CHAPTER 3

WEB PRE-FETCHING TECHNIQUES

3.1 Introduction

Pre-fetching is a process by which the user's past browsing history is analyzed and the most visited pages are fetched earlier from internet and made ready to be used in the assumption that users regularly visit favorite pages. Although web caching enhances web performance, one way to further increase the cache-hit rate is to predict future document requests and pre-fetch these objects. If pre-fetching decision is taken by users, 100% accuracy of pre-fetching is guaranteed. It eliminates the extra network traffic usually created by pre-fetching [162]. Web pre-fetching technique utilizes the spatial locality of web objects. Even if caching and pre-fetching techniques are complement to each other in terms of locality of references, they can be integrated together for increasing cache hit ratio and reducing latency [163, 164].

This chapter gives an overview of web pre-fetching and surveys some of the current work in this area. Section 2 describes the characteristics of web request prediction. Different types of web pre-fetching techniques are discussed in section 3. Section 4 provides an enumeration of pre-fetching algorithms and discussion of existing pre-fetching techniques. Performance enhancing heuristics for improving the performance of pre-fetching techniques are discussed in section 5. The design issues of web pre-fetching techniques are discussed in section 6. Pre-fetching is
not widely used due to some limitations. These limitations of pre-fetching are discussed in section 7.

3.2 Web Request Prediction

Predicting the next item in a sequence is called sequence prediction. Sequence prediction is fundamental to pre-fetching and without accurate prediction, pre-fetching is not worthwhile [165]. A prediction method is embedded within a pre-fetching system for accurate prediction. If the actual request made matches the predicted request, it is considered a success. Predictions can be performed by the server, proxy or client itself [166]. Some research studies suggest to perform prediction at server because it is visited by large number of users and it has enough information about how they visit the sites. Some studies suggest that proxy servers can perform more accurate predictions because their user population is much more homogeneous than in an origin server. Some other studies suggest that predictions must be performed by the client browsers because it knows better user's preferences. Some studies suggest that different parts of the web architecture must collaborate when performing predictions.

Existing prediction models can be either “point-based” or “path-based”. “Point-based” prediction models use frequency measures to predict user's next requests. Users previous navigation path are used in “Path-based” models for leveraging a statistical analysis of user's actions [167]. A neural network based parallel algorithm is used for predicting the current branch and updating the computations for predicting the next branches [168]. This path based prediction algorithm provides better accuracy and high performance compared to
conventional predictors. Historically-based prediction approaches cannot predict an action previously never taken. In content based prediction, the textual contents of recently requested pages are used as a guide to the current interests of the user [169]. The factors determining the success of a pre-fetch scheme are accuracy, timeliness, coverage and overhead [170]. Accuracy is the prediction accuracy when predictions are made. A pre-fetch request is useless if it is issued too early or too late relative to the actual use of the data. Coverage means the pre-fetching scheme must be able to cover most of the load misses. The network overhead and pre-fetching effect [162] are given in equations 3.1 and 3.2.

\[
\text{Network overhead} = \frac{\text{Number of discarded URL}}{\text{Total number of pre-fetching trial}} \quad (3.1)
\]

\[
\text{Pre-fetching effect} = \frac{\text{Number of pre-fetching hit}}{\text{number of cache hit}} \quad (3.2)
\]

In pre-resolving technique, browser or proxy performs DNS lookup before issuing a request to the server thereby eliminating DNS query time from user perceived latency. In pre-connecting technique, browser or proxy establishes a TCP connection to a server prior to the user's request. It addresses connection establishment time on the first request utilizing a persistent connection. In Pre-warming technique, browser or proxy sends a dummy HTTP HEAD request to the server prior to the actual request. Pre-warming addresses start-of-session latency at the server. Pre-fetching combines these three techniques [171].

Evaluation of prediction accuracy can be done by either batch or online [165]. The traditional approach to machine learning evaluation is the batch approach. In batch approach, data sets are separated into distinct training and testing data sets. After training the system using training data set, testing data set is used for evaluating its performance. Online evaluation approach is more realistic and it
matches the expected implementation of a system that learns from the past to improve its predictions in the future. Web pre-fetching performance metrics are based on prediction, resource usage and latency [22]. Predictions are the amount of objects predicted and pre-fetches are the amount of objects pre-fetched. Good predictions are the amount of objects predicted that are subsequently demanded by the user and bad predictions are amount of predicted objects that do not result in good predictions. Pre-fetch hits are the amount of pre-fetched objects that are subsequently demanded by the user. Pre-fetch misses are the amount of pre-fetched objects never demanded by the user. Precision or accuracy measures the ratio of good predictions to the total number of predictions. Recall or pre-fetch hit ratio measures the percentage of requested objects that were previously pre-fetched. Increase of traffic due to unsuccessfully pre-fetched documents is called wasted bandwidth. Network traffic increase due to pre-fetch related information interchange is called network overhead. Prediction time is the time needed for making a prediction by the predictor. The equations for calculating precision and applicability [22] are given in equations (3.3) and (3.4).

\[\text{Precision} = \frac{\text{Number of good predictions}}{\text{Total Number of predictions}} \quad (3.3)\]

\[\text{Applicability} = \frac{\text{Total number of predictions}}{\text{Total number of requests}} \quad (3.4)\]

A session is a period of sustained web activity by a user. The latency of retrieving a web document depends on several factors such as network bandwidth, propagation time and speed of the server and client computers [172]. Studies show that percentage of external latency in total access latency is dominantly high [173]. An effective pre-fetching for clients based on reference
access information can reduce the external latencies. User Perceived Latency (UPL) is the delay that an end user actually experiences when requesting a web resource. The reduction of UPL does not imply the reduction of network traffic but it may increase network traffic [174]. Usually people browse their favorite pages and clicks anchor to jump linked pages. Web browsers fetch in-line images when displaying web pages. Web surfing is page oriented service. Mirroring is the process of storing multiple copies of web pages near to the client side. Mirroring reduces retrieval time. However, mirroring requires target selection manually. The target range of pre-fetching is wider than the range of both mirroring and caching [175] as in Table 3.1.

**Table 3.1 Comparison of Mirroring, Caching and Pre-fetching Techniques (from [175])**

<table>
<thead>
<tr>
<th>Type</th>
<th>Target Selection</th>
<th>Target Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mirroring</td>
<td>Manually</td>
<td>Narrow</td>
</tr>
<tr>
<td>Caching</td>
<td>Automatically</td>
<td>Narrow</td>
</tr>
<tr>
<td>Pre-fetching</td>
<td>Automatically</td>
<td>Wide</td>
</tr>
</tbody>
</table>

Brian D. Davison argues that the current support for pre-fetching in HTTP/1.1 is insufficient and extensions to HTTP are suggested in [176]. One suggestion is to use the HTTP header HEAD to get meta-information about the resource. That would allow the client to use some characteristics of the response to determine whether the retrieval is useful. Another suggestion is to incorporate a mechanism to support mandatory headers in HTTP that allow for pre-fetching [177].
Web surfing patterns and their impact on web pre-fetching are discussed in [31, 178] and results shown that awareness about the web space and surfing sequence can greatly help in improving pre-fetching performance. Wcol, PeakJet2000, NetAccelerator, Smiley are some of the pre-fetching systems in which the linked pages are pre-fetched from user retrieved pages for improving latency [179, 180]. Threshold of pre-fetching can be controlled based on ping RTT and network bandwidth measurement. Several tools are available for bandwidth measurement such as cprobe, pathchar, pchar, nettimer, tcpanaly [181].

3.3 Types of Pre-fetching Techniques

Web pre-fetching predicts the subsequent requests of a user based on the current request of users and fetches them in advance. Pre-fetching can either be "speculative" or "informed" [182]. Speculative pre-fetching means the knowledge about future accesses is not perfect in the prediction system. In informed pre-fetching, look ahead to future access is certain and the clients inform about its future requirements. Based on the location of information used for determining the objects to be pre-fetched, pre-fetching can be categorized into local and server-hint pre-fetching [38]. In local pre-fetching, browser-client or proxy uses local information to determine which objects to pre-fetch. In server-hint Pre-fetching, server is able to use its content specific knowledge of the objects requested and reference patterns from the clients, to determine which objects should be pre-fetched. Pre-fetching can be applied in three ways in the web environment [43] as given below.

- Between Clients and Web Servers
• Between Proxy servers and Web Servers
• Between Clients and Proxy Servers

3.3.1 Pre-fetching Between Clients and Web Servers

Pre-fetching between clients and web servers are useful for reducing the user perceived latency. Padmanabhan et al. analyze the latency reduction and network traffic of pre-fetching between clients and web servers [172]. Their study shows that pre-fetching from web servers to individual clients can reduce latency at the expense of increasing the network traffic. Bestavros et al. present a model for speculative dissemination of WWW documents [49]. They have shown that the reference patterns observed at a web server can be used as an effective source of information to drive pre-fetching. Pre-fetching between clients and web servers increases network traffic and wastes bandwidth due to unnecessary pre-fetching and it may cause network delays.

3.3.2 Pre-fetching Between proxy Servers and Web Servers

Proxy server is used in the most of web environments for sharing the Internet access and acting as a firewall. This type of pre-fetching relies on the proxy for making prediction table from access log. Kroeger et al. investigate the performance limits of pre-fetching between web servers and proxies and they have shown that combining caching and pre-fetching at the proxies can reduce the client latency at least by 60% for high bandwidth clients [38]. A proxy software was developed in [180] for pre-fetching documents, links and embedded images. Bandwidth waste can be reduced by configuring it to pre-fetch only a certain number of hyperlinks and images. Markatos et al. proposed a pushing
scheme in which web servers regularly push their most popular documents to proxies, then proxies push these documents to clients [183]. They evaluate the performance of the strategy and found that this technique can anticipate more than 40% of client requests while increasing only 10% of network traffic.

3.3.3 Pre-fetching Between Clients and Proxy Servers

Pre-fetching can be done between browser clients having low bandwidth and proxy servers. Fan et al. proposed an approach to reduce latency by pre-fetching between clients and proxy servers [184]. This method takes advantage of idle time between user requests to either push or pull the documents to the clients by anticipating the documents that a user might reference next. In proxy-initiated pre-fetching, proxy only initiates the pre-fetching of objects from its cache to clients and there is no extra internet traffic. Their simulation results show that pre-fetching combined with large browser cache and delta-compression can reduce client latency up to 23.4%. Loon et al. design a proxy system that performs pre-fetching, image filtering and hoarding documents based on usage profiles [185].

3.4 Existing Pre-fetching Techniques/Algorithms

Links to be pre-fetched can be determined either randomly choosing a constant number of links [186] or based on the probabilities of web access patterns from web logs data [187]. A popular method for prediction is to predict next URL based on present URL. Large enough web access logs can build an accurate prediction model. History based pre-fetching uses the log of popular documents from web servers for finding the documents to be pre-fetched. This
method is difficult to apply in reality, because all web servers don't have a popularity list [188]. Web pages can be divided into different components such as text and non-text items. Text only pre-fetches take less time to transfer and need only less storage space. But the browser needs to connect to web servers for fetching images [186]. In page rank-based pre-fetching approach, the most important linked document is identified from the link structure of a requested document [189].

Significant factor of a pre-fetching algorithm is its ability for deciding which object to be pre-fetched in advance to reduce latency [190]. Algorithms used nowadays are either blind or user-guided [191]. A blind pre-fetching technique does not keep track of any user web access history. It pre-fetches documents based on the static URL relationship in a document. User-guided pre-fetching techniques are based on user provided preference URL links. It maintains a record of user frequently accessed web sites to guide pre-fetching of related URLs. Several pre-fetching algorithms are proposed in [18, 163, 191, 192] for improving the performance of pre-fetching. Pre-fetching models are proposed in [193] for parallel web pre-fetching on cluster server. Various pre-fetching techniques/algorithms are discussed below.

### 3.4.1 Interactive Pre-fetching

In interactive pre-fetching scheme, linked pages in retrieved pages are fetched and in-line images in linked pages are also fetched. Using interactive pre-fetching scheme, if the pre-fetching system collects all referenced pages contained in clients retrieved web pages, 69% of hit rate is possible [194]. The pre-fetching
system increases the load onto the host. Since this method pre-fetch all the hyperlinks found in web pages, it requires a lot of memory or disk spaces. The basic algorithm for interactive pre-fetching scheme [175] is given in Table 3.2.

Table 3.2 Interactive Pre-fetching algorithm (from [175])

<table>
<thead>
<tr>
<th>Input: base_page (the page of browser's requested)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Begin</td>
</tr>
<tr>
<td>anchor_set := SeekAnchor(base_page);</td>
</tr>
<tr>
<td>foreach A in anchor_set do</td>
</tr>
<tr>
<td>begin</td>
</tr>
<tr>
<td>Get(A);</td>
</tr>
<tr>
<td>included_set := SeekIncluded(A);</td>
</tr>
<tr>
<td>foreach I in included_set do</td>
</tr>
<tr>
<td>begin</td>
</tr>
<tr>
<td>Get(I);</td>
</tr>
<tr>
<td>End;</td>
</tr>
<tr>
<td>End;</td>
</tr>
<tr>
<td>End.</td>
</tr>
</tbody>
</table>

3.4.2 Link Pre-fetching

Link pre-fetching is a browser mechanism, which utilizes browser idle time to download documents that the user might visit in the near future. A web page provides a set of pre-fetching hints to the browser, and after the browser finishes loading the page, it starts pre-fetching specified documents and stores them in its cache. When the user visits one of the pre-fetched documents, it can be served up quickly out of the browser's cache. Fisher et al. devised a server driven approach for link pre-fetching [195]. In this approach, browser follows special directives
from the web server or proxy server that instructs it to pre-fetch specific documents. This mechanism allows servers to control the contents to be pre-fetched by the browser. The browser looks for either HTML <link> tag or an HTTP Link: header tag to pre-fetch the subsequent links. The Link: header can also be specified within the HTML document itself by using a HTML <meta> tag [55]. The browser observes these hints and queues up each unique request to be pre-fetched when the browser is idle.

3.4.3 Top-10 Approach

In Top-10 approach, the server periodically calculates the list of most popular documents [183]. The web servers periodically push their most popular documents to web proxies, which then push those documents to the clients. Top-10 approach is simple and easy to implement in client server architecture. It utilizes the server's active knowledge of their most popular documents. It does not give careful consideration to the client characteristics on the Web. For prediction, it collects only documents accessed frequently without analyzing the characteristics of requests.

3.4.4 Domain-Top Approach

The proxy's active knowledge of their most popular domains and documents with client access profiles are combined in the domain-top approach (DT). Proxy side is responsible for periodically calculating popular domains and popular documents in each domain. In this approach, proxy first finds popular domains then searches for popular documents in each domain. Based on the proxy's active knowledge of popular domains and documents, a rank list is prepared for
anticipating the client's future requests [33]. In this approach, pre-fetching occurs only in the proxy server.

Domain Top with Classification (DTC) method tries to improve DT pre-fetching by removing unreliable logs [196]. Figure 3.1 shows the system architecture of DTC. The preprocessing module and analyzing modules are used for deciding popular domains and popular pages. They are off-line processes and works only when proxy server is in leisure periods. Pre-fetching module is an on-line process to decide whether a page should be pre-fetched or not. DTC method tries to provide suitable numbers of pre-fetching pages for different domains.

![Figure 3.1 DTC System Architecture (Redrawn from [196])](image)

3.4.5 Access Tree and Association relation Methods

A history structure is maintained in the access tree for identifying the information about access order [33]. Access-Tree approach uses the history structure as a forest of trees of a fixed depth. The tree structure is updated almost
every time when client makes a request. The proxy constructs prediction table based on a heuristic method. Observing a stream of requests, proxy can estimate the dependencies between resources and compute the probabilities of events that resources are requested together. In this scheme, it is difficult for the proxy to manage and maintain the history structure. Access-Tree approach is so deterministic, but complex and difficult to implement. Association relation method maintains a graph for identifying the web content requests of the user [188]. Maintaining the association relation to the graph becomes a burden to the cache servers.

3.4.6 Semantic Pre-fetching

In semantics-based pre-fetching technique, prediction of future request is based on preferences of past retrieved documents in semantics [197, 198]. A semantic link pre-fetcher is proposed in [199], it utilizes information associated with the current web page's hyperlink set to predict pre-fetching web objects. Semantic pre-fetching techniques tend to capture the client surfing interest from users past access patterns and predict future preferences from a list of objects when a new page is visited. Semantic knowledge of web documents are automatically extracted and adaptive semantic nets are constructed between web documents.

The architecture of semantic pre-fetching scheme is shown in Figure 3.2. Agent at client side attempts to understand semantic knowledge from surfing activities and finds out semantic relationships between the documents. Predictor makes prediction about client future requests from semantic knowledge and
steals network bandwidth to pre-fetch web documents from the Internet. The pre-fetched documents are temporarily stored in the proxy’s cache. A neural net based predictive pre-fetching to tolerate latency is discussed in [190]. It relies on keywords in anchor texts of URLs to characterize user access patterns and on neural networks over the keyword set to predict future requests. Anchor text refers to the text that surrounds hyperlink definitions in web pages.

![Figure 3.2 Architecture of Semantic Pre-fetching (Taken from [198])](image)

### 3.4.7 Markov Model for Web Access Prediction

Markov models and hidden Markov models (HMM) are used for predicting web pages based on the probabilities of web access patterns [200, 201]. Traditional Markov models predict the next web page, a user will most likely access by matching the user’s current access sequence with the user’s historical web access sequences [201]. The 0-order, 1-order, ..., k-order Markov models are available for predicting the next page to be pre-fetched. The Markov chain is
build by concatenating hyperlinks. The first order Markov processor includes only
the data relative to one step forward event. The successive order Markov
processes includes information related to successive events. Since low order
Markov models do not look enough into the past for correctly discriminate user's
behavioral models, higher models are required for good predictions.

The hyperlinks in a page may direct to a set of another linked pages. Each of
such linked pages will be labeled with the "first order" label. Each hyperlink
founded in first order labeled pages is labeled as "second order". The successive
hyperlinks are labeled as "third order" and so on [202]. Each labeled page is
associated with a data structure storing statistical data, updated at each browser
access. The data structure keeps the frequency of accesses of the clients to each
page as conditional probabilities of transitions from one page to another.

Markov pre-fetching model can achieve a good hit ratio with reducing the
traffic load. Accuracy of prediction is improved in high order Markov models, but
it increases the cost and complexity. One drawback of higher order models for
prediction is that their size grows rapidly with their order. A tree-like Markov
model (TMM) is used for solving this problem by creating a multiplayer tree from
a historical web access sequence database where each node is a visited page and
each branch is the path to the node. Each node records a count indicating number
of times the user has visited the node along the same route. A K-order TMM can
register all web access information. Hybrid-order tree-like Markov model
(HTMM) intelligently merges TMM and a hybrid–order method that combines
varying order Markov models [201]. HTMM provides high coverage and good
scalability. An intelligent web pre-fetching based on Hidden Markov Model is proposed in [203] for making semantic-based pre-fetching decisions.

3.4.8 Prediction by Partial Match (PPM)

PPM model makes pre-fetching decisions by reviewing the URLs that clients on a particular server have accessed over some period [204]. PPM algorithm has three parameters: order, depth and threshold. Order is the length of the history substring that the algorithm uses to find a match. Depth is the number of accesses into the future the algorithm attempts to predict. Threshold is the minimum probability an access must have in order to be considered a pre-fetch candidate. The standard PPM model uses multiple Markov models to store historical URL paths [205]. Client sessions are represented by each Markov model. It uses any URL for a root node and records every subsequent URL in the tree rooted by the first URL. Every node represents an accessed URL in the server. A counter records the number of times the URL occurs in the path from the root node. Every path from every root node to a leaf node represents the URL sequence. The advantage of the standard PPM model is that it is easy to build and not very complex. Some variations of the standard model attempt to fix the height of the tree.

3.4.9 Longest Repeated Sequence Model (LRS)

LRS model is a variation of PPM model, which keeps the longest repeating subsequences and stores only long branches with frequently accessed URL predictors [206]. A sequence of URLs that a client accesses more than once is considered a repeated sequence. A longest sequence is a frequently repeating
sequence in which at least one occurrence of one subsequence belongs to an independent access session. LRS model offers a lower storage requirement and high prediction accuracy. But the tree keeps only a small number of frequently accessed branches and ignores pre-fetching for many less frequently accessed URLs. Due to this, overall pre-fetch hits are reduced.

3.4.10 Popularity Based Prediction Model

This model is a variation of the PPM model that builds common surfing patterns and regularities into the Markov predictor tree [205]. The model assigns long branches to popular URLs and shorter branches to less popular URLs. URLs popularity is the number of times users access it in a given period. In this model, relative popularity is used for calculating the popularities for all URLs. This model also makes space optimizations to the completed tree. This model makes searching and pre-fetching highly objective and efficient.

3.4.11 Data Mining Algorithms for Web Pre-fetching

A simple WWW data model for web pre-fetching is proposed in [206]. This model is used for describing the interest association rules in the web pages. Intelligent agents are responsible for mining the user’s interest and pre-fetching web pages based on the interest association repository. The data in the user’s local cache can be taken as the data sources of data mining. The web pages are sorted and simple association rules are applied to predict the user’s activities. The data mining agent based web pre-fetching system is shown in Figure 3.3.
Yang et al. presented an application of web log mining to obtain web-document access patterns to extend the caching and pre-fetching policies [207]. Huang et al. developed an access sequence miner for mining web logs to improve hit ratios of pre-fetching and caching [208]. Rule-assisted pre-fetching (RAP) identifies a set of association rules from the web server's access log [209]. RAP's access miner analyzes and discovers access patterns in the form of association rules. A data mining algorithm is proposed in [210] for generalized web pre-fetching.

3.4.12 Model based Predictive Pre-fetching

This is an integrated web caching and web pre-fetching model, based on statistical correlation between web objects. Yang et al. developed an algorithm
and shown that trade off between latency reduction and increased network traffic is achieved [211]. This model addressed the issues of pre-fetching aggressiveness, replacement policy and increased network traffic. The key components in this integrated model are replacement manager, pre-fetching agent, prediction queue and cache. In this model, a cut-off value is defined to control the increased network traffic. The pre-fetching agent only pre-fetches objects whose probabilities are above the cut off value.

3.4.13 Web Pre-fetching Using Display Based Prediction

A client side pre-fetcher constructs link graphs by gathering usage information of a client in order to analyze the reference patterns and predict the next request based upon overall displayed documents in web browser. The link graph contains both nodes and links. The node indicates a HTML document with a unique URL. The link points from a referring document to a referred document (hyperlink or embed-link). The access counter of a node is increased by one whenever a user fetches it or follows it [162].

3.4.14 Adaptive Pre-fetching

Adaptive pre-fetching system is developed to adapt to the user's browsing habits and interests [212]. Jiang et al. proposed an adaptive pre-fetch scheme, in which the number of files to be pre-fetched is depends on the user access history and network conditions [213]. A prediction module and a threshold module are developed in adaptive pre-fetching for determining the number of pages to be pre-fetched for single user and multi-user systems. They derived a formula for the pre-fetch threshold based on system load, capacity and cost based on time and
system resources to the user. The Pre-fetch scheme consists of two modules: prediction module and threshold module. The prediction module updates the history information after a user's requests for a new file and access probability is computed for each file. Access probability is conditional probability that the file will be requested by the user in the near future. Files whose access probabilities exceed or equal to the server's pre-fetch threshold are only pre-fetched. An adaptive pre-fetching scheme is proposed in [214] that dynamically adjust the pre-fetching aggressiveness in web servers. It uses a threshold to adjust the aggressiveness of pre-fetching and prevents the web servers from being overloaded. An adaptive pre-fetching strategy is proposed in [215] for boosting the performance of search engines by exploiting the temporal and spatial locality present in the stream of processed queries.

3.4.15 Evolutionary Algorithm for Web Prediction

Evolutionary algorithm is proposed in [167] for evolving a population of prediction machines to provide a prediction model for user's next requests at each session. The architecture of the prediction system is shown in Figure 3.4, composed of two modules: an evolutionary algorithm and request predictor. A stream of user sessions evolves a population of prediction machines. The best prediction machine is made available to the request predictor. It is able to predict user actions on live sessions. The user navigational path behaviour is exploited for predicting future requests in real time. Real time user adaptation prevents the use of statistical techniques on web logs and a good prediction rate is achieved.
3.4.16 Non-interfering Pre-fetching

A non-interfering pre-fetching system (NPS) avoids interference by effectively utilizing spare resources on the servers and the network [216]. Self interference means a pre-fetching service hurts its own performance by interfering with its demand requests. Cross interference means the pre-fetching service hurts other applications on the pre-fetching client or other clients. NPS monitors server load externally and restricts the pre-fetch load imposed on it to avoid interference at the server. It uses a set of heuristics to control resource usage at the client. In NPS architecture, prediction and resource management are the two important tasks. Predictor prepares prioritized lists of high-valued documents for pre-fetching. Resource manager limits the number of documents to pre-fetch and schedules the pre-fetch requests to avoid interference.

3.4.17 Comparison of Web Pre-fetching Techniques

Since interactive pre-fetching method pre-fetch all the hyperlinks found in the web page, it requires lot of memory or disk space. In link pre-fetching, the
browser observes the hints from web server or proxy server and queues up each unique request to be pre-fetched when the browser is idle. This mechanism allows servers to control the contents to be pre-fetched by the browser. Top-10 approach utilizes the server's active knowledge of their most popular documents. This approach does not analyze the characteristics of client requests. In domain-top approach, proxy's active knowledge of popular domains and documents are considered for anticipating the client's future requests and pre-fetching occurs only in the proxy server. Domain-top with classification method removes unreliable logs to improve pre-fetch hit ratio. Access tree approach maintains history structure and computes the probabilities of events that resources are requested together. But it is difficult to maintain and manage the history structure. In association relation method, maintaining the graph became a burden to the cache servers.

Semantic pre-fetching utilizes information associated with the current web pages hyperlink set to find out semantic relationships between the documents and semantic knowledge of web documents are extracted for predicting web objects to pre-fetch. Traditional Markov models predict the next web page by matching the user's current access sequence with user's historical web access sequences. High order Markov models improve accuracy of prediction by increasing the cost and complexity. Hybrid order tree-like Markov models provide high coverage and good scalability. In prediction by Partial Match approach, any URL is used as root node and records every subsequent URL in the tree rooted by the first URL. The standard PPM model is easy to build and not very complex. Longest repeated sequence model stores only long branches and offers a lower storage requirement and high prediction accuracy. But this
approach ignores pre-fetching for many less frequently accessed URLs and overall pre-fetch hits are reduced. Popularity based prediction model assigns long branches to popular URLs and shorter branches to less popular URL's to make searching and pre-fetching highly objective and efficient.

In data mining approach, simple association rules are applied to predict the user's activities to improve hit ratios of pre-fetching and caching model. Model based predictive pre-fetching is an integrated web caching and pre-fetching model used for controlling increased network traffic by defining a cut off value. In this approach, objects whose probabilities above the cut off value are only pre-fetched. Gathering usage information of clients and constructing link graphs are difficult in web pre-fetching using display based prediction. In adaptive pre-fetching, a formula is used for determining the pre-fetch threshold based on system load, capacity and cost. In this approach, files whose access probability exceeds or equal to the server's pre-fetch threshold are only pre-fetched. In real time evolutionary algorithm for pre-fetching, user navigational path behaviour is exploited. This algorithm prevents the use of statistical techniques on web logs and a good prediction rate is achieved. Non interfering pre-fetching system avoids interference by effectively utilizing spare resources on the server and the network. In this approach, resource manager limits the number of documents to be pre-fetched and schedules the pre-fetch requests to avoid interference.

3.5 Performance Enhancing Heuristics

Performance enhancing heuristics given in [185, 217] are listed below in order to improve the efficiency of pre-fetching. The combination of these heuristics provides a remarkable performance improvement.
At the start of a session, pre-fetch more frequently accessed documents

- Periodically check the refreshes of the objects stored in cache
- Pre-fetching is better to perform only when the network is idle
- No need to store the log of dependents (embedded images) in the user profile or web log, because they are accessed automatically upon access of the document and they change frequently over time
- Even after the user specifies "stop" button, download that document in background and store it as pre-fetched document
- CGI and dynamic pages are pre-fetched and retained in the cache for a short period of time
- Use persistent connection. It reduces TCP open/close connection overhead by sharing a connection among multiple HTTP requests. Requests for inlined images in the page are serialized within the connection.
- Use pipelining for reducing the number of packets to be transmitted. In server initiative pipelining, server retrieves the requested page and parses it to extract inlined image reference and transmits them in the order of references. In client initiative pipelining, client sends requests to the server for retrieving the inlined images without waiting for a previously requested image to arrive.

### 3.6 Design issues in Pre-fetching

Design issues in pre-fetching are based on effectiveness of pre-fetching, traffic on the network, load on the servers and transparency [186]. Usage of multiple threads, pre-fetching algorithm and cache replacement strategy determines the effectiveness of pre-fetching. As the number of pre-fetching link increases, the
traffic on the network increases, network becoming congested and the web access becoming very slow. The number of pre-fetches made has to be limited and unnecessary requests must be kept at a minimum. Measuring the performance of the network connection and server are necessary for determining the number of pre-fetch requests to be sent. The excess traffic on the network and load on the server should be controlled and unnecessary pre-fetching should be avoided.

The pre-fetching mechanism should be running in the background. The user need not be completely ignorant of the browser's pre-fetching. The user has the freedom to set the sites to be pre-fetched or user's opinion/access patterns are important in pre-fetching. There is no harm in having the user be aware of pre-fetching occurring, but pre-fetching is used for improving performance and user interference is unnecessary. In the browser or proxy server, pre-fetching options menu is necessary for user to turn off pre-fetching completely and to set the size of pre-fetch area/cache. If the user notices degradation in performance due to pre-fetching, the user can stop pre-fetching by turn off pre-fetching. Pre-fetching should be stopped when the user clicks on a link and leaves that page.

Even if several pre-fetching algorithms are existing, main issues are concerned with the selection of optimized pre-fetching algorithm. The pre-fetching technique should not cause network overloading and congestion. Decision of selecting a pre-fetching algorithm came down to either a tradeoff or a compromise between simplicity and functionality. As the number of pages that are pre-fetched increases, the chances of the user picking one of the pre-fetched pages increase and the bandwidth usage increase. Increasing prediction accuracy substantially increases the CPU and memory overhead [218]. Since pre-fetching
increases the cache hit rate and the load on the host, balance between the load
and the hit rate is to be considered for better performance. Fast prediction is
critical for real-time performance [219].

3.7 Limitations of Pre-fetching

Although web pre-fetching is regarded as an effective method to improve client
access performance, the associated overhead prevents it from being widely
deployed [214]. Pre-fetching may increase network traffic and load on the
network are the main limitations of pre-fetching techniques. Safe response
header is proposed in [175] to tell the client that the request may be repeated
without harm, but this is only useful after the client already has the header
information about the resource. The details about server overhead, side effects of
retrieval, problems with pre-fetching etc. are discussed in [175]. The inefficiencies
could be avoided if the content provider were able to decide what and when to
pre-fetch.

Access patterns on the documents may change over time. Interferences can
occur at any of the critical resources in the system such as server, network or
client [216]. Pre-fetching consumes extra resources on the server such as
processing time, memory space and disk. Queuing delays and packet drops are
increased due to the extra data packets to be transmitted over the network for
pre-fetching. Pre-fetching results in extra processing at clients and aggressive
pre-fetching pollutes memory and disk caches.

Pre-fetching introduces an important trade-off between the use of system
resources and prediction accuracy [220]. If more resources are used and more
pages are pre-fetched then it is more probable that some of the pre-fetched pages to be accessed. Pre-fetching system may misuse or abuse the system resources for increasing prediction accuracy. Pre-fetching may wastes resources by pre-fetching uncacheable web pages. Pre-fetching is not widespread in the web today [176] due to the following reasons.

- Bandwidth overhead and wastage of resources due to pre-fetching data that never utilized.
- Potential increase of queue lengths and delays from native pre-fetching approaches.
- Increase of Internet traffic and server load due to pre-fetching by clients.

3.8 Concluding Remarks

This chapter has been an attempt to give an overview of web pre-fetching techniques. Web pre-fetching is a strategy to hide the network latency by pre-fetching the anticipating web pages. But some pre-fetched pages may not be eventually requested by the users. Even if several pre-fetching techniques/algorithms are available, none of the techniques have attained maximum benefits due to network overhead incurred by pre-fetching. The overhead due to memory space needed for storing the pre-fetched sites and CPU overhead for processing the prediction algorithms can be negligible. Further research is to be carried out for finding an efficient technique to pre-fetch the web pages by utilizing the idle time of the network without increasing the web traffic.