Proceedings of the IEEE/WIC International Conference on Web Intelligence, WI'03, October 2003, Halifax, Canada, pp.579-583

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 can be used for accelerated Web prefetch. We have conducted experiments based on a novel hyperspace aware prefetch proxy and have studied the prefetch performance on several dominant hyperspace patterns. The study assesses the system's responsiveness and background loads for various user interaction duration, surfing and prefetch sequences. This paper also draws attention to the emergence of some regular patterns in contemporary webspace. The results show that webspace awareness can help in improving prefetch performance. This study also provides an interesting insight toward a framework where the professional content developers can gain more control towards authoring prefetch friendly collection for increased site responsiveness.

1. Introduction

For quite a few years, web researchers have begun to explore prefetch as one of the potential accelerator [1,2,3]. It has played a key role in hyper accelerating CPU systems. However, despite initial optimism its success in Web system has been remained surprisingly illusive. Indeed there are several difficulties. Web prefetch often creates excessive waste. Several studies found only about 2% of the prefetched date is actually used [4]. Unlike hardware systems, the bandwidth in web is a shared resource. Web prefetch seems to be a harder problem because typically the branching in instruction control flow is limited due to constructs of programming languages. There is no such limit in the number of hyperlinks. Thus it is easy to overwhelm a web prefetch system. Even it can create large TCP connection overhead [3]. It is interesting to note that most of the suggested schemes until now resorted to the *access frequency* as the principle beacon to guide the prefetch activities, though these varied in the recipe for formulating the ranking. Only access frequency based ranking is now known to be non-optimal [5]. It seems while the access frequency remain as an important clue, but it may not be enough to take prefetch technology to a point of maturity. Clearly, more innovations, particularly which can add new intelligence in prediction are required.

More recently, several new and promising beacons have been investigated. For example, in a previous work [6] we suggested discerning media types of composite hypermedia. Most modern pages now contain embedded entities such as banners, Java applets, flash presentations, etc. with varying rendering constraints. This work demonstrated that the prefetching of individual components within composite multimedia pages could be optimally scheduled based on their type and internal rendering dependencies. Indeed for some parts, prefetch can be altogether avoided [7] without any loss of responsiveness at par with indiscriminating prefetch. Results showed dramatic reduction of wasted prefetch (by almost 80%), and additional improvement in system responsiveness by about 3.6 times for heavily composite collections. Davison [8] examined another novel textual similarity-based prediction technique, which are made using the similarity of a model of the user's interest to the text in and around the hypertext anchors of recently requested Web pages.

In this paper, we present another interesting approach. This intelligence based prefetching utilizes 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 designers are already eager to spend serious efforts to develop aesthetically appealing pages and intuitive and friendly web interfaces. However, today there is very little handle available by which one can improve or affect the performance (other than reducing the graphics file sizes). In fact, the organization of Web structure can have tremendous impact on prefetching performance.

The modeling of content organization 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. Interestingly, an analysis of recent Web pages seems to suggest that although there is no concrete discipline, a few patterns do tend to emerge within collections. Ideal regular patterns seldom appear in the graph representing the link structure of a collection. However, in the generalized graph, a significant sub-graph tends to conform to a regular structure. 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 its principally used links this pruning tends to provide a few regular graph patterns. We call it dominant pattern. We easily found several major dominant patterns in massive number of collections. Below are some examples.

One common form is chain. For photo albums, slides show, PDF documents, multi-page forms (however, which are static), Web-based examinations & quiz forms on each page, surfer typically clicks Next to move though a form of sequential chains. One of its features is that one Web page only includes one principal hyperlink. 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, the page has other links as well. However, by conscious design, the author keeps only one dominant link. With their familiarity with the interface construct of 'virtual album', typical surfers tend to discover and follow the chain as intended.

Another frequently found dominant pattern we encountered is tree. Tree structure emerges in the central organization of complex portals. Also, it can be found in 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 Map Navigation, and the dominant links are the direction and the zoom level selectors. The direction navigators form tree. 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 is the complete sub-

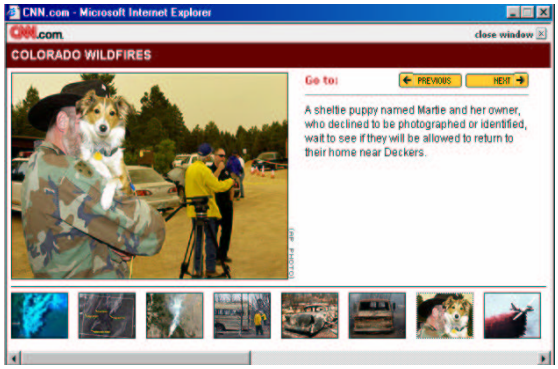

Fig. 1(a) An Example of Chain in Photo Album. The next and previous buttons represent the dominant links

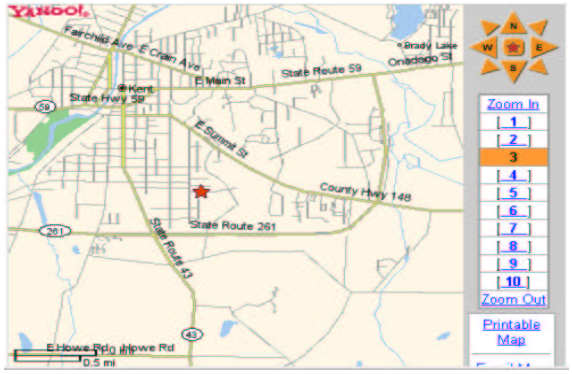

Fig. 1(b) An Example of Tree in Yahoo Map Navigation. The navigation buttons provide a dominant tree

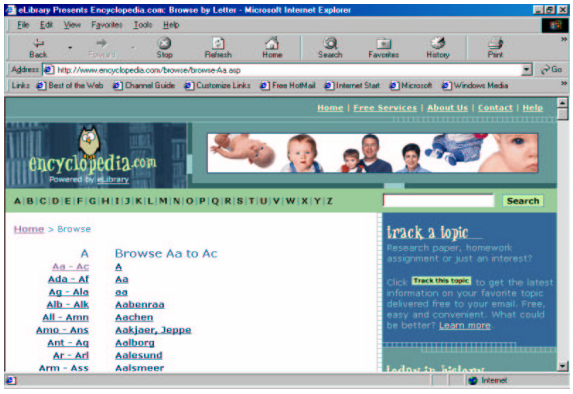

Fig. 1(c) An Example of Complete Graph. The links "A", "B", "C" etc. appear in all the pages

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

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. 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.

We also found several 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 common in many corporate portals designed to support multiple user groups who access a site with widely different perspectives. Therefore, we include a forth set called "a tree with complete core graph" in our study. In this paper, we will focus on the above four dominant patterns: 1) Chain, 2) Tree, 3) Complete graph, and 4) Tree with complete core.

3. User Reading Behaviors

The modeling of user reading pattern is also nontrivial. There are several complex factors. Text reading speed varies with individuals. It also depends on the content type. Also, pages served by modern servers today contains embedded entities such as banners, Java applets, flash presentations, etc. with varying rendering constraints, and bytes density. They generate variety of experiences beyond simple text or mono media reading. Also, various readers may have different psychological pattern guiding their browsing habit. 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 time between the events a user receives a requested page and sends out the next request. For the purpose of analyzing the prefetching performance, we normalize it with respect to the entropy of the page in bytes/sec. This notion allows us to be more general than reading time. The interaction time can be the time spent in watching an animation, listening a sound insert, or even 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 highly bounded by the design of the collection. By design the designer can further encourage surfer to follow certain sequences over others. We investigated the performance for graph specific surfing sequences. These will be shortly.

4. Recording Time for Implement Event

4.1 System Setup

For this experiment we developed an in-house "organization aware" prefetch Proxy, and a Script Browser. The proxy can be collocated with one surfing client, or can be placed at a slightly deeper egress point serving multiple clients. In our set up, we used the later. For performance analysis, we added detailed time trace code inside this proxy as well as in the Browser. The events were recorded as per the following model.

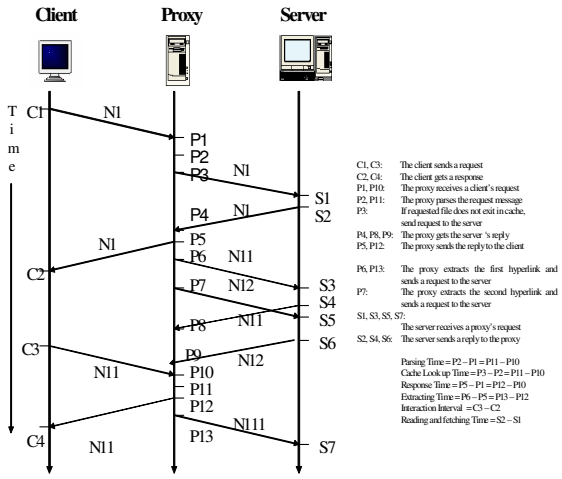

Fig. 2 Events Definitions and Time Distribution for ally Folded Prefetching (FFP)

4.2 Event Model & Logging

Fig. 2 shows the event model and the notation with an example. A user wants to view Web page N1, which contains two hyperlinks to pages N11 and N12. After finishing reading N1, surfer goes through N11, which has a hyperlink to Web page N111. Cn, Pn, and Sn respectively represent recording time on the client side, proxy side, and 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. Proxy finds it is not in cache. 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 the hyperlinks to N11 and N12 and prefetches them (P6 and P7). The proxy receives the server's replies (at P8 and P9). At C2, the client gets N1 and interaction begins. On the 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). Since, N11 is already in proxy cache it can be immediately returned to the client (P12).

4.3 Pattern Language

The reference collections were generated 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 or 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 a

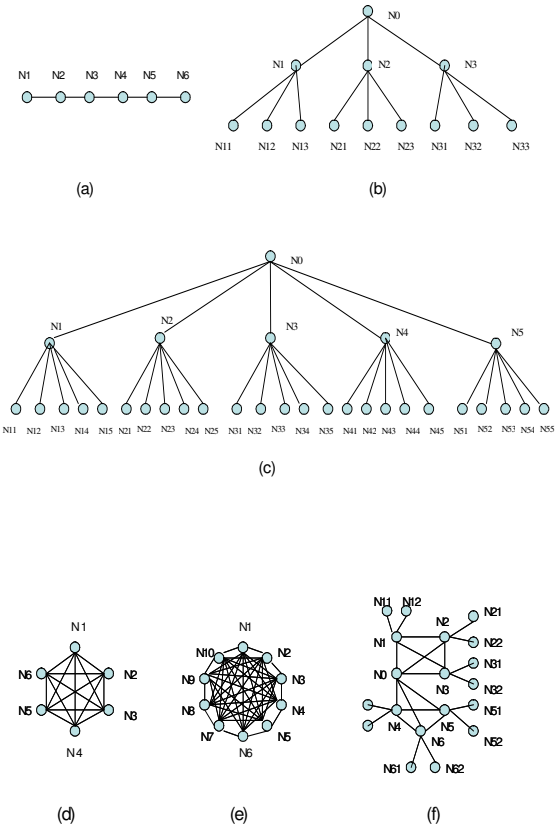

Fig. 3 A Chain, Tree, Fully Connected Graph, and Tree with Core Graph

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

Even within a dominant pattern graph class we tested instances with large number of sizes. The stochastic variation due to unpredictable network performance was not the principal focus of this study. Therefore, we avoided any

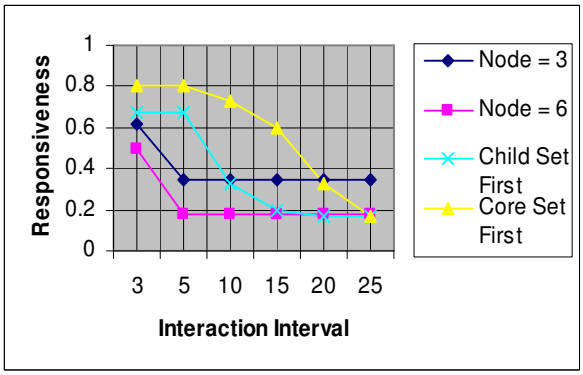

Fig. 4 Performance for Response Time in a Chain and a Tree with Core Graph

mass averaging, rather we present the performance based on specific pattern cases. To explain the patters for each class we included a sample sequence table.

The ultimate objective of any prefetch system is to reduce the user waiting time and increase systems responsiveness. On the other hand, the main cost factor is the amounts of unused data. Therefore, we present the impact on the two performance measures: 1) *response time*;

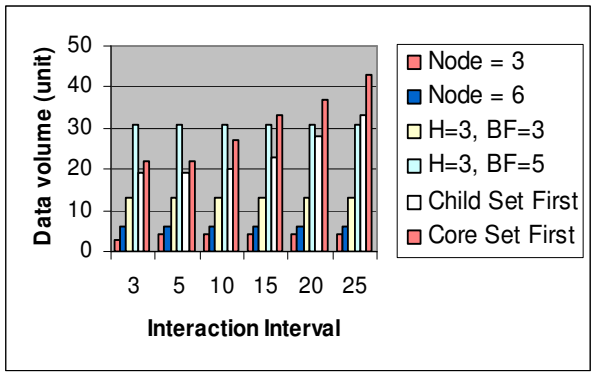

Fig. 5 Performance for Data Volume in a Chain, a Tree and a Tree with Core Graph

2) the amount of *data transfer*. The performance for response time is 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*. We also measured the

fetched data volume in both the cases and plotted the ratio. For each experiment, we chose 5 seconds, 10 seconds, 15 seconds, 20 seconds, and 25 seconds as five different groups of interaction interval.

5.1 Surfing a Chain

Fig. 3(a) shows a sample test hyper graph for Chain. In a chain, only one prefetch sequence is logical. For surfing sequence however, we conducted two experiments: *Half sequence* reading and *full sequence* reading.

1). Response Time Analysis: The typical performance for response time in chain is shown in 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 chain the surfer traverses, the more improvement of responsiveness performance we observed. We also found that there is a threshold interaction interval, beyond which the responsiveness becomes insensitive to it. This characteristic can be potentially used to design an optimum prefetch scheme which will not be overaggressive.

2) Data Volume Analysis: The performance for data volume in chain is shown in 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 extra data volume is one unit.

Type	Node	Prefetching Sequence	
		Left First	Right First
H = 3 BF = 5	0	1,2,3,4,5	5,4,3,2,1
	1	11,12,13,14,15	15,14,13,12,11
	2	21,22,23,24,25	25,24,23,22,21
	3	31,32,33,34,35	35,34,33,32,31
	4	41,42,43,44,45	45,44,43,42,41
	5	51,52,53,54,55	55,54,53,52,51
H =3 BF = 3	0	1,2,3	3,2,1
	1	11,12,13	13,12,11
	2	21,22,23	23,22,21
	3	31,32,33	33,32,31

Table 1 Lists of Prefetching Sequences in a Tree.

5.2 Surfing a Tree

For tree experiment we generated collections with various heights and breadths. For example, 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 used two prefetching sequences 1) *Left First* and 2) *Right First*. These are shown in table 1.

To test various surfing behavior within a tree reading, we let the surfer use three different surfing sequences: 1) *Depth First* (α), 2) *Breadth Firs t*(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, avoiding symmetrical cases. The sample node sequences for various runs for the sample graph are shown in table 2.

1). Response Time Analysis: The performance for response time in a tree with Left First and Right First as prefetching sequence are shown in Fig. 6 and Fig. 7 respectively. We observed that the improvement in responsiveness is the best when Web documents are read 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, then these are 0.6-0.7 times less.

Type	Surfing sequence		
	Depth First	Breadth First	Random
H = 3 BF = 5	0,1,11,12,13,14, 15,2,21,22,23, 24,25,3,31,32,33 ,34,35,4,41,42,4 3,44,45,5,51,52, 53,54, 55	0, 1, 2, 3, 4, 5, 11, 12, 13, 14, 15, 21, 22, 23, 24, 25, 31, 32, 33, 34, 35, 41, 42, 43, 44, 45, 51, 52, 53, 54, 55	0, 4, 41, 42, 2, 21, 22, 23, 24, 25, 3, 33, 31, 32, 34, 35, 1, 5, 52, 53, 54, 55, 51, 43, 44, 45, 11, 12, 13, 14, 15
H = 3 BF = 3	0, 1, 11,12,13,2, 21,22,23,3,31, 32, 33	0, 1, 2, 3, 11, 12, 13, 21, 22, 23, 31, 32, 33	0, 1, 2, 3, 11, 12, 13, 21, 22, 23, 31, 32, 33

Table 2 Lists of Surfing sequences in a Tree forth

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 value of relative responsiveness decreases gradually. The reason is the more interaction interval there is, the more prefetching is occurs. So the more improvements of performance are acquired.

2). Data Volume Analysis: As shown in Fig. 5 for this case, data volume is not significantly affected by prefetching sequence and surfing sequence. The total amount of transferring data is the same as without prefetching. When the branching factor is 5, data volume is 31 units; when the branching factor is 3, data volume is 13

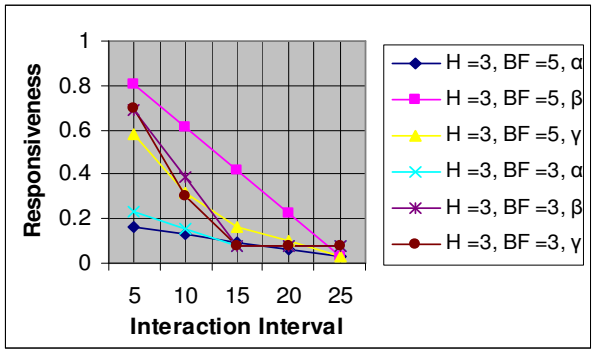

Fig. 6 Performance for Response Time in Tree with Left First (α– Depth First, β– Breadth First, γ– Random Connected

units. Therefore, a case where excessive overload is a concern the prefetch proxy should limit the branching factor allowable for prefetch.

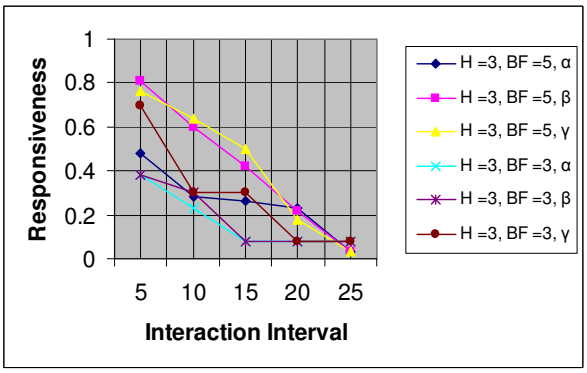

Fig. 7 Performance for Response Time in Tree with Right First (α– Depth First, β– Breadth First, γ– Random Connected Walk)

5.3 Suring a Complete Graph

Complete graph can vary with respect to the size of the clique. Two examples test graphs with 6 and 10 nodes are

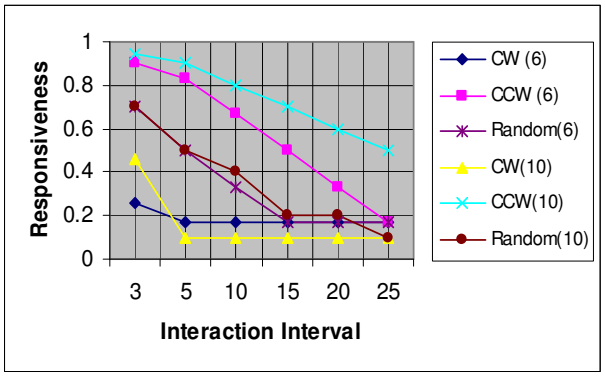

Fig. 8 Performance for Response Time in Complete Graph

shown in Fig. 3(d), and 3(e), with 6 and 9 hyperlinked nodes respectively. They are a complete graph. We consider only one case of *clockwise* prefetch sequence (anticlockwise prefetch creates symmetrical cases). With respect to it, we consider three types of surfing sequences. These are 1) *Clockwise (CW)*, 2) *Counter Clockwise (CCW)*, and 3) *Random Walk (RW)*. These walks are shown in Table 3.

Total Nodes	Surfing sequence		
	Clockwise	Counter Clockwise	Random Walk
6	1,2,3,4, 5,6	1,6,5,4, 3,2	1,4,6,2, 5,3
10	1,2,3,4, 5,6,7,8 9,10	1,10,9,8, 7,6,5,4,3,2	1,6,3,5,9,7,2,8,4,10

Table 3 Lists of Surfing sequences in a Graph

1). Response Time Analysis: In Fig. 8, 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 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 is shown in Fig. 9. Different surfing sequences result in different data volume. The amount of data in clockwise reading direction is always less than that in counter clockwise. Basically data volume for any reading order always increases when interaction interval increases. All of them produce a lot of extra amount of data compared to amount of transferred data without prefetching. The more nodes we move through, the more extra amount of data is produced.

5.4 Surfing a Tree with Core Graph

Fig. 3(f) shows an example of test 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 a tree structure. Two types of prefetching sequences are selected. We call them 1) *Core Set First* and 2) *Child Set First*. See Table 3. We use one Depth First surfing sequence here.

1). Response Time Analysis: The performance for response time in Tree with Core is shown in 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.

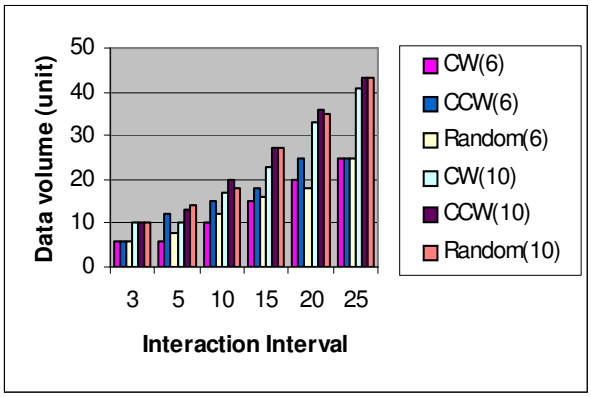

Fig. 9 Performance for Data Volume in Complete Graph

The responsiveness with Core Set First is up to 2 times less than that with Child Set First.

2). Data Volume Analysis: In Fig. 9, if Child Set First is selected as prefetching sequence, its performance 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.

Node	Prefetching Sequence		Surfing Sequence (Depth First)
	Child Set First	Core Set First	
0	1,2,3,5,4,6	1,2,3,5,4,6	0,4,41,42, 6,61,62,5, 51,52,1,11,12,2,21,22,3,31, 32
1	11,12,2,3,0	2,3,0,11,12	
2	21,22,3,0,1	3,0,1,21,22	
3	31,32,0,1,2	0,1,2,31,32	
4	41,42,0,5,6	5,6,41, 42,0	
5	51,52,0,6,4	6,4,51, 52,0	
6	61,62,0,4,5	4,5,61, 62,0	

Table 4 Sequences in a Tree with Core Graph.

6. Conclusions and Future Works

First generation of prefetch techniques depend primarily on "frequency" of access analysis. In this paper, we have presented a study, which suggests that smarter prefetching techniques can be developed if the structure of webspace can also be brought into consideration. This observation is promising yet it is in its very early stage. The result show how dramatically the performance can vary based on organizations and reading pattern. A whole new class of organization aware intelligent web techniques can be potentially developed based of match mismatch of the two.

6.1 Author Driven Organization

Now clearly the question is if such a scheme realizable? User interest is clearly 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. The main technological hindrance is that current HTTP or HTML has no mechanism, 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. 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 enhanced that will encourage content developer to mark at least one or two dominant hyperlink(s) among the links s/he embeds. The effort may not be more than adding alternate text for embedded images. Quite often, the link importance is already known by the content author. Content author generally follows a premeditated theme based mental organization to hyperlink the collection.

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.

6.4 Other Issues

Any prefetch is expected to react with the underlying passive caching sub-system, and it should be subject to further study. The technique 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 dramatically 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, pp29-45, 1999

[3] E. Cohen and H. Kaplan. Prefetching the Means for Document Transfer: A New Approach for Reducing Web Latency. Proc. of the IEEE INFOCOM 2000, March 2000.

[4] 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.

[5] Javed I. Khan, Ordering Prefetch in Trees, Sequences and Graphs, Tech. Report 1999-12-03, Kent State University, [URL http://medianet.kent.edu/ technicalreports. html.

[6] Javed I. Khan, Qingping Tao, Prefetch scheduling for composite hypermedia, Proceedings of IEEE International Conference on Communications (ICC2001), Finland, pp 768-773, June 2001.

[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, pp13-24, March 2001.

[8] 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 11-15, 2002, pages 159-168.

Proceedings of the IEEE/WIC International Conference on Web Intelligence, WI'03, October 2003, Halifax, Canada, pp.579-583