Scientific Research Publishing

Journal of Software Engineering and Applications, 2026, 19(3), 105-129
https://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

Investigation of Object-Based Web Caching Using a Framework Model for User Request Access Patterns

Richard Hurley, Robert Sturgeon

Department of Computer Science, Trent University, Peterborough, Canada
Email: rhurley@trentu.ca, rsturgeon@trentu.ca

How to cite this paper: Hurley, R. and Sturgeon, R. (2026) Investigation of Object-Based Web Caching Using a Framework Model for User Request Access Patterns. *Journal of Software Engineering and Applications*, **19**, 105-129.
https://doi.org/10.4236/jsea.2026.193006

Received: February 11, 2026
Accepted: March 22, 2026
Published: March 25, 2026

Abstract

In this paper, we utilize a novel framework model that generates the wide variety of user object-level request access patterns that are prevalent in the World Wide Web. The framework model consists of three sub-models: one for user file access, one for Web pages, and one for storage servers. Web pages are assumed to consist of different types and sizes of objects, which are characterized using several categories: articles, media, and mosaics. The framework model was developed to be general enough to be applicable to various distributed applications that are present in the WWW. The distributed application to which we chose to apply the User Request Access Pattern Model is Web caching. Web caching is a system that is intended to improve user website experience by reducing response times, improving availability, lowering server load, and reducing bandwidth. We then use a discrete-event simulation to investigate the performance of an object-based Web caching application under a variety of user request access patterns.

Keywords

User Request Access Patterns, Performance Modelling, World Wide Web, Simulation, Web Caching

1. Introduction

With the explosion in the number of mobile devices, tablets, laptops, etc., there has been a surge in WWW traffic stemming from users accessing information on websites. The type of information can range from the latest sports scores to weather forecasts to generative AI requests. The items requested (*i.e.*, objects) vary by size and number depending on the type of request. One of the largest impacts on a

distributed system will be an increasing load on Web servers resulting from the stream of user requests for information that can be found in the various Web pages (see Figure 1, which presents a simplified diagram that illustrates a user initiating a request to a Web server for a Web page).

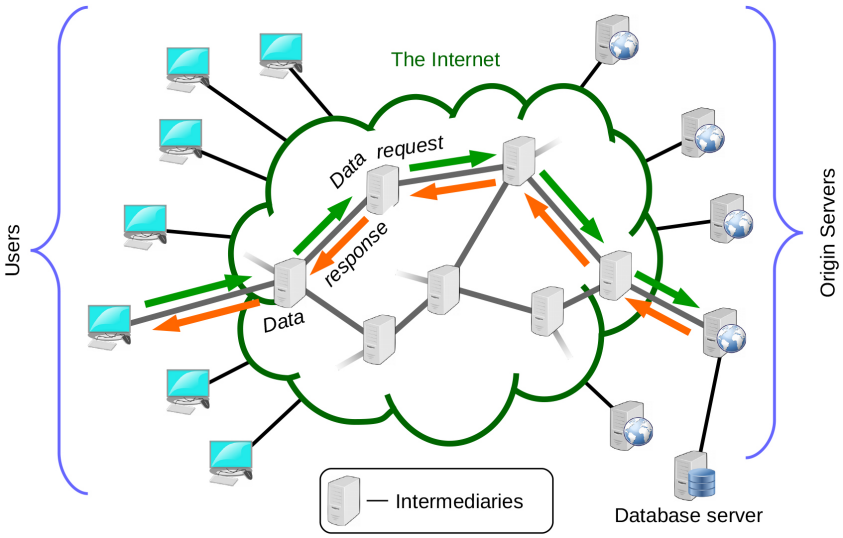

Figure 1. Simplified World Wide Web showing an example request/response chain.

User request access patterns also vary tremendously, given that Web pages themselves are vastly different in terms of the variety of objects that make up a Web page: text, images, audio, video, code, etc. [1]. It is important to be able to incorporate the wide variety of user access patterns when investigating the performance of distributed applications, such as Web caching [2] [3].

For this work, we utilized the user access pattern framework model developed by Sturgeon [3] to investigate the performance of an object-based Web caching system. The framework captures a wide variety of user (*Hypertext Transfer Protocol* (HTTP)) request patterns in the *World Wide Web* (WWW), which allows one to compare the relative performance of distributed applications such as Web caching [3]-[5]. The reality of the WWW is that it is complex: it involves the transmission of large and varied amounts of information via the Internet, through a variety of devices spanning many regions, countries, and even into space. The framework model attempts to capture much of this and allows for the relative comparison of the performance of various system configurations.

There has been some other research on modelling user request access patterns on the Web, but it is limited: most of it focuses on the Web page as the basic unit [6]-[8]. The work is done in [3] models at the object level and it is for that reason why this model will be used as a basis in this paper.

The distributed application that we will be investigating to incorporate the user access pattern framework model is Web caching: an important application that involves the temporary storage of frequently accessed Web page objects at various intermediary storage locations between users and Web servers. By storing Web

page objects *closer* to a user, performance benefits can be achieved (see Figure 2) [5] [7] [9]. The investigation is novel as we are investigating Web caching at the object level: a much finer granularity than an entire Web page. There has been some work done in this area in the past, but the focus was on miss ratios [10], where our performance criteria will be user response time.

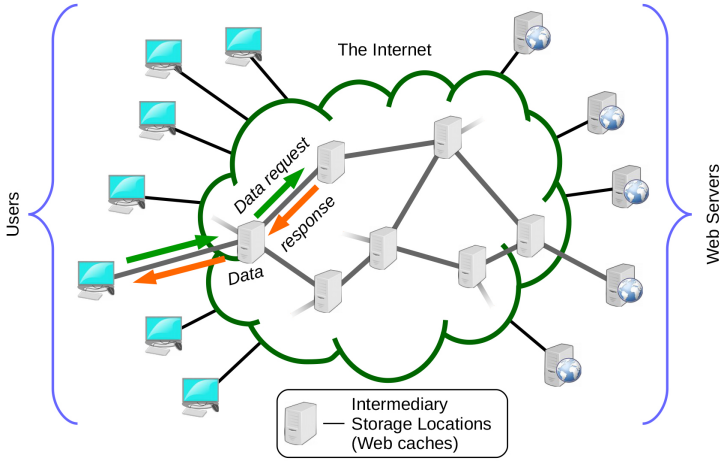

Figure 2. Example of a Web page request receiving a cached response.

The principle of operation for Web caching is as follows: as the response to a request for information travels via the Internet to the user, copies of Web page objects that make up the response are stored at the intermediary storage locations so that subsequent requests for the same information can be fulfilled by intermediary storage locations. This can reduce the time to fulfill the next request. Not all information can be stored in intermediary storage locations because of limited cache size and the fact that certain types of information change so rapidly, thus making caching impracticable [11].

For this paper, we used a discrete-event simulation model to investigate the performance of an object-based Web caching system utilizing an expanded model of user request access patterns. This paper is organized as follows: Section 2 gives an overview of the system, Section 3 summarizes the User Request Access Pattern Model, Section 4 introduces the Object-Based Web caching model with results, and finally, in Section 5, we present some concluding remarks.

2. System Overview

At the highest level, as shown in Figure 1 and Figure 2, we consider the WWW to consist of the following components: users, Web servers, the Internet, and the information being requested. The term *user* will refer to an agent that is requesting information—human or machine [12]. The *Internet* is the communications medium that interconnects the users and Web servers, typically through a series of intermediaries. Finally, *Web servers* are the devices that store the information in the form of Web pages and transmit results to the user to satisfy a request.

The composition of a Web page can be highly varied, ranging from a monolithic document of plain text to a complex collection of elements of various forms. Each Web page must contain at least one element, but normally consists of many elements called *Web page objects*. The Web page therefore can be viewed as a group of Web page objects. Web page objects have several properties, but for our study, we are primarily concerned with their size, as this has a direct effect on the retrieval of the Web page. There are dozens of types of Web page objects, but some of the common ones that make up a Web page are: *Hyper- Text Markup Language* (HTML), JavaScript, *Extensible Markup Language* (XML), *Cascading Style Sheets* (CSS), image, audio, video, octet [1] [7].

An important consideration regarding Web pages is that their content can change with time (referred to as *dynamic content*) [13] [14]. Although the specific content on a Web page may change, it still represents a single resource [15]. For example, a news main page such as https://www.cnn.com/ consists of a collection of news articles available on the site at some point in time. The collection, however, changes with time as current news articles are added and old ones are removed.

Another component within the WWW is the *Web server*, which has an important impact on user experience. Web servers, which range from a single computer to a collection of networked computers, hold the Web pages that users request [16]. The speed at which they can respond to requests contributes in large part to the user response time.

In a Web caching system, additional storage is provided at intermediary locations that serve as an extension of the Web servers in the storage hierarchy. These Web caches store copies of frequently accessed objects from the Web pages stored on the Web servers. The basic premise is that user response time would be improved if users are able to retrieve some of the objects from the Web caches as opposed to having to access them from the Web servers [17].

The final component of the system is the Internet. We recognize that the Internet is a complicated collection of protocols for communicating across arbitrary packet-switched networks, and is made up of hosts, routers, and networks [18]. That said, since we will be focusing on relative as opposed to absolute performance, we will abstract the network delay into the service time for a user request.

3. User Request Access Pattern Model

In this section, we present an overview of the framework model that we will be incorporating into our Web caching model. The User Request Access Pattern Model represents the process of users interacting with the WWW to obtain Web pages from the respective Web servers. One of the novel features of the model is that Web pages are represented at the object level. The main performance metric in which we are interested is *Mean Response Time* (MRT): the time from when a user requests a Web page until the Web page (and all of its objects) has been delivered. This paper will provide just enough detail on the model and its parameters so as to provide

the context for its application to our Web caching model. For a detailed description and analysis of the User Request Access Pattern Model, please see [2].

3.1. High-Level Model

The high-level model for user request access patterns is shown in **Figure 3**. This represents the total process of a user requesting a Web page from a server, and the server's subsequent response. To capture the behaviour of the users, servers, and Web pages, the model is composed of three sub-models: *User-Request Model*, *Server Model*, and *Web Page Model*. The User-Request Model manages the state of the users and their requests. Requests are received and processed by Web servers within the Server Model. The Web Page Model captures the Web pages in terms of their object composition and location.

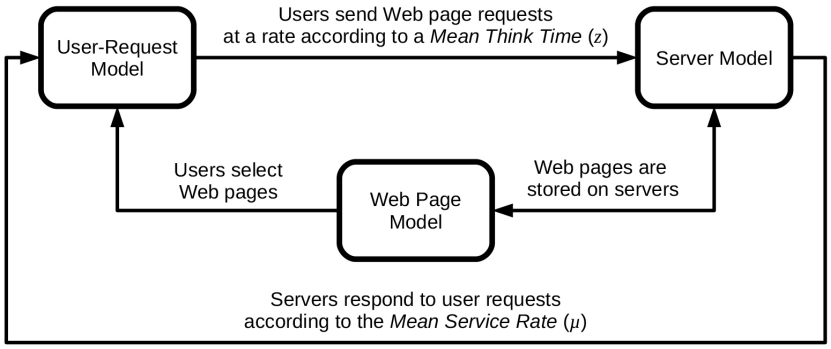

Figure 3. User request access patterns model.

The interaction of the three sub-models forms the *request-response process*. This process begins with the user selecting a Web page using the Web Page Selection Model (described in Section 3.2). The Web page request is transmitted to the Web server that will eventually supply the requested Web page to the user (Server Model described in Section 3.3). The time to service a request is based on many factors, including the composition of the Web page, the number and size of the Web page objects, as well as their media type.

Once the user receives the response to their request, they cease interacting with the system until their next request (based on the user's *think time*). The *think time* controls the rate at which users request Web pages. When not in the system waiting for a request to be served, a user is in a think state, which represents a user processing a previously retrieved Web page. A user will not request another Web page until they have completed their think state.

For this research, we are interested in *response time*: the time from the user initiating the request until receiving the response. Response time represents the sum of the transmission latency, the time in queue that the request awaits processing, and the request processing time.

3.2. User-Request Model

The User-Request Model represents the process of a user selecting and requesting

a Web page. This sub-model consists of two main components: the *User Set* and the *Web Page Selection Model*. The purpose of the User Set is to model the users in the system and how they interact with the WWW. We use a finite population model which incorporates the user think time in the request-response process as discussed in Section 3.1 [19].

The Web Page Selection Model represents the mechanism by which users select Web pages. The pages stored in a Web cache and their request probabilities vary over time. Pages such as news articles, viral videos, course assignments, memes, etc., become popular for periods of time and then eventually become accessed less frequently. To represent this behavior, we use a variant of the dynamic page reference model described in [20].

The dynamic page reference model for a system with M Web pages is shown in Figure 4. This model assumes that a Web page can be in one of two states: normal and popular. Web pages in the popular state have a higher request probability than that of the pages in the normal state, with v representing the ratio of page requesting rates for the popular to normal state.

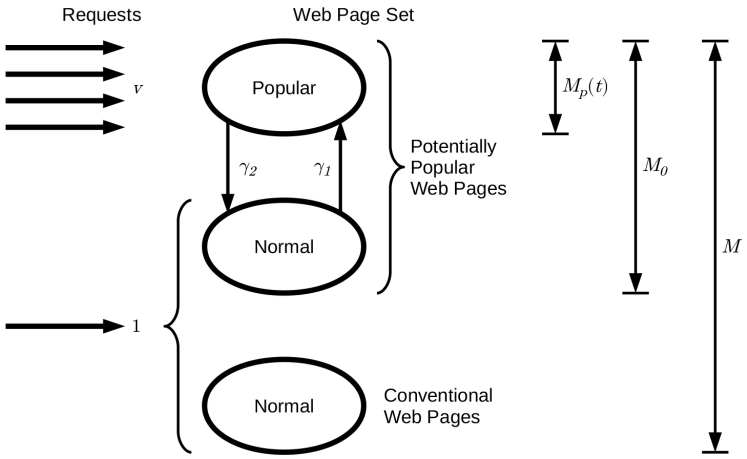

Figure 4. Web page selection model.

The model also assumes that there are two types of pages: conventional and potentially popular. Conventional pages remain in the normal state while potentially popular pages alternate between the normal and popular state based on a continuous-time Markov chain. Finally, we let $M_0 < M$ denote the number of potentially popular pages that are present in the system. With this type of model, the model is able to generate page requests with high coefficients of variation, an attribute that has historically shown to be desirable in such systems [20].

3.3. Server Model

Figure 5 presents the details of the Server Model. The model consists of one or more devices that store the Web pages and subsequent objects, and serve the requests. The role of each Web server is to process requests and transmit the response back to the user (in the form of the Web page and all of its individual objects) If a request

is received at a server that is busy, then the incoming request is placed in a wait queue (we assume *First-Come-First-Serve* (FCFS), however, this discipline may be modified to suit the desired investigation). Once a server completes a request, it then looks to the wait queue for the next request. If the wait queue is empty, then the server sits idle waiting for the next request.

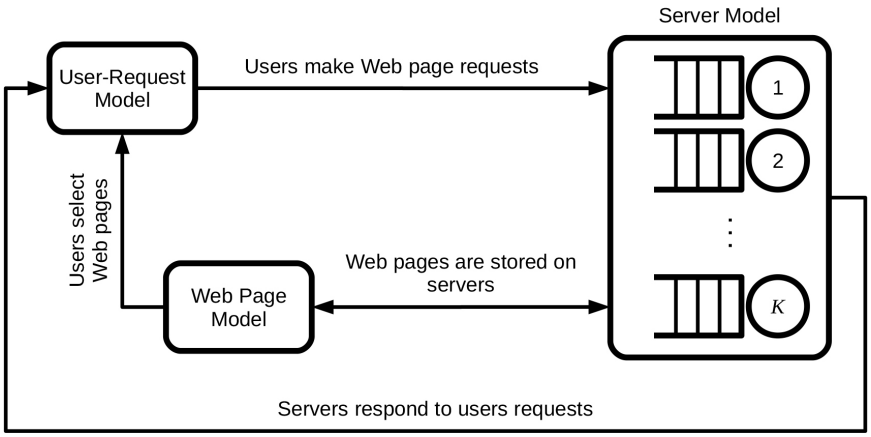

Figure 5. Web page request model, showing FCFS Web server queues.

The *request service time* (R_i), the main parameter of the Server Model, is the time to process and retrieve Web page i and transmit it to the user. The request service time (R_i) is dependent on Web page size (W_i), which in turn is dependent on the number and size of the objects from which the Web page is comprised (denoted by Q_i and Θ_{ij}, respectively). We assume that the Web page size (W_i) is equal to the transmission time (τ_i) to arrive at Equation (1).

$$\tau_i = W_i = \sum_{j=1}^{Q_i} \Theta_{ij} \tag{1}$$

The Web page size (and hence transmission time) is equal to the sum of the object sizes, where Web page i consists of a unique set of Q_i objects, and each object has a size Θ_{ij} The number of objects (Q_i) and their object type are an important aspect of our model, and one that can be varied as needed to emulate a variety of different environments.

We should note at this time that for Web caching, we simply substitute a different Server Model that incorporates the intermediary storage with the appropriate caching algorithms (see Section 4).

3.4. Web Page Model

The Web Page Model represents the totality of information that is requested and delivered to the user. It is represented as a collection of Web page objects that are transmitted from the Web server to the user upon request. We assume that a Web page request is not fulfilled until all objects that compose the Web page are received. Each Web page is unique and consists of a *Web Page Object Set*, which is determined by the *Web page category* (see Section 3.4.2).

3.4.1. Web Page Object Set

A Web page consists of a collection of one or more objects, which we refer to as the *Web page object set*. The composition of the object set is assumed to be determined by the number and type of objects. Each object consists of two main attributes: *object type* and *size*. The object type and size are based on the work of Ihm [7] and summarized in Table 1.

Table 1. Example Web page object parameters.

Object Type	Size	Examples
text	small	HTML, XML, text
script	small, medium	JavaScript
css	small	CSS
image	small, medium	.png, .gif, jpg
audio	small, medium	.ogg, .mp3, .wmv
video	medium, large	.flv, .mov, .mp4

While this set is not exhaustive, these six object types account for the majority of information being transferred via the WWW [7]. Recall that object sizes directly affect the Web page size, and thus influence the request service time.

3.4.2. Web Page Categories

The Web page category provides a means to define the composition of Web page objects within the model by specifying how objects are distributed based on their object type and size within the object set of each Web page. As noted in a previous section, the category of a Web page directly affects the retrieval time due to the potential differences in Web page sizes.

In this model, we specify three Web page categories: *article*, *media*, and *mosaic*, but recognize that other page categories and their compositions are certainly possible.

Article: Typically contains information about one specific topic. The composition consists of a large amount of script, text and images, but has very little audio or video information. Examples: news articles, Wikipedia article pages.

Media: While this category is similar to articles, the main

difference is that a media page provides more audio and/or video. Examples: image collages such as Google Images, Flickr, Instragram, Pinterest; image pages, music and video streaming.

Mosaic: Web pages that provide multiple links and summaries to other specific topics. Their size is primarily influenced by text and script, with some image information and a moderate amount of audio and/or video. Examples: search results such as Web searches, Pinterest image search, website main pages, and social media threads.

The breakdown of Web pages into Web page categories is an input parameter in our model. The Web pages in each category are configured according to an

input table that describes the way in which objects are to be distributed. Figure 6 shows a graphical example of this configuration for M Web pages and their distribution using the Web page categories. It also displays one example Web page from each of the three Web page categories to illustrate how the page category affects the object distribution.

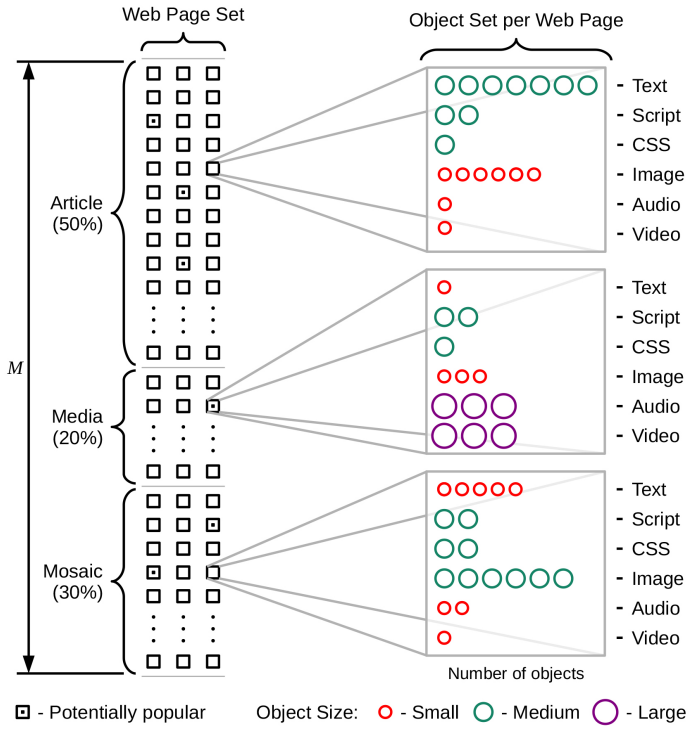

Figure 6. Example Web page composition based on category.

From Figure 6, we can observe that an *article* Web page is dominated by several text objects, each of a medium size, as well as several small images with only a few of each of the other objects. In contrast, *media* Web pages are dominated by a moderate number of large audio and video objects. The *mosaic* Web page, however, has several medium-sized image objects and a moderate number of other objects.

3.5. Important System Parameters

Based on the User Request Access Pattern Model, we developed a discrete-event simulation to investigate the impact of the various key parameters in order to determine a subset of which could be used to examine other distributed applications (such as Web caching). A detailed investigation and analysis of the model can be found in [2]. We will just summarize the results of the investigation in this section.

We start by examining the effects of the model parameters for Web page composition to determine their relationship to our main parameter of interest, MRT. Several key parameters contribute to the Web page composition, including: Web page category, object size, object type, and number of objects. Ultimately, the goal

is to establish the sizes of the Web pages in our system, which vary according to Web page categories.

In the previous two sections, we presented the *Server* and *Web Page* models, where the parameters that compose the Web Page Set were introduced. Now, in an attempt to further clarify the relationship between the parameters, we provide a visual summary (see Figure 7).

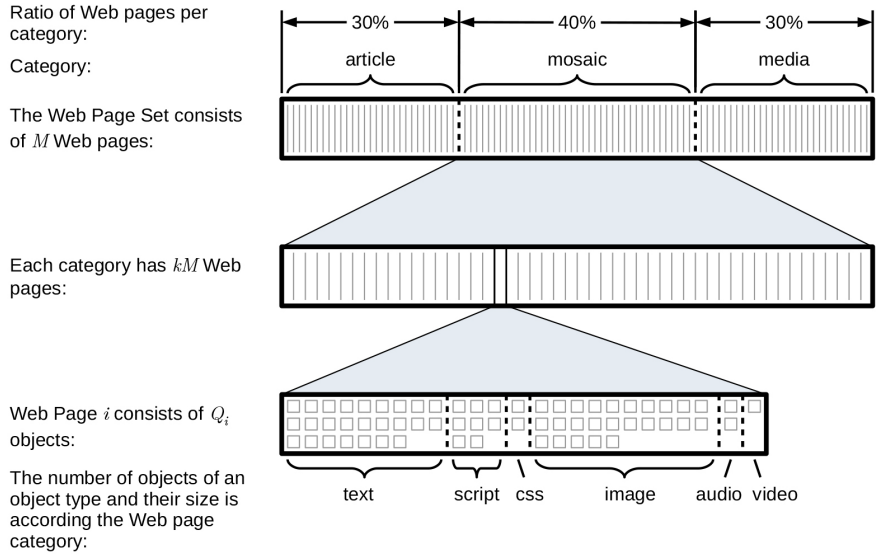

Figure 7. Visual summary of the composition of the Web page set.

The Web Page Set consists of M Web pages, where each Web page has a composition that follows one of three Web page categories: *article*, *mosaic*, *media*. The number of Web pages of each category is determined by the *Ratio of Web Pages per Web Page Category*. Each Web page *i*, has a *Web Page Object Set* consisting of *Qi* objects of various types and sizes. The object quantity, types and sizes for a given Web page are determined by the Web page category. The sum of the object sizes in the Object Set determines the size of each Web page (W_i).

The Ratio of Web Pages per Category describes how many Web pages of each category there will be in the Web Page Set (ratios are expressed as a percentage of M). We denote k_{category} to be the proportion that a *category* represents in the Ratio of Web Pages per Category. We initially assume the following ratios: $k_{\text{article}} = 30\%$, $k_{\text{mosaic}} = 40\%$, and $k_{\text{media}} = 30\%$ (this is referred to as our *Base* ratio scenario). These values were estimated to result in approximately the middle of the range of possible mean Web page size ($\overline{W}$) for all requests. These are simply parameters and we will examine other ratio scenarios at the end of Section 4.

In order to determine the size of each Web page (W_i), the trace analysis from [7] was used as a starting point. For each of the categories: *article*, *mosiac*, and *media*, a size and number of objects was established. For the User Request Access Patterns Model, the range of sizes was captured with three classes: *small* (Θ_{small}), *medium* (Θ_{medium}), and *large* (Θ_{large}).

Since we are only concerned with relative performance, we assume that $\Theta_{small} = 1$, and $\Theta_{large} = 100$. For Θ_{medium}, the results from [21] on the relative size of various documents were used to determine that 15 would be an appropriate choice. Finally, we assume the small object size ($\Theta_{small} = 1$) to be the base unit for all measures of data size in our model.

Web page categories are a key feature of the model as they allow for the characterization of the effect that different compositions of Web pages have on MRT. The size per Web page in each category should be sufficiently different from those of other categories as to impact the MRT when the Ratio of Web Pages per Category varies.

After investigating several scenarios and using the results from [7], we chose to use $W_{article}$ as the basis. Given this, we set the Web page size of W_{mosaic} to be approximately 4 times larger than $W_{article}$ and W_{media} to be about 8 times larger. These values are slightly higher than those determined by [7], as we wanted to exaggerate the effect. Again, the absolute value of $W_{article}$ is not important, as again we are only concerned with the relative size differences between the Web page categories.

The last parameter required to establish the Composition of the Web Page Object Set is the Number of Objects that a Web page contains. To accomplish this, one additional assumption was made based on the results from [22]: the average Number of Objects per Web page is approximately 100.

With these assumptions in place, **Table 2** displays the Number of Web Page Objects (Q_i) and Web page size (W_i) for Web pages based on their category.

Table 2. Number of Web page objects and Web page size for each Web page category.

Category	Q_i	W_i
Article	80	80
Mosaic	138	320
Media	63	640

The number of objects across the three categories (80, 138, and 63) has an average of 94, which is reasonable, given the target of 100 [22]. The values from **Table 2** for the object composition will be used for the simulation experiments.

The coefficient of variation of Web page request interarrival time (CV) is a metric used to characterize the variability of the Web page request process (coefficient of variations greater than three are common [5]). The parameters used to control this value come from the Web Page Selection Model (**Figure 4**). They are: the number of potentially popular Web pages (M_0), the popularity transition rate (γ_1 and γ_2), and the popularity factor (v). The previous work by [2] assisted with setting these parameters to appropriately generate a CV ≈ 3 for our examination of Web caching.

Finally, we define system load (ρ) to be the mean utilization over all servers in the system. It is influenced by several parameters, most notably: number of users

(N), mean Web page size for all requests ($\bar{W}$), number of servers (K), and Coefficient of Variation (CV). It is through changes to these submodel parameters that we are able to generate various system loads (ρ) for our investigation of object-based Web caching.

4. Object-Based Web Caching

In this section, we will investigate the performance of an object-based Web caching system that utilizes the detailed User Request Access Pattern model. More specifically, we will incorporate the results from Section 3.5 into our analysis.

4.1. Model

As mentioned in Section 3.3, Web caching is introduced by substituting the Server Model in Figure 5 with an expanded version that includes object-based Web caching, the results of which are shown in Figure 8.

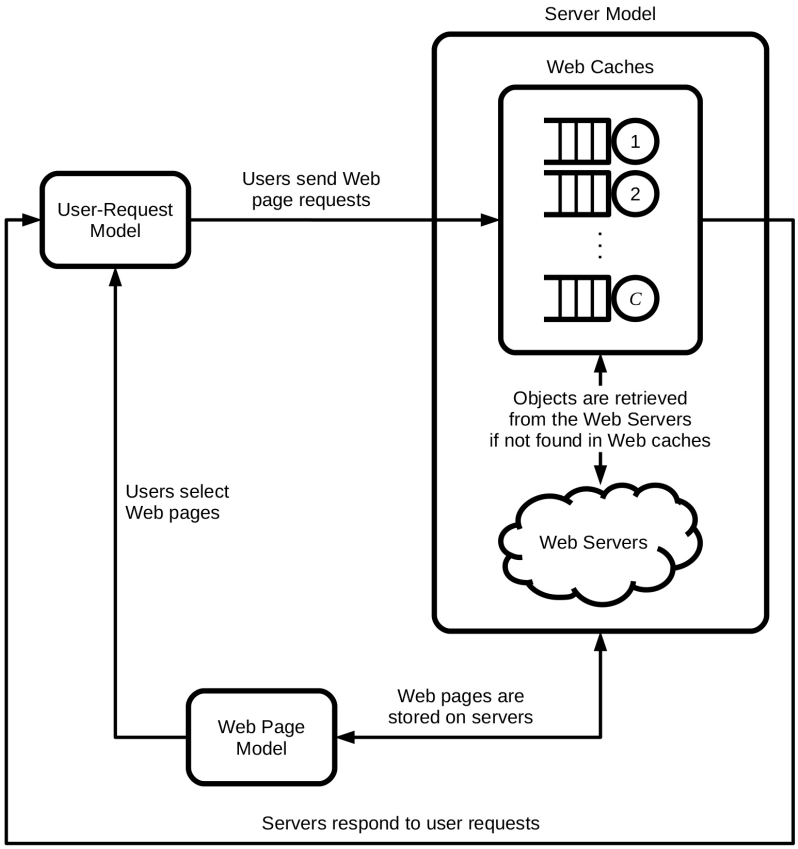

Figure 8. Web page request model with caching model.

The expanded Server Model consists of a homogeneous set of C Web caches, where each cache has its own queue and maintains a list of objects that it stores. Requests for Web page i are decomposed into a sequence of requests for each object in the Web page. These requests are received by the Web caches, eventually returning the required objects to the user.

This model also assumes that the transmission time for a cached object is less than that of an object had it been retrieved from the origin Web server. Each origin server also contains its own queue and processes the requests from the Web caches in FCFS order. A new parameter, the *Cached Object Access Factor* (δ), where $\delta \leq 1$, is introduced, which will allow us to modify the transmission latency for cached objects.

Recall from Section 3.3, that request service time (R_i) is proportional to the Web page size (W_i) and hence, the transmission time (τ_i). Equation (2) expands on Equation (1) to include the Cached Object Access Factor for objects that are cached.

$$\tau_i = \sum_{j=1}^{Q_i} \begin{cases} \Theta_{ij}, & \text{if object is in not cache,} \\ \delta\Theta_{ij}, & \text{if object is in cache.} \end{cases} \tag{2}$$

The cacheability of objects can vary greatly, from being easily cached to not being cacheable at all. Cacheability is influenced by a myriad of considerations, from the type of content to the frequency of updates to the server's caching strategies [4] [7] [14]. We capture a Web page object's cacheability with the parameter φ. This parameter represents the probability that an object is copied to cache after being requested, and is based on its Web page category and object type.

For flexibility, our model can have multiple Web page object cacheabilities—potentially one for every combination of Web page category and object type. To simplify, however, we propose three object cacheability levels for objects: *not cacheable* (φ_{not}), *less cacheable* (φ_{less}), and *more cacheable* (φ_{more}).

We have modified **Table 1** to include the Web page object cacheabilities (see **Table 3**). Notice from **Table 1**, that since our interpretation of the *text* object type incorporates several Web page object media types that represent dynamic objects, we assume that text objects are not cacheable (the only object type where this applies).

Table 3. Example Web page object parameters.

Object Type	Size	Cacheability
text	small	not cacheable
script	small, medium	less cacheable
css	small	more cacheable
image	small, medium	more cacheable
audio	small, medium	more cacheable
video	medium, large	more cacheable

Based on observations in [7], we assume the values of our object cacheabilities to be: $\varphi_{\text{not}} = 0\%$, $\varphi_{\text{less}} = 25\%$, and $\varphi_{\text{more}} = 75\%$. These values are interpreted as follows: for an object that is not cacheable, it will have to be retrieved from the origin server and there is no chance it will be copied to the cache. For a *less cacheable* object, when it is not already cached and has to be retrieved from the origin server, there is a 25% chance that it will be copied to cache. Finally, there is a 75% chance

of caching the object if it is in the *more cacheable* category.

Now, with respect to the individual Web cache servers, we assume that they each have a *Cache Capacity* which is represented by the parameter κ. When the sum of the cached object sizes in a server reaches the Cache Capacity, a *cache replacement policy* is utilized to determine which object (or objects) are to be removed to make room for incoming objects. Cache Capacity (κ) is assumed to be a percentage of the total cache space in a server. A κ value of 90% indicates that the cache is 90% full. Given that we are simulating the system, equilibrium implies that the initial transient (time to fill the cache) period is ignored.

For our model, we chose to implement a *Least Recently Used* (LRU) algorithm [4] as the cache replacement algorithm. To accomplish this, we track at which time objects are last accessed and remove the objects with the oldest reference time until there is enough room for the incoming object.

The algorithm we use for Web caching is summarized in Figure 9. It starts with a request for Web page i and ends with the transmission time (τ_i) for the Web page. For objects that have been cached, the transmission time includes the scaled value of $\delta\Theta_{ij}$; otherwise, the object size (Θ_{ij}) alone is added to the transmission time.

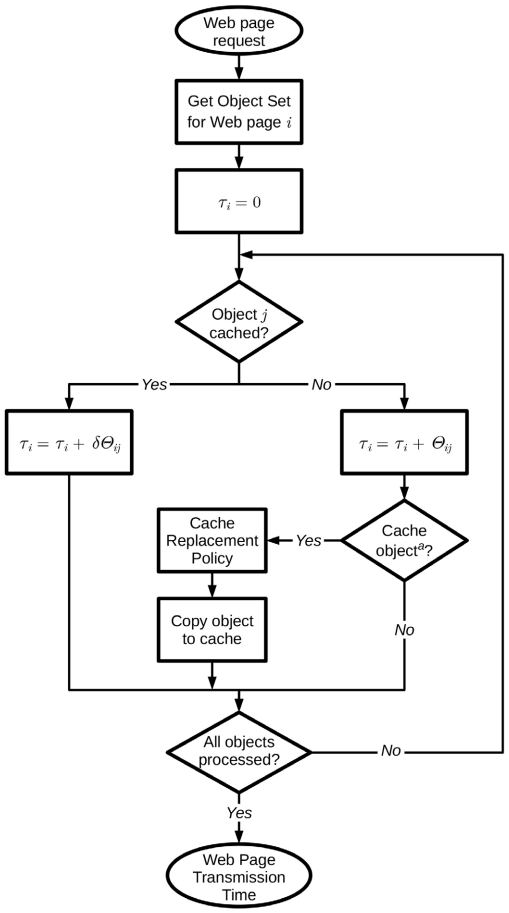

Figure 9. Process to calculate Web page transmission time with object level Web caching.

To complete the process, if an object is not already cached, it is considered for caching according to its Web page object cacheability. Before objects are copied to the cache, the cache replacement policy is used if there is not enough room in the Web cache server.

One final comment about the Web Caching Model: the intention of this work is to demonstrate the application of an enhanced Framework Model for user request access patterns. The distributed application that we have chosen is Web caching. The Server Model from Section 3.3 is expanded to include a vanilla-type object-level Web caching. We do not incorporate any of the advanced features of web caching, such as parallel object fetching, connection reuse, or HTTP/2/3 multiplexing. We are only focused on the relative performance using our Framework Model with Web caching against a system without Web caching. Incorporating advanced Web caching features such as parallel object fetching, connection reuse, and/or HTTP/2/3 multiplexing would most likely lead to a reduced mean response time. This would be a good topic for future research.

4.2. Simulation

As mentioned previously, we use a discrete-event simulation to test our model and conduct experiments. In this section, we will briefly discuss its implementation, validation and verification, and consider various model parameters and their impact on our main performance measure of interest, MRT. We evaluated our simulation in two ways: by conducting an execution trace, and by comparing simulated to analytic results (this will be discussed in Section 4.3).

A discrete-event simulation is a technique by which events are managed at discrete time units. In the simulation, time is captured by a global clock, which is advanced each time an event is processed [19]. Between each event, the status of the system remains unchanged. Events are generated randomly and added to a *Future Events List* (FEL) that is ordered by time such that next available event has the lowest time. A more in-depth discussion of the simulation can be found in [3].

4.3. Validation and Verification

To verify that our simulation was implemented correctly, we examined detailed sample output data from an execution trace. The parameters for this trace were chosen to permit a manual review of the data to be accomplished in a reasonable amount of time. For this trace, we used four Web servers, 100 users, 100 Web pages, and a simulation length of 200. We followed events and reviewed the status of each Web page request. The focus during this review was to ensure that each request went to the correct server, verified timings, including request service time, and monitored that the Web server queues were correct. Through this trace, we verified that the Web page request chain was fulfilled correctly.

In addition to the execution trace, we also compared our simulation results to analytic results. This was accomplished by running our simulation as a $M/M/1//N$ queuing system. We used one Web server with a think time of 100, the number of

Web pages set to 100, and zero potentially popular Web pages. In order to achieve a mean service rate of 1, we used a single Web page category and single Web page object media type that has a size of 1. Results of this experiment are shown in Table 4 and do establish that the scaled-down version of our simulation model is correct, as our simulated mean response time does seem to match that computed from analytical model (with a 95% confidence interval).

Table 4. Comparison of simulation MRT versus analytic MRT for an $M/M/1//N$ system for various values of N. $K = 1$, $\mu = 1$, $z = 100$, $M = 100$, $M_0 = 0$.

N	Simulated		Analytic Mean
	Mean	95% Confidence Interval	
10	1.10	[1.10, 1.10]	1.10
25	1.31	[1.30, 1.31]	1.31
50	1.90	[1.89, 1.90]	1.90
75	3.30	[3.30, 3.30]	3.31
100	8.18	[8.17, 8.19]	8.19

4.4. System Stability

The goal of examining system stability is to establish a simulation length that provides stable results (not affected by the transient period). We assume that when the performance measure (such as coefficient of variation) converges with the expected value, the simulation has reached equilibrium.

In Figure 10, we plot the measured CV value to an expected CV over various simulation lengths for $K = 5$ and $K = 10$ Web servers. These two scenarios appear to converge at a simulation length of around 30×10^6. Thus, going forward, we assume a reliable value of simulation length for experiments to be 80×10^6.

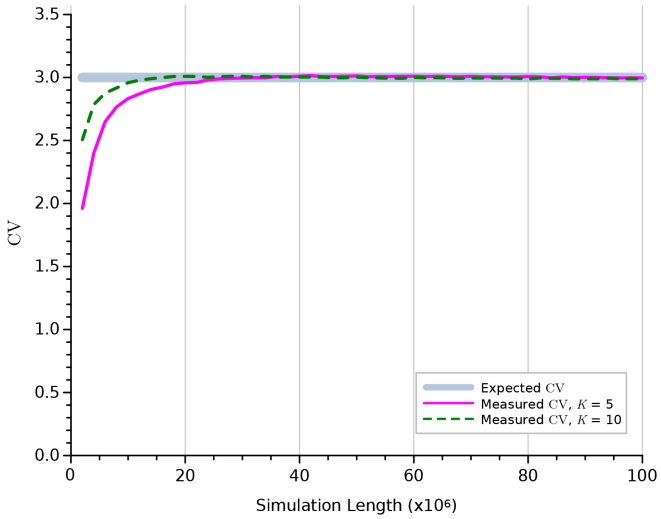

Figure 10. Effect of simulation length on coefficient of variation of Web page request interarrival time (CV). $v = 90$, $\rho \approx 85\%$ ($K = 5$, $N = 480$, $z = 35{,}000$.

4.5. Coefficient of Variation

The *coefficient of variation of Web page request interarrival time* (CV) is a metric we use for characterizing the Web page request process (coefficient of variations greater than three are common [20]). The parameters we use to control the coefficient of variation are: the *number of potentially popular Web pages* (M_0), the *popularity transition rates* (γ_1 and γ_2), and the *popularity factor* (v). For the number of potentially popular Web pages (M_0), we assume $M_0 = 0.1M$ (10% of files are potentially popular). Through experimentation, we found that the popularity transition rates $\gamma_1 = \gamma_2 = 200{,}000$ seem to achieve a range of CV from 1 - 5.

Figure 11 shows the effect that the coefficient of variation has our main performance metric, MRT. Each data point on the graph is a result of the average of 10 simulation runs. For completeness, we do show the confidence intervals on this graph but will not go forward. We observe from the graph that MRT seems to gradually increase linearly as CV increases. The variation in MRT also increases with CV, as can be seen in the anticipated increase in the confidence interval (95%). The reduced variation after CV = 4 makes sense as the CV levels off, so, too, would MRT variation. In experiments going forward, we will maintain $v = 90$ to keep CV ≈ 3.

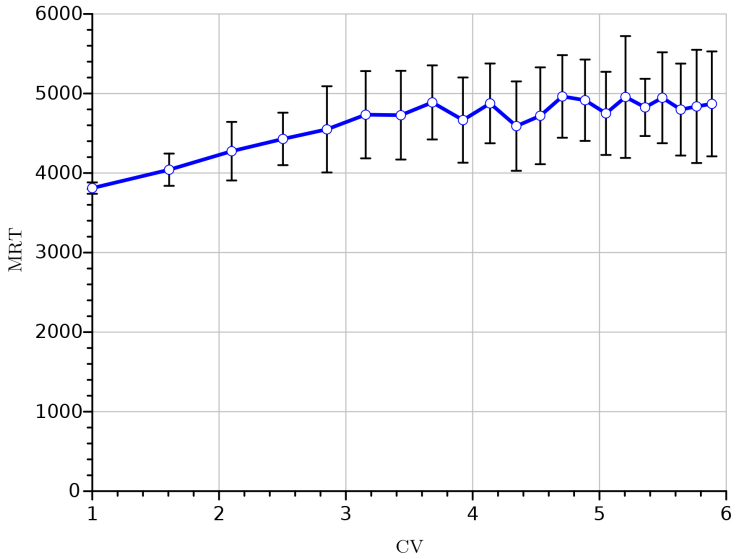

Figure 11. The effect of CV on MRT. $K = 5$, $M = 10{,}000$, $M_0 = 1000$, $\gamma_1 = \gamma_2 = 200{,}000$, $\rho \approx 87\%$ ($K = 5$, $N = 500$, $z = 35{,}000$).

4.6. Composition of the Web Page Set

The parameters that determine the composition of the Web page set, which we discussed in Section 3.4, will remain constant in our expanded server model. We will continue to consider the small object size ($\Theta_{small} = 1$) to be the base unit for all measures of data size in our model. In Table 5, we present the composition of the Web page object set by category that we use in our simulation. We have expanded the table to include Web Page Object Cacheability (φ).

Table 5. Composition of the Web page object set with Web caching.

Category	Object Type	Θ	Q	φ
Article	text	small	25	not
	script	small	20	less
	css	small	5	more
	image	small	25	more
	audio	small	5	more
	video	-	-	-
Mosaic	text	small	60	not
	script	medium	10	less
	css	small	5	more
	image	small	60	more
	audio	medium	2	more
	video	medium	1	more
Media	text	small	20	not
	script	small	15	less
	css	small	5	more
	image	medium	15	more
	audio	medium	5	more
	video	large	3	more

In Section 4.9, we will examine the effect of varying the ratios in the Web page categories when we consider three testing scenarios: *Base*, *Low*, and *High*. The corresponding values for testing scenario are found in Table 6. Notice how the values of mean Web page size ($\bar{W}$) correspond to the nomenclature used for the ratio testing scenarios (*Low* has the smallest $\bar{W}$ and *High* has the largest (*Base* is in the middle). For consistency, unless stated otherwise, we will use the Web Page Category Base Scenario with $C = 5$ when examining parameters in our expanded server model.

Table 6. Summary of final Web page category ratio test scenario.

Scenario	k_{article}	k_{mosaic}	k_{media}	$\bar{W}$
Low	70	10	20	216
Base	30	40	30	344
High	10	30	60	488

4.7. System Load

We know the effect of varying the number of users (N) and number of Web caches (C) has on the system load (ρ). The results are shown in Figure 12.

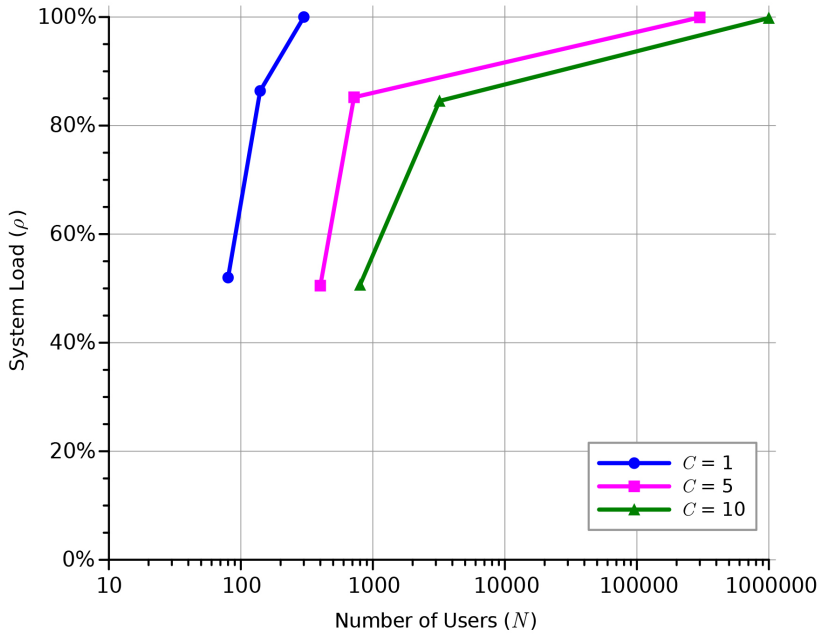

Figure 12. Effect of number of users on system load $\delta = 0.6$, $\kappa = 100\%$, $HR \approx 60$, $z = 35{,}000$, Web page category base scenario.

As one would expect, the results indicate that increasing the number of users leads to an increase in the load on the Web cache servers. Figure 12 indicates that the increase in system load tends to be mitigated through adding additional Web cache servers. A system with only one Web cache server gets overloaded ($\rho \approx 100$) quite quickly. Through having more servers ($C = 5$ or 10), the system can handle additional users but they too will be overloaded as the number of users gets very large. For the remainder of our experiments in this section, we will use $C = 5$ and $N = 720$ to aim for an approximate system load of 85%.

4.8. Cache Capacity

Next, we examine *Cache Capacity* (κ), which we will represent as a percentage of the maximum amount of cache storage available for objects. For the Web caching algorithm, the unit of storage will still be defined in terms of small object size ($\Theta_{small} = 1$): A large object (Θ_{large}) is 100 times larger than a small object, and a medium object (Θ_{large}) is 15 times larger. Once a cache is full, it cannot accept more objects. Cache capacity (κ) has a range of [0, 100]%, where 0% means that Web caching cannot occur, and 100% means that all objects that could be cached are cached.

Figure 13 shows the results of our experiment intended to investigate the effect that varying Cache Capacity (κ) has on *Hit Rate* (HR) and *Cache Size Hit Rate* (SHR). HR refers to the ratio of the number of Web page objects retrieved from a cache relative to the total number requested. On the other hand, SHR is the ratio of the amount of data retrieved from cache relative to the total size of the requested data.

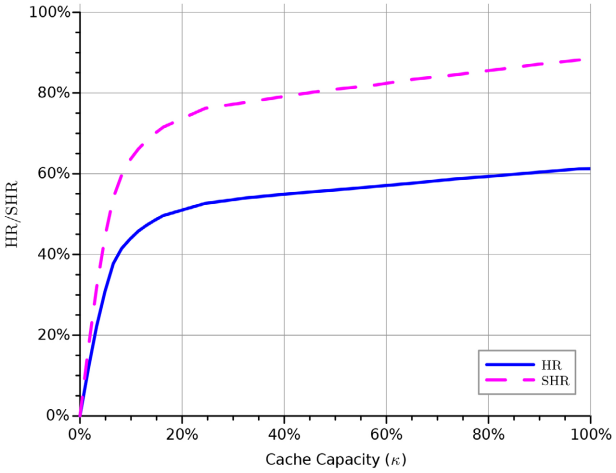

Figure 13. Effect of cache capacity on HR/SHR $\delta = 0.6$, $\rho \approx 85\%$, $C = 5$, $N = 720$, $z = 35{,}000$, Web page category base scenario.

In general, we can observe that the HR increases as cache capacity increases. This is reasonable since, as the cache capacity increases, more objects are cached and thus are more likely to be found in cache. Initially, for small values of κ (<10%), HR is low as objects are swapped out frequently, which makes it less likely for an object to be found in cache. HR does increase quickly as more objects are cached but starts to level after about 20% as fewer objects have to be removed, so the rate of change in HR slows down. Once κ reaches 100%, HR no longer changes, since all of the data that can be cached has been cached. For the Base ratio scenario, given that text files are not cacheable, a HR of about 62% is the theoretical maximum achievable.

As for SHR, the performance is similar to HR but has a higher upper bound. The difference is due to the fact that HR is a ratio of discrete values, whereas SHR compares quantities, and so the latter will have higher values.

In Figure 14, we examine the effects of cache capacity (κ) on system load (ρ).

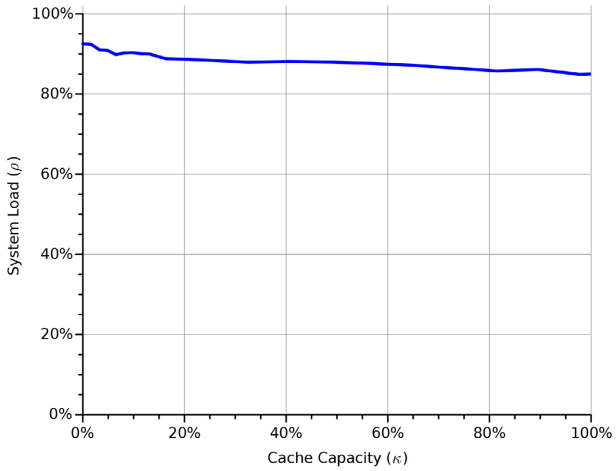

Figure 14. Effect of cache capacity on system load $\delta = 0.6$, $C = 5$, $N = 720$, $z = 35{,}000$, Web page category base scenario.

We observe that ρ decreases as κ increases. This makes sense, since for a constant number of users, a smaller κ will increase transmission times since fewer objects are cached and hence, more requests will be fulfilled at the original servers. When κ is at 100%, the system load corresponds to the results described in Section 4.7 for $C = 5$, where the system load was approximately 85% with $\kappa = 100\%$.

4.9. Ratio of Web Pages per Category

The results from Section 4.7 and Section 4.8 focused on the *Base* ratio scenario of the Web pages in a category. In this section, we consider the two other ratio scenarios: *Low* and *High*. The details of the ratio values and mean Web page sizes were presented in Table 6.

In addition to the ratio scenarios, we also consider four different cache capacities: *Small*, *Medium*, *Large*, and *No Caching*. The system without caching is used for reference. The values we chose for the three cache capacities (κ) correspond to the points in Figure 13 where the HR was approximately 20%, 40% and 60%. These correspond to cache capacities (κ) of about 3%, 7%, and 80%, respectively. The system with no caching would have a cache capacity of 0%.

4.9.1. Mean Response Time

In Figure 15, we plot the results for the three different Ratio Scenarios against MRT for the four different cache capacities (κ). In general, we observe that MRT increases as the W of the ratio scenarios increases. This is not surprising since W is proportional to request service time and is consistent with the results from [3]. For the Low scenario, there is less MRT variation between the four cache capacities. However, for the Base and High scenarios, the variation in MRT between cache capacities is more pronounced. We suspect that this is related to the larger SHR for the Base and High scenarios with respect to the Low scenario.

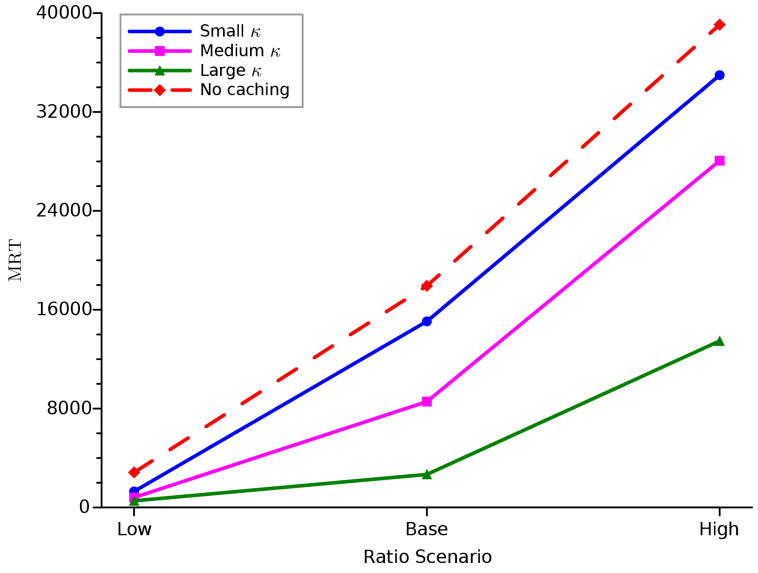

Figure 15. Effect of ratio scenarios on MRT $\delta = 0.6$, $C = 5$, $N = 720$, $z = 35{,}000$.

Another interesting observation from Figure 15 is that the MRT is improved through object-based Web caching, and the improvement tends to be more pronounced as the mean Web page size or cache capacity increases. It makes sense that as κ increases, MRT decreases since more cache size means more cache hits, thus reducing MRT. As well, smaller Web pages benefit less from changes in κ than larger ones. This is due to larger Web pages having to be removed more frequently than smaller ones for equivalent Cache Capacities. Finally, the system without Web caching seems to be the upper limit with respect to MRT: all of the systems that employ object-based Web caching tend to outperform the system without caching.

4.9.2. System Load

We now move on to examine the effects that the three ratio categories and four Web caching scenarios have on system load. The results are presented in Figure 16.

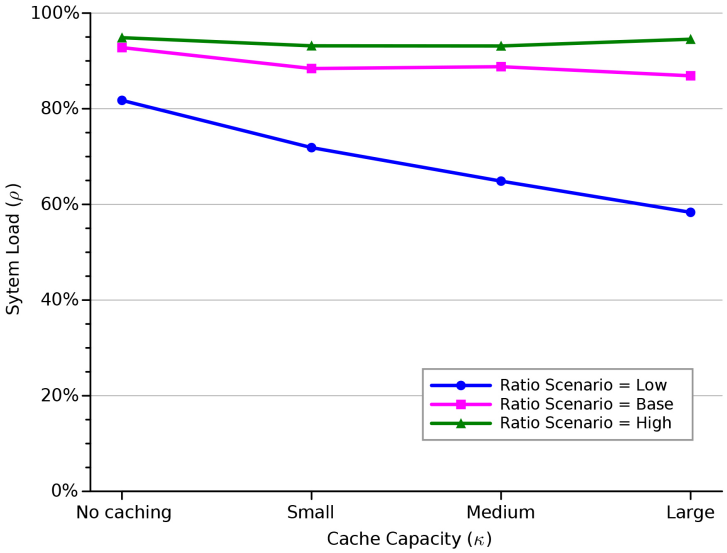

Figure 16. Effect of cache capacity on system load $\delta = 0.6$, $C = 5$, $N = 720$, $z = 35{,}000$.

The results indicate that as W of the ratio scenarios increases, ρ tends to also increase. This is a result of the fact that request service time would increase due to larger Web page size, and so system load would also increase as there would be more work at the Web servers.

Conversely, as the cache capacity increases, we see a decrease in system load with the Low Scenario and a slight decrease with the Low and Base scenarios (albeit, the decrease for the Low scenario is not as pronounced as with the Base scenario). This again is reasonable as smaller Web pages (with lower request service times) being cached more frequently will not generate as much work at the Web servers and hence, result in a lower system load.

There is little change for the High Scenario, possibly due to the system load being too large, so additional cache capacity is not as beneficial.

In summary, these results show that system load may be improved with Web caching, although with diminishing returns as the amount of data increases or the cache capacity decreases. That said, as we observed with MRT, object-based Web caching still leads to an improvement in system load over a system with no Web caching.

4.9.3. HR and SHR

Finally, in Figure 17, we show HR and SHR for the three chosen cache capacities (κ) and the three Web category ratio scenarios (the system with no caching would have a HR and SHR value of 0).

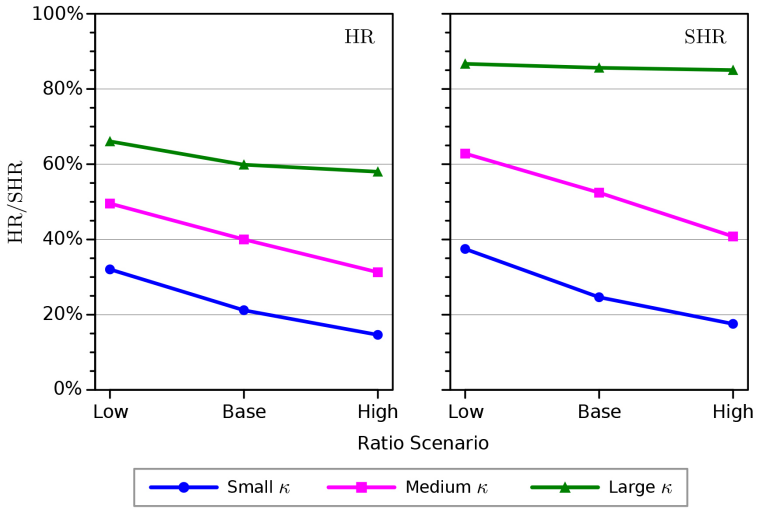

Figure 17. Effect of ratio scenarios on HR/SHR $\delta = 0.6$, $C = 5$, $N = 720$, $z = 35{,}000$.

In each case, HR and SHR tend to decrease with the larger W that results from the various ratio scenarios. This makes sense because, as W increases, the ability to fit objects into Web caches decreases as a result of the increased number of objects per Web page. We also observe that for the Base Scenario, the HR for the three κ values are approximately 20%, 40%, and 60%, which was our goal in choosing the values of κ.

Finally, Figure 17 shows that HR increases as κ increases. This is not surprising since, as the cache capacity increases, more objects are cached and thus more objects are found in cache (as was discussed in Section 4.8). This also corresponds to the decrease in MRT observed in Figure 15, which is again expected since an increase in HR would lead to a smaller Web page transmission time (τ), resulting in a decrease in MRT.

5. Summary

In this paper, we incorporated object-based caching system into a detailed User Request Access Pattern Model. We accomplished this by replacing the Web server sub-model, presented in Section 3.3, with a homogeneous set of Web caches (discussed in Section 4.1). When objects are cached (based on a cacheability factor

parameter), the service time (and hence transmission time) for user requests will be decreased. When requests are made for objects that are not cached (again, based on the cacheability factor parameter), the service and transmission times for the objects will be larger. In the event that there is not enough room for an object to be added to a cache, a LRU cache replacement policy is applied to remove objects from cache to make room.

We used a discrete-event simulation of the User Request Access Pattern Model that includes Web caching to evaluate the performance as we varied several model parameters. We found that as the cacheability of objects increases, there tends to be a reduction in both MRT and system load. When examining cache capacity, our results indicated that increasing the cache capacity seemed to lead to an increase in HR and SHR, and a decrease in MRT.

Our final set of experiments varied the ratio of Web pages per Web page category in a Web caching environment. We evaluated four Cache Capacity scenarios while varying our Web page category ratio test scenarios (No Web caching, Base, Low, High). The results seemed to demonstrate that the amount of data being requested (represented by ratios of Web pages per category) and cache capacity both had a positive effect on HR. Overall, we found that object-based Web caching tends to improve both MRT and server load.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Freed, N. and Kucherawy, M. (2017) Media Types. https://www.iana.org/assignments/media-types/media-types.xhtml

[2] Hurley, R. and Sturgeon, R. (2024) Framework to Model User Request Access Patterns in the World Wide Web. *Journal of Software Engineering and Applications*, **17**, 69-88. https://doi.org/10.4236/jsea.2024.172004

[3] Sturgeon, R. (2022) Modelling Request Access Patterns for Information on the World Wide Web. Master's Thesis, Trent University.

[4] Ali, W., Shamsuddin, S.M. and Ismail, A.S. (2011) A Survey of Web Caching and Prefetching. *International Journal of Advances in Soft Computing and its Applications*, **3**, 18-44. https://www.researchgate.net/publication/265986051_A_Survey_of_Web_Caching_and_Prefetching_A_Survey_of_Web_Caching_and_Prefetching

[5] Hurley, R. and Plumley, B. (2018) Comparison of Sender and Receiver-Initiated Load Balancing in a Distributed Web Caching System. *Proceedings of the* 33*rd International Conference on Computers and Their Applications* (*CATA*2018), Las Vegas, 19-21 March 2018, 45-50.

[6] Chen, J. and Cheng, W. (2016) Analysis of Web Traffic Based on HTTP Protocol. 2016 24*th International Conference on Software, Telecommunications and Computer Networks* (*SoftCOM*), Split, 22-24 September 2016, 1-5. https://doi.org/10.1109/softcom.2016.7772120

[7] Ihm, S. (2011) Understanding and Improving Modern Web Traffic Caching. Ph.D. The-

sis, Princeton University.

[8] Newton, B., Jeffay, K. and Aikat, J. (2013) The Continued Evolution of Web Traffic. 2013 *IEEE* 21*st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems*, San Francisco, 14-16 August 2013, 80-89. https://doi.org/10.1109/mascots.2013.16

[9] Shi, Z. and Fan, Z. (2025) Optimized Caching Strategy: A Hybrid of Least Recently Used and Least Frequently Used Methods. *Proceedings of the* 2025 5*th International Conference on Computer Network Security and Software Engineering*, Qingdao, 21-23 February 2025, 133-140. https://doi.org/10.1145/3732365.3732389

[10] Berger, D.S., Beckmann, N. and Harchol-Balter, M. (2018) Practical Bounds on Optimal Caching with Variable Object Sizes. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, **2**, 1-38. https://doi.org/10.1145/3224427

[11] Wang, J. (1999) A Survey of Web Caching Schemes for the Internet. *ACM SIGCOMM Computer Communication Review*, **29**, 36-46. https://doi.org/10.1145/505696.505701

[12] Evans, D. (2011) The Internet of Things: How the Next Evolution of the Internet Is Changing Everything. Cisco IBSG.

[13] Allison, C., Bramley, M. and Serrano, J. (1998) The World Wide Wait: Where Does the Time Go? *Proceedings.* 24*th EUROMICRO Conference*, Vasteras, 27-27 August 1998, 932-938. https://doi.org/10.1109/eurmic.1998.708124

[14] Hurley, R.T. and Li, B.Y. (2008) Effects of Dynamic Content on Web Caching. *Proceedings of the ISCA* 21*st International Conference on Parallel and Distributed Computing and Communication Systems* (*PDCCS*'08). New Orleans, 24-26 September 2008, 165-170.

[15] Gangemi, A. and Presutti, V. (2006) Towards an OWL Ontology for Identity on the Web. *Proceedings of the* 3*rd Italian Semantic Web Workshop*, **201**, 1-8. https://ceur-ws.org/Vol-201/43.pdf

[16] Tanenbaum, A.S. and van Steen, M. (2006) Distributed Systems: Principles and Paradigms. 2nd Edition, Prentice-Hall, Inc.

[17] Raza, A., Zaki, Y., Pötsch, T., Chen, J. and Subramanian, L. (2015) Extreme Web Caching for Faster Web Browsing. *Proceedings of the* 2015 *ACM Conference on Special Interest Group on Data Communication*, New York, 17-21 August 2015, 111-112. https://doi.org/10.1145/2785956.2790032

[18] Stallings, W. (2007) Data and Computer Communications. 8th Edition, Pearson Prentice Hall.

[19] Banks, J., *et al.* (2005) Discrete-Event System Simulation. 4th Edition, Pearson Education Inc.

[20] Hurley, R. and Plumley, B. (2016) Effectiveness of Load Balancing in a Distributed Web Caching System. *Proceedings of the* 7*th International Conference on Computer Modelling* (*ICCM*2016). Berkely, 1-4 August 2016, 46-60.

[21] Gonzalez-Canete, F.J., Casilari, E. and Trivino-Cabrera, A. (2007) Characterizing Document Types to Evaluate Web Cache Replacement Policies. *Fourth European Conference on Universal Multiservice Networks* (*ECUMN*'07), Toulouse, 14-16 February 2007, 3-11. https://doi.org/10.1109/ecumn.2007.11

[22] WebSiteOptimization.com (2012) Average Number of Web Page Objects Breaks 100. http://www.websiteoptimization.com/speed/tweak/average-number-web-objects/