(12)

EUROPEAN PATENT SPECIFICATION

(45) Date of publication and mention of the grant of the patent: 05.04.2006 Bulletin 2006/14

(21) Application number: 98944132.4

(22) Date of filing: 28.09.1998

(19) Europäisches Patentamt

European Patent Office

Office européen des brevets

(11) EP 1 018 085 B1

(54) METHOD AND SYSTEM FOR PREFETCHING INFORMATION

VERFAHREN UND SYSTEM ZUM VORAUSLADEN VON INFORMATION

PROCEDE ET SYSTEME DE PRE-EXTRACTION DE DONNEES

(84) Designated Contracting States: BE CH DE ES FR GB IE IT LI NL SE


(43) Date of publication of application: 12.07.2000 Bulletin 2000/28

(73) Proprietor: International Business Machines Corporation

Armonk, NY 10504 (US)

(72) Inventors:
- MALKIN, Peter, Kenneth
  Ardsley, NY 10502 (US)
- YU, Philip, Shi-lung
  Chappaqua, NY 10514 (US)

(74) Representative: Davies, Simon Robert

IBM, United Kingdom Limited, Intellectual Property Law, Hursley Park

Winchester, Hampshire SO21 2JN (GB)

(51) Int Cl.: G06F 17/30 (2006.01)

(86) International application number: PCT/GB1998/002920

(87) International publication number: WO 1999/017227 (08.04.1999 Gazette 1999/14)

(56) References cited:


Note: Within nine months from the publication of the mention of the grant of the European patent, any person may give notice to the European Patent Office of opposition to the European patent granted. Notice of opposition shall be filed in a written reasoned statement. It shall not be deemed to have been filed until the opposition fee has been paid. (Art. 99(1) European Patent Convention).
Description

[0001] The present invention is related to a method and apparatus for the dynamic prefetching of information or objects for clients, wherein the clients are preferably associated with hierarchy of proxy servers, such as may be involved in prefetching Web objects on the World Wide Web.

GLOSSARY OF TERMS

[0002] Internet: The network of networks and gateways that use the TCP/IP suite of protocols.

[0003] Client: A client is a computer which issues commands to the server which performs the task associated with the command.

[0004] Server: Any computer that performs a task at the command of another computer is a server. A Web server typically supports one or more clients.

[0005] World Wide Web (WWW or Web): The Internet application that lets users seeking information on the Internet switch connection from server to server and database to database by choosing ("clicking on") highlighted words or phrases of interest (known as hyperlinks). An Internet WWW server supports clients and provides information to the clients. The Web, which can be considered as the Internet with all of the resources addressed as URLs, uses HTML to display the information corresponding to URLs, and provides a point-and-click interface to other URLs.

[0006] Universal Resource Locator (URL): The URL is the method to uniquely identify or address information on the Internet, and may be considered a Web document version of an e-mail address. URLs may be accessed with a Hyperlink. An example of a URL identification is "http://www.philipyu.com:80/table.html". The URL has four components. Starting from the left of the example, the first component specifies the protocol to use (in this case http), separated from the rest of the locator by a ":". The next component is the hostname or IP address of the target host; this component is delimited by the double slash "//" on the left and on the right by a single slash "/" (or optionally a ":"). The port number is an optional component, and is delimited on the left from the hostname by a ":" and on the right by a single slash "/". The fourth component is the actual file name or program name (in this example, table.html), using further single slashes to separate directories and subdirectories in conventional fashion. In this example, the ".html" extension means that this is an HTML file.

[0007] HyperText Markup Language (HTML): HTML is a language used by Web servers to create and connect documents that are viewed by Web clients. HTML generates Hypertext documents.

[0008] Hypertext transfer protocol (HTTP): HTTP is an example of a stateless protocol, in which every request from a client to a server is treated independently. The server has no record of previous connections. At the beginning of a URL, "http:" indicates the requesting client and target server should communicate using the HTTP protocol regarding the specified resource.

[0009] Internet Browser or Web browser: A graphical interface tool that runs Internet protocols such as HTTP, and display results on the user's screen. The browser can act as an Internet tour guide, complete with pictorial desktops, directories and search tools for when a user "surfs" the Internet. In this application the Web browser is a client service which communicates with the World Wide Web.

[0010] Client cache: Client caches are typically used as a primary group (caches) of objects accessed by the client. In the WWW environment, client caches are typically implemented by web browsers and may cache objects accessed during a current invocation, i.e., a non-persistent cache, or may cache objects across invocations.

[0011] Caching proxies: Specialized servers in a network which act as agents on the behalf of the client to locate an object, possibly returning a cached copy. Caching proxies typically serve as secondary or higher level caches, because they are invoked as a result of cache misses from client caches.

[0012] META-tagging: The association of information with a given object. For example, in HTTP, information can be associated with both requests and responses in the fields of the HTTP header. For example, an HTTP client can specify the URL from which a returned page was requested.

[0013] As the World Wide Web (WWW or Web) increases in popularity, traffic over the Internet has increased considerably. The Web has now become one of the primary bottlenecks on network performance. For example, a Web user may experience noticeable delays when documents or information are requested by the user who is connected to a server via a slow network link. The Web's transferring of the information over the network to the user further increases the level of traffic over the network. The increased traffic reduces the bandwidth available for the requests of other users, contributing to additional potential delays.

[0014] To reduce access latencies for clients, the prior art has stored or cached copies of popular documents of information closer to the user at network nodes, from which the access latencies are more acceptable. The caching can be implemented at various points on the network. For example, a large university or corporation may have its own local cache, from which all the users subscribing to that network may fetch documents. A local cache may be implemented in the form of a specialized server.

[0015] A specialized server, which is called a caching proxy, may act as an agent on the behalf of clients to locate any potential cached copy of requested information. Caching proxies usually serve as secondary or higher level caches because caching proxies are concerned only with misses left over from failed client caches. Client
Previous work on the caching of Web documents has mainly been in the area of on-demand caching. On-demand caching means a caching decision is made after an object is requested. For example, Abrams et. al., "Caching Proxies: Limitations and Potentials", Proc. 4th International World Wide Web Conference, 1996, describes different on-demand caching policies that have been examined for the Web.

"Prefetching in World Wide Web" by Wang and Cowcroft, IEEE GlobeCom 1996, discloses client-initiated prefetching in the World-Wide Web (WWW). It is suggested to use server initiated prefetching by having the server preload pages to the client, but this is regarded less favourably than client-initiated prefetching because of the need to modify both the server and the client. "Prefetching links on the WWW" by Jiang and Kleinrock, IEEE International Conference on Communications, 1997, discloses maintaining, preferably at the client, statistical information regarding data reference patterns, so that objects with a high likelihood of being referenced can be prefetched. "An Interactive Prefetching Proxy Server for Improvement of WWW Latency" by Chinen and Yamaguchi, INET 97, discloses a prefetching algorithm implemented in a proxy server, which on its own initiative prefetches pages that are referenced in requested pages.

In an Internet Service Provider (ISP) network or a corporate network, different regional proxies or departmental proxies in the proxy hierarchy may observe different reference behavior or face different network traffic and delay. The presence of the proxy hierarchy further complicates the prefetch decision on whether all the intermediate proxies, the client, or a subset of them should perform the prefetch. Thus, a need exists for a prefetching method and system that can effectively coordinate prefetching activities with respect to the proxy server hierarchy and associated clients.

Accordingly, the present invention provides a method as defined in the claims for prefetching information from a content server via a communications network, said communication network including one or more proxy servers and one or more clients, the method comprising the steps of: requesting, at a client, information from the content server via at least one proxy server; establishing, at the server, prefetch hint information denoting information likely to be requested by the client subsequent to said requested information, based on past retrievals or accesses by said clients; annotating, at the server, the requested information with said prefetch hint information and returning the requested information and the prefetch hint information together to the client via the at least one proxy server; and prefetching, at the client or said at least one proxy server, information based upon the prefetch hint information and the requested information, thereby making the prefetched information more readily accessible.

Typically, the requested information comprises requested objects, and the select information comprises prefetch candidate objects, such as Internet data. In this case the requested object can reference the select object through a hypertext transfer protocol (HTTP) link associated with the Internet data, the proxy server identifying the relationship of the select object to the requested object through the HTTP protocol of the requested object universal resource locator, or URL, stored in an object header of the requested object.

Prefetching activities with respect to the proxy server hierarchy and associated clients. Accordingly, the present invention provides a method as defined in the claims for prefetching information from a content server via a communications network, said communication network including one or more proxy servers. This can then be used to define the prefetch hint information. The steps of tracking and organising preferably include establishing databases for storing reference data patterns in the proxy servers; arranging fields within each of said databases, the fields including object identifiers and object counts, the object identifiers distinguishing the requested objects from one another; the object counts indicating the number of times the requested objects were requested as observed by one of said servers; and associating access lists with said databases, each of said access lists containing one or more prefetch candidate objects associated with each requested object.

Such a database can be used to define the relationship between requested objects and prefetch candidate objects and to track access statistics on the requested objects and the prefetch candidate objects; the access statistics including a parent-object access frequency of requested objects and a child-object access frequency of the prefetch candidate objects; the parent-object access frequency representing the number of times in which a particular one of the requested objects is requested by at least one of said clients; the child-object access frequency representing the number of times in which one or more of said prefetch candidate objects are actually requested proximately following the access-
Generally only a subset of prefetch candidate objects will actually be prefetched and received at one of said proxy servers or one or more clients, typically the selection of this subset of the prefetch candidate objects as select information being based on factors such as the available storage size at one of said proxy servers and/or the access time at one of said proxy servers. Thus the prefetch candidate objects are prefetched in accordance with the priority associated with some corresponding prefetch value.

In the preferred embodiment, the step of annotating the requested information comprises the step of placing the prefetch hint information in a prefetch label having a Platform for Internet Content Selection (PICS) format, or a label protocol compatible with the PICS format. In a particular implementation of this (PICS) format, an object identifier category and a referrer identifier category are defined, and said method further includes the step of returning the referrer identifier category to the content server to identify any prefetch candidate objects which were actually referenced. Organizing the prefetch hint information according to a Platform for Internet Content Selection protocol, or a PICS-compatible protocol utilises a format already supported by many browsers.

Typically there is a hierarchy of said proxy servers. Defining a sending node and a receiving node in this hierarchy, the prefetch hint information is preferably modified from an initial state to a final state, the initial state determined by the sending node, the final state determined by the receiving node. It is the prefetch hint information in said final state which then forms the basis for the ultimate prefetch decision (assuming that prefetching select information is initiated by the receiving node).

In other words, the prefetch hint information can be interpreted and updated as it is transmitted through the hierarchy from the proxy servers to the clients. Preferably a proxy server updates the prefetch hint information if local references to the requested information exist and if the count of the local-level server exceeds a minimum threshold count (i.e. reliable statistics are therefore available locally). The prefetch hint information can also be updated based on caching status in the proxy hierarchy (eg to indicate where in the hierarchy a particular prefetch candidate object may be cached).

Thus in a hierarchy of proxy servers containing at least one proxy server representing a sending node and at least one proxy server representing a receiving node, the prefetch hint information can be transmitted from the sending node to the receiving node; with the initial prefetch hint information being modified to final prefetch information, the initial prefetch hint information determined by the sending node, the final prefetch hint information determined by the receiving node. The final prefetch hint information is then used as the basis for the ultimate prefetch decision. Typically the step of prefetching information is initiated by the receiving node.

The invention further provides a data processing system as defined in the claims for prefetching information from a content server via a communications network, said communication network including one or more proxy servers and one or more clients, the system method comprising: means for requesting, at a client, information from the content server via at least one proxy server; means for establishing, at the server, prefetch hint information denoting information likely to be requested by the client subsequent to said requested information, based on past retrievals or accesses by said clients; means for annotating, at the server, the requested information with said prefetch hint information and returning the requested information and the prefetch hint information together to the client via the at least one proxy server; and means for prefetching, at the client or said at least one proxy, server information based upon the prefetch hint information and the requested information, thereby making the prefetched information more readily accessible.

In a preferred embodiment, the requested information comprises requested objects, and the select information comprises prefetch candidate objects, wherein each of said proxy servers and at least one of the clients has a memory having at least a logical separation between a main cache and a prefetch cache, wherein the prefetched select information is stored in the prefetch cache.

It is preferred that the prefetching means prefetches objects and stores the prefetched objects in the prefetch cache based on a priority scheme, the priority scheme storing objects with a highest prefetch value first, with regard to the available memory of the prefetch cache, the prefetch value being based upon an individual or aggregate reference probability to the prefetch objects by one or more clients. In particular, the prefetch value may be based on an estimated access probability that a prefetch candidate object on the prefetch hint information list will be accessed by particular ones of the clients and on an access cost of the object based on requisite storage size for storing select ones of the prefetch candidate objects.

In a preferred embodiment, the prefetching means comprises a prefetch handler including computing means for computing said prefetch value, and evaluating means for evaluating the prefetch value computed by the computing means; said computing means computing the prefetch value for each prefetch object identifier on the prefetch hint information list; said evaluating means evaluating the prefetch values of multiple candidate prefetch objects and prioritizing the caching of candidate prefetch objects associated with the highest prefetch values on the prefetch hint information list. The
prefetching means may comprise a first prefetch handler in at least one of said proxy servers and a second prefetch handler in at least one of said clients (i.e. prefetching may be performed at different/multiple locations in the hierarchy).

[0033] A preferred embodiment also includes a client object request handler stored in the memory of one of said clients, the client object request handler searching the main cache and prefetch cache for a requested object prior to requesting the requested object from a proxy server or the network, the client object request handler notifying the content server or proxy server if the object is located in the prefetch cache by generating a prefetch referenced message to the server indicative that the prefetch object was actually accessed by the client. Similarly it is preferred that at least one of said proxy servers includes a prefetch referenced message handler for forwarding a prefetch referenced message after invoking a cache manager routine if a requested object is located in the prefetch cache. This ensures that the server can maintain an accurate record of access statistics even if it is not directly contacted for a child object because it is available from a prefetch cache at the client or in a proxy server.

[0034] The invention further provides a content or proxy server as defined in the claims for use in the data processing system, wherein the requested information comprises requested objects, and wherein the prefetched information comprises prefetch candidate objects, said server including: a statistic table for tracking access statistics on the requested objects and the prefetch candidate objects; the access statistics including a parent-object access frequency of requested objects and a child-object access frequency of the prefetch candidate objects; the parent-object access frequency representing the number of times in which a particular one of the requested objects is requested by at least one of said clients; the child-object access frequency representing the number of times in which one or more of said prefetch candidate objects are actually requested proximately following the accessing of said particular one of the requested objects; and generating means for generating the prefetch hint information based on information in the statistics table.

[0035] In one embodiment, the generating means accepts inputs of first counts, second counts, child object identifiers, and parent object identifiers from the statistics table, the generating means including evaluating means for evaluating the access relationship between parent objects and their associated child objects, the evaluating means placing child object identifiers on a prefetch hint information list if the child objects are referenced with adequate frequencies proximately and subsequently prior to references by the clients to corresponding parent objects. The adequate frequencies can be based on the ratio of child accesses to corresponding parent accesses. Typically this involves calculating a ratio of the second count to the first count for each of said parent object identifiers; comparing the ratio to a hint threshold; and grouping or associating child objects and associated parent objects on the prefetch hint information list if the ratio exceeds the hint threshold.

[0036] Preferably the server further comprises statistics updating means for updating the statistics table based upon a child object actually requested; the statistics updating means first searching for an identity of a referring one of said parent objects, the referring one, or referrer object, being referenced prior to the child object actually requested, the child object actually requested having its corresponding second count updated in the statistics table. The statistics table should also be updated upon receipt of a prefetch referenced message from a prefetch referenced message handler.

[0037] It is preferred that where the server is a proxy server, it further comprises prefetch updating means for receiving and updating the prefetch hint information list. The prefetch updating means compares the first counts to first local threshold counts, and invokes the generating means for each of the child object identifiers for which the first counts exceed the first local threshold counts, the prefetch updating means keeping the original prefetch hint information list if the first counts do not exceed the first local threshold counts. In a more sophisticated approach, the prefetch updating means compares the first counts to second local threshold counts, and (i) keeps the original prefetch hint information list if the first counts do not exceed the first local threshold counts; (ii) modifies the original prefetch hint information list if the first counts exceed the first local threshold counts but not the second local threshold counts; and (iii) replaces the original prefetch hint information list if the first counts exceed both the first and second local threshold counts.

[0038] Preferably the server further includes: a proxy object handler for returning a requested object to a lower level proxy server or client that requested the requested object, the proxy object handler returning the requested object and updating the statistics table accordingly if the requested object is not a prefetch object; and a prefetch cache manager for managing a prefetch cache. Thus the proxy object handler returns a requested object from the prefetch cache to a requesting client or proxy server, and marks the object as deletable from the prefetch cache if the object was requested pursuant to a prefetch request. The proxy object handler sends an object request to another server if the requested object is not in a main cache or the prefetch cache. The proxy object handler updates the statistics table accordingly if the requested object is not associated with a prefetch request and if the requested object is located in the prefetch cache or the main cache.

[0039] Viewed from another aspect, the invention provides a method for prefetching select objects from a content server to make the select objects readily available for clients associated with one or more proxy servers via a communications network; the method comprising the
steps of:

a) requesting requested objects from at least one of said proxy servers;
b) identifying data reference patterns of corresponding clients based on the past retrievals or accesses of requested objects by the clients;
c) developing prefetch hint information into a prefetch hint information list from the data reference patterns;
d) annotating the requested objects with the prefetch hint information;
e) prefetching select objects from at least one of said servers based upon the prefetch hint information and currently requested objects.

[0040] Thus the select information which is prefetched comprises a prefetch candidate object; and the step establishing prefetch hint information further comprises establishing prefetch hint information at a sending node in a hierarchy of said proxy servers; where the prefetch candidate object is passed through a hierarchy of said proxy servers; and the select information to prefetch at a receiving node in the hierarchy is based on the prefetch hint information.

[0041] There is typically a proxy server hierarchy associated with the network, the proxy server hierarchy including a lower-level server more closely associated with the clients than other proxy servers. In one preferred embodiment, the prefetch hint information is updated through the proxy server hierarchy if local references to the requested object exist and if the count of the local-level server exceeds a minimum threshold count; alternatively (or additionally) the prefetch hint information may be updated by analyzing the caching status at higher level hierarchy, the highest level of the hierarchy being most closely associated with the network.

[0042] Viewed from another aspect, the invention provides a processing system for enhancing a flow of data from servers to clients along a network, the processing system comprising:

proxy servers associated with the network;
a content server associated with the network;
clients configured for communicating with one or more of said proxy servers;
a statistics table being stored in at least one of said proxy servers, the statistics table storing parent object identifiers and associated child object identifiers, the statistics table containing first counts of parent objects, the first counts representing the frequency in which parent objects are referenced by the clients, the statistics table containing second counts of child objects, the second counts representing the frequency in which the corresponding child objects are referenced after select ones of said parent objects are referenced; and

generating means for generating a prefetch hint information list by accepting inputs of the first counts and the corresponding second counts to derive the prefetch hint information list, said generating means including:

calculating means for calculating a ratio of the second count to the first count for each of said parent object identifiers;
comparing means for comparing the ratio to a hint threshold, the comparing means grouping or associating child objects and associated parent objects on the prefetch hint information list if the ratio exceeded the hint threshold; and
prefetching means for prefetching child objects based on the prefetch hint information and parent objects actually requested by the clients.

[0043] In a preferred embodiment, each of said proxy servers has a first processor, a first memory, and a first storage means for storing data; the first memory having at least a logical separation between a first main cache and a first prefetch cache; the content server has a second processor, a second memory, and a second storage means for storing data; and at least one of the clients has a third processor, a third memory, and a third storage means for storing data, the third memory having at least a logical separation between a second main cache and a second prefetch cache, the prefetch handling means storing child objects in the first prefetch cache or the second prefetch cache.

[0044] In a preferred embodiment, the prefetching means includes computing means for computing a prefetch value, and evaluating means for evaluating the prefetch value computed by the computing means; said computing means computing a prefetch value for each child object identifier on the prefetch hint information list and prioritizing the caching of child objects with the highest prefetch values on the prefetch hint information list, the prefetch handling means managing available space in the prefetch cache.

[0045] Preferably the system further includes: prefetch generating means for generating the prefetch hint information list; the prefetch generating means comparing the first counts to first local threshold counts and second local threshold counts, the first local threshold counts based on statistics from one server, the second local threshold counts based on statistics from another server, said prefetch generating means invoking the generating means for each of the child object identifiers for which the first counts exceeds both the first local threshold counts and the second local threshold counts; the prefetch generating means keeping the original prefetch hint information list if the first counts do not exceed the first local threshold counts.

[0046] It is preferred that such a system further includes:

statistics generating means for generating the statistics
a proxy object handler being stored in one of said proxy servers, the proxy object handler invoking a prefetch hint information list update routine, the proxy object handler returning a requested object to a lower level proxy server if the lower level proxy server requested the requested object, the proxy object handler returning the requested object and invoking the statistics updating means if the requested object is not a prefetch object;

a prefetch cache manager for managing a prefetch cache associated with one of said proxy servers, the proxy object handler invoking the prefetch cache manager if the lower level server did not request the requested object;

a proxy object request handler being stored one of said proxy servers, the proxy object request handler returning objects to a requesting one of said clients and marking the object as deletable from the prefetch cache if objects are in the prefetch cache and if the object was requested pursuant to a prefetch request; the proxy object request handler sending an object request to another server if the requested object is not in a main cache or the prefetch cache, the proxy object request handler communicating with the statistics updating means if the requested object is not associated with the prefetch request and if the requested object is located in the prefetch cache or the main cache.

[0047] It is also preferred that the system further includes:

statistics updating means for updating the statistics table;

and a prefetch referenced message handler being stored in one of said proxy servers, the prefetch referenced message handler invoking the statistics updating means, the prefetch referenced message handler forwards a prefetch referenced message after invoking a cache manager routine if a requested object is located in the prefetch cache.

[0048] The invention further provides a processing system for enhancing a flow of data from servers to clients along a network, the processing system comprising:

proxy servers associated with the network;

a content server associated with the network;

clients configured for communicating with one or more of said proxy servers;

a statistics table being stored in the content server and at least one of said proxy servers, the statistics table storing parent object identifiers and associated child object identifiers, the statistics table containing first counts of parent objects, the first counts representing the frequency in which parent objects are referenced by the clients, the statistics table containing second counts of child objects, the second counts representing the frequency in which the corresponding child objects are proximately, temporally referenced after select ones of said parent objects are referenced;
generating means accepting inputs of first counts, second counts, child object identifiers, and parent object identifiers from the statistics table, the generating means including evaluating means for evaluating the access relationship between parent objects and their associated child objects, the evaluating means placing child object identifiers on a prefetch hint list if the child objects are referenced with adequate frequencies proximately and subsequently prior to references by the clients to corresponding parent objects; and

prefetching means for prefetching child objects based on the prefetch hint information and parent objects actually requested by the clients.

[0049] In a preferred embodiment, the prefetching means further comprises assigning means for assigning storage priorities to child objects identified on the prefetch hint information list and selective storage means for storing select ones of the child objects in accordance with the storage priorities. The assigning means preferably assigns storage priorities based on an estimated access probability that a child object on the prefetch hint information list will be accessed by particular ones of the clients and an access cost of the child object based on requisite storage size for storing select ones of the child objects.

[0050] Thus information can be prefetched in a proxy hierarchy to reduce object access time through the network (i.e. Internet) for clients. The select information is prefetched from a content server to make the select information readily accessible to clients associated with a content server in a network. The clients are preferably associated with proxy servers in the network. The clients request information from at least one of the proxy servers. The content server generates prefetch hint information for the clients based on past data access patterns of the requested information by the clients. The content server annotates the requested information with prefetch hint information. One or more proxy servers may dynamically update the prefetch hint information based on dynamic usage statistics. The prefetch hint information is updated as the object passes through the proxy hierarchy to reflect prefetch operations performed, caching status at higher levels of the proxy hierarchy, and other local considerations, such as local reference patterns. Select information may be prefetched from the content server or any of the proxy servers based upon the prefetch hint information and the requested information. The prefetch hint information may be assigned a prefetch value based upon reference access probability and storage considerations. The select information with the highest prefetch values in the prefetch hint information list are cached first and
potentially until the prefetch cache is filled.

[0051] A preferred embodiment of the invention will now be described in detail by way of example only with reference to the following drawings:

FIG. 1 is a block diagram showing content servers (i.e. Web sites), a network (i.e. Internet), a proxy server hierarchy, and clients;

FIG. 2 is a further block diagram of a content server, a client, and a proxy server in greater detail than FIG. 1;

FIG. 3A is a flow chart diagram illustrating a method for dynamically prefetching data from a content server via a hierarchy of proxy servers;

FIG. 3B is an illustrative example of a statistics table;

FIG. 4 is a flow chart diagram illustrating operation of the content server;

FIG. 5 is a flow chart diagram illustrating statistics updating;

FIG. 6 is a flow chart diagram illustrating generating of prefetch hint information;

FIG. 7 is a flow chart diagram illustrating client operation;

FIG. 8 is a flow chart diagram illustrating prefetching;

FIG. 9 is a flow chart diagram illustrating operation of a client object request handler;

FIG. 10 is a flow chart diagram illustrating operation of the proxy server;

FIG. 11 is a flow chart diagram illustrating operation of a proxy object request handler;

FIG. 12 is a flow chart diagram illustrating operation of a proxy referenced message handler;

FIG. 13 is a flow chart diagram illustrating operation of a proxy object handler; and

FIG. 14 is a flow chart diagram illustrating operation of a prefetch hint information update routine.

[0052] FIG. 1 shows a data processing system that includes clients 22, a hierarchy of proxy servers 24, a network 25, and content servers 20.

[0053] The clients may request information in the form of parent objects, child objects, requested objects, select objects, pages, requested information and select information. Parent objects are related to child objects such that parent objects may have historically proceeded subsequent requests for one or more associated child objects. Requested objects are analogous to parent objects, while select objects are analogous to child objects. Requested objects are objects which are actually requested by one or more clients. Meanwhile, select objects are objects which are grouped with or related to the requested objects such that the select objects may be proximately, temporally requested after the requested objects. Requested information includes requested objects, parent objects, and other forms of data requested by the clients. Select information includes child objects, select objects, and other forms of data that may be prefetched.

[0054] The clients 22 may request information from the content servers 20 or proxy servers 24. However, the proxy servers 24 usually act as intermediaries for the clients 22 with respect to the network 25 and content servers 20. Advantageously, the proxy servers 24 may fulfill the clients information requests without contacting the content servers 20, if the proxy server hierarchy contains the requested information or has the ability to complete a requested task. The data processing system of FIG. 1 may be applied to the Internet. For example, the content server 20 may comprise a Web site, while the network 25 may comprise the Internet.

[0055] FIG. 1 illustrates logical connections 26, as opposed to physical connections, between elements of the data processing system. The logical connections 26 represent the information flow of objects and requests within the data processing system. The logical connections 26 do not typically represent physical connections between the proxy servers 24. The logical connections 26 can change due to workload and/or physical events such as a node or a communications link failure. Different types of objects may follow different logical paths of transmission to reach the clients 22.

[0056] While virtually any number of clients 22 and proxy servers 24 may be used in practice, the illustrative example of FIG. 1 features four clients 22 and four hierarchical levels of proxy servers 24. In practice, any number of levels in the proxy server hierarchy are permitted and the clients 22 can be logically connected to any level of the proxy server hierarchy. For example, the hierarchy of proxy servers 24 could be a single proxy server. The present invention may be practiced with merely a single client 22 and a single proxy server 24.

[0057] As shown, the highest level proxy server is designated as a level-zero proxy server 30. "Level-zero" is an arbitrary designation which merely defines the status of the one proxy server with respect to another proxy server in the illustrative example of FIG. 1. The highest level proxy server may be logically connected to the network 25. The higher level proxy servers 21 are more closely, logically associated with the network 25 than the lower level server proxies 23. (It will be appreciated therefore that in this terminology, which regards the level-zero proxy server as being the top of a proxy hierarchy, the level-zero is "higher" than level-one, which in turn is "higher" than level-two, and so on).

[0058] A first level-one proxy server 35 and a second level-one proxy server 37 are logically coupled to the level-zero proxy server 30. The first level-one proxy server 35 is connected to a first level-two proxy server 40 and a second level-two proxy server 43. The first level two proxy server 40 is logically coupled to a first level-three proxy server 50 and a second level-three proxy server 55.

[0059] The clients 22 include a first client 600, a second client 603, a third client 700, and a fourth client 703. The first client 600 and the second client 603 are logically coupled to a first level-three proxy server 50. The third client 700 and the fourth client 703 are logically coupled
to the second level-three proxy server 55.

[0060] The data processing system is now described from the viewpoint of the second client 603. The second client 603 is coupled to its immediately higher level proxy. As illustrated, the immediately higher level proxy is the first level-three proxy server 50. The second client 603 is then coupled to the first level-two proxy server 40, the first level-one proxy server 35, and the level-zero proxy server 30, respectively. The second client 603 can access various content servers 20 through the network 25. Although a typical communication path for the second client 603 is via the immediately higher level proxy (i.e. the first level-three proxy server 50), the second client 603 or a lower level proxy may communicate directly to other higher level proxy servers 24 or the network 25. The second client 603 may even have its own client proxy to manage caching.

[0061] From the viewpoint of second client 603, certain proxy servers are not part of its proxy hierarchy. For example, the second level-three proxy server 55 and the second level-two proxy server 43, and the second level-one proxy server 37 are not part of the second client’s proxy server hierarchy.

[0062] Clients 22 may include, but are not limited to, personal computers, workstations, set top boxes, or the like. The network 25 may include, but does not need to include, the Internet, the World Wide Web, an Intranet, a local area networks (LANs), or the like.

[0063] The proxy servers 24 and the content server 20 may comprise various commercially available computers. For example, proxy server 24 may be embodied as the Internet Connection Server (ICS) sold by IBM Corporation. Similarly, the content server 20 may be embodied as a Lotus Go Web server, a Lotus Domino server, or the like (also available from IBM Corporation). The proxy server 24 or content server 20 can run on any computing node, which includes, but is not limited to, products such as S/390 SYSPLEX, SP2, or RS6000 workstations (also available from IBM Corporation). In sum, the proxy server 24 or the content server 20 may comprise any general purpose computer and associated software which can provide World-Wide-Web page accesses, remote file transfers, electronic mail, and transaction support to clients 22.

[0064] The data processing system of FIG. 1 may use the attributes of a conventional proxy hierarchy to maintain data transfer efficiency. For instance, if a requested object is not available locally on a lower-level proxy server, a request for the non-available object is made to the next higher-level proxy. If the higher-level proxy has previously cached the object in its cache memory, the higher-level proxy will pass down the requested object to the client via the lower-level proxy server. Otherwise, the higher-level proxy server will try to obtain the object from its next higher level proxy. Once the requested object is obtained, it is passed down to the lower-level proxy or the client requesting the object.

[0065] FIG. 2 depicts a more detailed example of the data processing system depicted by FIG. 1. The data processing system includes the proxy server 24, the content server 20, the network 25, and the client 22. The data processing system preferably includes prefetching unit 250, statistics tables 261, statistics updating unit 252, main caches 297, and prefetch caches 298.

[0066] The prefetching unit 250 preferably includes a first prefetch handler 248 in the proxy server 24 and a second prefetch handler 241 in the client 22. Statistics tables 261 include a first statistics table 293 associated with the proxy server 24 and a second statistics table 267 associated with the content server 20. The statistics updating unit 252 includes a first update statistics unit 264 for updating the first statistics table 293 and a second update statistics unit 269 for updating the second statistics table 267. The main caches 297 include a first main cache 294 in each proxy server 24 and a second main cache 296 in each client 22. The prefetch caches 298 include a first prefetch cache 296 in the proxy server 24 and a second prefetch cache 249 in the client 22.

[0067] A proxy server 24 is a computing node that can serve requests through the network 25. The proxy server 24 includes a first processor 280, a first memory 290, and a first storage unit 281. The first memory 290 may comprise, for example, random access memory (RAM). The first storage unit 281 may comprise, for example, a direct access storage device (DASD). The first memory 290 stores a proxy server logic 295, which is embodied as a computer executable code. The proxy server logic 295 is loaded from the first storage unit 281 into the first memory 290 for execution by the first processor 280. Details of the operation of the proxy server logic 295 are shown in FIG. 10.

[0068] The proxy server logic 295 includes a proxy object request handler 283, a proxy object handler 284, the first prefetch handler 248, a prefetch referenced message handler 285, a prefetch updating means 291, the first update statistics routine 264, a first statistics table 293, a first main cache 294, and a first prefetch cache 296.

[0069] Details of the operation of the proxy object request handler 283 are depicted in FIG. 11. Details of the operation of the proxy object handler 284 are shown in FIG. 13. The operation of the prefetch referenced message handler 285 is shown in FIG. 12. Details on the operation of the statistics updating unit 252 (i.e. the first update statistics routine 264) are shown in FIG. 5.

[0070] The operation of the prefetching unit 250 (i.e. the first prefetch handler 248) is shown in detail in FIG. 8. The operation of the prefetch updating unit 291 or prefetch hint information update routine is shown in FIG. 14. The proxy server logic 295 maintains the first main cache 294 and the first prefetch cache 296.

[0071] The content server 20 is a computing node that can serve requests through the network 25. The content server 20 includes a second processor 260, second memory 263, and second storage unit 265. The second memory 263 may comprise, for example, random access
The second storage unit 265 may comprise, for example, a direct access storage device (DASD). The second memory 263 stores the content server logic 268 or content server software.

The operation of the content server logic 268 is shown subsequently in greater detail with reference to FIG. 4. The content server logic 268 is embodied as computer executable code which is loaded from the second storage unit 265 into the second memory 263 for execution by the second processor 260. The content server logic 268 includes a second statistics table 267, a second update statistics unit 269 for updating the second statistics table 267, and generating unit 266 for generating prefetch hint information. The second statistics table 267 stores data on information requested by the clients and potential prefetch information associated with the requested information. The statistics tables 261 contribute to the generation of prefetch hint information which is based on the actual reference behavior of clients. Details on the statistics tables 261 are illustrated in FIG. 3B.

Operation of the second update statistics unit 269 is shown in greater detail with reference to FIG. 5. The generating unit 266 generates prefetch hint information from the information in the second statistics table 267 and the requested information. The generating unit 266 may be embodied as prefetch hint information (PHI) generation routine as depicted in FIG. 6.

The client 22 preferably includes a third processor 240, a third memory 245 and a third storage unit 242. The third memory 245 may comprise, for example, random access memory. The third storage unit 242 may comprise, for example, a direct access storage device (DASD). The third memory 245 stores the client logic 244 or client software. The operation of client logic 244 is subsequently shown in greater detail with reference to FIG. 7. The client logic 244 is embodied as computer executable code which is loaded from the third storage means 242 into the third memory 245 for execution by the third processor 240.

The client logic 244 includes a client object request handler 247, the second prefetch handler 241, a second main cache 246, and a second prefetch cache 249. The client object request handler 247 obtains requested objects for the client from the second main cache 246, the second prefetch cache 249, a proxy server 24, or a content server 20. Details on the operation of client object request handler 247 are shown in FIG. 9.

The client logic 244 maintains the second main cache 246 and the second prefetch cache 249. The second main cache 246 is used for storing frequently referenced objects, while the second prefetch cache 249 is used for storing objects before they actually get referenced. Prefetch objects are objects that have not been proximately or previously referenced yet in conjunction with referenced objects. In the preferred embodiment, the main caches 297 are managed by the conventional LRU (least recently used) algorithm. Those skilled in the art will also appreciate that any other replacement strategy can be applied to maintain the main caches 297.

The main caches 297 and prefetch caches 298 may be logically separated. The logical separation of the prefetch objects from the actually referenced objects in the main cache 297 is to prevent the prefetch objects from taking too much storage space. Various alternatives are available to achieve this objective of a logically separated prefetch cache and main cache. For example, a single cache may have an upper limit imposed on the number of prefetch object and/or an upper limit imposed on the space occupied by the prefetch objects. The main cache 297 and the prefetch cache 298 are meant to support different goals. The main cache 297 is to cache popular objects which have the potential to be repeatedly referenced. The prefetch cache 298 is only used to support a particular browsing session by prefetching objects that are most likely to get referenced next in that session. Under the preferred embodiment, if a lower level node in the hierarchy issues a prefetch request on an object in the prefetch cache of a higher level node, the object will be passed down to the lower level node and be marked as deletable from the prefetch cache 298 of the higher level node since the objective of the prefetch has been accomplished. When a prefetch object actually gets referenced, it may be moved to the main cache 297 depending upon the management policy of the main cache 297. Under the preferred embodiment using an LRU replacement policy, a prefetch object once referenced is moved to the main cache.

An alternative criterion can be devised for admitting an object from the prefetch cache 298 into the main cache 297. In Abrams et al., “Caching Proxies: Limitations and Potentials”, Proc. 4th International World Wide Web Conference, 1996, some cache replacement policies have been discussed which do take object size into account in the decision making process. Other cache replacement strategies are known to one of ordinary skill in the art.

FIG. 3A is a flow chart diagram illustrating a method for dynamically prefetching data for clients from a content server 20 or a proxy server hierarchy. First, in block 100 a client requests information from a proxy server or a content server 20. Second, in block 102 historic data access or data reference patterns at the content server 20 and proxy servers are identified. In practice, the order of completing blocks 100 and block 102 may be reversed.

Third, in block 104 the content server 20 generates prefetch hint information (PHI) on related accesses to the requested information based on the data reference patterns identified in block 102. Fourth, in block 105 the prefetch hint information may annotate the requested information. For example, the prefetch hint information may be conveyed as meta data piggy-backed with the requested information (i.e. object). With respect to the step of establishing or generating prefetch hint information, content servers 20 track usage/reference patterns from a large number of concurrent users or clients. In
practice, the content servers 20 determine which group of objects are related to the requested information or requested object. The content servers 20 generate and provide a prefetch hint information (PHI) with the requested object. In the prefetching step or immediately prior thereto, proxy servers may interpret the PHI in light of local considerations to decide which select objects to prefetch. **[0081]** Fifth, in block 106 the prefetch hint information gets updated as the requested information passes through the proxy server hierarchy. The updating in block 106 reflects prefetch operations performed and caching status at the higher levels hierarchy, and other local considerations such as local reference patterns. The proxy servers may modify the PHI to reflect its prefetch decision and various local considerations before passing the object to the next (lower) level proxy or the client station. As an object passes through the proxy hierarchy, the sending node sets an initial PHI, while the receiving node may derive a final PHI from the initial PHI. The final PHI defines which select objects to prefetch based on the requested objects. The prefetch operation and decision is receiver-initiated, not sender-initiated. **[0082]** Finally, in block 108 select information in prefetched based on prefetch hint information and the requested information. The proxy server or client decides whether to prefetch a subset of objects specified in the PHI based on the PHI information in the received object, the locally maintained reference information (if available) and other local consideration such as available storage size, and access time. **[0083]** Generating prefetch hint information in block 104 preferably uses the PICS protocol or a PICS compatible format (see below). A PHI generated in block 104 is stored with each cached object as part of its meta data so that when a cached object is requested, the PHI information of the object can be examined to initiate the appropriate prefetch operations. The PHI information can be provided through the PICS protocol. The PICS protocol provides information on the criticality of each PHI object and its characteristics such as size, and caching status on the higher hierarchy. **[0084]** Updating the prefetch hint information may be accomplished through the proxy server hierarchy. The proxy servers may collect local information and distribute or decentralize the prefetch decision down the proxy hierarchy. Furthermore, the information of which intermediate proxy has prefetched an object could have implication on the prefetch decision of the lower level proxies and the client station in order to optimize the storage and bandwidth usage. **[0085]** The proxy servers in the proxy server hierarchy cooperate with each other to provide a receiver-oriented, prefetch decision. The prefetch decision may be made collaboratively by each proxy and client. Prefetch hint information in the format of meta information annotates (i.e. piggybacks) requested objects. The prefetch hint information provides information on potential prefetch candidates. Intermediate proxy servers may modify the meta information or prefetch hint information based on local conditions, and store the meta information with an object when it is cached, so prefetch can be triggered upon later reference to the object. **[0086]** FIG. 3B illustrates a data format of the statistics table 261. The statistics table 261 shall refer generally to either the first statistics table 293 or the second statistics table 267. One or more proxy servers 24 maintain the first statistics table 293. The content server 20 may maintain the second statistics table 267. The formats for the first statistics table 293 and the second statistics table 267 are preferably substantially similar or identical to promote updating of the statistics table 261 at different proxy servers in the proxy server hierarchy. The statistics table 261 helps provide prefetch hint information (PHI) based on actual viewer reference behavior from the viewpoint of the proxy server or the content server 20 with which the statistics table 261 is associated. The statistics table 261 is capable of observing the references of all of the communicative clients beneath it in the proxy server hierarchy. **[0087]** For example, in FIG. 1, the first statistics table 293 in the first level-three proxy server 50 tracks the aggregate data access behavior of the first client 600 and the second client 603. Meanwhile, the first statistics table 293 in the second level-three proxy server 55 tracks the aggregate data access behavior of the third client 700 and the fourth client 703. The first statistics table in the first level-two proxy server 40 tracks the aggregate behavior of the first client 600, the second client 603, the third client 700, and the fourth client 703. Although the lower level proxy, such as the first level-three proxy server 50, may be better able to track the interests of the client 600, the lower level proxy may not be able to get enough reference statistics from a sufficient sample size of clients to provide meaningful prefetch hint information (PHI). **[0088]** In the preferred embodiment, when an object O is requested from the content server 20, the server 20 provides the initial prefetch hint information PHI based on the information in its second statistics table 267. As the object O passes down the proxy hierarchy, any proxy server 24 along the path to the client 22 can modify the prefetch hint information (PHI) in the object header to tailor the PHI to the local situation in the first statistics table 293, if there are enough local references to the object O. The first statistics table 293 may supersede the second statistics table 267 based upon factors such as the well-indexed interests or preferences of a particular client by a lower level proxy server. **[0089]** In the preferred embodiment, the content server 20 maintains a second statistics table 267, which is a database of data access frequencies of the clients 22 with respect to requested information. The database could be expanded to include data access frequencies of requesting proxy servers 24. Similarly, one or more proxy servers 24 maintain a first statistics table 293. The statistics table 261, for example, has a first array 383 with
As shown, the first data structure 304 and the second data structure 306 are records. Each record in the statistics table 261 consists of several fields, including parent object identifiers 301 (i.e. ObjectId), first counts 302 (i.e. TCount), and access list data pointers 303 (i.e. AList).

[0090] The parent object identifier field 301 contains the object identifier. For example, the parent object identifier 301 contains the URL of the object for Internet or related applications. The first count field 302 contains the number of times the parent object, associated with the parent object identifier, is referenced as observed by the server. That is, the first count field 302 contains the frequencies of actual requests for requested objects. The first count 302 represents a parent-object access frequency. The access list data pointer 303 contains a data pointer to an access list 370.

[0091] The statistics table 261 has an access list 370, which, for example, contains a first access record 340, a second access record 350, a third access record 360, and a fourth access record 380. As illustrated, the parent object identifier associated with the first data structure 304 has an access list 370 which consists of a linked list of the first access record 340, the second access record 350, and the third access record 360. The next record data pointer field 313 contains a pointer to the next access record on the access list 370 associated with a particular parent object identifier 301. The access list for parent object identifier 301 associated with the second data structure 306 only contains the fourth access record 380.

[0092] Each record or data structure on the access list consists of several fields, including a child object identifier 310 (i.e. AObjectId), a second count 312 (i.e. RCount), and a next record data pointer 313 (i.e. Nxt). The data structures may be in the format of general data structure, record, or arrays. As illustrated in FIG. 4, the data structures are records which form a linked list within the access list 370.

[0093] In an alternative embodiment, the data format of the statistics table may include a primary array related to a plurality of alternate secondary arrays in a tree structure. The primary array would contain information related to the parent objects and the secondary array would contain information on the child objects of corresponding parent objects.

[0094] The child object identifier field 310 contains the object identifier (URL) of an object. The parent object identifier 301 is related to one or more child object identifiers 310 on the access list. The child object identifiers 310 identify the potential prefetch candidates if the child’s parent object is actually referenced. The second count 312 contains the number of times the child object is referenced through its parent object. In other words, the second count is the frequency in which the child object is referenced subsequently, temporally and proximately to the prior reference of an associated parent object. The second count 312 represents a child object-access frequency.

[0095] The child object identifiers 310 on the access list 370 can be obtained dynamically based on observation of actual reference behavior of the clients. For example, in HTML, a Web page can have HTTP links to other Web pages. Those Web pages indicated by the HTTP links are potential child objects to the (parent) Web page containing the HTTP links. When the parent object is being viewed, if a reference (HTTP) link contained in the parent object is clicked, the corresponding child object gets referenced. A child object can have more than one parent as a reference link to the child object can be included in multiple other parent objects. Under the HTTP protocol, the request for the child object contains the parent URL as referer information in its header. So the server can dynamically relate a subsequent child reference to its parent (through the referer information in the child’s header). If a client frequently clicks or selects a child object, the child object may become a prefetch candidate or select information.

[0096] In addition to dynamically tracking the viewer reference behavior, static analysis or mining on Web logs can be performed periodically to derive information on groups of objects for the statistics table 261. The mining algorithm identifies groups of objects that are referenced together to obtain potential prefetch candidates for PHI. One method for mining Web logs to derive reference pattern is described in: “SpeedTracer: A Web Usage Mining and Analysis Tool”, by Kun-lung Wu, et al., IBM Research Report 20895, May 1997 (see IBM Systems Journal v37, n1, p89-105, 1998). The SpeedTracer algorithm provides information on frequent groups of pages referenced together in a user session and the frequent traversal path. The SpeedTracer algorithm is an in-depth type of mining that can uncover related references that are not immediately referenced together. It can also make better predictions on the next set of referenced objects based on the traversal path to a given object.

[0097] Other types of data mining algorithms can be used to identify data access patterns for the statistics table 261. For example, other data mining algorithms include classification and clustering, which can also be applied to the Web log analysis to obtain reference patterns. Furthermore, the PHI can also be supplied by the content provider based on its knowledge of the object relationships. For example, PHI can include, but is not limited to, patches and upgrades for requested software download, Java classes for the requested JAVA applet, and objects (such as GIFs) contained within the requested objects.

[0098] In the preferred embodiment, when collecting statistics, actual referencing of an object by a client station is distinguished from a prefetch request to an object. The first count 302 (i.e. TCount) and the second count 310 (i.e. RCount) in the statistics table 261 reflect actual references, not prefetching. That is to say a prefetch operation will not result in an update to the first statistics.
The prefetch hint information or the prefetch hint information list may use various data structures for annotating requested information or requested objects. For example, prefetch hint information may be communicated between requester and server nodes using piggybacked meta data. In a HTTP implementation, the information exchange can be included as meta data in an object header using existing web protocols. PICS ("Platform for Internet Content Selection") specifies a method of sending meta-information concerning electronic content. PICS is applicable to the (a) annotation of the requested information, (b) prefetching select information, prefetch candidates, or child objects, and (c) acknowledgment prefetching operations after information is prefetched.

PICS is known to one of ordinary skill in the art as a Web Consortium Protocol Recommendation. PICS was first used for sending values-based rating labels, such as "How much nudity is associated with this content", but the format and meaning of the meta-information is fully general. In PICS, meta-information about electronic content is grouped according to the "rating service" or producer-and-intended-usage of the information, and within one such group, any number of categories or dimensions of information may be transmitted. Each category has a range of permitted values, and for a specific piece of content, a particular category may have a single value or multiple values. In addition, the meta-information group (known as "PICS label") may contain expiration information. There are also facilities for permitting a PICS label to apply to more than one piece of electronic content. Each PICS label for a specific piece of electronic content may be added or removed from the content independently.

For example, an image file may be sent from a server with a PICS label whose "rating service" field indicates it contains values-based rating labels according to the "SafeSurf" rating system. According to the preferred embodiment of the present invention, as it passes through an enterprise proxy, the image file may get a new updated category value for the PICS label to reflect the local consideration of the "rating service". Thus, the client computer will only see the new category value of the PICS label. The HTTP protocol has been augmented so that its request headers and response headers support PICS. The technical bodies which define other common application protocols, such as NNTP, are now also considering adding PICS support. As part of these protocols, a list of the types of PICS labels desired may be included with a request. PICS also specifies a query format for requesting PICS information from a central label bureau server.

A sample PICS label is:

\[ \text{[PICS-1.1} \text{ "http://the.rating.service" labels for "http://the.content" exp "1997.07.09T08:15-0500" r (n 4 s 3 v 2)} \]

where the "n", "s", and "v" are transmit names for various meta-information types, and the applicable values for this content are 4 (for n), 3 (for s), and 2 (for v). Only software which recognizes the ID "http://the.rating.service" would know how to interpret these categories and values.

In a preferred embodiment, three different kinds of PICS labels are introduced. The first kind of PICS label, referred to as a prefetch label (i.e. P-label), is used by the server node to provide PHI. The second kind of PICS label, referred to as a request label (i.e. R-label), is used to request the prefetch candidate objects. The third kind of PICS label, referred to as a reference label (i.e. X-label), is used to provide feedback on the effectiveness of the prefetch, based on which prefetch objects actually get referenced.

The prefetch label can include, but is not limited to, any combination of the following categories including an identifier category, a probability category, a size category, and a caching category. The identifier category (i.e. ID category) has a value that conveys the URL of the prefetch candidate. The probability category (i.e. PROB category) has a value that conveys an estimate of the usefulness of the prefetch. In the preferred embodiment, an estimate of the reference probability is used. The reference probability indicates the probability that the prefetch candidate gets referenced. The size category has a value that conveys the size or memory requirements of the prefetch candidate. The caching category has a value that conveys information on any higher level proxy server that has a copy of the prefetch candidate object either in its first main cache 294 or its first prefetch cache 296. In the preferred embodiment, only the closest higher level proxy is identified.

The request label can include, but is not limited to, the following categories, including the identifier category (i.e. ID category). The value of the identifier category conveys the URL of the object to be prefetched. In the preferred embodiment, when a (proxy or client) node wants to prefetch a suggested prefetch candidate object, a HTTP head request is sent back to the content server 20 through the proxy hierarchy using the request label to convey the URL of the object to be prefetched. When a proxy server 24 receives a prefetch request from a lower hierarchy node, it will serve the request if the object is in its main or prefetch cache. Otherwise, it forwards the prefetch request to the next higher level proxy.

The reference label can include, but is not limited to, the following categories, including the object identifier category (Object-ID category) and the referrer identifier category (Referrer-ID category). The object identifier category has a value that conveys the URL of the prefetched object that gets referenced. The referrer identifier category has a value that conveys the URL of the object that contains in its PHI the object indicated by the object identifier category value.

In the preferred embodiment, when a prefetch object actually gets referenced, a HTTP head request is sent back to the content server 20 through the proxy hi-
eral implementation the reference label to convey the URL of the prefetch object referenced and its referrer object. This type of HTTP head request is referred to hereinafter as the prefetch referenced message. When a proxy server 24 receives a prefetch referenced message from a lower hierarchy node, it updates its first statistic table 293 and forwards the request to the next higher level proxy server.

[0108] FIG. 4 depicts an example of server logic 268 which updates the second statistics table 267 in the content server 20. The updating of the second statistics table 267 is triggered by an object request from a client 22. The content server 20 preferably generates prefetch hint information for annotation or insertion into the object header of the object prior to sending the requested object to the client 22.

[0109] Starting at reference block 405, the content server 20 waits for input. In decision symbol 410, if the input received is an object request for object O, the statistics updating unit 252 or second update statistic routine 269 is invoked in reference block 415. In reference block 420, generating unit 266 for generating prefetch hint information 266 or a prefetch generation routine is invoked. A detailed example of the generating unit 266 or PHI generation routine is later described with reference to FIG. 6. In reference block 425, the PHI is, for example, inserted into the object header of the requested object. The prefetch hint information may otherwise annotate the requested object or requested information in block 425. In the preferred embodiment, the PICS protocol may be used to annotate the requested object. The PHI is captured in the P-label, where the identifier category value reflects the URL of the prefetch candidate. In block 430, the requested object is returned to the requesting client or the requesting proxy server. In decision symbol 435, if the input received is a prefetch referenced message (i.e. a HTTP head request with the X-label indicating that a prefetch object O actually gets referenced) the statistics updating means 252 or the second update statistics routine 269 is invoked in step 440. For other types of inputs, which are not the focus of the present invention, such as an FTP (file transfer protocol) request, an appropriate miscellaneous handler in reference block 445 can be invoked.

[0110] FIG. 5 shows how the statistics updating unit 252 updates one of said statistics tables 261. The statistics updating unit 252 refers to the first update statistics routine 264 for updating the first statistics table 293 or the second update statistics routine 269 for updating the second statistics table 267, or both the first update statistics routine 264 and the second update statistics routine 269. The statistics table 261 refers to either the first statistics table 293 or the second statistics table 267, or to both the first statistics table 293 and the second statistics table 267. The statistic updating unit 252 first looks for information in a referrer object. The requested object is then placed on an access list or a prefetch list of the referrer object.

[0111] Beginning at reference block 505, the statistics table 261 is updated to reflect that object O has been referenced one more time by incrementing the first count. In other words, in block 505 TCount(O) is incremented by one. If object O is not yet in the statistics table 261, a parent object identifier is inserted into the statistics table 261 with TCount(O) initialized to one. The new parent object identifier replaces the least recently referenced entry in the statistics table 261, if no empty slot is available in the statistics table 261. In decision symbol 510, the requested object is checked for referrer information by, for instance, checking the object header of the requested object. For example, in HTTP protocol, referrer object is provided in the header to indicate which object contains the HTTP link to the requested object. If a referrer object (i.e. R) is found in the header, in step 520 the statistics table 261 is examined to see whether object O is already on the access list of object R. If so, the second count (i.e. RCount(O)) is incremented by one in block 530 for the requested object. Otherwise, in block 525 the requested object (i.e. Object O) is inserted into the access list of the parent object (i.e. object R) with the second count (i.e. RCount(O)) initialized to one.

[0112] FIG. 6 depicts an example of the generating unit 266 for generating prefetch hint information into a prefetch hint information list. The prefetch hint information list contains child object identifiers of child objects if the ratio of the second count to the first count exceeds a hint threshold. The prefetch hint information list represents a refined or filtered access list. That is, the generating unit 266 accepts inputs from the statistics table 261 and filters the inputs to obtain the prefetch hint information list.

[0113] The generating unit 266 for generating prefetch hint information represents a prefetch hint information (PHI) generation routine 266, which begins in reference block 610. In reference block 610, the access list of a parent object (i.e. object O) is checked to see whether or not it is empty (i.e. whether the Alist(O) is null). If the access list is not empty, in reference block 620, let Cj be the next child object (i.e. child object identifier) identified on the access list of parent object (i.e. object O).

[0114] In decision symbol 630, the reference probability of the child object Cj is compared with a hint threshold (i.e. TH), which is a prespecified threshold value (e.g. 0.75). In the preferred embodiment, the reference probability of Cj is defined to be the ratio of the second count to the first count (i.e. RCount(Cj)/TCount(O)). The ratio is an estimate of the probability that child object (object Cj) gets referenced after the parent object (object O) is referenced. If in decision symbol 630, the reference probability of Cj is larger than the hint threshold (i.e. TH), then the child object identifier (Cj) will be included in the PHI list in reference block 640. In other words, if the ratio exceeds the hint threshold, the child object identifier and associated information is placed on the prefetch hint information list.

[0115] The generating unit 266 may require software subroutines or software components to practice the com-
now test whether the product of RCount symbol 630, the comparing unit or evaluating unit may parent object (object O), if not yet included. At decision mine whether to be included in the prefetch list of the of the child object (object Cj) may be evaluated to deter-

(candidates can be made recursive. If a child object (object Cji) is selected as a prefetch candidate of parent object candidates from the statistics table 261.

[0117] Various alternative criterion can be used to select the prefetch candidates over past intervals can be used as the criterion. Furthermore, the selection procedure of the prefetch candidates can be made recursive. If a child object (object Cj) is selected as a prefetch candidate of parent object (object O), any child object (object Cji) on the access list of the child object (object Cj) may be evaluated to determine whether to be included in the prefetch list of the parent object (object O), if not yet included. At decision symbol 630, the comparing unit or evaluating unit may now test whether the product of RCount(Cj)/TCount(O) and RCount(Cji)/TCount(Cj) is larger than the hint threshold (TH).

[0118] FIG. 7 depicts an example of the operation of client logic 244. FIG. 7 shows that once an object is requested, the client object request handler 247 is invoked. Otherwise, the prefetch handler 241 may handle objects which are not requested, but need to be stored in the second prefetch cache 249 of the client 22.

[0119] Beginning in block 710, the client 22 waits for input. In decision symbol 720, if the input received is an object, the second prefetch handler 241 is invoked in reference block 730. In block 740, the client object handler is invoked to return the object to the requesting application at the client 22. If in decision symbol 750 the input received is an object request, the client object request handler 247 is invoked at reference block 760. For other types of inputs, which are not the focus of the present invention (such as a push request) an appropriate miscellaneous handler at reference block 770 can be invoked.

[0120] FIG. 8 depicts an example of the operation of the prefetching unit 250, including the first prefetch handler 248, or the second prefetch handler 241, or both the first prefetch handler 248 and the second prefetch handler 241. To summarize, the prefetching unit 250 uses the prefetch list generated by the generating unit 266 in FIG. 6 as an input to determine if child objects referenced on the prefetch hint information list warrant storing in one of the prefetch caches 298. The prefetching unit 250 calculates a prefetch value for each prefetch object candidate or child object identifier on the prefetch hint infor-

mation list. The prefetch value considers access probability of the child object and access cost of storing the child object. The child objects with the highest threshold values are cached first until the available space in one of the prefetch caches 298 is filled.

[0121] Starting in reference block 805, let L be the PHI list of prefetch candidate objects. The prefetch candidate objects may be expressed in terms of child object identifiers on the prefetch hint information list. Any prefetch candidate appearing in the local cache (i.e. prefetch cache or main cache) is removed from L. In block 810, the prefetch value (PV) of each object in L is calculated. In the preferred embodiment, the PV value is defined to be the reference probability multiplied by the access cost. The access cost is estimated based on the closest site caching the prefetch candidate object or the child object identified on the PHI list. As a practical matter, the child objects preferably are identified on the PHI list by child object identifiers rather than as the entire child objects to conserve storage space. The reference probability optimally comes from the probability category (PROB category) value and the closest caching site comes from the caching category value of the PICS label. In block 820, the prefetch candidate object with the highest PV value is selected. In reference block 830, the amount of prefetch buffer space available to accommodate the prefetch object candidate (i.e. oj) is determined. This space is either not currently in use or occupied by objects with lower PV values which are only a fraction (e.g. one half) of the PV value of the present prefetch object candidate being evaluated (i.e. Oj). Any storage space occupied by an object marked as deletable in one of the prefetch caches 298 is considered not currently in use. In decision symbol 840, one of the prefetch caches 298 is checked to see whether there is enough space for Oj. If so, in reference block 850, object Oj is prefetched into one of the prefetch caches 298. In block 860, Oj is removed from the PHI list (i.e. L). In reference block 870, if L is not empty, the step associated with reference block 820 is re-executed.

[0122] Although in the preferred embodiment each object is prefetched one at a time, multiple objects can be prefetched together in one request. In reference block 850, Oj can be put onto a prefetch list. In block 870, if L is empty, a prefetch request is then issued with the whole prefetch list. Furthermore, even if there is not enough storage space in the prefetch cache, a pending prefetch list can be maintained, so that when storage becomes available, additional prefetch can be initiated.

[0123] By using the HTTP protocol, the communication session can remain open during the prefetch to reduce communication overhead. There is no need to reestablish the session for each prefetch.

[0124] FIG. 9 depicts an example of the operation of the client object request handler 247. To summarize, the client object request handler 247 first searches the second main cache 246 and secondly searches the second prefetch cache 249. If the object is in neither the second
Starting at decision symbol 905, if the requested object O is found in the second main cache 246, the second prefetch handler 241 is invoked in reference block 925 to initiate appropriate prefetch operations based on the PHI associated with the object O. In reference block 930, the client object handler is invoked to return the requested object to the requesting application at the client. In decision symbol 910, if the requested object O is found in the second prefetch cache 249, a prefetch referenced message is sent to the content server 20 in block 920 so that its second statistics table 267 can be updated to capture the actual referencing of the prefetch object. In fact, as the prefetch referenced message propagates through the proxy hierarchy to the content server 20, the second statistics table 267 in each intermediate proxy will also get updated. In block 910, if the object is not in the second prefetch cache 249, an object request is send to the content server 20 through the proxy hierarchy in reference block 915.

The prefetch referenced messages can be batched. That is, a prefetch referenced list can be maintained and periodically a composite prefetch referenced message is sent to the higher level server to acknowledge the referencing of all objects on the prefetch referenced list.

FIG. 10 depicts an example of the operation of a proxy server logic 295. In summary, the input is sorted in accordance with whether the input is an object request, an object transmission, or a prefetch referenced message. Three different handlers or software routines are available depending upon the input: a proxy object request handler 283, a proxy object handler 284, and a prefetch reference message handler 285.

Starting at reference block 1005, the proxy server 24 waits for input. If in decision symbol 1010 the input received is an object request for an object O, the proxy server 24 checks whether there is already a pending request on that object O in block 1015. The pending request may be due to a prefetch request on that object O issued by this proxy server 24 or an earlier request to the object O originated from another proxy server node, which is lower in the proxy server hierarchy. If there is no pending request on the object O in reference block 1015, the proxy object request handler 283 is invoked in block 1020. In decision symbol 1025, if the input received is an object (from the higher level proxy), the proxy object handler 284 is invoked in reference block 1030. Otherwise, in block 1040, if the input received is a prefetch referenced message, the prefetch referenced message handler 285 is invoked in block 1045. For other types of inputs, which are not the focus of the present invention (such as the an FTP request) an appropriate miscellaneous handler in block 1060 can be invoked.

Beginning in decision symbol 1105, if the requested object O is found in the prefetch cache, the object O is returned to the requesting node in block 1130. In decision symbol 1135, if the request is a prefetch request, the object O in the first prefetch cache 296 is marked deletable in block 1150. Deleting objects from the first prefetch cache 296 makes room for future prefetch because the prefetch object O has now been prefetched by some nodes in the lower hierarchy of proxy servers 24. Otherwise, in reference block 1140, the cache manager is invoked to move the object O from the first prefetch cache 296 to the first main cache 294 since the object O has actually been referenced. In block 1160, a prefetch referenced message is sent to the content server 20 through the proxy hierarchy server. In block 1170, the update statistics routine is invoked to update the local first statistics table 293 to capture the actual referencing of the prefetch object O. In decision symbol 1110, if the object O is not found in the first main cache 294, an object request is send to the content server 20 through the proxy server hierarchy in reference block 1125. In decision symbol 1115, if the object request is not a prefetch request, the first update statistics routine 264 is invoked in block 1120 to update the local first statistics table 293 to capture the actual referencing of the object. In block 1180, the object O is returned to the requesting node.

FIG. 12 depicts an example of the operation of the prefetch referenced message handler 285. The prefetch referenced message handler 285 first looks at whether an object is in the first prefetch cache 296 or not. If the object is in the first prefetch cache 296, the cache manager routine is invoked prior to forwarding the prefetch referenced message. If the object is not in the first prefetch cache 296, the prefetch referenced message is forwarded.

Beginning at step 1205, the first update statistics routine 264 is invoked to update the local first statistics table 293 to capture the actual referencing of the
prefetch object (O). In decision symbol 1210, if the object 0 is in the first prefetch cache 296, the cache manager is invoked in reference block 1215 to move the object 0 from the first prefetch cache 296 to the main cache 294 since the object has actually been referenced. In block 1220, the prefetch referenced message is forwarded to the next higher level proxy.

Beginning at reference block 1305, the PHI update routine or the prefetch updating means 291 is invoked to update the PHI of object O in reference block 1310, if the object received is not a prefetch object, the first prefetch handler 248 is invoked in step 1315. In reference block 1320, the object is returned to the requesting node (in the lower hierarchy). Under the preferred embodiment using the PICS protocol, the caching category value in the P-label is updated if the corresponding prefetch candidate (indicated by the identifier category value) is prefetched by this proxy server 24. In block 1325, the cache manager is invoked to store the object in the main cache. In reference block 1355, the update statistics routine is invoked. In block 1330, if the prefetch object received is requested by a lower level node in the hierarchy, the object is returned to the requester in reference block 1350. Otherwise, it is requested by the current node, and the prefetch cache manager is invoked in reference block 1340 to store the object in the prefetch cache by making necessary replacement to make room for the object.

Beginning in decision symbol 1405, if the local count (i.e. TCount) of the object (O) in the local statistics table 293 is larger than a second minimum threshold (i.e. TCH), the original PHI in the object header is ignored and the PHI generation routine is invoked in block 1420 to generate the PHI based on the local first statistics table 293. This is because there are sufficient references to the object lower in the hierarchy of proxy servers 24 to generate the PHI based on the local behavior. Otherwise, the PHI modifier is invoked in reference block 1425 to modify the probability category value by taking an average of the local estimate of the reference probability and that in the header (set by the higher level hierarchy) for each object in the PHI list.

The prefetch scheme works even if some of the proxy servers 24 do not maintain the first statistics table 293 and track local reference patterns. The non-tracking proxy servers simply accept the probability (i.e. PROB) category value in the PHI passed down from the higher level proxy server or content server 20.

The prefetch scheme also works in a heterogeneous proxy-server environment, where some of the proxy servers are conventional proxy servers which do not understand the prefetch protocol and do not participate in the collaboration. These conventional proxy servers will neither interpret the object header to initiate the prefetch operations, nor maintain the first statistics table 293. The conventional proxy servers will only pass the meta data (e.g. the PICS labels in the header) with the object to the next level proxy server or store the meta data with the object if it decides to cache the object locally.

Even without the presence of the proxy hierarchy, the current invention can be applied to any network or client server environment. The object sender node can track reference patterns, derive PHI and include it in the object header. Furthermore, the object receiver node can store the PHI with the object in its cache. Upon referencing of the object, prefetch operations can be initiated based on the stored PHI and local factors such as cache size and access cost as discussed before.

Storing the PHI with the cached object provides additional opportunity for prefetching. Events such as more storage becoming available, additional references occurring or prefetch objects being received, can be used to trigger additional prefetch by examining the PHI of the cached objects. The prefetch operation is not restricted to the moment that an object is requested from the server as in the sender initiated approach.

In the preferred embodiment, a general prefetch policy for web and proxy servers has been described, but those skilled in the art will appreciate, however, that the present invention is applicable to any kind of situation where the objects to be prefetched have similar characteristics, and is not necessarily restricted to an Internet or WWW application.
parent and child proxies in the hierarchy, it can be easily adapted to include collaboration between sibling proxies. For example if no proxy in the upper hierarchy has cached the requested object or prefetch candidate, a proxy can also interrogate sibling proxies.

Claims

1. A method for prefetching information from a content server (20) via a communications network (25), said communication network including one or more proxy servers (24) and one or more clients (22), the method comprising the steps of:

- requesting (100), at a client, information from the content server via at least one proxy server;
- characterized by establishing (102, 104), at the server, prefetch hint information denoting information likely to be requested by the client subsequent to said requested information, based on past retrievals or accesses by said clients;
- annotating (105), at the server, the requested information with said prefetch hint information and returning the requested information and the prefetch hint information together to the client via the at least one proxy server; and
- prefetching (108), at the client or said at least one proxy server, information based upon the prefetch hint information and the requested information, thereby making the prefetched information more readily accessible.

2. The method of claim 1, wherein the requested information comprises requested objects, and wherein the prefetched information comprises prefetch candidate objects.

3. The method of claim 2, wherein the step of establishing prefetch hint information includes the steps of:

- tracking data reference patterns of requested information associated with said clients;
- organizing the data reference patterns into groups of object identifiers, the groups of object identifiers organized according to relationships between the requested objects and prefetch candidate objects;
- determining which of said groups have sufficient relationships between the prefetch candidate objects and the requested objects to warrant prefetching of said prefetch candidate objects as the prefetched information.

4. The method of claim 3, wherein the steps of tracking and organizing the data reference patterns further comprise the steps of:

- establishing a database (383, 385) defining a relationship between requested objects and prefetch candidate objects; and
- tracking in the database access statistics on the requested objects and the prefetch candidate objects; the access statistics including a parent-object access frequency of requested objects and a child-object access frequency of the prefetch candidate objects; the parent-object access frequency (302) representing the number of times in which a particular one of the requested objects is requested by at least one of said clients; the child-object access frequency (312) representing the number of times in which one or more of said prefetch candidate objects are actually requested proximately following the accessing of said particular one of the requested objects.

5. The method of claim 4, further comprising the step of selecting a prefetch list of prefetch candidate objects based upon the ratio of the child-object access frequency to the parent-object access frequency associated with the child-object access frequency, the prefetch list containing only prefetch candidate objects with ratios that exceed a threshold probability.

6. The method of any of claims 2 to 5, wherein the step of prefetching information comprises the steps of:

- prefetching a subset of prefetch candidate objects received at one of said proxy servers or one of said clients;

wherein the subset of the prefetch candidate objects is selected based on available storage size at one of said proxy servers.

7. The method of any of claims 2 to 6, wherein the step of prefetching information comprises the steps of:

- prefetching a subset of prefetch candidate objects received at one of said proxy servers or one of said clients;

wherein the subset of the prefetch candidate objects is selected based on access time at one of said proxy servers.

8. The method of any of claims 2 to 7, wherein the prefetched object and the requested object comprise Internet data and wherein the requested object references the prefetched object through a hypertext transfer protocol, HTTP, link associated with the Internet data, the server identifying the relationship of the prefetched object to the requested object through the HTTP protocol of the requested object universal resource locator, or URL, stored in an object header
of the requested object.

9. The method of any of claims 2 to 8, wherein the step of annotating the requested information comprises the step of placing the prefetch hint information in a prefetch label having a Platform for Internet Content Selection, PICS, format, or a label protocol compatible with the PICS format.

10. The method of claim 9, wherein the Platform for Internet Content Selection format includes an object identifier category for storing the identity of a prefetched object that subsequently gets referenced, and a referrer identifier category for storing the requested object associated with the prefetched object, and said method further includes the step of (160) returning the referrer identifier category to the content server to identify any prefetch candidate objects which were actually referenced.

11. The method of any of claims 1 to 10, further comprising the steps of:

organizing the proxy servers into a hierarchy of servers, the hierarchy having multiple levels of the proxy servers, the multiple levels being determined based on the proximity of the each of said proxy servers to the network, with the highest level being closest to the network; and

interpreting and updating the prefetch hint information transmitted through the hierarchy from the proxy servers to the clients.

12. The method of claim 11, wherein a proxy server updates the prefetch hint information if local references to the requested information exist and if the count of the local-level server exceeds a minimum threshold count.

13. The method of claim 11 or 12, wherein the prefetch hint information is updated based on caching status in the proxy hierarchy.

14. A system for prefetching information from a content server (20) via a communications network (25), said communication network including one or more proxy servers (24) and one or more clients (22), the system comprising:

means for requesting (100), at a server, the requested information and returning the requested information and the prefetch hint information together to the client via the at least one proxy server; and

means for prefetching (108), at the client or said at least one proxy, server information based upon the prefetch hint information and the requested information, thereby making the prefetched information more readily accessible.

15. The processing system of claim 14, wherein the requested information comprises requested objects, and wherein the select information comprises prefetch candidate objects, and wherein each of said proxy servers and at least one of the clients has a memory having a logical and/or physical separation between a main cache and a prefetch cache, wherein the prefetched select information is stored in the prefetch cache.

Revendications

1. Procédé de pré-extraction d’informations depuis un serveur de contenu (20) par l’intermédiaire d’un réseau de communications (25), ledit réseau de communications comprenant un ou plusieurs serveurs mandataires (24) et un ou plusieurs clients (22), le procédé comprenant les étapes consistant à :

demander (100), au niveau d’un client, des informations auprès du serveur de contenu par l’intermédiaire d’au moins un serveur mandataire,

caractérisé par

l’établissement (102, 104), au niveau du serveur, d’informations d’indications de pré-extraction désignant des informations susceptibles d’être demandées par le client après lesdites informations demandées, sur la base des récupérations ou accès passés par lesdits clients,

l’annotation (105), au niveau du serveur, des informations demandées avec lesdites informations d’indications de pré-extraction et le renvoi des informations demandées et des informations d’indications de pré-extraction ensemble au client par l’intermédiaire du au moins un serveur mandataire, et

la pré-extraction (108), au niveau du client ou dudit au moins un serveur mandataire, des informations sur la base des informations d’indications de pré-extraction et des informations demandées, en rendant ainsi les informations pré-extraites plus facilement accessibles.

2. Procédé selon la revendication 1, dans lequel les informations demandées comprennent des objets
demandés, et dans lequel les informations pré-extraites comprennent des objets candidats de pré-extraction.

3. Procédé selon la revendication 2, dans lequel l’étape consistant à établir des informations d’indications de pré-extraction comprennent des étapes consistant à :

retrouver des configurations de références de données d’informations demandées associées auxdits clients,
organiser les configurations de références de données en groupes d’identificateurs d’objets, les groupes d’identificateurs d’objets étant organisés conformément aux relations entre les objets demandés et les objets candidats de pré-extraction,
déterminer lesquels parmi lesdits groupes ont des relations suffisantes entre les objets candidats de pré-extraction et les objets demandés pour garantir la pré-extraction desdits objets candidats de pré-extraction en tant qu’informations pré-extraites.

4. Procédé selon la revendication 3, dans lequel les étapes consistant à rechercher et à organiser les configurations de références de données comprennent en outre les étapes consistant à :

établir une base de données (383, 385) définissant une relation entre les objets demandés et les objets candidats de pré-extraction, et retrouver dans la base de données les statistiques d’accès sur les objets demandés et les objets candidats de pré-extraction, les statistiques d’accès comprenant une fréquence d’accès aux objets parents des objets demandés et une fréquence d’accès aux objets enfants des objets candidats de pré-extraction, la fréquence d’accès aux objets parents (302) représentant le nombre de fois où un objet particulier parmi les objets demandés est demandé par au moins l’un desdits clients, la fréquence d’accès aux objets enfants (312) représentant le nombre de fois où un ou plusieurs desdits objets candidats de pré-extraction sont réellement demandés immédiatement après l’accès dudit objet particulier parmi les objets demandés.

5. Procédé selon la revendication 4, comprenant en outre l’étape consistant à sélectionner une liste de pré-extraction d’objets candidats de pré-extraction sur la base du rapport de la fréquence d’accès aux objets enfants à la fréquence d’accès aux objets parents associée à la fréquence d’accès aux objets enfants, la liste de pré-extraction ne contenant que des objets candidats de pré-extraction présentant des rapports qui dépassent une probabilité seuil.

6. Procédé selon l’une quelconque des revendications 2 à 5, dans lequel l’étape consistant à pré-extrait des informations comprend les étapes consistant à :

pré-extrait un sous-ensemble d’objets candidats de pré-extraction reçus à fun desdits serveurs mandataires ou à l’un desdits clients,

où le sous-ensemble des objets candidats de pré-extraction est sélectionné sur la base de la taille de mémoire disponible au niveau de l’un desdits serveurs mandataires.

7. Procédé selon l’une quelconque des revendications 2 à 6, dans lequel l’étape consistant à pré-extrait les informations comprend les étapes consistant à :

pré-extrait un sous-ensemble d’objets candidats de pré-extraction reçus au niveau de l’un desdits serveurs mandataires ou de l’un desdits clients,

où le sous-ensemble des objets candidats de pré-extraction est sélectionné sur la base d’un temps d’accès au niveau de l’un desdits serveurs mandataires.


9. Procédé selon l’une quelconque des revendications 2 à 8, dans lequel l’étape consistant à annoter des informations demandées comprend l’étape consistant à placer les informations d’indications de pré-extraction dans une étiquette de pré-extraction ayant un format de plate-forme pour une sélection de contenu du système Internet (sélection PICS), ou un protocole d’étiquette compatible avec le format de sélection PICS.

10. Procédé selon la revendication 9, dans lequel le format de plate-forme pour la sélection d’un contenu du système Internet comprend une catégorie d’identificateur d’objet destinée à mémoriser l’identité d’un objet pré-extrait qui est ensuite référencé, et une catégorie d’identificateur de module de référence destinée à mémoriser l’objet demandé associé à l’objet
pré-extrait, et ledit procédé comprend en outre l’étape consistant à (160) renvoyer la catégorie d’identificateur de module de référence au serveur de contenu pour identifier tout objet candidat de pré-extraction qui est référencé réellement.

11. Procédé selon l’une quelconque des revendications 1 à 10, comprenant en outre les étapes consistant à :

organiser les serveurs mandataires en une hiérarchie de serveurs, la hiérarchie comportant des niveaux multiples des serveurs mandataires, les niveaux multiples étant déterminés sur la base de la proximité de chacun desdits serveurs mandataires par rapport au réseau, le niveau le plus élevé étant le plus proche du réseau, et interpréter et mettre à jour les informations d’indications de pré-extraction transmises au travers de la hiérarchie depuis les serveurs mandataires aux clients.

12. Procédé selon la revendication 11, dans lequel un serveur mandataire met à jour les informations d’indications de pré-extraction si des références locales aux informations demandées existent et si le comptage du serveur de niveau local dépasse un comptage seuil minimum.

13. Procédé selon la revendication 11 ou 12, dans lequel les informations d’indications de pré-extraction sont mises à jour sur la base d’une situation de mise en antémémoire dans la hiérarchie de mandataires.

14. Système destiné à pré-extraires des informations d’un serveur de contenu (20) par l’intermédiaire d’un réseau de communications (25), ledit réseau de communications comprenant un ou plusieurs serveurs mandataires (24) et un ou plusieurs clients (22), le système comprenant :

un moyen destiné à demander (100), au niveau d’un client, des informations en provenance du serveur de contenu par l’intermédiaire d’au moins un serveur mandataire,

caractérisé par

un moyen destiné à établir (102, 104), au niveau du serveur, des informations d’indications de pré-extraction désignant des informations susceptibles d’être demandées par le client après lesdites informations demandées, sur la base de récupérations ou d’accès passés par lesdits clients, un moyen destiné à annoter (105), au niveau du serveur, les informations demandées avec lesdites informations d’indications de pré-extraction et à renvoyer les informations demandées et les informations d’indications de pré-extraction ensemble au client par l’intermédiaire du au moins un serveur mandataire, et un moyen destiné à pré-extraitre (108), au niveau du client ou dudit au moins un serveur mandataire, des informations sur la base des informations d’indications de pré-extraction et des informations demandées, en rendant ainsi les informations pré-extraites plus facilement accessibles.

15. Système de traitement selon la revendication 14, dans lequel les informations demandées comprennent des objets demandés, et où les informations sélectionnées comprennent des objets candidats de pré-extraction, et où chacun desdits serveurs mandataires et au moins l’un desdits clients comportent une mémoire ayant une séparation logique et/ou physique entre une antémémoire principale et une antémémoire de pré-extraction, où les informations sélectionnées pré-extraites sont mémorisées dans antémémoire de pré-extraction.

Patentansprüche

1. Verfahren zum Vorabrufen von Daten aus einem Inhaltsserver (20) über ein Datenübertragungsnetz (25), wobei das Datenübertragungsnetz einen oder mehrere Proxy-Server (24) und einen oder mehrere Clients (22) enthält, wobei das Verfahren die folgenden Schritte umfasst:

Anfordern (100) von Daten aus dem Inhaltsserver von einem Client über mindestens einen Proxy-Server;

gekennzeichnet durch

Einrichten (102, 104) von Vorabrufhinweisdaten im Server, die Daten kennzeichnen, die im Anschluss an die angeforderten Daten wahrscheinlich vom Client angefordert werden, auf der Grundlage von vergangenen Abruf- oder Zugriffsvorgängen durch die Clients;

Kommentieren (105) der angeforderten Daten im Server mit den Vorabrufhinweisdaten und Rückübertragen der angeforderten Daten zusammen mit den Vorabrufhinweisdaten über den mindestens einen Proxy-Server an den Client; und

Vorbrufen (108) von Daten im Client oder in dem mindestens einen Proxy-Server auf der Grundlage der Vorabrufhinweisdaten und der angeforderten Daten, wodurch die abgerufenen Daten problemloser zugänglich gemacht werden.

2. Verfahren nach Anspruch 1, wobei die angeforderten Daten angeforderte Objekte umfassen und wobei die vorabgerufenen Daten Vorabrufkandidatenobjekte umfassen.
3. Verfahren nach Anspruch 2, wobei der Schritt des Einrichtens von Vorabrufhinweisdaten die folgenden Schritte beinhaltet:

- Protokollieren von Datenbezugsmustern von angeforderten Daten, die den Clients zugeordnet werden;
- Organisieren der Datenbezugsmuster in Gruppen von Objektkennzeichnern, wobei die Gruppen von Objektkennzeichnern gemäß Beziehungen zwischen den angeforderten Objekten und Vorabrufkandidatenobjekten organisiert werden;
- Feststellen, welche der Gruppen ausreichende Beziehungen zwischen den Vorabrufkandidatenobjekten und den angeforderten Objekten aufweisen, um ein Vorabrufen der Vorabrufkandidatenobjekte als die vorabgerufenen Daten zu gewährleisten.

4. Verfahren nach Anspruch 3, wobei die Schritte des Protokollierens und Organisierens der Datenbezugsmuster außerdem die folgenden Schritte umfassen:

- Einrichten einer Datenbank (383, 385), die eine Beziehung zwischen angeforderten Objekten und Vorabrufkandidatenobjekten definiert; und
- Protokollieren einer Zugriffsstatistik bezüglich der angeforderten Objekte und der Vorabrufkandidatenobjekte in der Datenbank; wobei die Zugriffsstatistik eine Eltern-Objekt-Zugriffshäufigkeit von angeforderten Objekten und eine Kind-Objekt-Zugriffshäufigkeit der Vorabrufkandidatenobjekte enthält; wobei die Eltern-Objekt-Zugriffshäufigkeit (302) die Anzahl von Malen darstellt, die ein bestimmtes der angeforderten Objekte von mindestens einem der Clients angefordert wird; wobei die Kind-Objekt-Zugriffshäufigkeit (312) die Anzahl von Malen darstellt, die eines oder mehrere der Vorabrufkandidatenobjekte unmittelbar im Anschluss an den Zugriff auf das bestimmte der angeforderten Objekte tatsächlich angefordert werden.


6. Verfahren nach einem der Ansprüche 2 bis 5, wobei der Schritt des Vorabrufens von Daten die folgenden Schritte umfasst:

Vorabrufen einer Teilmenge von Vorabrufkandidatenobjekten, die in einem der Proxy-Server oder einem der Clients empfangen wurden; wobei die Teilmenge der Vorabrufkandidatenobjekte auf der Grundlage der verfügbaren Speichergöße in einem der Proxy-Server ausgewählt werden.

7. Verfahren nach einem der Ansprüche 2 bis 6, wobei der Schritt des Vorabrufens von Daten die folgenden Schritte umfasst:

Vorabrufen einer Teilmenge von Vorabrufkandidatenobjekten, die in einem der Proxy-Server oder einem der Clients empfangen wurden; wobei die Teilmenge der Vorabrufkandidatenobjekte auf der Grundlage der Zugriffszeit in einem der Proxy-Server ausgewählt werden.

8. Verfahren nach einem der Ansprüche 2 bis 7, wobei das vorabgerufene Objekt und das angeforderte Objekt Internetdaten umfassen und wobei sich das angeforderte Objekt durch einen der Internetdaten zugeordneten Hypertext-Transfer-Protokoll- (HTTP-) Link auf das vorabgerufene Objekt bezieht, wobei der Server die Beziehung des vorabgerufenen Objektes zu dem angeforderten Objekt durch das HTTP-Protokoll des Universal Resource Locator (URL) des angeforderten Objektes ermittelt, der in einem Objektvorsatz des angeforderten Objektes gespeichert ist.


11. Verfahren nach einem der Ansprüche 1 bis 10, das außerdem die folgenden Schritte umfasst:

Organisieren der Proxy-Server in eine Hierar-
chie von Servern; wobei die Hierarchie mehrere Ebenen der Proxy-Server aufweist, wobei die mehreren Ebenen auf der Grundlage der Nähe von jedem der Proxy-Server zum Netz festgelegt werden, wobei die höchste Ebene sich am nächsten beim Netz befindet; und
Interpretieren und Aktualisieren der Vorabrufhinweisdaten, die von den Proxy-Servern durch die Hierarchie an die Clients übertragen werden.

12. Verfahren nach Anspruch 11, wobei ein Proxy-Server die Vorabrufhinweisdaten aktualisiert, falls lokale Bezugsnahmen auf die angeforderten Daten vorliegen und falls der Zählerwert des Servers auf der lokalen Ebene einen Mindestschwellenzählerwert über- schreitet.


14. System zum Vorbrufen von Daten aus einem Inhaltsserver (20) über ein Datenübertragungsnetz (25), wobei das Datenübertragungsnetz einen oder mehrere Proxy-Server (24) und einen oder mehrere Clients (22) beinhaltet, wobei das System Folgendes umfasst:

   ein Mittel zum Anfordern (100) von Daten aus dem Inhaltsserver in einem Client über mindestens einen Proxy-Server;

gekennzeichnet durch
   ein Mittel zum Einrichten (102, 104) von Vorabrufhinweisdaten im Server, die auf der Grundlage von vergangenen Abrufvorgängen oder Zugriffs vorgängen durch die Clients Daten kennzeichnen, die im Anschluss an die angeforderten Daten wahrscheinlich vom Client angefordert werden;
   ein Mittel zum Kommentieren (105) der angeforderten Daten im Server mit den Vorabrufhinweisdaten und zum Rückübertragen der angeforderten Daten zusammen mit den Vorabrufhinweisdaten über den mindestens einen Proxy-Server an den Client; und
   ein Mittel zum Vorbrufen (108) von Daten im Client oder in dem mindestens einen Proxy-Server auf der Grundlage der Vorabrufhinweisdaten und der angeforderten Daten, wodurch die abgerufenen Daten problemloser zugänglich gemacht werden.

15. Verarbeitungssystem nach Anspruch 14, wobei die angeforderten Daten angeforderte Objekte umfassen und wobei die Auswahldaten Vorabrufkandidatenobjekte umfassen und wobei jeder der Proxy-Server und mindestens einer der Clients einen Speicher mit einer logischen und/oder physischen Trennung zwischen einem Hauptcachespeicher und einem Vorabrufcachespeicher aufweisen, wobei die vorabgerufenen Auswahldaten im Vorabrufcachespeicher gespeichert werden.
FIG. 2
REQUESTING REQUESTED INFORMATION FROM A PROXY SERVER OR A CONTENT SERVER;

IDENTIFYING DATA ACCESS REFERENCE PATTERNS AT THE CONTENT SERVER AND THE PROXY SERVER;

GENERATING PREFETCH HINT INFORMATION ON RELATED ACCESSES TO THE REQUESTED INFORMATION BASED ON THE USER REFERENCE PATTERNS IDENTIFIED;

ANNOTATING THE REQUESTED INFORMATION BEING SENT TO THE CLIENTS WITH PREFETCH HINT INFORMATION;

UPDATING THE PREFETCH HINT INFORMATION AS THE REQUESTED INFORMATION PASSES THROUGH THE SERVER HIERARCHY TO THE REQUESTING CLIENT; AND

PREFETCHING SELECT INFORMATION BASED ON THE UPDATED PREFETCH HINT INFORMATION AND THE REQUESTED INFORMATION.

FIG. 3A
FIG. 4

268

405
WAIT FOR INPUT

410
OBJECT REQUEST ?

NO

415
YES
UPDATE STATISTIC ROUTINE

420
PHI GENERATION ROUTINE

425
INSERT PHI IN OBJECT HEADER

430
RETURN OBJECT TO REQUESTER

435
PREFETCH REFERENCED MESSAGE ?

YES

440
UPDATE STATISTICS ROUTINE

445
MISCELLANEOUS HANDLER
UPDATE T COUNT (O) OF REQUESTED OBJECT O

INFORMATION ON REFERRER OBJECT R AVAILABLE?

OBJECT O ON ACCESS LIST OF OBJECT R

UPDATE R COUNT OF OBJECT O

INSERT OBJECT O ONTO ACCESS LIST OF OBJECT R

END ROUTINE

FIG. 5
610
ACCESS LIST OF OBJECT O EMPTY?

620
NO
NEXT OBJECT Cj' ON THE ACCESS LIST OF OBJECT O

630
R COUNT (Cj')
T COUNT (O) > TH

640
YES
ADD Cj AND ASSOCIATED INFORMATION TO PHI LIST

645
NO
ACCESS LIST OF OBJECT O FULLY SCANNED

END ROUTINE

FIG. 6
FIG. 7
SET L EQUAL TO THE LIST OF PREFETCH CANDIDATES FROM PHI AND REMOVE THOSE CURRENTLY IN LOCAL CACHE

CALCULATE PREFETCH VALUE (PV) FOR EACH OBJECT IN L

PICK THE OBJECT Oj FROM L WITH THE HIGHEST PV VALUE

DETERMINE SPACE AVAILABILITY FOR Oj

ENOUGH SPACE FOR Oj?

YES

PREFETCH Oj

NO

REMOVE Oj FROM L

L EMPTY?

YES

END ROUTINE

FIG. 8
FIG. II

1105
OBJECT IN PREFETCH CACHE?

NO

1130
RETURN OBJECT

YES

1110
OBJECT IN MAIN CACHE?

NO

1135
PREFETCH REQUEST?

YES

1125
SEND OBJECT REQUEST

NO

1140
CACHE MANAGER ROUTINE

1160
SEND PREFETCH REFERENCED MESSAGE

1150
MARK OBJECT DELETABLE

1180
RETURN OBJECT

1120
UPDATE STATISTICS ROUTINE

1170
UPDATE STATISTICS ROUTINE

1115
PREFETCH REQUEST

YES

END ROUTINE
FIG. 12

1205
UPDATE STATISTICS ROUTINE

1210
OBJECT IN PREFETCH CACHE?

1215
CACHE MANAGER ROUTINE

1220
FORWARD PREFETCH REFERENCED MESSAGE