A Statistics-based Web Cache Prediction Model

Anthony Y. Fu
Department of Computer Science
City University of Hong Kong
anthony@cs.cityu.edu.hk

Xiao Wu
Department of Computer Science
City University of Hong Kong
wuxiao@cs.cityu.edu.hk

Haohuan Fu
Department of CE & IT
City University of Hong Kong
fu.haohuan@student.cityu.edu.hk

ABSTRACT
Web cache has played a significant role in improving performance of web-based system that it can help pre-fetch expected files in advance. In this paper, we propose a novel web caching prediction model, which is based on the statistical information of the sequences of files been accessed, and we call it Statistics-based Web Cache Prediction (SWCP) model. We discuss SWCP model in detail in this paper, and explore the reason why this model improve the response time of the web server. In the experiment sections, we demonstrate using SWCP model can achieve both better hit rate and byte hit rate than former web caching algorithms.

Keywords
Statistics-based Web Cache Prediction Model, Web Cache Pre-Fetching

1. INTRODUCTION
Web server response time is an important measurement for the network and server performance. In order to improve the server response time, web cache is used to buffer the copies of objects, i.e. files, requested by clients. A good web cache strategy can reduce the latency and eventually minimize the "World Wide Wait".

Two typical ways can be used to enhance the web caching performance. One is the replacement algorithms, which used to decide what objects in the web cache is useless and shall be washed out of. The other one is the web cache pre-fetching technology, which can be implemented in a simple way [3, 4, and 12] or a complicated way, i.e. data mining [1, 4, and 18]. There exists many replacement algorithms, such as LRU (Least Recently Used), LFU (Least Frequently Used), SIZE [24], LRU-MIN [25] and SLRU [23] etc. And there also exists some hybrid algorithms [2, 8] etc. Former researchers have done a lot of work on web cache modeling [7, 19], hierarchical design [7, 11 21], and framework for contents management [5, 6] to achieve better web cache performance. Unfortunately web cache using pure replacement algorithms do not have intelligent schemes [9].

In this paper, we discuss a novel Web Cache Prediction Model (SWCP) model to do Web Cache Prediction. In section 1, we introduce an ideal pseudo website and the related notations. Section 3 outlines our practical model to give readers structure of our SWCP model. In Section 4, we present the specification of our web cache prediction model. Section 5 discussed the practical model using SWCP. And section 6 concludes the paper.

6th ACM-HK PGDay, Mar 12, 2005

2. IDEAL PSEUDO WEBSITE AND NOTATIONS
People around the world access thousands of websites everyday, and a dedicated website will be accessed by so many people from different places of the world. It seems that an accessing activity seems to be a random event. But in fact, maybe there is some kind of hidden principles behind of it. E.g. when somebody wants do down load a movie and the movie has two segments (seg1 and seg2) linked by 2 URL in a webpage, he will download seg1 and seg2 respectively. So when he is downloading seg1, we can say that there will be a high probability that he will download seg2. If we are using web cache in the web-server, seg2 should be pre-fetched in advance to improve the web-server response time. Here we have an ideal web site which can help demonstrate the hidden principle in the web accessing time sequence.

Suppose we have a pseudo website http://www.SWCP.org with the following constrains.

C1. The total number of files is limited, and it is denoted as *FG*;

C2. At the site, a limited number of *task types* are requested and it is denoted as *TT*;

C3. In each *task*, a serial of files are accessed, and each file differs from the others. The file number of each *task* is constrained to be the same for easier analysis, and it is denoted as *TA*;

C4. For each task type, every file in this *task type* is different from all files of the other *task types*;

C5. Use prediction method to do file pre-fetching. "*Hit*" is for prediction success and "*Miss*" is for prediction failure;

C6. Prediction is based on the file sequence of the tasks.

For example, there is *task* named *task1*, and the accessed file sequence of *task1* is *A, B, C, D*. According to C3 and C4, if *task1* has been accessed before, when *A* is accessed, the prediction mechanism may predict *B, C, D* will be accessed in sequentially. Thus *B, C,* and *D* hit. But *A* is missed according to C3 and C4.

According to C3 and C4, in case the file number in each *task type* is large enough, the *hit rate* of web cache prediction is $(TA-TT)/TA \times 100\%$, as shown in Formula (1) as well as Figure1. We can see that *TA* should be larger and *TT* should be less to get larger *HR*.

This ideal model shows the possibility for us to design a web cache prediction model according to web cache history. And it will be especially useful for websites with large *TA/TT* ratio to do web cache prediction. In the following sections, a new web cache prediction model based on statistics of file reuse profile is discussed. In addition, the acronyms that will be used in the following sections are shown in Table.1 for convenience.

$$HR = \frac{TA \times \frac{FG}{TT} - TT \times \frac{FG}{TT} - TA}{TA \times \frac{FG}{TT}} \times 100\% = \frac{TA \times \left(\frac{FG}{TT} - 1\right) - TT \times \frac{FG}{TT}}{TA \times \frac{FG}{TT}} \times 100\% \quad (1)$$

$$\approx \frac{TA - TT}{TA} \times 100\% \quad , 1 << \frac{FG}{TT}$$

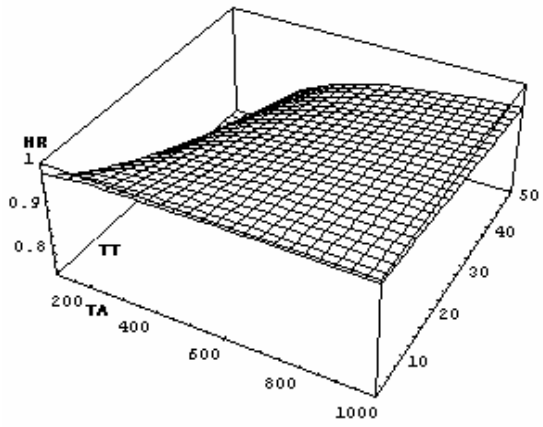

Figure. 1. The Hit-Rate(HR) trends with the growth of Task number(TA) and Task Type number(TT)

Table1. Acronyms used in SWCP Model

KUO	Key User Operation. It records the most important and helpful parameters to do web cache prediction
SDPC	Static Data Pre-Cache. A model to describe the prediction result
UOEM	User Operation Expression Model. An expression model to describe user's operations
SD	Static Data. Data without controlling information
DD	Dynamic Data. Data with controlling information
FG	File Group. It is an organization of SD objects
AOD	Access Orientation Degree. It records the measurement of the FG's orientation of being accessed

3. SWCP FRAMEWORD

Web data can be classified into 2 categories: Static Data (SD) and Dynamic Data (DD). Static Data do not have controlling information from the networks while the DD have, e.g. text, image, audio, video files and static web page file are SD. And ASP, JSP pages are DD.

Since SD is the major traffic in the network and is reused frequently, it is the objects we will use to predict for the different clients. Thus, it is critical that a pre-fetch approach should prevent the unnecessary SD from retransmission. In our approach, only SD could be predicted and pre-fetched into web cache during the web cache prediction process based on a statistic method. Figure2 shows the outline structure of SWCP approach.

There are several key components in SWCP to be introduced. Key User Operation (KUO) List records the information of operations that have been done by clients. This information includes time, user accessed objects, and other most important and helpful parameters to do web cache prediction. The KUO List is presented as a User Operation Expression Model (UOEM) List and is used to help people generate SD Pre-Cache (SDPC) List. The mapping from UOEM to SDPC is the main task, and it is shown in Figure3.

UOEM can get statistical information from the operations of client side. By applying some statistic methods on this kind of information, we can achieve the probable mapping rules from KUO List to SDPC List. After this process, the mapping rules are expected to be stable in a short period of time. But they still change with time.

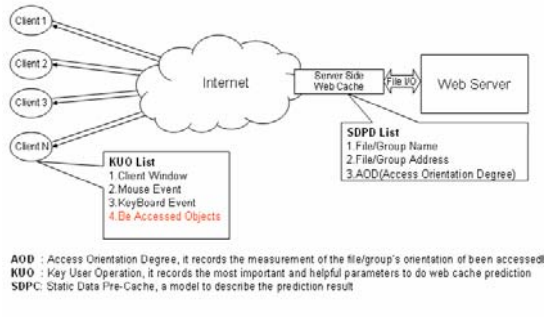

Figure. 2.Outline of Web Cache System

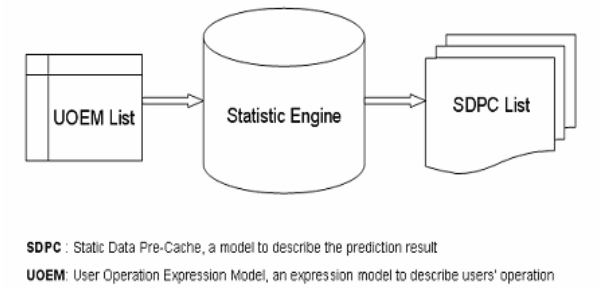

Figure. 3. UOEM List and SDPC List Mapping Process

Meanwhile, the Statistic Engine, which is core of SWCP, keeps processing with KUO, thus the updated mapping rules could be gotten. And SD can be pre-fetched from Web Server continuously.

4. SWCP COMPONENTS AND PREDICTION PROCESS

As described in the section 3, the SWCP includes KUO list which can be represented using User Operation Expression Model (UOEM). And the key operation is the mapping from UOEM list to SDPC list through Statistic Engine, i.e. in Figure3.

4.1 UOEM List (User Operation Expression Model List)

The basic process of Web Cache Prediction includes input, process, and output. In SWCP, the input is the UOEM List, which can be attained from both client side and server side. The contents

of UOEM list are very crucial for web cache prediction, because they are a kind of Key User Operation.

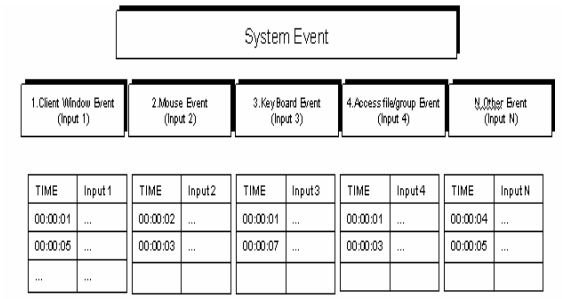

Figure. 4. System Event Type Demonstration

There are many kinds of items that can be involved into UOEM, including input-time, user behavior, and accessed files etc. System Event is used to describe these input-properties as shown in Figure4. For each event type, there is a timetable to record the input-properties at exact time.

Previous researches [18, 19] showed that these input-properties can facilitate web cache prediction. Different kinds of input-properties are used as input-history to do web cache prediction, yet SD history is the most commonly used one. In this paper, we also use SD history as basic reference.

For generalization, we use "object" instead of "file" later, as shown in Figure5.An object can be treated as a small part of a file, and can also be treated as containing several files. It is supposed that objects also have the properties described in the previous two paragraphs.

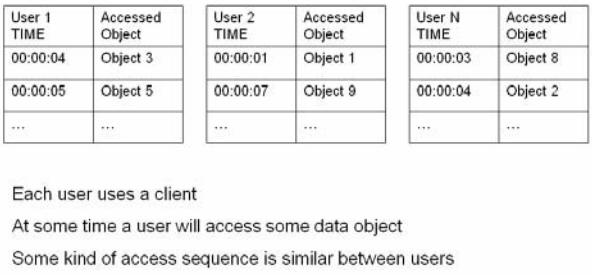

Figure. 5. User-accessing History on Objects

4.2 AODM (Access Orientation Degree Model)

For an object, Access Orientation Degree (AOD) is the only measurement to evaluate the likelihood of being accessed in the near future. If AOD becomes larger, the object's orientation of being accessed will become larger.

AODM is shown in Figure6. SD in Web Server can be divided into many groups. For each group, the AOD and Size for each file group are recorded. The AOD of each group is variable. The more access orientation, the bigger degree it will has. According to the AOD magnitude of each group, we can decide which group should be pre-fetched into the Web Cache. As we can not get an infinite large Web Cache, the size of each group has to be involved in for calculation facility.

File Group	AOD	Size (k)
FG1	……	……
FG2	……	……
FG3	……	……
FGX	……	……

AOD: Access Orientation Degree. The AOD bigger, the SD object to be accessed orientation larger.
FG : File Group, it is an organization of SD objects.
Size : The size of file group.

Figure. 6. Access Orientation Degree Model

4.3 Performance consideration- FG (File Grouping)

SD is composed of files. And these files have different sizes, ranging from Byte to Giga-Byte level. For performance consideration, they should be grouped. And the following is the reason to do file grouping and the detailed approach to do file grouping. The two FG mechanisms are shown in Figure7.

Suppose that there are 10,000 files with size of 1k, and each file is assigned as one group. The 'large' AODM List will have 10,000 records then performance problem will occur when process such a large AODM List. But if take 1,000 files as one group, we'll get an AODM List with only 10 groups. Considering the post-processing complexity, people can get a explicate comparison between the two mechanisms.

In case people get a very large file, for example, a movie file that sized 2G bytes, we cannot cache it into web cache for cache size limitation reason. Besides, it will take a long time to cache that large file. In such occasion, huge file should be sliced into pieces, e.g. we can slice it into 10M sized pieces. For convenience, grouping includes both grouping and slicing operation. File grouping process should be done in advance before the Web Cache Pre-Fetching process.

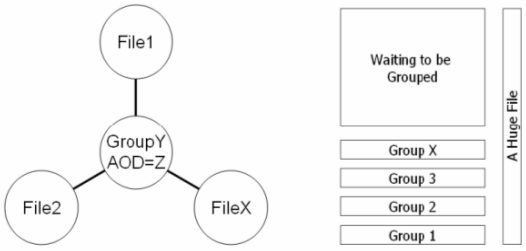

Figure. 7. File Grouping

4.4 Prediction Approach- UOEM to SDPC Mapping

The core part of SWCP is the statistical mapping from UOEM to SDPC. It contains two steps.

Step 1, UOEM to AODM mapping. SWCP process is evolved in predicting the most likely to-be-accessed FGs in the near future.

Step 2, AODM to SDPC mapping. It is a simple mapping from FGs to files.

Considering step 1, the known information is used to predict FGs. SWCP is used to do mapping from UOEM to AODM.

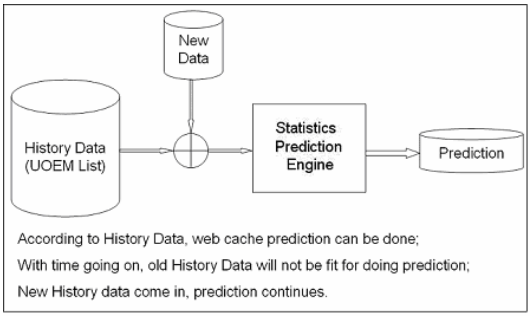

Figure. 8. SWCP Process

As shown in Figure8, we first get the UOEM List, i.e., the history data . Then, we could fetch the predicted result, AODM List by the predicting process which will be introduced in next section. . With time going on, users access new SD. The newly recorded UOEM items will become new data. As a result, the old history data and new data would be combined to make new history data. New prediction will be done according to the new history data.

4.5 Core approach - SWCP Algorithm for UOEM to AODM Mapping

In fact, it is difficult to give the perfect rule for the mapping process, because there are many significant things affect the accessed and to-be-accessed SD contents. For example, the working processes of the user, psychology matters, and internet speed will affect access intention, random access, and accessed targets are near to each other principle, etc.

But it is still possible to find some regular FG accessing characteristics. It is discussed that if somebody accesses FGs in one sequence, the sequences will happen again in the near future. On this basis, the most frequently repeated FG sequences in the history are the most possibly to be accessed again.

The formal description of SWCP process is defined here. As shown is Figure9, H stands for history data, e denotes for only 1 FG record, positioning in the tail of H; 2e denotes for only 2 FG records in the history, positioning in the tail of H; and so on…

H - e denotes for the history data except for the last FG records; *H - 2e* denotes for the history data except for the last two FG records; and so on…

E denotes for 1 arbitrary FG record; 2 *E* denotes for 2 arbitrary neighbored FG records; 3 *E* denotes for 3 arbitrary neighbored FG records; and so on…

The UOEM to AODM mapping can be explained as predicting AODM List from *H- xe* and *xe*.

For each e, find out all of the Es from *H - e*, where *E = e*. We will get $E_{1,1}$, $E_{1,2}$, $E_{1,3}$, etc…and then fill the repeated times of the same FG records respectively into AODM List to indicate access orientation degrees for each FG; For *2e*, find out all of the Es from *H - 2e*, where *2E = 2e*. We will get $E_{2,1}$, $E_{2,2}$, $E_{2,3}$, etc…and then fill the repeated times of the same FG records respectively into AODM List to indicate access orientation degrees for each FG; For *3e*, find out all of the Es from *H - 3e*, where *3E = 3e*. We will get $E_{3,1}$, $E_{3,2}$, $E_{3,3}$, etc…and then fill the repeated times of the same FG records respectively into AODM List to indicate access orientation degrees for each FG…

Do the above recurrent process, until a comfortable result is gotten. And choose FG with the highest AOD to be the predicted result target. Denote it as *P*. And pre-fetch it into the web cache.

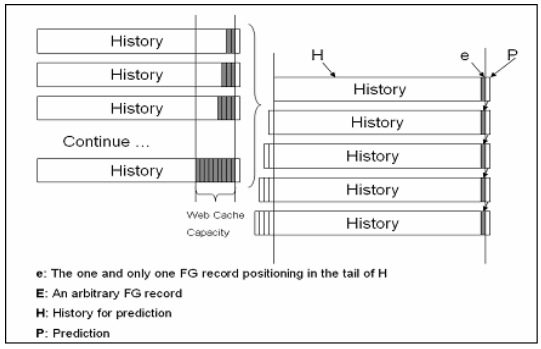

Figure. 9. SWCP Kernel

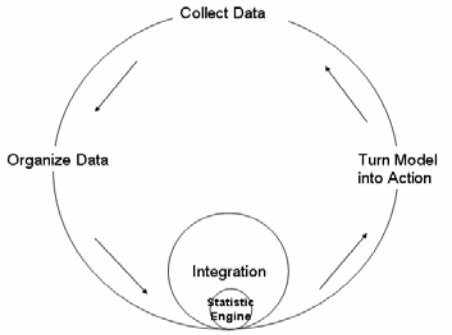

Figure. 10. Recurrent SWCP process

Finally, there is only one be predicted and be pre-fetched in. Continuous approaches will be based on previous process. The old *H* and generated *P* formulated the new *H*, and *P* becomes the new *e*. After the new process, we'll get the new P…

This process continues on, until meet the web cache capacity limitation. Finally, the expected contents will be pre-fetched in.

The algorithm above shows a closure controlling mechanism shown in Figure10. Suppose there is a long UOEM List. Through SWCP, web cache prediction was made. With time going on, new FG will be pre-fetched in the web cache. And then, add real accessed new FG to old history to formulate a new history to support new prediction.

5. PRACTICAL ASPECTS OF SWCP

We discuss the practical aspects for web cache pre-fetching in this section, which is applied on the basis of the previous SWCP Model.

5.1 Choose Effective UOEM - Web Requests

There are many kinds of methods to define UOEM. The most effective one is the accessed objects, i.e., Web Requests, which is also named as accessed files or file groups. Other UOEM parameters are not easy to collect. For example, if we want to choose user side operation on the computer, it won't be easy to get the user operation information. On the other hand, user operations on the computer do not reflect the user operation

exactly on the certain web site. When a user is viewing two web sites at the same time, he will access the two web sites alternatively. According to the user operations on the computer to predict the future is not quiet reasonable. But that's not to say, user operation can't become UOEM, someone has done some related works on this topic [19]. But it's not a quiet good choice. File group has been used in this model and approach.

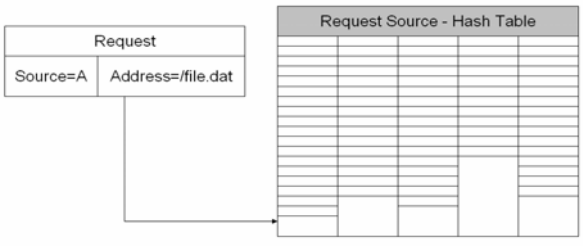

Figure. 11. Request Source Hash Table

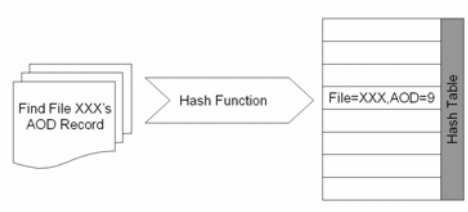

Figure. 12. AOD List Hash Table

When a request comes, according to the request source IP, it will be put into the history request pool. History request pool is a hash table, which can reduce the time complexity from O(n) to O(1), and each item of the hash table is a list. Each item in the list records the accessed files' directory and name. The reason of doing so is that, for the same web server there are always some kinds of similarities between the users to access it. The hash table is shown in Figure11.

For example, when a request comes, according to the request source IP, hash table will find that, it should be added to the first list. Then, the request is added to the end of list.

When requests come, the table will grow, when it reach a certain size, the table will be treated a history to do prediction. But the table will not grow unlimited. When size grows to a certain number, old history will be washed out of the table, because the near history will be more effective to predict the objects to be accessed in the near future. It is clear that, too old history will not be useful for prediction. Maybe the web server files have been changed already and the to-be-accessed orientation of are changed too. At the same time, the lists without 'many' records are washed out too, because and they will not make the hash table very big, on the other hand, they won't contribute much to the predication and sometime even affect the precision of prediction under the UOEM to AODM mapping mechanism.

5.2 Implement AODM - AOD List

As introduced before, AODM records the information of the orientation to access a file in the web server. In the practical model, AOD list is implemented as a hash table, which can reduce time complexity from O(n) to O(1), as shown in Figure12. When a file AOD should be increased, the file name will be searched in the hash table firstly. If the file could be found, the related AOD will be incremented. If the file cannot be found from the hash table, a new file record will be added to the hash table, and the related AOD will be set to 1.

Predicted file names will be drawn out from this table. Every time, the prediction process is completed, the prediction will be drawn from the AOD list according to the AOD number. For example, we want to draw out 20 prediction target, the files with AOD ranked top 20 will be drawn out.

5.3 File Grouping

File group mechanism includes two kinds of operations, grouping and slicing. The small sized file will be grouped, and the large sized file will be sliced. To reduce the program executive complexity we treat every file as a file group, while do not care whether it has small size or big size.

5.4 UOEM to AODM mapping - Prediction Generation

From the above work, we get the UOEM, request hash table, and the AODM, AOD list. We will show how to map the request hash table into the AOD list next. We choose the prediction to be one only. According to the last request of a client, the prediction will be done. E.g., the last request is from IP A, and the requested file is /file.dat. The request hash table will form a big list by link the lists one by one. It is Request History, as shown in Figure13.

When a record with accessed address /file.dat is found, the following record address will be added to AOD list from the head of the list. If the accessed address is in not in AOD list, a new AOD list record will be added, and the related AOD will be set to 1. If the accessed address is found in AOD list, the related AOD list record will increase AOD by 1. This process continues, till accesses the end of request history list.

To reduce the process complexity, the prediction depth is set to be '1'. Besides, experiment shows that it will not contribution much to the prediction although increasing the prediction depth will increase the hit rate and byte hit rate.

When the UOEM to AODM mapping process completed, the files with the largest orientation to be accessed in the near future will be found from AOD list.

5.5 AODM to SDPC mapping - Predicted file Generation

When one prediction process is completed, we need collect the prediction result from AOD list.

As the files with most orientation to be accessed in the near future are recorded in the AOD list with the largest AOD number, the AODM to SDPC mapping process is very simple. Just pre-fetch the files with the biggest AOD number into the cache. In practice, the cache is named object pool. Here, each object is a file in fact.

5.6 Web Cache - Object Pool

Object pool is used to store the cached files and is controlled by a hash table. It can improve the SWCP Model implementing performance. For insertion, find and deletion operations, the

average time complexity is O(1) when using hash table. And the complexity of the total algorithm complexity is O(n).

The web cache has a limited size. To demonstrate the SWCP, we only use FIFO for replacement algorithm. We have tried FIFO In the experiment and it can still get good experiment result, even using this simple replacement algorithm

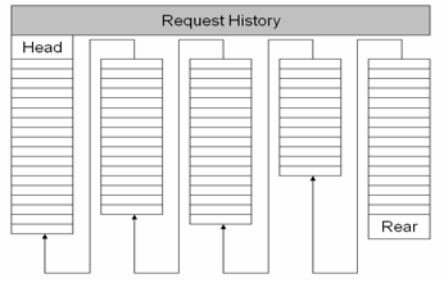

Figure. 13. Structure of Request History Lists

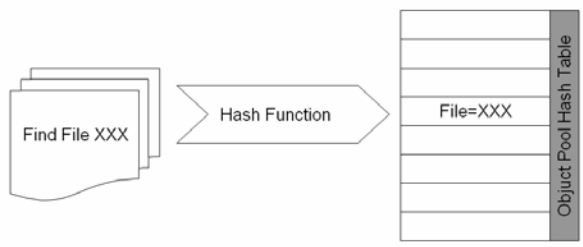

Figure. 14. Object Pool Hash Table

5.7 Add Replacement Algorithm to the Approach

In previous researches, there are a lot of contributions on replacement algorithm. In this paper, we focus on the prediction of to be accessed files in the near future, not on the replacement method. But SWCP Model could cooperate with replacement algorithms, E.g., LRU, LFU, LRU-MIN, SIZE, etc. In the experiment, LRU has been tried to do replacement in the object pool. The result is very exciting. Not LRU only, other replacement algorithms can also be used in the object pool for better performance.

6. SWCP ALGORITHM

We introduce the inputting data sources for SWCP firstly. As SWCP methods are based on different data sources and SWCP algorithm, and there are various kinds of data sources can be used, server side web log [4], including client side user-operation on devices, and user-called system event [20], etc., our SWCP method uses server side web log as SWCP source. There are web trace files in every web server, and they record requests information of client-side. Each record of a web trace contains client IP address, request URL, response type, and response time, etc. The reasons we use web log are:

(1) Obtaining Web log files is easier than client side user-operations log. Besides, it is very difficult to collect client side user-operation information or user-called system event information from all of client devices.

(2) Server side web log is more reasonable to be data SWCP because it is very simple and only includes most useful information of web site. Client side user-operations are more complicated and may include irrelevant information of other web sites.

There are 2 data structures defined in our SWCP algorithm. At the beginning of SWCP process, the web log records are loaded into a user-request pool p, which includes user record lists l_1, $l_2 \ldots l_n$. Users are identified by IP address. In each user record list l, it contains the user-accessed file *URL*s in a time ascending order.

As shown in Figure15, history request pool is a table of user-request lists ls, and each l is a list of accessed file *URL*s. For better performance, user-request pool is designed to be a hash table with user IP as primary-key. When a request comes, according to the request source IP address, new record will be added to the user-request pool p.

User-req pool p	User request list l_1	Accessed file URLs of l_1
	User request list l_2	Accessed file URLs of l_2
	…	…
	User request list l_n	Accessed file URLs of l_n

Figure. 15. User Request Pool

There are numbers of accessed file *URL*s in each user-request list. Let T_{min} be the minimum threshold of *URL* number. In SWCP process, the user-request lists with more than T_{min} *URL*s will be marked. Since minor *URL*s do not reflect user-access profile, we believe only these marked user-request lists will contribute to the SWCP process. On the other hand, the *URL* number in each l should not grow unboundedly. When it grows to an upper threshold T_{max}, old *URL*s can be treated as outdated and will be washed out (deleted) from the list. It is obvious that, very old data source will not be useful for new SWCP because both the user-accessing profile and web server files may have been changed already.

A new data structure is used to record user access orientation for each file *URL*. As shown in Figure16, access-orientation is an integer to measure user accessing orientation for a file. It is also a hash table with file *URL* as primary-key.

URL_1	URL_2	…	URL_k
access-orientation$_1$	*access-orientation*$_2$	…	*access-orientation*$_k$

Figure. 16. Access orientation hash table

At the beginning in the SWCP process, the value of *access-orientation*s are set to zero. While the SWCP processing carrying on, they will be increased respectively. Our SWCP process is not complicated. Suppose the last request from the users is URL_{last}. For each marked user-request list l, find out all of the *URL*s that equal to URL_{last}. If there are N_1 URL_1s behind URL_{last}, add N_1 to *access-orientation*$_1$; if there are N_2 URL_2s behind URL_{last}, add N_2 to *access-orientation*$_2$…until add N_k to *access-orientation*$_k$. Finally, the web cache prediction can be done according to the calculation of *access-orientation*s. Our SWCP procedure and web cache pre-fetching procedure can be presented into pseudo code, as shown in Table.2.

In line 20, M can be assigned according to web cache size and whole system performance. Larger the web cache size is and better the system performance is, larger value is assigned to M. As for URL_{p1}, URL_{p2}…URL_{pM} in line 22, there are two possible results. Either web cache will be full and cannot cache all of them, or web cache is large enough to hold all them. In the first case, the *URL*s that can not be cached will be discarded and web cache prediction process continues whereas in the second case, web cache prediction process will continue directly. This alternative method can only be used for the situation of small web cache size. In case of the large size of web cache, the system performance is critical. Our experiments in the following sections assume that our system has a high performance thus we apply the second situation in our design. In addition for better web cache hit-rate and byte hit-rate, replacement algorithm can be used to cooperate with SWCP method. In previous researches, there are a lot of contributions on replacement algorithm, e.g., FIFO, LRU, LFU, LRU-MIN, and SIZE etc. Web caching hit-rate and byte hit-rate can be better under the cooperation of web cache prediction method and web cache replacement algorithm. In our experiment, FIFO and LRU are used for the web cache replacement mechanism.

Table2. Algorithm Pseudo Code

```
1.  Load web log into user-request pool p;
2.  Mark the user-request lists ls containing more than
    Tmin records
3.  FOR each l having more than Tmax records
4.    Wash out outdated records;
5.  END FOR
6.  FOR each latest URL request
7.    FOR each URLj in URL1...URLk
8.      Set access-orientationj to 0;
9.    END FOR
10.   Get the latest requested URL URLlast;
11.   FOR each li in l1… ln
12.     Find all URL==URLlast;
13.     FOR each URLj in URL1…URLk
14.       Set integer Nj=0;
15.       Count the number (Nj) of URLjs that follow
    URLlast;
16.       access-orientationj= access-orientationj+ Nj ;
17.       END FOR
18.     END FOR
19.   END FOR
20.   Find out M URLs having the largest access-
    orientations from
21.     access orientation hash table, denoted as URLp1,
    URLp2…URLpM;
22.   FOR each URLj in URLp1, URLp2…URLpM
23.     IF URLj NOT in web cache
24.       copy URLj into web cache;
25.     END IF
26.   END FOR
27. END FOR
```

7. EXPERIMENTS

The trace-driven experiments have been done to compare the performance of our SWCP approach with some well-known cache replacement algorithms. We choose three basic replacement algorithms, LFU, LRU and SIZE, which respectively consider three most basic factors, reference number, last access time and data size. Besides, we also choose two more complex algorithms, the LRU-MIN and SLRU, which consider both the time and size. For the trace, we use the web trace of the website of the Department of Computer Science of City University of Hong Kong. For the convenience of the experiment, we truncated 1,000,000 records of request from the trace file of April 2003. In order to get more precise result from experiments, modifications have been done to the web trace traces. The records of request for items that are unable to be cached in are filtered out from the trace. We first did a general analysis about the trace data, to find out the largest hit-rate we could get, and the cache size needed for the largest hit-rate. Then, try the trace-driven experiment for every algorithm, including the SWCP approach, with 10 different cache sizes ranging from 200k to 100M bytes. We use the hit-rate and byte hit-rate as the criteria for performance evaluations. As we cannot get the information about response delay, we calculate the fetching traffic as the cost. Finally, for experiment feasibility, the experiment concentrate on cache pre-fetching and replacement problems only, and the consistency issues are left for further research on this topic.

Table 3. Trace data analysis and infinite cache experiment

Request number(request)	1,000,000
Total request size(byte)	19,452,658,735
Different client IP number(IP)	10,102
Different URL number(URL)	154,842
HR with infinite cache size	84.52%
BHR with infinite cache size	79.21%
Web cache size used(byte)	4,043,863,434

We performed a thorough analysis on the trace data first, in order to get the general information about the trace data, as shown in Table.3. We also did a trace-driven experiment with the infinite cache size to get the maximum possible hit rate and byte hit-rate.

From the analysis, we could know that in the 1,000,000 records, there are different requests from more than 10,000 different clients (regard one IP as one client), which request for more than 150,000 different URL paths. It is considered that this trace is large enough to exhibit the complex web request pattern. An analysis about the distribution of the requests to the size has been done, as shown in the Figure17 and Figure18.

In Figure. 17, we could see that there is a very clear bias to the small size data items in the request pattern, just like the pattern of most web applications. The number of requests for the data items with a size less than 1k is more than 30% of the total. Almost all the requests are for the data items less than 128k.

However, in most situations, the request number should not be considered the most significant. Hit or miss of a 1M bytes file has larger effect on the network performance than the hit or miss of a 1k bytes file. So we also need to consider the size. Figure18 shows the distribution of the request size. We could see that there is no such a clear bias to the small files in this distribution.

Although the small files occupy a very large part in the distribution of request number, they occupy a small part in the distribution of request size. In the distribution of size, the files ranging from 8k to 64M bytes make up the main part.

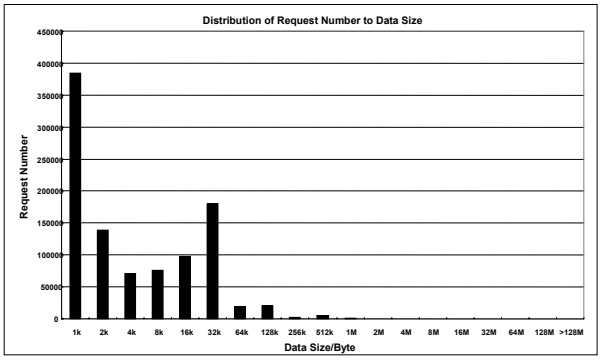

Figure. 17. Distribution of Request Number to Data Size

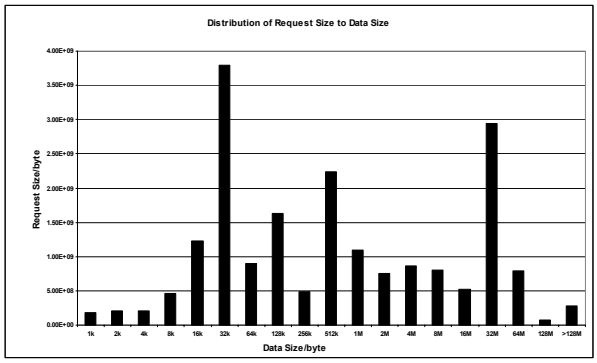

Figure. 18. Distribution of request size to data size

We have done the trace-driven experiment for our SWCP approach with 10 different cache sizes, compared with other five different cache replacement algorithms. The results are shown in Figure19, Figure20 and Figure21.

Figure19. shows the different hit rates for different algorithms. And among the different algorithms, the DM 600 means the SWCP approach uses the latest 600 request-history records to do prediction, and DM 1200 means the approach uses 1200 the latest request-history records. In the web cache, LRU is used as wash-out approach. We could see that, from pure LRU to LRU plus web cache prediction, a lot achievement has been gotten. And from this figure we could see that the SWCP approach has much better performance over any other pure replacement algorithms.

In the experiment, it is found that when the cache size is small, the advantage of the SWCP approach is even obvious. The performance of the other replacement algorithms has a clear dependence on the cache size. With the cache size decreased from the 100M to 200k, the hit ratios of the replacement algorithms fall down from 70% to only 20%. But for the SWCP approach, when the cache size is 100M, the hit rate still can achieve more than 80%, very close to the maximum cache hit rate. When the cache size falls down to only 200k, the hit rate is still more than 55%, The hit rate using our approach is much higher than using the replacement algorithms(20% at the most) with the same cache size.

Figure20. shows the comparison of byte-hit rate among different algorithms. We think that this comparison result is more important for judging the real performance of the algorithms in the practical network application. One of the main differences between web cache and traditional memory cache is that, data items should be deal with in different sizes in web cache. There is no doubt that a hit or miss of one 10M file has much more effect on the network traffic and latency than a hit or miss of one 10k file. So we think that byte hit-rate is a more reasonable criterion than hit rate. From Figure.20, we could see that, for the byte hit rate, the SWCP approach also has a much better performance than the other replacement algorithms.

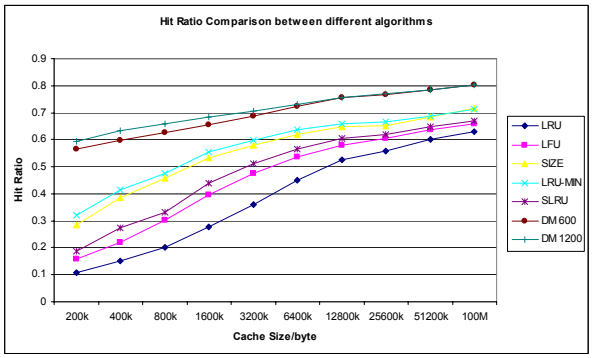

Figure. 19. Hit-rate comparison

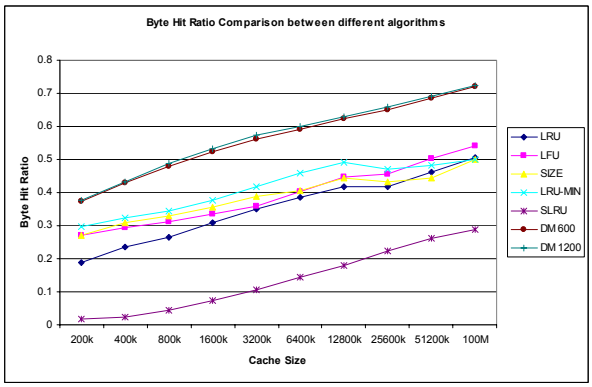

Figure. 20. . Byte hit-rate comparison

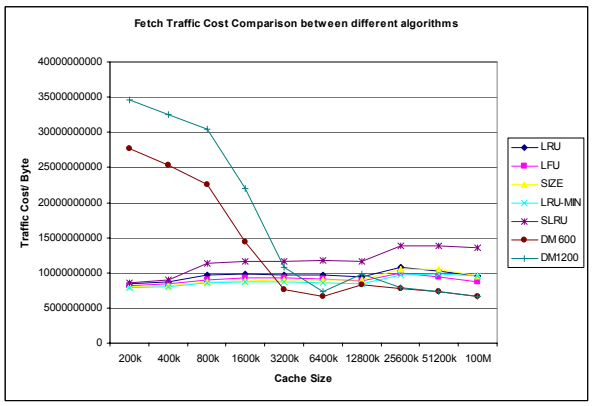

Figure. 21. Network traffic comparison

For the SWCP approach, one problem is the traffic cost caused by the pre-fetching. The items pre-fetched into the cache are regarded to have a large probability to be accessed in the future, but there is also probability that the items pre-fetched will not be accessed. In this way, the SWCP approach might cause much more useless traffic than the replacement algorithms. Figure21. shows the statistics result about this problem. When the cache size is very small, the SWCP approach does cause much more fetch traffic than the other algorithms. However, when the cache size grows larger, the data traffic of the SWCP method would become similar and even smaller than the replacement algorithms. Therefore, we believe that in most practical situations, the SWCP will not cause much wasted fetch traffic than the replacement algorithm.

8. CONCLUSIONS

There is a response time necessity in web accessing. In order to reduce response time, web cache is used, and web cache prediction for pre-fetching is a much creative method for web caching. In this paper, we theoretically analyzed a simple model website www.SWCP.org, and introduced how SWCP model does web cache prediction. In the last section, we formally defined the web cache prediction procedures for implementation and future mathematical analysis. Under the practical approach, we did experiment, and achieved satisfactory experiment results demonstrated the efficiency of SWCP approach.

9. REFERENCES

[1] Fu, Y.; Fu, H.; Au, P.; *An Integration Approach of Data Mining with Web Cache Pre-Fetching,* Lecture Notes in Computer Science on Second International Symposium on Parallel and Distributed Processing and Applications, December 2004.

[2] Xiao, L.; Zhang, X.; Andrzejak, A.; Chen, S.; *Building a large and efficient hybrid peer-to-peer Internet caching system*, IEEE Transactions on Knowledge and Data Engineering, Volume: 16, Issue: 6 , Pages:754 - 769, June 2004.

[3] Xu, J.; Liu, J.; Li, B.; Jia, X.; *Caching and prefetching for Web content distribution*, Computing in Science & Engineering , Volume: 06 , Issue: 4, Pages:54 - 59, July-Aug. 2004

[4] Yang, Q.; Zhang, H.H.; *Web-log mining for predictive Web caching*, IEEE Transactions on Knowledge and Data Engineering,, Volume: 15 , Issue: 4, Pages:1050 - 1053, July-Aug. 2003.

[5] Mohapatra, P.; Chen, H.; *WebGraph: A framework for managing and improving performance of dynamic Web content*, IEEE Journal on Selected Areas in Communications,, Volume: 20 , Issue: 7, Pages:1414 - 1425, Sep 2002

[6] Nikolaidis, I.; *Web Caching and Content Delivery*, IEEE Network, Volume: 16, Issue: 4, Pages: 5 - 5, July-Aug. 2002.

[7] Che, H.; Tung, Y.; Wang, Z.; *Hierarchical Web caching systems: modeling, design and experimental results,* IEEE Journal on Selected Areas in Communications, Volume: 20, Issue: 7, Pages: 1305 - 1314, Sep 2002.

[8] Psounis, K.; Prabhakar, B.; *Efficient randomized Web-cache replacement schemes using samples from past eviction times*, IEEE/ACM Transactions on Networking, Volume: 10 , Issue: 4, Pages:441 - 454, Aug. 2002.

[9] Tang, X.; Chanson, S.T.; *Coordinated en-route Web caching*, IEEE Transactions on Computers, Volume: 51 , Issue: 6, Pages:595 - 607, June 2002.

[10] Serpanos, D.N.; Karakostas, G.; *Proof for effective and efficient Web caching*, Electronics Letters, Volume: 38, Issue: 10, Pages: 490 - 492, 9 May 2002.

[11] Wong, K. Y.; Yeung, K. H.; *Site-based approach to Web cache design*, IEEE Internet Computing, Volume: 5, Issue:5, Pages:28 - 34, Sept.-Oct. 2001.

[12] Chou, W.; *Building an infrastructure for a powerful web presence*, IT Professional, Volume: 3, Issue: 6, Pages: 54 - 60, Nov.-Dec. 2001.

[13] Kenyon, C.; *The evolution of Web-caching markets*, Computer, Volume: 34, Issue: 11, Pages: 128 - 130, Nov. 2001.

[14] Menaud, J. M.; Issarny, V.; Banatre, M.; *A scalable and efficient cooperative system for Web caches*, IEEE Concurrency, Volume: 8 , Issue: 3, Pages:56 - 62, July-Sept. 2000.

[15] *Web Trace by April 2003*, Computer Department, City University of Hong Kong, April 2003.

[16] *Web Trace of uc*, sanitized-access.20031118.gz, http://www.ircache.net , 18th, November 2003.

[17] Busari, M.; Williamson, C.; *Simulation Evaluation of a Heterogeneous Web Proxy Caching Hierarchy*, Proceedings of MASCOTS, Cincinnati, OH, page 379-388, August 2001.

[18] Bonchi, F.; Giannotti, F.; Manco, G.; Renso, C.; Nanni, M.; Pedreschi, D.; Ruggieri S.; *Data mining for intelligent Web caching*, International Conference on Information Technology: Coding and Computing Proceedings, page 599 - 603, April 2001.

[19] Swaminathan, N.; Raghavan, S.V.; *Intelligent Pre-fetch in WWW Using Client Behavior Characterization*, 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, August - September, 2000.

[20] Rizzo, L.; Vicisano, L.; *Replacement Policies for a Proxy Cache*, ACM Transaction on Networking, Volume: 8, No. 2, April. 2000.

[21] Tewari, R; Dahlin, M; Vin, H.; Kay, J.; *Beyond Hierarchies: Design Considerations for the Distributed Caching on the Internet*, Proceedings of the 19th International Conference on Distributed Computing Systems, Austin, TX, June 1999.

[22] Mahanti, A.; Williamson, C.; *Web Proxy Work-load Characterization*, Technical Report, Department of Computer Science, University of Saskatchewan, February 1999.

[23] Aggarwal, C.; Wolf, L. Joel; Yu, P. S.; *Caching on the World Wide Web*, IEEE Transactions on Knowledge and Data Engineering, Volume: 11, No. 1, January - February 1999.

[24] Williams, S.; Abrams, M.; Standridge, C. R.; Abdulla, G.; Fox, E. A.; *Removal Policies in Network Caches for World-Wide Web Documents*, Proceedings of ACM SIGCOMM, page 293-305, 1996.

[25] Abrams, M.; Standridge, C. R.; Abdulla, G.; Williams, S. ; Fox, E. A..; *Caching proxies Limitations and potentials*, 4th International World-wide Web Conference, page 119-133 , Dec, 1995.