A Study for IE Browser Cache through Browser Tracing Tool

Yong Lu (lu@cs.wisc.edu)
Bo Zhan (bo@cs.wisc.edu)

May 12, 2000

1 Executive Summary

World wide web has been more and more connected to people's daily life. Internet explorer (IE)and Netscape are two main commercial web browsers. In both cases, browser cache is used to accelerate the browsing process. But people may ask how the browser cache works and if it really enhances the browsing performance? In this project, we study the browser cache in IE 5.0.

Since cache mechanism is based on the locality, user's browsing behavior plays an important role in the web browsing performance. Thus, the second part of this study focuses on the user behavior. By tracing the user browsing history, we aim to extract users' basic browsing behaviors and determine the extent to which the cache can help.

Finally, we implement a cache monitor tool to help user browse more efficiently.

Section 2 describes how to implement the browser tracing tool. The IE cache policy is analyzed in section 3 and the user behavior is studied in section 4. Some related works and speculations on the future investigation are presented in Sections 5 and 6. Figure 1 shows the basic structure of this project.

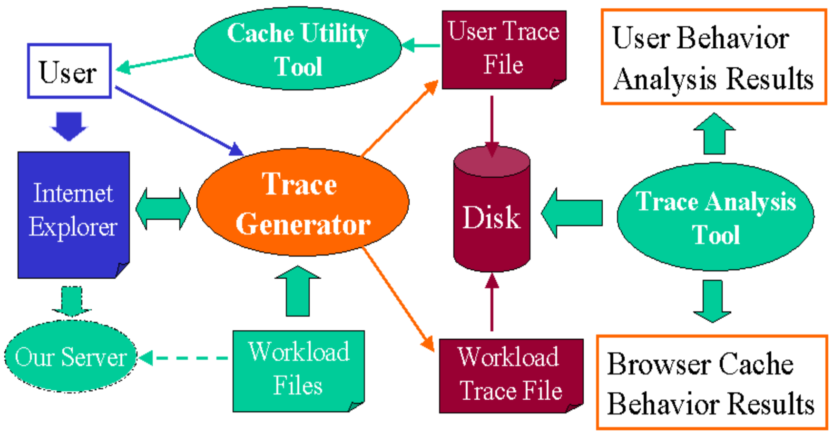


Figure 1: The general structure of the study.

2 Browser Tracing tool design and implementation

2.1 Web browser trace generator

Our program is designed on the Windows NT platform. Microsoft Corp. has developed Visual C++ and Visual Basic language based on its Windows platform. The main benefit of these languages is that they are high level languages and provide a lot of libraries to access the kernel operations. In our trace generator, we use such libraries to access the IE's web cache directory and get the cache information. When the trace generator is executed, it will start an IE instance and prepare to accept any actions from the user. The main functions of our trace generator are to record the user behaviors and test the web cache policy.

To log the user behaviors, our trace generator provides three basic features: going backward, going forward, and navigating (Figure 2). Meanwhile, the user can navigate the web page from the IE instance. Our trace generator records the actions both from the IE instance and the basic functions built in the trace generator. The trace file is stored on the user's local disk: C:/temp/myTraceFile.txt. Whenever the user starts the trace generator, the records will be appended to the end of the trace file, which is used to analyze the user behaviors. To test the web cache policy, our trace generator first opens the

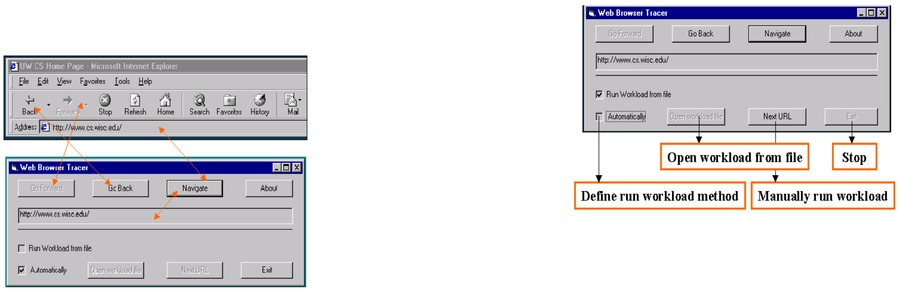


Figure 2: The interface for user trace collection (left); The interface for cache study (right)

workload file from disk, then we setup the navigating methods. Our trace generator provides two kinds of navigating methods. One is to click on the "Next URL" button, the generator reads the web page from the workload file one by one. The other is to automatically read the next web page request from the workload file once the last request is loaded. The second method save us a huge amount of time. However, our experiments confirmed that the first method is very important in helping us exam some special IE cache policies, especially the time interval.

2.2 The trace file generation

Our trace generator can generate user behavior trace file and workload trace file. Figure 3 shows this process. These two trace files have the same file format. This is very convenient for analysis. Before a page is navigated, the trace generator detects and put down the time and page name. After the page is downloaded completely, the trace generator will parse the page since it also includes gif files or frames, and record the time and file names. Also the generator will call the dynamic library, which will call a special window API to get other cache information, such as hit rate, file access time. In our analysis, we only use the hit rate as a reference, since we found that this window API has some bugs and has not been implemented completely. The workload trace file is written to the same directory of the workload file. The user behavior trace file is written to the C:/temp/myTraceFile.txt.

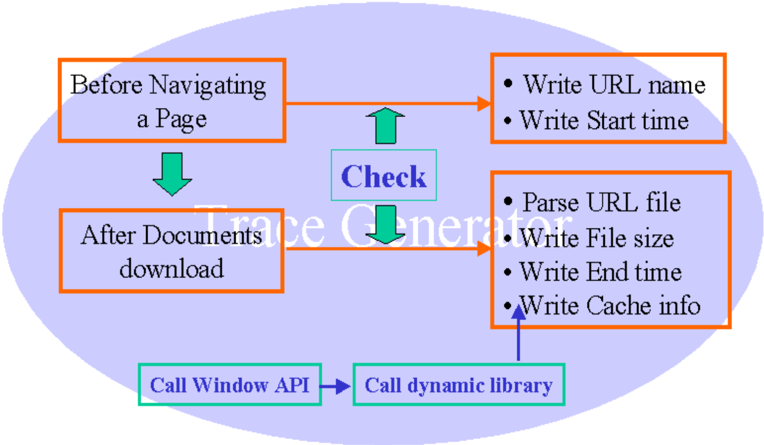


Figure 3: The process of trace file generation.

3 How does IE 5.0 cache work

In this section, we will discuss the analysis of IE 5.0 cache policies. We have not found a way to access the IE cache implementation document. But there are many cache related tools. Some claim that IE 3.0 or 4.0 is really just a random replacement policy and from IE 5.0, it is modified to LRU.

3.1 Study environment

1. Server
   Apache program is installed on a NT 4.0 machine as our server. There are three types of files with varied size stored on server:
   - Pure text file
   - Pure graph file, mainly .gif file
   - Text file embedded with two .gif file
2. Client
   Client side is any PC with IE 5.0. They are connected to the server through 56k modem (home) or Ethernet (on campus).
3. Workload
   The workload consists of a list of URLs contained in our dedicated server. They vary in file size, file type, browsing sequence, etc.
4. Browser tracing tool
   As afore-mentioned, the workload file will feed into browser tracing tool. At this time, it can invoke IE and browse automatically or manually. The trace file will be generated while browsing.

5. Cache monitor tool
   We programmed a cache size monitor tool. which checks the cache size regularly and reports the cache directory. Besides, we have downloaded a small tool that can report every flush of the old file, addition of the new file and the age of the oldest file. It supplements the tracing tool, which records all the addition of the files into cache. Through these tools, we can monitor the file activities in the cache directory effectively.

6. Cache analysis tool
   The input of cache analysis tool is the trace file. The output are all kinds of statistic information from the trace file: file name, file size, retrieval time, cache hit rate, etc.

3.2 Methodology

To determine how IE cache works, we treat IE browser as a black box and apply designed workloads as probes. The cache size limit is set beforehand, then the output trace file and cache directory files are analyzed (Figure 4). During the browsing, the cache directory is monitored and the useful information

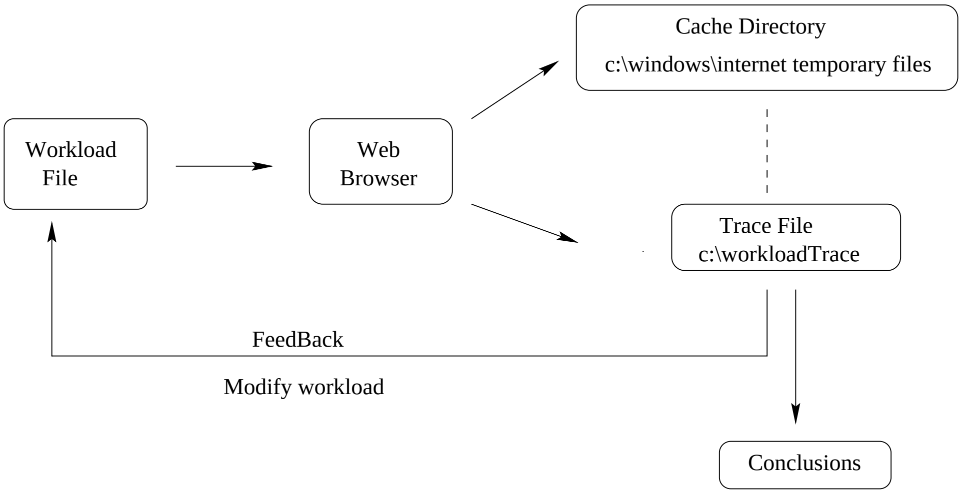


Figure 4: Analysis process

is recognized. Then, more meaningful workload is modified to plugin again. The process to understand the feedback information is not automatic. We begin with the simplest workload for basic replacement policy, then additional affecting factors are added.

3.2.1 Initial problems

After setting the cache size limit, we specify the workload size larger than the cache size so that the replacement can be induced. But in the initial experiments, we run the workload automatically and no replacement occurs even though the workload size is much larger than the cache size. Only when we apply the workload manually, cache directory accumulates to some extent and replacement can be caught occationally. When we restart the client computer, usually the cache directory size can be reduced greatly. Several factors should be taken into consideration, namely, the locality of the files (because we are fetching files from the same server), the file type, the browsing speed and the cache clean cycle of IE, etc. Interestingly, when we try the workload on a home PC (56K modem), the situation is better though the accumulated cache size still exceeds the cache limit. It seems that the browsing speed depends on the replacement policy.

To prove this hypothesis, we performed a set of experiments with various browsing interval time. The cache limit is set to 1M, the workload size is 6.5M with uniform file size of 130K. The cache directory size is measured while browsing. It takes about 1.4 seconds to get one 130K file from the server. Therefore, 1.4s interval corresponds to the automatic browsing. All the experiments are started from empty cache, though it may not be exactly 0 byte. The cache accumulation at different browsing speeds is shown in Figure 5. As the browsing interval increases, cache accumulation is reduced. Finally, it is controlled by the cache limit setting. Figure 6 shows the cache size variation with the browsing time. Interestingly, the first cache replacement happens only after the browsing time exceeds 10 minutes. In other words, the oldest file stays in the cache at least for 10 minutes. This observation explains all the unexpected problems. More experiments prove that the cache limit functions well as long as there are enough ”old” files in the cache. It took us much time to identify this, though it may not bother normal users, who often do not understand the cache functions.

Browsing Interval	Cache Size
1.3 sec	6.6 M
0.5 min	2.7 M
1 min	1.5 M
2 min	0.9 M
3 min	0.9 M

Figure 5: The relation between cache size accumulation and browsing interval.

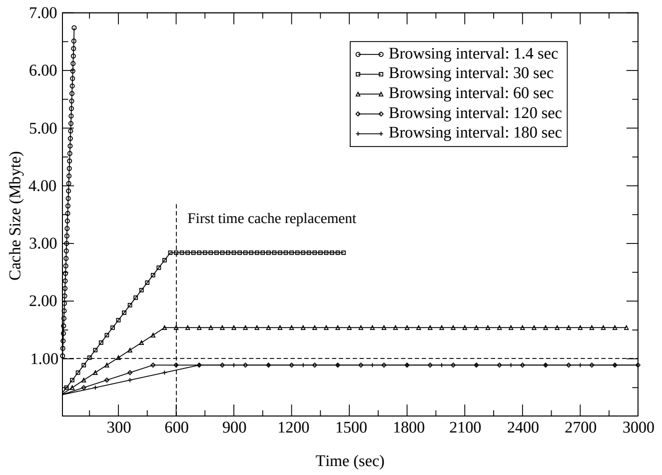


Figure 6: Cache size versus browsing time.

This cache policy is confirmed from documents on the IE website. The scavenger thread that clears up the IE cache has a minimum cache life limit of 10 mins. If an item in the cache is less than 10 min old, the scavenger thread does not remove the item from the cache and the hard disk is possibly filled up.

3.2.2 Basic replacement policy

To study the basic replacement policy, we use workload files that consist of uniform sized files with different browsing sequences. Figure 7 shows that the trace files show two dramatically different cache hit ratios, when using these two different browsing sequences. It is based on LRU (Least Recent Used) policy. The oldest file is replaced first.

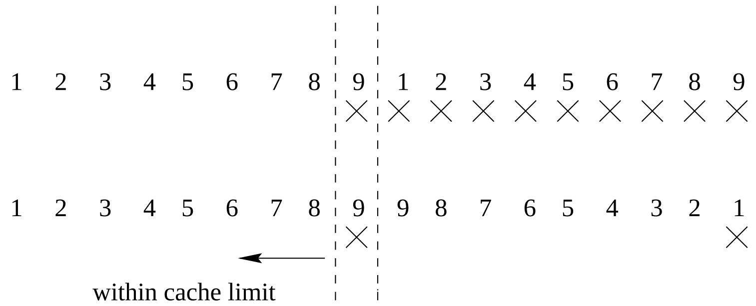


Figure 7: LRU replacement policy

3.3 Watermark

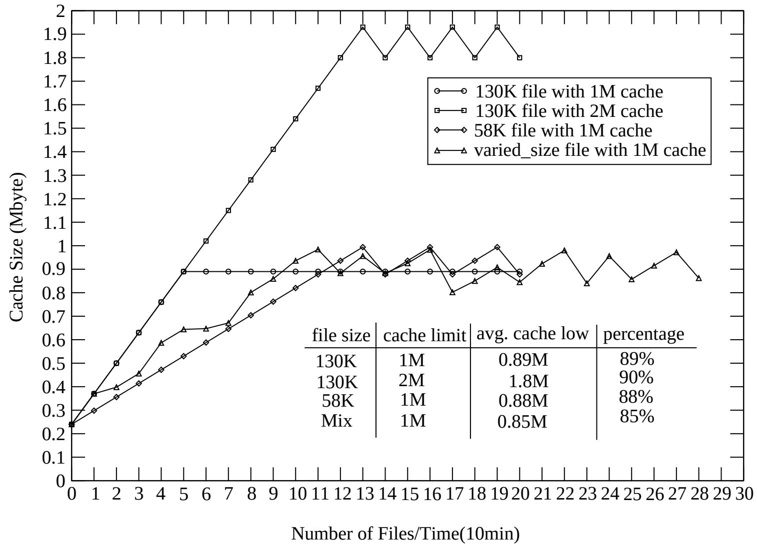


Figure 8: Cache size variation with time due to replacement.

Next, we study how much room is cleared when the cache is full. In this set of experiments, we run the workload at 10 minutes interval, therefore every file is qualified to be replaced if needed. Figure 8 shows the result. For 130k files, each replacement only removes one file, the cache size is therefore unchanged. For 58k files, each replacement can remove multiple files if needed, specifically each time 3 files are replaced. For the workload with distinct file size, multiple files are replaced in one time too. When the cache limit is set to 2M, 2 130K files are replaced each time from the cache. We assume there is a watermark that states the maximum cache size that can be cleared. From our average data shown in Figure 8, 85% of the cache limit is a reasonable watermark.

3.3.1 File browsing frequency

To test if the file visiting time is taken into the consideration for replacement, we design the workload file containing files that have been visited multiple times. By monitoring the cache directory files activities, we see that these files are treated specially. Knowing that there is a 10 minutes minimal stay in the cache, we expect an "effective age" for pages visited multiple times. We propose a tentative formula to calculate the effective age when the visiting times is below 4 times. For visiting rates above 4 times, it seems to have the similar stay to pages visited 3 times .

$$EffectiveAge = Age - VisitingTimes * 10(min) \tag{1}$$

So far, we have not completely understood how IE solves this issue. Another way is to write a simulator with our proposed policy and compare the difference.

3.3.2 File type

When user types in a URL address and the browser brings back the requested page, usually this page contains many types of files in addition to the text file, for instance, .gif file. Would IE cache policy treat those graph files differently? To answer this, we design two types of workload: one contains pure .gif file and pure text file; the other contains embedded gif file. For the first type of workload, .gif file is treated the same way as the text file. The oldest page is replaced first. But for the second type of workload, embedded .gif file is kept longer in the cache. Probably, those embedded .gif files can appear multiple times in different URL pages. Thus, they are assumed to have higher reusability. The detailed policy for embedded .gif file need further study.

3.3.3 File size

By varying the workload file size (from a few kbyte to 100 kbyte), we can study the effect of file size on the replacement. The result does not show any special treatment for different file size.

3.4 Summerize

Up to now, we have some ideas on how IE cache works. It is based on URL and includes some special treatments. In summary, we can write an algorithm based on what we have learned. It would be better if we can build an IE cache simulator based on this algorithm and run the real trace to see how well the cache simulator matches the real IE cache.

There are still other factors that we did not study. Our workload file is much simplified in two ways: the workload files are either pure text of pure .gif, and they are all from the same server. We did not study if the locality of the file will affect the replacement policy. Presumably, files from the same server are kept longer.

4 User browsing behavior analysis

In this section, we will analyze the user behaviors. The browser tracing program and a simple survey are distributed to users, who are mainly students and working people. But all have higher education. This may limit the representativity of data. On the other hand, the survey for this group of users reflects an optimistic scenario.

4.1 Trace file

The tracing program generates the trace file while user browses. The data contains roughly one week's browsing history from each user. Unfortunately, we did not get back all the trace files due to the lack of followup. All the current analysis is based on 6 users who own a PC (Figure 9). The result shows the average cache hit rates are similar (70%) for different users, while data from session to session varies much (from 25% to 80%) for each user. The byte hit rates are generally well below the hit rates except for user2. The maximum accessed file size is within 10M for one week, while the default cache size is 60M and the default cache stay is 20days. Hence, the cache size is far above necessary and the 30% cache miss is attributed to the randomness of the user browsing. This randomness is either from the random URL or the dynamic loaded pages. Therefore, we believe that cache help users, especially those who use lower connection speed. For example, the saving is 10 times on my home computer connected through a 56k modem, but for high speed LAN connection, there is much difference. On the other hand, the big cache setting occupies a large portion of disk.

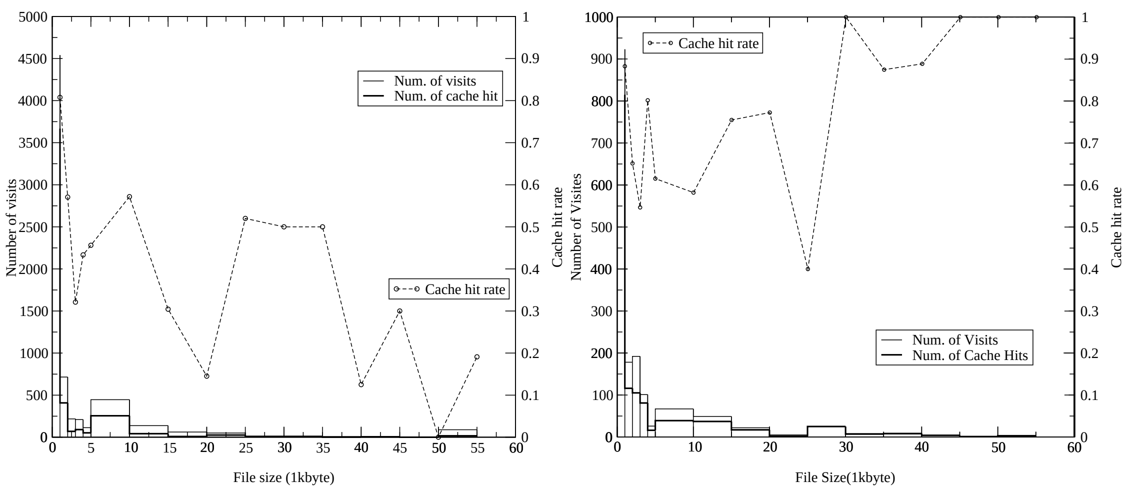
We can also conclude that user browsing behaviors represent the locality. Nearly for 70% time, they are visiting pages already visited. One can record the retrieval time of each file to see how much time can be saved.

User ID	Hit Rate	Byte Hit Rate	Avg. File Size(byte)	Total File Size(byte)
User 1	69.15%	34.47%	4206	16M
User 2	79.11%	78.53%	3124	5M
User 3	75.34%	32.17%	2025	1.5M
User 4	66.11%	35.47%	1655	1.5M
User 5	73.46%	33.80%	2563	5M
User 6	78.58%	42.53%	4216	2M

Figure 9: The statistics of user browsing trace.

4.2 Evaluation

For most users, the byte hit rate is less than the cache hit rate. Figure 10 (a) illustrates that the mostly accessed files have sizes below 5k. The cached file size distribution shows the same trend. Because the cache hit rate is lower for larger files, the average byte hit rate is lower than the cache hit rate. However, user2's access pattern in Figure 10 (b) is a special case, the cache hit rate is higher for larger files, which explains why the byte hit rate remains the same as the hit rate for user2. In fact, user2 has a favorable web site containing large graphic files. The big difference between the cache hit rate and byte cache hit



(a) (b)

Figure 10: (a) File size distribution for all users, (b) File size distribution for user2.

rate also reminds us that the bigger file cache miss will lower the performance more than smaller one. But the LRU basically treats everybody the same way.

4.3 Survey

Our survey covers some simple questions about the web browser and browser cache, from which we conclude:

- Netscape and IE are two main browsers: 42% prefer IE, 33% prefer Netscape, 17% choose either and 8.3% choose others. Hence, our study on IE presents generality.
- Most people(92%) knows cache, but only 17% adjusts the cache setting, while 67% people thinks that cache helps accelerate browsing. Even in our more educated user group, most of time, people leaves the control to the system.
- Speed and stability are the most concerns of the users.
- The monitor type tool is doubted. People prefer more intelligent browser that can configure itself. We wish we got this result earlier.

Overall, we think people would like to have cache utility ready and cache helps to some extent.

5 Cache Usage Display

We have shown how IE cache works and what typical user browsing behaviors are. As we all know, IE has a few cache configuration functions, though few users use it. It motivates us to design a small dynamic display tool working like a common disk or CPU utilization tool and help users make more intelligent usage of browser cache.

Our dynamic display tool leverages the existent IE browser tracing tool. It is simply a thread that keeps checking the information stored in the trace file. Presently, we only explore the cache hit ratio. This statistical information will provide users with general usage informations of his cache, and adjustment can be made as necessary. Figure 11 shows the user interface for this tool.

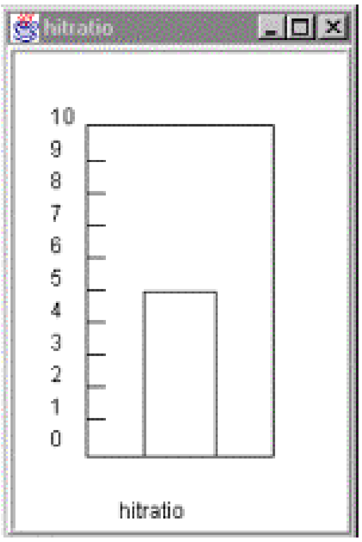


Figure 11: Cache hit rate monitor.

6 Related work

Basically, the cache optimizing for browser is only useful for those user who has very slow bandwidth such as using phone lines. For high speed LAN users, cache doesn't matter very much. Currently, there are five kinds of technique used for speed up web browser speed. One interesting technique is that they translate your DNS address to IP address so that it saves the time the browser asking DNS name server for the IP address!

Below is a basic summary of each technique in current commercial market:

1. ExtraDNS is a handy little system-tray tool that can speed up your access to the Internet. All you have to do to gain a noticeable improvement in performance is to run the program. Each time you access an Internet site, ExtraDNS sends requests to several DNS servers and caches the information locally. The next time you access the site, the request is handled by ExtraDNS, shaving up to several seconds off the time that would have been required to access the site. While the site is already loading, it checks its cache against the information returned by the DNS server, making sure it's up-to-date. You can edit the list of 10 DNS servers as you like and easily disable the program if needed. Excellent documentation is included. Reviewed on Feb 27 2000.

2. SpeedWiz is a set of utilities you can use to squeeze the last drop of performance from your PC. The Web Optimizer translates bookmark addresses for Internet Explorer and Netscape to IP addresses, speeding access to these sites. The Disk Optimizer lets you optimize virtual memory and name caching, and the System Optimizer lets you eliminate delays caused by your keyboard and menus. It also optimizes your cache for accessing hard drives and CD-ROMs.

3. InterQuick boosts your Internet access by optimizing the connection between your computer and the Internet. It sits between your browser and your Internet link, using a variety of tools to maximize the available bandwidth and performing most of its magic in the background. Tools include IQfetch, which pre-fetches pages linked to the current page you're viewing to speed up browsing; IQstore, which provides smart caching; IQfilter,which prevents content such as advertisements, pop-up windows, background sounds, images, and applets from being downloaded; IQschedule for scheduling site downloads; and IQblock for permitting or denying access to specific sites.InterQuick is highly configurable both for general settings and for providing site-by-site rules. It can be used on an individual PC or on a LAN sharing a single Internet connection.

4. Use InternetTweak (previously well known as NetMaster) to tweak your desktop and Internet settings. It provides access to a variety of settings you can change, including Windows DUN Performance, Internet Explorer Enhancements, Outlook Express Enhancements, and Netscape Enhancements

5. WinRamTurbo Pro is a handy little utility you can use to optimize your PC's performance. The attractive interface provides a clear graphical representation of memory usage, swap file usage, and CPU load. It also shows the memory load and available resources. Much more than a simple display program, WinRamTurbo Pro runs quietly in the background, doing its best to maximize real memory and minimize virtual memory usage, which can greatly enhance overall performance. Set the memory trigger amount and the amount to attempt to free. When your real memory level is reduced to the trigger level, WinRamTurbo Pro springs into action, reclaiming what memory it can up to the specified amount.

For user browsing behavior studies, at least we know that there is a group of people in Boston University who have done lots of work. They have a very large user pool. Similar file access pattern is observed. The cache study covers three different levels: individual user sessions, individual hosts and LAN level. They instrumented a version of Mosaic to get the trace file. Different cache strategies and the effects of cache size were studied. But they did not study the policy in details.

7 Future work

There are three aspects that we would like to explore further. The first is the cache replacement policy. Locality of the requests is one factor we would like to study.

Secondly, we think there may be other implement details. Instead of studying more details, which we think is more difficult, a reasonable cache simulator that involves all we have already known can be built. Then a trace driven simulation can be performed.

Thirdly, for the user behavior analysis, we would like to include more users with different background and connection speed. The evaluation could involve the time saved from cache. To study other web browsers, the tracing tool implementation may be different, but the way to analysis those is the same.

In the future, we will discuss how to study the Netscape's cache replacement policy. Although Microsoft does not provide an active control of Netscape web browser, we can generate the user behavior trace file by monitoring the mouse activities and keyboard. We can also obtain the real downloaded files for a web page by opening URL with TCP/IP protocol and getting file header information, which includes file modification time, file size, etc. In this way, we can analyze user behaviors. Also we can write a program to simulate mouse activity and run workload file from Netscape automatically. The result can be used to analyze Netscape cache replacement. Also we can write a small program to monitor the cache by reading the trace file from disk.

8 Conclusion

In this project, we studied web browser cache and related user browsing behaviors. All the study is performed on IE 5.0. The cache replacement policy is based on LRU with a minimum cache stay of 10 minutes. In addition, file visiting frequency and embedded graphic file are considered. When the cache is full, old files are flushed out until the cache size reduced to 85% of the full capacity.

The user trace file shows that cache helps the browsing. In evaluating the cache efficiency, cache hit rate and byte hit rate are used. The cache hit rate is more biased to smaller files, while the byte hit rate provides more objective information. The trace file shows that most accessed files are small. The LRU policy more favors smaller files, since the file size is excluded. However, larger file miss lower the performance more than smaller files. We think that cache management policy should involve the file size.

Finally, users rely more on the system intelligence than on themselves. The cache functions in IE usually are not fully understood and utilized by most users.