERENCES

[1] Achuthsankar S. Nair, J.S. Jayasudha, "Improving Performance by World Wide Web by Adaptive Web Traffic Reduction", Proceedings of World Academy of Science, Engineering and Technology, Volume 17, December 2006

[2] Anish Kumar Saha, Partha Pratim Deb, Moutishi Kar, D. Rudrapal, "An optimization technique of web caching using Fuzzy Inference System", International Journal of Computer Applications, Volume 43-No.17, April, 2012

[3] Annie P. Foong, Yu-Hen Hu and Dennis M. Heisey, "Adaptive web caching using logistic regression", IEEE, 1999

[4] Athena Vakali, George Pallis, "A study on web caching architectures and performance"

[5] Dhawaleswar Rao. CH, "Study of the web caching Algorithms for Performance Improvement of the response speed", Indian Journal of Computer Science and Engineering", Volume 3 – No. 2, April-March, 2012

[6] Farhan Mohamed, Abdul Samad Ismail, Siti Mariyam Shamsuddin, "Web caching and prefetching: Techniques and analysis in World Wide Web", Proceedings of the Postgraduate Annual Research Seminar, 2005

[7] Hossam Hassanein, Zhengang Liang and Patrick Martin, "Performance comparison of Alternative Web Caching Techniques", Proceedings of the Seventh International Symposium on Computers and Communications, 2002

[8] https://en.wikipedia.org/wiki/Bloom_filter

[9] Ismail Ari, Ahmed Amer, Robert Gramacy, Ethan L.Miller, Scott A. Brandt, Darrell D.E. Long, "ACME: Adaptive Caching using Multiple Experts"

[10] Lixia Zhang, Sally Floyd and Van Jacobson, "Adaptive Web Caching", April 25, 1997

[11] S.Sulaiman, Siti Mariyam Shamsuddin and A.Abraham, "Intelligent web caching using Adaptive Regression Trees, Splines, Random Forests and Tree Net"

[12] Scott Michel, Lixia Zhang, Sally Floyd, "Adaptive web caching: Towards a new global caching architecture", Computer Networks and ISDN Networks, November 1998

[13] Sean C. Rhea, Kevin Liang, Eric Brewer, "Value based web Caching", May 20-24, 2003

[14] Waleed Ali, Siti Mariyam Shamsuddin, "Neuro-fuzzy system in partitioned client-side web cache", Expert systems with applications, November 2011

[15] Waleed Ali, Siti Mariyam, Shamsuddin, Abdul Samad Ismail, "A Survey of Web caching and Prefetching", International Journal of Advanced Soft Computing Applications, Volume 3- No. 1, March 2011

[16] Yun Ji Na, Choon Seong Leem, Il Seok Ko, "ACASH: an adaptive web caching method based on the heterogeneity of web object and reference characteristics", Information Sciences, May 20, 2005