Published in the Proceedings of the
IEEE 2003 International Conference on Cyberworlds, CYBERWORLDS2003,
Singapore, December 2003, pp478-485

Webspace Surfing Patterns and Their Impact on 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 presents an interesting study that how the user surfing behavior with respect to the organization of a web space affects the performance of a prefetch enabled proxy. We have conducted experiments based on a novel hyperspace aware prefetch proxy and have studied the prefetch performance on several dominant hyperspace patterns including chain, tree, and complete graph sub-structures. The study assesses the system's responsiveness as well as the background loads for various user interaction duration, surfing and prefetch sequences. The results show that awareness about the web space and surfing sequence can greatly help in improving prefetch performance. Schemes such as these also conversely may enable professional content developers to create prefetch friendly collection for increased site responsiveness.

Keywords: Prefetch, User Interaction Behavior, Web Surfing, Web Engineering

1. Introduction

Although the core network speed is doubled every 9-12 months, the sluggishness of Web seems to be a persistent problem. Seemingly, the growth in the last mile speed is being continually outpaced by the growth in the complexity of contents, which now often contain communication and computing intensive multimedia such as animation, audio, video, applets, and scripts, far beyond simple embedded images in the plain HTML. For last few years, researchers have begun to explore prefetch as one of the potential accelerator. Previously it has been found to be effective to hyper accelerate hardware systems. However, its success in Web system has been remained surprisingly illusive.

1.1 Related Works

In one of the pioneering studies, Kroeger et al. [KrLM97] used traces of Web proxy activity to find out that a combined caching and prefetching proxy could reduce access latency as much as 60%. Palpanas and Mendelzon [PaMe99] demonstrated that a k-order Markov predictor scheme can reduce response time by a factor of up to 2. Pitkow and Pirolli [PiPi99] compared various methods to predict surfer's path from log traces such as session time, frequency of clicks, Levenshtein Distance analyses, etc. These studies focused on the problem of link transition probability estimation. Paths were ranked in order of the estimated probabilities and once a path is selected the suggestions are to prefetch the entire documents. Such prefetch has been found to cause excessive wasted prefetch adversely affecting the overall network bandwidth.

A number of research since then has suspected the advantage of using indiscriminate frequency only prefetch technique [KaPJ99, Khan99, Khan00, Davi01]. Cohen and Kaplan [CoKa00] suggested just doing pre-staging- such as pre-establishing TCP connection to avoid the waste. Kaashoek [KaPJ99] found that 0.5 objects are prefetched for each object explicitly fetched by the user through tracing Web server. Among these prefetched objects, only about 2% are actually used. Khan [Khan99, Khan00] demonstrated that only frequency ranked prefetching is non-optimum. Brian D. Davison [Davi01] showed that the current support for prefetching in HTTP/1.1 is rather insufficient. Existing prefetching implementations can cause problems with undesirable side effects and server abuse, and the potential for these problems may thwart additional prefetching development and deployment. Most of the previously suggested prefetch techniques took a statistical approach to the problem. Almost all of the suggested schemes resorted to the *access frequency* as the principle beacon to guide the prefetch activities, though these varied in the method for its estimation and/or prediction. 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. More innovations, particularly which can add new intelligence in selecting the right prefetch link and reducing waste in prefetch data are clearly required.

1.2 Other Approaches

Recently, several other techniques based on new concepts have been proposed to remedy some of the problems mentioned. For example, to reduce unnecessary prefetch, in 2001, we [KhTa01a] suggested

Published in the Proceedings of the
IEEE 2003 International Conference on Cyberworlds, CYBERWORLDS2003,
Singapore, December 2003, pp478-485

using the concept of "partial prefetching". Most Web pages served by modern servers today are composite and contain embedded entities such as banners, Java applets, flash presentations, etc. with varying rendering constraints. We [KhTa01b] demonstrated that the composite multimedia pages with internal rendering dependencies could be optimally divided into two parts, the lead segments and the stream segments. During operation only the lead segments are needed to be prefetched. Rest can be streamed in background only while the link is in use without any loss of responsiveness. 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 [Davi02] examined textual similarity-based predictions, 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 that may accelerate the Web prefetch substantially. 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*. 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 of these. The intent of this paper is to shed some lights in this interesting void. The paper is an attempt to study how intelligence about the content organization in a collection and reading modes can improve the prefetch performance of a Web system.

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. Clearly, the open implementation of any such scheme will be challenging today as current standards have little or no support for any such scheme. However, in this paper we focus on the potential gain if such a scheme can be facilitated.The paper is organized in the following way. Section 2 first presents a discussion on the existence of *dominant regular structures* and then focuses on the modeling of the user access and interaction patterns. Section 3 presents the architecture of a client side proxy based on prefetch system that we have implemented for this study. Finally, section 4 presents the performance of prefetch evaluated in details and followed by the analysis.

2. Web Surfing Patterns

2.1. Dominant Pattern Graph (DPG)

Today 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, 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. It decides how prefetching can be implemented and how much data have to be involved.

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. 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 hyperlink 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. The principality of hyperlinks can be determined by design or traced back by frequency sorting.

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, 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. These Web pages have a chain structure. 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

Published in the Proceedings of the
IEEE 2003 International Conference on Cyberworlds, CYBERWORLDS2003,
Singapore, December 2003, pp478-485

page of the main page. A tree my 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.

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. This type of Web page usually relies on a frame set. 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 large and deep 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.

2.2 User Surfing 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 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 selected major patterns of surfing paths based on the graph type. The choices however, are related to the original organization of the document. These will be explained in the experiment section.

3. Recording Time for Implement Event

3.1 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

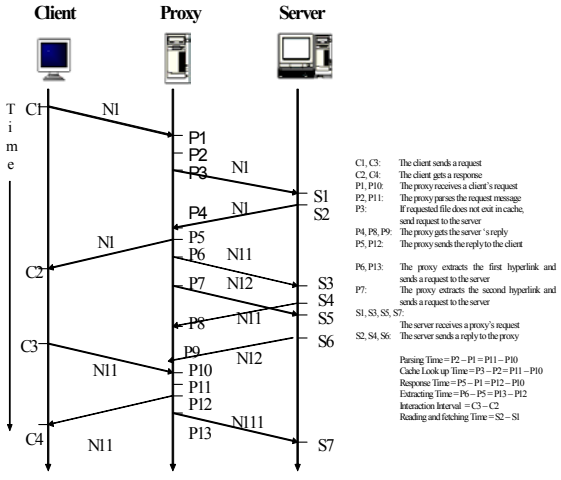


Fig. 1(a) Events Definitions and Time Distribution for Fully Folded Prefetching (FFP)

client, or can be at slightly deeper egress point serving multiple clients. In our set up, we used the later. For

Published in the Proceedings of the
IEEE 2003 International Conference on Cyberworlds, CYBERWORLDS2003,
Singapore, December 2003, pp478-485

performance analysis, in our system, we included the detailed time tracing code inside this proxy as well as in the Browser. We recorded time for all events happening at the client and the proxy as per the following event model.

3.2 Event Model & Logging

We used the following event model. Prefetch improves the response time in two ways. Fig. 1(a) shows the *fully folded prefetching* (FFP) and Fig. 1(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.

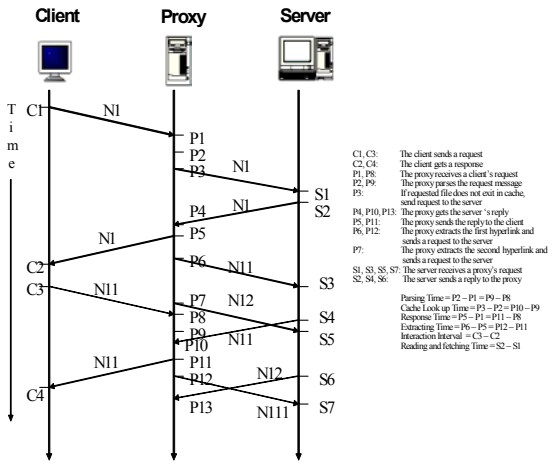


Fig. 1(b) Events Definitions and Time Distribution for Partially Folded Prefetching

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). In case of FFP (Fig. 1(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. 1(b)), N11 is not yet in the cache although request for it is already underway. Fig. 1(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.

3.3 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 a 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.

4. 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. The stochastic variation due to unpredictable network performance was not the principal focus of this study. Therefore, we avoided any mass averaging, rather we present the performance based on specific pattern cases. To explain

Published in the Proceedings of the
IEEE 2003 International Conference on Cyberworlds, CYBERWORLDS2003,
Singapore, December 2003, pp478-485

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. The main cost factor of any prefetch system is the amounts of data are fetched which are never used. Therefore, we present the impact on the two performance measures: 1) *response time*; 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 plotted the ratio. For each experiment, we separately recorded response time based on the different interaction interval. We chose 5 seconds, 10 seconds, 15 seconds, 20 seconds, and 25 seconds as five different groups of interaction interval.

4.1 Surfing a Tree

For tree experiment we generated collections with various heights and breadths. For example, in Fig. 2(a), 13 nodes are organized into a tree with three levels. Each of N0, N1, N2, N3, contains three hyperlinks. We used two prefetching sequences 1) *Left First* and 2) *Right First*. The actual node sequences are shown in table 1.

To test various surfing behavior, we let the surfer use three different surfing sequences: 1) *Depth First*, 2) *Breadth Firs t*, and 3) *Random Connected Walk*.

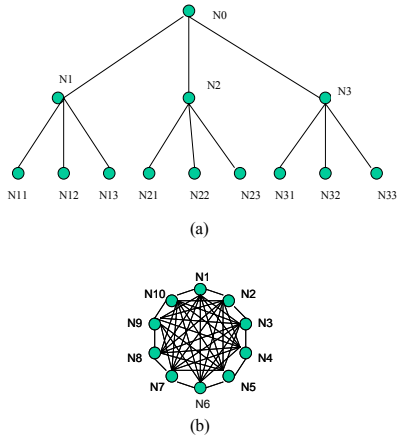


Fig. 2 Tree and Fully connected Graph

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.

Node	Prefetching Sequence	
	Left First	Right First
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 Tree.

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. 3 and Fig. 4 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. 3, when prefetching sequence is Left First, The responsiveness

with Random and Breadth First is up to 2.4 and 3.7

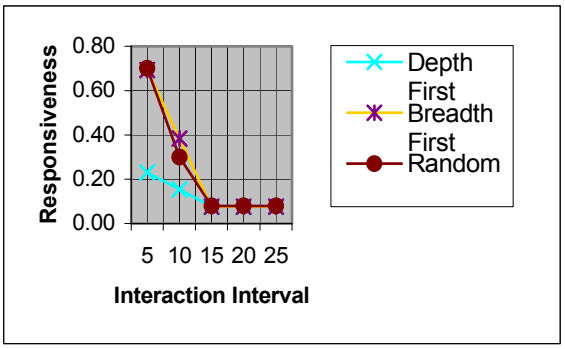


Fig. 3 Performance for Response Time in Tree with Left First

times less than that with Depth First respectively. In Fig. 4, 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.

Surfing sequence		
Depth First	Breadth First	Random
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 Tree forth

We also observed that no matter what the prefetching sequence is, 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.

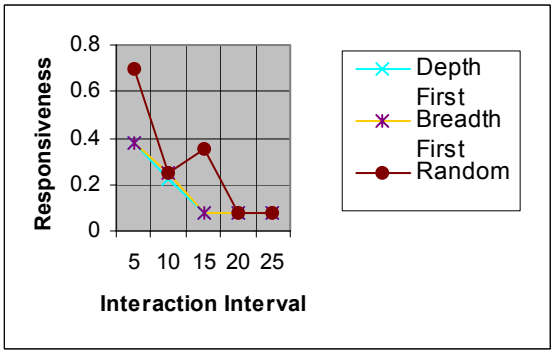


Fig. 4 Performance for Response Time in Tree with Right First

2). Data Volume Analysis:

The performance for data volume in a tree is shown in Fig. 6. Data volume is not affected by prefetching sequence and surfing sequence. Interaction interval does not affect its performance for data volume. The total amount of transferring data is the same as without prefetching. Data volume is 13 units.

4.2 Suring a Complete Graph

Fig. 2(b) shows a sample test hyper Complete graph. Each node contains 9 hyperlinks. They are a complete graph. We consider only one case of *clockwise* prefetch sequence (anticlockwise prefetch creates symmetrical cases). This is 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 for the sample graphs are shown in Table 4.

1). Response Time Analysis:

The performance for response time in a complete graph is shown in Fig. 5. 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

Node	Prefetching Sequence
	Clockwise
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
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

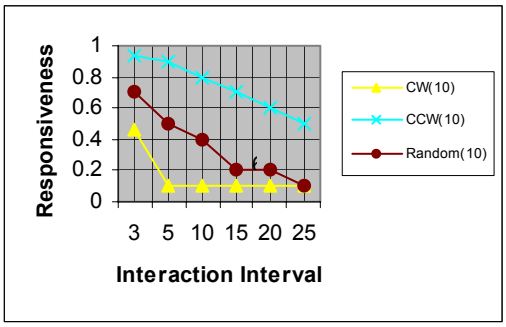
Table 3 Lists of Prefetching Sequences in a Complete Graph

Surfing sequence

Published in the Proceedings of the
IEEE 2003 International Conference on Cyberworlds, CYBERWORLDS2003,
Singapore, December 2003, pp478-485

Clockwise	Counter Clockwise	Random Walk
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



Error!Fig. 5 Performance for Response Time in Complete Graph

responsiveness increases in a gradual fashion.

2). Data Volume Analysis:

The performance for data volume in complete graph is shown in Fig. 6. Different surfing sequences result in different performance of 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 gradually when interaction interval increases gradually. 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.

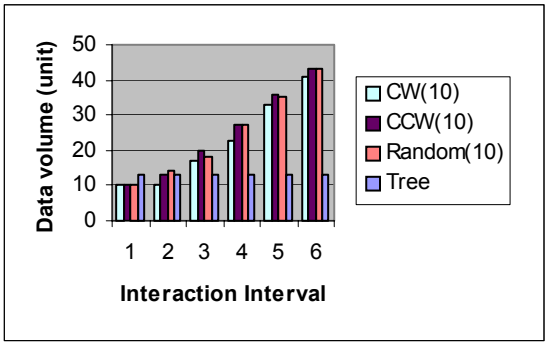


Fig. 6 Performance for Data Volume in Complete Graph

5. Conclusions and Future Works

First generation of prefetch techniques depend primarily on "frequency" of access analysis. In this paper, we have presented an interesting study, which suggests that smarter prefetching techniques can be developed if the structure of webspace and user reading behavior 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.

We observed the existence of dominant pattern graphs in the webspace particularly in large collections. The paper presents experiments on two types of abstract yet commonly occurring dominant patterns in hyperspace including tree and complete graph. Analysis suggests that, compared to a random prefetch system, the response time of a matched system (where the prefetch sequenced is matched with respect to the document organization) can be 1.6 times faster. Also, in the worst case, a completely mismatched system's response time can be about 1.7 - 11.3 times larger and result in 1.4 times more unnecessary data than a matched system.

5.1 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.

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**.

5.2 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 should 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. Also, the marking can be automatically generated by

many converters (such as PowerPoint® to HTML Converter).

5.3 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. A dominant chain can be discovered by simple modification of several currently available server tools and HTML chekers. 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.

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.

5.4 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.

Any prefetch is expected to react with the underlying passive caching sub-system. As a future work, one can study the impact of proxy cache size, media component classification, and discarding policies.

6. References

[CoKa00] 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.

[Davi01] Brian D. Davison. Assertion: Prefetching With GET Is Not Good. Proc. Of the 6th Int. Workshop on Web Caching and Content Distribution, June 20-22, 2001. [http://www.usenix.org/ events/ usits99].

[Davi02] 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.

[FMFM99] R. Fielding, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee Hypertesxt Transfer Protocal-HTTP/1.1. Tech. Rep. RFC 2616 (June), IETF. 1999. [http://www.ietf.org/rfc/rfc2616.txt]

[KaPJ99] 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.

[Khan99] Javed I. Khan, Ordering Prefetch in Trees, Sequences and Graphs, Technical Report 1999-12-03, Kent State University, [available at URL http://medianet.kent.edu/technicalreports.html, also mirrored at http://bristi.facnet.mcs.kent.edu/~javed/medianet.

[Khan00] Javed I. Khan, Active Streaming in Transport Delay Minimization, Workshop on Scalable Web Services, Toronto, pp95-102, August 2000.

[KhT01a] 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.

[KhT01b] Javed I. Khan, Qingping Tao, Prefetch scheduling for composite hypermedia, Proceedings of IEEE International Conference on Communications (ICC2001), Finland, pp 768-773, June 2001.

[KrLM97] T. Kroeger, D. D. E. Long & J. Mogul, Exploring the Bounds of Web Latency Reduction from Caching and Prefetching, Proc. of USENIX Symposium on Internet Technology and Systems, Monterey, December 1997, pp-319-328.

[PaMe99] T. Palpanas and A. Mendelzon, Web Prefetching Using Partial Match Prediction, WWW Caching Workshop, San Diego, CA, March 1999.

[PiPi99] 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