Published in the
International Journal of Web Intelligence and Agent System
IOS Press, Netherlands, vol.3, No.2, 2005, pp117-130

Exploiting Webspace Organization for Accelerating Web Prefetching

Javed I. Khan and Qingping Tao

Media Communications and Networking Research Laboratory
Department of Computer Science
Kent State University, USA
javed|qtao@kent.edu

Abstract

The paper explores how the structure of Webspace and the reading pattern of the surfer affect Web prefetch. We have conducted a series of experiments based on a new prefetch proxy and studied the prefetch performance on several dominant hyperspace structures including chain, tree, and complete graph sub-structures. The study assesses the system's responsiveness and the excess prefetching for various user interaction duration, surfing and prefetch sequences. The results show that the knowledge about the structure of Webspace can be used for intelligent prefetching. The study also offers some interesting insight for authors on how to design a prefetch friendly collection for increasing site responsiveness.

Keywords: Prefetch, User Interaction Behavior, Web Engineering

1. Introduction

For quite a few years, Web researchers have begun to explore prefetching as a potential accelerating technique [1, 2, 3, 4, 5] for web surfing. Prefetching has played a key role in hyper accelerating CPU systems. However, it is yet to meet similar level of success in web surfing. Web prefetch often creates excessive waste. Several studies found only about 2% of the prefetched data are actually used [6]. It is interesting to note that majority of the suggested web prefetching schemes resorted to the *access frequency* as the principle beacon to guide their prefetch activities. These techniques varied in the recipe for formulating the ranking. Unfortunately, only access frequency based ranking is recently found to be non-optimal [7]. While the access frequency remains an important clue, but it may not be enough. More innovation in techniques for intelligence path prediction and selection are required.

Interestingly, a few recent works can be found that have suggested the use of novel information- beyond access frequency. In a previous work [8], we suggested discerning media types of composite hypermedia- while selecting prefetch path. Most modern pages now contain embedded entities such as banners, Java applets, flash presentations, etc. with varying rendering constraints. This work demonstrated that the prefetch of individual components within composite multimedia pages could be optimally scheduled based on their types and internal rendering dependencies. Indeed for some parts, prefetch can be altogether avoided [8] without any loss of responsiveness compared to brute prefetch. Results showed considerable reduction of wasted prefetch (by almost 80%), and additional improvement in system responsiveness up to 3.6 times for heavily composite collections. Davison [9] examined another novel textual similarity-based prediction technique. This ingenious technique suggested the use of similarity of a model of the user's interest to the text in and around the hypertext anchors of recently requested Web pages in prefetch path selection.

In this paper, we discuss another potentially interesting beacon- the knowledge about hyperspace organization. A Web system is a conduit of communication between the two principal parties – the *content developer* and the *content reader*. The intermediate components – the server, the browser, the cache and the proxy-- all works as a mere facilitator in this communication. It seems therefore almost natural that the prefetch performance should be strongly dependent on the behavior of these two principals. This means, on one hand, the nature and organization of the content and on the other hand, the reading and interaction style of the reader should have an important impact on the prefetch performance. Interestingly, no previous study has focused on either. The intent of this paper is to shed some lights in this interesting void.

There are two related questions that naturally arise from the proposition. Is there any regular structure in the organization of the Web collection? Secondly, even if there is one, is it possible to exploit such structural information? In this paper, along with a performance study, we will discuss both. The paper is organized in the following way. Section 2 first presents a discussion on the general organization of the Webspace and explains the existence of *dominant regular structures*. Section 3 then focuses on the user access and interaction patterns. Section 4 then presents the architecture of a client side proxy based on prefetch system that we have implemented for this study. Finally, section 5 presents the performance.

2. Organization of Webspace

Web pages are becoming more and more sophisticated. Web designers are eager to spend serious efforts to develop aesthetically appealing pages and intuitive and friendly Web interfaces. However, currently there is very little handle available by which they can improve or affect the performance (other than reducing the graphics file sizes). Yet the organization of Web structure can have tremendous impact on prefetching performance. However, any such provisioning would require a formalism to describe Webspace organization. This is not trivial. Current Web contents come in various complex organizations. Web sites generally contain document collections. A *collection* can be viewed as well connected group of Web objects generally associated by some abstract theme. At first glace it seems collections are quite irregular. However, interestingly an analysis of recent Web pages seems to suggest that though ideal regular patterns seldom appear in the hyperlink graph representing the link structure of a collection but, a significant sub-graph tends to conform towards few regular structures.

In our modeling process, we therefore defined a concept called the **Dominant Pattern Graph (DPG)** of a collection. If a hyperlink graph is pruned to it's principally used links this pruning tends to provide a few regular graph patterns. We call it dominant pattern. The principality of hyperlinks can be determined from the design time specification by author or by frequency sorting. We easily found several major dominant patterns in massive number of collections. Below are some examples.

One common form of DPG is chain. For web-based photo albums, slides show, PDF documents, multi-page forms (however, which are static), Web-based examinations & quiz forms on each page, we typically click "Next" to move on. The surfer seems to be moving though a form of sequential chains. One of its features is that one Web page only includes one principal hyperlink. Only one Web document needs to be prefetched each time. Fig. 1(a) shows an example of photo album from CNN® news sites. This structure is now very common in the Web. We could find albums in hundreds of major news sites. Note that the page has other links as well. However, by conscious design, the author keeps only one dominant link. Also typical surfers do discover this specific organization. With their familiarity with the interface construct of 'virtual album', surfers tend to follow the chain as intended.

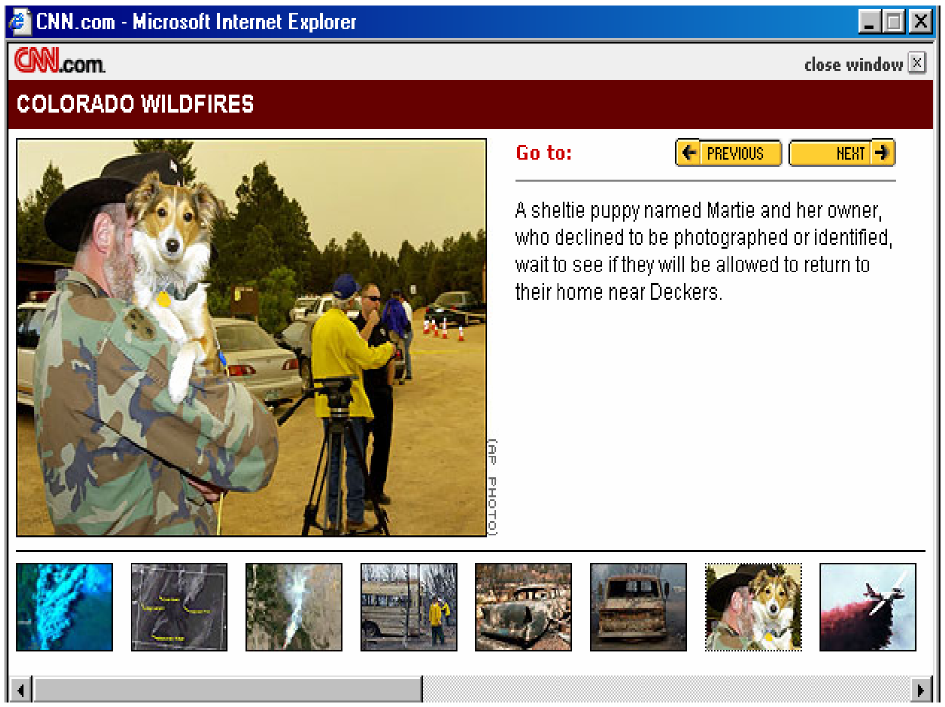
Another frequently found DPG we encountered is tree. Tree structure emerges in the central organization of complex portals. Also, it can be commonly found in the DPG of e-books, catalogues, directories, "Help" and "FAQ" pages. Each Web page includes its own hyperlinks to a set of child pages. Meanwhile, it is either a direct or indirect child page of the main page. Web page in Fig. 1(b) shows an example. It is a Navigable Map. The dominant links are the direction and the zoom level selectors. The direction navigators form tree. A tree may have many brunches. But an interface designer can often predictably guide readers towards certain brunches than others by design, and thus can reduce the branching factor of the dominant tree.

Another common dominant pattern we found is the complete sub-graph. A huge number of portal pages, particularly with sidebar and menu based organizations, show dominant patters in the form of a fully connected sub graph. Most online pages, particularly for e-books and online shops, with a common navigation side-bar or top-bar tends fall into this category of organization. Readers can easily move back and forth through any of the Web pages within the collection, no matter what the current page is. Each Web page is connected with each other. We consider this type of organization as the complete graph pattern. Fig. 1(c) shows an example for online encyclopedia with a dominant complete sub graph pattern. Also, in Fig. 1(b) the zoom levels forms a complete sub-graph.

In our study, we also found many other somewhat complex but regular patterns. An interesting one is a combination of complete graph sub-sections organized as hierarchical tree. Fig. 1(d) shows a typical example from the Kent State University's front portal. Each tab button leads to a new sub-collection. Each sub-collection has separate complete sub-graphs. This pattern appears with hierarchical table of contents, and each subgroup's table of content appearing in all pages within the subgroup. This organization is quite common in many large and deep portals (typically corporate) designed to support multiple user groups who access a site from different perspectives. Therefore, we include a forth set called "a tree with complete core" in our study.

Though, we found other more complex dominant patterns, in this paper, we will focus on the above explained four DPGs namely 1) Chain, 2) Tree, 3) Complete graph, and 4) Tree with core graph.

Published in the
International Journal of Web Intelligence and Agent System
IOS Press, Netherlands, vol.3, No.2, 2005, pp117-130



A sheltie puppy named Martie and her owner, who declined to be photographed or identified, wait to see if they will be allowed to return to their home near Deckers.

Fig. 1(a) An Example of Chain in Photo Album. The next and previous buttons represent the dominant links

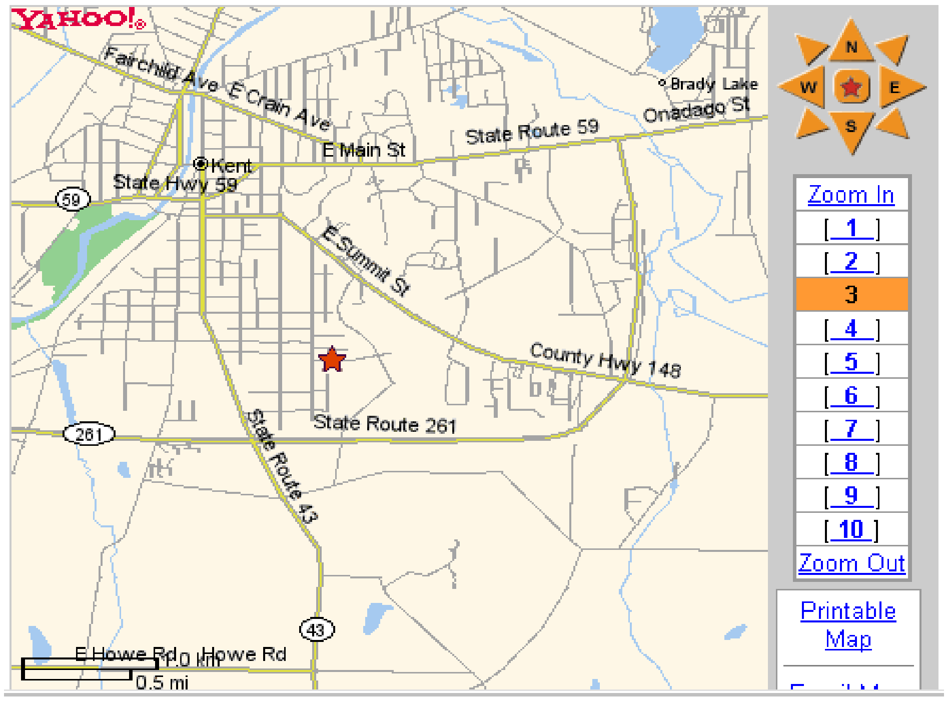
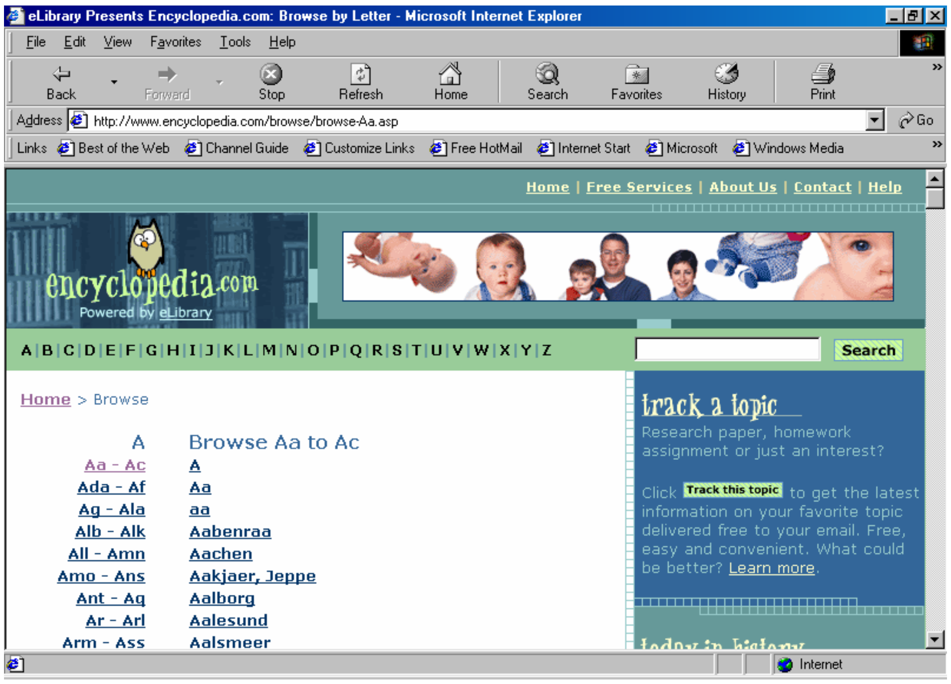


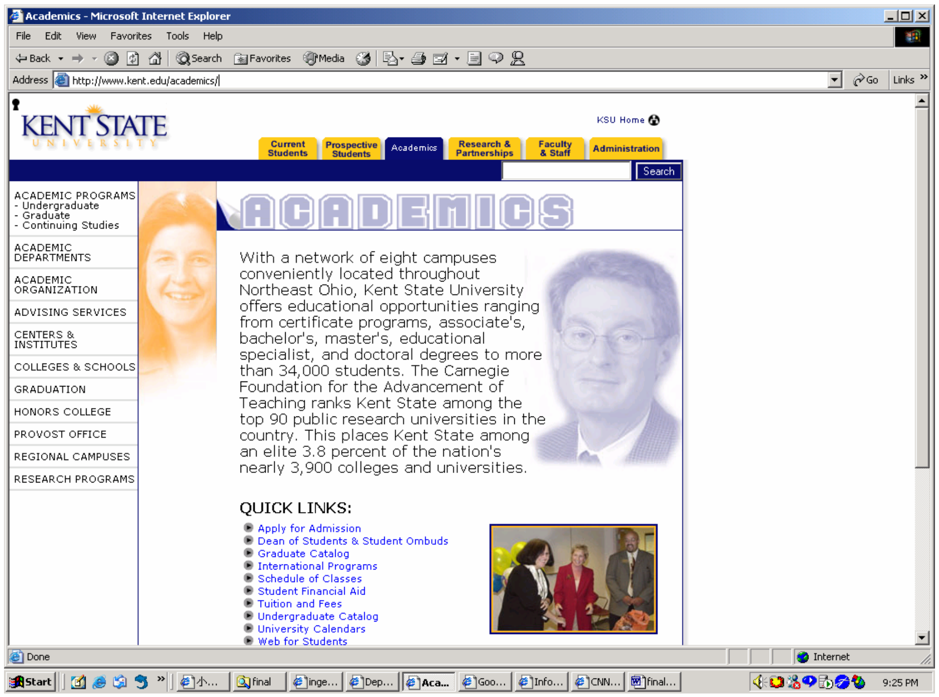
Fig. 1(b) An Example of Tree in Yahoo Map Navigation. The navigation buttons provide a dominant tree


eLibrary Presents Encyclopedia.com: Browse by Letter - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Print
Address http://www.encyclopedia.com/browse/browse-Aa.asp Go
Links Best of the Web Channel Guide Customize Links Free HotMail Internet Start Microsoft Windows Media
Home | Free Services | About Us | Contact | Help
encyclopedia.com Powered by eLibrary
A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z
Search
Home > Browse
A
Aa - Ac
Ada - Af
Ag - Ala
Alb - Alk
All - Amn
Amo - Ans
Ant - Aq
Ar - Arl
Arm - Ass
Browse Aa to Ac
A
Aa
aa
Aabenraa
Aachen
Aakjaer, Jeppe
Aalborg
Aalesund
Aalsmeer
track a topic
Track this topic
Learn more
today in history
Internet

Research paper, homework assignment or just an interest?

Click Track this topic to get the latest information on your favorite topic delivered free to your email. Free, easy and convenient. What could be better? Learn more.

Fig. 1(c) An Example of Complete Graph. The links “A”, “B”, “C” etc. appear in all the pages



With a network of eight campuses conveniently located throughout Northeast Ohio, Kent State University offers educational opportunities ranging from certificate programs, associate's, bachelor's, master's, educational specialist, and doctoral degrees to more than 34,000 students. The Carnegie Foundation for the Advancement of Teaching ranks Kent State among the top 90 public research universities in the country. This places Kent State among an elite 3.8 percent of the nation's nearly 3,900 colleges and universities.

Fig. 1(d) An Example of Tree with Core Graph. The Tabs take to another sent of complete graph sub menu

3. User Reading Behaviors

The modeling of user reading pattern is also nontrivial. There are several complex factors. Different reader has different text reading speed. It also depends on the content type. Most Web pages found in state-of-the-art sites today not only contain a simple parent HTML file with few embedded images. Pages served by modern servers today are complex and composite and contains embedded entities such as banners, Java applets, flash presentations, etc. with varying rendering constraints, and bytes per second viewing time. They generate variety of experiences beyond simple text reading. Also, various readers may have different psychological pattern guiding their browsing habit. For example, in the case of reading an online e-book, different readers view them in different surfing sequence. After finishing reading the instruction for chapter 1, some readers may continue reading section 1 of chapter 1, and other may skip to the instruction for chapter 2. Different answers will certainly result in different performance results for prefetching.

The detail modeling of the user behavior is quite complex. However, the goal of this study was to capture the essence. Therefore we limited the study on two core parameters-- 1) *relative interaction time*; 2) *surfing sequence* as elements of user interaction habit. *Interaction time* is defined as the time a reader spends on a certain page in the collection. It is the viewing duration or the interaction time between the events a user receives a requested page and sends out the second request. For the purpose of analyzing the prefetching performance, we call it interaction interval, and normalized it with respect to the entropy of the page in bytes/sec. This notion allows us to be more general than using just the reading time. The interaction time can be the time spent in watching an animation, in listening to a sound insert, or even in filling up a form. Usually, the more time readers spend on each Web page, the more Web pages can be acquired by prefetching.

The *surfing sequence* is a path of Web pages through which the user surfs. Typically the possible range of surfing sequences a surfer can follow is bounded by the design of the collection. The designer can further encourage surfer to follow certain sequences over others by tuning the layout and placement of the links. We investigated the performance for selected major patterns of surfing paths based on the graph type. The choices however, are related to the original DPG organization of the document. Therefore, these will be explained along with the DPG experiments.

4. Recording Time for Implement Event

4.1 System Setup

For this experiment we developed an in-house "organization aware" prefetch capable Proxy, and a script driven client Browser. The proxy can be collocated with a surfing client, or placed at slightly deeper egress point serving multiple clients. In our setup, we used the later. For performance analysis we inserted time tracing code inside the Proxy and the Browser. We recorded time for all events happening at the client and the proxy as per the following event model.

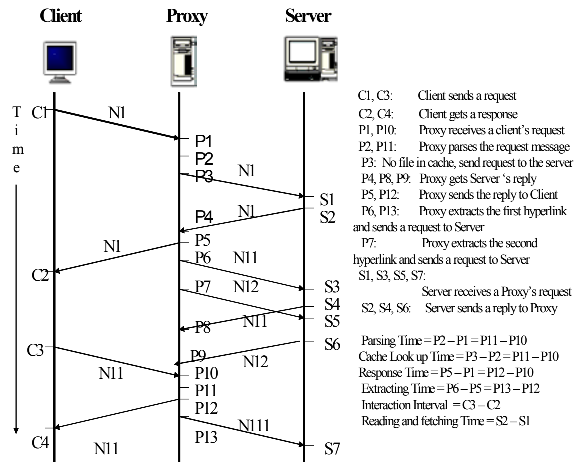

Client
Proxy
Server
Time
C1
C2
C3
C4
N1
N11
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10
P11
P12
P13
N12
N111
S1
S2
S3
S4
S5
S6
S7
C1, C3: Client sends a request
C2, C4: Client gets a response
P1, P10: Proxy receives a client's request
P2, P11: Proxy parses the request message
P3: No file in cache, send request to the server
P4, P8, P9: Proxy gets Server 's reply
P5, P12: Proxy sends the reply to Client
P6, P13: Proxy extracts the first hyperlink and sends a request to Server
P7: Proxy extracts the second hyperlink and sends a request to Server
S1, S3, S5, S7: Server receives a Proxy's request
S2, S4, S6: Server sends a reply to Proxy
Parsing Time = P2 – P1 = P11 – P10
Cache Look up Time = P3 – P2 = P11 – P10
Response Time = P5 – P1 = P12 – P10
Extracting Time = P6 – P5 = P13 – P12
Interaction Interval = C3 – C2
Reading and fetching Time = S2 – S1

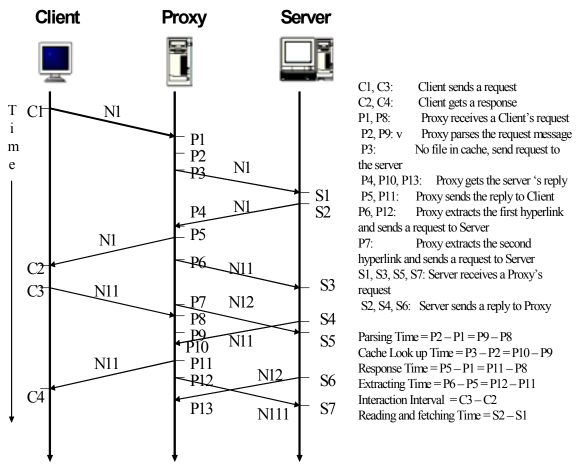
Fig. 2(a) Events Definitions and Time Distribution for Fully Folded Prefetching (FFP)

4.2 Event Model & Logging

As can be seen in the model, prefetch improves the response time in two ways. Fig. 2(a) shows the *fully folded prefetching* (FFP) and Fig. 2(b) shows the case of *partially folded prefetching* (PFP). We assume that a user wants to view Web page N1, which contains two hyperlinks to Web page N11 and N12. After finishing reading N1, it goes through N11, which has a hyperlink to Web page N111. Cn represents recording time on the client side, Pn represents recording time on the proxy side, and Sn is recording time on the server side.

After the proxy receives a request from the client (at P1), it parses the request message for the first document N1 (P2). The first request arrives with cold cache. It checks the cache directory and finds that there is no cached file for N1. So it establishes a connection to the server (P3). After getting response back from the server (P4), it sends N1 back to the client (P5). Meanwhile, the proxy extracts two hyperlinks to document N11 and N12 and prefetches them (P6 and P7) according to their priorities.

The proxy receives the server's replies (at P8 and P9). At C2, the client gets N1 and begins interaction. On the



C1, C3: Client sends a request
C2, C4: Client gets a response
P1, P8: Proxy receives a Client's request
P2, P9: v Proxy parses the request message
P3: No file in cache, send request to the server
P4, P10, P13: Proxy gets the server 's reply
P5, P11: Proxy sends the reply to Client
P6, P12: Proxy extracts the first hyperlink and sends a request to Server
P7: Proxy extracts the second hyperlink and sends a request to Server
S1, S3, S5, S7: Server receives a Proxy's request
S2, S4, S6: Server sends a reply to Proxy

Parsing Time = P2 – P1 = P9 – P8
Cache Look up Time = P3 – P2 = P10 – P9
Response Time = P5 – P1 = P11 – P8
Extracting Time = P6 – P5 = P12 – P11
Interaction Interval = C3 – C2
Reading and fetching Time = S2 – S1

Fig. 2(b) Events Definitions and Time Distribution for Partially Folded Prefetching

proxy side, we call the difference between value of P5 and P10 as interaction interval. After the proxy receives the second request from the client (P10), N11 is parsed (P11). In case of FFP (Fig-2(a)) N11 is already in proxy cache before the request for N11 arrives. By checking the cache directory, it realizes that document N11 has already been prefetched (P11). N11 can be immediately returned to the client (P12). Then the proxy continues to extract the hyperlink N111, which is embedded in document N11, and prefetches it from the server. In PFP (Fig. 2(b)), N11 is not yet in the cache although request for it is already underway. Fig. 2(b) illustrates the case. When the prefetch mechanism is turned off, then all documents are fetched using cold cache method. This is similar to the case of getting N1. We also allow passive caching to be disabled. When the passive caching is turned off then a document is removed from the proxy cache immediately after each time it is served.

4.3 Pattern Language

We also developed a set of reference collections with various organizations. This was performed by first generating a set of node documents each with a specified payload sizes. These were then linked in various ways as per the desired test pattern types.

Each hyperlink that belonged to the dominant pattern edge was given an additional attribute. It identified the hyperlink within the dominant pattern graph. We adopted a simple marking scheme as following.

For example, for Chain, we used **hyperlink attribute makers** <PATTERN=CHAIN.PREVIOUS> <PATTERN= CHAIN.NEXT> to identify the two dominant links. For Tree, the children links were marked with rank as <PATTERN=TREE.CHILD.n>. For Complete Graph, we ranked them as <PATTERN=FULL.SIBLING.n> to identify ordered siblings. For Tree with Complete Core, we ranked them as <PATTERN=TC.SIBLING.m>, and <PATTERN=TC.CHILD.n>, for identifying links to sibling and links to child sets respectively. We also provisioned an attribute marker <PATTERN=NOPREFETCH> to explicitly halt prefetching.

We then programmed the prefetch proxy to follow various **prefetch sequences** based on the dominant pattern markers found in the prefetched pages and the surfing sequence selected by the experimenter.

5. Performance Results Analysis

We evaluated a large number of collections with various organizations and various payloads. Even within a dominant pattern graph class we tested instances with large number of sizes. Given the space of this paper, we only include results from the specific but representative cases.

The objective of any prefetch system is to reduce the user waiting time and increase systems responsiveness. The main cost factor is the wasted prefetch- i.e. the data fetched but never used. We present the impact on both the performance measures: 1) *response time*; 2) the amount of *data transfer*.

The performance for response time was evaluated by the responsiveness. We define lag-time as the time the users have to wait after clicking a hyperlink. (C_i-C_{i-1}), where *i* is even. Relative responsiveness is the ratio of cumulative lag time experienced with active prefetching to that without any prefetching. (C_i-C_{i-1}), where *i* is odd is the *interaction time*.

The performance for data transfer was evaluated by recording the fetched data volumes with and without prefetch enabled. We calculated the ratio to show the relative overhead. Finally, for each experiment we also varied the interaction interval. We chose 5 seconds, 10 seconds, 15 seconds, 20 seconds, and 25 seconds as five different groups of interaction interval.

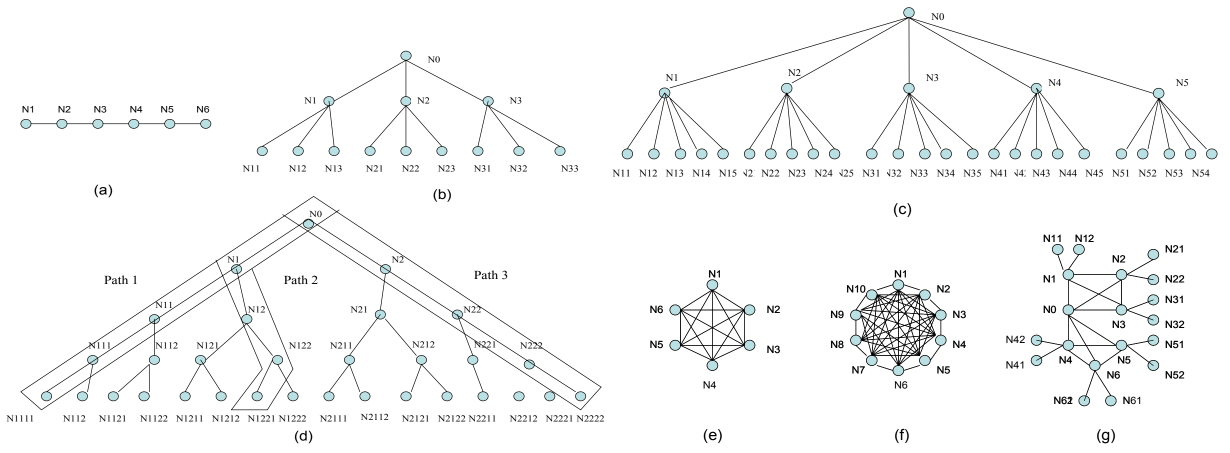

N1 N2 N3 N4 N5 N6
(a)
N0
N1 N2 N3
N11 N12 N13 N21 N22 N23 N31 N32 N33
(b)
N0
N1 N2 N3 N4 N5
N11 N12 N13 N14 N15 N22 N23 N24 N31 N33 N34 N35 N41 N43 N44 N45 N51 N52 N53 N54
(c)
N0
N1 N2
Path 1
Path 2
Path 3
N11 N12 N21 N22
N111 N112 N121 N122 N211 N212 N221 N222
N1111 N112 N1121 N1122 N1211 N1212 N1221 N1222 N2111 N2112 N2121 N2122 N2211 N2212 N2221 N2222
(d)
N1 N2 N3 N4 N5 N6
(e)
N1 N2 N3 N4 N5 N6 N7 N8 N9 N10
(f)
N11 N12 N21 N2 N1 N22 N31 N0 N3 N32 N42 N51 N41 N4 N5 N6 N52 N62 N61
(g)

Fig. 3 A Chain, Two Full Trees, a Path in Tree, Two Complete Graphs, and a Tree with Core Graph

5.1 Chain

Fig. 3(a) shows a sample test hyper graph for Chain. Here the nodes are connected in a sequence. N1 is the first view object. In a chain only one prefetch sequence is logical. For surfing sequence however, we conducted two experiments. *Half sequence* reading and *full sequence* reading. In the half sequence reading for the above graph the surfer would only read N1, N2, and N3; in full sequence reading, the surfer would visit through N1, N2, N3, N4, N5, and N6.

1). Response Time Analysis:

The performance for response time in chain is shown in Table 6 and Fig. 4. For half sequence reading the maximum improvement in responsiveness we observed is about 1.86 times. In full sequence reading of all the documents, the responsiveness improved about 4.56 times. Actually, the more documents the surfer views, the more improvement of responsiveness performance we can acquire, since we can view all documents as prefetched except for N1. We found that the system can be designed so that the responsiveness is not affected by interaction interval. The minimum interaction interval can guarantee that one Web document could be prefetched.

2) Data Volume Analysis:

The performance for data volume in chain is shown in Table 7 and Fig. 5. When the surfing sequence is N1, N2, and N3, the maximum amount of data is 4 units. Compared to the data volume without prefetching, only one extra unit data volume was increased. If we view all 6 documents, 6 units of data volume will be transferred and no extra amount of data is produced. So, whatever the surfing sequence is, the maximum extra data volume is one unit.

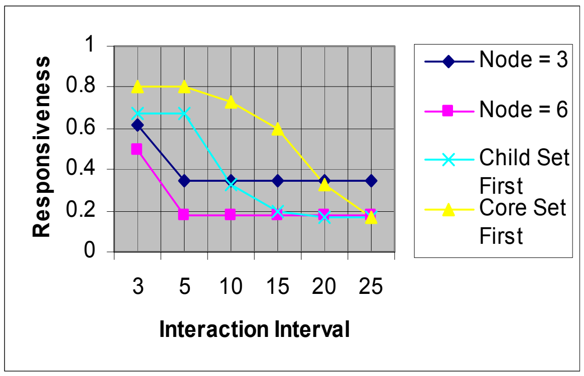


Fig. 4 Performance for Response Time in a Chain and a Tree with Core Graph

5.2 Analysis of Tree

Published in the
International Journal of Web Intelligence and Agent System
IOS Press, Netherlands, vol.3, No.2, 2005, pp117-130

For tree experiment we generated collections with varying heights and breadths. Two examples are shown. In Fig. 3(b), 13 nodes are organized into a tree with three levels (height *H* equals 3). Each of N0, N1, N2, N3, contains three hyperlinks. The branch factor (*BF*) equals 3. In Fig. 3(c), height also equals 3, but branch factor is 5. Each of N0, N1, N2, N3, N4 and N5 contains five hyperlinks. We considered two types of tree reading 1) full tree reading and 2) a path in a tree reading. Unlike chain, a tree can be surfed in several orders. We used two prefetching sequences 1) *Left first* and 2) *Right First*. The corresponding node sequences are shown in table 1.

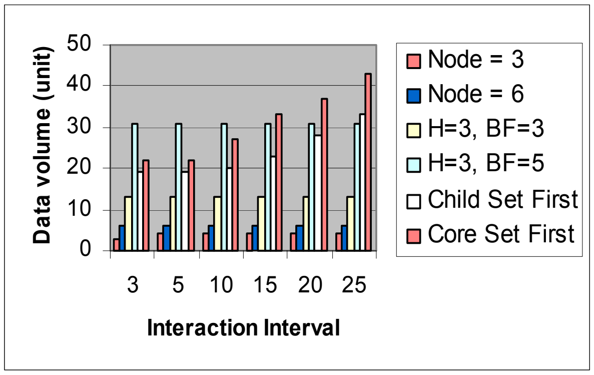


Fig. 5 Performance for Data Volume in a Chain, a Tree and a Tree with Core Graph

5.2.1 Full Tree Reading

To test various surfing behavior of the full tree reading we let the surfer use three different surfing sequences: 1) *Depth First* (α), 2) *Breadth First* (β), and 3) *Random Connected Walk* (γ). Except for the Random Connected Walk, the both depth first and breadth first ordered the nodes left to right. We repeated the experiment only for Canonical cases. Symmetrical cases were not considered. The sample node sequences for various runs for the sample graph are shown in table 2.

Type	Node	Prefetching Sequence	
		Left First	Right First
H = 3 BF = 5	N0	N1,N2,N3,N4,N5	N5,N4,N3,N2,N1
	N1	N11,N12,N13,N14,N15	N15,N14,N13,N12,N11
	N2	N21,N22,N23,N24,N25	N25,N24,N23,N22,N21
	N3	N31,N32,N33,N34,N35	N35,N34,N33,N32,N31
	N4	N41,N42,N43,N44,N45	N45,N44,N43,N42,N41
	N5	N51,N52,N53,N54,N55	N55,N54,N53,N52,N51
H =3 BF = 3	N0	N1,N2,N3	N3,N2,N1
	N1	N11,N12,N13	N13,N12,N11
	N2	N21,N22,N23	N23,N22,N21
	N3	N31,N32,N33	N33,N32,N31

Table 1 Lists of Prefetching Sequences in a Full Tree

1). Response Time Analysis:

The performances for response time in a full tree with Left First and Right First as prefetching sequence are shown in Table 6, Fig. 6 and Fig. 7 respectively.

Type	Surfing sequence		
	Depth First	Breadth First	Random
H = 3 BF = 5	N0, N1, N11, N12 N13, N14, N15, N2, N21, N22, N23, N24, N25, N3, N31, N32, N33, N34, N35, N4, N41, N42, N43, N44, N45, N5, N51, N52, N53, N54, N55	N0, N1, N2, N3, N4, N5, N11, N12, N13, N14, N15, N21, N22, N23, N24, N25, N31, N32, N33, N34, N35, N41, N42, N43, N44, N45, N51, N52, N53, N54, N55	N0, N4, N41, N42 N2, N21, N22, N23, N24, N25, N3, N33, N31, N32, N34, N35, N1, N5, N52, N53 N54, N55, N51, N43, N44, N45, N11, N12, N13, N14, N15
H = 3 BF = 3	N0, N1, N11, N12 N13, N2, N21, N22, N23, N3, N31, N32, N33	N0, N1, N2, N3, N11, N12, N13, N21, N22, N23, N31, N32, N33	N0, N1, N2, N3, N11, N12, N13, N21, N22, N23, N31, N32, N33

Table 2 Lists of Surfing Sequences in a Full Tree

We observe that the prefetching sequence and surfing sequence affect the performance for response time. The improvement in responsiveness is the best when we compare reading Web documents in Depth First manner compared to Breadth First or Random. In Fig. 6, when prefetching sequence is Left First, The responsiveness with Random and Breadth First is up to 2.4 and 3.7 times less than that with Depth First respectively. In Fig. 7, when prefetching sequence is Right First, the responsiveness with Random and Breadth First is up to 0.6 and 0.7 times less than that with Depth First respectively. We also observed that no matter what the prefetching sequence is, with the branching factor increasing, the impact of prefetching performance always increases. In addition, with growing interaction interval, the relative responsiveness improves (value decreases) gradually. There seems to be an intuitive explanation. The more is the interaction interval the more is the scope of effective prefetching. It seems it might be possible to determine a matched prefetching state where a web page designer might be able to space the links while controlling the content volume. 2). Data Volume Analysis:

The performance for data volume in a full tree is shown in Table 7 and Fig. 5. Whatever the branching factor is, data volume is not affected by the prefetching sequence or the surfing sequence. Interaction interval does not affect overall data volume. The total amount of transferred data is the same as without prefetching. However, what matters is the branching. When the branching factor is 5, data volume is 31 units; when the branching factor is 3, data volume is 13 units. Therefore, a case where excessive overload is a concern the prefetch proxy should limit the branching factor allowable for prefetch.

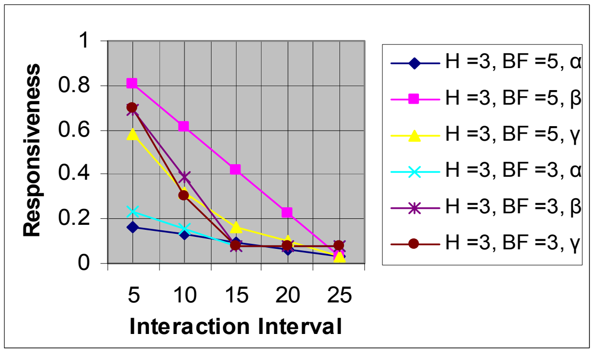


Fig. 6 Performance for Response Time in Tree with Left First (α- Depth First, β– Breadth First, γ– Random Connected Walk)

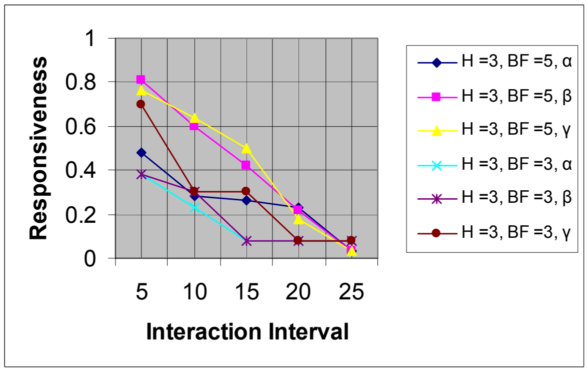


Fig. 7 Performance for Response Time in Tree with Right First (α– Depth First, β– Breadth First, γ– Random Connected Walk)

5.2.2 Paths in a Tree

In this set of experiments (Fig. 3(d)) we consider the case where the surfer chooses to visit only one path from root to leave in a tree. However, we again choose three variants.1) *Left Path (Path 1),* 2) *Right Path (Path 2)*, and 3) *Random Chain Walk (Path 3)*. For the graph shown in Fig. 3(d), in path 1, the surfing sequence is N0, N1, N11, N111, and N1111 in order; in path 2, the surfing sequence is N0, N1, N12, N122, and N1221; and in Path 3 the sample random surfing sequence is N0, N2, N22, N222, and N2222. We conducted experiment based on two different prefetching sequences: 1) *Left First (LF)* and 2) *Right First (RF)*. We adopt the same implementation methods as in the experiment with a full tree.

1). Response Time Analysis:

The performance for response time in one path in a tree reading is shown in Table 6 and Fig. 8. We observe that path 1's responsiveness with Left First prefetching sequence is the same as path 3's one with Right First prefetching sequence.

We can also find that path 3's responsiveness with Left First prefetching sequence is the same as path 1's responsiveness with Right First prefetching sequence. If interaction interval is 5 seconds, the response time with prefetching is the same as that without prefetching, since the next page we will move through is not a prefetched document. However, whatever prefetching sequence is, either Left First or Right First, path 2 always has the same change for the responsiveness value.

When prefetching sequence is Left First, the prefetching performance in path 1 is better than that in path 2 and path 3. The responsiveness with path 2 and path 3 is up to 2 and 4 times less than that with path 1 respectively. With growing interaction interval, the system responsiveness always increases in a gradual fashion for path 1, path 2, and path 3.

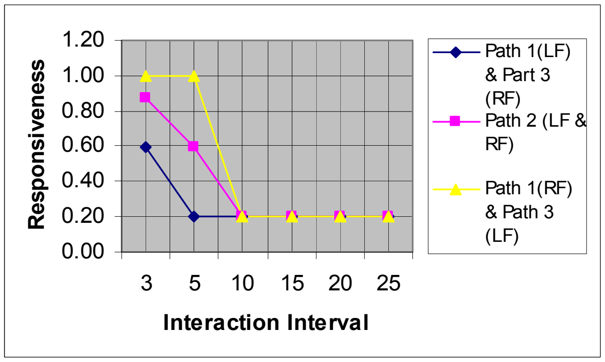


Fig. 8 Performance for Response Time in Paths of a Tree

2). Data Volume Analysis:

The performance for data volume in one path in a tree reading is shown in Table 7 and Fig. 9. If interaction interval is 5 seconds, path 1's data volume with Left First prefetching sequence is (5 units) same as the path 3's one with Right First prefetching sequence. The path 3's data volume with Left First prefetching sequence is (9 units) same as the part 1's one with Right First prefetching sequence. No matter what prefetching sequence path 2 uses, its data volume is 7 units. With Left First prefetching sequence, the amount of unnecessary data in path 2 and path 3 is up to 40% and 80% more than that in path 1 respectively. Once it reaches 10 seconds, the performance for data volume in part 1, part 2, and part 3 are the same. They are all 9 units no matter what is their prefetching sequence.

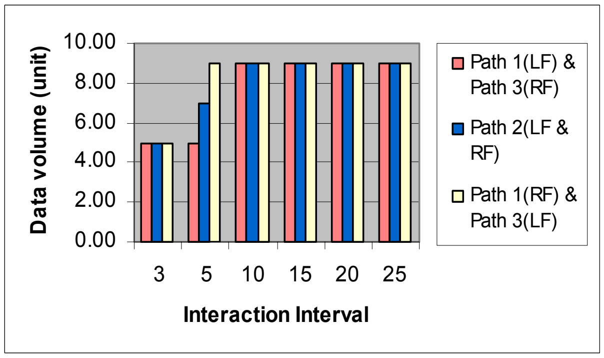


Fig. 9 Performance for Data Volume in Paths of a Tree

5.3 Complete Graph

A complete graph varies only with respect to the size of the clique. Two examples respectively with 5 and 9 hyperlinks are shown in Fig. 3. We consider the case of clockwise prefetch sequence (anticlockwise prefetch creates symmetrical cases). The nodes in prefetch sequence are shown in Table 3. With respect to it, we consider three different types of surfing sequences. These are 1) Clockwise (CW), 2) Counter Clockwise (CCW), and 3) Random Walk (RW). These walks are shown in Table 4.

1). Response Time Analysis:

The performance for response time in a complete graph is shown in Table 6 and Fig. 10. As expected, no matter how many nodes they have, the prefetching performance in clock matched reading direction is always better than that in counter clock matched case.

The prefetching performance in random reading direction is in between clockwise and in counter clockwise directions. The responsiveness with Random and Counter Clockwise is up to 5.3 and 10.3 times less than that with CW respectively. With growing number of nodes, the impact of prefetching performance increases. With growing interaction interval, the system responsiveness increases in a gradual fashion.

2). Data Volume Analysis:

The performance for data volume in complete graph is shown in Table 7 and Fig. 11. Different surfing sequences result in different performance of data volume. The amount of data in matched sequence (clockwise) reading direction is always less than that in reversed sequence (counter clockwise). We again note that the data volume for any reading order always increases gradually when interaction interval increases gradually. All of them produce a lot of extra amount of data compared to the amount of transferred data without prefetching. The more nodes we move through, the more extra amount of data is produced.

Total Nodes	Node	Prefetching Sequence
		Clockwise
6	N1	N2,N3,N4,N5,N6
	N2	N3,N4,N5,N6,N1
	N3	N4,N5,N6,N1,N2
	N4	N5,N6,N1,N2,N3
	N5	N6,N1,N2,N3,N4
	N6	N1,N2,N3,N4,N5
10	N1	N2,N3,N4,N5,N6,N7,N8,N9,N10
	N2	N3,N4,N5,N6,N7,N8,N9,N10,N1
	N3	N4,N5,N6,N7,N8,N9,N10,N1,N2
	N4	N5,N6,N7,N8,N9,N10,N1,N2,N3
	N5	N6,N7,N8,N9,N10,N1,N2,N3,N4

Published in the
International Journal of Web Intelligence and Agent System
IOS Press, Netherlands, vol.3, No.2, 2005, pp117-130

	N6	N7,N8,N9,N10,N1,N2,N3,N4,N5
	N7	N8,N9,N10,N1,N2,N3,N4,N5,N6
	N8	N9,N10,N1,N2,N3,N4,N5,N6,N7
	N9	N10,N1,N2,N3,N4,N5,N6,N7,N8
	N10	N1,N2,N3,N4,N5,N6,N7,N8,N9

Table 3 Lists of Prefetching Sequences in a Complete Graph

Total Nodes	Surfing sequence		
	Clockwise	Counter Clockwise	Random Walk
6	N1,N2,N3,N4, N5,N6	N1,N6,N5,N4, N3,N2	N1,N4,N6,N2, N5,N3
10	N1,N2,N3,N4, N5N6,N7,N8, N9,N10	N1,N10,N9,N8, N7,N6,N5,N4, N3,N2	N1,N6,N3,N5, N9,N7,N2,N8, N4,N10

Table 4 Lists of Surfing sequences in a Graph

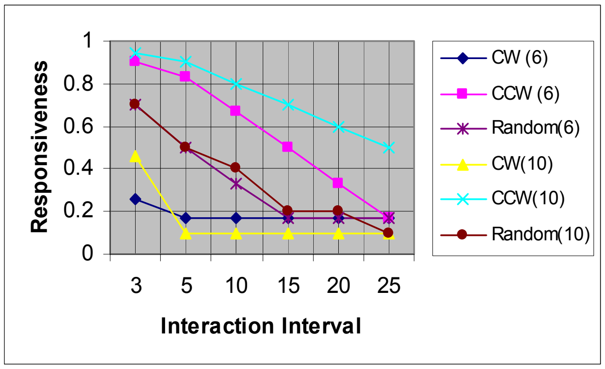


Fig. 10 Performance for Response Time in Complete Graph

5.4 Tree with Core Graph

Fig. 3(g) shows an example of test tree with core graph. Here one core consists of N0, N1, N2, and N3. We refer to it as core 1. Another core consists of N4, N5, and N6. We refer to it as core 2. Each node is a parent in the core. It has its own children. For instance, N0 is a member of core 1. Meanwhile, it is the parent of three children, N4, N5, and N6, which are members of core 2. Core Set means all members of the core are fully connected. Child Set is connected via a tree structure with the core. With respect to this DPG, two types of prefetching sequences are selected. We call them 1) *Core Set First* and 2) *Child Set First*. We use one Depth First surfing sequence here (shown in Table 5).

1). Response Time Analysis:

The performance for response time in Tree with Core is shown in Table 6 and Fig. 4. With interaction interval increased, the value of responsiveness decreases gradually for both Core Set First and Child Set First. However, if we use Child Set First as prefetching sequence, its performance improvement for responsiveness is better than Core Set First. That means Child Set First prefetch closely matches the Depth First surfing sequence. The responsiveness with Core Set First is up to 2 times less than that with Child Set First.

2). Data Volume Analysis:

The performance for response time in Tree with Core is shown in Table 7 and Fig. 5. If Child Set First is selected as prefetching sequence, its performance improvement for data volume is better than Core Set First. The amount of unnecessary data with Core Set First is up to 43% more than that with Child Set First.

With interaction interval increased, the data volume increases gradually for both of them, and the extra amount of data also increase gradually.

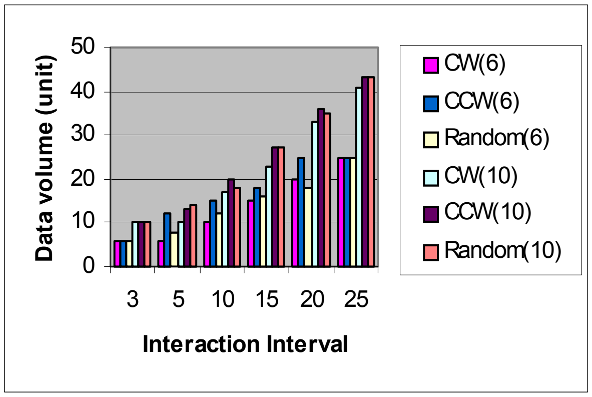


Fig. 11 Performance for Data Volume in Complete Graph

6. Conclusions and Future Works

First generation of prefetch technique suggested schemes dependant primarily on “frequency” of access analysis. In this paper, we presented an study on the impact of web-space organization and corresponding surfing sequences on prefetch. It seems to suggest that smarter prefetching techniques can be developed if the structure of Webspace and user reading behavior can also be brought into consideration.

We have observed the existence of dominant pattern graphs in the Webspace particularly in large collections. The paper presents experiments on several types of abstract yet commonly occurring dominant patterns in hyperspace including chain, tree, complete graph, and complex tree with core. Here is the brief summary of the overall findings. For the cases studied we found that compared to a random prefetch system (organization unaware), the response time of a matched system (where the prefetch system can take advantage of the Webspace organization) can be 1.6 - 6.3 times faster. Not only that, it can also dramatically bring down the amount of unnecessary prefetch down by a factor of 1.8 - 2.0 or more. Also, in the worst case, a completely mismatched system's response time can be about 1.7 - 11.3 times slower and can result in 1.3 - 1.4 times more unnecessary data transfer than a well matched system.

6.1 Author Driven Organization

Now clearly the question is if such a scheme realizable? User interest is probably their. With the maturity of Web content design industry, now much interest exists in the design of aesthetically as well as fast accessible Web pages. Indeed, we suspect the interest is probably much ahead than what current technology supports.

Organization Type				Responsiveness					
				3 s	5 s	10 s	15 s	20 s	25 s
Chain	3 Nodes			0.62	0.35	0.35	0.35	0.35	0.35
	6 Nodes			0.50	0.18	0.18	0.18	0.18	0.18
Tree	Full Tree	BF=3	Depth First		0.23	0.15	0.08	0.08	0.08
			Breadth First		0.69	0.38	0.08	0.08	0.08
			Random		0.70	0.30	0.08	0.08	0.08
		BF=5	Depth First		0.18	0.15	0.10	0.08	0.03
			Breadth First		0.81	0.61	0.42	0.23	0.03
			Random		0.58	0.32	0.16	0.10	0.03
	Paths in Tree	Path 1	Left First	0.60	0.20	0.20	0.20	0.20	0.20
			Right First	1.0	1.0	0.20	0.20	0.20	0.20
		Path 2	Left First	0.87	0.60	0.20	0.20	0.20	0.20
			Right First	0.87	0.60	0.20	0.20	0.20	0.20
		Path 3	Left First	1.0	1.0	0.20	0.20	0.20	0.20
			Right First	0.60	0.20	0.20	0.20	0.20	0.20
Complete Graph	6 Notes	Clockwise		0.26	0.17	0.17	0.17	0.17	0.17
		Counter Clockwise		0.90	0.83	0.67	0.50	0.33	0.17
		Random		0.70	0.50	0.33	0.17	0.17	0.17
	10 Notes	Clockwise		0.46	0.10	0.10	0.10	0.10	0.10
		Counter Clockwise		0.94	0.90	0.80	0.70	0.60	0.50
		Random		0.70	0.50	0.40	0.20	0.20	0.10
	Child Set First			0.67	0.67	0.33	0.20	0.17	0.17

Tree with Core Graph	Core Set First	0.80	0.80	0.73	0.60	0.33	0.17

Table 6 the Performance for Response Time in All Organization Types

Organization Type				Data Volume					
				3 s	5 s	10 s	15 s	20 s	25 s
Chain	3 Notes			3	4	4	4	4	4
	6 Notes			6	6	6	6	6	6
Tree	Full Tree	BF=3	Depth First	13	13	13	13	13	13
			Breadth First	13	13	13	13	13	13
			Random	13	13	13	13	13	13
		BF=5	Depth First	31	31	31	31	31	31
			Breadth First	31	31	31	31	31	31
			Random	31	31	31	31	31	31
	Paths in Tree	Path 1	Left First	5	5	9	9	9	9
			Right First	5	9	9	9	9	9
		Path 2	Left First	5	7	9	9	9	9
			Right First	5	7	9	9	9	9
		Path 3	Left First	5	9	9	9	9	9
			Right First	5	5	9	9	9	9
Complete Graph	6 Notes	Clockwise		6	6	10	15	20	25
		Counter Clockwise		6	12	15	18	25	25
		Random		6	8	12	16	18	25
	10 Notes	Clockwise		10	10	17	23	33	41
		Counter Clockwise		10	13	20	27	36	43
		Random		10	14	18	27	35	43
				19	19	20	23	28	33

Tree with Core Graph	Child Set First						
	Core Set First	22	22	27	33	37	43

Table 7 the Performance for Data Volume in All Organization Types

Node	Prefetching Sequence		Surfing Sequence (Depth First)
	Child Set First	Core Set First	
N0	N1,N2,N3,N5, N4,N6	N1,N2,N3,N5, N4,N6	N0,N4,N41,N42, N6,N61,N62,N5, N51,N52,N1,N11, N12,N2,N21,N22, N3,N31, N32
N1	N11,N12,N2, N3, N0	N2,N3,N0,N11, N12	
N2	N21,N22,N3, N0,N1	N3,N0,N1,N21, N22	
N3	N31,N32,N0, N1,N2	N0,N1,N2,N31, N32	
N4	N41,N42,N0, N5,N6	N5,N6,N41, N42,N0	
N5	N51,N52,N0, N6,N4	N6,N4,N51, N52,N0	
N6	N61,N62,N0, N4,N5	N4,N5,N61, N62,N0	

Table 5 Lists of Sequences in a Tree with Core Graph

The main technological hindrance is that current HTTP or HTML has no mechanism, which designers can use to author a prefetch friendly collection. Currently there is no standard technique to express a hyperspace pattern. However, the simple **hyperlink attribute markers** we have used for the sake of this experiment suggest that a marking language can easily be developed to provision content driven pattern specification. Any trivial extension of it, along with an "organization aware" browser or proxy that can support some prefetch sequencing policy, can significantly accelerate Web surfing at ease in a **prefetch-friendly collection**.

6.2 Authoring Tools

Indeed, authoring tools can also be easily enhanced that will encourage content developer to mark at least one or two dominant hyperlink(s) among the links s/he embeds. The efforts should not be more than adding alternate text for embedded images. Quite often, it is already known by the content author. Content author generally follows a premeditated theme based mental organization to hyperlink the collection. Also, the marking can be automatically generated by many converters (such as PowerPoint® to HTML Converter).

6.3 Finding Patterns in Legacy HTML

Interestingly, dominant organization of a collection can often be reverse engineered at post production stage (such as by log or frequency analysis). A pre-existing collection can be potentially made prefetch friendly with some simple automated document analysis in many special cases. For example, it is relatively easy to identify chains. Almost out of any collection, a dominant chain can be discovered by simple modification of several currently available server tools. Prefetch chain always increases surfing responsiveness and it does not fetch any extra load. Also, the documents involved in a dominant pattern tend to be co-located in a single server. For example, a complete graph cluster is generally placed in single directory.

6.4 Other Issues

An interesting advance problem will be to extract pattern information when the hyperspace spans multiple servers and multiple collections. Perhaps an HTTP extension can used to see if the dominant pattern can be found. We suspect reading time will show high correlation with media type and content. Additional study can be performed to determine the extents.

The beacon suggested here can be combined with other techniques currently known. An approach based on intelligent analysis of surfer's bookmarks, history of recently visited pages, and nearby Webspace structure, combined with data reduction techniques such as one based on partial prefetch can potentially yield a powerful prefetch system with quite accelerated surfing performance.

7. References

[1] T. Kroeger, D. D. E. Long & J. Mogul, Exploring the Bounds of Web Latency Reduction from Caching and Prefetching, Proc. of USENIX Symp. on Internet Technology and Systems, Monterey, December, 1997, pp. 319-328.

[2] P. Pirolli and J. E. Pitkow, Distributions of surfers' paths through the World Wide Web: Empirical characterizations, Jounral of World Wide Web, v.1-2, 1999, pp. 29-45.

[3] A Non-interfering Deployable Web Prefetching System**,** R. Kokku, P. Yalagandula, A. Venkatramani, M. Dahlin, Proceedings of the USENIX Symposium on Internet Technologies and Systems, March, 2003, pp. 183-196.

[4] Storage allocation in Web prefetching techniques, Daniel D. Zeng, Fei-Yue Wang, Sudha Ram, Proceedings of 4th ACM Conference on Electronic Commerce (EC-2003), San Diego, California, June, 2003, pp. 264-265.

[5] Yuna Kim, Jong Kim. Web Prefetching Using Displayed-Based Prediction. IEEE/WIC International Conference on Web Intelligence (WI'03), Halifax, Canada, October, 2003. pp. 486-489.

[6] M. Frans Kaashoek, Tom Pinckney, and Joshua A. Tauber, Dynamic Documents: Extensibility and Adaptability in the WWW, http://www.pdos.lcs.mit.edu/papers/www94.html.

[7] Javed I. Khan, Qingping Tao, Partial Prefetch for Faster Surfing in Composite Hypermedia, the 3rd USENIX Symposium on Internet Technologies USITS'01, San Francisco, March, 2001, pp13-24.

[8] Javed I. Khan, Qingping Tao, Prefetch Scheduling for Composite Hypermedia, Proceedings of IEEE International Conference on Communications (ICC2001), Finland, June, 2001, pp. 768-773.

[9] Brian D. Davison, Predicting Web Actions from HTML Content, In *Proceedings of the* The Thirteenth ACM Conference on Hypertext and Hypermedia (HT'02), College Park, MD, June, 2002, pp159-168.