สถาปัตยกรรมฟาร์มแคชอัจฉริยะสำหรับการจัดการเครือข่าย

นางสาวสุภาวดี หิรัญพงศ์สิน

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรดุษฎีบัณฑิต
สาขาวิชาวิทยาการคอมพิวเตอร์ ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2554
ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

บทคัดย่อและแฟ้มข้อมูลฉบับเต็มของวิทยานิพนธ์ตั้งแต่ปีการศึกษา 2554 ที่ให้บริการในคลังปัญญาจุฬาฯ (CUIR)
เป็นแฟ้มข้อมูลของนิสิตเจ้าของวิทยานิพนธ์ที่ส่งผ่านทางบัณฑิตวิทยาลัย
The abstract and full text of theses from the academic year 2011 in Chulalongkorn University Intellectual Repository(CUIR)
are the thesis authors' files submitted through the Graduate School.

INTELLIGENT CACHE FARMING ARCHITECTURE FOR NETWORK MANAGEMENT

Miss Supawadee Hiranpongsin

A Dissertation Submitted in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy Program in Computer Science

Department of Mathematics and Computer Science

Faculty of Science

Chulalongkorn University

Academic Year 2011

Copyright of Chulalongkorn University

Thesis Title	INTELLIGENT CACHE FARMING ARCHITECTURE FOR NETWORK MANAGEMENT
By	Miss Supawadee Hiranpongsin
Field of Study	Computer Science
Thesis Advisor	Assistant Professor Pattarasinee Bhattarakosol, Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Doctoral Degree

……………………………………………….. Dean of the Faculty of Science
(Professor Supot Hannongbua, Dr. rer. nat.)

THESIS COMMITTEE

……………………………………………….. Chairman
(Assistant Professor Nagul Cooharojananone, Ph.D.)

……………………………………………….. Thesis Advisor
(Assistant Professor Pattarasinee Bhattarakosol, Ph.D.)

……………………………………………….. Examiner
(Siripun Sanguansintukul, Ph.D.)

……………………………………………….. External Examiner
(Assistant Professor Ohm Sornil, Ph.D.)

……………………………………………….. External Examiner
(Surapant Meknavin, Ph.D.)

สุภาวดี หิรัญพงศ์สิน : สถาปัตยกรรมฟาร์มแคชอัจฉริยะสำหรับการจัดการเครือข่าย. (INTELLIGENT CACHE FARMING ARCHITECTURE FOR NETWORK MANAGEMENT) อ. ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ. ดร. ภัทรสินี ภัทรโกศล, 97 หน้า.

การแคชเว็บ (Web Caching) นับเป็นเทคนิคที่มีประสิทธิภาพเทคนิคหนึ่งที่ได้รับการยอมรับว่าสามารถปรับปรุงคุณภาพการให้บริการบนอินเทอร์เน็ตได้เป็นอย่างดี ตัวอย่างเช่น ลดระยะเวลาที่ผู้ใช้งานรอการตอบกลับจากเครื่องให้บริการ และลดการใช้งานแบนด์วิดท์ของเครือข่าย เป็นต้น อย่างไรก็ดีการให้บริการบนอินเทอร์เน็ตยังคงพบกับปัญหาเมื่อปริมาณของผู้ใช้งานอินเทอร์เน็ตเพิ่มขึ้น เนื่องมาจากข้อจำกัดทางฮาร์ดแวร์และนโยบายการบริหารจัดการแคช เพื่อเป็นการแก้ไขปัญหาข้างต้นงานวิจัยนี้จึงได้นำเสนอสถาปัตยกรรมฟาร์มแคชอัจฉริยะ (Intelligent Cache Farming Architecture - ICFA) เพื่อเป็นต้นแบบสถาปัตยกรรมการแคชทางเลือกที่ได้รวมกลไกของการให้คำแนะนำไว้ด้วย พฤติกรรมการเรียกใช้งานบนอินเทอร์เน็ตของผู้ใช้งานจะถูกนำมาวิเคราะห์และนำมาใช้งานสำหรับบริหารจัดการกลุ่มแคช ดังนั้นประสิทธิภาพของการให้บริการอินเทอร์เน็ตจะถูกปรับปรุงอย่างมากและชัดเจนยิ่งขึ้น สถาปัตยกรรมที่นำเสนอนี้ถูกทดสอบด้วยการจำลองเหตุการณ์การใช้งานอินเทอร์เน็ต และสถาปัตยกรรมแบบดั้งเดิมที่ไม่มีการจัดกลุ่มของแคชบนสภาพแวดล้อมเครื่องกลแบบเสมือน (virtual machine environment) ผลลัพธ์ที่ได้จากการจำลองเหตุการณ์ชี้ให้เห็นว่า การวัดระยะเวลาการรอ (delay) ของสถาปัตยกรรม ICFA ลดลงได้มากกว่า 44% ในขณะที่อัตราความแม่นยำ (hit ratio) และอัตราขนาดของข้อมูลในแคชที่สามารถตอบสนองความต้องการของผู้ใช้งานได้ (byte hit ratio) เพิ่มขึ้นมากกว่า 20% และ 43% ตามลำดับ

ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์ ลายมือชื่อนิสิต...
สาขาวิชา วิทยาการคอมพิวเตอร์ ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก………………
ปีการศึกษา 2554

5073890623 : MAJOR COMPUTER SCIENCE

KEYWORDS : WEB CACHING / PROXY SERVER / RECOMMENDER SYSTEM / WEB CLASSIFICATION / WEB USAGE PATTERN

SUPAWADEE HIRANPONGSIN : INTELLIGENT CACHE FARMING ARCHITECTURE FOR NETWORK MANAGEMENT. ADVISOR : ASST.PROF. PATTARASINEE BHATTARAKOSOL, Ph.D., 97 pp.

Web caching is widely recognized as an effective technique that improves the quality of service (QoS) over the Internet, such as reducing user latency and network bandwidth usage. However, it still meets problems when the number of users increases due to limitations of hardware and management policy of caches. Therefore, this research proposes the Intelligent Cache Farming Architecture (ICFA) to be an alternative caching architecture model integrating with the recommending mechanism. The proposed architecture is the cache grouping mechanism where browsing characteristics are applied to improve the performance of the Internet services. In order to prove the proposed architecture, the trace-driven simulation and the traditional caching model, which is the original model without grouping criteria for the web cache, are implemented in a virtual machine environment. The results indicate that the delay measurement of the ICFA is dropped more than 44%, while the hit rate and the byte hit rate of the ICFA increase more than 20% and 43%, respectively.

Department : Mathematics and Computer Science	Student's Signature
Field of Study : Computer Science	Advisor's Signature
Academic Year : 2011	

ACKNOWLEDGEMENTS

During my years as a Ph.D. student, I have received a lot of tuition, care and friendship from several people, some of which I wish to thank here. I would like to thank the Office of Higher Education Commission, Ubon Rajathanee University, and Chulalongkorn University for their financial support.

I would like to express my deepest gratitude to my advisor, Assist.Prof.Dr. Pattarasineee Bhattarakosol, to whom with her advice, guidance and care, help me to overcome all the difficulties of the process of research and make this dissertation possible.

My thanks also goes to dissertation committee, Assist.Prof.Dr. Nagul Cooharojananone, Dr. Siripun Sanguansintuku, Assist.Prof.Dr. Ohm Sornil, and Dr. Surapant Meknavin for their advices and guidance about the research activities.

I would like to thank the Department of Computer Science, Purdue University for their facility support during my visiting scholar in 2009-2010, especially, Distinguished.Prof.Dr. Douglas E. Comer for encouragement, guidance and spending time on my research discussion.

I would also like to thank all lecturers and colleagues at the Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University, the Thai Students Association and friends in Purdue University, and all my friends, for their help, warmest care and support, especially Mr. Worrachat Tangsurat who gave me the idea of writing a program.

Last but not least, I would like to express my sincere gratitude and deep appreciation to my parents, and my family for constant encouragement, love, and supports throughout my life.

CONTENTS

page

page

List of Tables

List of Figures

Figure page

CHAPTER I

INTRODUCTION

1.1 Introduction and Problem Review

Since a high volume of information is available over the Internet via the web-based system, growth rate of web usage is rapidly enlarged infinitely. The quality of services (QoS) of Internet Service Providers (ISPs) is then highly affected and various strategies have been settled to maintain their services. Unfortunately, none of these strategies can completely satisfy their customers. Thus, a long delay and a low throughput rate occur.

One important metric of the QoS is the performance where different variables are applied to identify this indicator, such as an average delay, an average hit rate, an average byte hit rate, etc. The values of these indicators are influenced by different factors, which are such as the limitation of hardware, unqualified service management system, etc. As a consequence, the termination of the browsing transactions is performed [1-2] and the performance drops.

In order to avoid this problem, many techniques have been proposed and applied for managing the cache. Originally, the cache replacement algorithm was used to manage a single cache [3-7]. Currently, the cache management is based on the cache farming architecture where composed caches work together. Two different management models of cache farming are a hierarchical model [8], and a distributed model [9]. The hierarchical caching model requires some intermediate caches arranged as a tree-like structure in the network and allows those caches to work together in a parent-child relationship. This process satisfies many users who request web documents. Each retrieval process leaves a copy of the requested web document in every cache levels through the traversal paths. Unfortunately, this hierarchical caching model has several problems associated with a caching hierarchy, such as an additional overhead at every hierarchy, a bottom necks issue at upper level caches making high client latency, and the wastes of the proxy cache space [10].

On the other hand, the distributed caching model has no intermediate caches. None of the copies are saved at the intermediate caches. When a request is issued, the content search will be performed over the distributed caches. The distributed method is achieved in a cooperative proxy caching unless its implementation encounters several problems, such as having complicated network system, high bandwidth usage, and administrative issues [10]. Other issues for a rough comparison between hierarchical and distributed caching architectures are shown in [11].

Based on the problems mentioned above, a cache management mechanism considering types of transactions was proposed. Additionally, the user behavior should be used as a management factor when the cache is performed [12]. Thus, this research proposes a new architecture of cache farm integrating the concept of recommender system. All arising behaviors are considered to manage the incoming transactions so the retrieval time and bandwidth can be maintained as expected. Consequently, the proposed solution can be called as an economical system because the number of caches needs not expand when the number of users expands [13-14].

1.2 State of problem

According to the literature [15], the performance of a browse is independent of the type of browsed file for all existing cache management algorithms. As a consequence, all ISPs always extend volumes of their service caches for every certain period of time. This management method causes system complexity and increases cost of management. Therefore, this research proposes a possible solution to manage the cache farm so that the performance of the service or the QoS can satisfy many users.

The new architecture, called as the Intelligent Cache Farming Architecture (ICFA), is implemented to manipulate cache farm in an economical way. This proposed architecture using the concepts of recommender system can maintain the QoS that is the response time without expanding cache sizes when the number of users increases. Therefore, the behavior of users over the Internet must be correctly defined in order to be able to classify the right groups of browsed webs. Consequently, this new architecture can maintain or increase the performance of cache management although the number of users is altered [16].

1.3 Research objectives

The main objectives of this research are the following:

- To propose a new architecture of cache farming with the recommender system concepts. The proposed architecture is the groups of caches where user behaviors are applied to improve the performance of the Internet services.

- To enhance or maintain the QoS without expanding sizes of caches when the number of users increases. The proposed architecture and management mechanism will be applied for web caching system.

1.4 Scopes of the Study

In this research, the scopes of work are constrained as following:

- To study the patterns of retrieved webs over the Internet in academic areas. The studied patterns are used as the grouping criteria for the web classification.

- To perform the experiments which are independent of the efficiency of peripheral devices supplied by various venders.

- To evaluate the performance of proposed architecture and algorithms on the simulation model using the data of web browsing provided by the volunteer university as the experimental data.

1.5 Contribution

The Intelligent Cache Farming Architecture (ICFA) is proposed in this research to be an alternative caching architecture model integrating with the recommending mechanism. The behaviors of users are analyzed and applied to manage groups of caches so the performance of the Internet services is highly and obviously improved. The components of the ICFA are the web pattern database system (WPDB), the proxy manager (PM), and the specific proxy servers (SPSs). Within the PM, however, there is an important module called the automatic classification module (ACM) that is responsible for classifying the retrieved webs. The ACM applies the concepts of the recommender system to classify the right groups of browsed webs based on their patterns. The proposed architecture was proved by the trace-driven simulation and the traditional caching model in a virtual machine environment.

1.6 Research Plans

In order to achieve the defined objectives above, the following tasks will be stated by mean of appropriate principal related works and theoretical techniques:

- Study concepts of related technologies, such as cache management policies, user requirements, web classifications, proxy systems, hierarchical and distributed systems, and recommender systems

- Define problem of the study and review various kinds of literatures relating to this research

- Collect data that is Squid log files, named access.log obtained from University Office of Information Technology of Chulalongkorn University to analyze the patterns of browsed webs. Moreover, the log files are also gathered from Computer Center of Nakhon Pathom Rajabhat University to use for the simulation experiments as the experiment data

- Define model among the relationships of data achieved by grouping mechanism of browsing behavior
- Define a prototype of network according to the defined model of user’s browse
- Develop mechanism to maintain the database system and an algorithm to manage all functions of system which is the dynamic mechanism
- Prove the performance of defined prototype by simulating the system on virtual machine environment
- Test and evaluate the functions of system and report the results

1.7 Benefits of the Research

Although the number of users continuously increases without expanding the size of caches in the farm, the proposed architecture will:

- Help reducing the retrieval time and also increasing the hit rate and the byte hit rate since the caching system has a pattern of Internet services.
- Maintain the cost and avoid administrative issue due to URL’s grouping and simple structure of the proxy cache farming system

1.8 Organization of the Dissertation

The rest of this dissertation is organized into five chapters as follows.

Chapter II is the background information and the methods related to this research. This chapter describes the implementations of the proxy server, the types of the proxy server, the architectures of the web proxy caching, the cache replacement policies, and the recommender system.

Chapter III is the explanation of data collection and analysis methods. The data used in this research is the Squid log file, named access.log. The data collection section presents Squid Log Format Overview, Data Sources, and Log File Filtering. The data analysis is the grouping mechanism which includes Web directory-based grouping and Browsing behavior-based grouping.

Chapter IV describes the new architecture and the modified algorithm proposed to manage caches in the farm as well as the system workflow. The proposed architecture comprises of the web profile database system, the proxy manager system, and the specific proxy server system. The proposed algorithms consist of the information update process algorithms, which include the daily update process algorithm, the weekly update process algorithm, and the monthly update process algorithm, and the web pattern classification algorithm. For the last section, this chapter presents the system workflow.

In Chapter V is the implementation and experimental results of this research. In order to evaluate and compare the caching performance of the proposed architecture to the traditional caching architecture, which is original model without grouping criteria for the web cache, the trace-driven simulation is performed by implementing both models in a virtual machine environment. The simulation results to demonstrate the performance of the proposed architecture are elaborated as well.

Finally, Chapter VI is the discussion and conclusion on the proposed cache farming architecture applying the recommending mechanism.

CHAPTER II

BACKGROUND AND LITERATURE REVIEWS

This chapter describes the implementations of the proxy server, the types of the proxy server, the architectures of the web proxy caching, the cache replacement policies, and the recommender system. The implementations of the proxy server present the caching proxy server, the web proxy server, the transparent proxy server, and the squid proxy server including its installation and configuration. The forward proxy and the reverse proxy are presented in the common types of the proxy server. Three types of the architectures of the web proxy caching are the hierarchical, the distributed, and the hybrid caching architectures. The various cache replacement policies defined in the proxy cache are such as Least Recently Used (LRU), Least Frequently Used (LFU), Greedy Dual-Size (GDS), etc. The recommender system comprises of two basic approaches: the content-based filtering (CB) and collaborative filtering (CF).

2.1 Implementations of Proxy Server

A proxy server [17-20] is a hardware that is located between Web servers and one or more clients. It facilitates access to the content on the Internet. A proxy server receives requests from clients who search some resources from other servers. These clients connecting to the proxy server request some services, such as files, web pages, or other resources available from different servers. However, the proxy servers are implemented for one or more of functions as following:

2.1.1 Caching Proxy Server

A caching proxy server services the requested objects without contacting the original server of the objects. It retrieves the contents of requested objects which are saved on its cache to send them back to the users. The caching proxy server conserves the local copies of popular resources. This significantly reduces the upstream bandwidth usage and cost of the Internet Service Providers (ISPs), hence, increasing its performance. Figure 2.1 shows the basic operation of caching proxy server.

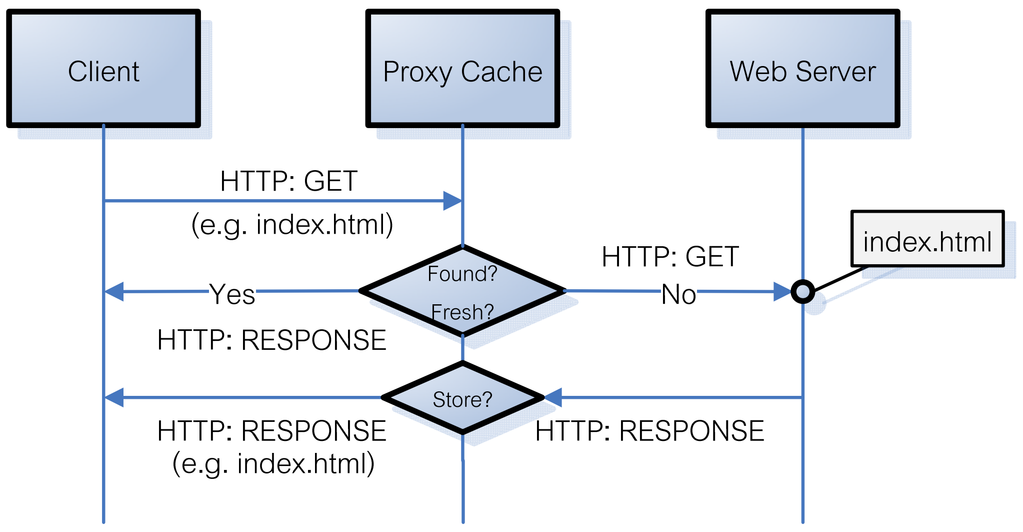


Figure 2.1: The basic operation of caching proxy server.

Referring to Figure 2.1, when a proxy cache receives a request made by a client, it will determine whether the requested object is already in its cache by examining the list of stored objects. Moreover, the proxy cache decides whether the stored object is valid for sharing (fresh) by examining available information of that object, such as the object creation date, storage date, expiration date, client and server preferences, etc. However, the decision can be made accurately by the help of the HTTP version 1.1. If the stored object is fresh, it will be retrieved from the local cache, and then sent as an HTTP response to the client. In this case, there is no need to fetch the objects from the object's web server, resulting in the retrieval time and bandwidth saving.

However, if a fresh copy of the object cannot be found in the cache, it must be retrieved from the original web server. The retrieved object will be stored in the cache if the disk space is available and finally the cache forwards this object as the response to the client.

2.1.2 Web Proxy Server

A web proxy server is a server focusing on the World Wide Web traffic. Many web proxy servers attempt to block unpleasant the web contents. Other web proxy servers re-format the web pages for a specific purpose, such as smart phones. However, the network operators can also deploy these proxies to detect computer viruses and other unreceptive contents served on remote web pages.

In order to provide caching services and security, several organizations, such as academic organizations, commercial organizations, etc., implement the proxy servers to their network system. Generally, the traditional web proxy is set up as non-transparent proxy to the client application, which must be configured to use the proxy either manually or with a configuration script.

2.1.3 Transparent Proxy Server

A transparent proxy server is also known as an intercepting proxy server or forced proxy server. It is between the client's browser and the Internet. Generally, the transparent proxy combines a proxy server with a gateway. When the connections are initiated by the clients through the gateway, they are recognized and redirected through the proxy cache without configuring clients. For this reason, this type of proxy server is extremely attractive to network administrators. Additionally, the administrators have no need specific instructions for web applications and their versions, then they can control over the traffics sent to the caches.

However, most Internet Service Providers (ISPs) or other organizations that provide the Internet services use this implemented proxy server to save their upstream bandwidth and also improve user delay or response times by caching.

2.1.4 Squid Proxy Server

Squid proxy server [21-22] is a full-featured caching proxy server [23] that caches the web content closer to a client than its original location of origin web server.

Various kinds of web objects, including the popular network protocols that are such as HTTP, FTP, etc., can be supported the caching of the Squid proxy. Normally, the squid is implemented to perform transparent caching; however, it can implement caching of Secure Sockets Layer (SSL) and Domain Name Server (DNS). Furthermore, the squid also support a variety of caching protocols, such as Internet Cache Protocol (ICP), the Cache Array Routing Protocol (CARP), and the Web Cache Coordination Protocol (WCCP), etc.

2.1.4.1 Installation

The squid is open source software which was originally designed to run on Unix-based systems. However it can also be run on windows systems. The squid proxy server is usually installed on a separate server. If the system is running on, for example, Ubuntu, one of the Unix-based systems, the command to install the squid software is "**sudo apt-get install squid**". The first implementation of the squid proxy server was performed in the Harvest Project. The squid proxy is a high-performance proxy caching server and widely used in the organizations especially in the academic field [24].

The ISPs employ the squid proxy servers to provide faster web retrieval process and reduce user's latency, particularly for delivering rich media contents, such as video contents. Additionally, the operators of the web site frequently locate the squid proxy server as the content accelerator. This accelerator caches frequently retrieved content and alleviates loading on the web servers. In addition, the squid proxy is used in the content delivery networks to improve load balancing and to handle all traffic spikes for popular web contents.

2.1.4.2 Configuration

Referring to the installation of Squid mentioned previously, both of the Unix-based and Windows-based systems can implement the squid proxy server. Since this research working on Ubuntu, the detailed configuration of Squid is based on Ubuntu afterward. The configuration of the squid is to edit the directives contained within the configuration file named "/etc/squid/squid.conf" that a large number of options are included. Some directives might be also modified. They are listed below. For more squid's configurations in various systems, see the references in [17], [22], [25].

- To set the squid proxy server to listen on TCP port 8888 instead of the default TCP port 3128, the http_port directive is changed to: "**http_port 8888**"

- To set the Internet use to be available only for users with 161.200.126.0/24 subnetwork, the squid's access control is configured as follows: add "**acl example_network src 161.200.126.0/24**" at the ACL section of the /etc/squid/squid.conf file. Then, add "**http_access allow example_network**" at the http_access section of the /etc/squid/squid.conf file

After the /etc/squid/squid.conf file is configured and saved, the squid server application is restarted to operate the changes of the squid.conf file using the command entered in a terminal prompt: "**sudo /etc/init.d/squid restart**".

2.2 Types of Proxy Server

This section describes two common types of proxy server that are a forward proxy server and a reverse one. The forward proxy acts as a specific group of content consumers while the reverse one acts as the origin web server and helps the specific group of servers to deliver the content [26].

2.2.1 Forward proxy server

The most common form of the proxy server is the forward proxy server that is generally used to pass the requests from an internal network to the Internet through a firewall. However, the requests from the internal network, or the Intranet, can be rejected or allowed to pass through the firewall. Instead of passing through the Internet, the requests will be fulfilled by serving from the cache of the forward proxy. Figure 2.2 shows a forward proxy configuration. The procedure of the forward proxy is described as follows.

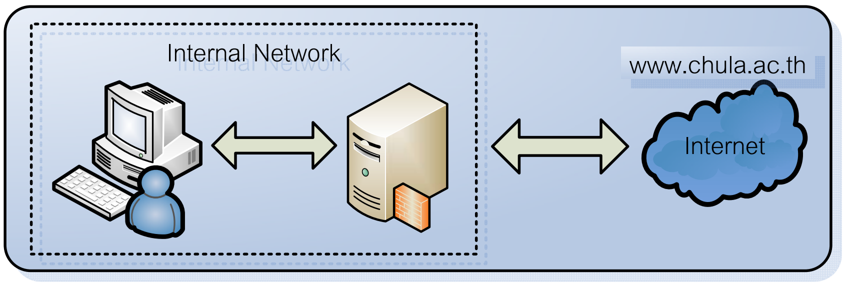


Figure 2.2: Forward proxy server.

1. Check the validation of a request: the forward proxy server verifies the request. If the request is not valid or blocked by the proxy, the forward proxy will reject the request and then the client will receive an error or a redirect. If it is not so meaning the request is valid, the forward proxy will check the requested information on its cache.

2. Check the existence of requested contents in the cache: the forward proxy serves the cached contents to the user if the contents are found in the proxy cache. If the contents are not found, the request is sent through a firewall to the content's web server. Afterword, the forward proxy server will transmit these contents to the client and copy them for the future requests on its cache.

2.2.2 Reverse proxy server

Another common form of a proxy server is the reverse one that is generally used to pass the requests from the Internet, through a firewall to the internal or private networks. This type of proxy server appears to users to be an ordinary web server; so that the requests are forwarded to this reverse proxy server that handles those requests. The reverse proxy server is used to prevent the Internet users from directly or unwanted accessing to the sensitive data residing on the content servers under the internal network. If the caching is allowed, the reverse proxy will serve the cached contents rather than sending all requests to the origin content servers. Figure 2.3 shows a reverse proxy configuration. The procedure of the reverse proxy is described as follows.

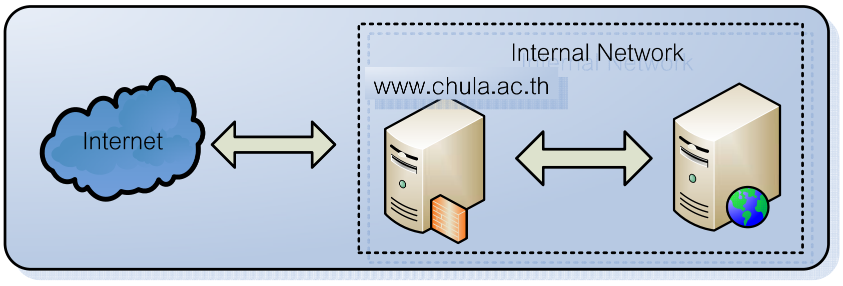


Figure 2.3: Reverse proxy server.

1. Check the validation of a request: the reverse proxy server verifies the request. If the request is not valid or blocked by the proxy, it will not continue to process the request and then the client will receive an error or a redirect. If it is valid, the reverse proxy server will check whether the requested contents are cached.

2. Check the existence of requested contents in the cache: the reverse proxy server serves the cached contents to the user who requests that web contents if the requested contents are found in its cache. If not, the reverse proxy server will request the contents from the original content server and serves them to the requesting user. In addition, the reserve proxy also keeps the copied contents in its cache for future requests.

The advantages of the reverse proxy setting are (i) reducing the bandwidth usage on the internal network, and (ii) allowing the web contents even though the web server is offline.

For the first advantage, the bandwidth usage is not required on the internal network when the cached contents are served directly from the reverse proxy server. Thus, the bandwidth usage on the internal network is reduced. Another advantage is that the reverse proxy server can provide the web contents when the web server is offline. Since the web server can be offline for various reasons, such as crashing of a hardware or software of the web server, or the downtime of the web server during the routine maintenance period. Therefore, whenever the web server is offline, the web contents are available to the Internet users because the contents are served from the reverse proxy cache. Furthermore, there are several reasons for installing the reverse proxy server are such as Load balancing, Security, Encryption, Compression, etc [27-28].

2.3 Architectures of Web Proxy Caching

Proxy caching is an example of technologies which overcome the two main problems which are server overloading and network congestion [11], [27], [29-31]. Using this approach, the content objects of retrieved webs are moved closer to the user for faster retrieval [24], [32]. Then, the user requests will be redirected to and served from the proxy device. Instead of requesting the pages from the original web server, they can be quickly retrieved from the cache of proxy server, by the same or different clients the next time that pages are requested. Hence, the access time is shortening and the significant network resources, such as bandwidth and processing resources are conserved. Figure 2.4 shows a single caching proxy configuration.

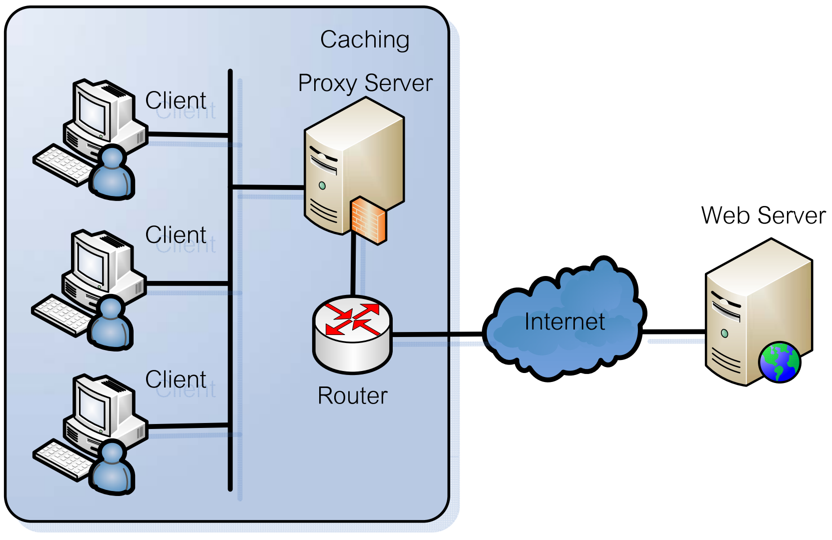


Figure 2.4: A single caching proxy.

2.3.1 Hierarchical caching architecture

A caching hierarchy is one of the caching models set up in order to coordinate proxy caches in the same network system. This hierarchical caching model locates the caches in each level of the network as illustrated in Figure 2.5. There are the client or browser caches at the bottom, the institutional caches, the regional caches, the national caches, and the original web server at the top of cache levels [10], [33]. A request made by the client is satisfied firstly at the client cache if the request is found on this level cache. If it is not so, the request is redirected to the next level caches, the institutional cache, the regional caches, and so on, to find the requested object. However, if the requested object is not found at any level of the caches, the national cache will directly contact the object's web server. Either at any cache or at the original web server that the requested object is found, the object travels down the hierarchy and leaves its copy at every level cache along its traversal path. Further requests for the same requested object travel up the caching hierarchy until the object is hit at some cache level.

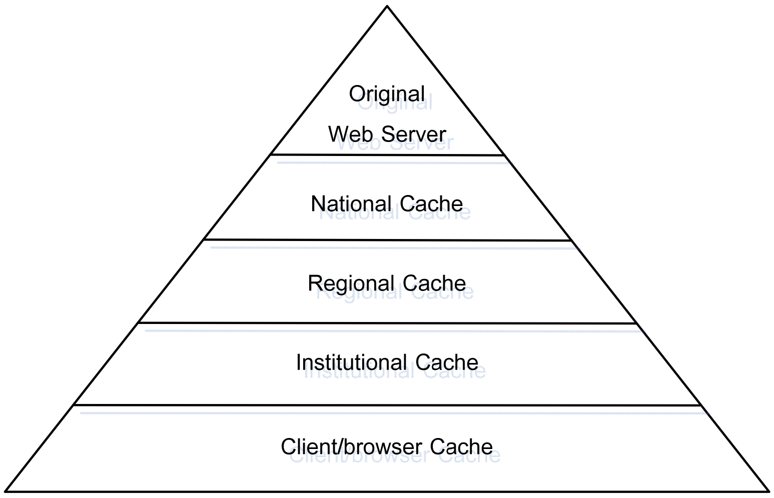


Figure 2.5: Hierarchical caching architecture.

The hierarchical web caching was proposed in the Harvest project [8] firstly since 1996. Then, Adaptive Web caching [34] and Access Driven cache [35] focusing on the improvement of the hierarchical caching, are introduced. In order to improve this caching model to support the Internet usages rapidly growing, many research papers studied on the architecture of caches by managing different cache sizes as hierarchical structure of a web cache farm. Foygel and Strelow [36] applied a prefetching algorithm to the hierarchical web caches. Moreover, the Internet Small Computer System Interface (iSCSI) protocol [37] to communicate between a lower-level; and, its higher-level proxy server was proposed with a new web caching scheme. In the year 2005, the Hierarchical Web Caching Placement and Replacement (HCPR) algorithm was drawn out [38]. This algorithm places the most frequently referenced documents close to users in the leaf nodes of the hierarchy. Then, the Content Distribution Network (CDN) architecture was applied to the hierarchical web caching technology by Yang and Chi [39].

Originally, the researches on hierarchical caching model are usually reserved for depth-first search (DFS) exploration. Thus, a general framework of hierarchical caches that can be used by breadth-first search (BFS) was proposed [40]. This method can guarantee to terminate and to traverse every transitions of the state space. However, there is very little research made on the web cache performance for the different combinations of replacement policy. In order to evaluate the performance of web caching, mostly, performance metrics are captured by Hit Rate (HR) and Byte Hit Rate (BHR) [41-42]. Additionally, a cost function model was implemented based on both performance metrics to investigate the suitability of applied policies over the two-level hierarchical cache model [43]. The result indicated that the performance obtains higher when the lower-level cache uses the LFU or LRU and the upper-level cache uses the GDS.

Despite the benefits of hierarchical caching architecture, this caching architecture has several problems associated with a caching hierarchy [9-10]:

1. The implementation of this caching hierarchy need to place the proxy servers at the key access points in the network. Consequently, this caching model frequently requires significant coordination among proxy servers.

2. Every cache level introduces additional delays.

3. The bottleneck problem may occur at high level caches due to travelling up the caching hierarchy, resulting in having long queuing delays.

4. There are many copies of the same requested object stored at different cache levels.

2.3.2 Distributed caching architecture

According to the problems of the hierarchical caching architecture mentioned above, the distributed caching architecture is set up to provide the requested webs made by the clients. The distributed caching model [9], [46], has no other intermediate caches than the institutional caches, which serve the miss of other caches. In order to make a decision which the institutional cache serves a miss web document, each institutional cache keeps the meta-data information informing the contents of other institutional caches. However, the meta-data information is more efficient and scalable when a hierarchical distribution mechanism is applied. The hierarchy in the distributed cache model is only used to distribute the information of directories indicating the web document's location, not the actual copies of the web documents.

The benefits of this distributed caching model are less congested, no additional disk space, better load sharing, and more fault tolerant. Nevertheless, a deployment of a large-scale distributed caching model encounters various problems, such as high connection times, high bandwidth usage, administrative issues, etc. [10]. Figure 2.6 shows the structure of the distributed caching architecture.

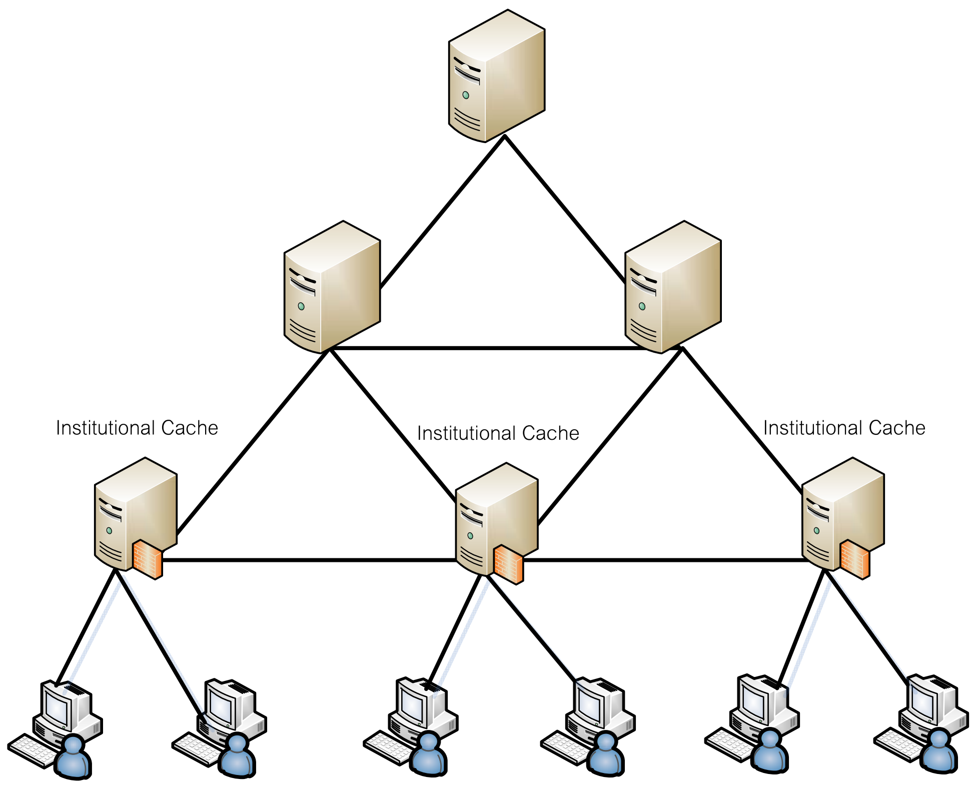


Figure 2.6: Distributed caching architecture.

There are several approaches employed to the distributed caching. The Internet Cache Protocol (ICP) [44] designed by the Harvest group is message-based scheme that communicates between caches using a simple query message. A different approach is taken in Cache Array Routing Protocol (CARP) [45] that distributes the URL among neighbor caches storing only the documents whose URL is hashed to it. Additionally, Provey and Harrison proposed a distributed Internet cache [46]. With this scheme, the upper level caches are replaced by directory servers. These servers contain the location hints about the documents which are kept at each cache. A hierarchical meta-data-hierarchy can be used to increase the efficiency and scalability of distribution of these location hints. Tewari *et al*. proposed a similar approach to employ a distributed caching system where the location hints are locally replicated at the institutional caches [9]. Wang introduced Cachemesh [47] that is a distributed cache system for the World Wide Web. In the Cachemesh, caching servers create a cache routing table among them, and then each server becomes the designed server for a number of the web sites. When the user requests are raised, they are forwarded to the proper caching server according to the cache routing table.

In the year 1998, a novel protocol, named Cache Digest [48], was proposed to optimize the communication among cooperative proxy caches. This technique allows proxy caches to make a summary of their contents in a compact format, called as digest. Each proxy cache can use this digest to identify which neighbors are likely to contain the requested web document. Other similar techniques to the cooperative web caching were proposed, for example, the Relais project [49] and Summary Cache protocol [50]. Each cache exchange messages indicating their content and also keep the local directories to find the documents in other caches. In the late 2010, the centrally managed cache cooperation architecture (CRISP) [51] was proposed to minimize the cost of communication among cooperative web caching proxies. With this CRISP, the cooperative caching proxies share their caches using a central mapping service with a complete directory of cached objects of participating caches. Unfortunately, the investigation of the reference [52] pointed out that this model causes the congestion over the network. So, the techniques of web caching and web prefetching were implemented in the proxy system to solve the problems of server load and congestion control [53].

2.3.3 Hybrid caching architecture

The hybrid caching architecture is the caching model combining the benefits of two cooperative caching models: the hierarchical and the distributed caching models. With this hybrid model, the caches cooperate with other caches at the same cache level or at a higher cache level using distributed caching as shown in Figure 2.7.

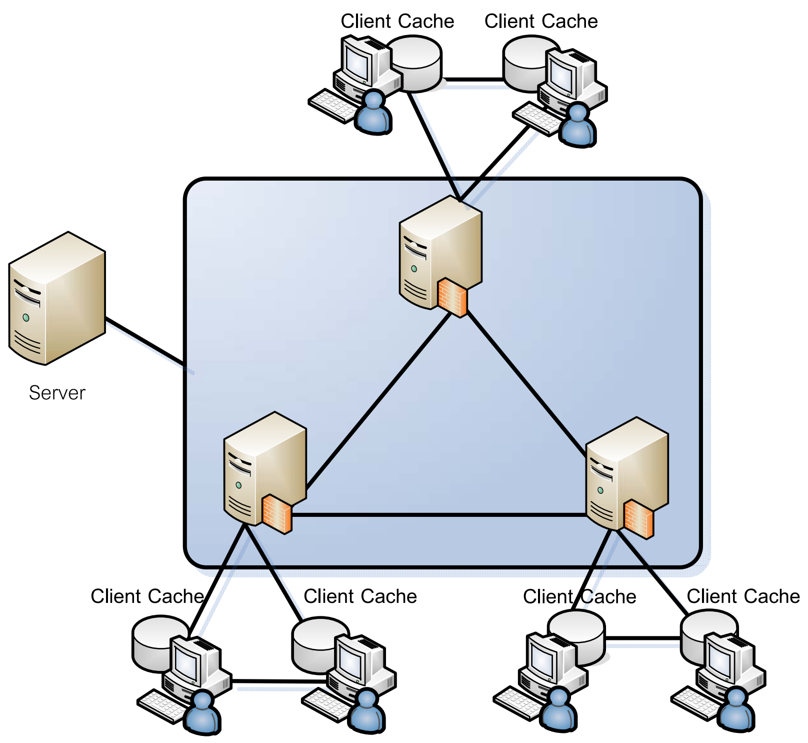


Figure 2.7: Hybrid caching architecture.

Some researches employ the hybrid caching model to improve the performance of cooperative web caching. Rabinovich *et al*. [54] proposed a technique to limit the cooperation among neighbor caches to avoid obtaining documents from remote caches. Additionally, Baek et al. [55] proposed one interesting hybrid model where the reference table was employed at each level in the hierarchy except the lowest level. Another solution in this model focused on some significant factors which are related to QoS such as communication between the caches and cache contents [11].

In order to illustrate the overall properties of three architectures of the cooperative caching proxies [33], Table 2.1 presents the comparison among hierarchical, distributed, and hybrid caching architectures as follows.

Table 2.1: Comparison among hierarchical, distributed, and hybrid caching architectures.

Features	Hierarchical	Distributed	Hybrid
Parent caches	congested	slight congestion	slight congestion
User latency	high	low	low
Connection time	short	long	long
Bandwidth usage	low	high	low
No.of Hierarchies	less than four	one	one - two
Transmission time	high	low	low
Network traffic	unevenly distributed	evenly distributed	evenly distributed
Disk space usage	Significant	low	low
Placement of caches in strategic locations	vital	not required	up to ISP
Freshness of cached contents	difficult	easy	easy
Hit ratio	high	very high	high - very high
Response time	moderate	fast	fast
Duplication of objects	high	low	low

2.4 Cache Replacement Policies

Since the cache replacement policy is a part of the effectiveness of proxy caches. Thus, the studies of cache replacement policies have been performed continually in order to improve various cost metrics, such as the hit ratio, the byte hit ratio, the response time, and the total cost. The cache replacement policy evicts the object stored in the cache to make a space for a new cached object when the cache is full due to a finite capacity of the cache. Several cache replacement policies are used to choose the object to be evicted or replaced. They can be classified into different categories as follows.

2.4.1 Recency-based policies

The primary factor of the recency-based policies is recency. The rationale of this category is that recently referenced objects will be referenced again in the near future.

- **Least Recently Used (LRU)** is the most popular policy that evicts or replaces the least recently referenced objects. With this policy, the cache will eventually fill with the most recently referenced objects.

- **Pitkow/Recker [32]** evicts the objects in the LRU order by considering the object's size. Then, the largest file sizes are removed first.

- **LRU-Threshold [56]** is similar to the traditional LRU, but the objects that are larger than a certain threshold size is not cached.

- **SLRU (Segmented LRU) [7]** partitions the cache into two segments called protected (stored the popular objects) and unprotected. When the object is requested for the first time, the object is inserted into the unprotected segment. When the object hit occurs in the unprotected segment, the object is moved to the protected segment. Moreover, both segments apply the LRU strategy. However, only the deletion of the objects is performed in the unprotected segment. If there is not enough space in the protected segment then the least recently used object is moved back as the most recently used object into the unprotected segment.

- **LRU* [57]** is the LRU variant. This policy inserts the referenced objects in a LRU list. The popularity indicator is a request counter counted from the hit of the referenced object, incrementing one per hit time and used as the hit count of each object. If the requested object is hit, the hit count is incremented by one and the cached object is moved to the beginning of the list. Additionally, the least recently used object is checked its hit count at each round of the replacement process. The object is evicted from the list if the hit count is equal to zero. Otherwise, the hit count is decreased by one and the object is moved to the beginning of the list.

2.4.1.1 The advantages of the recency-based policies

- These policies perform very well when web traffic exhibits high temporal locality. That means many users are interested in the same object at the same time.

- These policies are mostly popular due to simple implementation and practically good performance in various situations.

2.4.1.2 The disadvantages of the recency-based policies

- The size of the web is not considered in the LRU replacement process even though it is useful for balancing the load of the cached objects.

- In more static system environments, the frequency can be an important factor to make a decision in removing or caching the object.

2.4.2 Frequency-based policies

The primary factor of the frequency-based policies is the popularity of the object. The rationale of this category is that only a small set of objects is popular, and those objects should be cached.

- **Least Frequently Used (LFU)** is the simple policy that replaces the least frequently referenced objects. In the other hand, the cache implementing the LFU policy will be filled with the most frequently referenced objects.

- **HYPER-G [57]** combines three traditional replacement policies; they are the LFU, the LRU, and the SIZE. Whenever the cache needs the space to keep the new object, the LFU policy is used to make a decision in replacing the object first. Then the LRU performs. If there are several cached objects that cannot be chosen to replace. Finally, the SIZE is applied if the use of the LRU policy cannot determine which the cached object will be evicted.

- **swLFU [57]** applies both of the LFU and the LRU in the replacement process. This policy uses a weighted frequency counter as a factor to cache the object. The lowest weight w_i of the object i is replaced first. In the case that two or more objects have the same weighted frequency value, then the LRU is used to manage those objects. Moreover, the swLFU has one extension called Aged-swLFU or A-swLFU. The A-swLFU policy evicts the LRU-object on every k replacement.

- **LFU-Aging [7]** is a variant of the LFU avoiding the cache pollution of the LFU. In order to avoid this pollution, the LFU-Aging uses two parameters. The first parameter is to limit the average number of the requests for all cached objects. If the average number of the requests is over the limit, all request counters are divided by two, causing the inactive objects to lose their popularity. The second one is to limit the value of the request counters. Thus, it can control how long a previously popular object will stay in the cache.

- **LFU with Dynamic Aging (LFUDA) [58]** is similar to the LFU using a dynamic aging technique to avoid the cache pollution problem. The dynamic aging technique increments the cache age by one when the cached object is accessed. For this reason, the cache age of the recently popular objects becomes a larger value. Thus this can prevent previously popular objects from polluting the cache because the LFUDA evicts the object that has the smallest cache age.

2.4.2.1 The advantages of the frequency-based policies

- These policies are suitable for the static system environments such that the object popularity does not change much in the specific time period, such as a day or a week.

2.4.2.2 The disadvantages of the frequency-based policies

- The LFU-based policies require most complexity of cache management.

- The simple LFU policy causes the cache pollution. This situation occurs when the popular objects requested frequently during one time period can be stored in the cache even they are not requested for a long time period. That is because the LFU uses the frequency count as the factor for replacing the object in the cache.

- These kinds of replacement policies require other factors to break the replacement process when two or more objects have the same frequency count.

2.4.3 Size-based policies

The primary factor of the size-based policies is the size of the object. The rationale of this category is that most objects are small-size objects, thus if a large-size object is removed, the cache can be make room for multiple smaller ones.

- **LRU-MIN [59]** minimizes the number of replaced documents. The LRU-MIN evicts the least recently used object whose size is greater than *S*, the least recently used object whose size is greater than *S*/2, the least recently used object whose size is greater than *S*/4, and so on, where *S* is the size of incoming object. The LRU-MIN uses $\lfloor \log_2(document\,size) \rfloor$ as its primary key and the last accessing time as the secondary key, in the sense that the cache is partitioned into several size ranges and document removal starts from the group with the largest size range.

- **SIZE [57]** is the representative policy for this category. SIZE evicts the largest object first. In case, there are many objects with the same size stored in the cache, then the LRU is applied to those objects.

- **Largest File First (LFF)** evicts the largest object to make the most space in the cache for the new ones. Additionally, both size and age of the object are frequently considered in choosing the object to be replaced.

2.4.3.1 The advantages of the size-based policies

- These strategies work very well if the largest object is less popular.

2.4.3.2 The disadvantages of the size-based policies

- The SIZE policy emphasizes too much on the object size. It should be implemented with some techniques, such as adding a priority queue or limiting the time of cached objects, to improve the cache hit. The LRU-MIN is also focused more on the size of the cached objects even it takes advantage of the LRU policy to replace the object.

2.4.4 Function-based policies

Function-based policies are key-based policies using multiple combined keys without sequential order among those keys. Each key can be differently weighted in the cost function. Moreover, the most valuable object will be retained in the cache considering the cost function defined in the different way. Generally, the object with the smallest value is evicted first. The rationale of this category is that more factors considered together can help to achieve a higher hit ratio.

- **GreedyDual-Size (GDS) [5]** assigns a value of benefit to each object. Initially, a value is set to cost/size. The object with the smallest value of benefit is evicted first. The object's value of benefit is decreased when the object is removed from the cache while this value is increased to the original value when the object is hit.

- **GDS with Frequency (GDSF) [58]** is a variant of the GDS that considers the frequency count of the referenced object as well. The GDSF optimizes the object hit ratio by storing the popular smaller objects in the cache. A value of benefit is assigned as the object's frequency count divided by its size, plus the cache age factor used to limit the influence of previously popular documents, as described above for LFUDA.

- **GD* [7]** is the extended GDS, called as GD* or GDSP algorithm proposed by Jin and Bestavros [61]. However, the performance of web caching algorithms in relation to the type of document being cached had been studied that the GD* works best for the image, the HTML and the multimedia documents comparing to the LRU, the LFU-DA, and the GDS algorithms [62].

- **LUV (Least-Unified Value) [7]** is proposed by Bahn *et al*. [63]. The LUV uses the reference potential of an object as a factor. The reference potential is estimated by using a function that relates the reference potential of an object to its most recent reference. For example, a function is $f(x) = \left(\frac{1}{2}\right)^{\lambda x}$, with $0 \le \lambda < 1$, where *x* measures the time until the last reference. However, the problem of this policy is to choose the right choice of λ.

2.4.4.1 The advantages of the function-based policies

- There is no fixed combination of factors, resulting in no bias for many objects. According to the proper alternative of weighting factors, one of factors can optimize any performance metric.

- A different number of factors are chosen to handle the different situations of the workloads.

2.4.4.2 The disadvantages of the function-based policies

- The challenging task of this category is to choose appropriate weights of factors. Some reviews assume the weights from the study of the web trace. Although this assumption is simple, it is an error prone approach. That is because the workloads of the webs always change and require some adaptive setting of the factors. However, this adaptive setting adds the complexity to the replacement process.

- The latency should not be used in the value calculation. For this reason, it can introduce inaccurate latency estimation. The factors influencing to the latency are such as many components on the path between the web server and the proxy server or the client, new technologies supporting the object movement, etc. Therefore, using latency can lead to improper replacement decisions.

2.4.5 Randomized policies

The eviction decisions of these policies have no require complex data structures and high computation overhead.

- RAND [57] evicts the object randomly from the cache. The RAND policy requires no state information, causing to save both memory and processing power. However, the RAND uses only simple random function based on equal probability for each object. Thus it does not perform well.

- HARMONIC [60] improves the performance using a non-uniform probability distribution. Each object has a probability inversely proportional to its specific cost that is cost/size. The object with lower costs is evicted first. However, this policy requires a simple data structure in order to keep the cost value of objects in the cache.

2.4.5.1 The advantages of the randomized policies

- These randomized policies do not require special data structures of the object's insertion and deletion while the object's search can be supported by special data structures.

- The implementation of these policies is simple.

2.4.5.2 The disadvantages of the randomized policies

- The evaluation of these policies is unwieldy. For example, the obtained results from the different simulations with the same web trace are slightly different.

In fact, the number of users over the Internet via web usages is rapidly increasing and causes a high delay for every browse since the amount of browsed content cannot be served and saved in the proxy's cache. Thus, various researches have proposed a new caching algorithm to manage contents in the cache even though many cache replacement policies are implemented before, such as the GDS [5], the GD* [6], etc. Moreover, the workload over the caching strategy had been considered by Haverkort *et al*. [7]. This study leaded to the development of the Class-Based Least-Recently Used (C-LRU), which is a refinement of standard LRU. The C-LRU manages the cache by partitioning it into classes; each one is reserved for objects of a particular size range and the LRU algorithm is applied within a class. However, some algorithms of document replacing consider more than one factor into their scheme to improve caching performance. In the late 2005, Frequency REcency and Size Cache Replacement (FRES-CAR) [64] considering the document reference recency, frequency, and size was proposed. Moreover, Bian and Chen introduced the Least Grade Replacement (LGR) [65] applying the perfect-history into account for web cache optimization. Another replacement policy is named as Semantic and Least Recently Used (SEMALRU) [66]. The SEMALRU evicts objects that are less related to an incoming object or least recently used object. Thus, only related objects are stored in the cache. Although, various new replacement strategies were implemented to enhance the caching performance, mostly encourages the locality problem. Thus, Dump and Clear [67] that was presented to solve the locality problem based on Petersen Graph [68] topology.

2.5 Recommender System

A recommender system (RS) is a customization tool in an e-commerce system.

The RS generates the personalized recommendations that match with the taste of the users [69]. The goal of the RS is to generate the significant recommendations to a group of users for items or products that those users might be interested in them. Additionally, the RS achieves in solving the problem of information overloading; it provides users with more proactive and personalized information services. Generally, two basic recommendation approaches used in the RS are Content-based (CB) and Collaborative Filtering (CF) approaches.

2.5.1 Content-based (CB) approach

The CB approach recommends the items that are similar to the ones that the user liked in the past or matched to attributes of the user. The similarity of items is calculated based on the features associated with the compared items. For example, if a movie belonging to the comedy genre is positively rated by a user, then the system can learn to recommend other movies from this genre [70].

2.5.2 Collaborative Filtering (CF) approach

The CF approach recommends to the user the items that other users with similar tastes liked in the past [71]. The taste similarity of two users is calculated based on the similarity in the rating history of the users. Thus, the CF approach is called as "people-to-people correlation" [72]. Moreover, this approach is the most popular and widely implemented technique in the recommender system [73-74].

Since the development of the Recommender System (RS) in the area of information retrieval is widely implemented in the information search algorithm to shorten down the search time, the METIOREW technique was implemented for web search mechanism [75]. This technique applied the document evaluation results from the users as the "intelligent bookmark" to find the most relevant web pages under search topics. On the other hand, Amazon.com system uses the Item-to-Item Collaborative Filtering [76] as its recommender system. As same as Amazon.com, RS is extremely implemented in many internet activities and services, such as course management systems [77], e-learning system [78-79], etc. Furthermore, the modification of the RS is applied on some products or service systems to improve the efficiency of the RS. For example, the RS with the fuzzy scatter difference was presented [80]. Additionally, the time contexts and group preferences are used to improve the customer profile in collaborative systems [81].

CHAPTER III

DATA COLLECTION AND ANALYSIS

This chapter describes the data collection method and the data analysis method. The data used in this research is the Squid log file, named access.log. The data collection section presents Squid Log Format Overview, Data Sources, and Log File Filtering. The data analysis is the grouping mechanism which includes Web directory-based grouping and Browsing behavior-based grouping. They are elaborated as follows.

3.1 Data Collection

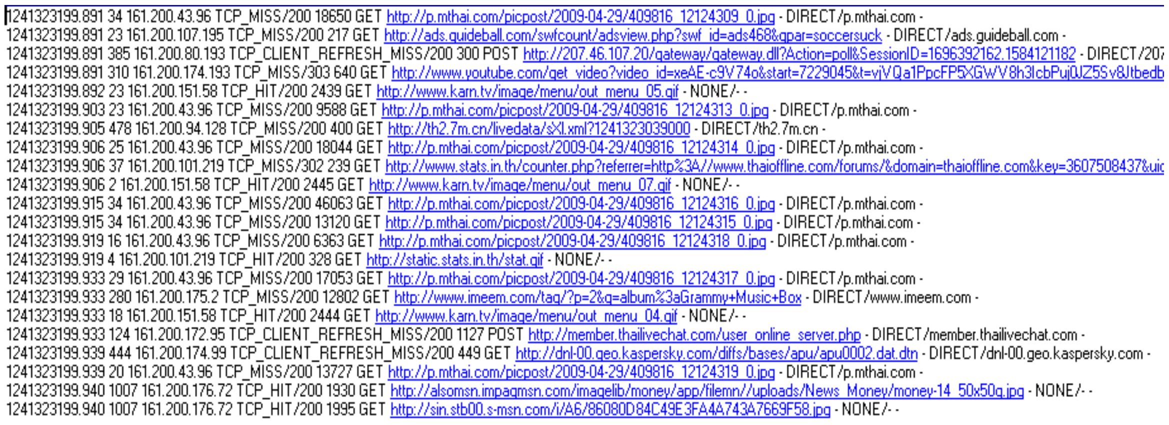
3.1.1 Squid Log Format Overview

Since the Squid is the general proxy caching server, especially for academic area [24], therefore in this research will analyze the access.log, a type of log file of Squid, in order to identify suitable groups of webs requested by users. Figure 3.1 shows the standard access.log format; its usage is "*logformat <name> <format specification>*", where *<name>* is the name of the log format and *<format specification>* is a string with embedded % format codes [82]. The default formats available in Squid are *squid*, *squidmime*, *common*, and *combined*. However, this research used the "*squid*" format that is the most used one of default formats shown as below:

$$\textit{logformat}\ \overbrace{\textit{squid}}^{\textit{<name>}}\ \overbrace{\textit{\%ts.\%03tu \%tr \%>a \%Ss/\%03Hs \%<st \%rm \%ru \%un \%Sh/\%<A \%mt}}^{\textit{<format specification>}}$$

Figure 3.1: The *squid* log format.

Referring to Figure 3.2, each record in the access.log file usually consists of (at least) 10 columns separated by one or more spaces. For example, the access.log record is "1241323199.891 34 161.200.43.96 TCP_MISS/200 18650 GET http://p.mthai.com/picpost/2009-04-29/409816_12124309_0.jpg - DIRECT/p.mthai.com - ”. The details of this example record are described as demonstrated in Table 3.1.



```
1241323199.891 34 161.200.43.96 TCP_MISS/200 18650 GET http://p.mthai.com/picpost/2009-04-29/409816_12124309_0.jpg - DIRECT/p.mthai.com -
1241323199.891 23 161.200.107.195 TCP_MISS/200 217 GET http://ads.guideball.com/swfcount/adsview.php?swf_id=ads468&gpar=soccersuck - DIRECT/ads.guideball.com -
1241323199.891 385 161.200.80.193 TCP_CLIENT_REFRESH_MISS/200 300 POST http://207.46.107.20/gateway/gateway.dll?Action=poll&SessionID=1696392162.1584121182 - DIRECT/207
1241323199.891 310 161.200.174.193 TCP_MISS/303 640 GET http://www.youtube.com/get_video?video_id=xeAE-c9V74o&start=7229045&t=vjVQa1PpcFP5XGWV8h3IcbPuj0JZ5Sv8Jtbedb
1241323199.892 23 161.200.151.58 TCP_HIT/200 2439 GET http://www.karn.tv/image/menu/out_menu_05.gif - NONE/- -
1241323199.903 23 161.200.43.96 TCP_MISS/200 9588 GET http://p.mthai.com/picpost/2009-04-29/409816_12124313_0.jpg - DIRECT/p.mthai.com -
1241323199.905 478 161.200.94.128 TCP_MISS/200 400 GET http://th2.7m.cn/livedata/sXl.xml?1241323039000 - DIRECT/th2.7m.cn -
1241323199.906 25 161.200.43.96 TCP_MISS/200 18044 GET http://p.mthai.com/picpost/2009-04-29/409816_12124314_0.jpg - DIRECT/p.mthai.com -
1241323199.906 37 161.200.101.219 TCP_MISS/302 239 GET http://www.stats.in.th/counter.php?referrer=http%3A//www.thaioffline.com/forums/&domain=thaioffline.com&key=3607508437&uic
1241323199.906 2 161.200.151.58 TCP_HIT/200 2445 GET http://www.karn.tv/image/menu/out_menu_07.gif - NONE/- -
1241323199.915 34 161.200.43.96 TCP_MISS/200 46063 GET http://p.mthai.com/picpost/2009-04-29/409816_12124316_0.jpg - DIRECT/p.mthai.com -
1241323199.915 34 161.200.43.96 TCP_MISS/200 13120 GET http://p.mthai.com/picpost/2009-04-29/409816_12124315_0.jpg - DIRECT/p.mthai.com -
1241323199.919 16 161.200.43.96 TCP_MISS/200 6363 GET http://p.mthai.com/picpost/2009-04-29/409816_12124318_0.jpg - DIRECT/p.mthai.com -
1241323199.919 4 161.200.101.219 TCP_HIT/200 328 GET http://static.stats.in.th/stat.gif - NONE/- -
1241323199.933 29 161.200.43.96 TCP_MISS/200 17053 GET http://p.mthai.com/picpost/2009-04-29/409816_12124317_0.jpg - DIRECT/p.mthai.com -
1241323199.933 280 161.200.175.2 TCP_MISS/200 12802 GET http://www.imeem.com/tag/?p=2&q=album%3aGrammy+Music+Box - DIRECT/www.imeem.com -
1241323199.933 18 161.200.151.58 TCP_HIT/200 2444 GET http://www.karn.tv/image/menu/out_menu_04.gif - NONE/- -
1241323199.933 124 161.200.172.95 TCP_CLIENT_REFRESH_MISS/200 1127 POST http://member.thailivechat.com/user_online_server.php - DIRECT/member.thailivechat.com -
1241323199.939 444 161.200.174.99 TCP_CLIENT_REFRESH_MISS/200 449 GET http://dnl-00.geo.kaspersky.com/diffs/bases/apu/apu0002.dat.dtn - DIRECT/dnl-00.geo.kaspersky.com -
1241323199.939 20 161.200.43.96 TCP_MISS/200 13727 GET http://p.mthai.com/picpost/2009-04-29/409816_12124319_0.jpg - DIRECT/p.mthai.com -
1241323199.940 1007 161.200.176.72 TCP_HIT/200 1930 GET http://alsomsn.impagmsn.com/imagelib/money/app/filemn//uploads/News_Money/money-14_50x50q.jpg - NONE/- -
1241323199.940 1007 161.200.176.72 TCP_HIT/200 1995 GET http://sin.stb00.s-msn.com/i/A6/86080D84C49E3FA4A743A7669F58.jpg - NONE/- -
```

Figure 3.2: Examples of the browsing records stored in the access.log file.

Table 3.1: The detailed access.log record.

Column name	Format codes	Descriptions	Example
timestamp	*%ts.%03tu*	A UNIX timestamp as UTC seconds with a millisecond resolution	1241323199.891
time elapse	*%tr*	The elapsed time or response time (milliseconds)	34
source IP	*%>a*	The IP address of the requesting instance, the client source IP address	161.200.43.96
function/status_code (result code)	*%Ss/%03Hs*	Squid request status (TCP_HIT, TCP_MISS, etc) / HTTP status code (200, 302, etc)	TCP_MISS/200 (The message means a valid copy of the requested object was not in the cache and successful actions)
file size (bytes)	*%<st*	The size is the amount of data delivered to the client including HTTP headers	18650
transferred/request method	*%rm*	The request method to obtain an object (GET, POST, etc)	GET
destination URL	*%ru*	This column contains the URL requested	http://p.mthai.com/picpost/2009-04-29/409816_12124309_0.jpg
rfc931	*%un*	The ident lookups (User name) for the requesting client. If no ident information is available, a "-" will be logged	-
peer status/peer host (hierarchy code)	*%Sh/%<A*	Squid hierarchy status (DIRECT, DEFAULT_PARENT, etc) / Server IP address or peer name	DIRECT/p.mthai.com
content type of the object	*%mt*	The content type of the object as seen in the HTTP reply header. If the object don't have any content type, and thus are logged "-"	-

3.1.2 Data Sources

The study of web browsing behavior is performed using sampling data from the oldest and the largest university of Thailand named Chulalongkorn University, henceforth called “Chula”. Since this university provides various branches of study programs and researches to serve societies both in national and international, there will be varieties of data to be retrieved over the Internet. Therefore, various browsed information, various languages, and varieties of locations are existed from the requests. Additionally, these requests of the Internet services must pass the proxy server and transactions will be recorded in the access.log. As a consequence, these transactions, or browsed information, can be applied as the input data set in the behavioral analysis process over the Internet. Based on the access.log data obtained from the University Office of Information Technology of Chula, collected from January to August 2009, is 1.5 TB (compressed files); the number of transactions is approximately 50 million transactions per day.

In addition, there is a volunteer university; Nakhon Pathom Rajabhat University (NPRU), by Computer Center had provided 2 months information during October to November 2010, in the volume of 8 GB. This data set that is 56,762,933 transactions is used as the experiment data set after the grouping condition is defined.

3.1.3 Log File Filtering

After the access.log data are analyzed, the next process is to clean or filter the irrelevant information and fields in the log file. Based on the format of the access.log, there are many fields being stored, such as the retrieval time or the timestamp, the file size, the destination URL, and access situation (hit/miss), etc. However, this research uses only two fields: the destination URL, and the file size. Considering the destination URL field, the URLs considered in the analysis procedure are only the host name, not full path name of the URL. As illustrated in Table 3.1, the browsed URL is http://p.mthai.com/picpost/2009-04-29/409816_12124309_0.jpg, thus, the host name “p.mthai.com” is used. The popularity indicator is a counter counted from the existing of the URL host name, incrementing one per existing time and used as the browse frequency of each URL. This method is called as the frequency-based analysis. The consequence result of this method is the “Frequency Pattern (FP)”; this FP shows the popularity of webs in a certain period of time.

Nevertheless, the information contained in the destination URL field does not only show the host name, but also be used to identify the type of webs implicitly. Generally, the types of webs can be classified as static and dynamic webs [83]. The dynamic webs will be specified using the specific symbols; “cgi-bin”, a question mark “?”, or a suffix “.cgi” contained in the URL string. Thus, in this research, the dynamic web will be assigned with special character to differentiate from the static web.

It is the fact that the value of “file size” is varied based on the type of webs (dynamic or static), status of web servers, or traffic conditions during the called web is delivered. Thus, in this research, the average number of “file size” of each unique URL is calculated and be used as the representative of the web size before plotting the histogram of retrieved “file size” to see distribution, using all records in the access.log. This result is to indicate the cache load as needed. The consideration based on the web size is called as “Loading Size Pattern (LSP)”, and this value can be applied as a weighting value in the architecture designing process of the cache farm.

By conclusion, the meaningful information that is extracted from the access.log to analyze the browsing behavior of users will be stored in two main databases; they are the web identification database (WIDB) and the web classification database (WCDB), for managing the caches in the farm. The information structures of both the WIDB and the WCDB are shown in Table 3.2 and Table 3.3, respectively as follows:

Table 3.2: The WIDB structure.

Column Name	Data Type	Description
URLText	varchar	The host name of the URL
BeginDate	date	The first time of date that the URL is stored
UpdateDate	date	The last time of date that the information is updated
URLType	varchar	The type of webs: static or dynamic
Cnt_W	integer	The weekly counter
URLActCode	varchar	The active code: active or inactive
AccumFreq	integer	The accumulated frequency of each URL during a week
AccumSize	integer	The accumulated file size of each URL during a week
Freq	integer	The browse frequency of the URL (per a week)
Size	integer	The average size of the URL (per a week)
GroupCode	varchar	The group code
Primary key is URLText		
Foreign key are URLType, GroupCode		

Referring to Table 3.2, the URL host name is represented by the URLText. When the access.log data was filtered, the meaningful information will be stored in the WIDB for future process of web classification. The new URL is specified BeginDate and UpdateDate with the present date of the first existing time in the database. Then, the UpdateDate will be updated for every weekend when the RAM performs. As mentioned previously, the types of browsed webs (URLType) consisting of static and dynamic will be assigned in different characters. Thus, the static web is identified by "0" while the dynamic web is assigned with the character "1".

In order to follow the browses of the users for every week, the weekly counter (Cnt_W) is defined that is running from 0 (zero) to 4. Every new URL stored in the WIDB will be assigned the Cnt_W value with 0 (zero), and then incremented by one at every weekend until the Cnt_W value is equal to 4, the fourth week, that is the maximum value before re-counting in the next month. Furthermore, the active code (URLActCode) to recognize whether the URL is the active URL is used together with the Cnt_W. The URLActCode is also evaluated for every weekend as same as the Cnt_W. The URLActCode consists of four digits initiated with “0000” that represents the four-week action of the URL; the value of each digit is either “0” or “1”. The value “0” stands for the inactive URL while the active URL is referred with the value “1”. If the URL is active that means the URL is requested at least one time during a week, the URLActCode will be updated with “1” at the *n*th position, where *n* is Cnt_W value. Otherwise, the URLActCode is the original value at the initial state. For example, once new URL is inserted in the WIDB, the URLActCode of this new URL is assigned as “1000” at the first weekend. If this new URL still be browsed in the second week, the URLActCode will be updated with “1” at the *2*th position, where *2* is the Cnt_W value; the URLActCode is identified with “1100”. On the other hand, the URLActCode will be “1000” if there is no browse of this new URL during the second week.

As a consequence of filtering the access.log data, the frequency and the size of each retrieved URL are available. Both of the frequency and the size of the web are accumulating counted week by week; the former is indicated in the AccumFreq field while the latter is indicated in the AccumSize field. Moreover, the AccumFreq value is used as the real browse frequency (Freq) whereas the average file size of retrieved web (Size) can be calculated from the AccumSize value divided by the AccumFreq value. However, the group code (GroupCode) of the classified URLs is indicated in the WIDB after the web classification process is performed using the Freq value and the Size value as indicators.

Table 3.3: The WCDB structure.

Column Name	Data Type	Description
GroupCode	varchar	The group code
GroupName	varchar	The group name
GrFreq	integer	The average frequency of the browsed webs in each group (per a week)
StdFreq	integer	The standard deviation of frequency of the browsed webs in each group (per a week)
GrSize	integer	The average file size of the browsed webs in each group (per a week)
StdSize	integer	The standard deviation of web size of the browsed webs in each group (per a week)
UpdateDate	date	The last time of date that the information is updated
Primary key is GroupCode		

Referring to Table 3.3, each group of web patterns is named by the group code (GroupCode) which is running from "1", "2", and so on; so that it depends on the number of grouping criteria. If the number of the grouping criteria is equal to four, the group code will be stating at "1" and ending at "4".

In this research, the WCDB contains the boundary value of the browse frequency, $bv_F(i)$, and the boundary value of the web size, $bv_S(i)$, for every group (i) that will be performed for every weekend. Moreover, all boundary values, $bv_F(i)$ and $bv_S(i)$, are applied to the web classification process. The $bv_F(i)$ is derived from the average frequency of the browsed webs (GrFreq) and the standard deviation of frequency of the browsed webs (StdFreq) of group (i). The $bv_S(i)$ is derived from the average file size of the browsed webs (GrSize) and the standard deviation of web size of the browsed webs (StdSize) of group (i). When the boundary values contained in this WCDB are updated with the new values, the value in the UpdateDate field is up to date as the present date.

However, all values, GrFreq, GrSize, StdFreq, and StdSize, contained in the WCDB are derived from the equations shown in Table 3.4.

Table 3.4: Equations used in the WCDB.

Term	Equation
The average frequency for group *i* : *GrFreq(i)*	$GrFreq(i) = \frac{\sum_{m \in N} f_m(i)}{\sum_{m \in N} r_m(i)}$
The average file size for group *i* : *GrSize(i)*	$GrSize(i) = \frac{\sum_{m \in N} s_m(i)}{\sum_{m \in N} r_m(i)}$
The standard deviation of browse frequency for group *i* : *StdFreq(i)*	$StdFreq(i) = \sqrt{\frac{\sum_{m \in N} (f_m(i) - GrFreq(i))^2}{(\sum_{m \in N} r_m(i) - 1)}}$
The standard deviation of file size for group *i* : *StdSize(i)*	$StdSize(i) = \sqrt{\frac{\sum_{m \in N} (s_m(i) - GrSize(i))^2}{(\sum_{m \in N} r_m(i) - 1)}}$
Notation: *i* = the *i*th of classified group f_m = the browse frequency for Web m s_m = the web size for Web m r_m = the total number of requests for Web m N = set of all browsed Webs	

3.2 Data Analysis : Grouping Mechanism

As the fact that the cache management is the significant issue of all Internet Service Providers. Therefore, the content-grouping criterion is an important factor to achieve the performance of Proxy servers. Although the research of [84] had proposed the group fetching while the research of [12] the record in cache will be evicted based on LRU policy under each containing group. Thus, this research will consider the combination of these two mechanisms by finding the suitable grouping criterion and time for group fetching.

3.2.1 Web Directory-based Grouping

Generally, users search their desired information by looking at the default groups that appeared on the search engine screen, such as entertainment, sports, computers, books, etc. However, there is no evidence to support that the grouping technique of the proxy cache is implemented based on this standard knowledge of human being. The group-based management of file cache was proposed [84] by grouping the files that have possibility to be accessed together. This method is based on the predicting model while another grouping technique proposed in [12] used content of the browsed information as the primary grouping condition. However, this grouping mechanism is applied only in a small faculty; therefore the content-based grouping is possible to be managed by the department-IP.

In this research, after considering the data set obtaining from the University Office of Information Technology of Chula, the concept of Internet Innovation Research Center Co., Ltd., Thailand, the owner of the Truehits.net had been applied. In the year 2009, this company had classified user requirements into 19 groups, named Thailand Web Directory at Truehits.net (http://directory.truehits.net/). Since Web directories are directories where organize Web sites by subject, and is usually maintained by humans instead of software [85]. Using Web directories, the chance that web pages will be classified in the wrong categories is low due to viewing and checking the pages by humans [86].

According to the advantage of Web directory, this research used Thailand web directory at Truehits.net which are business, computer, internet, shopping, travel, mobile, real estate, entertainment, news, games, finance, car, sports, government, health, art, education, person, and adult as presented in Table 3.5, in order to match the webs to the right groups. However, under each group, there are subgroups that contain more specific information where users can identify their needs when browse. For example, Table 3.6 – Table 3.7 show the subgroups of the business group and the travel group, respectively.

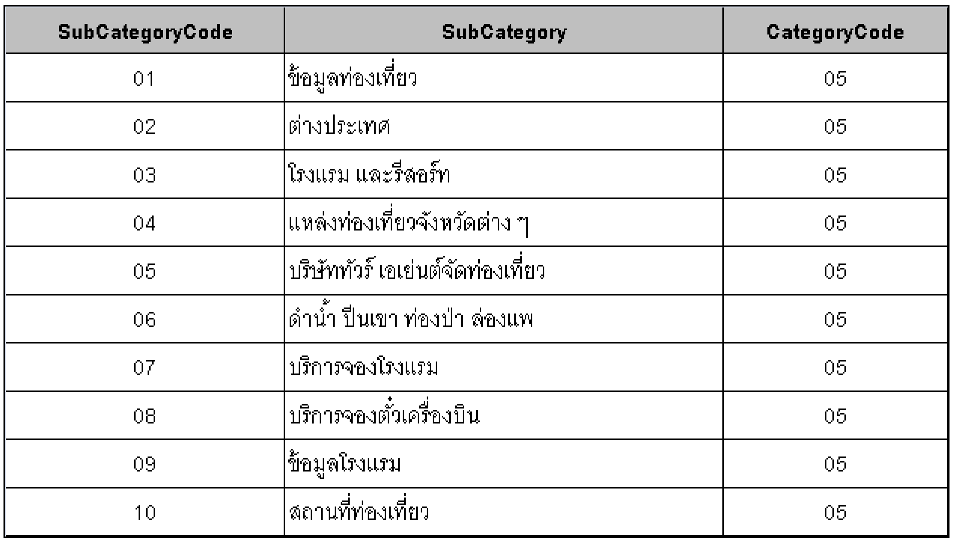
Table 3.5: Thailand Web Directory at Truehits.net.

CategoryCode	Category
01	Business
02	Computer
03	Internet
04	Shopping
05	Travel
06	Mobile
07	Realestate
08	Entertainment
09	News
10	Games
11	Finance
12	Car
13	Sports
14	Government
15	Health
16	Art
17	Education
18	Person
19	Adult

Table 3.6: The subgroups of the business group.

SubCategoryCode	SubCategory	CategoryCode
01	เครื่องใช้สำนักงาน	01
02	โฆษณา ศิลปะ บันเทิง และภาพยนตร์	01
03	อื่นๆ	01
04	ธุรกิจขนาดเล็ก และขนาดกลาง (SME)	01
05	ธุรกิจขายตรง (MLM)	01
06	นำเข้า,ส่งออก,ขนส่งสินค้า	01
07	เครื่องใช้ไฟฟ้า	01
08	สมัคร และจัดหางาน	01
09	สื่อสาร อิเล็กทรอนิกส์ โทรคมนาคม	01
10	พลังงาน (น้ำมัน ไฟฟ้า ก๊าซธรรมชาติ)	01
11	สายการบิน	01
12	อีคอมเมิร์ซ	01
13	ธุรกิจอาหารและฟาสต์ฟู้ด	01
14	เครื่องปรับอากาศ	01
15	อุปกรณ์อิเล็กทรอกนิกส์,ไฟฟ้า	01
16	การตลาดแบบเครือข่าย ทำงานจากที่บ้าน	01
17	หารายได้จากการเล่นอินเทอร์เน็ต	01
18	งานแสดงสินค้า งานประชุม งานเทศกาล	01
19	อุตสาหกรรมการผลิต	01
20	ผู้ให้บริการโทรคมนาคม ผู้จำหน่าย	01
21	ที่ปรึกษา คอนซัลท์	01
22	บริการทางโทรศัพท์	01
23	สิ่งพิมพ์,สำนักพิมพ์,เกี่ยวกับงานพิมพ์	01
24	แฟรนไชส์	01
25	ร้านอาหาร,ห้องจัดเลี้ยง,บาร์	01
26	บริษัทโฮลดิ้ง	01
27	จัดซื้อจัดจ้าง	01
28	เครื่องจักร เครื่องกล	01
29	ผลิตภัณฑ์	01
30	อุปกรณ์ถ่ายภาพ ฉายภาพ โสต	01
31	ออกแบบตกแต่ง	01
32	การเกษตร	01
33	เกี่ยวกับกฎหมาย	01
34	ระบบบัญชี	01

Table 3.7: The subgroups of the travel group.



SubCategoryCode	SubCategory	CategoryCode
01	ข้อมูลท่องเที่ยว	05
02	ต่างประเทศ	05
03	โรงแรม และรีสอร์ท	05
04	แหล่งท่องเที่ยวจังหวัดต่าง ๆ	05
05	บริษัททัวร์ เอเย่นต์จัดท่องเที่ยว	05
06	ดำน้ำ ปีนเขา ท่องป่า ล่องแพ	05
07	บริการจองโรงแรม	05
08	บริการจองตั๋วเครื่องบิน	05
09	ข้อมูลโรงแรม	05
10	สถานที่ท่องเที่ยว	05

In summary, the overview of Thailand Web Directory at Truehits.net is shown in Table 3.8.

Table 3.8: Overview of Thailand Web Directory at Truehits.net.

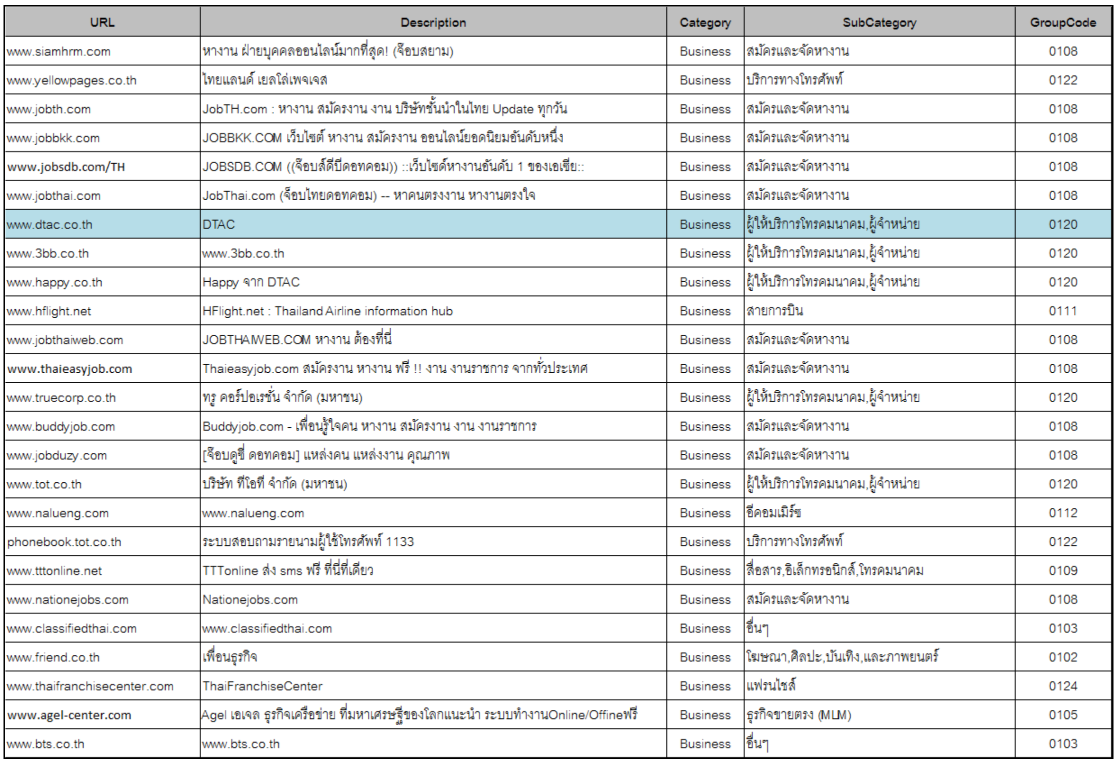
CategoryCode	Category	The number of subcategories
01	Business	34
02	Computer	10
03	Internet	20
04	Shopping	24
05	Travel	10
06	Mobile	6
07	Realestate	4
08	Entertainment	13
09	News	10
10	Games	12
11	Finance	12
12	Car	8
13	Sports	18
14	Government	34
15	Health	10
16	Art	10
17	Education	26
18	Person	16
19	Adult	6

Referring to Table 3.9, some examples of the URLs contained in Thailand Web Directory at Truehits.net are listed according to their groups and subgroups. The group of each URL is denoted by the GroupCode. The GroupCode consists of four digits; the first two digits refer to the category code and the last two digits refer to the subcategory code. The matching method is searching each URL, the host name filtered from the log record, along with the categories including the subcategories. Therefore, the

GroupCode of the URL is assigned when the searching URL can be matched to a web directory.

For example, the searching URL is “www.dtac.co.th” that is found in the subgroups, called “ผู้ให้บริการโทรคมนาคม ผู้จำหน่าย”, under the business group, then the group of the searching URL is defined by GroupCode “0120”, where “01” is the business group code (Table 3.5) and “20” is the subgroup code of “ผู้ให้บริการโทรคมนาคม ผู้จำหน่าย” (Table 3.6).

Table 3.9: Examples of URLs contained in Thailand Web Directory at Truehits.net.



URL	Description	Category	SubCategory	GroupCode
www.siamhrm.com	หางาน ฝ่ายบุคคลออนไลน์มากที่สุด! (จ๊อบสยาม)	Business	สมัครและจัดหางาน	0108
www.yellowpages.co.th	ไทยแลนด์ เยลโล่เพจเจส	Business	บริการทางโทรศัพท์	0122
www.jobth.com	JobTH.com : หางาน สมัครงาน งาน บริษัทชั้นนำในไทย Update ทุกวัน	Business	สมัครและจัดหางาน	0108
www.jobbkk.com	JOBBKK.COM เว็บไซต์ หางาน สมัครงาน ออนไลน์ยอดนิยมอันดับหนึ่ง	Business	สมัครและจัดหางาน	0108
www.jobsdb.com/TH	JOBSDB.COM ((จ๊อบส์ดีบีดอทคอม)) ::เว็บไซต์หางานอันดับ 1 ของเอเชีย::	Business	สมัครและจัดหางาน	0108
www.jobthai.com	JobThai.com (จ๊อบไทยดอทคอม) -- หาคนตรงงาน หางานตรงใจ	Business	สมัครและจัดหางาน	0108
www.dtac.co.th	DTAC	Business	ผู้ให้บริการโทรคมนาคม,ผู้จำหน่าย	0120
www.3bb.co.th	www.3bb.co.th	Business	ผู้ให้บริการโทรคมนาคม,ผู้จำหน่าย	0120
www.happy.co.th	Happy จาก DTAC	Business	ผู้ให้บริการโทรคมนาคม,ผู้จำหน่าย	0120
www.hflight.net	HFlight.net : Thailand Airline information hub	Business	สายการบิน	0111
www.jobthaiweb.com	JOBTHAIWEB.COM หางาน ต้องที่นี่	Business	สมัครและจัดหางาน	0108
www.thaieasyjob.com	Thaieasyjob.com สมัครงาน หางาน ฟรี !! งาน งานราชการ จากทั่วประเทศ	Business	สมัครและจัดหางาน	0108
www.truecorp.co.th	ทรู คอร์ปอเรชั่น จำกัด (มหาชน)	Business	ผู้ให้บริการโทรคมนาคม,ผู้จำหน่าย	0120
www.buddyjob.com	Buddyjob.com - เพื่อนรู้ใจคน หางาน สมัครงาน งาน งานราชการ	Business	สมัครและจัดหางาน	0108
www.jobduzy.com	[จ๊อบดูซี่ ดอทคอม] แหล่งคน แหล่งงาน คุณภาพ	Business	สมัครและจัดหางาน	0108
www.tot.co.th	บริษัท ทีโอที จำกัด (มหาชน)	Business	ผู้ให้บริการโทรคมนาคม,ผู้จำหน่าย	0120
www.nalueng.com	www.nalueng.com	Business	อีคอมเมิร์ซ	0112
phonebook.tot.co.th	ระบบสอบถามรายนามผู้ใช้โทรศัพท์ 1133	Business	บริการทางโทรศัพท์	0122
www.tttonline.net	TTTonline ส่ง sms ฟรี ที่นี่ที่เดียว	Business	สื่อสาร,อิเล็กทรอนิกส์,โทรคมนาคม	0109
www.nationejobs.com	Nationejobs.com	Business	สมัครและจัดหางาน	0108
www.classifiedthai.com	www.classifiedthai.com	Business	อื่นๆ	0103
www.friend.co.th	เพื่อนธุรกิจ	Business	โฆษณา,ศิลปะ,บันเทิง,และภาพยนตร์	0102
www.thaifranchisecenter.com	ThaiFranchiseCenter	Business	แฟรนไชส์	0124
www.agel-center.com	Agel เอเจล ธุรกิจเครือข่าย ที่มหาเศรษฐีของโลกแนะนำ ระบบทำงานOnline/Offineฟรี	Business	ธุรกิจขายตรง (MLM)	0105
www.bts.co.th	www.bts.co.th	Business	อื่นๆ	0103

In order to prove the performance of Thailand Web Directory at Truehits.net, some data sets are used as the data test set; the number of transactions is 2,669,656,230 transactions of 2,911,467 unique URLs. The result of applying these 19 main groups based on the definition of Truehits.net is that there are 62.86% of URLs (1,830,065 unique URLs) that cannot be identified their group because these URLs are not match to the defined categories. Thus, the defined groups of Truehits.net are not suitable for URL cache grouping management.

3.2.2 Browsing Behavior-based Grouping

As a consequence of the Web directory-based grouping that had been tested, all records of access.log are analyzed. Initially, the access.log files collected from January - June 2009 are used, and then the daily browsing behavior and the weekly browsing behavior are plotted shown in Figure 3.3 and Figure 3.4, respectively, to consider the patterns of browsing behavior.

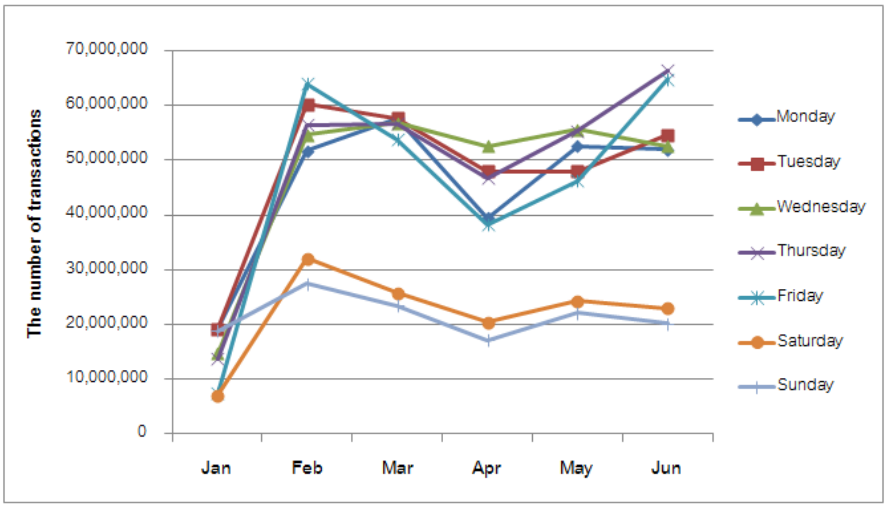


Figure 3.3: The daily browsing behavior.

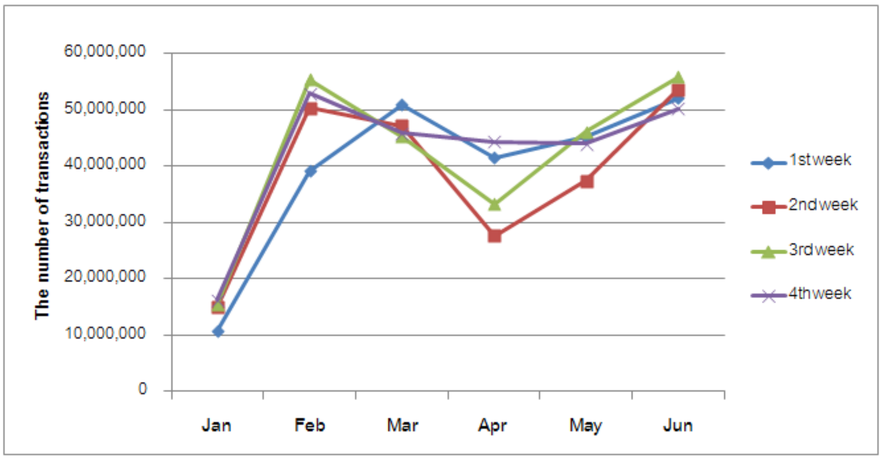


Figure 3.4: The weekly browsing behavior.

Referring to Figure 3.3, the chart shows that the daily browsing behavior has an obvious browsing pattern. This browsing pattern is the time series with sine wave pattern. According to this wave pattern, the volume of web browsing is likely to become lower in the long weekend or vacation time period, such as January or April - May. However, this volume becomes continuously higher after breaking, especially, in the study time period, such as February – March or June. Furthermore, the chart also shows that there is a small number of browsing during the weekend; Saturday - Sunday.

As shown in Figure 3.4, the pattern of the weekly browsing behavior is similar to the pattern of the daily browsing behavior. For this reason, the browsed webs are tightly requested in the study period while they are decreasingly requested in the vacation period.

However, the frequency of each retrieved URL is also determined. Based on the FP and the LSP mentioned previously, after considering the existing values of the FP, there are three different groups can be identified. The first group, low frequency or LF, is the group of URLs that were browsed less than three times per week. This group is approximately 58% of URLs. Moreover, most of these websites are webs where their pages obtained from various server locations, such as facebook.com, or live.com. In addition, webs with frequency only one or two times per week are authorized webs. The second group, medium frequency or MF, is 17% of total URLs retrieving from 4 to 8 times per week. The last group, high frequency or HF, is 25% of total URLs and it is retrieved more than 8 times per week.

As a consequence of the LSP-based analysis, the web size can be grouped to three specific ranges. The fist range, normal size (NS), 69% of all webs, ranges from the file size between 0 and 4,000 bytes. This NS can be categorized into two sub-ranged as small size (SS), and medium size (MS). The SS is the group for the browsed files with sizes less than 1,100 bytes while the MS is the group for the browsed files with sizes between 1,100 bytes and 4,000 bytes. The second range, large size (LS), over 30% of all webs, is ranging from the file size between 4,000 bytes and 10 Mbytes. The third range, extra size (ES), almost 1% of all webs, is for all files that have size larger than 10 Mbytes. For last range, the ES range, this rang is independently grouped as another group due to the sizeable webs though a few webs are browsed. Table 3.10 shows the grouping criteria for web cache based on the FP and LSP.

Table 3.10: The types of grouping criteria.

Types	Grouping Criteria	Loading Ratio (%)
Type 1	LF with SS, MF with LS	25.11
Type 2	LF with MS, MF with NS	23.37
Type 3	HF with All of size ranges excepting ES	25.62
Type 4	All of frequency groups with ES	25.89

Referring to Table 3.10, the probability that a retrieved URL will be a member of a group is approximately equal to 0.25 for every group. These criteria are employed to setup the proxy caches in the ICFA. Nevertheless, groups of webs are altered when the patterns of browsed webs have been changed. Consequently, the system configuration will occur in the classification process. This group classification must be performed in a fixed schedule, such as one week, or one month, depending on the business objective and policy of each organization. Details of the ICFA are elaborated in the following chapter.

CHAPTER IV

PROPOSED METHOD

This chapter describes the proposed cache farming architecture model, the proposed algorithms, and the system workflow. The proposed cache farming architecture is the cache management model which comprises of the web profile database system, the proxy manager system, and the specific proxy server system. The proposed algorithms consist of the update process algorithms, which include the daily update process algorithm, the weekly update process algorithm, and the monthly update process algorithm, and the web pattern classification algorithm. For the last section, this chapter presents the system workflow.

4.1 Proposed architecture: Intelligent Cache Farming Architecture (ICFA)

According to the limitation of proxy caches of ISPs and the existing policy of cache management, the number of caches must be increased when the number of users expanded. Consequently, the complexity of cache management occurs. Thus, in this research, a new architecture, ICFA, has been proposed to eliminate this weakness.

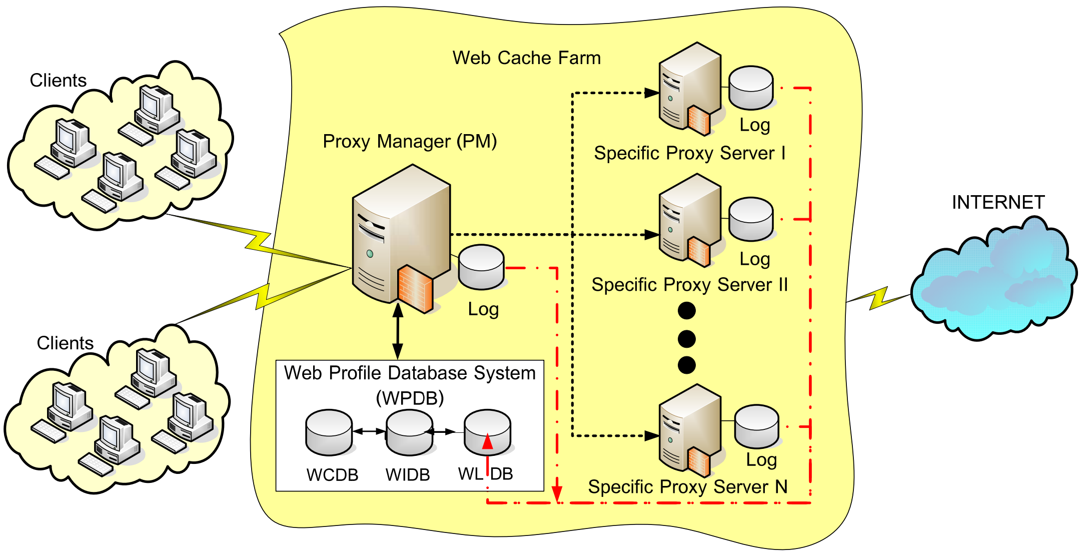


Figure 4.1: The Intelligent Cache Farming Architecture (ICFA).

Referring to Figure 4.1, the ICFA consists of different groups of specific proxy servers (SPS) where the main part is the proxy manager (PM) that cooperates with data in the web profile database system (WPDB). Each part of ICFA will be described as follows.

4.1.1 Web Profile Database System (WPDB)

The web profile database system (WPDB) consists of three sub-databases: a web classification database (WCDB), a web identification database (WIDB), and a log database (WLDB), as illustrated in Figure 4.1.

These sub-databases are implemented for web classifications when users call for webs. Among these three databases, the WCDB and the WIDB are used in the web classification process when retrieving URLs. The WCDB contains both a boundary value of the browse frequency (bv_F) and a boundary value of the web size (bv_S). These data are calculated and used to identify the group whenever a request of the user arrives at the PM. While the WIDB is a database that stores the information of each web, for example, the host name of web, the browse frequency, the size of web, etc. Moreover, the information stored in the WIDB is used with both boundary values stored in the WCDB to identify the pattern of browsed webs by the automatic classification module (ACM) in the PM.

Consider the situation that a request is issued from a client. This request will be recorded into the WLDB; so, the system can be recovered when a failure or an unexpected event arises. Additionally, the records in the WLDB are not obtained from only the retrieved command from the ACM, but also the retrieved commands recorded in the local log database of each SPS, or caches. The transferring of data from all local log databases from each SPS is the offline mode and performed before the RAM starts its evaluation only.

Nevertheless, all data of the WPDB system are necessary for the PM processes that are described as the following section.

4.1.2 Proxy Manager (PM)

The proxy manager (PM) is an assigned proxy server in the cache farm, responsible in classifying webs into groups of webs depending on browsing patterns mentioned previously. However, this system consists of six modules: Record Analyzer Module (RAM), Automatic Classification Module (ACM), Gateway-like module (GM), Internet Communication Module (ICM), Squid Cache Module (SCM), and WPDB Communication Module (WPDB CM). One significant function of this system is the adaptation of the recommending mechanism in the transaction identification when the transaction is indefinable in the normal classification process. Figure 4.2 demonstrates the PM's architecture to perform all tasks. All details of these modules are elaborated below.

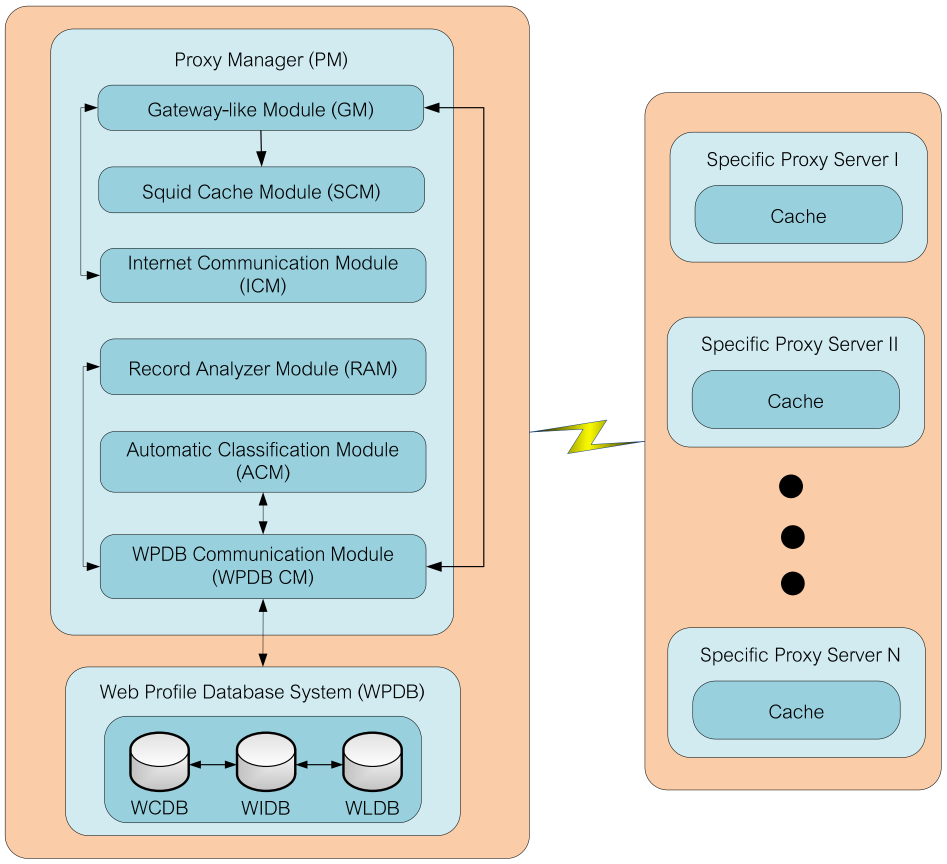

Proxy Manager (PM)
Gateway-like Module (GM)
Squid Cache Module (SCM)
Internet Communication Module (ICM)
Record Analyzer Module (RAM)
Automatic Classification Module (ACM)
WPDB Communication Module (WPDB CM)
Web Profile Database System (WPDB)
WCDB
WIDB
WLDB
Specific Proxy Server I
Cache
Specific Proxy Server II
Cache
Specific Proxy Server N
Cache

Figure 4.2: The Proxy Manager's Architecture.

4.1.2.1 Record Analyzer Module

The RAM is the only module that works individually for pattern classification. This module deals with all URLs stored in the WLDB. The outcomes of RAM computing are new values of browse frequencies and web sizes that must be updated to the WIDB every end of the day. These values are used to justify all unknown webs received from the ACM. Moreover, the re-analyzing of all boundary values, $bv_F(i)$ and $bv_S(i)$, for every group (i) will be performed for every weekend because webs are always inserted, updated, and deleted every second. These new values will overwrite the previous values in the WCDB.

However, these new boundaries may not be suitable since the number of webs in the existing groups under the new defined indexes might break the load balancing policy of the network. Thus, webs with their frequencies and sizes that are closed to the upper or lower bounds of their neighbors are changed to their neighbors until the number of webs in each group is significantly equivalent. Then, the new boundary values, $bv_F(i)$ and $bv_S(i)$, for every group (i) will be recalculated and stored in the WCDB. As a result of RAM process, the size of available caches for each group when passing the process of the ACM may change to obtain high performance and maintain the load balance of each group properly when use.

4.1.2.2 Automatic Classification Module

After all boundaries are calculated by the RAM, the ACM will use them for web grouping. As mentioned in the RAM section, the pattern analysis of web browsing will be performed every weekend. Thus, the process of the ACM will automatically run every weekend as same as the RAM. The group categorization of the ACM is performed based on the most popularity system, called the recommender system. Since this system has two different approaches for recommending users, the content-based (CB) approach and the collaborative-filtering (CF) approach [87-88], so the ACM integrates both of them to gain highest efficiency in the web categorization process.

Based on the calculation values obtained from the RAM process, the CF and CB are applied to identify webs into their suitable groups. Since there are various methods of the CF classification process, the user-based collaborative filtering technique [88], is applied to identify webs in the WIDB. By applying this technique, the webs that have similar pattern share the same proxy cache. For example, if the web X and Y have similar web patterns then the web X can be classified into Group I, and the web Y can be classified into Group I.

Although the real content of each web is unidentified, thus, in this research, the browsing frequency is used as the indicator for users' preference based on the CF rule. Moreover, the file size is used as the web's characteristic based on the CB concept. The algorithm of the ACM is described in Section 4.2.

4.1.2.3 Gateway-like Module

The gateway-like module (GM) is responsible for classifying all arrived transactions from the Internet; so, they will be sent to the suitable proxy servers. The situation of a transaction can be either classifiable or unclassifiable. If the requested transaction is classifiable, then, the request will be sent via the ICM to the classified SPS that links to the PM system, as shown in Figure 1. Otherwise, the request will be sent to the SCM to retrieve the required information. The GM can classify the request using information in the read only mode of the WIDB.

4.1.2.4 Internet Communication Module

The Internet Communication Module (ICM) is responsible for sending a message from the PM to the SPSs according to the classification result of the ACM. The connection between this module and the SPS is the connection-oriented using TCP/IP to guarantee the delivery of data. Moreover, the format of the sending message is the same format of the packet received from the GM. The communication type of the ICM to any SPSs is the simplex method because every SPS will return the requested page to users directly.

4.1.2.5 Squid Cache Module

The Squid Cache module (SCM) is a cache module in the PM. The responsibility of this module is to serve all unidentified webs. Therefore, the load of this module depends on the existing of number of undefined URLs.

4.1.2.6 WPDB Communication Module

Every process of the PM must use data from the WPDB system. This system consists of three sub-databases: WLDB, WIDB, and WCDB. Therefore, the WPDB communication module (WPDB CM) is responsible for creating a connection to the WPDB system to retrieve the required data. The connection performed by this module is the connection-oriented using TCP/IP to prevent occurring of the network congestion and adding of the maintenance costs due to the transmission loss.

4.1.3 Specific Proxy Servers

The specific proxy server (SPS) is a group of caches defined based on contents in the WIDB. Since there are many types of defined boundaries in the WCDB that related to webs in the WIDB, therefore, there are many installed SPSs in the PM. The rule to install a cache or SPS is that a cache can serve more than one group of web patterns, but not vice versa.

However, the size of each SPS is relied on the weekly calculation of the RAM. As a consequence, the size of each SPS is said to be dynamic to maintain the performance of each server which affects to the performance of the entire system. Nevertheless, after the SPS sends all records in its local log database to the WLDB, the SPS will reset its log database and start recording new uses for the next evaluation time.

4.2 Proposed algorithms

In this section, the algorithms for information update process and the algorithm for web classification are presented. The algorithms for information update process are performed in the RAM while the algorithm for web classification is functioned in the ACM.

Table 4.1: Parameters used in algorithms.

Parameters	**Data in databases**	**Definition**	**Databases**
url	URLText	The host name of the URL	WIDB
accumFreq	AccumFreq	The accumlate frequency of each URL during a week	WIDB
accumSize	AccumSize	The accumlate file size of each URL during a week	WIDB
f_m	Freq	The browse frequency of the URL (per a week)	WIDB
s_m	Size	The average file size of the URL (per a week)	WIDB
url_ActCode	URLActCode	The active code: active or inactive	WIDB
cnt_W	Cnt_W	The weekly counter	WIDB
avg_F	GrFreq	The average frequency of the browsed webs in each group (per a week)	WCDB
std_F	StdFreq	The standard deviation of frequency of the browsed webs in each group (per a week)	WCDB
avg_S	GrSize	The average file size of the browsed webs in each group (per a week)	WCDB
std_S	StdSize	The standard deviation of web size of the browsed webs in each group (per a week)	WCDB
updataDate	UpdateDate	The last time of date that the information is updated	WIDB, WCDB
groupCode	GroupCode	The group code	WIDB, WCDB

Based on the WIDB structure in Table 3.2 and the WCDB structure in Table 3.3, some information contained in both two databases are used as parameters in this section. Table 4.1 shows the parameters used in all proposed algorithms. The detailed algorithms are formally given as follows:

4.2.1 Algorithms for Information Update Process

Since webs are always updated their contents to support a variety of requirements of the Internet users, all information relating to characteristics of those webs need to be updated. In this research, the RAM performs this update process of the information stored in both the WIDB and the WCDB. As mentioned previously, there are three sub-process algorithms: the daily update process algorithm (DUPA), the weekly update process algorithm (WUPA), and the monthly update process algorithm (MUPA). The next section elaborates all algorithms of the RAM.

4.2.1.1 Daily Update Process Algorithm (DUPA)

Referring to Figure 4.3, a function of the DUPA is to read records in the WLDB, and then to update the frequency and the size of each retrieved URL, and finally to indicate both of new values into the WIDB for every end of the day. As a fact that there are several new URLs found when the log data is performed, thus, each new URL will be inserted into the WIDB with its size and frequency assigned to be 1 for the first existing time in the database. On the other hand, the existing URLs will be updated their frequency incremented by one for every time when the URL is found. Not only the frequency of existing URL is increased but also the size of this URL will be accumulated counted.

```
Daily Update Process Algorithm (DUPA):

/* Updating accumFreq and accumSize in the WIDB */

1. FOR every url stored in the WLDB DO
2.     IF (the url is found in the WIDB) THEN
3.         accumFreq = accumFreq + 1;
4.         accumSize = accumSize + size of url;
5.     ELSE inserting the url into the WIDB with accumFreq = 1 and accumSize = size of url;
6.     END IF
7. END FOR
```

Figure 4.3: Daily Update Process Algorithm (DUPA).

4.2.1.2 Weekly Update Process Algorithm (WUPA)

Referring to Figure 4.4, the WUPA is responsible for updating some information stored in the WIDB and the WCDB, respectively, for every end of the week. In the beginning of the process, the WUPA calculates the browse frequency (f_m) and the web size (s_m) of every URL to indicate in the WIDB. These two values are used in the ACM process after finishing the RAM process. Then, the adjustments of the *cnt_W* and the *url_ActCode* are performed; the *cnt_W* is incremented by one, and the *url_ActCode* is revised if the retrieved URL is the active URL. After this state, the accumFreq and the accumSize can be cleared and the updateDate is set to the present date. The next process of the WUPA is re-calculating of all boundary values, $bv_F(i)$ and $bv_S(i)$, for every group (i) that are contained in the WCDB.

```
Weekly Update Process Algorithm (WUPA):

 /* Updating  $f_m$, $s_m$, url_ActCode, cnt_W, and updateDate in the WIDB, and
    updating $bv_F(i)$ and $bv_S(i)$ and updateDate in the WCDB */

1. FOR every url in the WIDB DO
2.     $f_m$ = accumFreq; $s_m$ = accumSize/accumFreq;
3.     cnt_W = cnt_W + 1;
4.     IF ($f_m$ != 0) THEN
5.         updating url_ActCode with "1" at the nth position, where n is the cnt_W value;
6.     END IF
7.     accumFreq = 0; accumSize = 0;
8.     updateDate = NOW;
9. END FOR
10. FOR each group (i) in the WCDB DO
11.     calculating $bv_F(i)$ and $bv_S(i)$;
12.     updateDate = NOW;
13. END FOR
```

Figure 4.4: Weekly Update Process Algorithm (WUPA).

4.2.1.3 Monthly Update Process Algorithm (MUPA)

There are several reasons that affect to the use of webs. These reasons are such as changing of user's requirements, developing of most webs, etc. Consequently, some webs stored once in the database are never used. However, data overloading in the database system are concerned in order to manage and maintain all data. Therefore, the MPUA is introduced to control this situation; this process is performed every end of the month. As illustrated in Figure 4.5, after the first week if the classified web stored in the WIDB is inactive for the last three consecutive weeks; *url_ActCode* = *"1000"* and *cnt_W* = 4, it will be removed from the WIDB.

```
Monthly Update Process Algorithm (MUPA):

/* Removing the inactive urls from the WIDB */

1. FOR every url where url_ActCode = "1000" and cnt_W = 4 DO
2.      removing url ;
3.      cnt_W = 0;
4. END FOR
```

Figure 4.5: Monthly Update Process Algorithm (MUPA).

4.2.2 Algorithm for Web Classification: Web Pattern Classification Algorithm (WPCA)

After the WUPA of the RAM is completed, the outcomes of the WUPA, f_m, s_m, $bv_F(i)$, and $bv_S(i)$, are used for web grouping in the WPCA. The WPCA is called by the ACM and performed every weekend. The WPCA process is classifying the webs to the right group or SPS. Figure 4.6 shows the classification of the requested web that is sent to the corresponding group or the SPS using two boundary values, the bv_F and the bv_S, as indicators.

Let m be the set of different requested webs, named $\{w_1, w_2,\ldots,w_m\}$; each w_m has the browse frequency denoted by f_m , and the web size denoted by s_m.

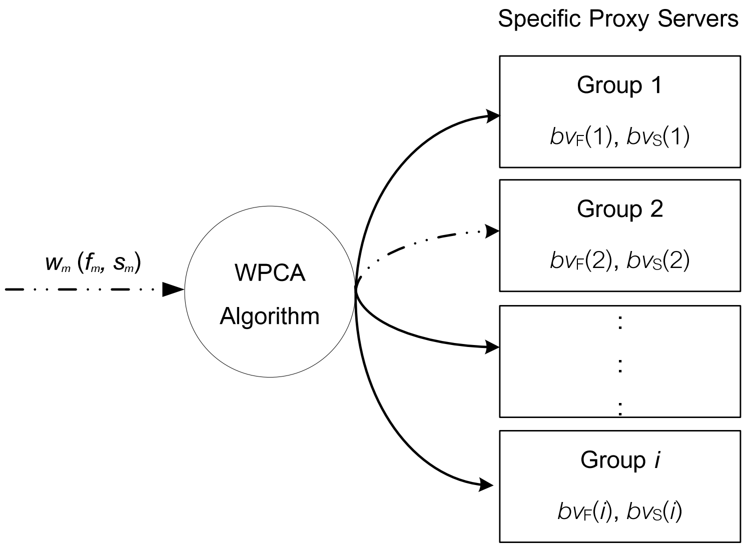


Figure 4.6: The Web Classification Mechanism of the WPCA using the boundary values.

Referring to Figure 4.3, each group has the bv_F and the bv_S which the bv_F of group (i) denoted by $bv_F(i)$, and the bv_S of group (i) denoted by $bv_S(i)$, where i is the i^{th} of classified group. Then, the $bv_F(i)$ and the $bv_S(i)$ are defined as Equation (1) and Equation (2), respectively.

Given $F_l(i), F_u(i), S_l(i), S_u(i) \in \Re$, and $F_l(i) < F_u(i)$ and $S_l(i) < S_u(i)$. Let $F_l(i) = avg_F(i) - std_F(i)$ and $F_u(i) = avg_F(i) + std_F(i)$, where $avg_F(i) \in \Re$ is the average of the browse frequency of group (i) and $std_F(i) \in \Re$ is the standard deviation of the browse frequency of group (i). In addition, $S_l(i) = avg_S(i) - std_S(i)$ and $S_u(i) = avg_S(i) + std_S(i)$, where $avg_S(i) \in \Re$ is the average of the web size of group (i) and $std_S(i) \in \Re$ is the standard deviation of the web size of group (i). Then,

$$bv_F(i) = [F_l(i), F_u(i)] = \{x \mid F_l(i) \leq x \leq F_u(i)\} \qquad (1)$$

$$bv_S(i) = [S_l(i), S_u(i)] = \{y \mid S_l(i) \leq y \leq S_u(i)\} \qquad (2)$$

Since $f_m, s_m \in \Re$, and $f_m \in bv_F(i) \leftrightarrow F_l(i) \leq f_m \leq F_u(i)$ and $s_m \in bv_S(i) \leftrightarrow S_l(i) \leq s_m \leq S_u(i)$. Thus, if $f_m \in bv_F(i)$ and $s_m \in bv_S(i)$, then the w_m is classified to that group (i). However, the data management and maintenance are recognized to handle data overloading in the database. For this reason, if the classified w_m is inactive for a certain period of time, within one month, it will be removed from the database. The detailed WPCA is formally given as shown in Figure 4.4.

Referring to Figure 4.7, the ACM with the WPCA will handle the unidentified URL. Since the values in WIDB contain the real f_m and s_m during a week, every f_m and s_m of each web will be compared with $bv_F(i)$ and $bv_S(i)$ in the WCDB for grouping. Therefore, there are two possible situations of this comparison. The first situation is the normal situation where webs satisfy the condition of group (i) in the WCDB, $f_m \in bv_F(i)$ and $s_m \in bv_S(i)$, then, those webs are classified as webs of group (i). The second situation occurs when there is no suitable condition in the WCDB for webs in the WIDB, either $f_m \notin bv_F(i)$ or $s_m \notin bv_S(i)$, or neither of them. So, these webs will be identified to the group with highest avg_F under the assumption that the more popular request, the more chance to meet requirement. However, since the execution of the WPCA may require scanning of all URLs in the database at every classification decision, the time complexity of the algorithm would incur $O(n)$ where n is the number of URLs in the database.

```
Web Pattern Classification Algorithm (WPCA):

/* the $f_m$ and the $s_m$ of url are stored in the WIDB
   the $bv_F(i)$ and the $bv_S(i)$ are contained in the WCDB */

INPUT: urls stored in the WIDB;

1. BEGIN
2. FOR every url DO
3.     IF ($f_m \in bv_F(i)$ and $s_m \in bv_S(i)$) THEN
4.         classify url to a corresponding SPS of group ($i$);
5.     ELSE automatically classify url to a SPS of group ($i$) that has the highest $avg_F(i)$;
6.     END IF
7. END FOR
8. END

OUTPUT: The group ($i$);
```

Figure 4.7: The Web Pattern Classification Algorithm (WPCA).

4.3 System workflow

Referring to Figure 4.8, the system workflow is presented. The proxy manager (PM) receives a user request, then classifies the web into a group of web patterns, and finally pushes the user request to a corresponding specific proxy server (SPS) or the Squid cache module (SCM) inside the PM. The classified web is sent to the SPS while the unclassified web is pushed to the SCM to retrieve the required information. After that the assigned SPS or the SCM handles the user request. It searches the web object on its cache and returns the object to the user. If it is not available, the SPS or the SCM will fetch the web object from the object's Web server, then not only return that object to the user, but also keep the object in its cache.

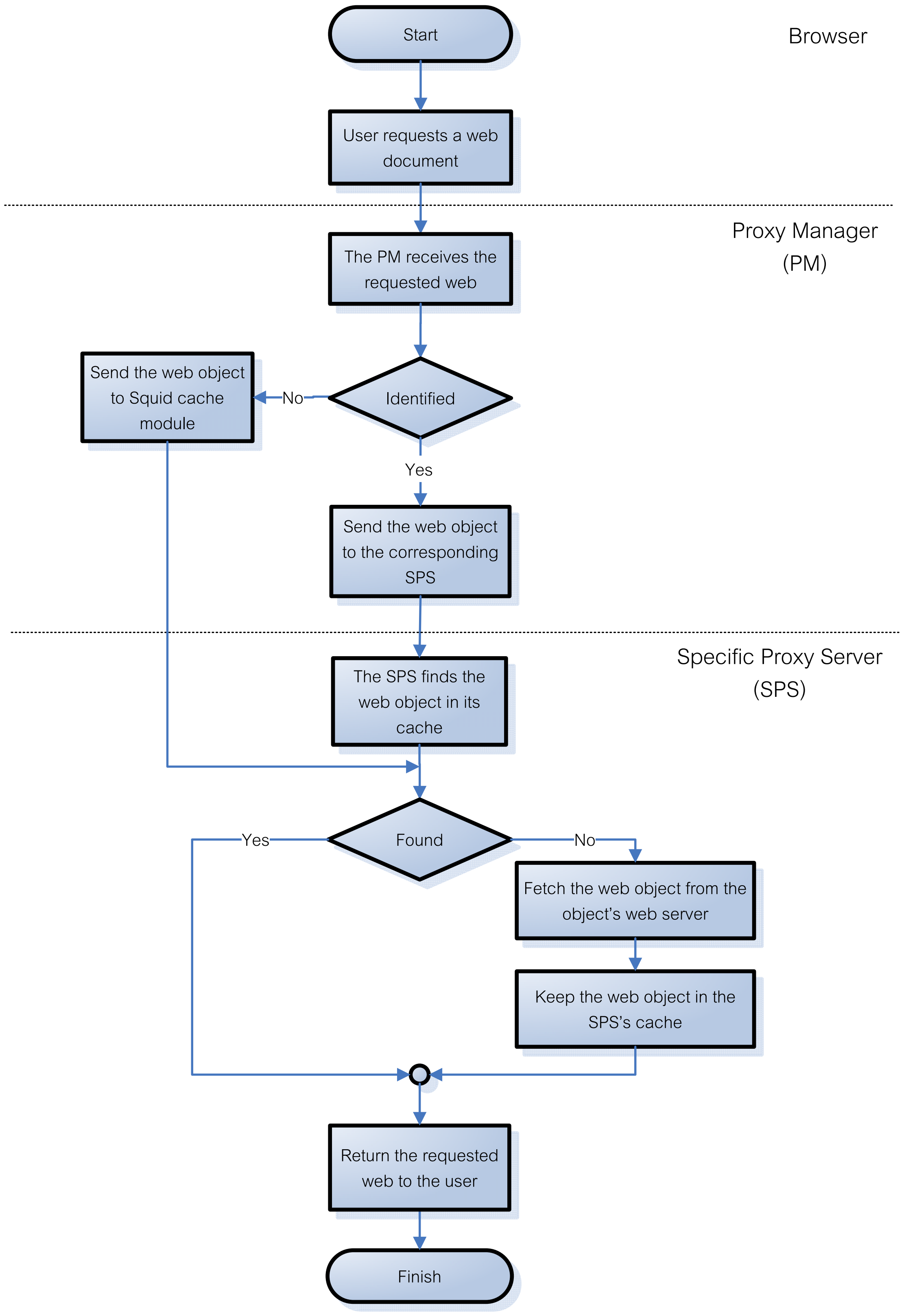

Browser
Start
User requests a web document
Proxy Manager (PM)
The PM receives the requested web
Identified
No
Send the web object to Squid cache module
Yes
Send the web object to the corresponding SPS
Specific Proxy Server (SPS)
The SPS finds the web object in its cache
Found
Yes
No
Fetch the web object from the object's web server
Keep the web object in the SPS's cache
Return the requested web to the user
Finish

Figure 4.8: System workflow.

CHAPTER V

IMPLEMENTATION AND EXPERIMENTAL RESULTS

In order to evaluate and compare the caching performance of the proposed ICFA to the traditional caching architecture (TCA), which is the original model without grouping criteria for the web cache, the trace-driven simulation is performed by implementing both models in a virtual machine environment. The simulation results to demonstrate the performance of the proposed architecture are described as follow.

5.1 Implementation

The implementation for the simulation environment system is performed on HP 2 Quad Cores with XEON Processors and 16 GB main memory running Ubuntu 10.04 Desktop. Moreover, several softwares are installed on this environment system. The database management system employs MySQL Server version 5.1, and the phpMyAdmin running on Apache2 Web server is used to deal with MySQL Server. However, all mechanisms maintaining the database system, algorithms, and all functions of system are developed using Java (openjdk-6-jre-headless openjdk-6-jdk). Since Squid is a high-performance proxy caching server for web clients and widely used in academic organization [24]; this simulation system is implemented by Squid 2.7 STABLE7.

Based on the default replacement strategies in Squid 2.7 software, four cache replacement policies [89] are applied to assess the efficiency of both the proposed ICFA and the original TCA models. The first policy is lru, Squid's original list based Least Recently Used (LRU). The LRU policy keeps recently referenced documents but evicts the least recently accessed documents. The second policy is heap GDSF (Greedy-Dual Size Frequency). The GDSF policy is a variant of the Greedy Dual-Size policy taking into account frequency of reference. This policy optimizes document hit rate by remaining smaller popular documents in cache, so it has a better chance to get the hit.

The last two policies which are heap LFUDA (Least Frequently Used with Dynamic Aging), and heap LRU. These are the third and the fourth policy, respectively. The LFUDA policy is a variant of the LFU using a dynamic aging policy to accommodate shifts in the set of popular documents. The LFUDA maintains popular documents in the cache regardless of their sizes; and, thus, it optimizes byte hit rate at the expense of hit rate since one large, popular document will prevent many smaller, slightly less popular documents from being cached. The last policy, the heap LRU, is LRU policy implemented using a heap.

Table 5.1: Performance metrics used in cache replacement policies.

Performance Metric	Definition
Hit Ratio (HR)	$HR = \frac{\sum_{m \in N} h_m}{\sum_{m \in N} r_m}$
Byte Hit Ratio (BHR)	$BHR = \frac{\sum_{m \in N} s_m \cdot h_m}{\sum_{m \in N} s_m \cdot r_m}$
Average Byte (Avg.Byte)	$Avg.Byte = \frac{\sum_{m \in N} s_m}{\lVert N \rVert}$
Average Response Time of Hit (Avg.RTH)	$Avg.RTH = \frac{\sum_{m \in N} rt_m \cdot h_m}{\sum_{m \in N} rt_m \cdot r_m}$
Average Response Time (Avg.RT)	$Avg.RT = \frac{\sum_{m \in N} rt_m}{\lVert N \rVert}$
Notations: r_m = the total number of requests for document *m* h_m = the total number of hits for document *m* s_m = the size of document *m* rt_m = the response time for document *m* N = the set of all browsed documents $\lVert N \rVert$ = the size of N	

However, performance metrics are used to evaluate a cache replacement policy. Classical performance metrics [59], [90] which are Hit Rate (HR), Byte Hit Rate (BHR), Average Byte (Avg.Byte), Average Response Time of Hit (Avg.RTH), and Average Response Time (Avg.RT), are measured in the experiments. Their definitions are shown in Table 5.1. The HR is defined as the percentage of requests that can be satisfied by the cache. The BHR is the number of bytes satisfied from the cache as a fraction of the total bytes requested by clients. The Avg.Byte is the average number of bytes requested by clients. The Avg.RT is the average number of the response time or an amount of time that a server starts to process a request and stops after it has received the return. Finally, the Avg.RTH is the average number of response time that causes a hit in the cache.

5.2 Simulation model

Based on the defined groups in Section 3.2.2, the WCDB contains four pairs of boundaries according to four groups of webs in the WIDB. Thus, there are four SPSs in the simulation system and one SCM inside the PM for undefined webs. Figure 5.1 shows the simulation system environment for the ICFA.

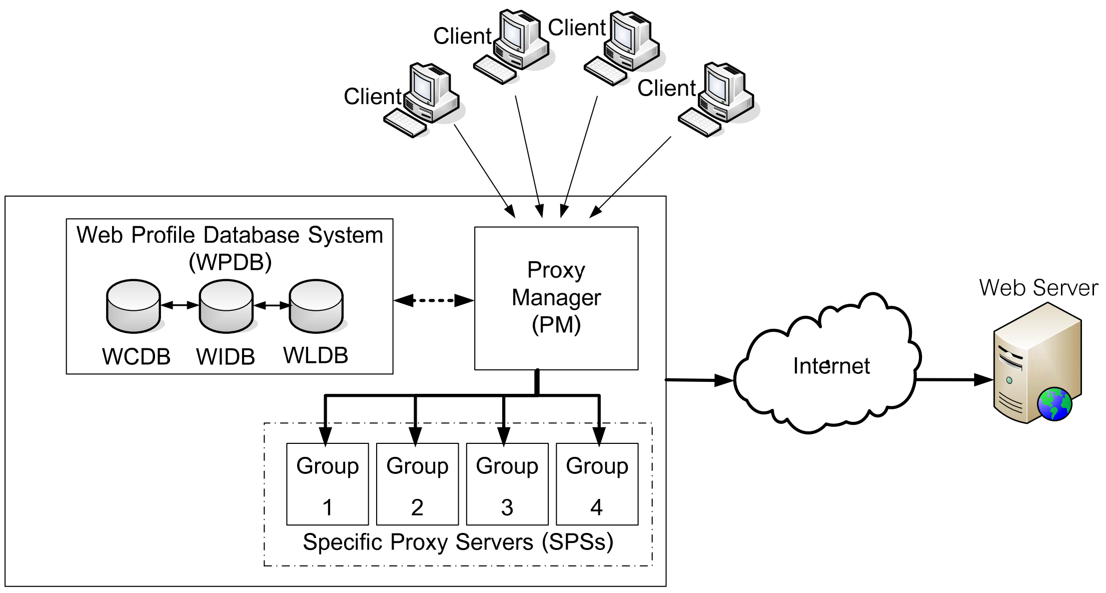


Figure 5.1: The simulation system environment for the proposed ICFA.

As mentioned in Section 3.1.2, the access.log data is not only retrieved from Chula, but also obtained from NPRU. The Chula's log data is used as the samples of retrieval behavior to identify types of browsing characteristics while the NPRU's log data is used as the experimental data after the types of grouping criteria are defined. In order to prove the proposed architecture, the simulation with parameters shown in Table 5.2 is performed on the environments of two models: the ICFA and the TCA.

Table 5.2: Simulation parameters.

Parameters	Values
Number of days for experiment data	60 days (October-November 2010)
Total transactions for experiment data	approximately 57 million records
Cache size for each server	256 MB
Cache replacement policies	lru, heap LRU, heap LFUDA, GDSF

5.3 Simulation results

This section presents the comparisons of caching efficiency between the proposed ICFA and the TCA model. The simulation results are stated in three experiments as follows:

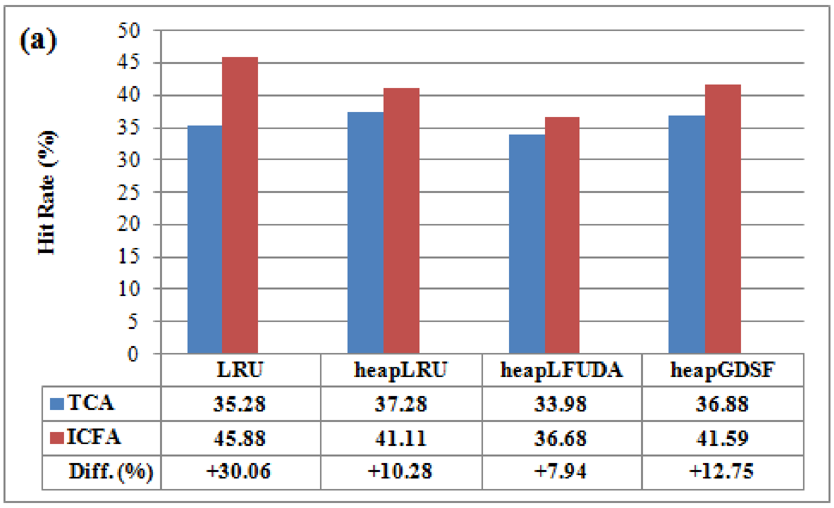
Experiment 1: The objective of the first experiment is to show that the implementation of the ICFA applying the WPCA algorithm improves the caching performance comparing to the implementation of the TCA in different replacement policies.

Experiment 2: The second experiment is to investigate which cache replacement policy is suitable for the proposed ICFA and the TCA measured by the HR, the BHR, and the Avg.RT.

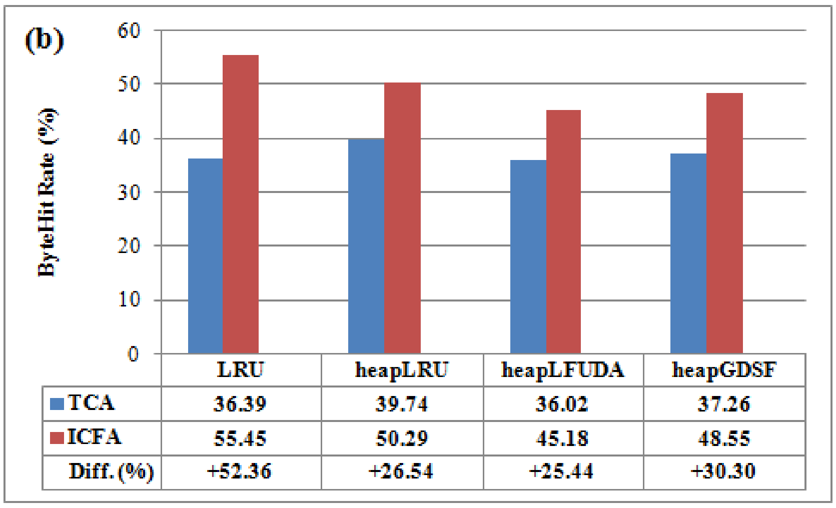
Experiment 3: The third experiment shows the effectiveness of the ICFA which can enhance the performance of caching when the number of users (requests) is higher. Additionally, daily caching performance of the ICFA comparing to the TCA is demonstrated.

5.3.1 Experiment 1: *The cache effect of different replacement policies for the comparison between the ICFA and the TCA*

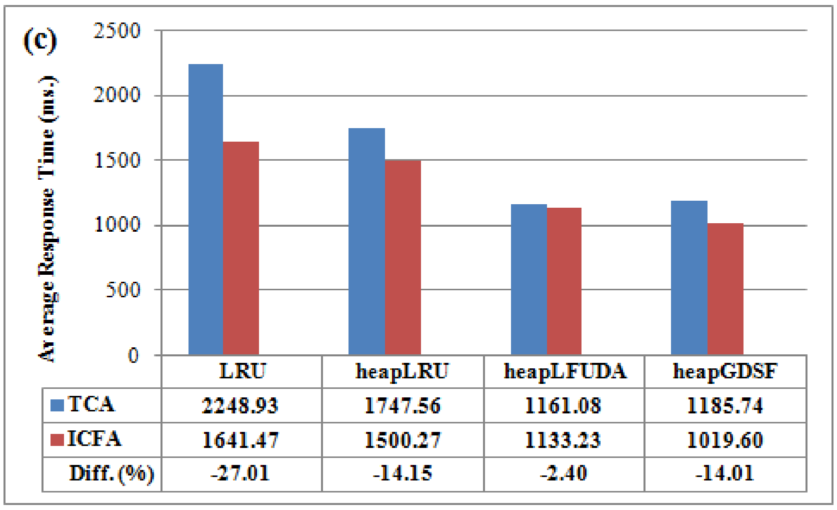
As shown in Figure 5.2, the ICFA yields significantly better performance, compared to the TCA in every cache replacement polices. In addition, the results also present that the LRU used by the ICFA is better than other three algorithms, the heap LRU, the heap LFUDA, and the heap GDSF. For this reason, the HR and the Avg.RT of the ICFA are enhanced approximately 30% while the BHR increases more than 52%.



	LRU	heapLRU	heapLFUDA	heapGDSF
🟦 TCA	35.28	37.28	33.98	36.88
🟥 ICFA	45.88	41.11	36.68	41.59
Diff. (%)	+30.06	+10.28	+7.94	+12.75



	LRU	heapLRU	heapLFUDA	heapGDSF
🟦 TCA	36.39	39.74	36.02	37.26
🟥 ICFA	55.45	50.29	45.18	48.55
Diff. (%)	+52.36	+26.54	+25.44	+30.30

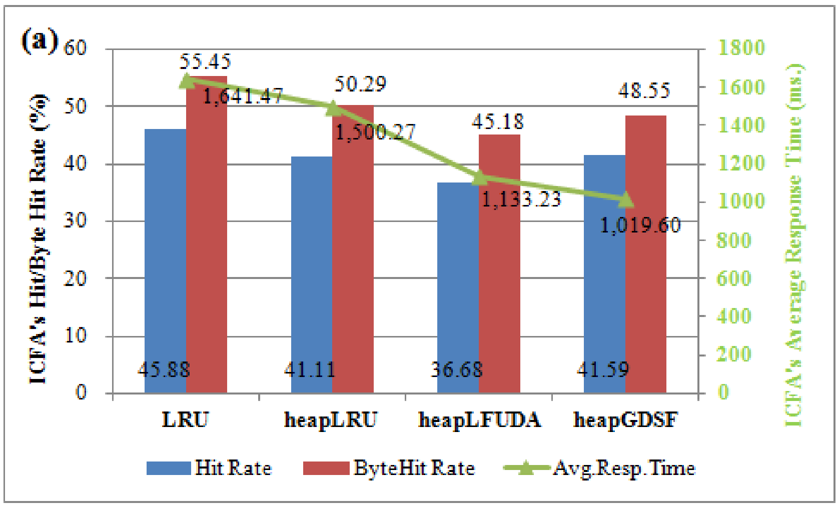


	LRU	heapLRU	heapLFUDA	heapGDSF
🟦 TCA	2248.93	1747.56	1161.08	1185.74
🟥 ICFA	1641.47	1500.27	1133.23	1019.60
Diff. (%)	-27.01	-14.15	-2.40	-14.01

Figure 5.2: Comparisons of caching performance between the ICFA and the TCA in different metrics; (a) Hit Rate, (b) Byte Hit Rate, and (c) Average Response Time.

5.3.2 Experiment 2: *The cache effect of different replacement policies measured by various performance metrics for the ICFA and the TCA*

Generally, the cache replacement algorithm is automatically implemented in the cache box of proxy server. As mentioned previously, there are four replacement algorithms listed in Squid software, thus, one of these algorithms will be selected to set up the policy for cache management. For this reason, the cache performances investigated by the HR, the BHR, and the Avg.RT for different cache replacement policies are observed in the ICFA and the TCA.



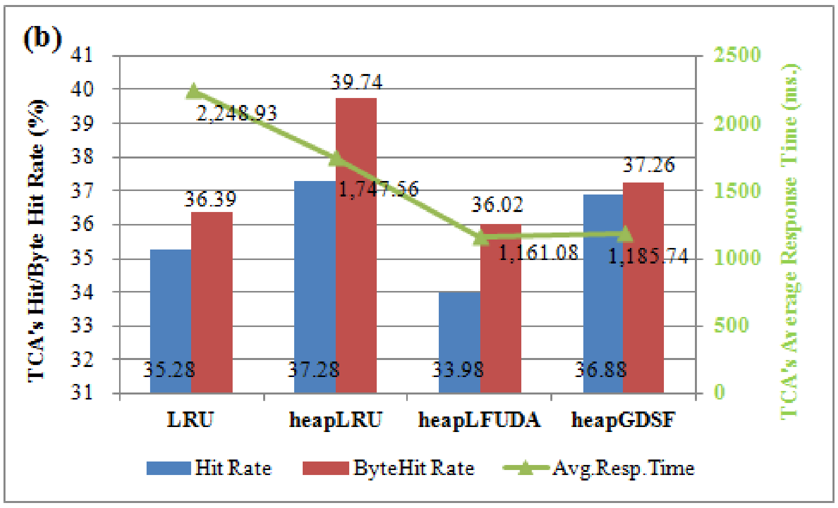


Figure 5.3: Comparisons among cache replacement policies measured by Hit Rate, Byte Hit Rate and Average Response Time in models; (a) the ICFA, and (b) the TCA.

Referring to Figure 5.3 (a), the experimental results show that the LRU outperforms other three policies assessed by the HR (45.88%) and the BHR (55.45%) while the heap GDSF achieves the Avg.RT of 1,019.60 milliseconds. On the other hand, the results of the TCA illustrated in Figure 5.3 (b) show that the heap LRU performs better than the others; this result is confirmed by the highest HR (37.28%) and the highest BHR (39.74%). Additionally, the Avg.RT of 1,161.08 milliseconds is accomplished by the heap LFUDA.

Since some metrics are not achieved at the same time, a policy performs very well in terms of one performance metric but poorly in terms of the others [62]. As the results, the LRU can be selected as the suitable policy for the ICFA while the heap LRU can be selected to perform in the TCA due to the achievement of two-three of all measured metrics.

After considering the results of Experiment 2, however, one interesting of these results is that both of the ICFA and the TCA models obtain the similar results of the HR and the BHR for the heap LFUDA. That is the heap LFUDA shows the worst results for both metrics.

5.3.3 Experiment 3: *The cache effect of increasing requests*

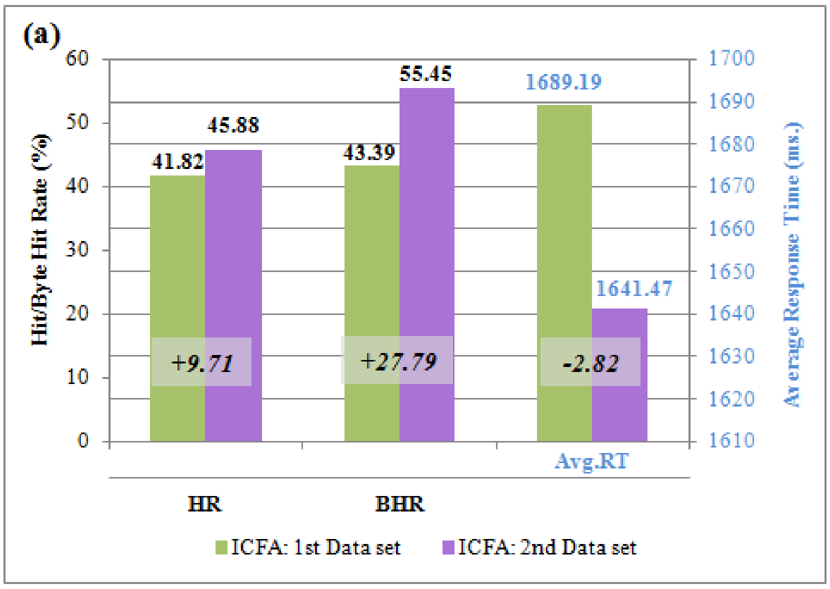
In this experiment, the performance metrics for increasing requests or browses are observed. All performance results including the caching performance overviews, the caching performance of each cache in the ICFA, and finally daily caching performance for the comparison between the ICFA and the TCA are shown in the following sections.

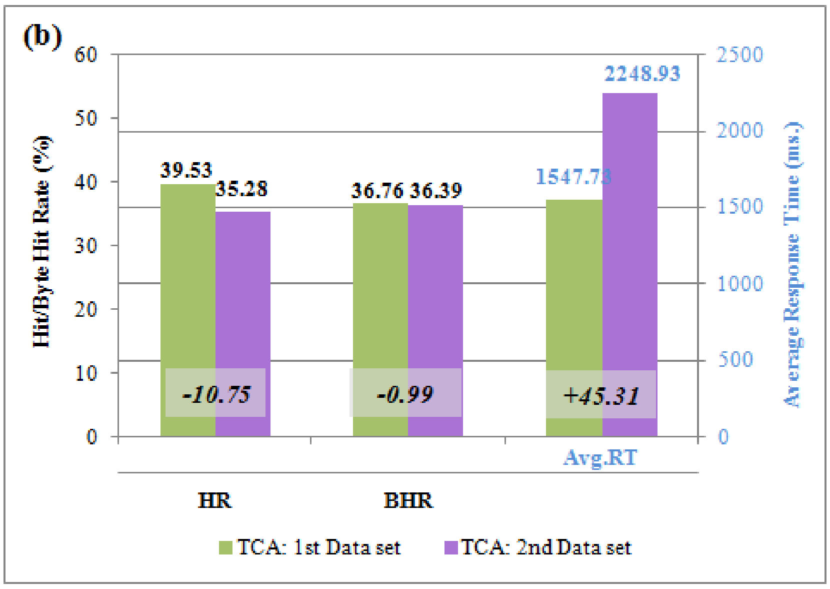
5.3.3.1 Caching performance overviews

Since the Internet users' satisfaction is an important issue for the Internet services, every organization providing the services needs to recognize this importance. The QoS must be maintained, even though the services requested from the users increase.

In order to quantify the caching performance of the ICFA and the TCA, the experimental data is divided into two different data sets: the first data set (smaller data set) and the second data set (larger one). The first data set is the NPRU's log data collected in October 2010; they are approximately 18 million records. The second data set is the NPRU's log data collected in November 2010; they are approximately 39 million records. Then, both of data sets are performed sequentially in this experiment.

However, the cache replacement policy needs to be the same for both the ICFA and the TCA for testing. As a testing consequence in Experiment 2, each cache of the ICFA and the TCA for this experiment employs the Least Recently Used (LRU) algorithm as their cache replacement policy; so, the similarity in individual cache management can be directly compared without bias or any external influencers.


(a)
Hit/Byte Hit Rate (%)
Average Response Time (ms.)
60
50
40
30
20
10
0
1700
1690
1680
1670
1660
1650
1640
1630
1620
1610
41.82
45.88
43.39
55.45
1689.19
1641.47
+9.71
+27.79
-2.82
HR
BHR
Avg.RT
ICFA: 1st Data set
ICFA: 2nd Data set


(b)
Hit/Byte Hit Rate (%)
Average Response Time (ms.)
60
50
40
30
20
10
0
2500
2000
1500
1000
500
0
39.53
35.28
36.76
36.39
1547.73
2248.93
-10.75
-0.99
+45.31
HR
BHR
Avg.RT
TCA: 1st Data set
TCA: 2nd Data set

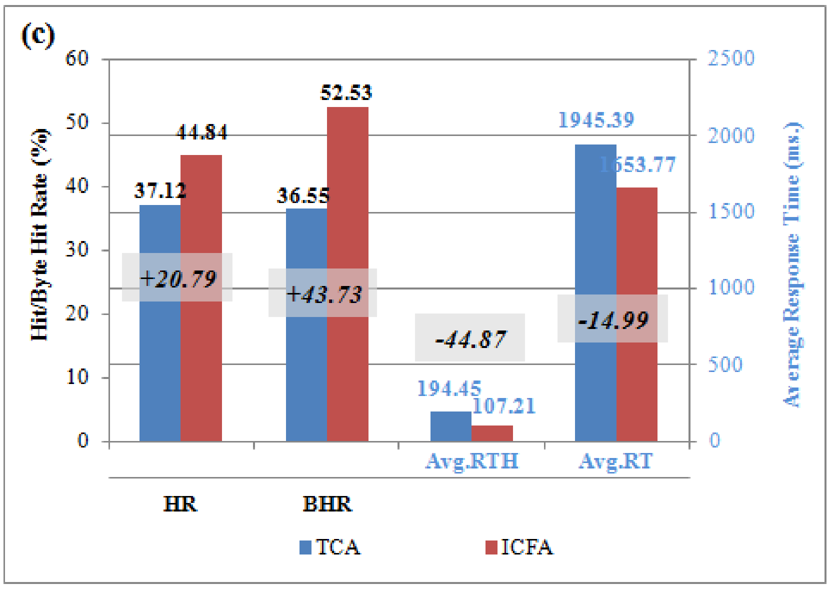


Figure 5.4: Caching performance overviews.

Referring to Figure 5.4 (a), this figure shows the ICFA's caching performance in different amount of requests. The results show that all performance metrics, the HR, the BHR, and the Avg.RT, are obviously improved when the amount of the requests increases. The HR increases more than 9% while the BHR significantly increases, that is approximately 28%. The last metric, the Avg.RT, decreases almost 3%. In contrast, the TCA's caching performance is poor in all terms of metrics when the amount of requests increases as shown in Figure 5.4 (b). The HR and the BHR reduce more than 10% and approximately 1%, respectively, and especially the Avg.RT increase almost 46%.

As illustrated in Figure 5.4 (c), considering all data sets to the comparison between the ICFA and the TCA, the results show that the ICFA performs significantly better than the TCA in every metrics. The HR and the BHR increase more than 20% and 43%, respectively. As for the other two metrics, the Avg.RT reduces approximately 15% and the Avg.RTH extremely reduces to 44%. Moreover, in order to verify the presented simulation results, the statistical data analysis of the HR, the BHR, the Avg.RTH, and the Avg.RT is performed using SPSS statistic software.

Since the *t*-test algorithm is the procedure comparing the means of two groups or (one-sample) compares the means of a group with a constant, the *t*-test algorithm is used to determine the difference of obtained performance results between the ICFA and the TCA using Paired-Samples *t*-test technique [91]. The tested data is the results of performance metrics in each day of both models; they are 60 records of data. At 95% of confidence level, the alternative hypothesis of this testing is that the means of the performance metric for two models are significantly different while the null hypothesis is that the means of the performance metric for two models are significantly equal. Table 5.3 shows the results of statistical data analysis using Paired-Samples *t*-test technique.

Table 5.3: Results of the Paired Samples Test.

Paired Samples Test

		Paired Differences							
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper	t	df	Sig. (2-tailed)
Pair 1	ICFA - TCA	6.42969	6.25790	.80789	4.81310	8.04627	7.959	59	.000

(a) Paired Samples Test for HR.

Paired Samples Test

		Paired Differences							
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper	t	df	Sig. (2-tailed)
Pair 1	ICFA - TCA	13.21046	7.90937	1.02110	11.16725	15.25367	12.938	59	.000

(b) Paired Samples Test for BHR.

Paired Samples Test

		Paired Differences							
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper	t	df	Sig. (2-tailed)
Pair 1	ICFA - TCA	-365.87584	844.54212	109.02992	-584.04420	-147.70747	-3.356	59	.001

(c) Paired Samples Test for RT.

Paired Samples Test

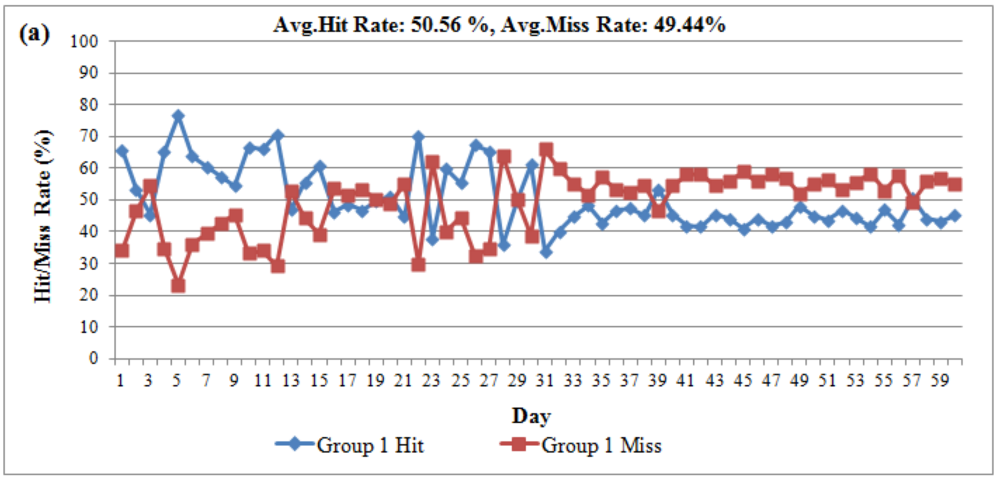
		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	ICFA - TCA	-93.82736	94.84970	12.24504	-118.32963	-69.32508	-7.662	59	.000

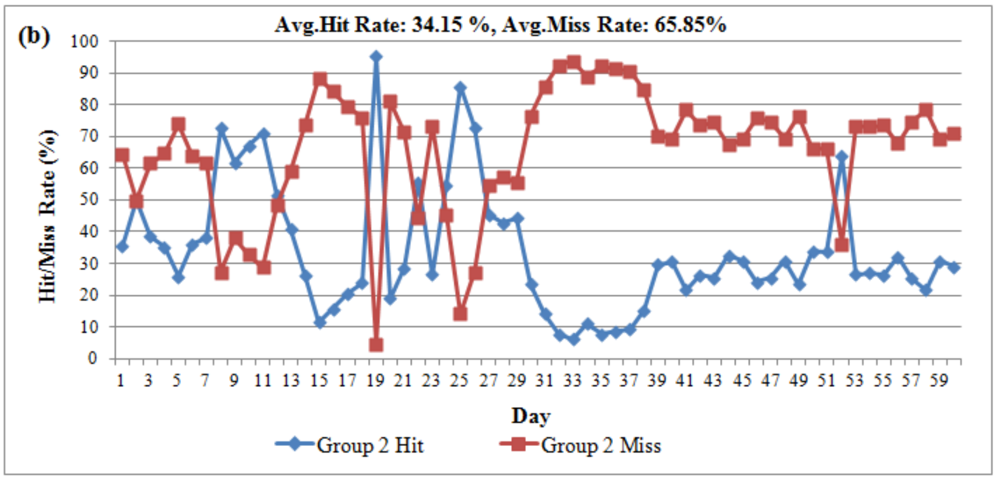
(d) Paired Samples Test for RTH.

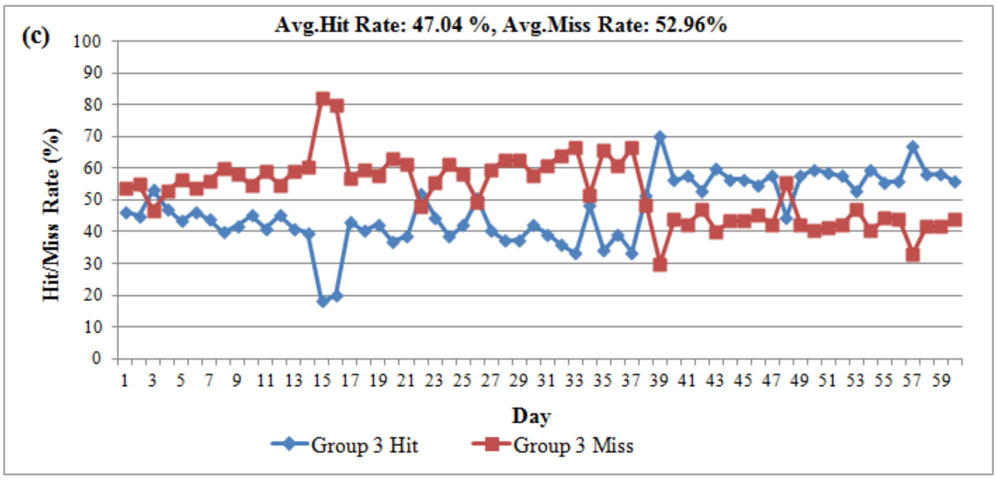
Considering the Probability Value (p-value) shown in Table 5.3, the p-values in every performance metric is less than 0.05 that are 0.000 for the HR, the BHR, and the RTH, and 0.001 for the RT. By convention, if the p-value is less than 0.05 (95% of confidence level), the null hypothesis is rejected. Thus, there is a statistically significant difference in the performance metrics between the ICFA and the TCA at a 95% of confidence level.

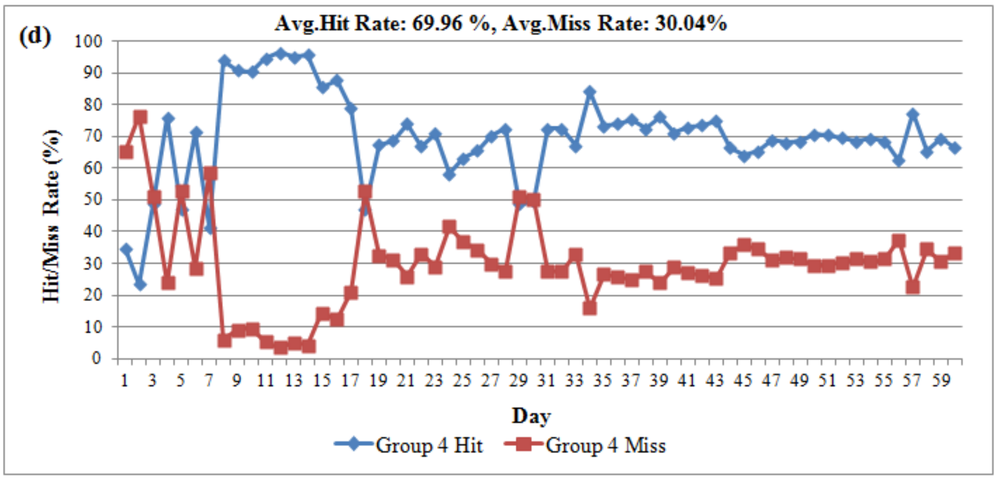
5.3.3.2 Caching performance of the caches in the ICFA

This section demonstrates the measured performance of each cache in the ICFA. Based on the different types defined in Table 3.10, Type 1 is assigned to Group 1 in Figure 5.5 (a), Type 2 is assigned to Group 2 in Figure 5.5 (b), and so on. In addition, the SCM in the PM is assigned to Group Main in Figure 5.5 (e). Figure 5.5 displays the daily hit/miss rates of caches in the ICFA.




(b)
Avg.Hit Rate: 34.15 %, Avg.Miss Rate: 65.85%
Hit/Miss Rate (%)
Day
Group 2 Hit
Group 2 Miss


(c)
Avg.Hit Rate: 47.04 %, Avg.Miss Rate: 52.96%
Hit/Miss Rate (%)
Day
Group 3 Hit
Group 3 Miss


(d)
Avg.Hit Rate: 69.96 %, Avg.Miss Rate: 30.04%
Hit/Miss Rate (%)
Day
Group 4 Hit
Group 4 Miss

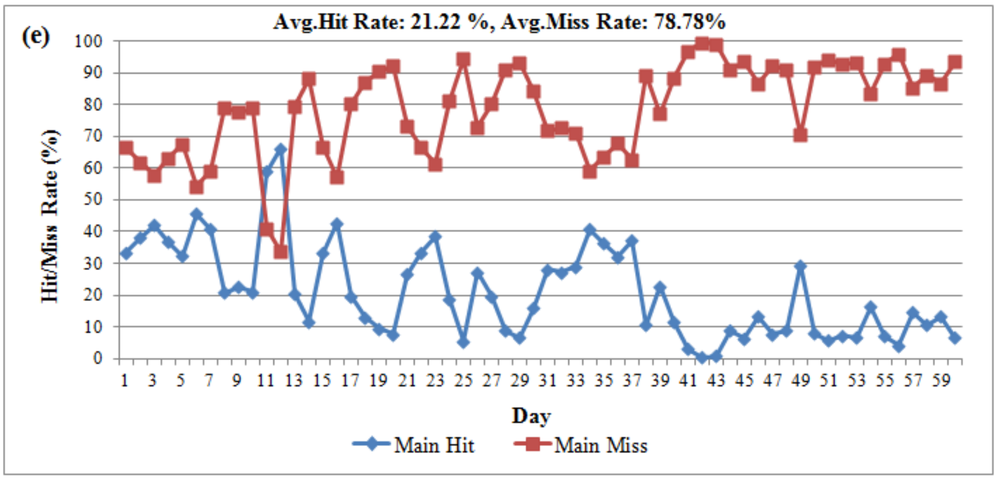


Figure 5.5: Daily Hit/Miss Rate of each cache in the ICFA; (a) Group 1, (b) Group 2, (c) Group 3, (d) Group 4, and (e) Group Main.

As shown in Figure 5.5, it comes out that the average hit rate of Group 1 to Group 4 is approximately 50% while the average hit rate of Group Main is only 21.22% which is a half of the average hit ratio of the four groups. From the first 30 days (the first data set) depicted in Figure 5.5 (a) - (d), the contents in Group 1 and Group 4 are hardly changed and easily to be found since Group 1 and Group 4 contain mostly static webs whereas Group 2 and Group 3 are mostly dynamic webs. Consequently, the hit rates of Group 1 and Group 4 are higher than the hit rates of Group 2 and Group 3.

Focussing on the last 30 days (the second data set) shown in Figure 5.5 (a) – (d), all active URL will be updated their groups according to their patterns. Thus, the new result of web classification shows that some URLs belonging to Group 1 and Group 2 might be changed their groups to Group 3 as well as some URLs of Group 3 that are mostly dynamic webs might be classified to Group 1 and Group 2. Thus, the hit rate of Group 3 increases whereas the hit rates of Group 1 and Group 2 decrease.

As for Group Main, it contains various data due to serving the unclassified webs. However, after a re-classifying process, these unclassified webs are identified to their group. Thus, the hit rate of this group will decrease as shown in Figure 5.5 (e).

5.3.3.3 Daily caching performance

Figure 5.6 - Figure 5.9 provide a daily plot of the performance of the ICFA compared to the TCA.

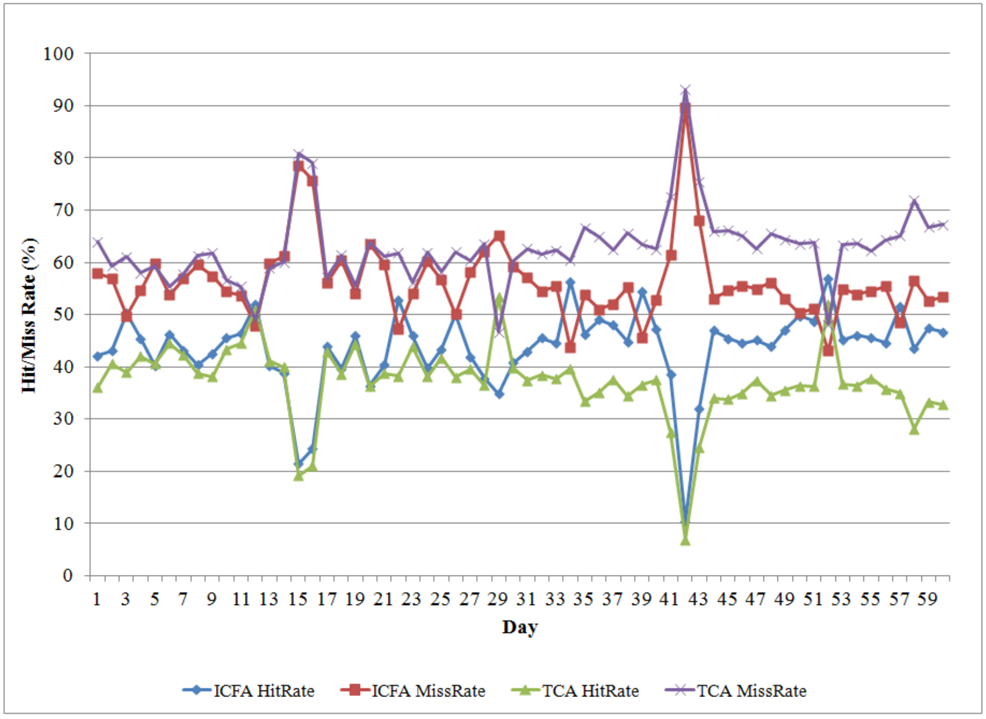


Figure 5.6: Daily comparison of caching performance between the ICFA and the TCA measured by Hit Rate and Miss Rate.

Referring to Figure 5.6, at the beginning of the measurement process, the hit rate of the ICFA (blue line) is a bit higher than the TCA (green line). Nevertheless, the variety of webs in each cache is small when starting the ICFA but all requested webs, based on the defined patterns, will be daily accumulated to the system until most of the requests are available in the prospered cache. Thus, the number of hits based on the ICFA has potential to be much better. However, if retrieved URLs have no pattern as expected, the daily hit rates of both the ICFA and the TCA will be dropped.

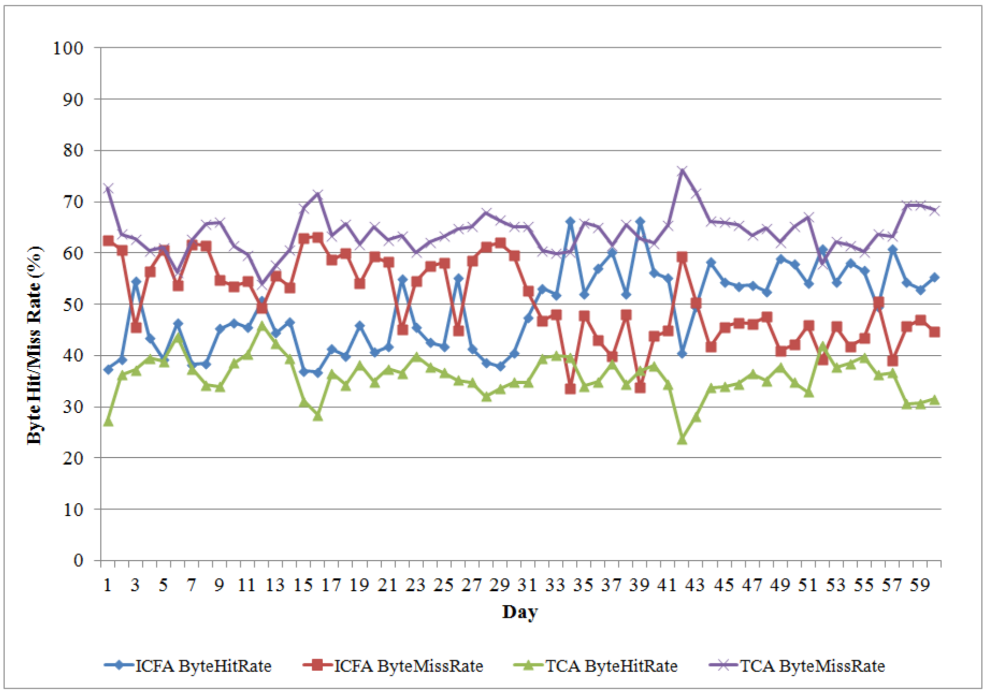


Figure 5.7: Daily comparison of caching performance between the ICFA and the TCA measured by Byte Hit Rate and Byte Miss Rate.

Since the hit rate is increased under the system of the ICFA, the simulation also illustrates that the number of byte hits is increased, as shown in Figure 5.7. It results from the grouping mechanism of cache because webs with the similar characteristics are stored and manipulated in the same area. Thus, the possibility of the required web is found in the managed caches of the ICFA is higher than storing different groups of webs in the same area as the TCA. As a result, the byte hit rate of the TCA (green line) is lower than the byte hit rate of the ICFA (blue line) while the byte miss rate of the TCA (purple line) is higher than the byte miss rate of the ICFA (red line).

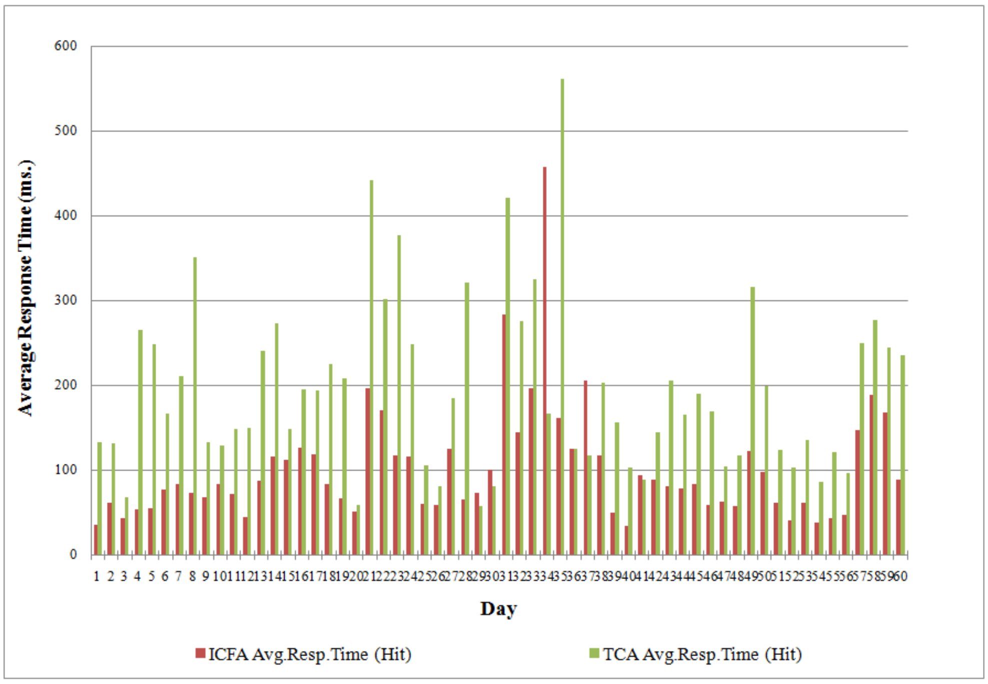


Figure 5.8: Daily comparison of caching performance between the ICFA and the TCA measured by Average Response Time of Hit.

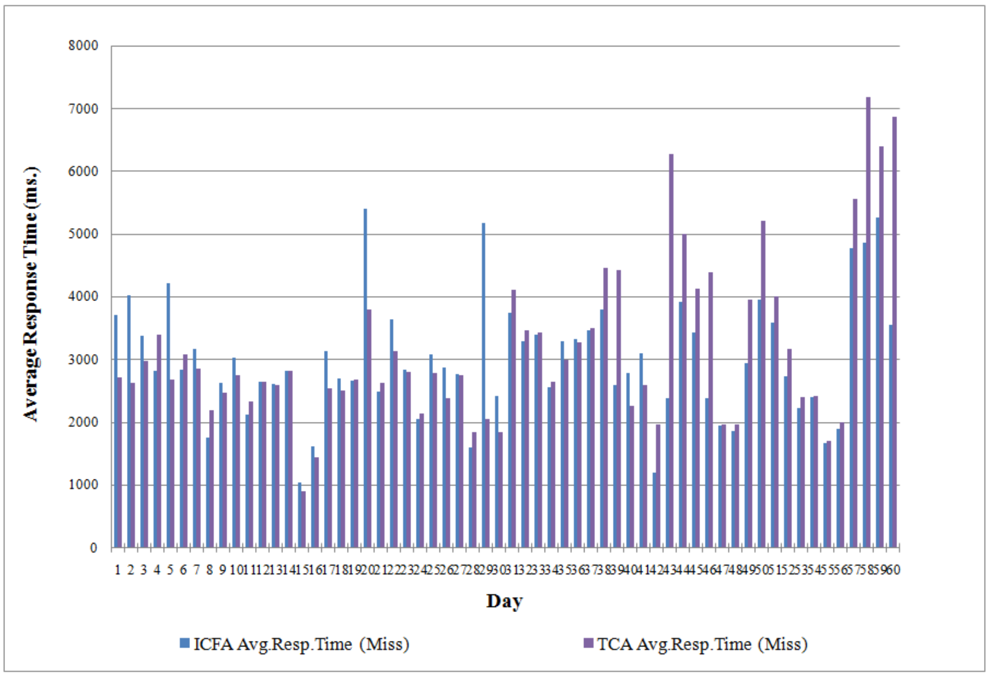


Figure 5.9: Daily comparison of caching performance between the ICFA and the TCA measured by Average Response Time of Miss.

Since the requested web in the caches of the ICFA is faster than in the TCA; the response time to obtain the required information is shorter than the TCA system (see Figure 5.8). Consequently, the users in the ICFA are faster served than the users in the TCA. Thus, the user latency is reduced. On the contrary, as can be seen in Figure 5.9, the retrieving information from the external sources of the ICFA architecture is larger than the retrieval time of the TCA. This is the result from losing the overhead to search the non-existing content in each cache.

The test results confirm that the cache management policy of the ICFA is very efficient and the QoS can be maintained even though the number of transactions continuously increases. This is the outcome from categorizing and grouping webs based on the available retrieval pattern; then, these classified webs are stored in the individual cache based on the defined groups.

CHAPTER VI

DISCUSSION AND CONCLUSION

6.1 Discussion

Since the number of Internet users is continuously increase, the issue of quality of service (QoS) is an important concern of users. Every Internet Service Provider (ISP) generally groups their customers based on their applications or organizational profiles. However, this criterion does not fit well for cache management to serve users when the size of users is expanded. Therefore, various cache management mechanisms are implemented to maintain retrieval process of users over the Internet environment.

In this research, it has shown that the proposed architecture, ICFA, is easy to manage and reduce the retrieval time comparing with the traditional cache systems that arrange the caches in the distributed model or the hierarchical model. The distributed web proxy caching [9], [52], [92] employ sophisticated caching and searching schemes to distribute and search the cached web documents. Using sophisticated caching scheme increases the complexity of proxy's management, while the proposed ICFA is designed as one layer of distributed architecture with one management scheme, the PM. Therefore, the complexity of the ICFA is less than the existing distributed web proxy caching.

Considering the hierarchical caching architecture [39], [93], this architecture needs intermediate caches with the high-quality algorithms to avoid vastly loading in the caches that will result in high retrieval time. However, this concept does not support economical issue when the number of usages increases. Therefore, the ICFA has many advantages than the hierarchical system because of the function of recommending system. According to efficiency of the recommending function, the increasing number of users will not affect to the retrieval time. Thus, there is no obligatory to expand size of available caches of ICFA although the number of users is increasing. Furthermore, there is no change in the retrieval time under such situation. Referring to the efficiency of the recommender system, users can save time to obtain their required files from the file repository or database. Therefore, applying the concepts of recommender system for classifying webs will also decrease the retrieval time from the cache farm without effects from the implemented cache replacement algorithm.

However, an interesting issue of choosing a replacement policy to implement in the system is a challenging task. This is because the policy that performs best in all environments is impossible due to various workload characteristics. One workload may make the policy performs well, and another make it performs less well. Moreover, the measurement of caching performance is important as well, thus, the hit rate and the byte hit rate are often used to perform this task. Nevertheless, other factors need to be considered for maintaining the service performance, such as implementation issue, cache size, processing power requirement, memory consumption, and where the cache server is installed [60].

According to the results of Experiment 2 presented previously, the best cache replacement algorithm for the simulation models is the LRU confirmed by the HR and the BHR. Based on the design rationale of the LRU, one of replacement policies in the recency-based category, the recently referenced documents will be referenced again in the near future. Since this research focuses on studying in the academic areas which have the obvious objective of Internet usage, most users are interested in the same or similar web documents at about the same time. So, a recently accessed web requested by one user is likely to be accessed again from other users in the near future. Consequently, this situation corresponds to the rationale of the LRU; so that the LRU is the suitable policy for the proposed ICFA environment.

Furthermore, the proposed architecture arranges the number of caches based on the number of web patterns and also the ratio of these existing groups while most of the cache architecture do not consider in the existing usage ratio among usage contents [38-39], [52], [94]. Thus, the adaptation of cache sizes will perform only within the available caches in the farm, based on the result from RAM. Unlike the proposed

ICFA technique, the existing cache farm will be re-implemented or add more cache size for every increasing number of users. Therefore, the proposed technique can save the maintenance cost in buying new hardware and time to install new cache into the legacy system.

6.2 Conclusion

Currently, the Internet becomes an important resource for people around the world since it contains various interesting contents that mostly are presented as web pages controlled by web servers. However, retrieving information from any webs within the ISPs or organizations required proxy and cache management system to filter suitable information flow in and out the organization networks. Thus, accessing performance must depend on the cache management mechanism of the proxy server and the cache farm.

One major problem of the ISPs is the unlimited increasing size of their customers where this can affect to the performance of the entire service system when cache management mechanism cannot serve all requests as suitable as it should. Various transactions must be delivered out to the Internet to retrieve information from the original source where some can find contents in the cache area. Thus, the response time for users will not be fast as expected, or sometimes, the transaction has gone down according to the congestion traffic of the Internet. So, the QoS of ISP can be dropped according to users' disappointments. Many cache management techniques were proposed to solve the problems stated above. Even though there are various techniques have been proposed and installed to increase the service quality, none of them can maintain the service quality as expected when the number of transactions expands.

This research proposed a new architecture for cache management, called Intelligent Cache Farming Architecture (ICFA). The proposed ICFA integrates the concepts of the recommender system, using frequencies and browsed file sizes, to maintain the quality of service (QoS) that is the response time for the Internet usages. The components of the BBCMM are the WPDB, the PM, and the SPSs; within the PM, there is an important module, called as the ACM. This module is responsible for classifying all retrieved webs.

The experiment has demonstrated that under the systematic organization with clear objective of browses, most of the important metrics have been changed positively although the number of transactions increases neither adding nor adjusting new hardware, including the change in the cache management policy. Therefore, this ICFA can be an eco-proxy system that provides high serving performance for any organizations.

References

[1] Wooster, R.P., and Abrams, M. Proxy Caching that Estimates Page Load Delays. Computer Networks and ISDN Systems. 29, 8-13 (1997): 977-986.

[2] Mardesich, J. The Web is no shopper's paradise. Fortune. (1999): 188-198.

[3] O'Neil, E. J., O'Neil, P. E., and Weikum, G. The LRU-k page replacement algorithm for database disk buffering. ACM SIGMOD Record. 22, 2 (1993): 297-306.

[4] Williams, S., Abrams, M., Standridge, C. R., Abdulla, G., and Fox, E. A. Removal Policies in Network Caches for World-Wide Web Documents. Proceedings of ACM SIGCOMM. (1996): 293-305.

[5] Cao, P., and Irani, S. Cost-aware WWW Proxy Caching Algorithms. Proceedings on USENIX Symposium on Internet Technologies and Systems. (1997): 193-206.

[6] Jin, S., and Bestavros, A. Greedy-dual* Web Caching Algorithm: Exploiting the Two Sources of Temporal Locality in Web Request Streams. Computer Communications. 22 (2000): 174-183.

[7] Haverkort, B. R., Khayari, R. E. A., and Sadre, R. A Class-based least-recently used caching algorithm for world-wide web proxies. Lecture Notes in Computer Science [Online]. 2794 (2003): 273–290. Available from: http://www.springerlink.com/content/7khpur60hn9q2npp/[2003, September 18].

[8] Chankhunthod, A., Danzig, P.B., Neerdaels, C., Schwartz, M.F., and Worrell, K.J. A Hierarchical Internet Object Cache. Proceedings of the 1996 Annual Conference on USENIX Annual Technical Conference. (1996): 153-164.

[9] Tewari, R., Dahlin, M., Vin, H.M., and Kay, J.S. Beyond hierarchies: Design considerations for disturbed caching on the Internet. Proceedings of the 19th

IEEE International Conference on Distributed Computing Systems ICDCS '99. (1999).

[10] Rodriguez, P., Spanner, C., and Biersack, E.W. Analysis of Web caching architectures: hierarchical and distributed caching. IEEE/ACM Transactions on Networking (TON). 9, 4 (2001): 404-418.

[11] Oneis, M.S.E., Barada, H., and Zemerly, M.J. Towards An Efficient Web Caching Hybrid Architecture. Proceedings of the 4th International Conference On Information Technology (ICIT 2009). (2009).

[12] Bhattarakosol, P., and Ngamaramvaranggul, V. An Internet web management policy for government organization. Proceedings of the 18th APAN Conference, Network Research Workshop. (2004): 249-255.

[13] Bhattarakosol, P., and Srisujjalertwaja, W. Customer-oriented policy for proxy management system. Proceedings of International Computer Symposium. (2004): 1168-1173.

[14] Hiranpongsin, S., and Bhattarakosol, P. Intelligent Caching Algorithm for Web Cache Farming System. Proceedings of 2009 International Conference on Wireless Information Networks & Business Information System WINBIS '09. (2009).

[15] Khayari, R.E.A., Best, M., and Lehmann, A. Impact of Document Types on the Performance of Caching Algorithms in WWW Proxies: A Trace Driven Simulation Study. Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA '07).1 (2005): 737-742.

[16] Jianhui, L., Tianshu, H., and Chao, Y. Research on WEB Cache Prediction Recommend Mechanism Based on Usage Pattern. Proceedings of the 1st

International Workshop on Knowledge Discovery and Data Mining (WKDD 2008). (2008): 473-476.

[17] Healy, J., Director, Networks and Systems, Squid (WWW Proxy Server). [2008, March].

[18] Proxy Server [Online]. Available from: http://en.wikipedia.org/wiki/Proxy_server. [2009, October 21].

[19] Proxy Server [Online]. Available from: www.webopedia.com/TERM/P/proxy_server.html. [2009, October 21]

[20] Proxy Systems [Online]. Available from: www.unix.org.ua/orelly/networking/firewall/ch07_01.htm. [2009, October 24]

[21] Squid (software) [Online]. Available from: http://en.wikipedia.org/wiki/Squid_(software). [2009, October 24]

[22] Squid: Optimising Web Delivery [Online]. Available from: http://www.squid-cache.org/. [2009, October 24].

[23] Squid – Proxy server [Online]. Available from: https://help.ubuntu.com/11.04/serverguide/C/squid.html. [2009, October 24].

[24] Feng, S., Zhang, J., and Zeng, B. Design of the Visualized Assistant for the Management of Proxy Server. Proceedings of the 3rd. International Symposium Electronic Commerce and Security. (2010): 204-208.

[25] Banerjee, B. Squid: Master this Proxy Server. LINUX FOR YOU. (2003): 55-62.

[26] Hofmann, M., and Beaumont, L. Content Networking Architecture, Protocols, and Practice. Elsevier, CA,: Morgan Kaufmann Publishers, (2005).

[27] Vakali, A.I., and Pallis, G.E. A Study on Web Caching Architectures and Performance, Proceedings of the 5th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI 2001). (2001): 1-5.

[28] Ajwalia, P. Proxy Server [Online]. Available from: http://www.engiguide.com/Paper-Presentation/Proxy%20Server.pdf [2010, January 10].

[29] Che, H., Tung, Y., and Wang, Z. Hierarchical Web Caching Systems: Modeling, Design and Experimental Results. IEEE Journal on Selected Areas in Communications. 20, 7 (2002): 1305-1314.

[30] Xu, Z., Bhuyan, L., and Hu, Y. Tulip: A New Hash Based Cooperative Web Caching Architecture. The Journal of Supercomputing. 35 (2006): 301–320.

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

[32] Wang, J. A Survey of Web Caching Schemes for the Internet. Computer Communication Review. 29, 5 (1999): 36-46.

[33] Oneis, M.S.E., Zemerly, M.J., and Barada, H. Intelligent Exploitation of Cooperative Client-Proxy Caches in a Web Caching Hybrid Architecture. Computational Intelligence and Modern Heuristics: Artificial Intelligence. InTech Publisher, (2010).

[34] Michel, S., Nguyen, K., Rosenstein, A., Zhang, L., Floyd, S., and Jacobson, V. Adaptive Web caching: towards a new caching architecture. Computer Network and ISDN Systems. (1998).

[35] Yang, J., Wang, W., Muntz, R., and Wang, J. Access driven Web caching. UCLA Technical Report # 990007. (1999).

[36] Foygel, D., and Strelow, D. Reducing Web Latency with Hierarchical Cache-Based Prefetching. Proceedings of the 2000 International Workshop on Parallel Processing, (2000).

[37] Lim, H., and Du, D. H. C. Design considerations for hierarchical Web proxy server using iSCSI. Proceeding of 2003 Symposium on Applications and the Internet. (2003): 414-417.

[38] Li, W., Wu, K., Ping, X., Tao, Y., Lu, S., and Chen, D. Coordinated Placement and Replacement for Grid-Based Hierarchical Web Caches. Lecture Notes in Computer Science. 3795 (2005): 430-435.

[39] Yang, F. H., and Chi, C. H. Using Hierarchical Scheme and Caching Techniques for Content Distribution Networks. Proceedings of the Third International Conference on Semantics, Knowledge and Grid. (2007): 535-538.

[40] Mateescu, R., and Wijs, A. Hierarchical Adaptive State Space Caching Based on Level Sampling. Lecture Notes in Computer Science. 5505 (2009): 215-229.

[41] Busaria, M., and Williamson, C. ProWGen: a synthetic workload generation tool for simulation evaluation of web proxy caches. Computer Networks. 38, 6 (2002): 779-794.

[42] Shi, L., and Zhang, Y. Optimal Model of Web Caching. Proceedings of the Fourth International Conference on Natural Computation. (2008): 362-366.

[43] Shi, L., Yao, P., Wei, L., and Tao, Y. Cost-benefit Analysis of the Web Hierarchy Caching Model. Information Technology Journal. 11, 3 (2012): 364-367.

[44] Wessels, D., and Claffy, K. Internet cache protocol (ICP), version 2, RFC 2186.

[45] Valloppillil, V., and Ross, K. W. Cache array routing protocol v1.0, Internet Draft <draft-vinod-carp-v1-03.txt>.

[46] Povey, D., and Harrison, J. A distributed Internet cache. Proceedings of the 20th Australian Computer Science Conference. (1997).

[47] Wang, Z. Cachemesh: a distributed cache system for World Wide Web, Web Cache Workshop. (1997).

[48] Rousskov, A., and Wessels, D. Cache Digest. Proceedings of 3rd International WWW Caching Workshop. (1998).

[49] Relais: cooperative caches for the World Wide Web [Online]. Available from: http://www-sor.inria.fr/projects/relais/. [2008, September 30].

[50] Fan, L., Cao, P., Almeida, J., and Broder, A. Z. Summary cache: A scalable wide-area web cache sharing protocol. IEEE/ACM Transactions on Networking. 8, 3 (2000): 281-293.

[51] Gadde, S., Rabinovich, M., and Chase, J. Reduce, reuse, recycle: an approach to building large Internet caches. Proceedings of the HotOS'97 Workshop [Online]. Available from: http://www.cs.duke.edu/ari/cisi/crisprecycle/crisp-recycle.htm. [2010, May 2].

[52] Piatek, M. Distributed web proxy caching in a local network environment. Available from: www.acm.org/src/subpages/papers/piatek.src.2004.pdf. [2008, March 15].

[53] Manikandan, C. V., Manimozhi, P., Suganyadevi, B., Radhika K., and Asha, M. Efficient load reduction and congession control in Internet through multilevel Border Gateway Proxy Caching. Proceeding of 2010 IEEE International Conference on Computational Intelligence and Computing Research. (2010): 1-4.

[54] Rabinovich, M., Chase, J., and Gadde, S. Not all hits are created equal: cooperative proxy caching over a wide-area network. Computer Networks And ISDN Systems. 30, 22-23 (1998): 2253-2259.

[55] Baek, J., Kaur, G., and Yang, J. A New Hybrid Architecture for Cooperative Web Caching. Journal of Ubiquitous Convergence and Technology. 2, 1 (2008).

[56] Abrams, M., Standridge, C. R., Abdulla, G., Williams, S., and Fox, E. A. Caching proxies: limitations and potentials. Proceedings of the 4th International WWW Conference. (1995).

[57] Podlipnig, S., and Boszormenyi, L. A Survey of Web Cache Replacement Strategies. ACM Computing Surveys. 35, 4 (2003): 374–398.

[58] Dilley, J., Arlitt, M., and Perret, S. Enhancement and Validation of Squid's Cache Replacement Policy. HPL-1999-69. (1999).

[59] Balamash, A., and Krunz, M. An Overview of Web Caching Replacement Algorithms. IEEE Communications Surveys: The Electronic Magazine of Original Peer-Reviewed Survey Articles. 6, 2 (2004).

[60] Wong, Kin-Yeung. Web Cache Replacement Policies: A Pragmatic Approach. IEEE Network. (2006): 28-34.

[61] Jin, S., and Bestavros, A. Greedy-dual* web caching algorithm: Exploiting the two sources of temporal locality in web request streams. Computer Communications. 22 (2000): 174–283.

[62] Lindemann, C., and Waldhorst, O. Evaluating the impact of different document types on the performance of web cache replacement schemes. Proceedings of IEEE Int'l Performance and Dependability Symposium. (2002): 717–726.

[63] Bahn, H., Koh, K., Noh, S.H., and Min, S.L. Efficient replacement of nonuniform objects in web caches. IEEE Computer. 35 (2002): 65–73.

[64] Pallis, G., Vakali, A., and Sidiropoulos, E. FRES-CAR: an adaptive cache replacement policy. Proceedings of the International Workshop on Challenges in Web Information Retrieval and Integration. (2005): 74-81.

[65] Bian, N., and Chen, H. A Least Grade Page Replacement Algorithm for Web Cache Optimization. Proceedings of the First International Workshop on Knowledge Discovery and Data Mining. (2008): 469-472.

[66] Geetha, K., Gounden, N. A., and Monikandan, S. SEMALRU: An Implementation of modified web cache replacement algorithm. World Congress on Nature & Biologically Inspired Computing. (2009): 1406-1410.

[67] Zhang, B., and Wu, H. A new distributed caching replacement strategy. IEEE 3rd International Conference on Communication Software and Networks. (2011): 167-170.

[68] Holton, D. A., and Sheehan, J. The Petersen Graph. Cambridge University Press, NY.

[69] Hauger, S., Tso, K.H.L., and Schmidt-Thieme, L. Comparison of Recommender System Algorithms focusing on the New-Item and User-Bias Problem. Proceedings of the 31st Annual Conference of the German Classification Society on Data Analysis, Machine Learning, and Applications. (2007).

[70] Ricci, F., Rokach, L., and Shapira, B. Introduction to Recommender Systems Handbook. Recommender Systems Handbook, Springer, (2011): 1-35.

[71] Schafer, J.B., Frankowski, D., Herlocker, J., and Sen, S. Collaborative filtering recommender systems. The Adaptive Web. Springer Berlin/Heidelberg (2007): 291–324.

[72] Schafer, J.B., Konstan, J.A., and Riedl, J. E-commerce recommendation applications. Data Mining and Knowledge Discovery. 5(1/2), 115–153 (2001).

[73] Breese, John S., Heckerman, D., and Kadie, C. Empirical analysis of predictive algorithms for collaborative filtering. Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98). Morgan Kaufmann. (1998): 43–52.

[74] O'Donovan, J., and Dunnion, J. A framework for evaluation of collaborative recommendation algorithms in an adaptive recommender system. Proceedings of the International Conference on Computational Linguistics (CICLing-04). (2004): 502–506.

[75] Bueno, D., Conejo, R., and David, A. A. METIOREW: An Objective oriented content based and collaborative recommending system. Lecture Notes in Computer Science. 2266 (2002): 310-314.

[76] Linden, G., Smith, B., and York, J. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. IEEE Internet Computing. 7, 1 (2003): 76-80.

[77] Itmazi, J., and Megías, M. Using Recommendation Systems in Course Management Systems to recommend Learning Objects. The International Arab Journal of Information Technology. 5, 3 (2008): 234-240.

[78] Bobadilla, J., Serradilla, F., Hernando, A., and MovieLens. Collaborative filtering adapted to recommender systems of e-learning. Knowledge-Based Systems. 22 (2009): 261-265.

[79] Souali, K., Afia, A. E., Faizi, R., and Chiheb, R. A new recommender system for e-learning environments. Proceedings of International Conference on Multimedia Computing and Systems. (2011): 1-4.

[80] Gong, Y., and Xue, Q. Study on internet recommendation system of collaborative filtering based on scatter difference. Proceedings of International Conference on Computer, Mechatronics, Control and Electronic Engineering. 1 (2010): 160-163.

[81] Julashokri, M., Fathian, M., Gholamian, M. R., and Mehrbod, A. Improving Recommender System's Efficiency Using Time Context and Group Preferences. Advances in Information Sciences and Service Sciences. 3, 4 (2011): 162-168.

[82] Squid configuration directive logformat [Online]. Available from: http://www.squid-cache.org/Doc/config/logformat/. [2009, May 30].

[83] Ravi, J., Yu, Z., and Shi, W. A survey on dynamic Web content generation and delivery techniques. Journal of Network and Computer Applications. 32, 5 (2009): 943-960.

[84] Amer, A., Long, D.D.E., and Burns, R.C. Group-Based Management of Distributed File Caches. Proceedings of the International Conference on Distributed Computing Systems. (2002): 525-534.

[85] Web Directory [Online]. Available from: http://websearch.about.com/od/enginesanddirectories/a/subdirectory.htm. [2009, November 6].

[86] Advantages of web directory [Online]. Available from: http://www.iboldesign.com/ibol-news/advantages-of-web-directory.html. [2009, November 6].

[87] Balabanovic, M., and Shoham, Y. Fab: Content-based, collaborative recommendation. Communications of the ACM. 40, 3 (1997): 66-72.

[88] Ricci, F., Rokach, L., Shapira, B., and Kantor, P.B. Recommender Systems Handbook, Springer, New York,: Dordrecht Heidelberg London, (2011).

[89] Cache replacement policy [Online]. Available from: http://www.squid-cache.org/Versions/v2/2.7/cfgman/cache_replacement_policy.html. [2010, May 25].

[90] Bahn, H. Web cache management based on the expected cost of web objects. Information and Software Technology. 47 (2005): 609–621.

[91] SPSS Annotated Output: T-test [Online]. Available from: http://www.ats.ucla.edu/stat/spss/output/Spss_ttest.htm. [2011, November 26].

[92] Touch, J. The LSAM Proxy Cache – a Multicast Distributed Virtual Cache. Proceedings of the 3rd Int. WWW Caching Workshop. (1998).

[93] Che, H., Wang, Z., and Tung, Y. Analysis and Design of Hierarchical Web Caching Systems. Proceedings of IEEE INFOCOM'01. 3 (2001): 1416-1424.

[94] Laoutaris, N., Syntila, S., and Stavrakakis, I. Meta Algorithms for Hierarchical Web Caches. Proceedings of 2004 IEEE Int. Conf. on Performance, Computing, and Communications. (2004): 445-452.

Biography

Name: Miss Supawadee HIRANPONGSIN.

Date of Birth: 1st April, 1978.

Educations:

- Ph.D., Program Computer Science and Information Technology, Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University, Thailand, (June 2007 - May 2012).
- Visiting scholar, Department of Computer Science, Purdue University, USA, (September 2009 - August 2010).
- M.S., Program Computer Science, Faculty of Applied Statistics, National Institute of Develpoment Administration (NIDA), Thailand, (October 2004 - March 2007).
- B.Sc., Program Mathematics, Department of Mathematics, Faculty of Science, Chulalongkorn University, Thailand, (May 1996 - April 2000).

Publication papers:

- S. Hiranpongsin and P. Bhattarakosol, "Enhancing the QoS of Proxy Cache Using the Squid Log File," In Proceedings of International Conference on INTERNET STUDIES (NETs2012), Bangkok, Thailand, August 17-19, 2012 (In Press).
- S. Hiranpongsin and P. Bhattarakosol, "Intelligent Caching Algorithm for Web Cache Farming System," In Proceedings of 2009 International Conference on Wireless Information Networks & Information System (WINBIS 09), Kathmandu, Nepal, February 27- March 1, 2009.
- S. Hiranpongsin and P. Bhattarakosol, "Intelligent Cache Farming Architecture with the Recommender System," Journal of Engineering Science and Technology (JESTEC), Vol. 4(2), pp. 206-219, June 2009.
- S. Hiranpongsin and P. Bhattarakosol, "Intelligent Cache Farming Architecture with the Recommender System," In Proceedings of 2008 International Conference on Networks, Applications, Protocols, and Services (NetApps2008), Universiti Utara Malaysia, Malaysia, November 21-22, 2008.
- S. Hiranpongsin and P. Bhattarakosol, "Intelligent cache farming Architecture for E-Business services," In Proceedings of 7th International Conference on e-Business 2008 (INCEB 2008), Bangkok, Thailand, pp. 140-147, November 6-7, 2008.

Work:

- Analyst, Funds and Liquidity Management Department, Treasury Division, Bangkok Bank P.C.L. (Headquarters), (2000 - 2004).

Scholarship:

- THE 90th ANNIVERSARY OF CHULALONGKORN UNIVERSITY FUND (Ratchadaphiseksomphot Endowment Fund), (May 2011).
- The Inter-University Network (UniNet) Unit under the Office of the Higher Education Commission (OHEC), Thailand. (February 2010).
- University Development Commission (UDC) Scholarship from Ubon Rajathanee University, Thailand. (October 2004 – September 2011).
- Chulalongkorn University Scholarship (1997 - 2004).