Modelling Request Access Patterns for Information on the World Wide Web

A Thesis Submitted to the Committee on Graduate Studies
in Partial Fulfillment of the Requirements for the
Degree of Master of Science
in the Faculty of Arts and Science

TRENT UNIVERSITY
Peterborough, Ontario, Canada

Applied Modelling and Quantitative Methods
M.Sc. Graduate Program

September 2022

Abstract

Modelling Request Access Patterns for Information on the World Wide Web

Robert C. Sturgeon

In this thesis, we present a framework to model user object-level request patterns in the World Wide Web. This framework consists of three sub-models: one for file access, one for Web pages, and one for storage sites. Web Pages are modelled to be made up of different types and sizes of objects, which are characterized by way of categories.

We developed a discrete event simulation to investigate the performance of systems that utilize our model. Using this simulation, we established parameters that produce a wide range of conditions that serve as a basis for generating a variety of user request patterns. We demonstrated that with our framework, we can affect the mean response time (our performance metric of choice) by varying the composition of Web pages using our categories. To further test our framework, it was applied to a Web caching system, for which our results showed improved mean response time and server load.

Keywords: performance modelling, World Wide Web, Internet, Web caching, discrete event simulation (DES)

Acknowledgments

Developing this thesis has been a great learning experience and a fantastic opportunity for personal growth. Foremost, I am extremely grateful to my supervisor, Dr Richard Hurley, for the many years of commitment and patience (and slogging through the many drafts). I would also like to express my deepest appreciation to my committee, Dr Robson De Grande, Dr James Parker, and Dr Sabine McConnell, for their encouragement and feedback. I would like to recognize staff and faculty for giving me the background and encouragement over my years at Trent University, namely: Dr Brian Patrick, Brian Hircock, Bonnie McKinnon, and Nancy Smith, to name a few. Also, I would be remiss to forget my Trent friends that I have made over the years, especially: Kevin and Bethany (my officemates); and, Callum, Graham, and Greg (Team ln).

I cannot express my sincere gratitude enough to my family and friends for your continued support. I am especially grateful to my wife, Emily, and my children James, Claire, and Meg; as well as my parents, Lois and Carl, and in-laws, Hilary and John. Your love and support kept me going for this long adventure.

Emily, thank for your unending support, understanding, patience, proof-reading and love.

Contents

List of Figures

List of Tables

List of Symbols

C	Number of Web caches
CV	Coefficient of variation of Web page request interarrival time
HR	Cache hit rate
i	Web page identifier
j	Web page object identifier
k	Ratio of Web pages per Web page category
MRT	mean response time
K	Number of servers
M	Number of Web pages
M_0	Number of potentially popular Web pages
$M_\mathrm{p}(t)$	Number of Web pages currently popular
N	Number of users
$p_i(t)$	Probability of selecting Web page
Q	Number of Web page objects
R	Request service time
SHR	Cache size hit rate
t	Time
v	Popularity factor
W	Web page size
γ_1	Normal-popular transition rate
γ_2	Popular-normal transition rate
δ	Cached object access factor
Θ	Web page object size
Θ_small	Small object size (base unit of data size)
κ	Cache capacity
λ	Request rate
μ	Mean service rate
ρ	System load
φ	Web page object cacheability
φ_not	Not cacheable
φ_less	Less cacheable
φ_more	More cacheable
τ	Web page transmission time

Chapter 1

Introduction

Waiting for a Web page to load can be one of modern life's most frustrating occasions. The *World Wide Web* (WWW) is a system that has only recently — in the past few decades — become ubiquitous to modern society as a fast and effective way to connect people with an enormous amount of information. However, the size and complexity of the WWW makes it a challenging study in terms of performance and user experience.

In this thesis, we present a framework that models *Hypertext Transfer Protocol* (HTTP) request patterns in the WWW which then allows us to compare the relative performance of various architectures, and the impact these have on user experience. As a framework, our model can be adapted to a wide range of conditions that can be used by researchers as a tool with their own models.

An important feature of our framework is how information is represented. This is accomplished with a *Web Page Model*, which determines the composition of Web pages. Each Web page is modelled to consist of a unique set of objects, where each object has a type and a size. The composition of Web pages is determined by the *Web page categories*, which in turn determines the number of objects as well as their type and size. In this thesis, we provide an illustrative set of categories which represent an anecdotal representation of Web pages as they may appear to the user. However, the configuration of Web page categories is a model parameter that is meant to be modified according to the research requirements.

We use computer simulation to implement and test our model using a set of basic parameters, and then utilize our model in an investigation of Web caching. The main goal is to provide an unique framework to compare the performance of various scenarios for user request patterns in the WWW. This is accomplished by modelling the process of a user requesting and subsequently receiving a Web page at the object-level within the WWW.

The remainder of this chapter is divided into the following parts: the general environment, a brief discussion of Web caching, and an outline of the Thesis. The discussion of the general environment includes a high-level description of the system, which is then followed by a more detailed description of the system components and processes involved. Next we discuss Web caching, the application used in this research to test the feasibility of the model. Finally, we discuss then remaining organization and goals of the thesis.

1.1 General Environment

At a broad level, 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 an agent that is requesting information — human or machine [18]. The *Internet* is the communications media that interconnects the users and Web servers. *Web servers* are the devices that store the information and transmit it to the user. Finally, the information is the component found generally in the form of Web pages.

Figure 1.1 is a simplified representation of the WWW that outlines a user making

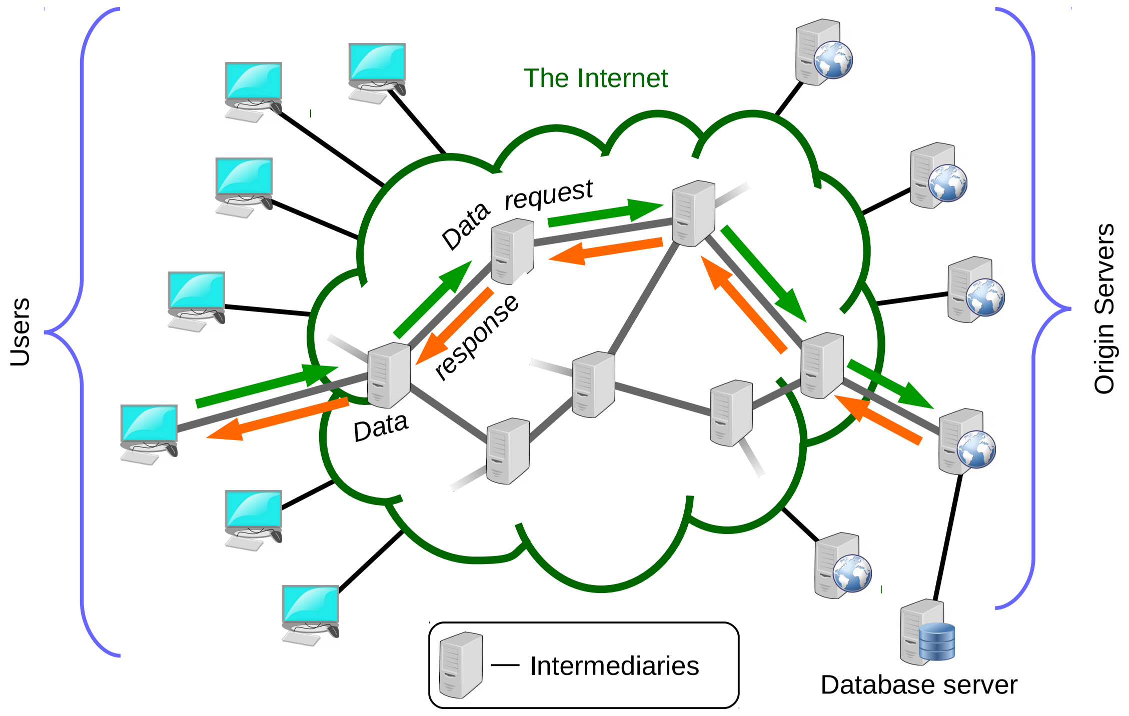


Figure 1.1: Simplified World Wide Web Showing an Example Request/Response Chain.

a request to a Web server for a Web page. The path that is takes is through the Internet and is routed through multiple intermediaries. Let us consider the details of the main concepts just presented and the process of these components interacting to provide information. The detailed discussion of these two concepts that follow includes: the user, the information, the network, and the Web server.

Although the main goal of the WWW is to connect users to information, nearly half of the traffic on the WWW is due to automated processes [26]. These machine actors are used for various activities, much of which include information indexing for search engines and analytic algorithms.

A user requires the ability to interface with the WWW, an action which is achieved through various types of software [17,19]. In the case of a human user, this software is referred to as a *Web browser*, but for some automated users, it would be called a *spider*. We will use the term *browser* for the software that is used to interact with the WWW [19]. Generally the job of this software is to initiate requests, negotiate network access, and render the data received into a format that can be utilized by the user.

The information available within the WWW varies widely in its amount, size, composition, and format. Some common examples of information types are text, audio, video, and images. Also, these types of information exist in several technological formats, such as plain text or binary information, and may come from a multitude of information sources. Information is accessed by determining its location on the WWW using a *Uniform Resource Identifier* (URI) (for example `www.trentu.ca`). URI are logical addresses for discrete documents, referred to as *Web pages.* While the URI is a logical address, its resolution is beyond the scope of our work [27].

A Web page is a unique document that is composed of various types of information. Web pages are stored on a Web server, to which the user issues requests. A Web server responds to a user's request for a Web page by retrieving the Web page from its storage and delivering it through the Internet. Interconnection between Web pages is a fundamental property of the WWW, and is accomplished through hyperlinks within Web pages to other Web pages [10].

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. In the WWW, a Web page object is an element of data of arbitrary content, whose type is structured and named according to an international registration organization (the *Internet Assigned Numbers Authority* (IANA)) [20]. Web page objects have several properties, which include size and encoding (such as ASCII text or binary data), but for our model 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 [20,26]:

***hyper-text markup language* (HTML):** A language that is used as the basis for encoding information in Web pages. HTML provides the method to build the Web page structure, and includes elements such as: text information, references to other Web page content (for example an image), and links other Web pages.

Javascript: A programming language that is used for a variety of purposes, but in general it facilitates user interaction with a Web page.

***Extensible Markup Language* (XML):** A markup language standard that is used to encode information in many WWW services.

***Cascading Style Sheets* (CSS):** A language used to control the display layout of Web page object.

image: Binary representation of a picture.

audio: Sound information.

video: Visual and sound information.

octet: Information element that may contain any arbitrary information encoded in binary. Examples include applications and PDF documents.

An important consideration regarding Web pages is that their content can change with time (referred to as *dynamic content*) [2,24]. Although the specific content on a Web page may change, it still represents a single resource [35]. For example, a news main page such as `www.cbc.ca/news` 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. Regardless of the news articles available, `www.cbc.ca/news` is still viewed as a single Web page.

Another component within the WWW is the *Web server*, which has an important impact on the experience of the user. Web servers, which range from a single computer to a collection of networked computers, hold the Web pages that users request [39]. The speed at which they can respond to the request contributes in large part to the response time.

Fundamentally, a Web server consists of the following components: an interface to the Internet, processors for requests, storage, and software [38]. There are several ways that these components can be organized within Web servers. In the case of a single computer, all of the components are connected within one machine, where the storage is a locally accessed file system. Typically, single computers are used for small personal Web servers. For larger systems, such as for the purposes of education, government and commerce, a Web server is made up of a collection of multiple machines, with each one performing a specific function. In such a system there is some combination of dedicated machines to interface with the Internet, process requests, and manage information. Information in large systems is stored and managed using a *database management system* (DBMS), which may be a system within the local Web server facility or be a remote system [38].

The WWW would not be as ingrained in society were not for the underlying networking technology that now spans most of the world (ie, the Internet). The Internet is a collection of protocols for communicating across arbitrary packet-switched networks, and is made up of three main components hosts, routers, and networks [38]. The WWW is an application-layer process in the OSI network model that runs on the Internet [38].

1.2 Web Caching

Web caching is an important technique utilized by the Internet and can have a great deal of impact on performance and other factors. In general, Web caching involves the temporary storage of frequently accessed Web page objects at various intermediary storage locations between a user and a Web server. By storing Web page objects geographically closer to a user, access latencies can be improved [36]. Some important potential benefits of Web caching include: improved request response times [40], reduced load on Web servers [36], reduction of network bandwidth [40], data redundancy if the Web server is down through availability of cached data [40], transparent to the user [40], and demand based (based on the popularity of the Web page) so no administration is required to decide what to cache [34].

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. A simplified example of Web caching shown in Figure 1.2, illustrates an intermediary storage location working as a Web cache to potentially reduce the time to service a request.

Not all information can be stored in an intermediary storage location because cache storage size is small relative to the amount of data available in the WWW. As well, certain types of information change rapidly thus making caching impracticable. Therefore, various Web caching topologies and approaches for handling dynamic information have been developed to improve Web page response times [40]. While the implementation of our model considers only *distributed* Web caching, three Web caching topologies that determine the geographical configuration of Web cache site are: *hierarchical, distributed, and hybrid.*

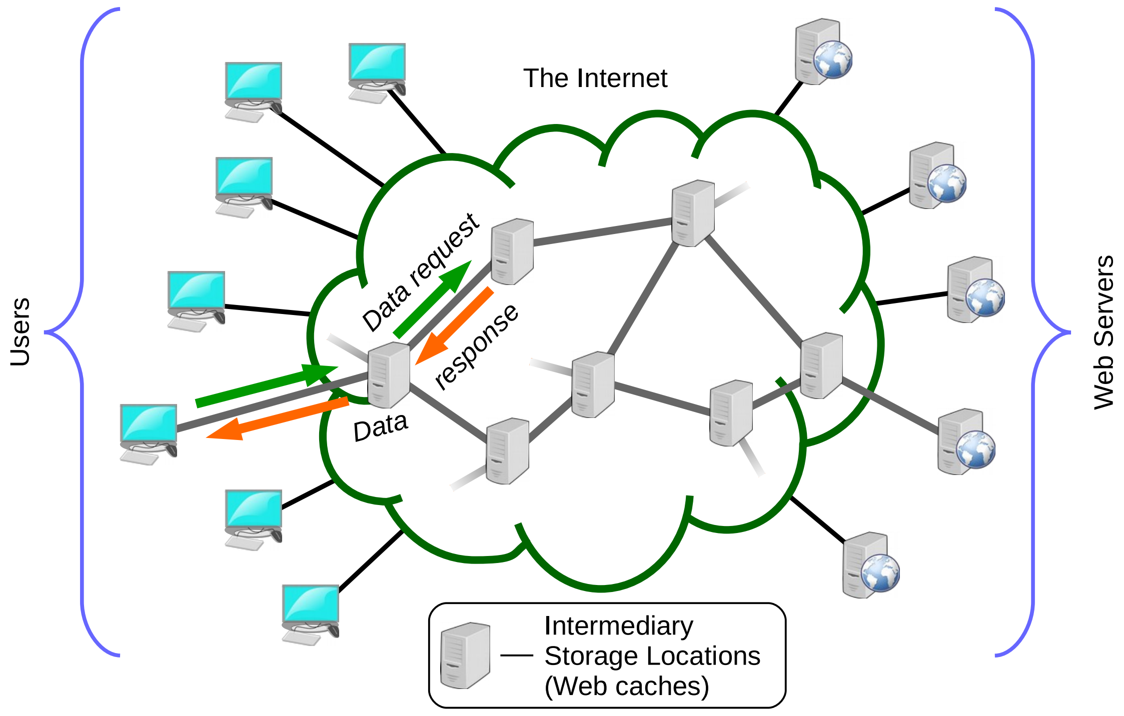


Figure 1.2: Example of a Web Page Request Receiving a Cached Response.

1.3 Thesis Outline

The reminder of the thesis consists of five additional chapters. Chapter 2 (*Background*) discusses the background work as it relates to our thesis. This includes a brief history and overview of the WWW, including Web caching. This is then followed by a literature review of modelling the WWW.

In Chapter 3 (*Model*) we present the Web Page Request Access Pattern Model. The overall model is described in the *User-Web Interaction Model*, which is made up of the *User-Request Model*, the *Server Model*, and the *Web Page Model*; all of which are discussed in detail.

Chapter 4 (*Implementation and Results*) discusses our implementation of the Web Page Request Access Pattern Model using *discrete event simulation* (DES). This is followed with verifying our implementation by comparing simulated results to analytic ones. We complete this chapter by presenting several scenarios that explore a range of input into our model, without considering Web caching.

Our model is extended in Chapter 5 (*Web Caching Model and Results*) with an expanded server model (Web caching). Continuing from Chapter 4, we add the caching model to our implementation. Chapter 5 then compares our results between the base model and the expanded server model.

We end the thesis by summarizing the results of the work and discussing the their contributions in Chapter 6 (*Conclusions and Future Research*). During the course of developing this work, many interesting ideas came up that would enhance or be an exciting application of it. However, to keep the scope reasonable many of these had to be set aside. In Section 6.2 (*Suggestions for Future Research*) we explore of some of these ideas.

Chapter 2

Background

2.1 Introduction

In this thesis, there are two topics that on their own represent substantial bodies of work in the field of Computer Science: *modelling* and the *World Wide Web*. The intersection of these two topics results in a significant amount of research available.

In this chapter, we summarize the important aspects of modelling and the WWW as they relate to our thesis. In particular, we consider the factors that affect our main performance measure of interest, *mean response time* (MRT). In the next section we discuss the WWW, including its history, an overview of its function, and Web caching. Following our discussion of the WWW we then discuss modelling it in Section 2.3.

2.2 The World Wide Web

The WWW is a system to organize various forms of information and present it to users. Information is stored and retrieved in this system across a world-wide heterogeneous set of networked computers. The user interface and the data sources are independent of one another and are interconnected through a well defined interface.

While the WWW was conceived only about thirty years ago, it has several historical influences [9].

One of the first mentions of interconnected data retrieval systems was presented in 1945 [12]. This conceptual system, the *Memex*, represented the concept of a data storage device that links, indexes, and searches books, records, and communications, and other data. The *Memex* was conceived to be mainly mechanical using microfilm and other electro-mechanical devices of the time, and was not specifically implemented.

The WWW in its current form was conceived by Sir Tim Berners-Lee in 1989 [9]. Berners-Lee identified the need to manage large amounts of information while working with *European Organization for Nuclear Research* (CERN) and the *Large Hadron Collider* (LHC). While the needs were motivated by his experience at CERN, Berners-Lee envisioned that it would be global [10].

At the same time that Berners-Lee's concept was gaining traction, the focus of hypertext systems being used was in the user interfaces of a common application, rather than a wide-area heterogeneous system [10]. Figure 2.1 shows Berners-Lee et al.'s original concept of the WWW. Here the client devices such as personal computers (PCs), Mac computers, and other client devices are connected to data storage devices, such as servers and external databases. Clients are connected to data storage through a large scale network (the Internet) which communicate by way of common addressing and protocols. Some devices which do not inherently communicate with the common protocol can still be access on the WWW using a gateway (examples of which are shown in Figure 2.1).

An important aspect of the WWW is *hypertext*, a term coined by Ted Nelson who had been researching and developing hypertext systems since the 1960s [9,17,33]. The concept of hypertext views data as non-linear nodes related to one another through links. Some examples of hypertext nodes Berners-Lee envisioned for the WWW are

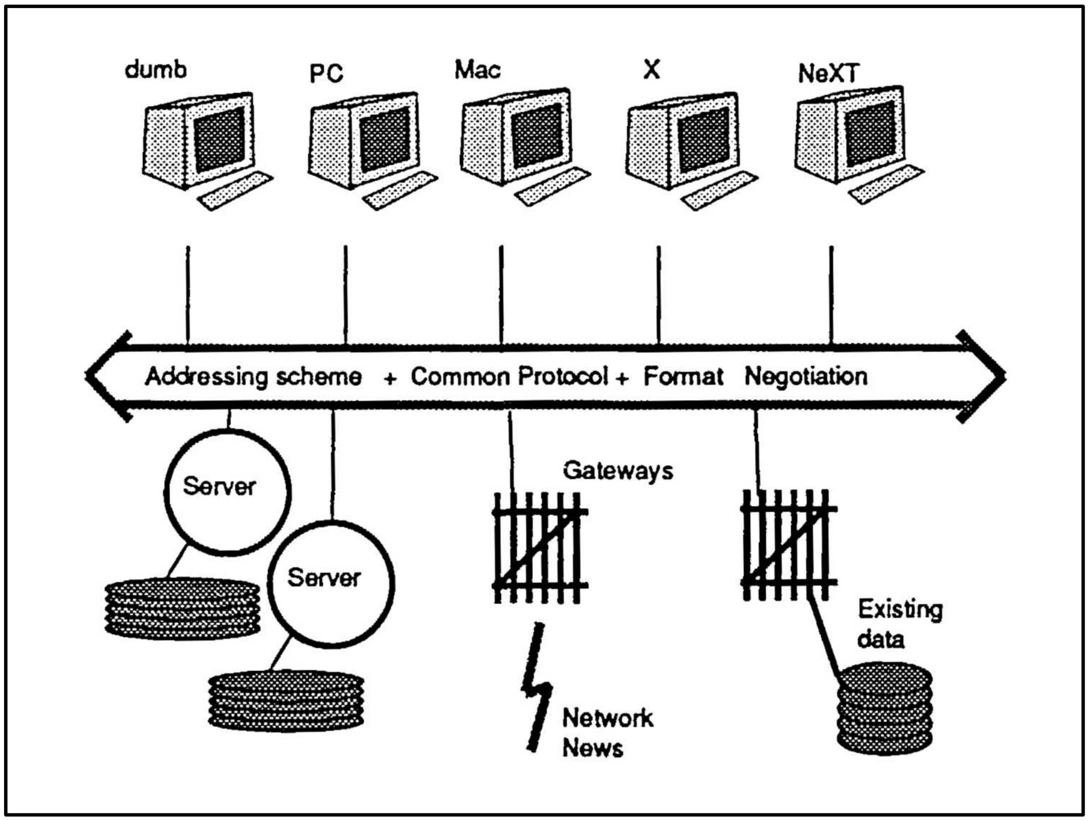


Figure 2.1: Berners-Lee's *The W^3 architecture in outline* [10].

people, projects, and documents [9]. *Hypermedia* expands the concept of hypertext with non-text content, such as graphics, speech and video [9]; Conklin goes even further to potentially include "tastes, odors, and tactile sensations" [17].

In addition, Conklin described four categories of hypertext: macro literary systems, problem exploration tools, browsing systems, general hypertext technology [17]. The concepts these categories encompass support large online libraries, problem solving, reference, and general purpose systems used for reading, writing, collaboration, and more. Hypertext, as implemented in the WWW, addresses all of Conklin's categories, and are addressed in our model.

We now provide a brief introduction to HTTP for two reasons. First, while our model does not directly consider HTTP, it is a fundamental component of the WWW and as such, has some influence over the nature of our model. Second, it is the basis for examining other works that have modeled the WWW

HTTP is an application level protocol for transferring a variety of resources (Hypermedia) on the WWW [19]. In its most basic form, it facilitates a network connection between a *user agent* and an *origin server* to communicate requests and responses between one another. This request and response process is referred to as a *request/response chain*, and consists of request and response methods, status, destination addresses, and other parameters and data. The request/response chain typically consists of intermediaries, which include proxies, gateways, and tunnels. Each intermediary can modify messages, and perform various actions. One notable action for intermediaries is to provide cached responses to requests, instead of relaying the request further along the request chain.

An important aspect of HTTP is how it establishes and regulates network connections. Depending on the version, connections are either opened and closed for each resource request, or can be persistent for the entire request chain. Newer versions improve the protocol transport requirements, for example, by reducing repetitive and verbose protocol metadata [8]. As well, as HTTP has evolved, concurrent request pipelining has improved.

2.2.1 Web Caching

An important technology that can affect performance is *Web caching*. The goal of caching is to alleviate network congestion and server overload by distributing Web page content over multiple locations in the WWW [1,23,36,40]. This is accomplished by storing Web page objects that are likely to be used closer to the user. Some key advantages of Web caching are: reduced Web page request-response times, server workload reduction, improved Web server availability, and reduced network bandwidth [1,36].

There are several approaches to Web caching, but in general they can be summarized in three types: *browser cache*, *proxy cache*, and *origin server cache* [1]. As proxy Web caching plays a significant role in decreasing the request response time and network bandwidth reduction [1], our work focuses on this approach. There are many considerations in Web caching, such as which content to keep that is most likely to be reused effectively and how to efficiently organize the storage. Three typical metrics used when discussing Web caching are:

Response time: Time from the user initiating the Web page request to its completion [1].

Hit rate: Percentage of requests that are served from previously cached documents [1,36].

Byte hit rate: Size of the data that is retrieved from the Web cache with respect to the size of the data requested [1].

The storage size of a Web cache is limited so deciding which objects to keep is contentious; ideally, it is popular content that will most likely be reused [40]. Web cache replacement policies are used to remove data to make room for new data. Wang [40] describes three main types of caching algorithms: *traditional*, *key-based*, and *cost-based*. Traditional algorithms include *least recently used* (LRU) and *least frequently used* (LFU). LRU and LFU are relatively simple and efficient, however they tend not to account for download latency and object size [1]. As well, LFU can also suffer from storing obsolete objects indefinitely.

There are also several algorithms associated with key-based caching, which include: *Size*, *LRU-MIN*, *LRU-Threshold*, *Hyper-G*, and *Lowest Latency First* [40]. These algorithms remove objects based on a primary key. For example, in the case of Lowest Latency First, objects are assigned a key based on their download latency, and objects are evicted based on the object with the key representing the lowest download latency.

Cost-based algorithms use a cost function to determine which objects are evicted.

These cost functions often use time as a measure of cost, but can also include other metrics (or combination of metrics) such as cost-to-fetch, size, transfer time, and cost-to-size ratio to name a few. Some examples of cost-based algorithms include: *GreedyDual-Size* (GD-Size), *Least Normalized Cost Replacement* (LCN-R), *Size-Adjusted LRU* (SLRU), and *Server-assisted* scheme [40].

Chankhunthod et al. mentions that there are other considerations in which objects are not cached [15]. One example is that password protected objects are not cached. In addition, objects may be rejected based on metadata such as their URI, *time to live* (TTL), and size (this is in contrast to the *Size* Web caching algorithm, as objects may not even make it into the Web cache if they exceed a threshold size).

The size and topology of the Internet present significant challenges in physically building a caching system where users and the data are widely dispersed. A Web caching architecture strives to locate cache servers so that their placement is most effective. Three common approaches to organize Web caches are as follows:

Hierarchical: In this system, Web caches are organized in a multilevel hierarchy, with the users at the bottom. The levels of the hierarchy can be simplified as bottom, institutional, regional, and national. A Web cache request is referred up from lower levels to the higher ones until it is satisfied. If the highest level is a miss, then the request is sent to the origin server. Once the request is satisfied, the result propagates back to the user down the hierarchy, with copies of the data being left at each cache server in the request chain. A substantial disadvantage of hierarchical caching is that the bandwidth and storage requirements increase substantially toward the upper levels. Therefore, as the usage of the Web cache system increases, the upper level nodes become a bottleneck [15,34,40].

Figure 2.2 presents an example scenario to illustrate an hierarchical Web caching environment. Two users, Alice and Bob, request the same Web page from different geographic locations, and thus different points in the hierarchy. Alice is first to make the request and receives a response. Bob follows Alice's request, but is able to avoid the call to the origin server and obtains the data from a Web cache, thus reducing the overall time to satisfy the request.

Distributed: The distributed Web cache system avoids the chain of requests through multiple levels by making the institutional nodes solely responsible for data storage. The metadata contents of each Web cache is shared amongst all Web caches in the system in order to refer requests to one another. If a Web cache is unable to fulfill a request from its own storage, the Web cache seeks it from another Web cache according to the metadata it has on the content of its peers. In the event of a miss, the Web cache will request the data from an origin server. Although a distributed caching system avoids the bottleneck of the hierarchical system and allows better load sharing, two main problems are higher connection times and higher bandwidth usage [34,36,40].

Using the same caching scenario presented in Figure 2.2, Figure 2.3 shows it using a distributed caching topology.

Hybrid: Hybrid caching consists of a hierarchical organization of cooperating distributed Web caches. When a request cannot be satisfied by the distributed cooperating Web caches at a level, it is referred to the next level. The number of cooperating Web caches at a particular level is important so as to minimize retreival latency [36,40].

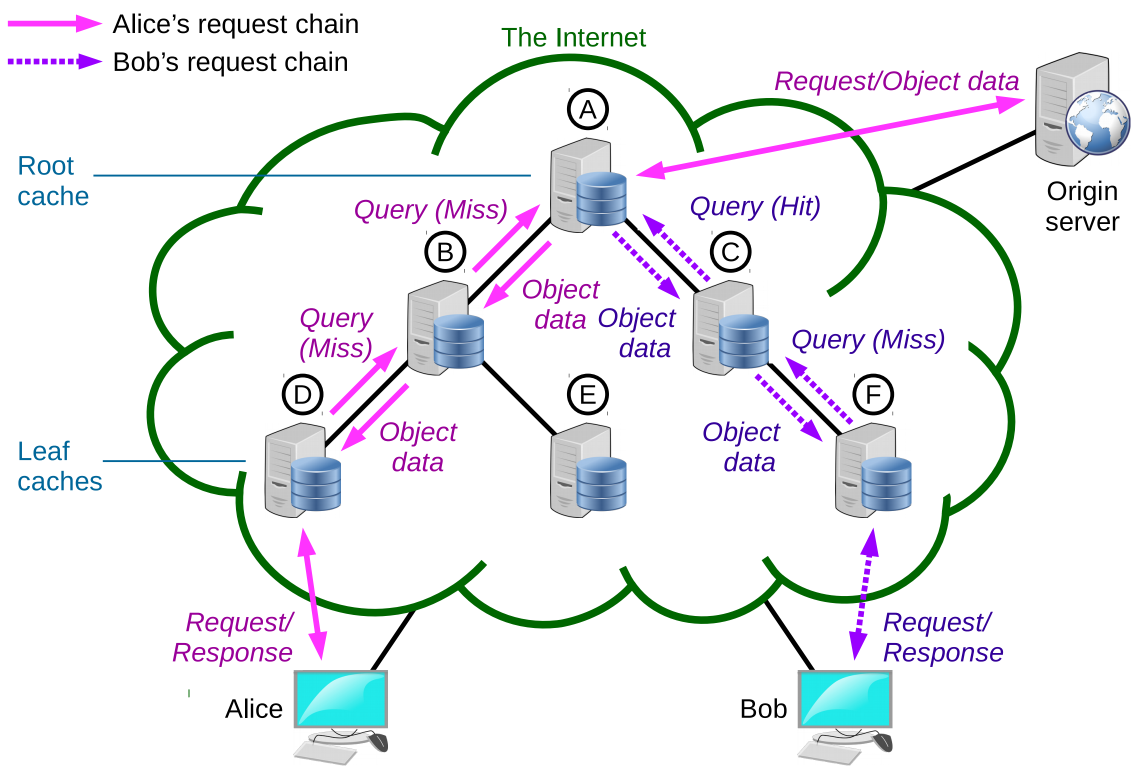


Figure 2.2: Hierarchical Web Caching Example.

1. Alice makes a request for a Web page.
2. The request is directed to the institutional Web cache at node D.
3. Node D does not contain data to fulfill the request, so it queries its parent, node B.
4. Node B also cannot fulfill the request so it is referred to node A.
5. Node A also does not have the requested data, and since node A is the root Web cache the request must be referred to the Web server.
6. Node A receives the object data, stores it, and passes a copy to node B. This is propagated back down the chain until Alice's receives the Web page. Now nodes A, B, and D have copies of this Web page data.
7. Next, Bob requests the same Web page as Alice.
8. His request is directed to the institutional Web cache at node F.
9. The request is referred from node F up the hierarchy to node A, where the copy is found from Alice's request.
10. The object data is passed back down the request chain, storing copies at each node along the way, and Bob's request is complete.
11. At this point any subsequent requests for the same Web page can be fulfilled by all nodes in our hierarchy except node E. Any request from a user connected to node E would require node E to obtain the data from node B.

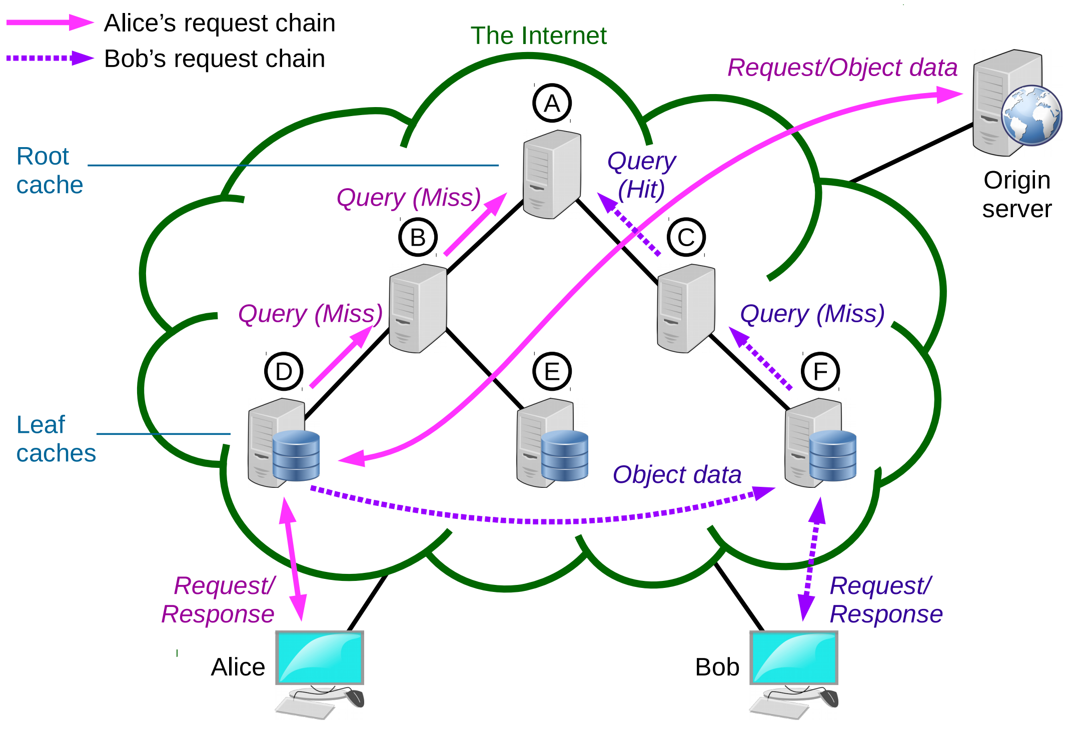


Figure 2.3: Distributed Caching Example.

1. Alice makes a request for a Web page.
2. The request is directed to the institutional Web cache at node D.
3. As was illustrated in Figure 2.2 the request moves up the network, unfulfilled, until it reaches the root Web cache, node A.
4. Instead of referring the request to the Web server, however, a miss notice is sent back down the network until it arrives at node D.
5. Node D requests the Web page data from the Web server, stores a copy of it, and completes the request to Alice.
6. An 'Advertisment' is now propagated from node D up to node A, notifying all of the nodes in the branch that node D has a copy of the data for the Web page.
7. Next, Bob requests the same Web page as Alice.
8. His request is directed to the institutional Web cache at node F.
9. The request is referred from node F up the hierarchy to node A, where a reference is found from Alice's request.
10. A notice of a request hit, and the location of the data, propagates back down the branch to node F.
11. Node F negotiates directly with its sibling, node D, and obtains the data to complete Bob's request.
12. At this point nodes D and F possess copies of the Web page, and nodes A, B, and C have references to it. If node E gets a request for the Web page, then it would find out from node B that the data can be found at node D, from which it can get the data.

Aside from the expense of a Web caching system, there are several disadvantages of caching:

- While it has shown improved request times for popular documents, less popular documents can result in a "long circuitous path of numerous failed checks" [40].
- In some cases, stale data may be provided to the user [40].
- Individual proxy nodes, in particular in the higher levels in a hierarchical system, represent a bottleneck as well as a single point of failure [40].
- Web cache misses may increase access latency [36].
- By avoiding requests directly to the origin server, caching makes it more difficult for content providers to maintain metrics on there pages [40].

Cáceres et al. discuss that other factors not accounted for may oversimplify the situation [13]. For example, low-level details such as Web cookies are shown in trace-driven simulation to affect the cachability of recourses.

2.3 Modelling the World Wide Web

In order to narrow the scope to relate to our work, we have paid particular attention to works that model Web page retrievals at the object-level, as this is an important component of our model. In this section, we start with a review of models that principally examine object-level WWW traffic patterns. This is then followed by an overview of models that investigate file migration in a distributed file system, including Web caching, and is the basis for our model.

We start our review into the nature of traffic in the WWW with [16], who created a model of the pattern of traffic on a simulated network that attempts to replicate patterns on a real network. In the model, they considered the interaction of system components, including: HTTP, network protocol, browsers, and Web servers. As well, they accounted for complex browsing patterns. For example, users intentionally or accidentally opening multiple browsers, users abandoning pages, and the ability of browsers to request multiple pages at once.

Work prior to [16] tended to use Web pages as the basic unit, whereas in contrast, Choi and Limb use the *Web-request*, which they identify as a sequence of requests that consist of multiple resources. The results of their paper showed that they were able to generate synthetic traffic from the model and found that it corresponds well with the trace data.

The work in [16] identifies two kinds of objects: a main object and objects that are linked from the main object, called in-line objects. The data in the model is mapped to alternating states: retrieving a Web-request (HTTP on), and a viewing period (HTTP off). On-time is a function of the number of in-line objects, in-line inter-arrival time, main-object size, in-line-object size, and a stall time (the tim between TCP bursts). These states and the relationship of the model entities are shown in Figure 2.4. The authors also collected Web-interaction trace data from Georgia Tech campus for 24,000 client browsing sessions of 1,900 clients. They excluded certain data that may have been affected by campus servers, and machine-generated traffic. Trace data were are separated parsed into Web-requests, which identified main objects and their associated in-line objects. Object classification was achieved by examining its file extension, which was further backed up by examining its MIME type.

To address the procedure and timing of Web page request patterns, we utilize the ideas presented by [2]. Their model addresses the problem of users experiencing Web browsing degradation as a result of slow response times due to the the increasing volume of traffic. They identify a wide range of causes, including: network delay, browser design, server overload or poor HTML page design. To analyze the situation,

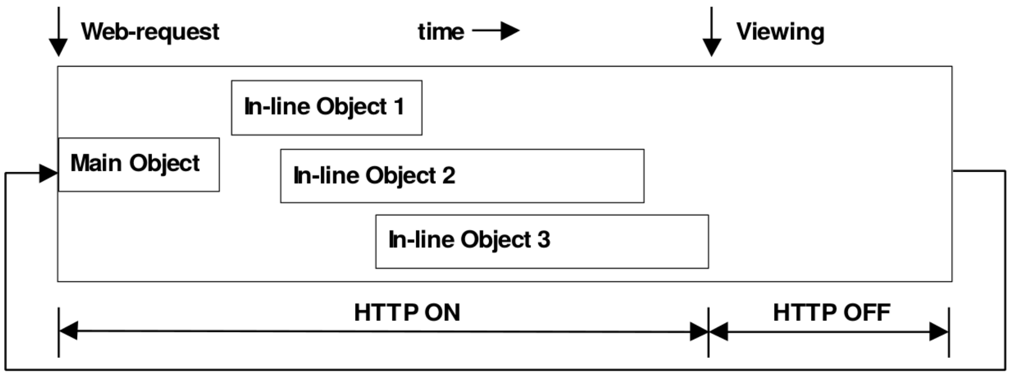


Figure 2.4: Overview of Basic HTTP Model [16].

a structured timing model was presented to capture the total delay and methods for measuring constituent parts of total delay.

The model represents the period of time from when a user requests a *Uniform Resource Locator* (URL) to the resolution of all of it constituent parts (files, applets, etc); this period is referred to as *Closure Over the Full Resolution of a URL* (CURL). Figure 2.5 shows an example of a CURL. CURL assumes that two fundamental standards associated with the WWW are HTML and HTTP, where HTML the defines the structure of Web pages, and HTTP forms a client-server, request-response model, and that all requests are handled successfully.

A CURL is generalized in three phases: (A) client-side HTML parse and request generation, (B) server-side request processing, (C) client-side rendering time. Phase (C) may result in further requests to the server. During a CURL, there are two types of network transit times: N_{CS}, the time taken for the communication phase between the client and server; and, N_{SC}, the time taken for the communication phase between the server and client.

The model presented in [2] decomposes the factors in the request-response process that contribute to the MRT. In particular, they conclude that network overhead is not substantial and rendering content on the client side and server processing time account for most of the wait [2]. While the paper is a bit dated, the work can provide a relative basis for the input parameter values chosen in our model implementation, and a comparison tool for model validation.

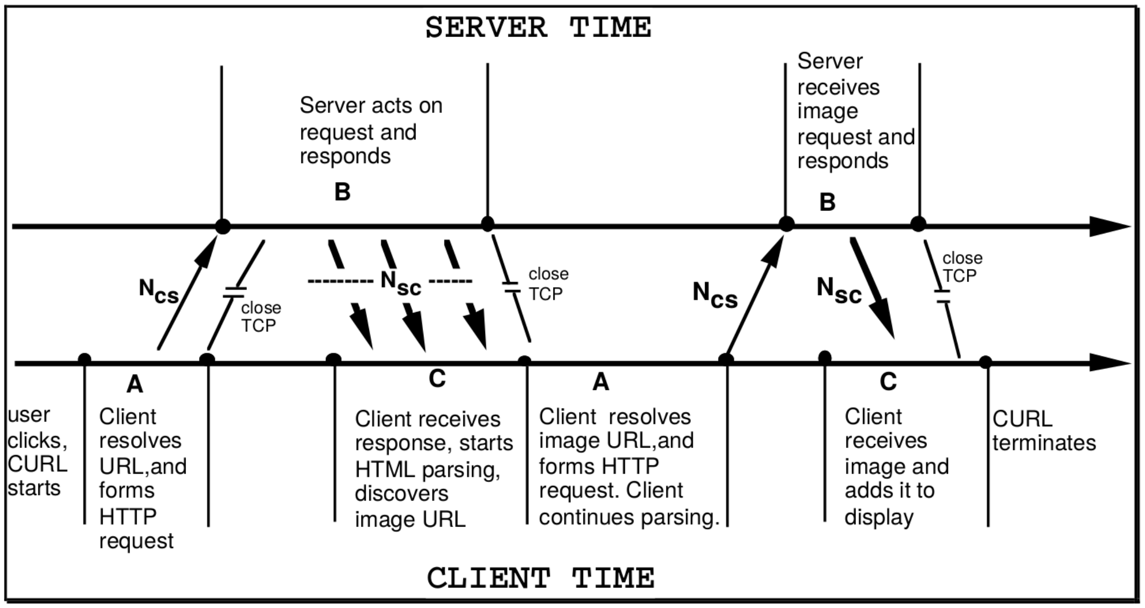

SERVER TIME
Server acts on request and responds
B
Server receives image request and responds
B
N_{CS}
close TCP
N_{SC}
close TCP
N_{CS}
N_{SC}
close TCP
A
C
A
C
user clicks, CURL starts
Client resolves URL,and forms HTTP request
Client receives response, starts HTML parsing, discovers image URL
Client resolves image URL,and forms HTTP request. Client continues parsing.
Client receives image and adds it to display
CURL terminates
CLIENT TIME

Figure 2.5: Example of a CURL [2].

N_{CS}	Client to server network transmission time.
N_{SC}	Server to client network transmission time.

Recognizing the rapid change in the nature of the WWW, [26] analyzed a significant amount of real Web data, and presented a novel analysis technique that provides insight into Web traffic patterns [26,27]. Using their results, they developed a Web caching approach designed for low-bandwidth and resource-limited developing world environments. These works are of particular interest as they provide specific data on general Web page object types and their rates of occurrence in WWW while being recent enough to be applicable to our work.

The data set utilized in [26] was large and detailed enough to provide insight into dynamic user-side interactions, and content-based Web caching approaches. Previous work by others focused on static Web pages, which does not require full content, just access logs (as it typically used in trace-driven WWW traffic analysis). The authors use five years of WWW traffic data (2006 to 2010) from the *CoDeeN* content distribution network, a globally distributed proxy system spread over 187 countries, seeing over 70,000 users per day. Their data set consisted of a sample of the month of April for each year, isolating four countries: United States, Brazil, China, and France. The overall data set represented 48 to 137 million requests, 689 to 1,903 GB traffic, and 70 to 152 thousand users per month.

Ihm developed and employed their own analysis technique referred to as *Stream-Structure* [26,27]. The input data that was analyzed was a temporally related stream of HTTP objects, with no indication of their relationship to Web pages. The *Stream-Structure* algorithm analyzes the objects within the data stream to identify groups of related objects that correlate as a Web page request. The authors compare *Stream-Structure* to two alternatives. The first was a *time-based* approach in which groups of objects are considered to be from the same page request if the idle time between objects is short enough, based on a predetermined threshold [7,30,37]. The other technique was *type-based* in which an HTML object is considered to be the main Web page object, and all subsequent non-HTML objects are assumed to be part of the same request [16]. One problem that they identified is that client-side interaction (such as in embedded code) can cause longer client-side idle times, thus leading to inaccurate results in *time-based* approaches [26]. As well in the case of the *type-based* approach, misclassified data can result in erroneously identifying a request that is actually part of another request[26].

StreamStructure uses three stages to associate objects to Web pages: (1) grouping of streams, (2) detecting main objects, and (3) identifying initial pages. The entities and their relationships that result from this algorithm are described in Figure 2.6 In grouping of streams, the algorithm relates objects that have the same HTTP *referer* [sic] field to create groups of multiple independent streams. The referrer (misspelled in the HTTP standard) is an HTTP field which contains the address of the originator of the request [19]. An empty referrer field indicates a the root of a new stream. In [26], Ihm assumes that the information gathered using this referrer field can be relied on because it is enabled by default in most browsers being used at the time.

In the second stage, the main object is detected from the grouped streams identified in the first stage [26]. This uses a combination of previously mentioned type-based and time-based approaches. Objects that are not identified as a main object are then labeled as embedded objects and linked to the main object that preceded them. Finally, in the last stage, the initial pages are identified from the main objects, which also include user-side interactions. This stage again uses a time-based approach, but since the idle time includes *domain name server* (DNS) lookups which can vary widely, a different approach is used. Instead, the occurrence of the event *DOMContentLoaded* reported to a Google Analytics server indicates that the Web page is successfully loaded, and from that point the time-based approach is used to identify initial pages [26].

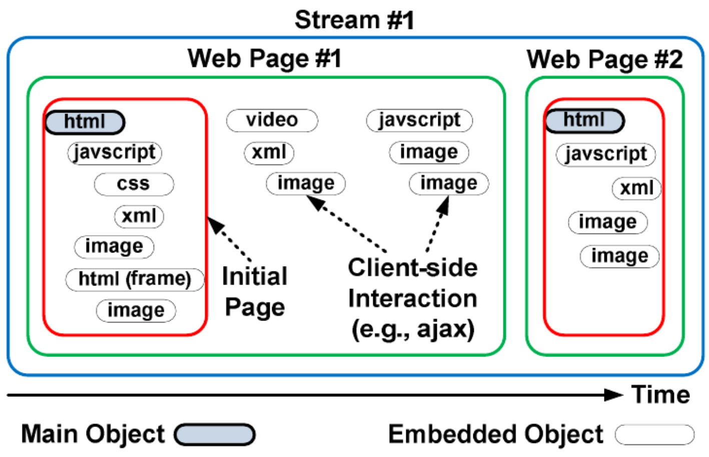


Figure 2.6: Streams, Web Pages, Initial Pages, Client-side Interactions, and Main / Embedded Objects [26].

Using their *StreamStructure* algorithm to analyze their data, Ihm developed a Web traffic model [26]. They divided all Web pages into three groups, categorized by total page time as follows: *short* — less than 25th percentile, *medium* — 25th to 75th percentile, and *large* — more than the 75th percentile. Of the many IANA media types, they considered the following eight Web page objects: HTML, CSS, XML, JavaScript, image, flv-video (Flash video), other-video, and octet. A summary of their observations are shown in Table 2.1.

Total Page Time	Request frequency (%)	Median Number of Objects	Additional Observations
short	5	4	• consist mostly of HTML • mainly related to search activities
medium	40	12	• have many images • mixture of content between small and large • eg. news, blogging
long	55	30	• heavy client interaction • higher percentage of video and octet-stream bytes (indicative of video watching and large file downloads)

Table 2.1: Summary of Web Traffic Model Results [26].
(based on approximately 8.6×10^6 requests from the US in 2010.)

In [26], Ihm went on to examine Web caching, with a focus on comparing *object-based* to *content-based* caching. In object-based caching the whole object is cached, which means that if the content remains unchanged but its URLs changes, it is uncacheable. Content-based caching (also known as micro-caching), however, splits an object into chunks and caches each chunk [26,41]. Content-based caching is protocol independent. The authors found that content-based caching is more effective than object-based caching. While hit rates between the two were similar, the byte hit rate for content-based caching is around two times that of object-based caching. When calculating ideal byte hit rate based on infinite cache storage, they also found that object-based caching results in 27-37 %, which is in contrast to an actual byte hit rate of 17-28 %.

Object cacheability was also discussed in [26]. To determine the cacheability of objects, they used HTTP fields from their full data from 2010. Overall, about 20-

32 % of URLs were uncacheable, which represents roughly 22-28 % of total requests, and 10-15 % of total bytes. HTML and JavaScript were found to be less cacheable, which they assume is because they tend to be dynamically generated. Interestingly, the authors noted some differences in the cacheability of some content types between countries, due to the regional popularity of some applications and browsing behaviour.

The effect of various cache replacement policies with respect to types of Web resources were examined in [21]. Examples of resources they considered include images, text, video, which relate to our work. They used a workload trace of HTTP requests from a cache research project (IRCACHE Web Caching project). The content-types they considered were: applications, audio, images, text, and video, which they analyzed the request characteristics according to document size and content-type. One notable assumption that they made with respect to our work is that they eliminated requests that are dynamically generated as these are considered not cacheable. For the purpose our work, we are most interested in their results as shown in Table 2.2. We will see in Chapter 4, that with this data, we can establish a frame of reference for our own object sizes.

	App.	Audio	Images	Text	Video
Mean (Kbytes)	41	123	4	14	260
Median (Kbytes)	3	7	1	3	573
Std. Dev. (Kbytes)	761	948	21	104	974
Requests (%)	8.08	0.28	75.54	15.77	0.12
Bytes (%)	33.51	3.49	36.33	23.15	3.19
Distinct documents (%)	26.90	38.00	45.98	33.92	75.02

Table 2.2: Trace Characteristics by Content-type of Documents [21].

In [25], Hurley developed a model to investigate file migration in a distributed file system. The distributed file system under study consisted of a collection of homogeneous file storage sites and a finite number of user workstations interconnected by a high-speed LAN. Each storage site consists of a single processor, and is represented as a single server queue. A file request is served by the storage site that contains the file, whose location is predetermined. Requests are placed into the storage site queues in a first-come first-served fashion. Service times are exponentially distributed. If a storage site becomes congested, as indicated by a measure of queue length, a file migration is initiated. The file i which has the highest *possible gain* (PG_i for some file i) is chosen for migration and it (along with corresponding requests) are moved to the storage site with the smallest queue length.

File request popularity, which is a feature adopted in our work, was modelled using a piecewise function to produce the request probability distribution, p_i [25]. Two states were used to model file popularity, *popular* and *normal*, which affect the level of variability in the file access intensities. A file in a *popular* state is requested more frequently so it has a higher request probability. The ratio of request probabilities of popular to normal files is v, and there are assumed to be M files, of which M_p are popular files, and $M - M_p$ are normal. Equation (2.1) presents the dynamic version of request probabilities. The probability of requesting a page at time t is $p_i(t)$.

$$p_i(t) = \begin{cases} \frac{v}{vM_p(t)+(M-M_p(t))} & \text{if file } i \text{ is in the popular state} \\ \frac{1}{vM_p(t)+(M-M_p(t))} & \text{if file } i \text{ is in the normal state} \end{cases} \tag{2.1}$$

In addition to the probability of requesting a Web page, Hurley uses two types of files in the model, *potentially popular* and *conventional.* Potentially popular files transition from a normal to popular state at a rate of γ_1, and from popular to normal state at a rate of γ_2. Conventional files always stay in the normal state. Potentially normal files are represented as $M_0 \leq M$. Figure 2.7 displays this dynamic scenario with its *potentially popular* and *conventional* files.

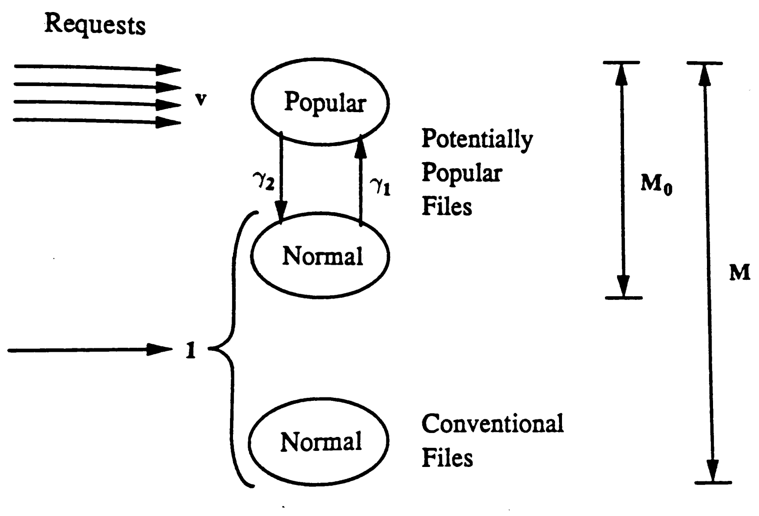


Figure 2.7: Relationship Between Dynamic File Reference Model Parameters [25].

The model has since been adapted to explore Web caching in the WWW [22,24]. In this work, Web caches are synonymous to the file storage sites in the original research, and the files being migrated now represent Web pages. Several topics have been successfully explored in these subsequent investigations, including distributed and hierarchical Web caching architectures, partitioning, effects of dynamic content, the effects of file size, and load balancing in Web caching [23,24].

2.4 Summary

We have presented an introduction to the history and functionality of the WWW. This included a discussion of HTML and HTTP, which are both important for understanding the WWW. As well, an overview of Web caching was given, as this will be used as an application for our model. Our overview of the WWW was then followed by a review of five other models. Some of these models were conducted at the object level, which is important to us in our own model. Much of that other work relies on collecting trace data and applying a model to classify the nature of the data [2,16,21,26]. However, in contrast, Hurley uses a queuing model to represent the interarrival of files [25]. In that queuing model, the basic unit of data is individual files, whereas in our research (Chapter 3), we build upon [25] to develop a model that has the Web page object as the basic unit of data.

Chapter 3

Model

3.1 Introduction

In this chapter, we present our User-Web Interaction Model that represents the process of users interacting with the WWW to obtain Web pages. We use a queuing model to capture the behaviour of the user requests that are sent to the Web servers containing the Web pages, and the return of the requested pages. A novel feature of our model is that Web pages are represented at the object-level. The main performance metric in which we are interested is MRT: the time from when a user requests a Web page until the Web page (and all of its objects) has been delivered. This performance measure can influence a user's experience substantially [17].

Our *Web Page Request Access Pattern Model* provides a facility to evaluate a variety of parameters that affect performance which will allow us to generate a multitude of scenarios. The reality of the WWW is that it is complex: it involves the transmission of large and varied amount of informations via the Internet, through a variety of devices spanning many regions, countries, and even into space. This model attempts to capture much of this and allows for the relative comparison of the performance of various system configurations.

The remainder of this chapter is organized as follows: in Section 3.2.1, we present the *User-Web Interaction Model*, which captures the overall behaviour where users submit requests for Web pages to Web servers, and then receive the desired Web pages. This high-level model is comprised of three sub-models: the *User-Request Model* (Section 3.2.2) which represents the set of users that make requests for Web pages as well as how they select Web pages, the *Server Model* (Section 3.2.3) which consists of a set of Web servers that respond to user requests, and the (Section 3.2.4) which represents the Web pages in the system.

3.2 Model

3.2.1 User-Web Interaction Model

The high-level model for User-Web Interaction is shown in Figure 3.1 and represents the total process of a user requesting a Web page from a server, and that server's subsequent response. To capture the behaviour of the users, servers, and Web pages, the model in Figure 3.1 is comprised 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. The basis of the model is an $M/M/K/N/N$ queuing system; with K Web servers and N users [28].

The interaction of our three sub-models form the *request-response process*. This process begins with the user selecting a Web page using the Web Page Selection Model described in Section 3.2.2. The Web page request is transmitted to the Web server that eventually supplies the requested Web page to the user. If the Web server is currently busy servicing another user request, then the request is placed into a wait queue; otherwise the request is serviced immediately. The time to service a

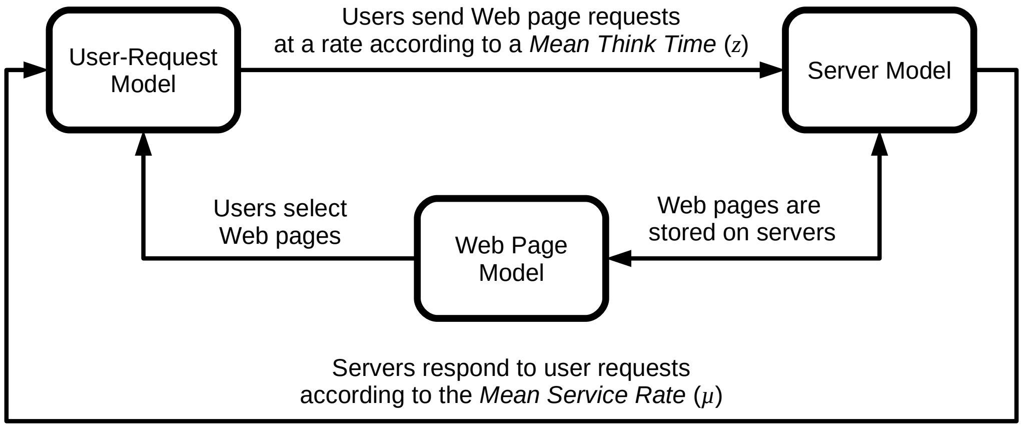

User-Request Model
Users send Web page requests at a rate according to a Mean Think Time (z)
Server Model
Users select Web pages
Web Page Model
Web pages are stored on servers
Servers respond to user requests according to the Mean Service Rate (μ)

Figure 3.1: User-Web Interaction Model.

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 (this is referred to as the *think time*).

The timing of the request-response process for a single user is demonstrated in Figure 3.2. This shows the user alternating between two states: *thinking* and *waiting*. Each of these states have a corresponding duration, the *think time* (z) and the *reponse time*. The *think time* is a parameter in our model. The *reponse time* is the time from the user initiating the request until receiving the response; with the MRT of all requests that occur in the system being our main performance measure. Response time represents the sum of the transmission latency, the time in queue that the request awaits processing, and the request processing time — which we refer to as *request service time* (R_i).

Web servers can be in one of two states: *idle* or *busy* (see Figure 3.2). While a server is *idle* it is waiting to receive a request. While *busy*, a server is processing a request and can only process one at a time. If another request is received by the server, it goes to the server's wait queue. When a server becomes idle, we assume that requests waiting in its queue are processed on a *first-come-first-serve* (FCFS) basis.

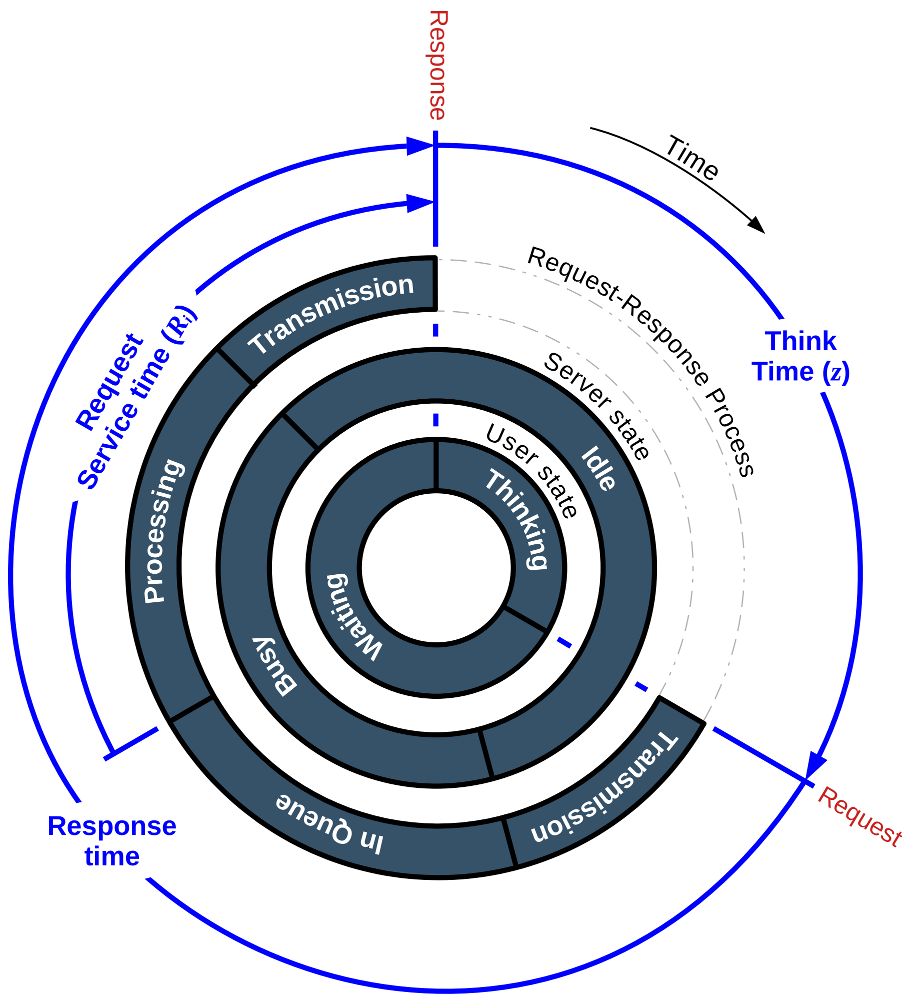


Figure 3.2: Request-Response Timing Cycle

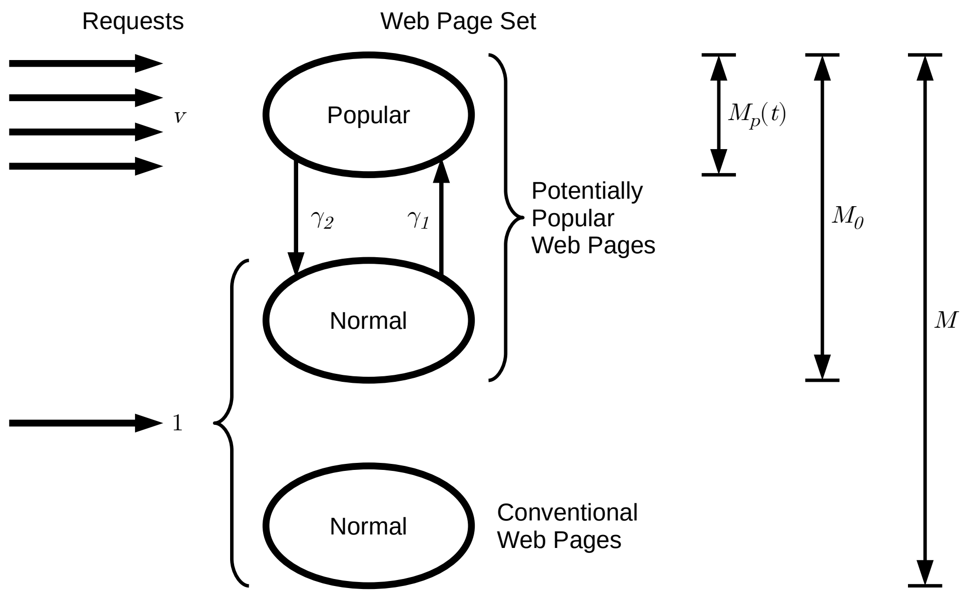
The *think time* controls the rate at which users request Web pages. A lower think time leads to a greater load on the Web servers. When not in the system waiting for a request to be served, a user is in a think state, which represents a user processing a previously retrieved Web page. We assume that the *think time* for users is independent and exponentially distributed, with a mean z. A user will not request another Web page until they have completed their think state.

3.2.2 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 3.2.1 [6].

The Web Page Selection Model represents the mechanism by which users select Web pages. Our model incorporates the *File Reference Model* developed in [25] and shown in Figure 3.3, where the probability that a user selects a particular Web page is not uniform, but instead dynamic.

The model uses a Web page popularity that varies according to a discrete Markov Chain. With this approach, the M Web pages are assumed to be either in a *popular* or *normal* state: where popular pages have a higher chance of being selected than normal ones (by a factor of v). However, not all pages are able to be popular; a subset of Web pages are assumed to be potentially popular (M_0) and all others are considered conventional ($M - M_0$). This is done to achieve a higher coefficient of variation, which is typical of a Web page request in the WWW environment [3]. The potentially popular pages are assumed to alternate between normal and popular states, whereas conventional Web pages remain in the normal state. The rates at which potentially popular Web pages transition to popular is governed by an exponentially distributed parameter γ_1, and return to the normal state at a rate of γ_2. At any given time there are $M_{\mathrm{p}}(t)$ Web pages currently popular. The probability of selecting a Web page, i, at time t, is represented by the equation for $p_i(t)$ shown in Equation (3.1).



Legend	
M	number of Web pages
M_0	number of potentially popular Web pages
$M_{\mathrm{p}}(t)$	number of Web pages currently popular
v	popularity factor
γ_1	normal-popular transition rate
γ_2	popular-normal transition rate

Figure 3.3: Web Page Selection Model [25].

$$p_i(t) = \begin{cases} \dfrac{v}{vM_{\mathrm{p}}(t) + (M - M_{\mathrm{p}}(t))}, & \text{if Web page } i \text{ is in the popular state} \\ \\ \dfrac{1}{vM_{\mathrm{p}}(t) + (M - M_{\mathrm{p}}(t))}, & \text{if Web page } i \text{ is in the normal state} \end{cases} \tag{3.1}$$

3.2.3 Server Model

The set of Web servers in our model consist of one or more devices which store the Web pages and service 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).

Figure 3.4 presents the detail of our Server Model within the User-Web Interaction Model. When a request is received at a server, the server is marked *busy.* We assume that Web servers can only process one request at a time. If a request is received at a server that is busy, then the incoming request is placed in the server's wait queue (we assume FCFS, however, this discipline may be modified to suit the desired investigation). Once a server completes a request, its status is set to *idle*, and then it 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.

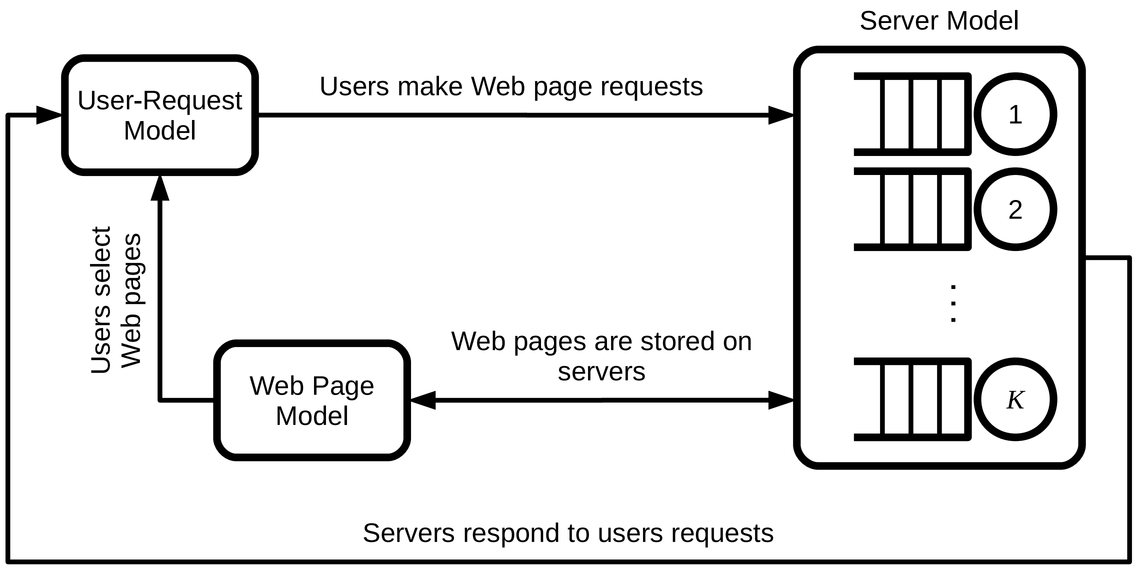

Server Model
User-Request Model
Users make Web page requests
1
2
K
Users select Web pages
Web pages are stored on servers
Web Page Model
Servers respond to users requests

Figure 3.4: 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 to transfer it to the user. It begins when a Web server removes a request from its queue, and completes once the page (and all its objects) is transfered to the user. The Web page size (W_i) influences the request service time, as we assume that Web page size is equal to its transmission time (τ_i). We show in Equation (3.2) that the Web page size is 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 service time is assumed to be exponentially distributed with a mean of

$\mu^{-1}\tau_i$ (Equation (3.3)). Thus, the service time for a Web page is proportional to its Page Size. The number of objects (Q_i) and their object type is an important aspect of our model, and one that can be varied as needed to suit the desired Web Access Model.

$$\tau_i = W_i = \sum_{j=1}^{Q_i} \Theta_{ij} \tag{3.2}$$

$$R_i = \mu^{-1}\tau_i \tag{3.3}$$

3.2.4 Web Page Model

The Web Page Model represents the totality of information that is requested and delivered to the user. It is modelled as a collection of Web page objects that are transmitted from the Web server to the user upon request. Each Web page object is assumed to be transmitted independent of one another, but 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 the *Web Page Object Set*, which is determined according to a *Web Page Category*.

Web Page Object Set

A Web page consists of a collection of one or more Web page objects, which we refer to as the *Web page object set*. The composition of the object set is assumed to be determined by both the number and type of objects. The distribution of the objects and their object types is influenced by the Web page category, which is a parameter of the Web page and will be discussed shortly.

Each object consists of two main attributes: *object type* and *size*. The *Web Page Object parameters* are summarized in Table 3.1, which shows each object type with their assumed size and examples. We have generalized the media types described in [26] into the following *object types*: *text, program script, CSS, image, audio, and video*. While this set is not exhaustive, these six Web page object categories account for the majority of information being transferred via the WWW [26]. As shown in Equation (3.3), object sizes directly effect the Web page size, and thus influences the request service time (R). Size is categorized as either small, medium, or large.

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

Table 3.1: Example Web Page Object Parameters

Web Page Category

The next feature of each Web page is the *Web page category* which characterizes the composition of Web pages as defined by our model and as presented to the user. 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. It is assumed that the page category of a Web page affects the retrieval time of Web pages due to the potentially inherent differences in Web page sizes and Web page object cacheability.

In our model, we specify three Web page categories: *article*, *media*, *mosaic*. These choices are based on anecdotal observations, somewhat influenced by [26], while other page categories and their compositions are certainly possible. The following summarizes our three Web page categories:

article: Typically contains information about one specific topic. Its composition is greatly influenced by a substantial amount of script and text and a large number of images, but has very little audio or video information.

Examples: news articles, Wikipedia article pages.

media: While this page category is similar to articles, the main difference is that a media page provides more audio and/or video. Media is dominated by large amounts of video and/or audio information, and a moderate to large amount of image information.

Examples: image collages such as Google Images, Flickr, Pinterest; image pages, music and video streaming.

mosaic: Web pages that provides multiple links and summaries to other specific topics. Their Page Sizes are primarily influenced by text and script, with some image information and a moderate amount of audio and/or video.

Examples: search results such as Google Web search, Pinterest image search; website main pages, social media threads, Wikipedia's main page.

The breakdown of Web pages into Web page categories in our system is an input parameter. Web pages in each page category are configured according to an input table that describes the way in which objects are to be distributed. Figure 3.5 show a graphical example of this configuration for M Web pages and their distribution using the Web page categories. Some of the Web pages are shown to be potentially popular, according to M_0. Figure 3.5 displays one example Web page from each of the three Web page categories to illustrate how the page category affects the object distribution. Each example object is represented as a bar graph showing the number of objects per object type, where the size and colour of the object indicates its size.

From Figure 3.5, we can observe that the Web page that is an *article* 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 to an *article* Web page, *media* Web

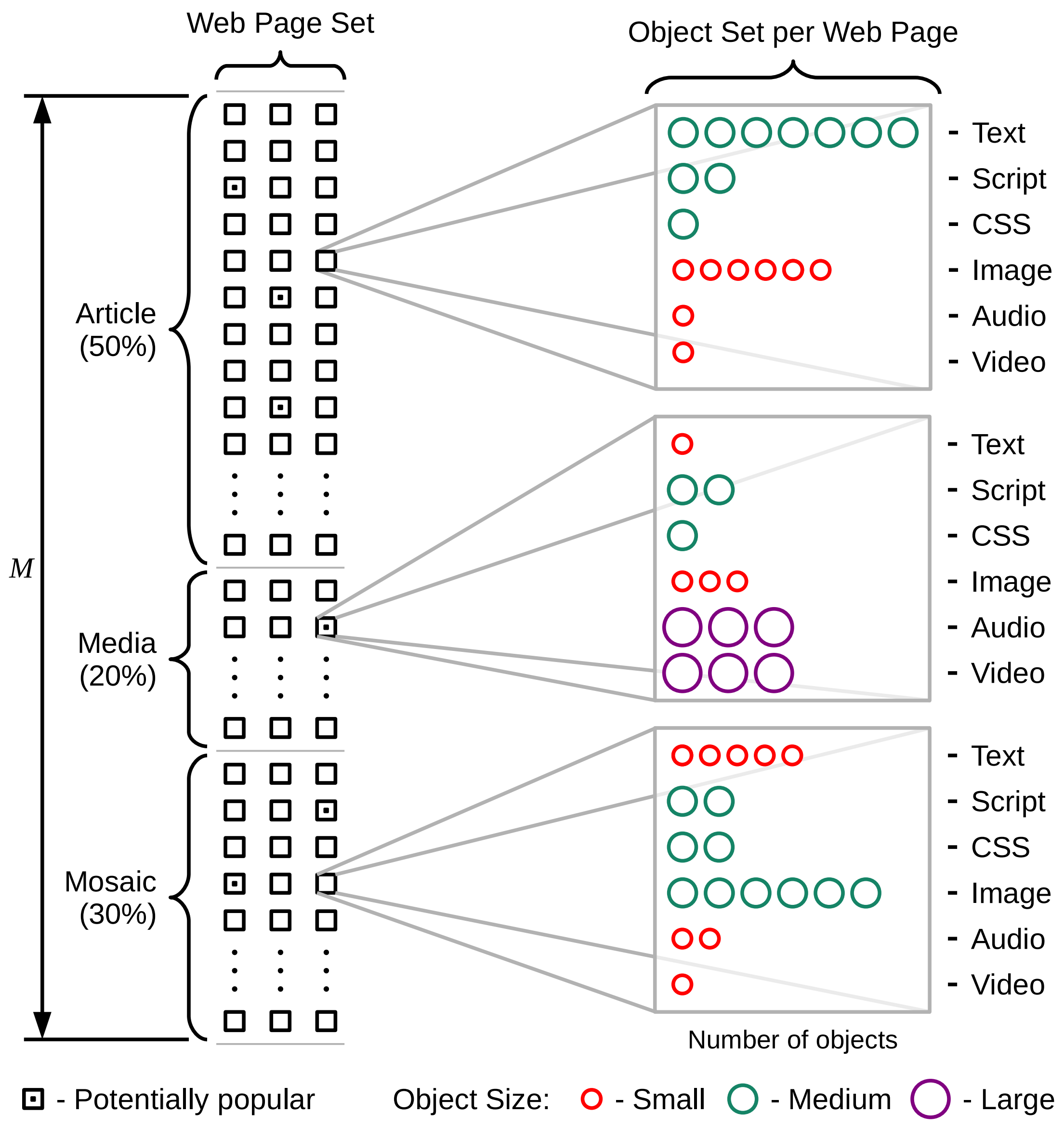


Figure 3.5: Example Web Page Composition Based on Category.

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.

Chapter 4

Implementation and Results

4.1 Introduction

Based on our model presented in Chapter 3, we have developed a simulation to test our model and conduct experiments. In this chapter, we will briefly discuss its implementation, validation and verification, and consider various model parameters. 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 4.1.1).

Our examination of model parameters will begin with Web page composition and determine its 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, as well as number of objects. With these parameters established, we then identify and examine the parameters that affect simulation stability and system load. As well, we investigate coefficient of variation of Web page request interarrival times, including its effect on MRT. Finally, we present an experiment that varies the ratio of Web page categories to observe its effect on MRT.

We use a *discrete event simulation* (DES) approach to implement our model. DES 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[6]. 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 is provided in the Simulation Implementation Overview (Appendix A).

4.1.1 Validation and Verification of the Model

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 that 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 4.1 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).

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

Table 4.1: Comparison of Simulation MRT versus Analytic MRT for an $M/M/1//N$ System for Various Values of N.
$K = 1$, $\mu = 1$, $z = 100$, $M = 100$, $M_0 = 0$

4.2 Model Parameters

4.2.1 Composition of the Web Page Set

The *Web Page Set* consists of several parameters that affect the performance of our system, in particular MRT. We refer to the collection of parameters that determine the make up of the Web Page Set as the *Composition of the Web Page Set.* Ultimately our goal here is to establish the sizes of all Web pages in the system, which vary according to Web page categories. There are several model parameters that influence the Web page size, which we present here.

Recall that in Chapter 3.2 we introduced our *Server* and *Web Page* models. In these models we described the parameters that compose the Web Page Set. Figure 4.1 provides a visual summary of the relationship between these parameters. The Web Page Set consists of M Web pages, where each Web page has a composition that follows one of our three Web page categories: *article*, *mosaic*, *media.* The number of Web pages of each category in our Web Page Set is broken down by the *Ratio of Web Pages per Web Page Category.* Each Web page, i, has a *Web Page Object Set*, which consists of Q_i objects of various types and sizes. The object types and sizes, as well as the quantity 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).

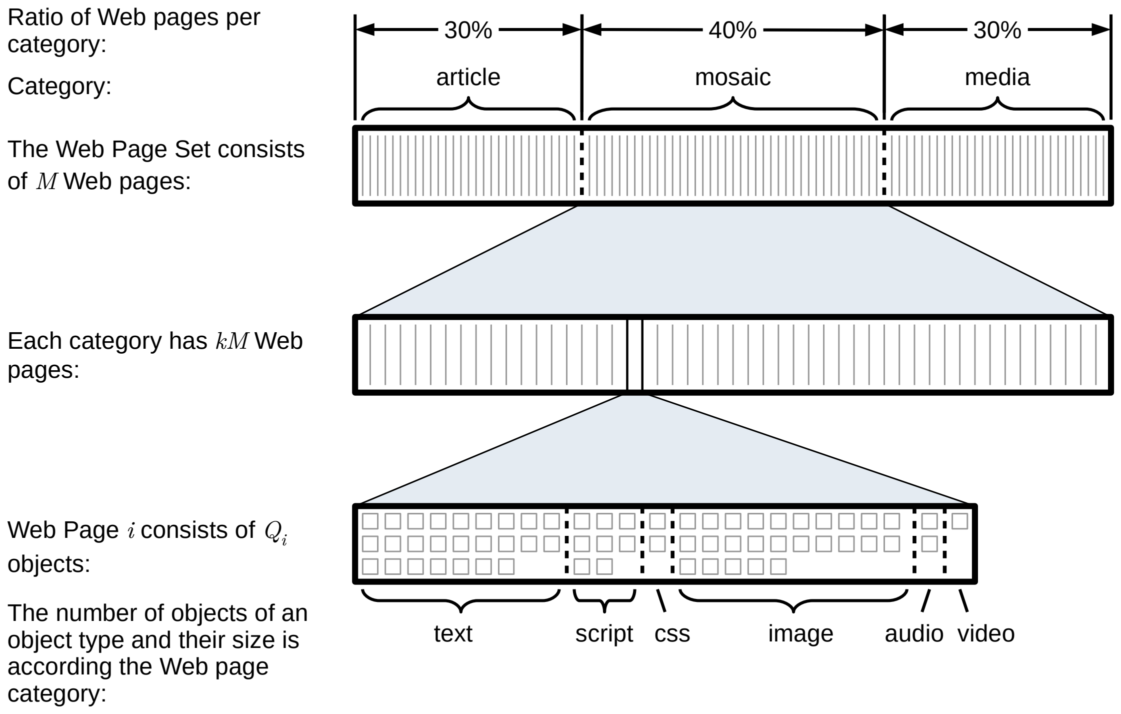

Ratio of Web pages per category:
Category:
The Web Page Set consists of M Web pages:
Each category has kM Web pages:
Web Page i consists of Q_i objects:
The number of objects of an object type and their size is according the Web page category:
30%
40%
30%
article
mosaic
media
text
script
css
image
audio
video

Figure 4.1: Visual Summary of the Composition of the Web Page Set.

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, and are expected to be greater than $0\,\%$. We now introduce the term k_{category} to be the proportion that a *category* represents in the Ratio of Web Pages per Category. Thus, there are kM Web pages per category.

We next introduce the concept of *Web Page Category Base Scenario* to be the basis for establishing other model parameters in this section. It has the following ratio of Web pages per category: $k_{\text{article}} = 30\,\%$, $k_{\text{mosaic}} = 40\,\%$, and $k_{\text{media}} = 30\,\%$. The values for the Base Scenario were estimated to result in approximately the middle of the range of possible mean Web page size for all requests. Later in this chapter, we will investigate the effect that the Ratio of Web Pages per Category has on MRT.

As stated earlier, the main performance measure of interest in our research is MRT, which we assume to be proportional to the request service time. Since request service time is proportional to Web page size (Equation (3.3)), we can conclude that Web page size is proportional to MRT. Thus, we can then predict the effect that Ratio of Web Pages per Category has on MRT by observing the *Mean Web Page Size for All Requests* ($\overline{W}$). This is an output value of the simulation, and is the sum of all requested Web page sizes divided by the number of requests. We can use Equation (4.1) to estimate the *Expected Mean Web Page Size for All Requests* ($E\left(\overline{W}\right)$).

$$E\left(\overline{W}\right) = k_{\text{article}}W_{\text{article}} + k_{\text{mosaic}}W_{\text{mosaic}} + k_{\text{media}}W_{\text{media}} \tag{4.1}$$

The *Composition of the Web Page Object Set* is important as it establishes the size of each Web page. Consider Table 3.1, which lists the Web page objects in our model. We expand this table to represent our complete Composition of the Web Page Object Set, which now includes the Web page categories and their associated Web page object media types. Each category and object type has a categorical object size, and a number of objects. This updated table representing Composition of the Web Page Object Set is shown in Table 4.2. We continue our discussion by laying out criteria to establish values we used to complete Table 4.2.

We begin by establishing the categorical Object Sizes (Θ) for each object type. The Object Sizes given in Table 4.2 are influenced by the percentage of bytes per page type described in [26], combined with an anecdotal representation of a Web page in each category. The Composition of the Web Page Object Set attempts to generalize the various Web Page Object Media Types of real world objects. Consider for example, video which represents a large range of data sizes with many considerations, such as: video length, quality, and format. In our model, we have captured this range of sizes with three categories: *small* (Θ_{small}), *medium* (Θ_{medium}), and *large*

Category	Object Type	Θ	Q
Article	text	small	25
	script	small	20
	css	small	5
	image	small	25
	audio	small	5
	video	—	—
Mosaic	text	small	60
	script	medium	10
	css	small	5
	image	small	60
	audio	medium	2
	video	medium	1
Media	text	small	20
	script	small	15
	css	small	5
	image	medium	15
	audio	medium	5
	video	large	3

Table 4.2: Composition of the Web Page Object Set.

(Θ_{large}). In our simulation, we assigned a value for each Object Size.

We assume that these Object Sizes are fixed in our model with $\Theta_{\text{small}} = 1$, and $\Theta_{\text{large}} = 100$ (Θ_{medium} will be discussed in the following paragraph). As we are only interested in the relative effect of object sizes, we do not attempt to correlate them to bytes or octets. Thus, we assume the small object size ($\Theta_{\text{small}} = 1$) to be the base unit for all measures of data size in our model.

In order to define the value for the Medium Object Size (Θ_{medium}) we looked to previous research. Figure 4.2 summarizes the mean document sizes from [21] (refer to Table 2.2 in Chapter 2), in which the document sizes have been scaled to our small and large Object Sizes (between 1 and 100) and sorted from smallest to largest. By choosing the median of the document sizes (which is *Application* and equal to 14.3) as our Medium Object Size, we approximate the shape of the graph. We can therefore set $\Theta_{\text{medium}} = 15$. The resulting three Object Sizes have been overlaid on Figure 4.2 as dashed lines to show their relative sizes.

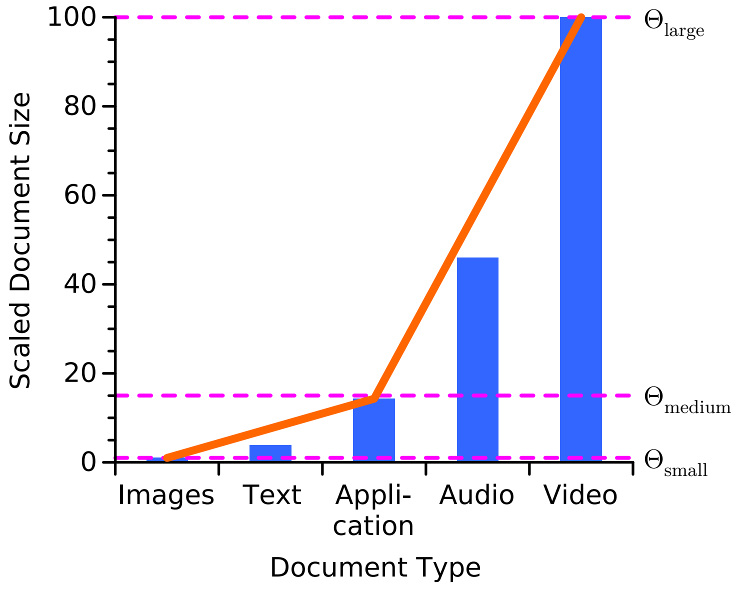


Figure 4.2: Document Size with Respect to Document Type (based on [21], where the document sizes have been scaled to between 1 and 100).

Web page categories are a key feature of our model, as they allow us to characterize the effect that different compositions of Web pages have on MRT. Thus, 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. While examining Web traffic patterns in [26], the authors modelled Web pages as being *short*, *medium*, or *long* in terms of total page load times. They observed that *medium* pages took 3 times longer than *short* pages, and *long* pages took 6 times longer than *short* pages.

We apply their observations of *short*, *medium*, and *long* pages to our Web page categories (*article*, *media*, *mosaic*) in how they affect MRT. This provides a basis for us to establish our differences in mean Web page size between categories. However, to exaggerate this effect we, used higher multipliers (4 and 8) than those observed in [26]. Thus, using W_{article} as the basis, we set the approximate Web page size per category to be:

$$\begin{aligned} W_{\text{mosaic}} &\approx 4\,W_{\text{article}} \\ W_{\text{media}} &\approx 8\,W_{\text{article}} \end{aligned} \tag{4.2}$$

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.

Our last task in establishing the Composition of the Web Page Object Set is to determine the Number of Objects that a Web page contains. For this task we use Equation (4.2) as a guide. The Number of Objects is an important consideration for this research because we will be applying our model in Chapter 5, by introducing object-level Web caching. We assume that it is important for Web pages to have a sufficient Number of Objects available for caching. To accomplish this, we introduce one additional assumption that the average Number of Objects per Web page is approximately 100 [4]. With these criteria in mind, and influenced by [26], we arrive at the Number of Web Page Objects (Q) that are shown in Table 4.2.

Based on Table 4.2, Table 4.3 shows the number of objects and the Web page size for each Web page according to their category. Finally, Figure 4.3 provides a visual summary of the data from Table 4.3.

The number of objects across our three categories (80, 138, and 63) have an average of 94, which we feel is reasonable, as it is near our goal of 100. The resulting size of each Web page corresponds to the size ratios determined by Equation (4.2), that is:

$$\begin{aligned} W_{\text{article}} &= 80 \\ W_{\text{mosaic}} &= 320 = 4.0\, W_{\text{article}} \\ W_{\text{media}} &= 640 = 8.0\, W_{\text{article}} \end{aligned}$$

(recall that all size units are in terms of $\Theta_{\text{small}} = 1$). Therefore, we will use the values from Table 4.2 for the object composition for all remaining experiments.

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

Table 4.3: Number of Web Page Objects and Web Page Size for each Web page category.

4.2.2 System Stability

The goal of examining system stability is to establish a simulation length which provides stable results (not affected by the transient period). We assume that when the

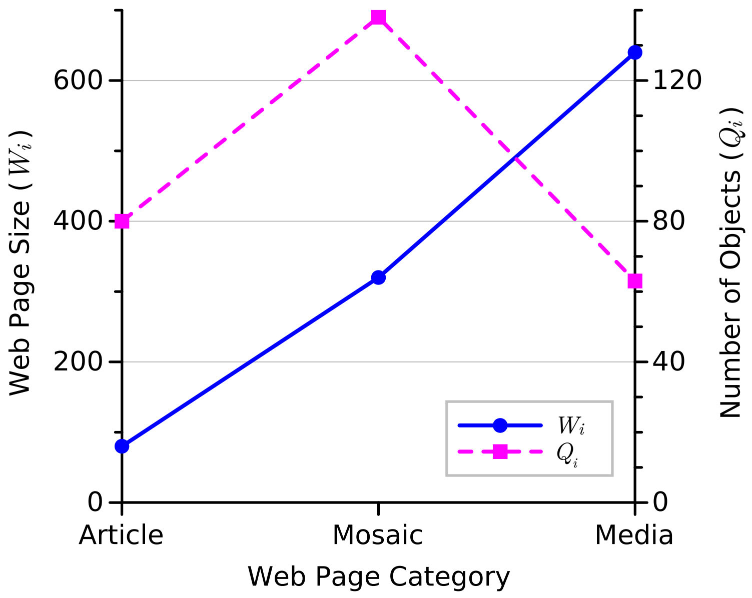


Figure 4.3: Summary of Web Page Composition.

performance measure (such as coefficient of variation) converges with the expected value, the simulation has reached equilibrium.

In Figure 4.4, 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.

4.2.3 Coefficient of Variation of Web Page Request Interarrival Time

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 [25]). 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

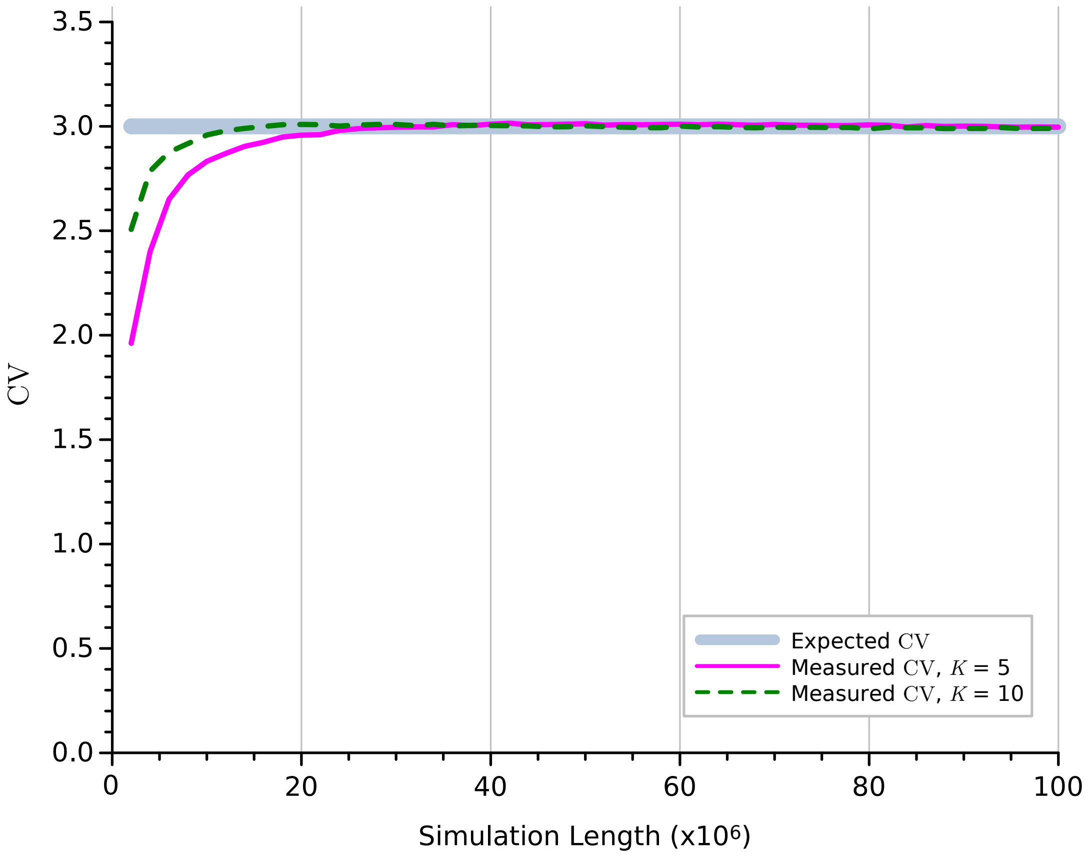


Figure 4.4: Effect of Simulation Length on Coefficient of Variation of Web Page Request Interarrival Time (CV).
$v = 90$, $\rho \approx 85\,\%$ ($K = 5$, $N = 480$, $z = 35{,}000$, Web Page Category Base Scenario)

popular) [25]. 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.

To determine the popularity factor (v), we examined the effect that varying v has on the coefficient of variation; these results are shown in Figure 4.5. We can see in Figure 4.5 that CV increases with respect to v, with little variation. As v increases CV begins to level off, which is expected [25].

Finally, Figure 4.6 shows the effect that the coefficient of variation has on MRT. This graph shows 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 %) [25]. The reduced variation after CV = 4

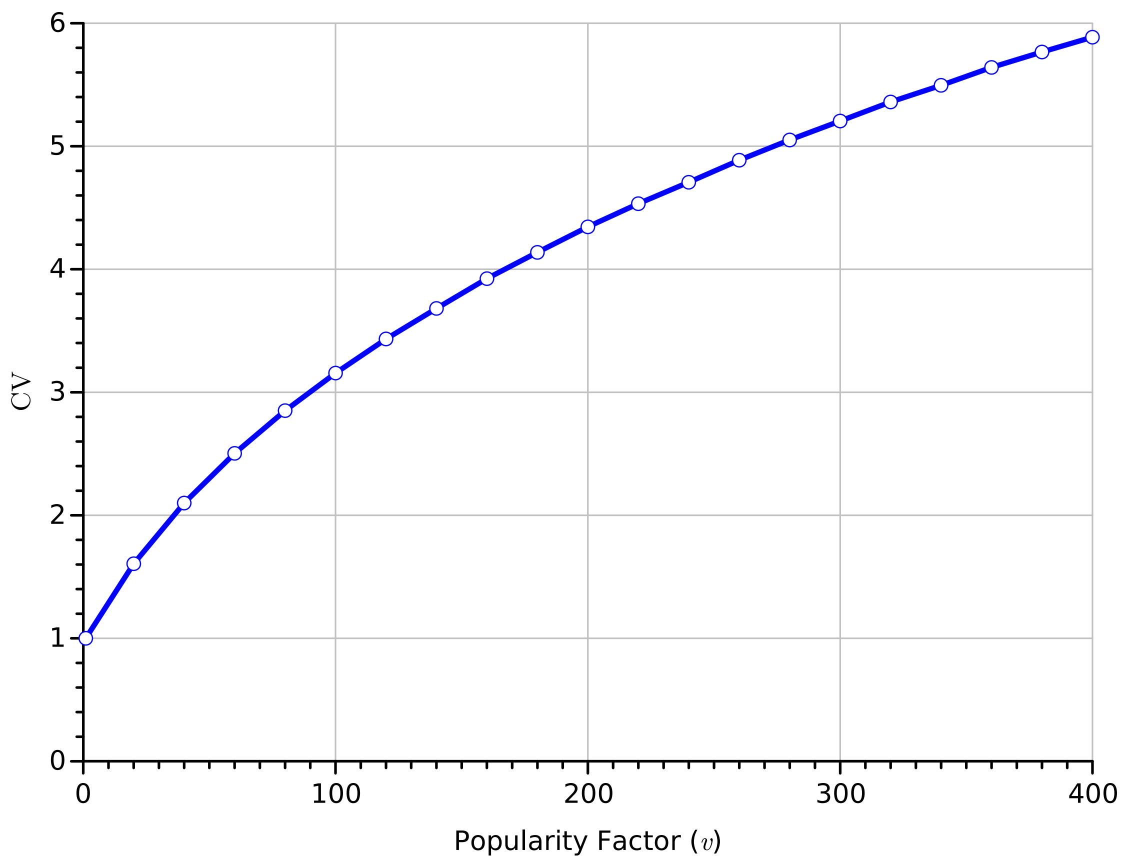


Figure 4.5: The Effect of Popularity Factor (v) on CV.
$K = 5$, $M = 10{,}000$, $M_0 = 1{,}000$, $\gamma_1 = \gamma_2 = 200{,}000$,
$\rho \approx 87\,\%$ ($K = 5$, $N = 500$, $z = 35{,}000$, Web Page Category Base Scenario)

seems to follow the leveling that was shown in Figure 4.5, which 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.

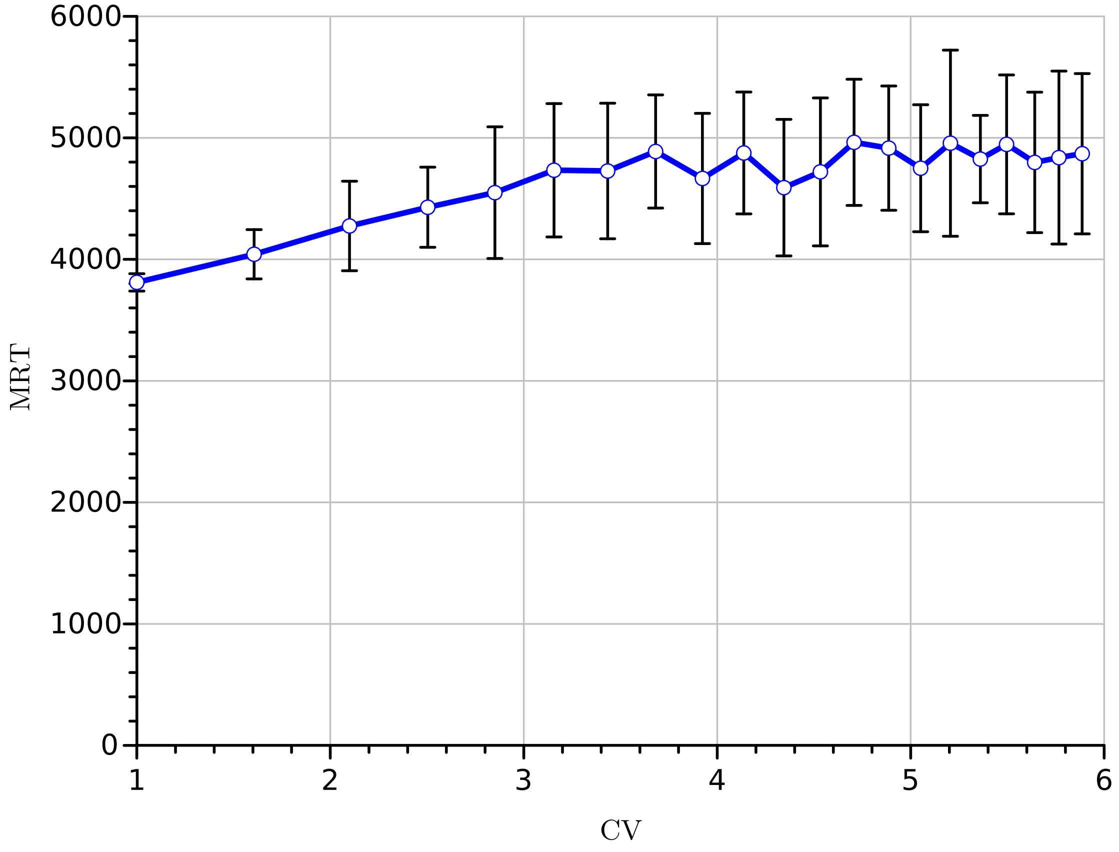


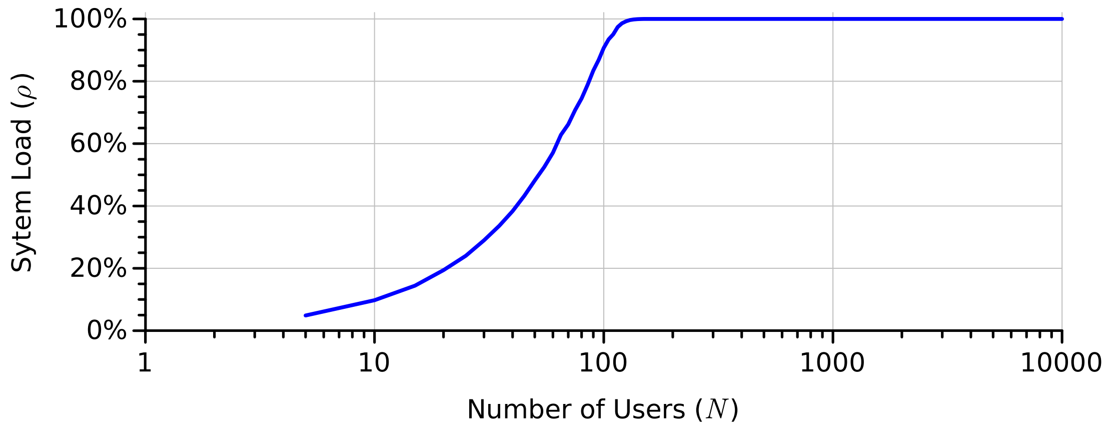
Figure 4.6: The Effect of CV on MRT.
$K = 5$, $M = 10{,}000$, $M_0 = 1{,}000$, $\gamma_1 = \gamma_2 = 200{,}000$,
$\rho \approx 87\,\%$ ($K = 5$, $N = 500$, $z = 35{,}000$, Web Page Category Base Scenario)

4.2.4 System Load

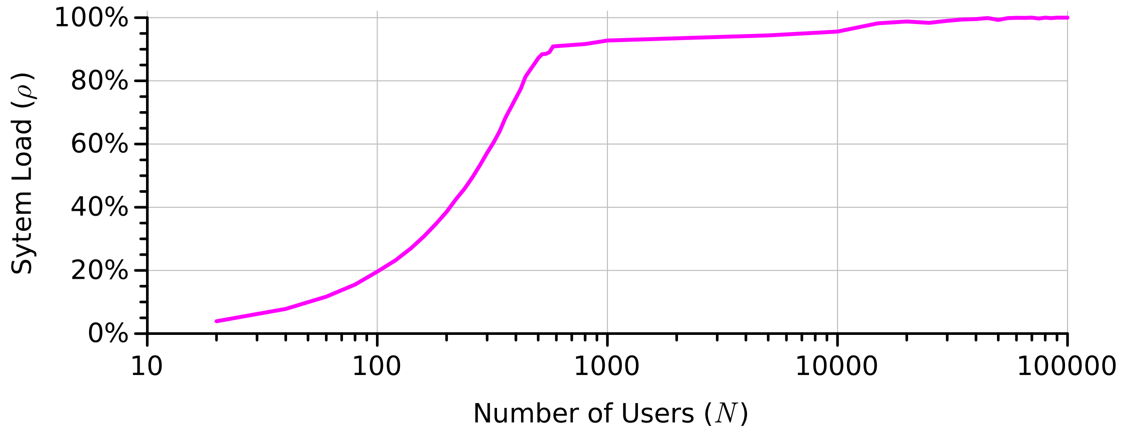
We define system load (ρ) to be the mean observed utilization of all servers over the duration of the simulation. It is influenced by several parameters, most notably: number of users (N), mean Web page size for all requests ($\overline{W}$), number of servers (K), and coefficient of variation (CV).

In Figure 4.7, we examine how the system load varies with the number of users and the number of servers (where $K = \{1, 5, 10\}$), with a constant Mean Web Page Size for All Requests (determined by Web Page Category Base Scenario). We see that as the K increases, N must increase to achieve equivalent ρ, and for all servers system load increases with N. In all three Web server scenarios, there is a distinct leveling off at higher system loads, where the system load transitions from a high rate of change to a lower one until $\rho = 100\,\%$. These results generally follow expected ones [28].

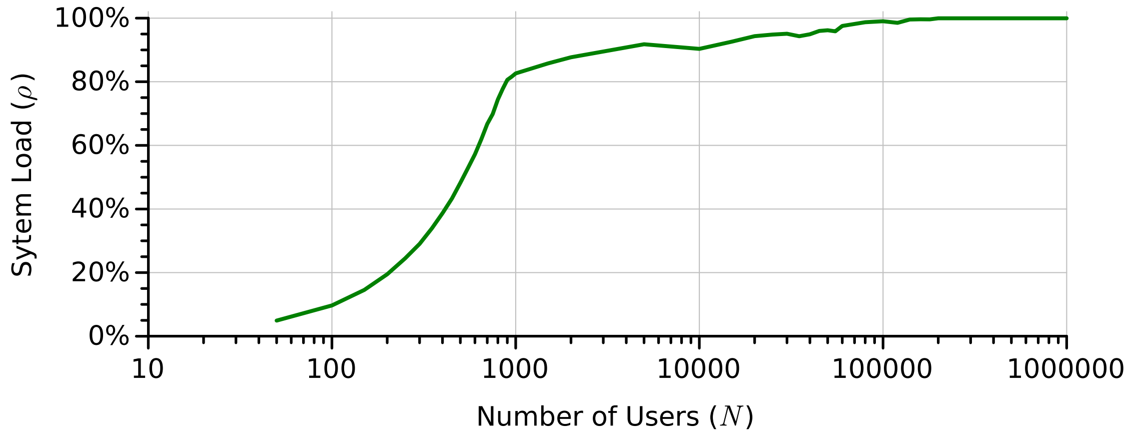
Figure 4.8 highlights system load versus the number of users and is shown in Figure 4.7 for lower values of N. We can see that for all three servers the rate of system load increases nearly linearly until the change points mentioned above, at which point the change in system load transitions to a considerably lower rate. As well as a reduced rate of system load, we can see a notable amount of variation (this was not as evident in Figure 4.7 due to different sample periods). The results agree with expected results [28].



(a) $K = 1$



(b) $K = 5$



(c) $K = 10$

Figure 4.7: Varying System Load with Number of Users.
$M = 10{,}000$, $M_0 = 1{,}000$, CV ≈ 3, Web Page Category Base Scenario.

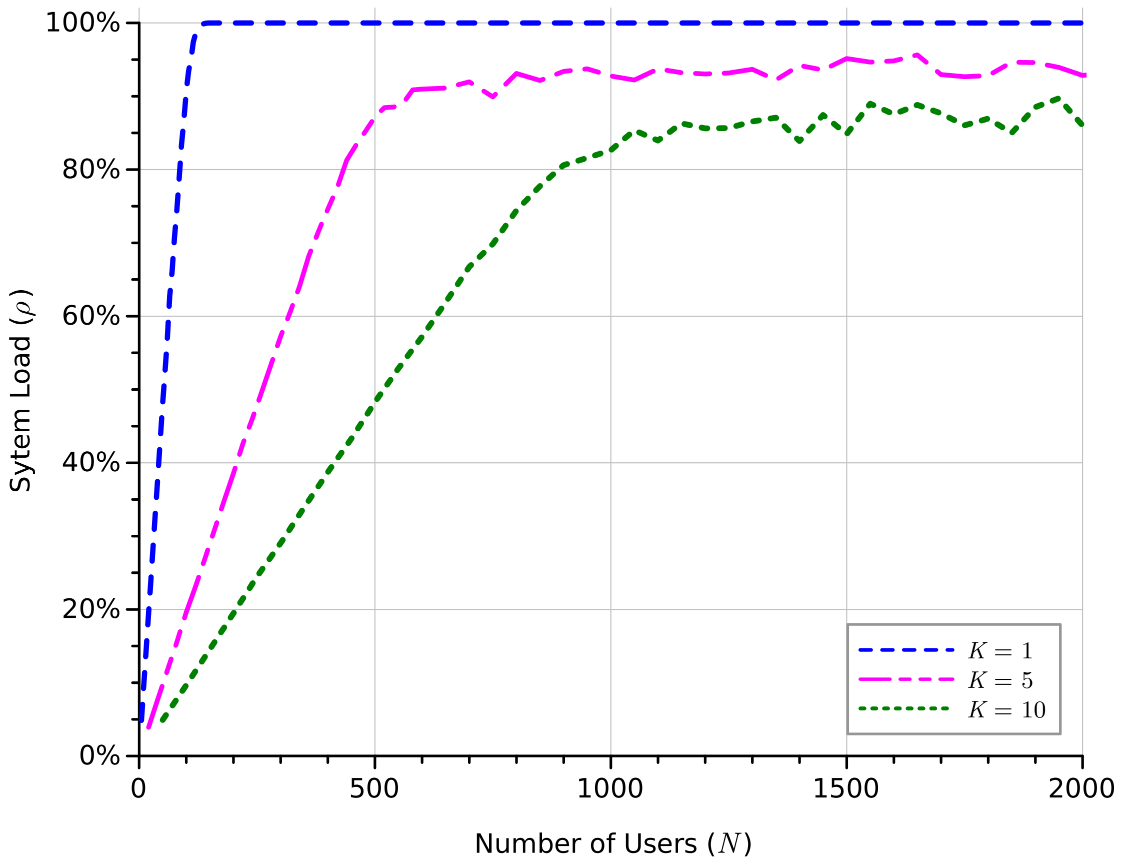


Figure 4.8: Varying System Load with Number of Users, Highlighting Results for $N < 2000$.
$M = 10{,}000$, $M_0 = 1{,}000$, CV ≈ 3, Web Page Category Base Scenario.

We continue our investigation of system load in Figure 4.9, where we observe the effect of varying the number of users on MRT. Lower values of N have minimal effect on MRT. However, as N continues to increase, the effect on MRT transitions to a higher rate of change. This transition corresponds to approximately the same points where the change in slope levels off in Figure 4.7. After this, the rate of change in MRT increases linearly until system load approaches 100 %. It is evident in Figure 4.9 that increasing K reduces the MRT.

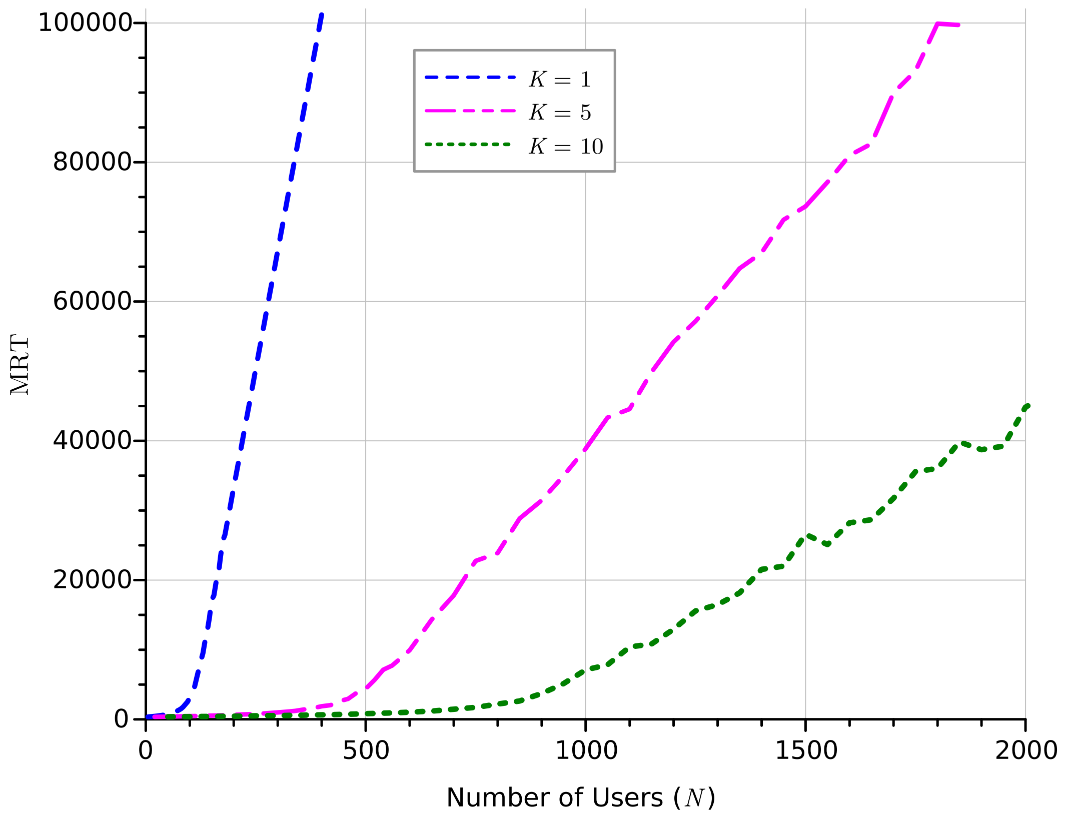


Figure 4.9: Effect of Number of Users on MRT.
$M = 10{,}000$, $M_0 = 1{,}000$, $K = 5$, CV ≈ 3, Web Page Category Base Scenario.

4.3 Varying Web Page Category

As was mentioned in Section 4.2, our analysis of Web page categories is a key feature of our model as it characterizes the effect that different compositions of Web pages have on MRT. In this section, we present results of varying the *Ratio of Web Pages per Web Page Category* on MRT. As well, we develop two additional Web page category scenarios (in addition to our Web Page Category Base Scenario) to use in experiments in Chapter 5.

Table 4.4 shows the thirty-six *Web Page Category Ratio Test Scenarios* that we have developed, which are made up of permutations of Ratio of Web Pages per Web Page Category in increments of 10 %, starting at 10 %. Figure 4.10 presents a graph of the results of the various test scenarios on Mean Web Page Size for All Requests ($\overline{W}$). We can see a general trend where, with low ratios of Article Web Pages, $\overline{W}$ tends to be high, being dominated by the larger Mosaic and Media Web Pages. Conversely, with higher ratios of Article Web Pages, Mosaic and Media Web Pages tend to contribute less to $\overline{W}$. This makes sense since the relative difference in Web page size has Mosaic and Media Web Pages 4 and 8 times larger than Article Web Pages.

The Mean Web Page Size for our Base Scenario is shown in Figure 4.10, and has a value of 344.7. This puts it roughly in the middle of the minimum and maximum Mean Web Page Sizes (158 and 554). This confirms our goal to have the Mean Web Page Size for our Base Scenario be near the middle range of $\overline{W}$. As well, using Equation (4.1), we can compare this value to the Expected Mean Web Page Size for All Requests of $E\left(\overline{W}\right) = 344$. Since the mean value nearly matches our measured value, it helps to verify our implementation of Web page categories.

Scenario	k_{article}	k_{mosaic}	k_{media}
1	10	10	80
2	10	20	70
3	10	30	60
4	10	40	50
5	10	50	40
6	10	60	30
7	10	70	20
8	10	80	10
9	20	10	70
10	20	20	60
11	20	30	50
12	20	40	40
13	20	50	30
14	20	60	20
15	20	70	10
16	30	10	60
17	30	20	50
18	30	30	40
19	30	40	30
20	30	50	20
21	30	60	10
22	40	10	50
23	40	20	40
24	40	30	30
25	40	40	20
26	40	50	10
27	50	10	40
28	50	20	30
29	50	30	20
30	50	40	10
31	60	10	30
32	60	20	20
33	60	30	10
34	70	10	20
35	70	20	10
36	80	10	10

Table 4.4: Web Page Category Ratio Test Scenarios.

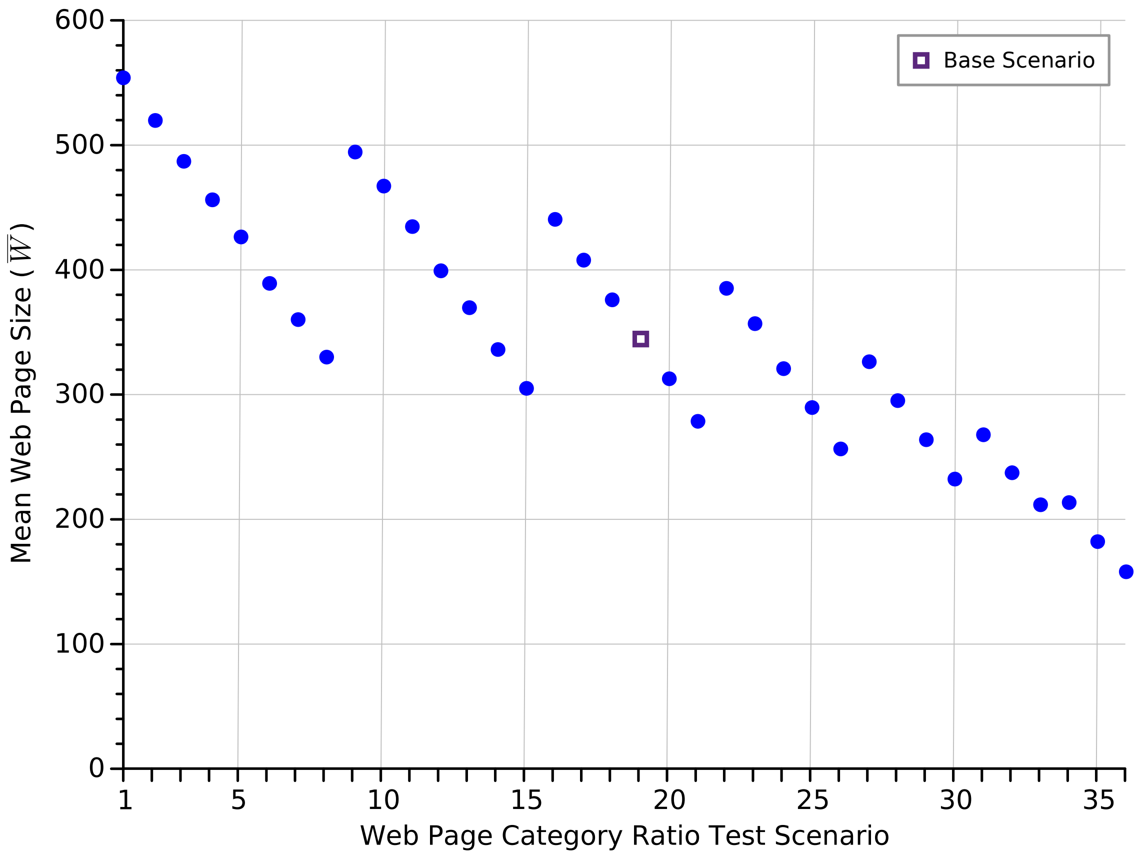


Figure 4.10: Effect of Varying the Ratio of Web Pages per Web Page Category has on the Mean Web Page Size for All Requests.

Next we examine how the Mean Web Page Size affects MRT and system load. These results are shown in Figure 4.11. We can observe that MRT increases non-linearly as $\overline{W}$ increases, and at higher values of $\overline{W}$, ρ flattens out around $\rho = 90\,\%$. This is inline with the results we saw in Section 4.2.4. The system load for our Base Scenario is 85.3 %. This is not a coincidence, as we have chosen parameters so that Base Scenario results in $\rho \approx 85\,\%$.

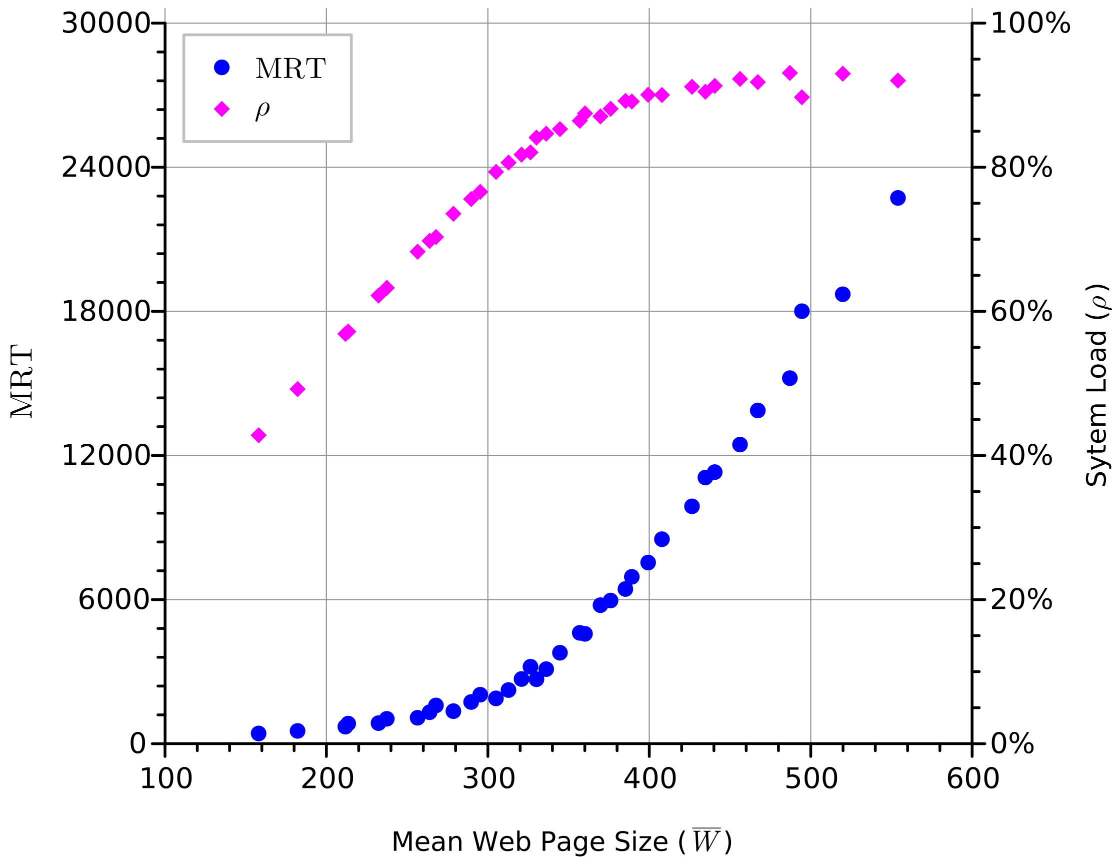


Figure 4.11: The Effect that Mean Web Page Size has on MRT and System Load. $M = 10{,}000$, $M_0 = 1{,}000$, CV ≈ 3, $K = 5$, $N = 480$, $z = 35{,}000$

Of particular interest for the experiments in Chapter 5, is that we can now choose three Final Web Page Category Ratio Test Scenarios to represent the range of Web page category permutations. Along with our Base Scenario, we have added Low and High scenarios. These three Test Scenarios are summarized in Table 4.5.

We can see the effect that these Test Scenarios have on MRT in Figure 4.12, in which MRT increases as $\overline{W}$ increases. This makes sense since $\overline{W}$ is proportional to request service time. The relationship between $\overline{W}$ and MRT is non-linear, but we can see that the differences between the values of the MRT and the values of $\overline{W}$ are pronounced. One of our goals in Section 4.2.1 was that each Web page category be different enough to impact MRT. We assume that these differences meet this goal.

Scenario	k_{article}	k_{mosaic}	k_{media}	$E\left(\overline{W}\right)$	Measured $\overline{W}$
Low	70	10	20	216	213.5
Base	30	40	30	344	344.7
High	10	30	60	488	487.1

Table 4.5: Summary of Final Web Page Category Ratio Test Scenario.

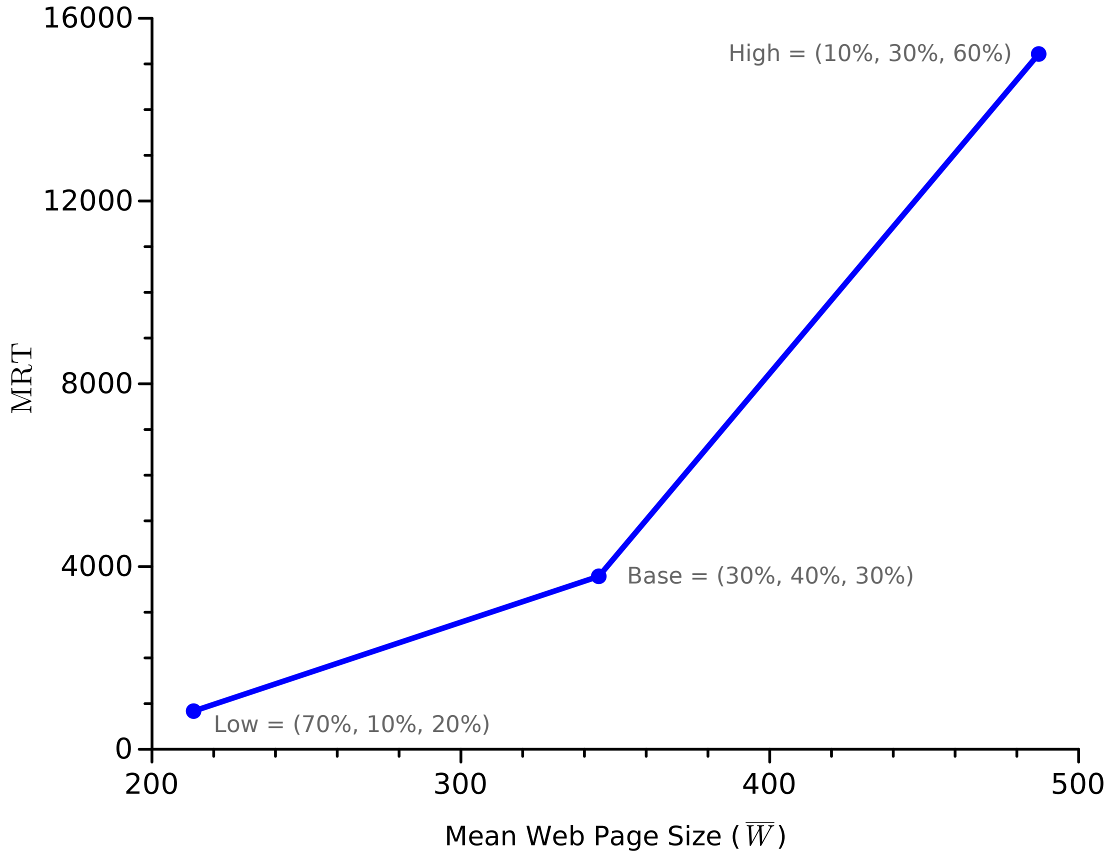


Figure 4.12: The Effect of Our Final Web Page Category Ratio Test Scenario on MRT.

$K = 5$, $N =$, $z = 35{,}000$, CV ≈ 3, $\rho_{\text{Low}} \approx 57\,\%$, $\rho_{\text{Base}} \approx 85\,\%$, $\rho_{\text{High}} \approx 93\,\%$

4.4 Summary

In this chapter, we implemented our Web Page Request Access Pattern Model using DES, verified it using manual trace analysis, and validated our simulation using analytical results. We established and evaluated system parameters, which included parameters that determine the composition of the Web page set, CV, and the system load. We demonstrated that there is a relationship between MRT and the composition of the Web Page Set by examining the effect that Web page categories have on MRT for various Ratios of Web Pages per Web Page Category. Based on those results, we developed three Web Page Category Ratio Test Scenarios (Base, Low, and High) that we will use for experiments with Web Caching in Chapter 5.

Chapter 5

Web Caching Model and Results

5.1 Introduction

Having developed our Web Page Request Access Pattern Model in Chapter 3, and examined its baseline performance in Chapter 4, we now apply it by incorporating it into an application that involves Web access. The application we chose is Web caching which was discussed in Chapter 2. Web caching is a system that is intended to improve user experience by reducing response times, improving availability, lowering server load, and reducing bandwidth. Implementing a Web caching system is a complex and expensive endeavor, but with our model, one can evaluate various Web caching strategies with relatively little physical investment.

5.2 Model

We have adapted our model by replacing the Web server sub-model with an expanded version that includes object-level Web caching, the results of which are shown in Figure 5.1. The expanded server model (Web caching) consists of a homogeneous set of C Web caches, where each cache 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.

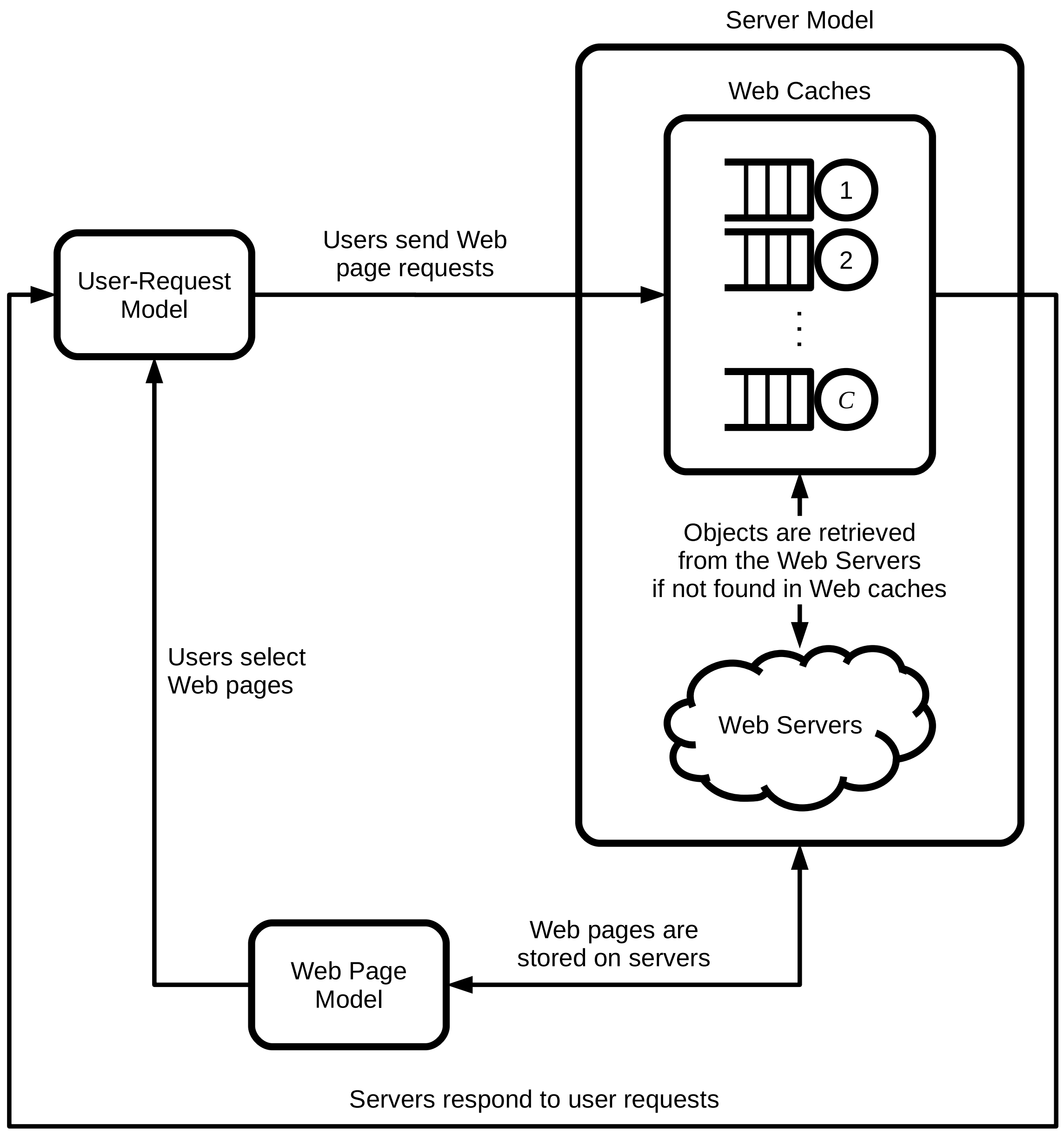

Server Model
Web Caches
1
2
C
User-Request Model
Users send Web page requests
Objects are retrieved from the Web Servers if not found in Web caches
Web Servers
Users select Web pages
Web pages are stored on servers
Web Page Model
Servers respond to user requests

Figure 5.1: Web Page Request Model with Caching Model.

This expanded server model is based on the assumption that an object that is cached has a transmission time that is smaller than if it were retrieved from the origin server. We introduce a new parameter, the *Cached Object Access Factor* (δ), where $\delta \leq 1$, that allows us to modify the transmission latency for cached objects.

Recall from Chapter 3, Equation (3.3), that request service time (R_i) is proportional to the Web page transmission time (τ_i). This, in turn, is proportional to the Web page size. Equation (5.1) expands on Equation (3.2) to include the Cached Object Access Factor for objects that are cached. For R_i, Equation (3.3) remains as it was defined in Chapter 3.

$$\tau_i = \sum_{j=1}^{Q_i} \begin{cases} \Theta_{ij}, & \text{if object is in not cache,} \\ \delta\Theta_{ij}, & \text{if object is in cache.} \end{cases} \tag{5.1}$$

In the WWW, the cacheability of objects can vary greatly, from being easily cacheable to not cacheable at all, which is influenced by a myriad of considerations [1,23,26,36,40]. We capture this behavior of cacheability with a parameter, the *Web page object cacheability* (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. Our system 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 cacheabilities levels: *not cacheable* (φ_{not}), *less cacheable* (φ_{less}), and *more cacheable* (φ_{more}). We have expanded Table 3.1 by adding the Web page object cacheabilities (see Table 5.1).

Based on observations in [26], we assume the values of our object cacheabilities to be as follows: $\varphi_{\text{not}} = 0\,\%$, $\varphi_{\text{less}} = 25\,\%$, and $\varphi_{\text{more}} = 75\,\%$. Also, 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. This implies that 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. Conversely, there is a 75 % chance of caching the object if it is in the *more cacheable* category.

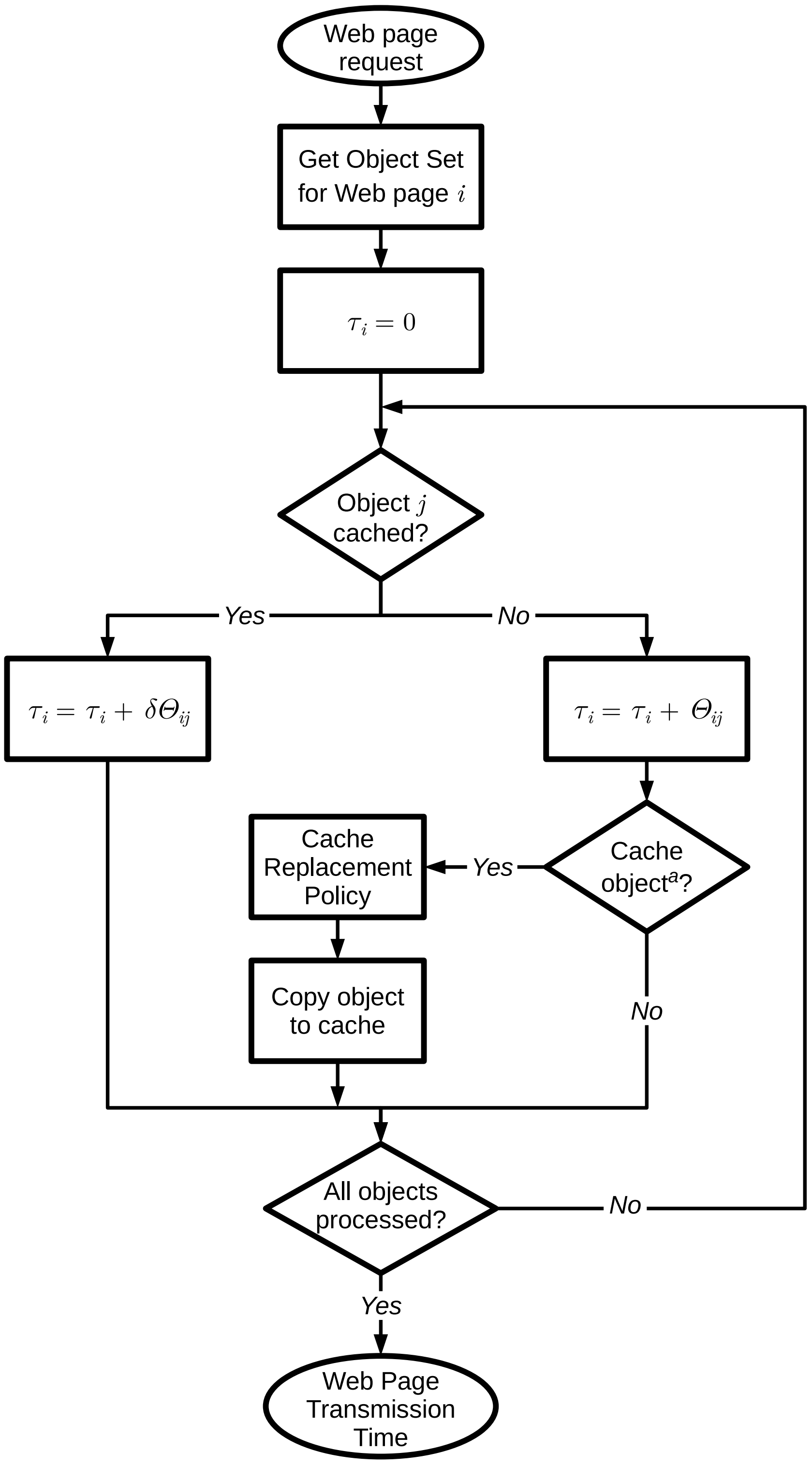
Our Web cache servers are assumed to have a *Cache Capacity* which is represented

Object Type	Size	Cacheability	Examples
text	small	not cacheable	HTML, XML, text
script	small, medium	less cacheable	JavaScript
css	small	more cacheable	CSS
image	small, medium	more cacheable	.png, .gif, jpg
audio	small, medium	more cacheable	.ogg, .mp3, .wmv
video	medium, large	more cacheable	.flv, .mov

Table 5.1: Example Web Page Object Parameters

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 removed to make room for an incoming object. For our cache replacement policy, we chose to implement an LRU algorithm [1]. 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.

We summarize our Web caching process in Figure 5.2, which starts with a request for Web page i and ends with its transmission time (τ_i). This process examines the cache status of all the objects in the Web page. For objects that have been cached, the Web page transmission time includes the Cached Object Access Factor (δ) along with the sizes (Θ_{ij}). Otherwise, the size alone is added to the Web page transmission time. 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.



[a] Accoding to Web page object cacheability (φ)

Figure 5.2: Process to Calculate Web Page Transmission Time with Object Level Web Caching.

5.3 Model Parameters

In Chapter 4, we explored the parameters from our base model. Many of these parameters still apply to our expanded server model. For example, since CV is a function of the Web Page Selection Model (which we have not altered), we assume the results from Chapter 4 will still apply to this model. However, some other parameters need to be reexamined, such as system stability and system load. With the introduction of new parameters, we need to look at their effect as well. This includes an examination of the effect of Cache Capacity (κ) and Cached Object Access Factor (δ).

It is interesting to note that, with the additional complexity that Web caching has added to our model, we have found that the run-time performance of the simulation is much slower than that of the base simulation. This can, in large part, be attributed to the added complexity of managing the cached content of the Web cache servers. For example, each object's last access time must be maintained and ordered so that the oldest objects can be removed by the LRU algorithm.

5.3.1 Composition of the Web Page Set

The parameters that determine the composition of the Web page set, which we established in Chapter 4.2.1, will remain constant in our expanded server model. In particular, this includes the *Composition of the Web Page Object Set* defined in Table 4.2. As such, the number and size of the objects remain unchanged. We will continue to consider the small object size ($\Theta_{\text{small}} = 1$) to be the base unit for all measures of data size in our model. In Table 5.2 we have expanded Table 4.2 to include Web Page Object Cacheability (φ).

At the end of Chapter 4, we explored how varying the ratio of Web pages per Web page category affects the mean Web page size for all requests ($\overline{W}$), and consequently

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

Table 5.2: Composition of the Web Page Object Set with Web Caching.

the MRT. Later in this chapter, we will re-examine the effect of varying Ratios of Web Pages per Category in this new environment of Web caching using the test scenarios: *Base*, *Low*, and *High*. The Web Page Category Base Scenario that we defined in Chapter 4 will continue to be used in this chapter. As a reminder, for the Web Page Category Base Scenario we chose the Ratios of Web Pages per Category to be: $k_{\text{article}} = 30\,\%$, $k_{\text{mosaic}} = 40\,\%$, and $k_{\text{media}} = 30\,\%$. This ratio resulted in $\overline{W} \approx 344$ and $\rho \approx 85\,\%$ with $K = 5$. Thus, for consistency, we will continue to use the Web Page Category Base Scenario with $C = 5$ when examining parameters in our expanded server model.

5.3.2 System Stability

To determine simulation stability in Chapter 4, we considered the effect that simulation length had on CV. For the Web caching environment, we examined the effect of simulation length on the cache hit rate (HR) as well as CV. HR refers to the ratio of the number of Web page objects retrieved from a cache to the total number requested. Figure 5.3 shows the HR and CV for Web caching assuming an infinite Cache Capacity (κ), a Cached Object Access Factor (δ) of 0.6, and $C = 5$. We observe that while CV converges quickly as it did without Web caching, it takes much longer for HR to reach equilibrium. This longer time is due the time it takes for the cacheable objects in the system to fill the caches. There is an initial increase in HR, but after around 50×10^6 the change is slight, only to meet the expected HR at approximately 140×10^6. Thus, we assume that the system is stable at 150×10^6, which is the simulation length that we will use for all of our experiments with Web caching.

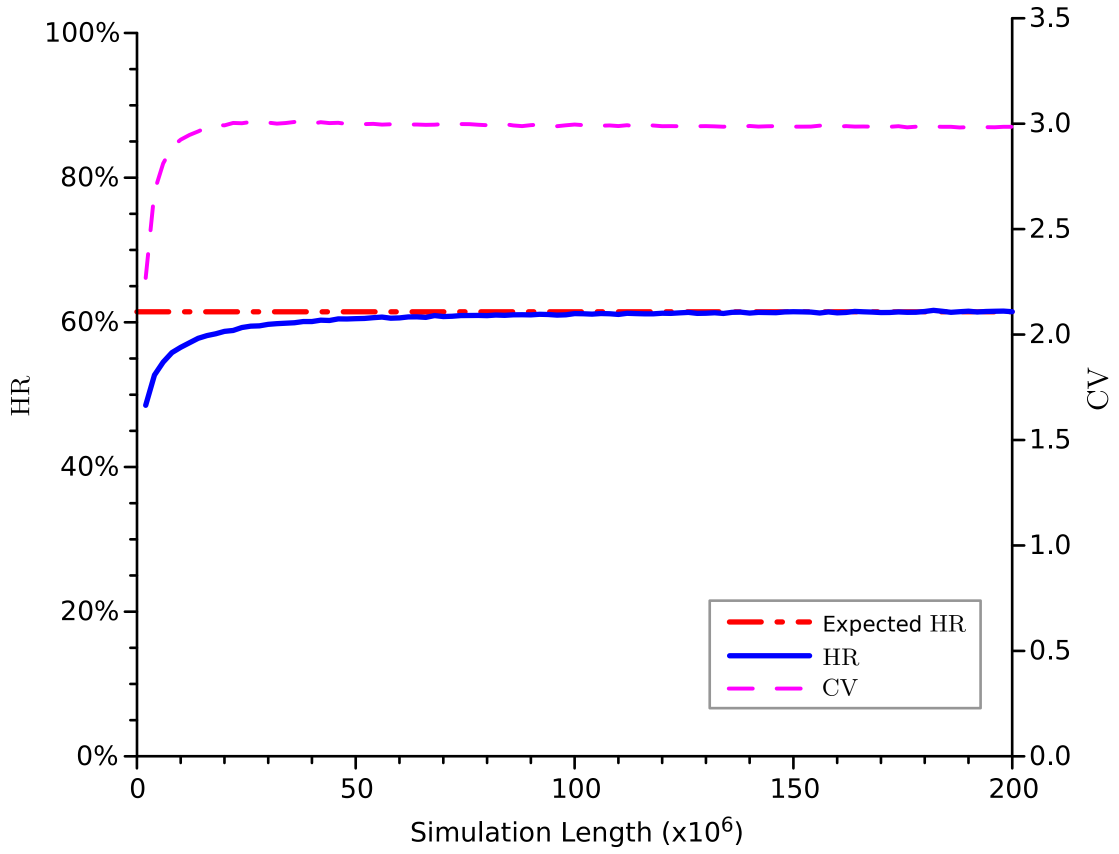


Figure 5.3: Effect of HR on System Stability.
$\delta = 0.6$, $\kappa = \infty$, $\delta = 0.6$, $\rho \approx 90\,\%$ ($C = 5$, $N = 770$, $z = 35{,}000$, Web Page Category Base Scenario)

5.3.3 Cached Object Access Factor

The *Cached Object Access Factor* (δ) represents the reduction by which the transmission time of a cached object is improved. We restrict it to a practical range of $\delta \leq 1$ (although theoretically δ can be greater than 1, this would represent a penalty for Web caching.) As δ decreases we expect that MRT will decrease because we are reducing the total time to satisfy each cached Web page request. We do observe this effect in Figure 5.4 where we examine the effects of δ on MRT and system load (ρ). We also see that system load (ρ) decreases as δ decreases. It is interesting to note that, when $\delta = 1.0$, the system load is approximately 85 % and MRT is approximately 3,878. This is consistent with the results with no Web caching that were shown in Figure 4.12 (for the Web Page Category Base Scenario when $\rho \approx 85\,\%$).

As we continue our experiments with our expanded server model, we will focus on one value for δ. In [29] they found that, at best, caching and prefetching together reduced total latency by 60 %, and so, we will use $\delta = 0.6$.

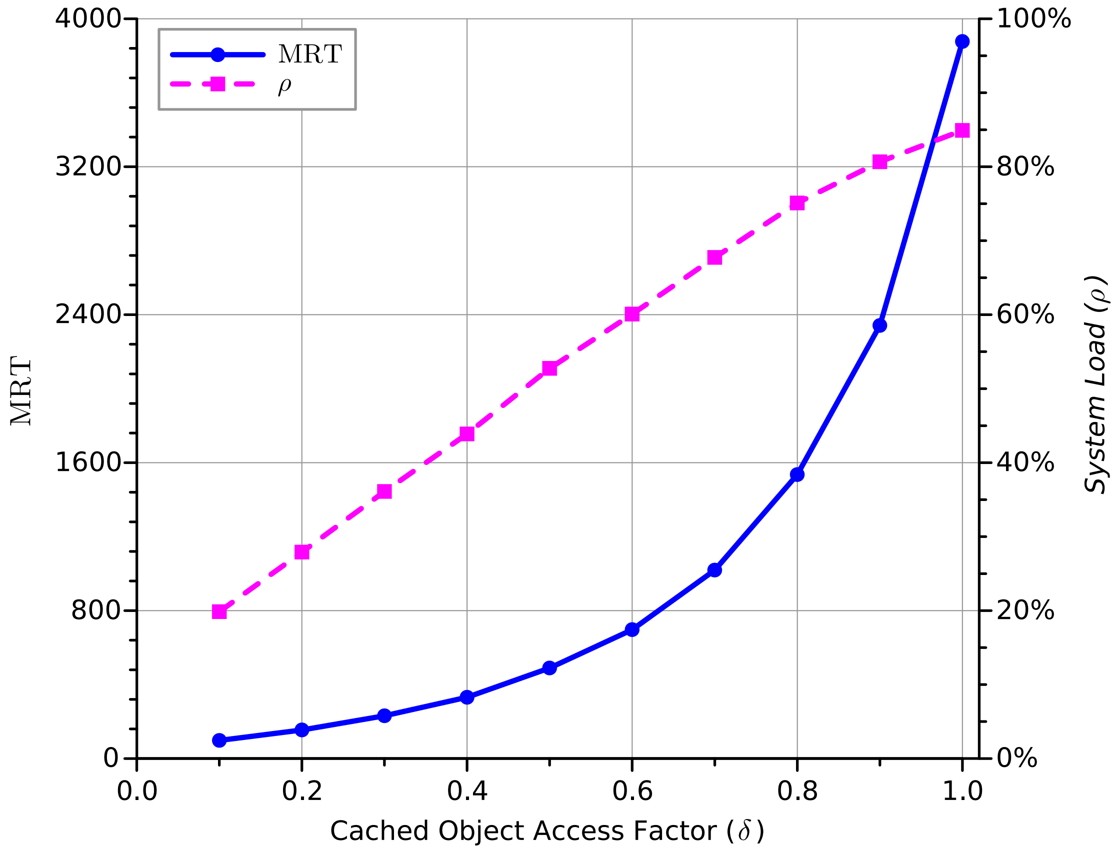


Figure 5.4: Effect of Cached Object Access Factor on MRT.
$C = 5$, $\kappa = \infty$, HR $\approx 60\,\%$, $N = 480$, $z = 35{,}000$,
Web Page Category Base Scenario

5.3.4 System Load

Due to the substantial increase in simulation run-time for our expanded server model, we were unable to get a comprehensive data set for system load as we did in Section 4.2.4. Instead we focus on the following three load scenarios: low $\approx 50\,\%$, medium $\approx 85\,\%$, and high $\approx 100\,\%$, the results of which are shown in Figure 5.5, where we plot number of users (N) against system load (ρ) for three values of the number of Web caches (C).

When compared to the results in Section 4.2.4, we can see that system load follows the same general pattern. As the number of users increases, the system load increases quickly until shortly after 85 %, when the rate of change of system load decreases slowly as it approaches 100 %. With our expanded server model, however, the number of required users is higher for equivalent system loads without caching. In the case of $\rho \approx 85\,\%$ with $C = 5$ cache servers, the number of users is 720, as opposed to with no caching, which is 480. As we saw in the previous section Web caching reduces the system load, so it make sense that the number of users is greater to maintain equivalent system loads. For the remainder of our experiments in this chapter, we will use $C = 5$ and $N = 720$ to aim for an approximate system load of 85 %.

5.3.5 Cache Capacity

Next we examine the *Cache Capacity* parameter (κ), which is the maximum amount of storage space that a Web cache server has available to cache objects. The units for capacity are in terms of small object size ($\Theta_{\text{small}} = 1$). If the amount of data currently stored in a cache is 100 % of the Cache Capacity then it will not accept more objects. Cache capacity has a range of $[0, \infty)$ (where 0 means that Web caching does not occur, and ∞ means that everything is cached).

In addition to expressing Cache Capacity in terms of small object size, it is useful to compare it to the maximum space required to cache all Web pages in the system. This maximum storage space is a function of the composition of the Web page set (defined in Table 5.2), the number of Web pages, and the number of Web caches. Given $M = 10{,}000$ and $C = 5$, we expect that the maximum cache size for Web Page Category Base Scenario is approximately 614,000. This value applies to experiments

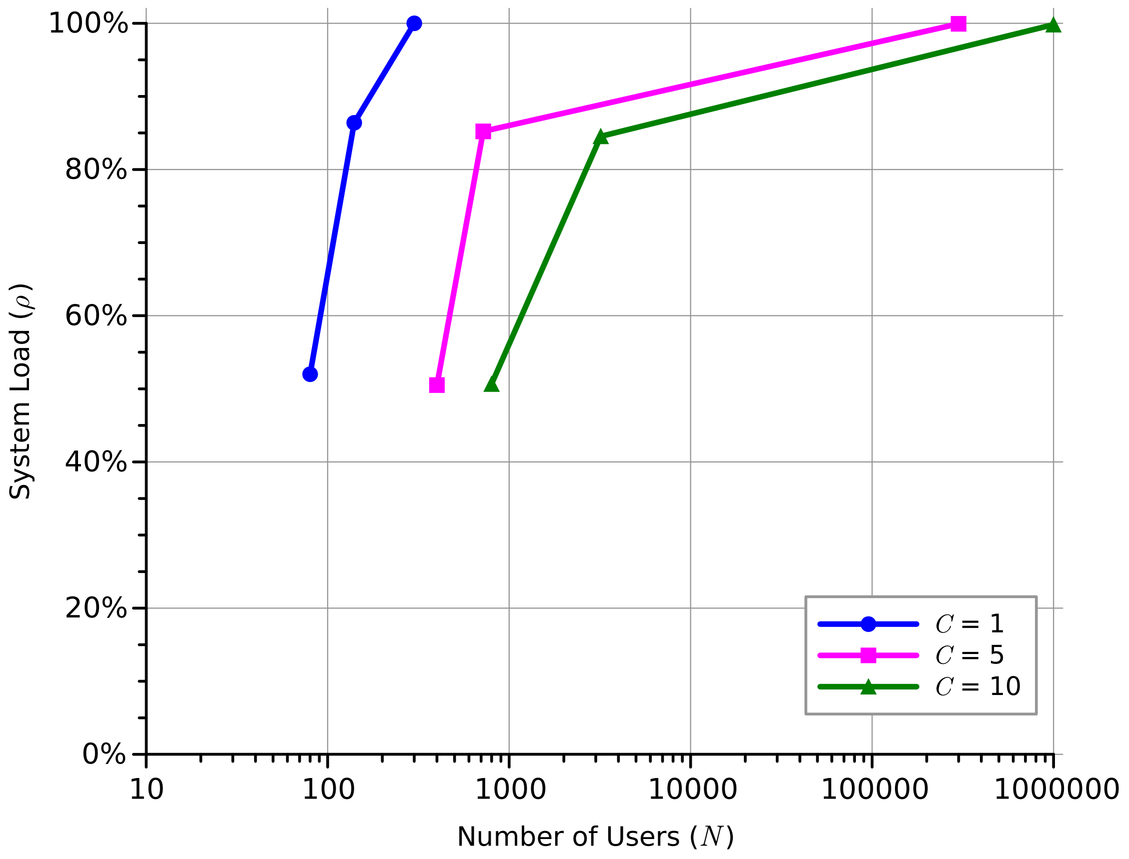


Figure 5.5: The Effect of the Number of Users on System Load.
$\delta = 0.6$, $\kappa = \infty$, HR $\approx 60\,\%$, $z = 35{,}000$, Web Page Category Base Scenario.

in this chapter using the Web Page Category Base Scenario, where $\kappa = \infty$.

Figure 5.6 shows the results of our experiment intended to investigate the effect that varying κ has on HR and SHR. *Cache size hit rate* (SHR) is the ratio of the size of data retrieved from cache to the total size of the requested data. Overall, the HR increases as cache capacity increases, since, as the cache capacity increases more objects are cached and thus are more likely to be found in cache. As we increase κ, HR is low but increases quickly until between approximately 7 %, where at lower cache capacities objects are swapped out frequently which make it less likely for an object to be found in cache. However, as the cache capacity increases, fewer objects have to be removed so change the rate of change of HR slows down. After 20 % HR continues increasing gradually until κ reaches 100 %. Once κ reaches 100 %, HR no longer changes, since all of the data that can be cached has been cached. In addition, SHR follows similar pattern as HR, but has a higher magnitude (HR is a ratio of discrete values, whereas SHR compares quantities, and thus will have higher magnitudes that HR).

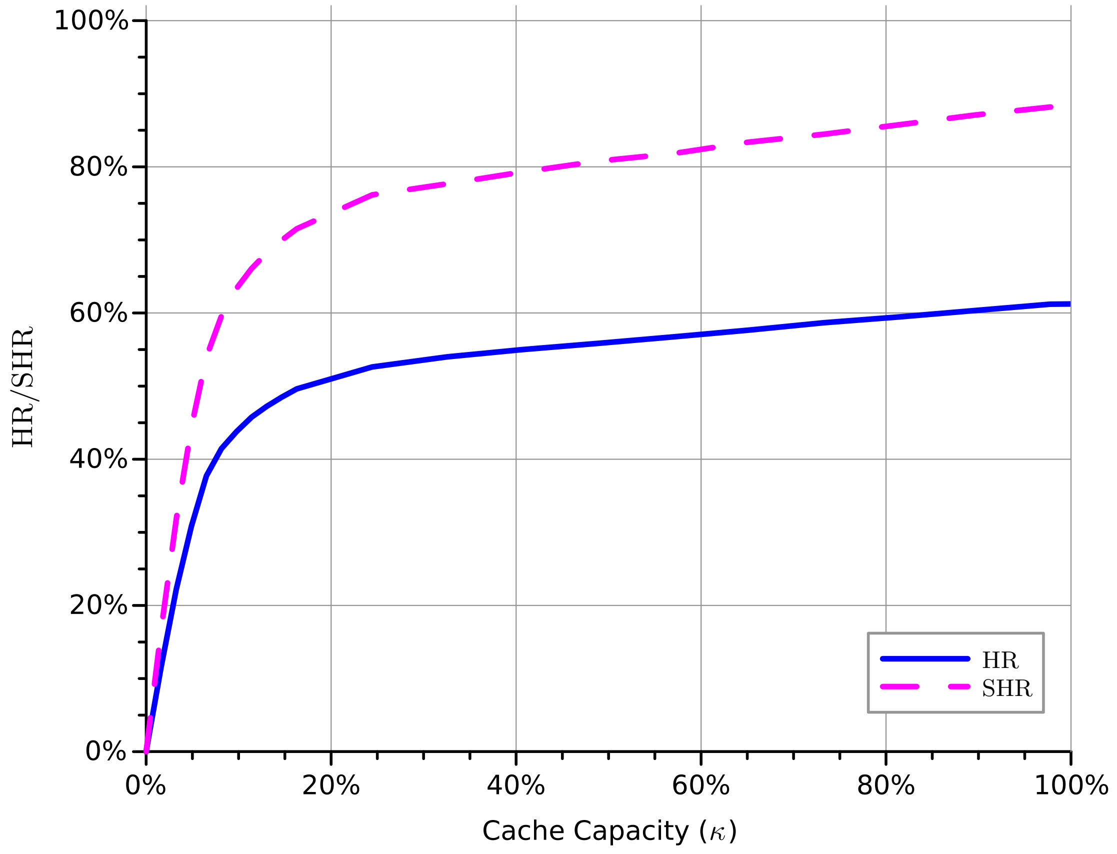


Figure 5.6: The Effect of Cache Capacity on HR.
$\delta = 0.6$, $\rho \approx 85.2\,\% \ldots 92.4\,\%$ ($C = 5$, $N = 720$, $z = 35{,}000$,
Web Page Category Base Scenario)

From Figure 5.6, we see that the HR approaches 62 % (when κ is at 100 %). This is the maximum HR for our Web Page Category Base Scenario which is the composition of the Web page set. The observed HR for each Web page category is detailed in Table 5.3 with respect to the expected HR. The expected HR is the ratio of cacheable objects per category to the total number of objects per Web page category (from Table 5.2). We can see that the observed values are very close to expected ones.

Category	Expected HR (%)	Observed HR (%)
Article	68.8	68.2
Mosaic	56.5	56.2
Media	68.3	67.7

Table 5.3: Observed HR Compared with Expected HR by Web Page Category

In Figure 5.7, we examine the effects of cache capacity (κ) on system load (ρ), we observe that ρ decreases as κ increases. This makes sense, since for a constant number of users, a smaller κ will increase transmission time which is expected to increase wait times and thus system load. When κ is at 100 %, the system load corresponds to the results described in Section 5.3.4 for $C = 5$, where the system load was approximately 85 % with $\kappa = \infty$.

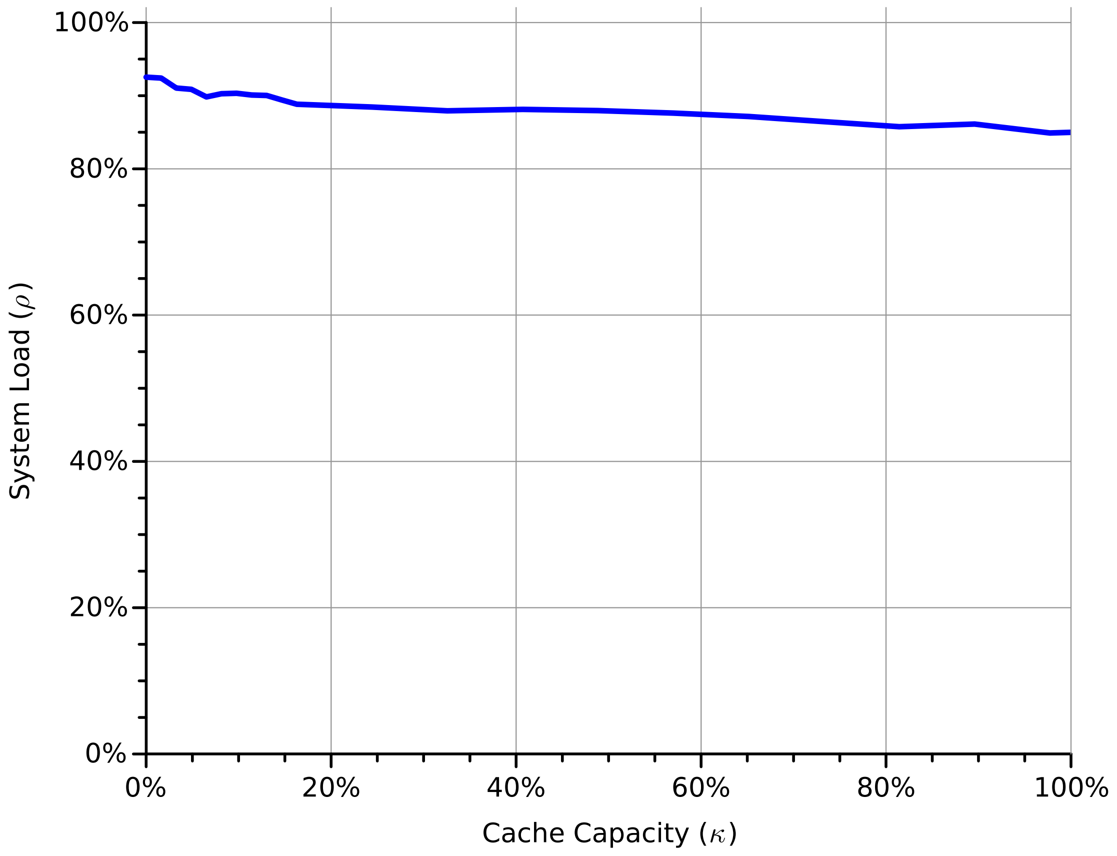


Figure 5.7: The Effect of Cache Capacity on HR.
$\delta = 0.6$, $C = 5$, $N = 720$, $z = 35{,}000$, Web Page Category Base Scenario

5.4 Varying Web Page Category with Web Caching

In this section, we continue our exploration of the impact of the ratio of Web pages per category that we began in Chapter 4.3. That experiment explored 36 Web page category ratio test scenarios covering a wide range of mean Web page sizes for all requests ($\overline{W}$). We selected three target ratios (*Low*, *Base*, *High*) to be representative of the range of $\overline{W}$ (for brevity, we refer to these scenarios as *Ratio Scenarios*). For this experiment, we have incorporated Web caching to our previous experiment (from Chapter 4.3).

In addition to our Ratio Scenarios, we consider four Web caching scenarios: *Small*, *Medium*, and *Large* Cache Capacities (κ), and a *No Caching* scenario for reference. The values we chose for the three Cache Capacities (κ) correspond to the points in Figure 5.6 where the HR was approximately 20 %, 40 % and 60 %, which correspond to maximum cache sizes of about 3 %, 7 %, and 80 % respectively.

In Figure 5.8 presents the results for Ratio Scenarios versus MRT for different cache capacities (κ). In general, we observe that MRT increases as the $\overline{W}$ increases, which is consistent with our results from Chapter 4.3. This is not surprising since $\overline{W}$ is proportional to request service time. For the Low Scenario, there is less MRT variation between the three cache capacities. However, for the Base Scenario and the High Scenario, the variation in MRT between cache capacities is more pronounced. This is related to the larger SHR for the Base and High scenarios with respect to the Low Scenario.

Overall from Figure 5.8, we can see that the MRT is improved through Web caching, and the improvement is more pronounced as the $\overline{W}$ increases or κ increases. It makes sense that as κ increases, MRT decreases since more cache size means more cache hits, thus reduced 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.

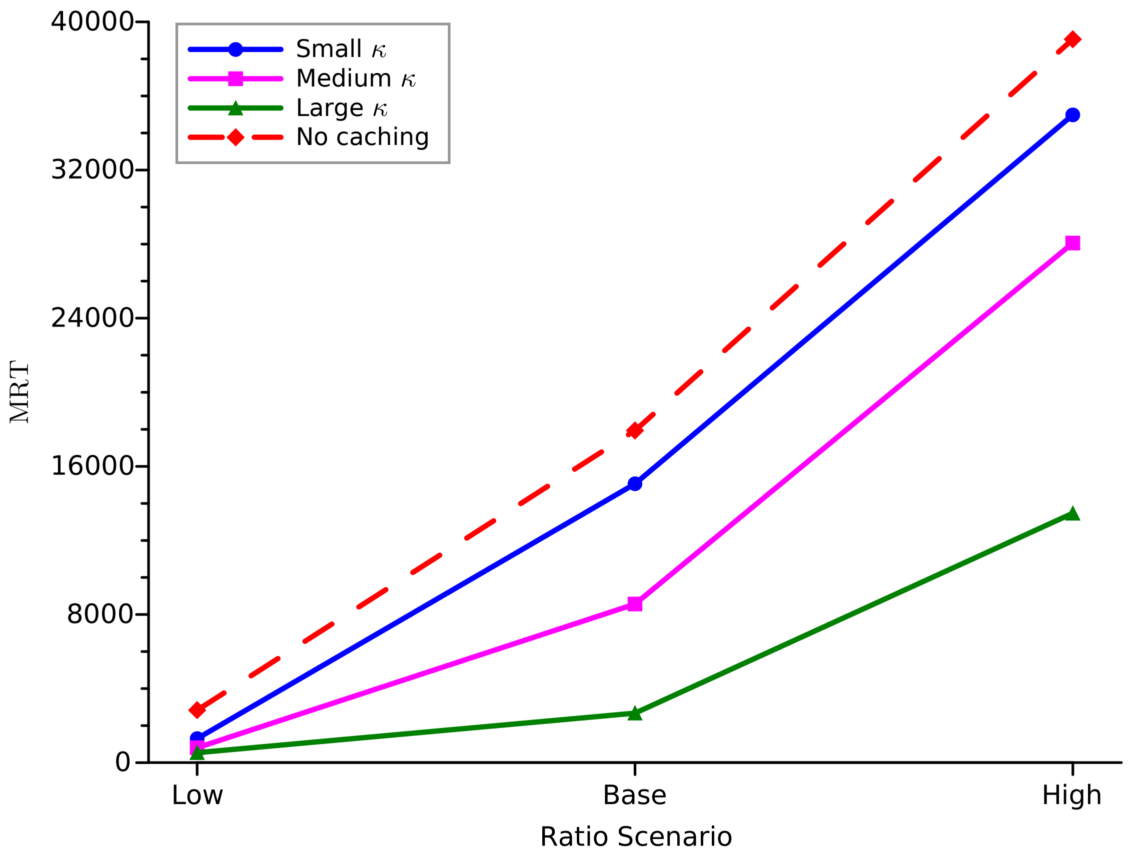


Figure 5.8: Effect of Ratio Scenarios on MRT.
$\delta = 0.6$, $C = 5$, $N = 720$, $z = 35{,}000$

The effects on system load versus our Ratio Scenarios and our four Web caching scenarios is shown in Figure 5.9. These results plot κ with respect to ρ to highlight how system load relates to caching. We observe that as $\overline{W}$ increases, ρ increases. This is because as request service time increases (from an increase in Web page size), system load would also expect to increase. As κ increases, we see a decrease in system load with the Low Scenario and a slight decrease with the Base Scenario. It makes sense that smaller Web pages (and thus lower request service times) being cached more frequently will result in a lower system load. There is little change for the High Scenario, possibly due to the system load being large enough that it leveled off (as was observed in Section 4.2.4). 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.

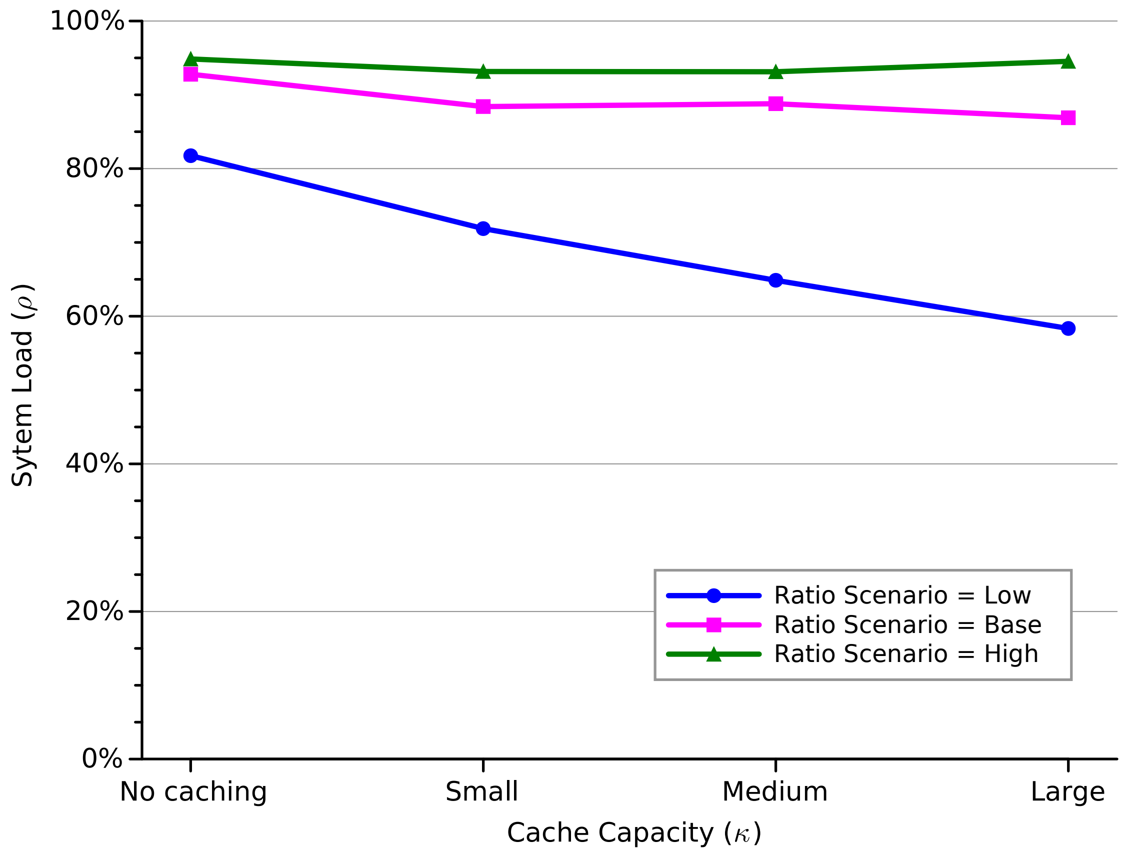


Figure 5.9: Effect of Varying Cache Capacity on System Load.
$\delta = 0.6$, $C = 5$, $N = 720$, $z = 35{,}000$

Finally, in Figure 5.10, we show HR and SHR for the three chosen cache capacities (κ). In each case, HR and SHR tend to decrease with larger $\overline{W}$. This makes sense because, as $\overline{W}$ increases, the ability to fit objects into Web caches decreases as a result of the increased number of objects per Web page. We observe that for the Base Scenario the HR for the three κ are approximately 20 %, 40 %, and 60 %, which was our goal in choosing our values of κ.

Figure 5.10 also shows that HR increases as κ increases, which 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 5.3.5). This corresponds to the decrease in MRT observed in Figure 5.8, which is expected since, with an increase in

HR, Web page transmission time (τ) decreases, thus decreasing the request service time and consequnetly MRT.

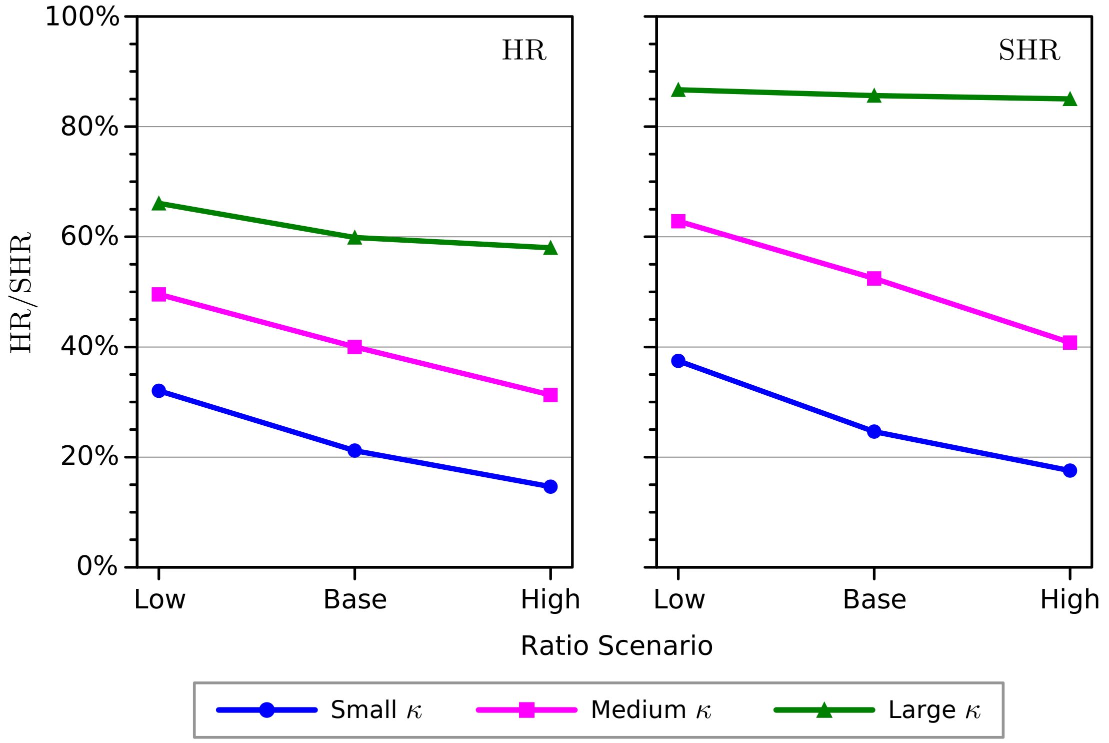


Figure 5.10: Effect of Ratio Scenarios on HR.
$\delta = 0.6$, $C = 5$, $N = 720$, $z = 35{,}000$

5.5 Summary

In this chapter, we incorporated our Web Page Request Access Pattern Model into an object-level caching system. We did this by replacing the Web server sub-model, presented in Chapter 3, with a homogeneous set of Web caches. These Web caches respond to user requests with an improved service time when objects are cached. A Cached Object Access Factor is applied to objects to represent an improved Web page transmission time. When requests are made for an object that is not cached, the Web page object cacheability parameter is used to determine if it is cacheable. 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 modified our simulation from Chapter 4 to include Web caching. This model introduced several Web caching parameters that we evaluated. We found that the Cached Object Access Factor does result in a reduction in both MRT and system load. When examining cache capacity, our results showed that increasing this leads to an increase HR and SHR, and a decrease in MRT.

Our final experiment 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 (Base, Low, High). The results demonstrated 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 Web caching both improved MRT and server load.

Chapter 6

Conclusions and Future Research

6.1 Contributions

This research aimed to develop a model to represent the process of object-level requests to Web pages in the *World Wide Web* (WWW). We used *mean response time* (MRT), which is the time that it takes to fulfill a user request, as the main measure of user experience. In Chapter 3, we established our model, the *User-Web Interaction Model*, that consists of three sub-models: the *User-Request Model*, the *Server Model*, and the *Web Page Model*. Our model represents Web pages at a finer granularity the than previous work that represented Web pages at the file level. Our model can form the basis of a unique framework that can be utilized for future research that requires Web page request access patterns.

Our main contribution is the *Web Page Model*, which represents the Web pages in the system, and is used to define the composition of the set of Web pages available in the system for user requests. The Web Page Model categorizes the composition of Web pages at the object-level, and represents the presentation of the Web page from the user's perspective. We generalize a variety of objects into six types, each of a specific size. In the Web Page Model, we characterize the composition of Web pages with three Web page categories (*article*, *mosaic*, and *media*), each with its own combination of Web page objects of varying size.

The User-Request Model represents the Web page selection and request by a user. It uses a file reference model to represent which Web pages are selected by a user, in such way that a higher *coefficient of variation of Web page request interarrival time* (CV) is attained. Our Server Model contains a homogeneous set of parallel Web servers. Each server has its own queue and fulfills the users requests.

In order to study the performance of the various parameters on user experience, we developed a *discrete event simulation* (DES) for our Web Page Request Access Pattern Model and presented the research in Chapter 4. We developed additional parameters that affect the composition of the Web page set, and established their values. One of those, the ratio of Web pages per Web page category, is used to determine how many Web pages of a particular category are available to the user. We also determined a set of ratios, referred to as the *Base Scenario*, for determining other parameters.

From our User-Request Model, we considered the parameters that affect CV, and established those to maintain a CV of approximately 3. When examining the effect of varying the number of users, we found that system load decreased as the number of servers increased. As well, we found that the MRT increases with the number of users.

Since the composition of Web pages is a distinct feature of our Web Page Request Access Pattern Model, we also examined the effects of varying Web page categories in Chapter 4. In this experiment, we developed 36 test scenarios, where we varied the ratio of Web pages per category in each scenario. Our results showed that we could predictably control the mean Web page size for all requests by varying the ratio of Web pages per category. We also found that MRT increased as the mean Web page size for all requests increased. We proposed three scenarios (*Low*, *Base*, and *High*) to be representative of the 36 test scenarios. The parameters that we established in Chapter 4 produce a wide range of conditions, that will serve as a basis for generating a variety of user request patterns in the WWW.

The final aim in our research was to incorporate our Web Page Request Access Pattern Model into Web server application. The one we chose was Web caching, and this was explored in Chapter 5. We adapted our base model to regard the Web servers as Web cache servers with limited storage capacity. It assumes that a cached object has a lower transmission time than one that is not cached. To produce a lower time we developed the *cached object access factor* parameter. The Web page composition, developed in Chapter 3, was expanded to include object cacheability, a parameter which represents the probability that an object is cached. Since cache size is limited, we use a *least recently used* (LRU) algorithm as a cache replacement policy.

The implementation of our Server Model was expanded to include Web caching, and was described in Chapter 5. While investigating the cached object access factor, we found it effective at reducing MRT, with diminishing effect as it was reduced. The base value for the cached object access factor was established to be 0.6. This means that the contribution to transmission time of an object that is found in cache is 60 % of that of an uncached object. We examined cache capacity and found that increasing cache capacity affects HR, with lower sizes having a large effect on HR and with diminishing returns as cache capacity increases.

We concluded Chapter 5 with an investigation into varying Web page categories with Web caching. We accomplished this by using our three scenarios (*Low*, *Base*, and *High*) from Chapter 4, which produce three distinct and consistent mean Web page sizes for all requests. Along with these Web page category scenarios, we also examined four categories of cache capacity (*small*, *medium*, and *large*, as well as one with *no caching* for reference). We found that for the no caching system, the results were consistent with those in Chapter 4, which showed that as we increased the mean Web page size for all requests the MRT increased. When Web caching was utilized, our results showed that HR increases with an increase in cache capacity. We also found that the system load decreased as cache capacity increased, although with diminishing returns as mean Web page size for all requests increased. Our main parameter of interest, MRT, was shown to improve with Web caching, especially as cache capacity increased.

Overall, we found that with our expanded server model (Web caching) access latencies can be improved (improved MRT) and can reduce load on Web servers. These are two important features of Web caching, and as such were were able effectively incorporate caching into our Web Page Request Access Pattern Model.

6.2 Suggestions for Future Research

6.2.1 Additional Object Types

A goal of our framework is to allow model components to be incorporated into other research. We proposed six Web page object media types that generalize the many that are available in the WWW. One such example of this generalization is the text object type, which is intended to be representative of *hyper-text markup language* (HTML) and *Extensible Markup Language* (XML) objects. These objects, when compared to one another, have been shown in previous studies to have different request rates and represent different request size proportions. As well, for simplicity we assumed that text objects are not cacheable at all, however, 10-20 % of HTML objects can contribute to byte savings from Web caching. In contrast, XML provides little savings. Another object type, octet, was left out of our model, and it represents a moderate amount of data transferred on the WWW, especially for larger Web pages. Octet objects are often related to video watching and large files. It would be interesting to see the effect of modelling these, and other object types.

6.2.2 Web Caching Applications

In our research, the Web caching model that we developed was limited in scope, and as such, there are a number of areas that interest to which our framework can be expanded. One area is the caching architecture. We used a simple homogeneous distributed caching architecture, however hierarchical and hybrid architectures would also be suited for our model. To do this, a multi-stage server configuration could be used. The cached object access factor (which we model as a single value parameter) could be adapted to be applied to the various configurations of the architecture being modeled. For example, in the case of a hierarchical cache, which has layers of caches based on geographic levels, cached object access factor would vary according to the level of the cache, smaller for regional caches and larger for national caches.

Another caching consideration is cache replacement policy, which is the algorithm used to determine how a cache adds and replaces objects. We chose to implement an LRU algorithm. There are several other replacement policies of interest, including *least frequently used* (LFU), *Size*, *LRU-MIN*, *Least Normalized Cost Replacement* (LCN-R), and *Size-Adjusted LRU* (SLRU) to name a few. Some of these were reviewed in Chapter 2.

6.2.3 Variations on Web Page Composition

We regarded our Web page categories as being representative of the Web page object composition as presented to the user. Our model established three categories based on anecdotal classification from a user's perspective. However, there are different ways to characterize the composition.

One example is to categorize Web pages terms of page load time: *short*, *med*, *long* (based on observations from [26]). These categories are composed of similar

Web page object media types as we developed in our model, with the addition of the octet type. As well, HTML and XML could replace our text object type. Short Web pages represent a small amount of Web traffic volume, and are dominated by HTML objects, and image objects. Medium Web pages make up about a third of the traffic volume, with about half of the objects being image, some video, and a smaller percentage of HTML. About half of the traffic volume consists of long Web pages, with images making up much of the content, and video and octet a smaller amount. The number of Web page objects in these categories vary substantially, with an order of magnitude difference between each one.

6.2.4 The Internet of Things

Another application for our framework is the *Internet of Things* (IoT). The IoT is a concept of interconnecting small devices and other systems for exchanging data. Examples of IoT devices include: smart home devices (such as: appliances, security systems, and lighting), health care devices, personal jewelry, embedded systems, and sensor networks, to name a few. The types of IoT devices span a large range of computing capabilities and data requirements. Some assumptions to characterize the Web page composition are as follows:

- IoT devices represent a high ratio of Web pages with respect to our developed categories.
- A page is made up of a small number of objects
- Small object size.
- Low cachability.

With the decrease in cost and miniaturization of computing devices their availability have increased substantially. Our framework could provide the opportunity to study the effects of the IoT.

6.2.5 Vehicular Computing

For a final topic of future research, we look at the evolving field of information technology in vehicular computing; more specifically, *vehicular cloud computing*. In general, vehicular computing is the transfer of information between vehicles and other networks. Some applications of vehicular computing include: urban surveillance, road safety, traffic efficiency, passenger access to Internet, messaging, and infotainment [11]. Recent innovations in cloud computing are being researched for use with vehicular computing.

Cloud computing uses a shared pool of computing resources for on-demand computation or storage systems [5]. Advantages of cloud computing include a large pool of resources that can scale dynamically to current demands with a high availability. Cloud computing facilitates ad-hoc networks, with large numbers of fast-changing vehicular computing nodes. This can allow computation to be distributed to network edges (that is, the vehicles) where underutilized vehicle resources can be used. [11,31].

Modelling such a system as a vehicular cloud adds a number of complications beyond the model presented in this thesis. One aspect to consider is the complex set of traffic flow models consisting of several mathematical models which include time-variant acceleration, speed, and position parameters. As well, the network topology goes beyond a simple client-server model, where there exists a large number of vehicle nodes that form a combined client-server and peer-to-peer network. A model for a vehicular cloud can be modelled as a hybrid of peer-to-peer and client-server models such as described in [32], and for peer-to-peer queueing networks we can look at [14].

Some of the information in a vehicular cloud could be modelled in part, as we have in this thesis (to represent typical user behaviour in the WWW). However, the model would need to be expanded to combine additional information concerning urban surveillance, road safety, and traffic efficiency. Some typical sources of information in a vehicular network are [11,31]:

Social networks: represent the most popular form of information in the network, and contain a large amount of data with a lot of variation in the amount and frequency.

Infrastructure: data does not alter frequently. Examples of which include: road network data, road description, geographical information, points of interest, local business data.

Urban surveillance: consists of images or video data of somewhat constant size with regular frequency.

Traffic data: vehicle telemetry that allows for the temporal and spacial tracking of vehicular traffic.

Local sensors: can include environmental, smart phone, and vehicle sensors. These would result in relatively low data sizes, but with highly dynamic and frequent transmission.

These information sources would be represented as Web page categories, each being a mix of static and dynamic data of various sizes and complexity.

Bibliography

[1] W. Ali, S. M. Shamsuddin, and A. S. Ismail. “A survey of web caching and prefetching”. In: *Int. J. Advance. Soft Comput. Appl.* Vol. 3. 1. ICSRS Publication, 2011, pp. 18–44.

[2] C. Allison, M. Bramley, and J. Serrano. “The World Wide Wait: Where Does the Time Go?” In: *Proceedings. 24th EUROMICRO Conference (Cat. No.98EX204).* Vol. 2. Aug. 1998, 932–938 vol.2. DOI: `10.1109/EURMIC.1998.708124`.

[3] M. F. Arlitt and C. L. Williamson. “Web Server Workload Characterization: The Search for Invariants”. In: Proc. of the 1996 ACM Sigmetrics, 1996, pp. 126–137.

[4] *Average Number of Web Page Objects Breaks 100.* 2012. URL: `http://www.websiteoptimization.com/speed/tweak/average-number-web-objects/` (visited on 11/04/2021).

[5] L. Badger et al. *NIST SP 800-146 Cloud Computing Synopsis and Recomendations.* Tech. rep. National Institute of Standards and Technology (NIST), 2012.

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

[7] P. Barford and M. Crovella. “Generating Representative Web Workloads for Network and Server Performance Evaluation”. In: *SIGMETRICS Perform. Eval. Rev.* 26.1 (June 1998), pp. 151–160. ISSN: 0163-5999. DOI: `10.1145/277858.277897`. URL: `https://doi.org/10.1145/277858.277897`.

[8] M. Belshe, R. Peon, and E. M. Thomson. *RFC 7540. Hypertext Transfer Protocol Version 2 (HTTP/2).* 2015.

[9] T. Berners-Lee. *Information Management: A Proposal.* Tech. rep. CERN, 1990.

[10] T. Berners-Lee et al. “World-Wide Web: The Information Universe”. In: *Electronic Networking* 2.1 (Spring 1992), pp. 52–58.

[11] A. Boukerche and R. E. De Grande. “Vehicular cloud computing: Architectures, applications, and mobility”. In: *Computer Networks* 135 (2018), pp. 171–189. ISSN: 1389-1286. DOI: `https://doi.org/10.1016/j.comnet.2018.01.004`. URL: `https://www.sciencedirect.com/science/article/pii/S1389128618300057`.

[12] V. Bush. “As We May Think”. In: *The Atlantic Monthly* (1945).

[13] R. Cáceres et al. “Web Proxy Caching: The Devil is in the Details”. In: *SIGMETRICS Perform. Eval. Rev.* 26.3 (Dec. 1998), pp. 11–15. ISSN: 0163-5999. DOI: `10.1145/306225.306230`. URL: `http://doi.acm.org/10.1145/306225.306230`.

[14] C. D. Carothers and R. Lafortune. “A Case Study in Modeling Large-Scale Peer-to-Peer File-Sharing Networks using Discrete Event Simulation”. In: *In Proceeding of the International Mediterranean Modeling Multiconference*. 2006, pp. 617–624.

[15] A. Chankhunthod et al. “A Hierarchical Internet Object Cache”. In: *In proceedings of the 1996 USENIX technical conference*. 1995, pp. 153–163.

[16] H.-K. Choi and J. O. Limb. “A Behavioral Model of Web Traffic”. In: *Proceedings of the Seventh Annual International Conference on Network Protocols*. ICNP ’99. USA: IEEE Computer Society, 1999, p. 327. ISBN: 0769504124.

[17] J. Conklin. *A Survey of Hypertext*. Tech. rep. Microelectronics and Computer Technology Corporation, 1987.

[18] D. Evans. *The Internet of Things: How the Next Evolution of the Internet Is Changing Everything*. Tech. rep. 2011.

[19] R. Fielding et al. *RFC 2616, Hypertext Transfer Protocol – HTTP/1.1*. 1999. URL: `http://www.rfc.net/rfc2616.html`.

[20] N. Freed and M. Kucherawy. *Media Types*. @ONLINE April 27, 2017 `https://www.iana.org/assignments/media-types/media-types.xhtml`. 2017.

[21] F. González-Cañete, E. Casilari-Pérez, and A. Triviño. “Characterizing Document Types to Evaluate Web Cache Replacement Policies”. In: Feb. 2007, pp. 3–11. DOI: `10.1109/ECUMN.2007.11`.

[22] R. T. Hurley, W. Feng, and B. Li. “Partitioning in Distributed and Hierarchical Web-Caching Architectures: A Performance Comparison”. In: *Proc. of the16th International Conference on Computer Applications in Industry and Engineering* 11.13 (Nov. 2003).

[23] R. Hurley and B. Li. “A Performance Investigation of Web Caching Architectures”. In: (2008).

[24] R. Hurley and B. Li. “Effects of Dynamic Content on Web Caching”. In: ().

[25] R. Hurley. “An Investigation of File Migration in a Distributed File System”. PhD thesis. University of Waterloo, 1994.

[26] S. Ihm. “Understanding and Improving Modern Web Traffic Caching”. PhD thesis. Princeton University, 2011.

[27] S. Ihm and V. S. Pai. “Towards Understanding Modern Web Traffic”. In: *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*. IMC ’11. Berlin, Germany: ACM, 2011, pp. 295–312. ISBN: 978-1-4503-1013-0. DOI: `10.1145/2068816.2068845`. URL: `http://doi.acm.org/10.1145/2068816.2068845`.

[28] L. Kleinrock. *Queueing Systems Volume I: Theory.* Wiley Interscience, 1975.

[29] T. Kroeger, D. Long, and J. Mogul. "Exploring the Bounds of Web Latency Reduction from Caching and Prefetching". In: (Dec. 1997).

[30] B. Mah. "An empirical model of HTTP network traffic". In: *Proceedings of INFOCOM '97* 2 (1997), 592–600 vol.2.

[31] R. Meneguette et al. "Vehicular Edge Computing: Architecture, Resource Management, Security, and Challenges". In: *ACM Comput. Surv.* 55.1 (Nov. 2021). ISSN: 0360-0300. DOI: 10.1145/3485129. URL: https://doi.org/10.1145/3485129.

[32] R. Mu and F. Zhao. "CDN and P2P Network Model Based on HCDN Technology". In: *Journal of Software Engineering* (2015), pp. 469–486. DOI: 10.3923/jse.2015.469.486.

[33] T. H. Nelson. "Getting it out of our system". In: *Information Retrieval: A Critical Review* (1967). Ed. by G. Schechter, pp. 191–210.

[34] D. Povey and J. Harrison. "A Distributed Internet Cache". In: *Proceedings of the 20th Australian Computer Science Conference* (Feb. 1997).

[35] V. Presutti and A. Gangemi. "Towards an OWL Ontology for Identity on the Web". In: (2020).

[36] P. Rodriguez, C. Spanner, and E. W. Biersack. "Analysis of Web Caching Architectures: Hierarchical and Distributed Caching". In: *IEEE/ACM Trans. Netw.* 9.4 (Aug. 2001), pp. 404–418. ISSN: 1063-6692. DOI: 10.1109/90.944339. URL: http://dx.doi.org/10.1109/90.944339.

[37] F. D. Smith et al. "What TCP/IP protocol headers can tell us about the web". In: *Proceedings of the Joint International Conference on Measurements and Modeling of Computer Systems, SIGMETRICS 2001, June 16-20, 2001, Cambridge, MA, USA*. ACM, 2001, pp. 245–256. DOI: 10.1145/378420.378789. URL: http://doi.acm.org/10.1145/378420.378789.

[38] W. Stallings. *Data and Computer Communications.* 8th ed. Pearson Prentice Hall, 2007.

[39] A. S. Tanenbaum and M. van Steen. *Distributed Systems: Principles and Paradigms.* 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006. ISBN: 0132392275.

[40] J. Wang. "A Survey of Web Caching Schemes for the Internet". In: *SIGCOMM Comput. Commun. Rev.* 29.5 (Oct. 1999), pp. 36–46. ISSN: 0146-4833. DOI: 10.1145/505696.505701. URL: http://doi.acm.org/10.1145/505696.505701.

[41] X. S. Wang, A. Krishnamurthy, and D. Wetherall. "How Much Can We Micro-Cache Web Pages?" In: *Proceedings of the 2014 Conference on Internet Measurement Conference.* IMC '14. Vancouver, BC, Canada: ACM, 2014, pp. 249–256. ISBN: 978-1-4503-3213-2. DOI: 10.1145/2663716.2663739. URL: http://doi.acm.org/10.1145/2663716.2663739.

Appendices

Appendix A

Simulation Implementation Overview

To achieve our results, we use *discrete event simulation* (DES), which is a simulation approach where events are managed at discrete time units. Events are generated randomly (except for the *end of simulation event*) and added to a *Future Events List* (FEL). The FEL is ordered by the event time, where the *next event* has the lowest time. Time is managed by a master clock, which is advanced when the *next event* is processed. Between each time unit, the status of the system remains unchanged. The simulation time scale is not directly related to real time, consequently, time is only used as a basis for relative comparison of performance.

There are three general classes of events: *request-response*, *popularity changes*, and *simulation control*, which are described as follows:

Request-response: uses two events to implement the User-Web Interaction Model (Section 3.2.1). These are the *user request events* and *server response events*. Effectively, a user alternates cyclicly between request and response, as described in Figure 3.2.

Upon simulation initialization, every user in the system starts in a *thinking* state, and has a *user request event* scheduled. When a *user request event* is received, the user state is set to *waiting* and the request is either serviced (if the server is *idle*) by creating a *server response event*, or placed in the server's queue (if the server is *busy*). When a *server response event* is received, the user state is set to *thinking* and the response is considered complete. At this point, the request-response cycle starts again by creating a new *user request event* for the user. If there are already requests in the current server's queue after the current *server response event* is handled, the next queued request is served by creating a new *server response event* (according to the queueing discipline presented in Section 3.2.1).

Popularity changes: are responsible for changing the state of potentially popular Web pages. These events are independent of the other event types. Potentially popular Web pages alternate between *normal* and *popular*, as described in Section 3.2.2 and Section 4.2.3.

Simulation control: uses one event type, the *end of simulation event*, which when encountered, terminates the simulation. A single *end of simulation event* is scheduled during the simulation initialization, and is an input variable that is used to determine the simulation length. The simulation length is discussed in Section 4.2.2 and Section 5.3.2.

The details of these events are shown in Figure A.1, which provides an overview of the main simulation event handling loop.

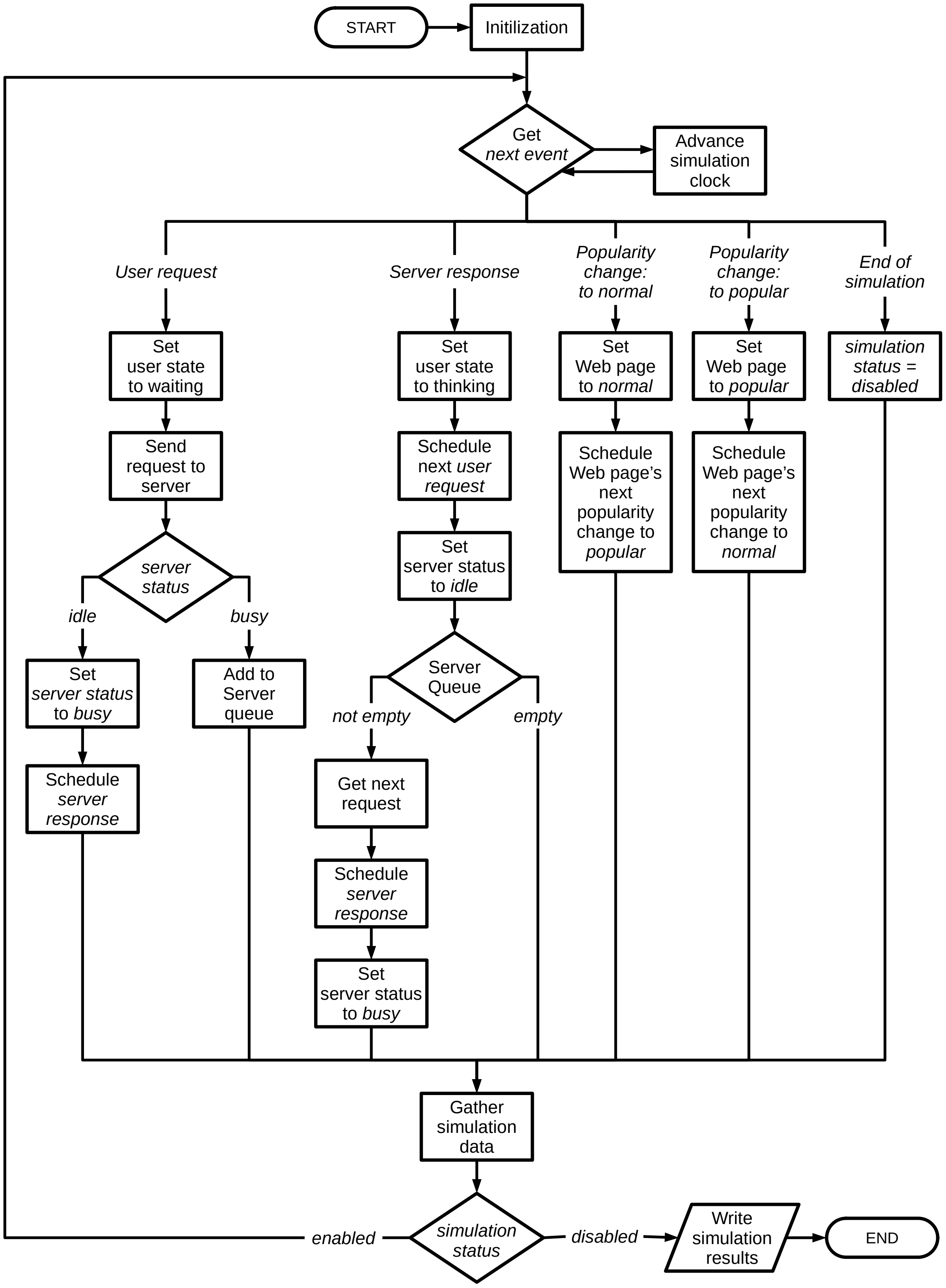


Figure A.1: Main Simulation Event Handling Loop.