Cache Pre-fetching system based on data mining on Web

Yongbiao Li*

Human resource management Department, Jinan University, Huangpu Street 601# Tianhe Distric, Guangzhou City,Guangdong Province, 510632, P.R.C

**Corresponding author's e-mail: gql.jnu@gmail.com*

Received 1 March 2014, www.cmnt.lv

Abstract

From 20th century 90's to now on, Internet and data mining techniques had developed rapidly and became mature, kinds of application on Web data mining had been proposed to the market. In this paper, we would first introduce the development of cache Pre-fetching technique, and then present a cache pre-fetching System model based on Web data mining, details of each implementation would follow. Our aim was to enhance caching effectiveness, and network accessing speed. Such technique could be applied in personnel, educational, and official information managing system in database of educational scope. Accessing speed of educational information system for numerous teachers and students, benefit high school personnel management, and also the effective scientific structuralize educational management.

Keywords: Web data mining, sequential mining, cache pre-fetching system

1 Introduction

While WWW had become the most popular service in Internet, Web work load grew explosively, the problem came along was the simultaneous growth of network ping. How to raise the responding speed had become the most urgent problem. Caching technique was an effective solution and widely used in Web clients and servers, but most of them implemented using traditional memory paging strategy, for example LRU (Least Recently Used). Thomas MK et al. [1] stated that: As the ratio of dynamic content and personal service on WWW kept rising, performance improvement brought by caching was not obvious. Meanwhile, Chen Xin et al. [2] stated that: Hit rate of web cache would usually float from 24% to 45%. Pre-fetching technique was a further extension of cache technique, enhancement of Web system could implemented in different ways:

Pre-fetching mechanism could further enhance hit rate into around 60% to 80%, reducing network accessing latency to improve quality of service (QoS). Compared to cache technique, pre-fetching mechanism was pertinence; hence the memory space usage was relatively small, which was able to fulfil personalization need, showing users' personal interest. Pre-fetching mechanism could also smooth work load of network, the usage of limited network resource would be more effective. Therefore, the study of Web pre-fetching technique was important to reduce web access latency and raise QoS.

In this paper, we would combine pattern features of Web proxy server service flow and cache mechanism; then studied about the design of Web pre-fetching strategy. By studying the relation between cache size, replacement algorithms and cache hit rate, explored a kind of Web proxy server side Web cache replacement algorithms with high hit rate. With the use of Web log and data mining skills, linked up the internal attributes of Web page with cache size and hit rate, in order to optimize the usage of proxy server.

2 Related Work

2.1 WEB CACHE TECHNIQUE

Cache technique was well developed in many fields, like operating system, distributed file system, but Web cache was different from those traditional cache systems.

Firstly, "file" would be the unit for Web cache system operations like save, edit, and replace, and its size for request, save, and transmission operations on the Internet various. Therefore, replacement strategy had to consider not just frequency and recency, but file size also.

Secondly, the cost for traditional cache to maintain different cache object was basically equal; but in Web cache, the cost of Web cache was related to the cost of getting Web cache object, and path and server for data transmission, thus the time for downloading different object various.

Moreover, there was usually a small number of accessing program in traditional cache; but for Web cache, other than client side, there might be a great number of client connections, and this kind of connections were usually come from decades to thousands clients. Besides, cache consistency need maintenance in Web cache.

Many countries were using Web cache, for example, the JANET from Britain, DFN from Germany, FREEnet of Russia, SingNet of Singapore, ThaiSARN from Thailand, and etc. All of them were national Web cache system, providing high speed cache service with low price. CERNET from China introduced a level-structural Web cache project: built a L1 cache system in nation center, then built L2 cache system in each connect school network, forming a cache hierarchy through cache interacting protocol in CERNET scope.

There were large number of Web cache replacement algorithms, GDSF (Greedy Dual Size Frequency), GDSize (Greedy Dual Size), LFU (Least Frequently Used), LRU (Least Recently Used) were those representatives.

Performance of cache replacement strategy relied on the practical properties of Web access; there was none of recent strategy performed well under different access condition. Ways to make the replacement strategy adaptive to different Web access properties had been a great concern.

2.2 WEB PRE-FETCHING TECHNIQUE

Pre-fetching was aimed to hide communication latency, and could be classified into the following models:

(1) Pre-fetching algorithm based on access probability

Numbers of literature implemented Web pre-fetching based on the pre-fetching algorithm of Markov process. Traditional Markov chain model was a simple and effective prediction model, but the prediction accuracy was relatively low. XING Yong-Kang et al. [3] stated and built a multi-Markov chain user browsing prediction model based on classification of user. The works [4, 5] used hidden Markov process to raise prediction accuracy. Su Zhong et al. [6] used N-gram prediction model to predict the Web access request might be occurred in the future.

(2) Pre-fetching algorithm based on data mining

According to historical and recent access data, with the use of data mining technique, predicted possible future behavior of user, in order to prefetch the related Web page. Data in user's data buffer could be used as historical data for data mining. The works [7, 8] mined interest relation rules using data mining, applied those rules as pre-fetching foundation to predict pages. Pre-fetching based on data mining was more suitable for user personalize recommendation.

(3) Pre-fetching algorithm based on Web semantics

Zhu et al. [9] proposed to extract features of user session, then classified semantically. While responding user's request, server would calculate user accessing path, and the distances between user and each category center, in order to confirm the type of session. According common features of session category, predicted access-possible documents, pre-transmit them to client side. T.I.brahim et al. [10] introduced semantic web page pre-fetching with the use of neural network. By extracting hyperlinks in web page, using keywords described in hyperlink text as input of neural network; output of neural network would be the basis for pre-fetching. Browsing path of user would be the training sample for the learning of neural network

(4) Pre-fetching algorithm based on network performance

JIN et al. [11] studied Web intelligent boost technique based on RTT (round trip time) and other network performance index. Proposed an intelligent pre-fetching control technique and new cache replacement method based on the service analysis on web proxy server and measurement of network RTT. R.P .Klemn et al. [12] designed and implemented pre-fetching agent WebCompanion in Java. The pre-fetching algorithm was based on an estimated RTT, only applied pre-fetching for those Web object with relative long respond time and low resource usage.

(5) Pre-fetching algorithm based on popularity

E.P. Mareatos et al. [13] proposed a classical Top-10 method, basic concept was to find out the Top-10 popular document (named it asTop-10) on server periodically; when clients sent request, server would send the Top-10 to them. X.Chen et al. [14] implemented Web pre-fetching using the popularity-based PPM model. Declared four levels for popularity of URLs' access mode, then constructed a prediction tree using these models for pre-fetching. Through post-processing, reconstruction and other methods, reduced space usage of PPM algorithm, and raised the accuracy of prediction.The recent study of cache and pre-fetching development was to discover a unified cache-and-pre-fetching model according to Web object browsing features, raise adaptability of cache strategy and pre-fetching algorithms, achieved a better performance in a reasonable time and space usage.

(6) Pre-fetching system

Griffioen et al. [15] had studied the pre-fetching and cache model of file system, assuming that cache and pre-fetching shared the same cache space. Result had shown that cache pre-fetching-unifying model could improve performance of cache system. Z.Jiang et al. [16] cooperated the use of server sides with client sides to implement pre-fetching, studied pre-fetching mechanism based on network work load and waiting time of users. The paper studied the pre-fetching control problem also, but not yet considered the competition of cache space between pre-fetching mechanism and cache mechanism. Pei cao et al [17] studied the combination of cache and pre-fetching in file system. Proposed and analyzed two combination strategy: proactive strategy and conservative strategy; through simulation test, these two method could reduce application latency for more than 50%.

N.J.Tuah et al. [18] combined pre-fetching and cache to raise the utilization rate of memory. Deduct the calculation formula of pre-fetching threshold under two interacting models of cache and pre-fetching. Then it made a conclution that limitation for the number of pre-fetching object was no longer needed once the access probability met the close value. In the two used models, improvement of access time had been considered, while the whole system resource usage had not.

Yang et al. [19] retained a fixed cache space for pre-fetching object in unified system of cache and pre-fetching while pre-fetching Web object. pre-fetching model in the system was constructed from the mined visit path from log file. Real data declared that cache performance in cache pre-fetching unified system was better than cache-only system.

3 Our Pre-fetching System

While update frequency of network resources kept raising, performance improvement by cache was no longer obvious. Numbers of studies had shown that pre-fetching would gain great benefit only when cooperating with suitable cache algorithms.

3.1 PRE-FETCHING SYSTEM MODEL

As shown in the figure 1, pre-fetching system model concludes several components: log file processing, relation (sequence) pattern mining, relation rules, buffer management based on prediction, cache buffer, pre-fech queue, log file.

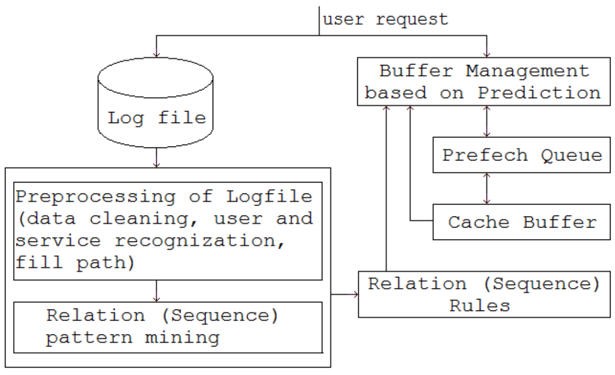

user request
Log file
Buffer Management based on Prediction
Prefech Queue
Cache Buffer
Preprocessing of Logfile (data cleaning, user and service recognization, fill path)
Relation (Sequence) pattern mining
Relation (Sequence) Rules

FIGURE 1 Pattern Mining Model in Web

4 Log File Processing

Quality of data preprocessing was closely related to the efficiency and result of modeling mining. Data preprocessing included data cleaning, path filling, and users, session, and services recognition.

(1) Data cleaning

The main task for data cleaning was to remove redundant data that was not related to pattern mining in data source. At server side, the page requested by user presented as numbers of .html (or .asp/.jsp) files, numbers of image file or script files. What we were going to study was the relation between pages and pages, which was independent from images or scripts in them. And log file would record all transmission of files, hence, files could be removed in order to reduce total size of data.

(2) Path filling

Part of the user accessing information in Web log might be incomplete. For instance, while there was no direct chain relation between the recent accessing page and previous requested page, and the requesting page was a recently requested page, so we could assume that the user was using the back button of browser; which invoking the local cached page, this should not be recorded in server side log. Heuristic rules were be used for the situation, filling the user path with inference of network topology structure.

(3) Users recognition

That was hard to identify a user by only user access log. With respect to practical experimental environment, network environment under CERNET was the only consideration, using fixed IP as a sign to specify user; and assumed that a user was corresponding to one and only one IP, thus recognized different users.

(4) Session recognition

Session referred to consecutively requested pages by a user, pages accessed by different users belonged to different sessions. Session recognition was to separate user accessing record into plural independent session records.

(5) Service recognition

Service recognition was to transfer user session into smaller, more accurate, and relatively semantic user accessing service, which was the page sequential for user to visit specific information. While service recognition, filtration of part of the service should be applied; for example, some users just visited the web page but was not interested about web content, services liked this could be filtered, avoided too much relations generated.

5 Relation (Sequence) Pattern Mining

After the filtration of log file, we might start using the ASM sequential mining algorithm to generate rule table, then decided cache-prefetching strategy based on those rules generated. Figure 2 is the ASM sequential pattern mining algorithms:

```
ASM sequential pattern mining
/* Log: filtered log records,
   min_sup: minimum support threshold,
   min_con: minimum confidence threshold */
{
  L1=Find_frequent_1-sequence(Log);
  C2=Asm_gen(L1);
  For each candidate c in C2,
    c.support=COUNT(c);
    L2={c in C2 | c.support ≥ min_sup};
    For each sequence l:p→q in L2,
      If(l.support/p.support ≥ min_con)Then
      Add l to Rule_table;
  Return Rule_table;
}
Procedure Asm_gen(L1)
{
  For each sequence l1 in L1,
    For each sequence l2 in L2.
      If((l1.IP = l2.IP)and(l1.T=l2.T)and
         (l1.P=l2.P-1)){
        c = l1.sequence→l2.sequence||
                      l1.IP||l1.T||l2.P;
        Add c to C2;
      }
}
```

FIGURE 2 ASM sequential pattern mining

We would use a practical example to explain the algorithm. After data cleaning at the first stage, and then index each accessed pages of user, we could obtain an initial log as table 1 shown.

First of all, scanned through these logs, calculated browsing count (support threshold, 7th, 8th lines in the algorithm) of each page, and then recorded them by a 3-tuple group (IP, T, P). As shown in table 2, (202.116.32.46, 1, 3) of sequence 3 meant that user with IP 202.116.32.46 accessed the page with sequence number of 3 at the 3rd position in 1st service.

After table 2 was obtained, filtered those sequence with support larger than the minimum support (here assumed the minimum support be 2), obtained sequence 1, 2, 3, 5, 7, 8, 9, 10. After that, compared the 3-tuple group of each sequence pair (19th to 21st lines in the algorithm); if the pair met requirement, then put it into candidate set with sequence length of 2, as shown in table 3.

The ASM pattern mining algorithm would stop after the generation of candidate set with sequence length of 2 finished. After that, calculated the global confidence for each rule p→q, divided by confidence of p for each global confidence equal to p→q (11th line in algorithm). Assumed the minimum confidence be 1/2, removed the rule of global confidence (as the rule had been fulfilled), obtaining rule table 4 at last.

Notably, here we used local confidence for each user IP; for the same rule p→q, sum of each local confidence equal to the global confidence.

TABLE 1 Filtrated Log

IP	Visited Page Sequence
202.116.32.46	(1,2,3)(4,5)
202.116.32.47	(6,5)(1,2,3)
202.116.32.48	(7,8)(9,10)
202.116.32.49	(1,2)(9,10)
202.116.32.50	(5,1,3)
202.116.32.51	(7,8)(11,9,10)

TABLE 2 Candidate Set with Sequence Length 1

Sequence number	Support	(IP,T,P)
1	4	(202.116.32.46,1,1)
		(202.116.32.47,2,1)
		(202.116.32.49,1,1)
		(202.116.32.50,1,2)
2	3	(202.116.32.46,1,2)
		(202.116.32.47,2,2)
		(202.116.32.49,1,2)
3	3	(202.116.32.46,1,3)
		(202.116.32.47,2,3)
		(202.116.32.50,1,3)
4	1	(202.116.32.46,2,1)
5	3	(202.116.32.46,2,2)
		(202.116.32.47,1,2)
		(202.116.32.50,1,1)
6	1	(202.116.32.47,1,1)
7	2	(202.116.32.48,1,1)
		(202.116.32.51,1,1)
8	2	(202.116.32.48,1,2)
		(202.116.32.51,1,2)
9	3	(202.116.32.48,2,1)
		(202.116.32.49,2,1)
		(202.116.32.51,2,2)
10	3	(202.116.32.48,2,2)
		(202.116.32.49,2,2)
		(202.116.32.51,2,3)
11	1	(202.116.32.51,2,1)

TABLE 3 Candidate Set with Sequence Length 2

Sequence	Support	(IP,T,P)
1→2	3	(202.116.32.46,1,2) (202.116.32.47,2,2) (202.116.32.49,1,2)
1→3	1	(202.116.32.50,1,3)
2→3	2	(202.116.32.46,1,3) (202.116.32.47,2,3)
4→5	1	(202.116.32.46,2,2)
5→1	1	(202.116.32.50,1,2)
6→5	1	(202.116.32.47,1,2)
7→8	2	(202.116.32.48,1,2) (202.116.32.51,1,2)
9→10	3	(202.116.32.48,2,2) (202.116.32.49,2,2) (202.116.32.51,2,3)
11→9	1	(202.116.32.51,2,2)

TABLE 4 Rule Table

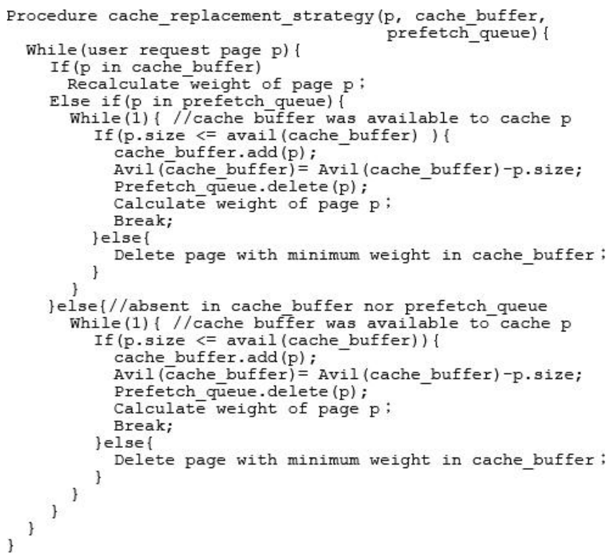
IP	Access rule	Local confidence	Global confidence
202.116.32.46	1→2	1/4	3/4
202.116.32.47	1→2	1/4	
202.116.32.49	1→2	1/4	
202.116.32.46	2→3	1/3	2/3
202.116.32.47	2→3	1/3	
202.116.32.48	7→8	1/2	1
202.116.32.51	7→8	1/2	
202.116.32.48	9→10	1/3	1
202.116.32.49	9→10	1/3	
202.116.32.51	9→10	1/3	

5.1 BUFFER MANAGEMENT BASED ON PREDICTION

There were two main processes for buffer management based on prediction:

1. Handled requested page in cache buffer from recent user
2. Handled pages which were going to be visited in pre-fetching Queue

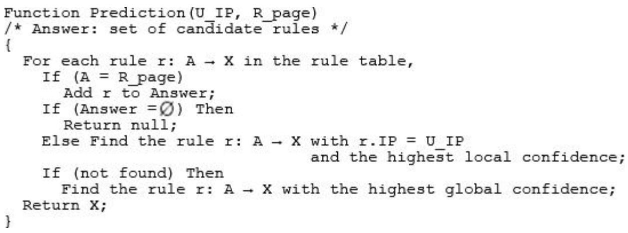
Handling requested page in cache buffer from recent user, practically was a kind of common cache buffer function, only once the user sent a page request, proxy server would first check the existence of requesting page in cache buffer, returned it to user if exists, sent page request to remote server if no existence. The algorithm is as follow as Figure 3:



```
Procedure cache_replacement_strategy(p, cache_buffer,
                                        prefetch_queue){
  While(user request page p){
    If(p in cache_buffer)
      Recalculate weight of page p;
    Else if(p in prefetch_queue){
      While(1){ //cache buffer was available to cache p
        If(p.size <= avail(cache_buffer) ){
          cache_buffer.add(p);
          Avil(cache_buffer)= Avil(cache_buffer)-p.size;
          Prefetch_queue.delete(p);
          Calculate weight of page p;
          Break;
        }else{
          Delete page with minimum weight in cache_buffer;
        }
      }
    }else{//absent in cache_buffer nor prefetch_queue
      While(1){ //cache buffer was available to cache p
        If(p.size <= avail(cache_buffer)){
          cache_buffer.add(p);
          Avil(cache_buffer)= Avil(cache_buffer)-p.size;
          Prefetch_queue.delete(p);
          Calculate weight of page p;
          Break;
        }else{
          Delete page with minimum weight in cache_buffer;
        }
      }
    }
  }
}
```

FIGURE 3 Cache replacement strategy

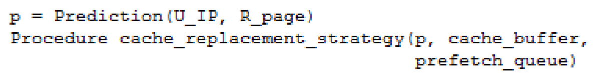
After the sequence pattern mining, a rule table had been obtained. Then the predicted buffer management would match according to the rule table specified by user IP; matching principle was to choose page with same IP and highest confidence in rule table (if that page was not in cache), if the one with equal IP could not be found, then chose the page with highest global confidence, as the page might be interested by other users [14], and the prefetching algorithm is shown as Figure 4:



```
Function Prediction(U_IP, R_page)
/* Answer: set of candidate rules */
{
  For each rule r: A → X in the rule table,
    If (A = R_page)
      Add r to Answer;
    If (Answer =∅) Then
      Return null;
    Else Find the rule r: A → X with r.IP = U_IP
                               and the highest local confidence;
    If (not found) Then
      Find the rule r: A → X with the highest global confidence;
  Return X;
}
```

FIGURE 4 Prediction algorithm

As the concept of the process was consistence to cache replacement strategy, therefore we could invoke the previous cache_replacement_strategies, difference in between was that the requesting page p was not the user clicked page, but the page predicted according to Rule Table. The algorithm for the whole process is Figure 5 as follow:



```
p = Prediction(U_IP, R_page)
Procedure cache_replacement_strategy(p, cache_buffer,
                                     prefetch_queue)
```

FIGURE 5 Invoking cache_replacement_strategy for prediction

In the cache replacement algorithm, the main concept was to remove pages with lowest weight when cache buffer was not big enough; and the weight setting should be related to visit count, size, staying time for cache and plural factors of the page, thus the formula for hypothetical weight as follow:

In the cache:

Wn(p) = L+(Wn-1(p)*T_stay/(T_cur-T_ref))/size(p)

In the above formula:

Wn(p) was the weight of page p,

L was accommodation coefficient, purpose was to avoid cache pollution

F (p) was the usage frequency for page p,

Wn-1(p) was the original weight of p,

T_stay was staying time of p,

T_cur was current time referencing p,

T_ref was the time of last reference of p,

size(p) was size of p

In prefetch queue:

Wn(p) = Pro(p)/size(p)+Wn-1(p)*1/(T_cur-T_pre)

In the above formula:

Wn(p) was the weight of page p,

Pro(P) was confidence of p

size(p) was size of p

Wn-1(p) was the original weight of p,

T_cur was current time referencing p,

T_ref was the time of last reference of p,

Formulas above followed one principle: new weight of page was related to original weight, also, shorter staying time, shorter user access interval, greater access probability; smaller page, greater weight.

5 Conclusions

We mainly proposed a Cache-Prefetching System model based on Web Data Mining, and made detail introduction of implementation of each part, included the way to generate rule table from server logs, page replacement strategy of cache buffer manager, and weight calculation of each pages. Prospect was to implement those concepts, and compared the efficiency to each cache scheduling algorithms like LRU and LFU. With the use of practical result, raise caching effectiveness by kept adjusting page weight formula and replacement strategy of cache. Such technique could be applied in personnel, educational, and official information managing system in database of educational scope. Accessing speed of educational information system for numerous teachers and students, benefit high school personnel management, and also the effective scientific structuralize educational management.

Acknowledgments

This work was supported by Major Program of National Natural Science Foundation of China (61133014, 61272415, 61272413), the CEEUSRO project of Guangdong province, China (2012A080102007, 2011J430004, 2011B090400324, 2010A011200038, 2012B040305008), Science and Technology Planning Project of Guangzhou city (2013Y2-00071), The Project for Engineering Research Center of Guangdong Province (GCZX-A1103), JNU innovation project (JNU-J-CXGCWK2014006) and the Fundamental Research Funds for the Central Universities (21613336).

References

[1] Kroeger T M, Long D D E, Mogul J C 1997 Exploring the bounds of Web latency reduction form caching and perfecting *Proceedings of the USENIX Symposium on Internet Technologies and Systems California USENIX Association* 13-22

[2] Chen Xin, Zhang Xiao dong 2003 Accurately modeling workload interactions for deploying prefetching in web sevrers *Proceedings 2003 International Conference on Parallel Processing* 427-35

[3] Xing Yong-Kang, Shao-Ping M A 2003 Modeling user navigation sequences based on multi-markov chains *Chinese Journal of Computers* **26**(11) 1510-17

[4] Wang Shi, Gao Wen, Li Jin-Tao, Huang Tie-Jun 2001 Mining Interest Navigation Patterns Based on Hidden Markov Model *Chinese Journal of Computers* **24**(2) 152-57

[5] Xu Huan-Qing, Wang Yong-Cheng A Web Pre-fetching Model Based on Analyzing User Access Pattern *Journal of Software* **14**(6) 1142-47

[6] Su Zhong, Shao-Ping M A, Yang Qiang, Zhang Hong-Jiang 2002 An N-Gram Prediction Model Based on Web-Log Mining *Journal of Software* **13**(l) 136-41

[7] Xu Bao-Wen, Zhang Wei-Feng 2001 Applying Data Mining to Web Pre-Fetching *Chinese Journal of Computers* **24**(4) l-7

[8] Yang Q, Zhang H H 2003 Web-log mining for predictive Web caching *IEEE Transactions on knowledge and Data Engineering* **15**(4) 1050-53

[9] Zhu Pei-Dong, Lu Xi-Chengg 1999 Traffic Smoothing of WWW Presending *Chinese Journal of Computer* **22**(6) 668-71

[10] Xu C, Ibrahim T I 2004 A Keyword-Based Semantic Prefetching Approach in Internet News Services *IEEE Transactions on knowledge and Data Engineering* **16**(5) 601-11

[11] Jin Zhi-Gang, Zhang Gang, Shu Yan-Tai 2001 Intelligent Prefetch and Cache Techniques based on Network Performance *Journal of Computer Research & Development* **38**(8) 1000-4

[12] Klemn R P 1999 Web Companion a friendly client-side Web prefetching agent *IEEE Transactions on knowledge and Data Engineering* **11**(4) 577-94

[13] Mareatos E P, Chronaki C E 1998 A top-10 approach to prefetching the Web *Proceedings of the Eighth Annual Conference of the Internet Society* Geneva

[14] Chen X, Zhang X 2003 A popularity-based prediction model for Web prefetching *Computer* **36**(3) 63-70

[15] Griffioen J, Appleton R 1994 Reducing file system latency using a predictive approach *Proc of USENIX Summer Conefrenee* 197-207

[16] Jiang Z, Kleinrock L1998 Web prefrtching in a mobile environment *IEEE 1nt Conference on Communications* **5**(5) 25-34

[17] Pei Cao, Edward W, Feltn Anna R 1995 A Study of Integrated Prefetching and Caching Strategies *Proc of the ACM SIGMETRICS Conference on Measurement and Modeling of ComputerSystems*

[18] Tuah N J, Kumar M, Venkatesh S 2003 Resource aware Speculative prefetching in Wireless Networks *Wireless Networks* **9**(1) 61-72

[19] Yang Qiang, Zhang Henry Hanning 2001 Integrating Web Prefetching and Caching Using Prediction Models *World Wide Web* **4**(4) 299-321

[20] Padmanabhan V N, Mogul J C 1996 Using predictive prefetching to improve World Wide Web latency *Proceedings of the ACM SIGCOMM’96 Conference* 22-36

[21] Jiang Z, Kleinrock L 1998 An adaptive network prefetch seheme *IEEE Jounral on Selected Areas in Comm-unieations* **16**(3) 358-68

[22] Jiang Z, Kleinrock L 1998 Web prefetching in a mobile environment *IEEE Int Conference on Communications* **5**(5) 25-34

[23] Palpanas T, Mendelzon A 1998 Web Prefetching using Partial match Prediction *Technical Report CSRG-376 Dept.of CS,Univ.of Toronto*

[24] Mahanti A, Eager D, Williamson C 2000 Temporal locality and its impact on Web proxy Cache Performance *Performance Evaluation* **42**(2-3) 187-203

[25] Chen X, Zhang X 2003 A Popularity-based Prediction model for Web Prefetching *Computer* **36**(3) 63-70

[26] Nanopoulos A, Katsaros D, Manolopoulos Y 2003 A data mining algorithm for generalized web Prefetching *IEEE Transactions on Knowledge and Data Engineering* **5**(5) 1155-69

Authors

Li Yongbiao, China

University studies: Jinan University
Scientific interest: Information systems