IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

ORACLE CORPORATION and ORACLE U.S.A. INC.,	)	
Plaintiffs,	)	
v.	)	C.A. No. _____
EPICREALM LICENSING, LP,	)	
Defendant.	)	

COMPLAINT FOR DECLARATORY JUDGMENT

Plaintiffs Oracle Corporation and Oracle U.S.A. Inc., for their complaint against Defendant epicRealm Licensing, LP ("epicRealm"), allege as follows:

THE PARTIES

1. Plaintiff Oracle Corporation is a corporation organized and existing under the laws of the State of Delaware, having its principal place of business at 500 Oracle Parkway, Redwood Shores, California 94065.

2. Plaintiff Oracle U.S.A. Inc. is a corporation organized and existing under the laws of the State of Colorado, having its principal place of business at 500 Oracle Parkway, Redwood Shores, California 94065. Oracle U.S.A. Inc. is a wholly owned subsidiary of Oracle Corporation (together referred to herein as "Oracle").

3. Defendant epicRealm is a limited partnership organized and existing under the laws of the State of Delaware, having its principal place of business at 558 S. Central Expressway, Richardson, Texas 75080-6126.

JURISDICTION AND VENUE

4. This is a complaint for declaratory relief under the patent laws of the United States. This Court has subject matter jurisdiction over this action pursuant to 28 U.S.C. §§ 1331, 1338(a), 2201(a) and 2202.

5. This Court has personal jurisdiction over epicRealm by virtue of its organization under the laws of the State of Delaware.

6. Venue is proper in this judicial district pursuant to 28 U.S.C. §§ 1391(b), (c) and 1400(b).

THE EPICREALM PATENTS

7. U.S. Patent No. 5,894,554 (the "'554 patent") issued on April 13, 1999 and is entitled, System for Managing Dynamic Web Page Generation Requests by Intercepting Request at Web Server and Routing to Page Server Thereby Releasing Web Server to Process Other Requests.

8. U.S. Patent No. 6,415,335 (the "'335 patent") issued on July 2, 2002 from a division of U.S. Patent Application No. 08/636,477, now the '554 patent, and is entitled, System and Method for Managing Dynamic Web Page Generation.

9. epicRealm asserts that it is the owner and assignee of the '554 and '335 patents (the "epicRealm patents") and has the right to enforce the epicRealm patents against Oracle. Copies of the epicRealm patents are attached hereto as Exhibits A and B.

EPICREALM'S PATENT INFRINGEMENT LAWSUITS

10. On May 2, 2005, epicRealm, then known as epicRealm Licensing, LLC, filed a patent infringement complaint in the District Court for the Eastern District of Texas (Marshall Division) accusing defendants Autoflex Leasing, Inc., eHarmony.com, Inc.,

Friendfinder Network, Inc., Grande Communication Networks, Inc., IJL-NCP, LLC and Transplace Texas, LP of infringing the epicRealm patents. That action was assigned Civil Action No. 2:05-CV-163 (the "'163 action") and was originally assigned to United States District Judge T. John Ward. On June 10, 2005, epicRealm filed a First Amended Complaint in the '163 action.

11. On August 5, 2005, epicRealm filed a patent infringement complaint in the District Court for the Eastern District of Texas (Marshall Division) accusing defendants Franklin Covey Co., Clark Consulting, Inc., The Macerich Company, Safelite Group, Inc., Herbalife International of America, Inc. and Pink Sheets, LLC of infringing the epicRealm patents. That action was assigned Civil Action Number 2:05-CV-356 (the "'356 action") and was originally assigned to United States District Judge T. John Ward. On November 2, 2005, epicRealm filed a First Amended Complaint in the '356 action.

12. On November 2, 2005, Judge Ward consolidated the '163 and '356 actions (the "Consolidated epicRealm Actions").

13. On November 9, 2005 and November 16, 2005 the Consolidated epicRealm Actions were reassigned to United States District Judge David Folsom.

14. On January 27, 2006, epicRealm filed a Second Amended Complaint in the Consolidated epicRealm Actions accusing defendants Autoflex Leasing, Inc., eHarmony.com, Inc., Friendfinder Network, Inc., Grande Communications Networks, Inc., Transplace Texas, LP, Franklin Covey Co., Clark Consulting, Inc., Macerich Company, Safelite Group, Inc., Herbalife International of America, Inc. and Pink Sheets, LLC of infringing the epicRealm patents.

EPICREALM'S PATENT INFRINGEMENT ACCUSATIONS AND ORACLE'S REASONABLE APPREHENSION OF SUIT

15. In its Second Amended Complaint, epicRealm alleges that the named defendants infringe the epicRealm patents because they use systems and methods for managing requests for dynamic web pages.

16. Oracle sells and licenses software products to numerous customers for use in conjunction with the delivery of dynamic web pages. Oracle has entered into agreements requiring Oracle to indemnify most of these customers in the event a customer is sued for patent infringement for its use of licensed Oracle software.

17. Safelite Group, Inc. ("Safelite"), a defendant in the '356 action, is a customer of Oracle in that Safelite has purchased and licensed certain software products, including but not limited to Oracle E-Business Suite Release 11i, (collectively, the "Oracle Software") from Oracle.

18. On January 26, 2006, Safelite filed a Third Party Complaint against Oracle and International Business Machines Corp. ("IBM") in the '356 action (the "Safelite Action").

19. In its Third Party Complaint and in a January 26, 2006 letter from Safelite to Oracle (the "Safelite Letter"), Safelite alleged that under an "Oracle License & Services Agreement" dated November 29, 2002, as amended, (the "Oracle Agreement"), it purchased and licensed the Oracle Software from Oracle.

20. In its Third Party Complaint and in the Safelite Letter, Safelite alleged that epicRealm's infringement contentions are broad enough to encompass or implicate "virtually any system/method that generates web pages with dynamic content" including "Oracle products licensed to Safelite."

21. In its Third Party Complaint and in the Safelite Letter, Safelite alleged that under the Oracle Agreement and under common law it is entitled to indemnification from Oracle if a product provided by Oracle to Safelite is accused of infringing a third party's intellectual property rights, and Safelite demanded that Oracle agree to defend and indemnify Safelite from any and all epicRealm claims that any Oracle product infringes the epicRealm patents.

22. In its Third Party Complaint, Safelite prayed that the District Court for the Eastern District of Texas hold Safelite entitled to be fully indemnified against any monetary judgment, against Safelite and in favor of epicRealm, as to any claim that any Oracle product provided or licensed to Safelite infringes any claim of the epicRealm patents.

23. In a January 25, 2006 letter from epicRealm to Clark Consulting Inc. ("Clark"), a defendant in the '356 action, epicRealm alleged that a system or method that employed proprietary server software "could, if used to generate web pages with dynamic content, be configured in a way that would infringe the claims of the epicRealm patents." The January 25, 2006 letter from epicRealm to Clark (the "epicRealm Letter") is attached hereto as Exhibit C.

24. In the epicRealm Letter, epicRealm stated that Clark was required to provide discovery disclosures to epicRealm regarding Clark's use of proprietary server software for generating web pages.

25. In the '356 action, epicRealm demanded and received discovery from Safelite regarding Safelite's use of the Oracle Software.

26. In June 2006, epicRealm settled its dispute with Safelite and stipulated to a dismissal with prejudice of its claims against Safelite in the Consolidated epicRealm Actions.

The District Court for the Eastern District of Texas entered an order dismissing epicRealm's claims with prejudice on June 29, 2006.

27. In June 2006, Safelite stipulated to dismissing with prejudice its Third Party Complaint against Oracle and IBM, pursuant to an agreement whereby Oracle contributed money to Safelite's settlement with epicRealm. The District Court for the Eastern District of Texas entered an order dismissing Safelite's Third Party Complaint with prejudice on June 29, 2006.

28. epicRealm's patent infringement allegations in the Consolidated epicRealm Actions against Safelite and others, the epicRealm Letter, and epicRealm's demand for discovery of Oracle information from Safelite cause Oracle to have a reasonable apprehension that (1) epicRealm will accuse Oracle, Oracle's products and/or Oracle's customers of infringing one or more claims of the epicRealm patents, and/or (2) additional Oracle customers will sue Oracle for indemnity as a result of epicRealm's patent infringement claims.

FIRST CLAIM FOR RELIEF

(Declaratory Judgment Of Noninfringement Of the epicRealm Patents)

29. Oracle incorporates by reference Paragraphs 1 through 28 above.

30. By virtue of epicRealm's patent infringement allegations in the Consolidated epicRealm Actions against Safelite and others, the epicRealm Letter, and epicRealm's demand for discovery of Oracle information from Safelite, an actual controversy exists between Oracle and epicRealm as to whether Oracle, Oracle's products and/or one or more Oracle customers infringe the epicRealm patents.

31. Oracle has not and does not infringe, literally or under the doctrine of equivalents, either directly, indirectly or willfully, any valid and enforceable claim of the epicRealm patents.

32. Pursuant to 28 U.S.C. §§ 2201 and 2202, a judicial determination of the respective rights of the parties with respect to Oracle's noninfringement of the epicRealm patents is necessary and appropriate under the circumstances.

SECOND CLAIM FOR RELIEF

(Declaratory Judgment Of Invalidity Of The epicRealm Patents)

33. Oracle incorporates by reference Paragraphs 1 through 32 above.

34. By virtue of epicRealm's patent infringement allegations in the Consolidated epicRealm Actions against Safelite and others, the epicRealm Letter, and epicRealm's demand for discovery of Oracle information from Safelite, an actual controversy exists between Oracle and epicRealm as to the validity of the epicRealm patents.

35. Each claim of the epicRealm patents is invalid for failure to meet one or more of the conditions of patentability specified in 35 U.S.C. §§ 101, 102, 103 and/or 112.

36. Pursuant to 28 U.S.C. §§ 2201 and 2202, a judicial determination of the respective rights of the parties with respect to the validity of the epicRealm patents is necessary and appropriate under the circumstances.

PRAYER

WHEREFORE, Oracle requests entry of judgment in its favor and against epicRealm as follows:

a. Declaring that Oracle and its software products that are used in conjunction with the delivery of dynamic web pages do not infringe, literally or under the doctrine of equivalents, either directly, indirectly or willfully, any valid and enforceable claim of the epicRealm patents;

b. Declaring that the claims of the epicRealm patents are invalid;

c. Decreeing this case an "exceptional case" within the meaning of 35 U.S.C. § 285 and awarding reasonable attorneys' fees to Oracle; and

d. Awarding Oracle such other costs and further relief as the Court deems just and proper.

MORRIS, NICHOLS, ARSHT & TUNNELL LLP

/s/ Mary B. Graham

Mary B. Graham (#2256)
1201 North Market Street
P.O. Box 1347
Wilmington, DE 19899
302.658.9200

Attorneys for Plaintiffs
Oracle Corporation and Oracle U.S.A. Inc.

OF COUNSEL:

James G. Gilliland
Joseph A. Greco
Chad E. King
Robert J. Artuz
Marilee S. Chan
TOWNSEND AND TOWNSEND AND CREW LLP
379 Lytton Avenue
Palo Alto, CA 94301
650.326.2400

Dated: June 30, 2006
526725

US005894554A

United States Patent [19]

Lowery et al.

[11] Patent Number: 5,894,554

[45] Date of Patent: Apr. 13, 1999

[54] **SYSTEM FOR MANAGING DYNAMIC WEB PAGE GENERATION REQUESTS BY INTERCEPTING REQUEST AT WEB SERVER AND ROUTING TO PAGE SERVER THEREBY RELEASING WEB SERVER TO PROCESS OTHER REQUESTS**

[75] Inventors: **Keith Lowery**, Richardson; **Andrew B. Levine**, Plano; **Ronald L. Howell**, Rowlett, all of Tex.

[73] Assignee: **InfoSpinner, Inc.**, Richardson, Tex.

[21] Appl. No.: **08/636,477**

[22] Filed: **Apr. 23, 1996**

[51] **Int. Cl.**[6] **G06F 13/14**; G06F 13/20

[52] **U.S. Cl.** **395/200.33**; 395/200.68; 395/200.75; 395/680; 707/10; 707/104

[58] **Field of Search** 358/400; 395/800, 395/700, 200.68, 200.75, 200.53, 680, 200.33; 707/104, 10

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,866,706	9/1989	Christophersen et al.	370/85.7
5,341,499	8/1994	Doragh	395/700
5,392,400	2/1995	Berkowitz et al.	395/200
5,404,522	4/1995	Carmon et al.	395/650
5,404,523	4/1995	DellaFera et al.	395/650
5,404,527	4/1995	Irwin et al.	395/700
5,452,460	9/1995	Distelberg et al.	395/700
5,532,838	7/1996	Barbari	358/400
5,751,956	5/1998	Kirsch	395/200.33
5,761,673	6/1998	Bookman et al.	707/104

OTHER PUBLICATIONS

"Beyond the Web: Excavating the Real World Via Mosaic", Goldberg et al, Second International WWW, Oct. 17, 1994.

PCT International Search Report, Aug. 21, 1997.

Primary Examiner—Thomas C. Lee

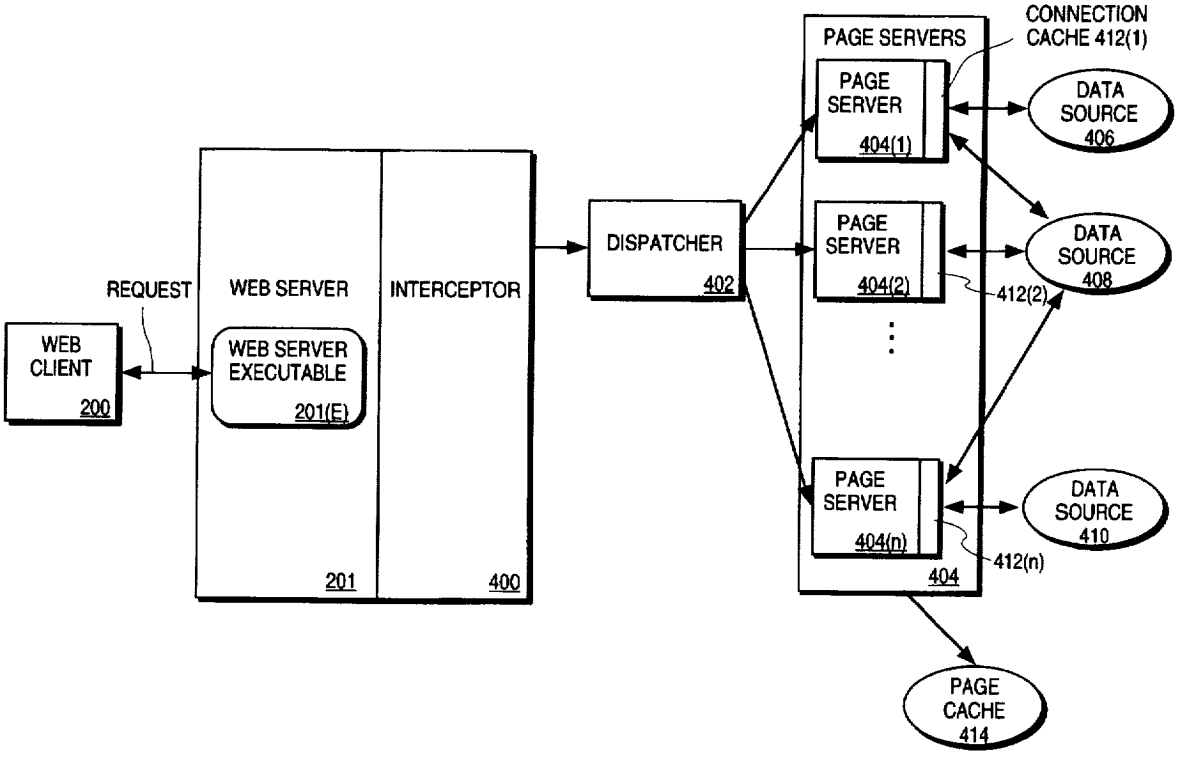
Assistant Examiner—Rehana Perveen

Attorney, Agent, or Firm—Blakely, Sokoloff, Taylor & Zafman LLP

[57] **ABSTRACT**

The present invention teaches a method and apparatus for creating and managing custom Web sites. Specifically, one embodiment of the present invention claims a computer-implemented method for managing a dynamic Web page generation request to a Web server, the computer-implemented method comprising the steps of routing the request from the Web server to a page server, the page server receiving the request and releasing the Web server to process other requests, processing the request, the processing being performed by the page server concurrently with the Web server, as the Web server processes the other requests, and dynamically generating a Web page in response to the request, the Web page including data dynamically retrieved from one or more data sources.

11 Claims, 5 Drawing Sheets



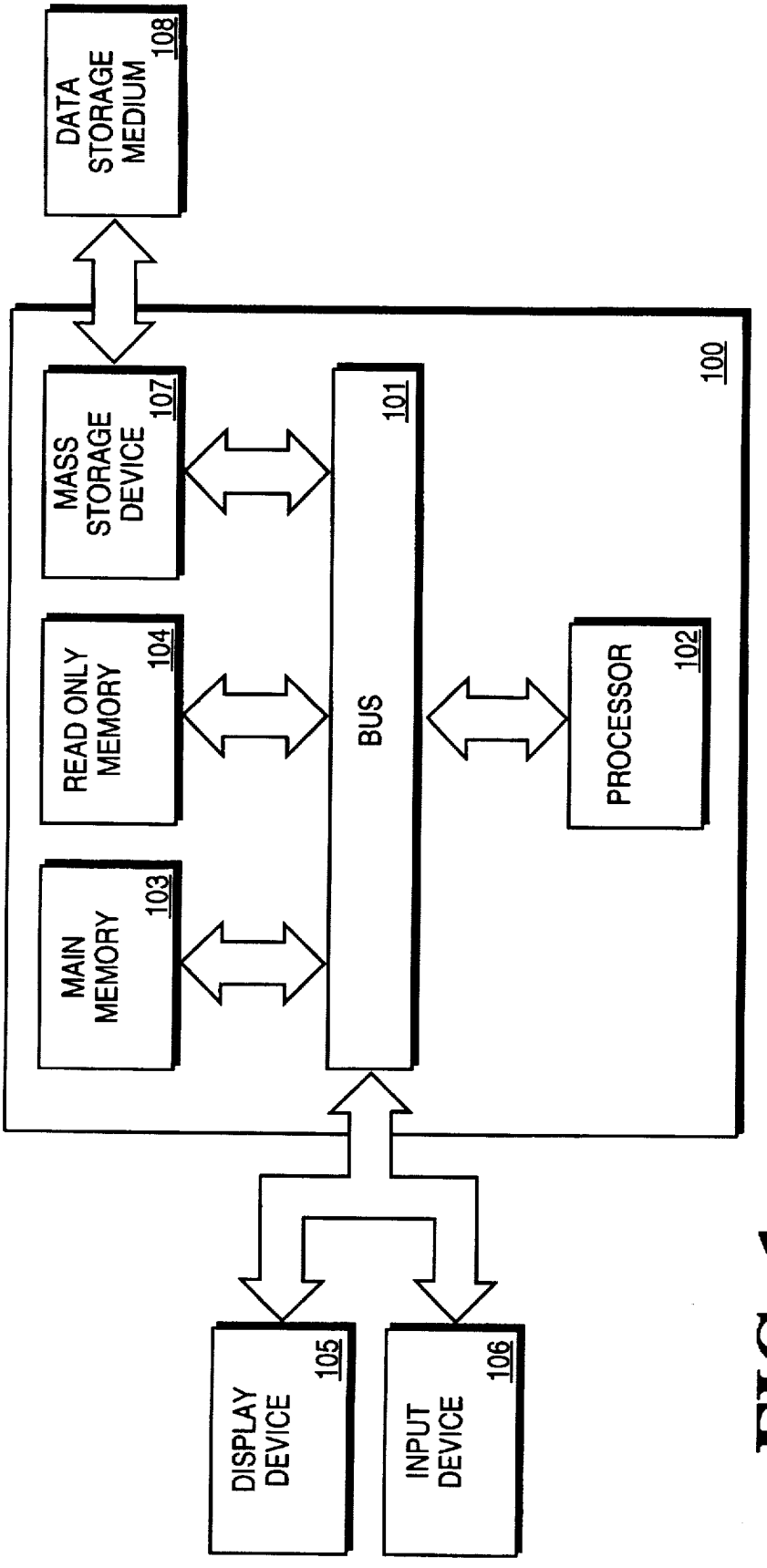

DATA STORAGE MEDIUM
108
100
MASS STORAGE DEVICE
107
101
READ ONLY MEMORY
104
BUS
PROCESSOR
102
MAIN MEMORY
103
DISPLAY DEVICE
105
INPUT DEVICE
106

FIG. 1

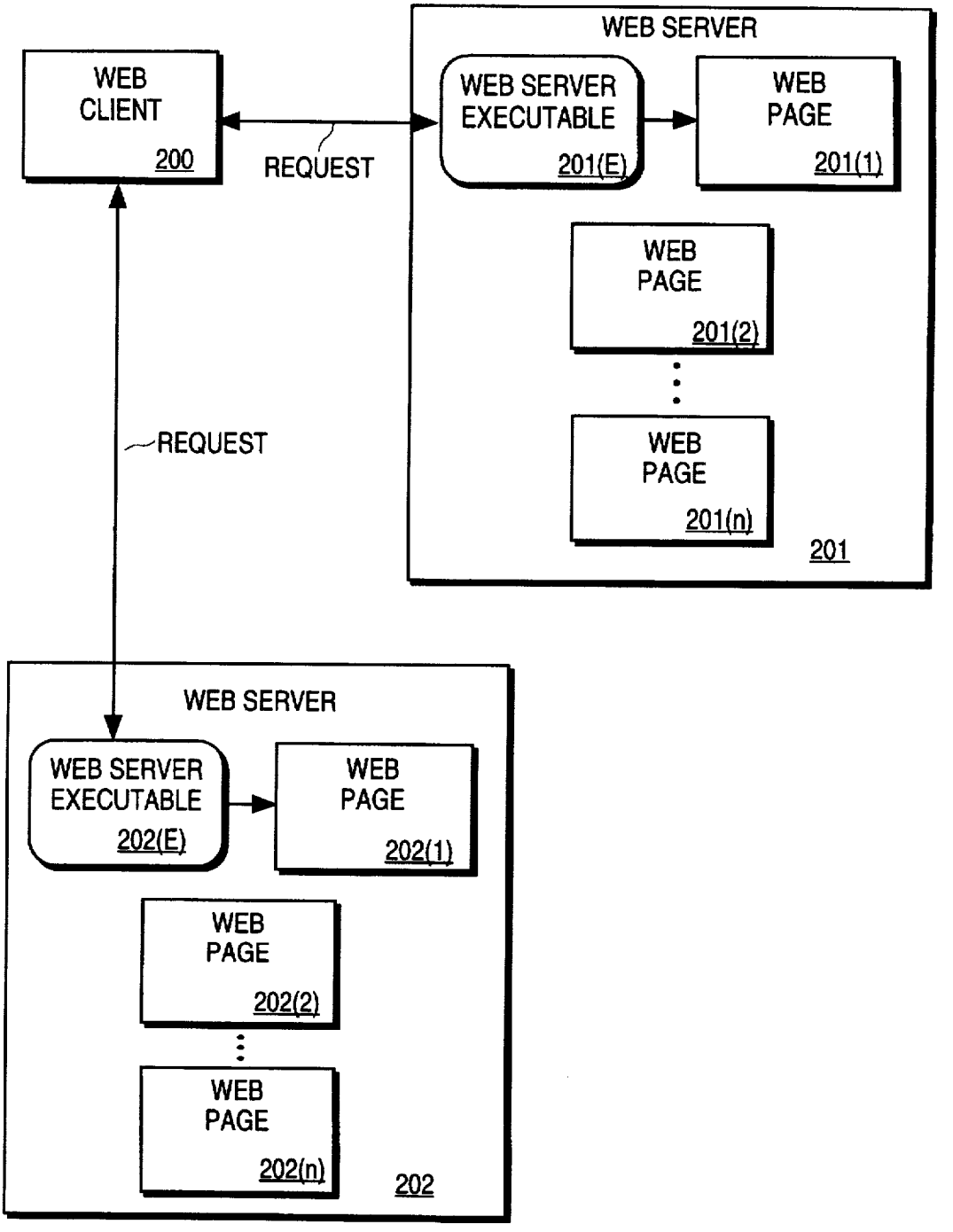


FIG. 2 (PRIOR ART)

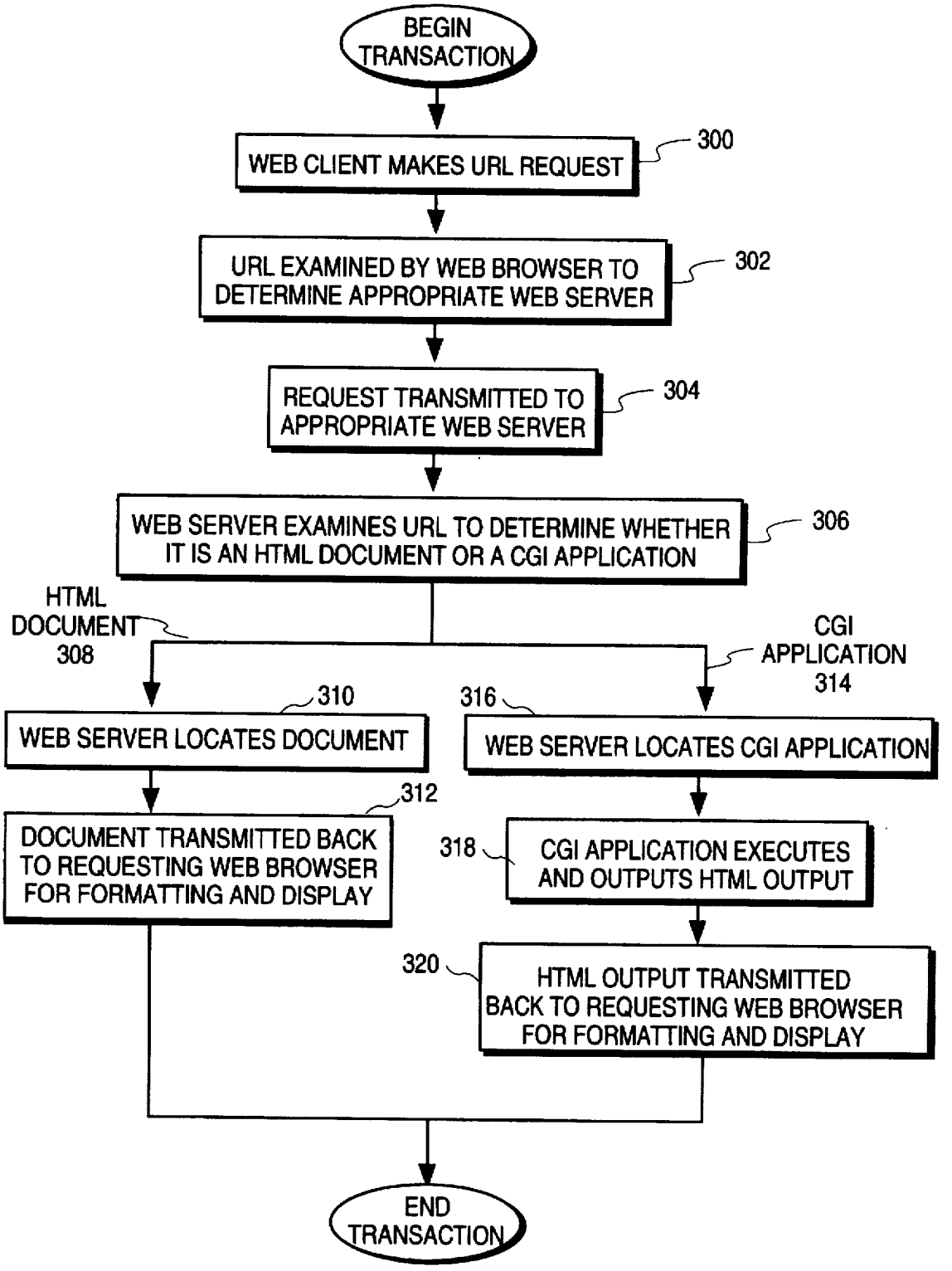


FIG. 3 (PRIOR ART)

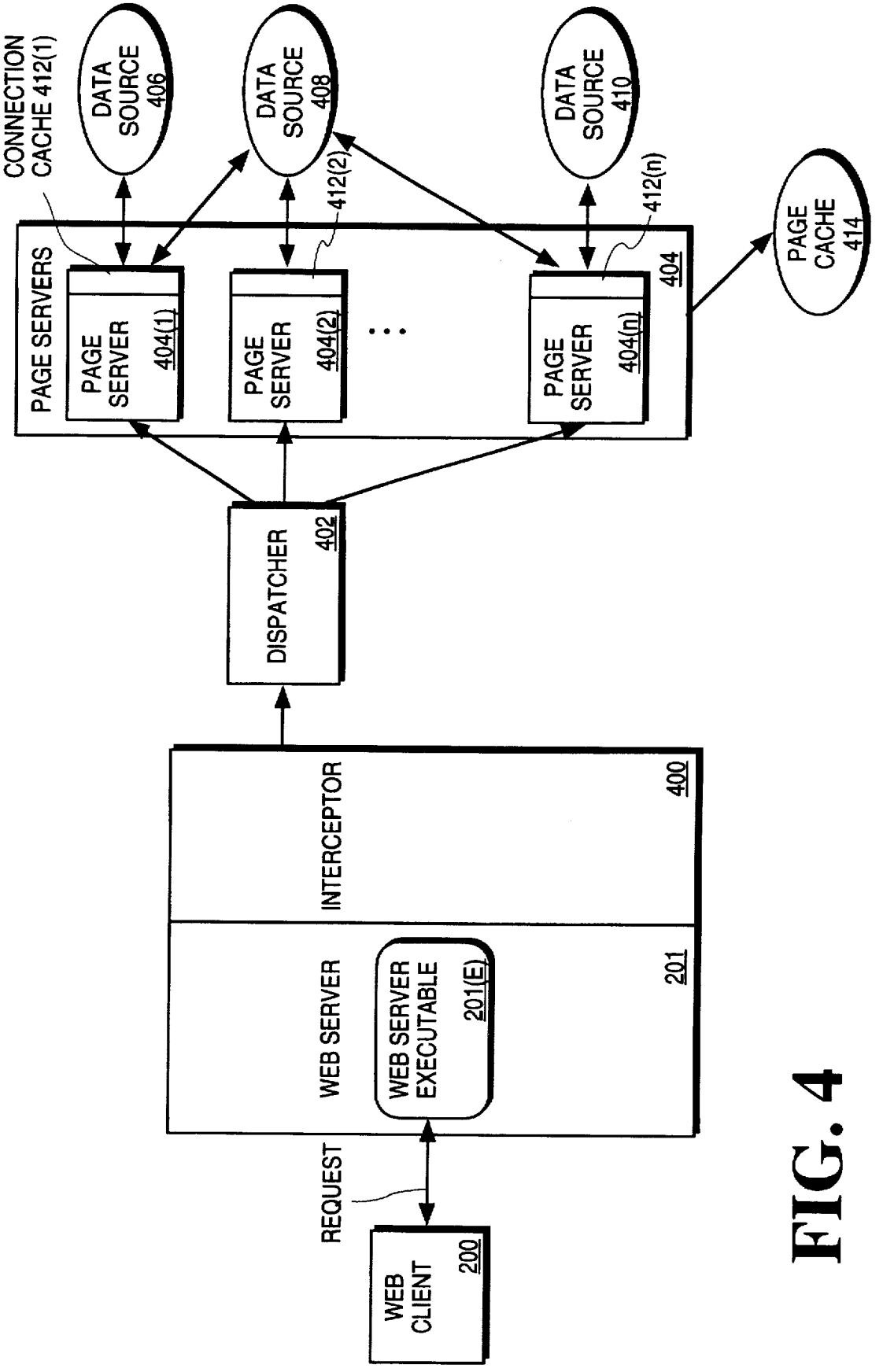

CONNECTION CACHE 412(1)
DATA SOURCE 406
DATA SOURCE 408
DATA SOURCE 410
412(2)
412(n)
PAGE CACHE 414
PAGE SERVERS
PAGE SERVER 404(1)
PAGE SERVER 404(2)
PAGE SERVER 404(n)
404
DISPATCHER 402
INTERCEPTOR 400
WEB SERVER
WEB SERVER EXECUTABLE 201(E)
201
REQUEST
WEB CLIENT 200

FIG. 4

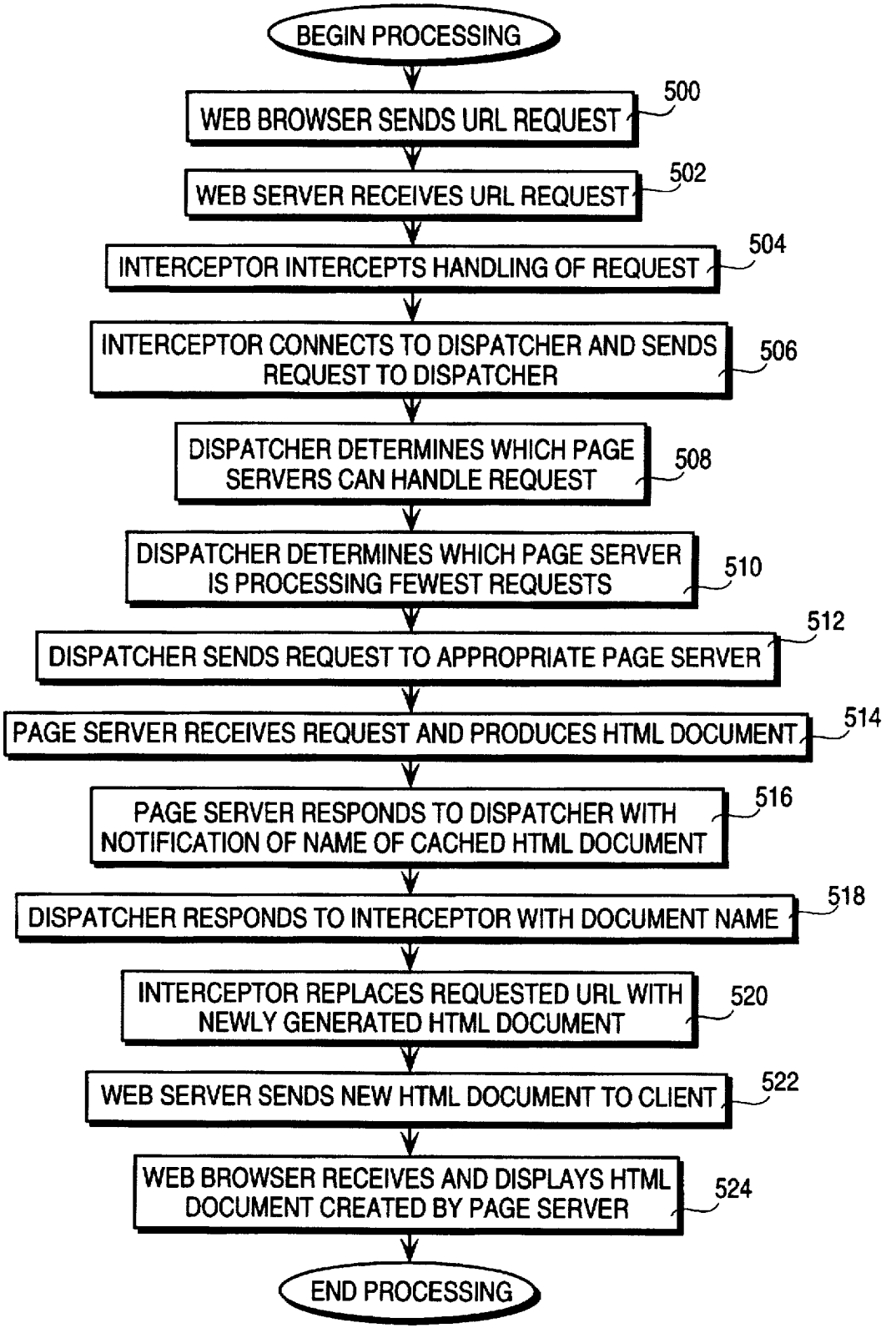

BEGIN PROCESSING
WEB BROWSER SENDS URL REQUEST
500
WEB SERVER RECEIVES URL REQUEST
502
INTERCEPTOR INTERCEPTS HANDLING OF REQUEST
504
INTERCEPTOR CONNECTS TO DISPATCHER AND SENDS REQUEST TO DISPATCHER
506
DISPATCHER DETERMINES WHICH PAGE SERVERS CAN HANDLE REQUEST
508
DISPATCHER DETERMINES WHICH PAGE SERVER IS PROCESSING FEWEST REQUESTS
510
DISPATCHER SENDS REQUEST TO APPROPRIATE PAGE SERVER
512
PAGE SERVER RECEIVES REQUEST AND PRODUCES HTML DOCUMENT
514
PAGE SERVER RESPONDS TO DISPATCHER WITH NOTIFICATION OF NAME OF CACHED HTML DOCUMENT
516
DISPATCHER RESPONDS TO INTERCEPTOR WITH DOCUMENT NAME
518
INTERCEPTOR REPLACES REQUESTED URL WITH NEWLY GENERATED HTML DOCUMENT
520
WEB SERVER SENDS NEW HTML DOCUMENT TO CLIENT
522
WEB BROWSER RECEIVES AND DISPLAYS HTML DOCUMENT CREATED BY PAGE SERVER
524
END PROCESSING

FIG. 5

SYSTEM FOR MANAGING DYNAMIC WEB PAGE GENERATION REQUESTS BY INTERCEPTING REQUEST AT WEB SERVER AND ROUTING TO PAGE SERVER THEREBY RELEASING WEB SERVER TO PROCESS OTHER REQUESTS

FIELD OF THE INVENTION

The present invention relates to the field of Internet technology. Specifically, the present invention relates to the creation and management of custom World Wide Web sites.

DESCRIPTION OF RELATED ART

The World Wide Web (the Web) represents all of the computers on the Internet that offer users access to information on the Internet via interactive documents or Web pages. These Web pages contain hypertext links that are used to connect any combination of graphics, audio, video and text, in a non-linear, non-sequential manner. Hypertext links are created using a special software language known as HyperText Mark-Up Language (HTML).

Once created, Web pages reside on the Web, on Web servers or Web sites. A Web site can contain numerous Web pages. Web client machines running Web browsers can access these Web pages at Web sites via a communications protocol known as HyperText Transport Protocol (HTTP). Web browsers are software interfaces that run on World Wide Web clients to allow access to Web sites via a simple user interface. A Web browser allows a Web client to request a particular Web page from a Web site by specifying a Uniform Resource Locator (URL). A URL is a Web address that identifies the Web page and its location on the Web. When the appropriate Web site receives the URL, the Web page corresponding to the requested URL is located, and if required, HTML output is generated. The HTML output is then sent via HTTP to the client for formatting on the client's screen.

Although Web pages and Web sites are extremely simple to create, the proliferation of Web sites on the Internet highlighted a number of problems. The scope and ability of a Web page designer to change the content of the Web page was limited by the static nature of Web pages. Once created, a Web page remained static until it was manually modified. This in turn limited the ability of Web site managers to effectively manage their Web sites.

The Common Gateway Interface (CGI) standard was developed to resolve the problem of allowing dynamic content to be included in Web pages. CGI "calls" or procedures enable applications to generate dynamically created HTML output, thus creating Web pages with dynamic content. Once created, these CGI applications do not have to be modified in order to retrieve "new" or dynamic data. Instead, when the Web page is invoked, CGI "calls" or procedures are used to dynamically retrieve the necessary data and to generate a Web page.

CGI applications also enhanced the ability of Web site administrators to manage Web sites. Administrators no longer have to constantly update static Web pages. A number of vendors have developed tools for CGI based development, to address the issue of dynamic Web page generation. Companies like Spider™ and Bluestone™, for example, have each created development tools for CGI-based Web page development. Another company, Haht Software™, has developed a Web page generation tool that uses a BASIC-like scripting language, instead of a CGI scripting language.

Tools that generate CGI applications do not, however, resolve the problem of managing numerous Web pages and requests at a Web site. For example, a single company may maintain hundreds of Web pages at their Web site. Current Web server architecture also does not allow the Web server to efficiently manage the Web page and process Web client requests. Managing these hundreds of Web pages in a coherent manner and processing all requests for access to the Web pages is thus a difficult task. Existing development tools are limited in their capabilities to facilitate dynamic Web page generation, and do not address the issue of managing Web requests or Web sites.

SUMMARY OF THE INVENTION

It is therefore an object of the present invention to provide a method and apparatus for creating and managing custom Web sites. Specifically, the present invention claims a method and apparatus for managing dynamic web page generation requests.

In one embodiment, the present invention claims a computer-implemented method for managing a dynamic Web page generation request to a Web server, the computer-implemented method comprising the steps of routing the request from the Web server to a page server, the page server receiving the request and releasing the Web server to process other requests, processing the request, the processing being performed by the page server concurrently with the Web server, as the Web server processes the other requests, and dynamically generating a Web page in response to the request, the Web page including data dynamically retrieved from one or more data sources. Other embodiments also include connection caches to the one or more data sources, page caches for each page server, and custom HTML extension templates for configuring the Web page.

Other objects, features and advantages of the present invention will be apparent from the accompanying drawings and from the detailed description.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** illustrates a typical computer system in which the present invention operates.

FIG. **2** illustrates a typical prior art Web server environment.

FIG. **3** illustrates a typical prior art Web server environment in the form of a flow diagram.

FIG. **4** illustrates one embodiment of the presently claimed invention.

FIG. **5** illustrates the processing of a Web browser request in the farm of a flow diagram, according to one embodiment of the presently claimed invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The present invention relates to a method and apparatus for creating and managing custom Web sites. In the following detailed description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent to one of ordinary skill in the art, however, that these specific details need not be used to practice the present invention. In other instances, well-known structures, interfaces and processes have not been shown in detail in order not to unnecessarily obscure the present invention.

FIG. **1** illustrates a typical computer system **100** in which the present invention operates. The preferred embodiment of

the present invention is implemented on an IBM™ Personal Computer manufactured by IBM Corporation of Armonk, N.Y. An alternate embodiment may be implemented on an RS/6000™ Workstation manufactured by IBM Corporation of Armonk, N.Y. It will be apparent to those of ordinary skill in the art that other computer system architectures may also be employed.

In general, such computer systems as illustrated by FIG. **1** comprise a bus **101** for communicating information, a processor **102** coupled with the bus **101** for processing information, main memory **103** coupled with the bus **101** for storing information and instructions for the processor **102**, a read-only memory **104** coupled with the bus **101** for storing static information and instructions for the processor **102**, a display device **105** coupled with the bus **101** for displaying information for a computer user, an input device **106** coupled with the bus **101** for communicating information and command selections to the processor **102**, and a mass storage device **107**, such as a magnetic disk and associated disk drive, coupled with the bus **101** for storing information and instructions. A data storage medium **108** containing digital information is configured to operate with mass storage device **107** to allow processor **102** access to the digital information on data storage medium **108** via bus **101**.

Processor **102** may be any of a wide variety of general purpose processors or microprocessors such as the Pentium™ microprocessor manufactured by Intel™ Corporation or the RS/6000™ processor manufactured by IBM Corporation. It will be apparent to those of ordinary skill in the art, however, that other varieties of processors may also be used in a particular computer system. Display device **105** may be a liquid crystal device, cathode ray tube (CRT), or other suitable display device. Mass storage device **107** may be a conventional hard disk drive, floppy disk drive, CD-ROM drive, or other magnetic or optical data storage device for reading and writing information stored on a hard disk, a floppy disk, a CD-ROM a magnetic tape, or other magnetic or optical data storage medium. Data storage medium **108** may be a hard disk, a floppy disk, a CD-ROM, a magnetic tape, or other magnetic or optical data storage medium.

In general, processor **102** retrieves processing instructions and data from a data storage medium **108** using mass storage device **107** and downloads this information into random access memory **103** for execution. Processor **102**, then executes an instruction stream from random access memory **103** or read-only memory **104**. Command selections and information input at input device **106** are used to direct the flow of instructions executed by processor **102**. Equivalent input device **106** may also be a pointing device such as a conventional mouse or trackball device. The results of this processing execution are then displayed on display device **105**.

The preferred embodiment of the present invention is implemented as a software module, which may be executed on a computer system such as computer system **100** in a conventional manner. Using well known techniques, the application software of the preferred embodiment is stored on data storage medium **108** and subsequently loaded into and executed within computer system **100**. Once initiated, the software of the preferred embodiment operates in the manner described below.

FIG. **2** illustrates a typical prior art Web server environment. Web client **200** can make URL requests to Web server **201** or Web server **202**. Web servers **201** and **202** include Web server executables, **201**(E) and **202**(E) respectively, that perform the processing of Web client requests. Each Web server may have a number of Web pages **201**(1)–(n) and **202**(1)–(n). Depending on the URL specified by the Web client **200**, the request may be routed by either Web server executable **201**(E) to Web page **201** (1), for example, or from Web server executable **202**(E) to Web page **202** (1). Web client **200** can continue making URL requests to retrieve other Web pages. Web client **200** can also use hyperlinks within each Web page to "jump" to other Web pages or to other locations within the same Web page.

FIG. **3** illustrates this prior art Web server environment in the form of a flow diagram. In processing block **300**, the Web client makes a URL request. This URL request is examined by the Web browser to determine the appropriate Web server to route the request to in processing block **302**. In processing block **304** the request is then transmitted from the Web browser to the appropriate Web server, and in processing block **306** the Web server executable examines the URL to determine whether it is a HTML document or a CGI application. If the request is for an HTML document **308**, then the Web server executable locates the document in processing block **310**. The document is then transmitted back through the requesting Web browser for formatting and display in processing block **312**.

If the URL request is for a CGI application **314**, however, the Web server executable locates the CGI application in processing block **316**. The CGI application then executes and outputs HTML output in processing block **318** and finally, the HTML output is transmitted back to requesting Web browser for formatting and display in processing block **320**.

This prior art Web server environment does not, however, provide any mechanism for managing the Web requests or the Web sites. As Web sites grow, and as the number of Web clients and requests increase, Web site management becomes a crucial need.

For example, a large Web site may receive thousands of requests or "hits" in a single day. Current Web servers process each of these requests on a single machine, namely the Web server machine. Although these machines may be running "multi-threaded" operating systems that allow transactions to be processed by independent "threads," all the threads are nevertheless on a single machine, sharing a processor. As such, the Web executable thread may hand off a request to a processing thread, but both threads will still have to be handled by the processor on the Web server machine. When numerous requests are being simultaneously processed by multiple threads on a single machine, the Web server can slow down significantly and become highly inefficient. The claimed invention addresses this need by utilizing a partitioned architecture to facilitate the creation and management of custom Web sites and servers.

FIG. **4** illustrates one embodiment of the presently claimed invention. Web client **200** issues a URL request that is processed to determined proper routing. In this embodiment, the request is routed to Web server **201**. Instead of Web server executable **201**(E) processing the URL request, however, Interceptor **400** intercepts the request and routes it to Dispatcher **402**. In one embodiment, Interceptor **400** resides on the Web server machine as an extension to Web server **201**. This embodiment is appropriate for Web servers such as Netsite™ from Netscape, that support such extensions. A number of public domain Web servers, such as NCSA™ from the National Center for Supercomputing Applications at the University of Illinois, Urbana-Champaign, however, do not provide support for

this type of extension. Thus, in an alternate embodiment, Interceptor **400** is an independent module, connected via an "intermediate program" to Web server **201**. This intermediate program can be a simple CGI application program that connects Interceptor **400** to Web server **201**. Alternate intermediate programs the perform the same functionality can also be implemented.

In one embodiment of the invention, Dispatcher **402** resides on a different machine than Web server **201**. This embodiment overcomes the limitation described above, in prior art Web servers, wherein all processing is performed by the processor on a single machine. By routing the request to Dispatcher **402** residing on a different machine than the Web server executable **201**(E), the request can then be processed by a different processor than the Web server executable **201**(E). Web server executable **201**(E) is thus free to continue servicing client requests on Web server **201** while the request is processed "off-line," at the machine on which Dispatcher **402** resides.

Dispatcher **402** can, however, also reside on the same machine as the Web server. The Web site administrator has the option of configuring Dispatcher **402** on the same machine as Web server **201**, taking into account a variety of factors pertinent to a particular Web site, such as the size of the Web site, the number of Web pages and the number of hits at the Web site. Although this embodiment will not enjoy the advantage described above, namely off-loading the processing of Web requests from the Web server machine, the embodiment does allow flexibility for a small Web site to grow. For example, a small Web site administrator can use a single machine for both Dispatcher **402** and Web server **201** initially, then off-load Dispatcher **402** onto a separate machine as the Web site grows. The Web site can thus take advantage of other features of the present invention regardless of whether the site has separate machines configured as Web servers and dispatchers.

Dispatcher **402** receives the intercepted request and then dispatches the request to one of a number of Page servers **404** (1)–(n). For example, if Page server **404** (1) receives the dispatched request, it processes the request and retrieves the data from an appropriate data source, such as data source **406**, data source **408**, or data source **410**. Data sources, as used in the present application, include databases, spreadsheets, files and any other type of data repository. Page server **404** (1) can retrieve data from more than one data source and incorporate the data from these multiple data sources in a single Web page.

In one embodiment, each Page server **404**(1)–(n) resides on a separate machine on the network to distribute the processing of the request. Dispatcher **402** maintains a variety of information regarding each Page server on the network, and dispatches requests based on this information. For example, Dispatcher **402** retains dynamic information regarding the data sources that any given Page server can access. Dispatcher **402** thus examines a particular request and determines which Page servers can service the URL request. Dispatcher **402** then hands off the request to the appropriate Page server.

For example, if the URL request requires financial data from data source **408**, dispatcher **402** will first examine an information list. Dispatcher **402** may determine that Page server **404**(3), for example, has access to the requisite data in data source **408**. Dispatcher **402** will thus route the URL request to Page server **404**(3). This "connection caching" functionality is described in more detail below, under the heading "Performance."

Alternately, Dispatcher **402** also has the ability to determine whether a particular Page server already has the necessary data cached in the Page server's page cache (described in more detail below, under the heading "Performance"). Dispatcher **402** may thus determine that Page server **404**(1) and **404**(2) are both logged into Data source **408**, but that Page server **404**(2) has the financial information already cached in Page server **404**(2)'s page cache. In this case, Dispatcher **402** will route the URL request to Page server **404**(2) to more efficiently process the request.

Finally, Dispatcher **402** may determine that a number or all Page servers **404**(1)–(n) are logged into Data source **408**. In this scenario, Dispatcher **402** can examine the number of requests that each Page server is servicing and route the request to the least busy page server. This "load balancing" capability can significantly increase performance at a busy Web site and is discussed in more detail below, under the heading "Scalability".

If, for example, Page server **404**(2), receives the request, Page server **404**(2) will process the request. While Page server **404**(2) is processing the request, Web server executable **201**(E) can concurrently process other Web client requests. This partitioned architecture thus allows both Page server **404**(2) and Web server executable **201**(E) to simultaneously process different requests, thus increasing the efficiency of the Web site. Page server **404**(2) dynamically generates a Web page in response to the Web client request, and the dynamic Web page is then either transmitted back to requesting Web client **200** or stored on a machine that is accessible to Web server **201**, for later retrieval.

One embodiment of the claimed invention also provides a Web page designer with HTML extensions, or "dyna" tags. These dyna tags provide customized HTML functionality to a Web page designer, to allow the designer to build customized HTML templates that specify the source and placement of retrieved data. For example, in one embodiment, a "dynatext" HTML extension tag specifies a data source and a column name to allow the HTML template to identify the data source to log into and the column name from which to retrieve data. Alternatively, "dyna-anchor" tags allow the designer to build hyperlink queries while "dynablock" tags provide the designer with the ability to iterate through blocks of data. Page servers use these HTML templates to create dynamic Web pages. Then, as described above, these dynamic Web pages are either transmitted back to requesting Web client **200** or stored on a machine that is accessible to Web server **201**, for later retrieval.

The presently claimed invention provides numerous advantages over prior art Web servers, including advantages in the areas of performance, security, extensibility and scalability.

Performance

One embodiment of the claimed invention utilizes connection caching and page caching to improve performance. Each Page server can be configured to maintain a cache of connections to numerous data sources. For example, as illustrated in FIG. **4**, Page server **404**(1) can retrieve data from data source **406**, data source **408** or data source **410**. Page server **404**(1) can maintain connection cache **412**(1), containing connections to each of data source **406**, data source **408** and data source **410**, thus eliminating connect times from the Page servers to those data sources.

Additionally, another embodiment of the present invention supports the caching of finished Web pages, to optimize

the performance of the data source being utilized. This "page caching" feature, illustrated in FIG. **4** as Page cache **414**, allows the Web site administrator to optimize the performance of data sources by caching Web pages that are repeatedly accessed. Once the Web page is cached, subsequent requests or "hits" will utilize the cached Web page rather than re-accessing the data source. This can radically improve the performance of the data source.

Security

The present invention allows the Web site administrator to utilize multiple levels of security to manage the Web site. In one embodiment, the Page server can utilize all standard encryption and site security features provided by the Web server. In another embodiment, the Page server can be configured to bypass connection caches **412**(1)–(n), described above, for a particular data source and to require entry of a user-supplied identification and password for the particular data source the user is trying to access.

Additionally, another embodiment of the presently claimed invention requires no real-time access of data sources. The Web page caching ability, described above, enables additional security for those sites that want to publish non-interactive content from internal information systems, but do not want real-time Internet accessibility to those internal information systems. In this instance, the Page server can act as a "replication and staging gent" and create Web pages in batches, rather than in real-time. These "replicated" Web pages are then "staged" for access at a later time, and access o the Web pages in this scenario is possible even if the Page server and dispatcher are not present later.

In yet another embodiment, the Page server can make a single pass through a Web library, and compile a Web site that exists in the traditional form of separately available files. A Web library is a collection of related Web books and Web pages. More specifically, the Web library is a hierarchical organization of Web document templates, together with all the associated data source information. Information about an entire Web site is thus contained in a single physical file, thus simplifying the problem of deploying Web sites across multiple Page servers. The process of deploying the Web site in this embodiment is essentially a simple copy of a single file.

Extensibility

One embodiment of the present invention provides the Web site administrator with Object Linking and Embedding (OLE) 2.0 extensions to extend the page creation process. These OLE 2.0 extensions also allow information submitted over the Web to be processed with user-supplied functionality. Utilizing development tools such as Visual Basic, Visual C++ or PowerBuilder that support the creation of OLE 2.0 automation, the Web site administrator can add features and modify the behavior of the Page servers described above. This extensibility allows one embodiment of the claimed invention to be incorporated with existing technology to develop an infinite number of custom web servers.

For example, OLE 2.0 extensions allow a Web site administrator to encapsulate existing business rules in an OLE 2.0 automation interface, to be accessed over the Web. One example of a business rule is the steps involved in the payoff on an installment or mortgage loan. The payoff may involve, for example, taking into account the current balance, the date and the interest accrued since the last payment. Most organizations already have this type of business rule implemented using various applications, such as Visual Basic for client-server environments, or CICS programs on mainframes. If these applications are OLE 2.0 compliant, the Page server "dynaobject" HTML extension tag can be used to encapsulated the application in an OLE 2.0 automation interface. The Page server is thus extensible, and can incorporate the existing application with the new Page server functionality.

Scalability

One embodiment of the claimed invention allows "plug and play" scalability. As described above, referring to FIG. **4**, Dispatcher **402** maintains information about all the Page servers configured to be serviced by Dispatcher **402**. Any number of Page servers can thus be "plugged" into the configuration illustrated in FIG. **4**, and the Page servers will be instantly activated as the information is dynamically updated in Dispatcher **402**. The Web site administrator can thus manage the overhead of each Page server and modify each Page server's load, as necessary, to improve performance. In this manner, each Page server will cooperate with other Page servers within a multi-server environment. Dispatcher **402** can examine the load on each Page server and route new requests according to each Page server's available resources. This "load-balancing" across multiple Page servers can significantly increase a Web site's performance.

FIG. **5** illustrates the processing of a Web browser request in the form of a flow diagram, according to one embodiment of the presently claimed invention. A Web browser sends a URL request to a Web server in processing block **500**. In processing block **502**, the Web server receives the URL request, and an interceptor then intercepts the handling of the request in processing block **504**. The interceptor connects to a dispatcher and sends the URL request to the dispatcher in processing block **506**. In processing block **508**, the dispatcher determines which Page servers can handle the request. The dispatcher also determines which Page server is processing the fewest requests in processing block **510**, and in processing block **512**, the dispatcher sends the URL request to an appropriate Page server. The Page server receives the request and produces an HTML document in processing block **514**. The Page server then responds to the dispatcher with notification of the name of the cached HTML document in processing block **516**. In processing block **518**, the dispatcher responds to the interceptor with the document name, and the interceptor then replaces the requested URL with the newly generated HTML document in processing block **520**. The Web server then sends the new HTML document to the requesting client in processing block **522**. Finally, the Web browser receives and displays the HTML document created by the Page server at processing block **524**.

Thus, a method and apparatus for creating and managing custom Web sites is disclosed. These specific arrangements and methods described herein are merely illustrative of the principles of the present invention. Numerous modifications in form and detail may be made by those of ordinary skill in the art without departing from the scope of the present invention. Although this invention has been shown in relation to a particular preferred embodiment, it should not be considered so limited. Rather, the present invention is limited only by the scope of the appended claims.

We claim:

1. A computer-implemented method for managing a dynamic Web page generation request to a Web server, said computer-implemented method comprising the steps of:

routing said request from said Web server to a page server, said page server receiving said request and releasing

said Web server to process other requests, wherein said routing step further includes the steps of intercepting said request at said Web server, routing said request from said Web server to a dispatcher, and dispatching said request to said page server;

processing said request, said processing being performed by said page server while said Web server concurrently processes said other requests; and

dynamically generating a Web page in response to said request, said Web page including data dynamically retrieved from one or more data sources.

2. The computer-implemented method in claim **1** wherein said step of processing said request includes the step of identifying said one or more data sources from which to retrieve said data.

3. The computer-implemented method in claim **2** wherein said step of dynamically generating said Web page includes the step of dynamically retrieving said data from said one or more data sources.

4. The computer-implemented method in claim **3** wherein said step of processing said request includes the step of said page server maintaining a connection cache to said one or more data sources.

5. The computer-implemented method in claim **3** wherein said step of processing said request includes the step of logging into said one or more data sources.

6. The computer-implemented method in claim **3** wherein said step of dynamically generating said Web page includes the step of maintaining a page cache containing said Web page.

7. The computer-implemented method in claim **3** wherein said page server includes custom HTML extension templates for configuring said Web page.

8. The computer-implemented method in claim **7** wherein said step of processing said request further includes the step of inserting said dynamically retrieved data from said one or more data sources into said custom HTML extension templates.

9. A networked system for managing a dynamic Web page generation request, said system comprising:

one or more data sources;

a page server having a processing means;

a first computer system including means for generating said request; and

a second computer system including means for receiving said request from said first computer, said second computer system also including a router, said router routing said request from said second computer system to said page server, wherein said routing further includes intercepting said request at said second computer, routing said request from said second computer to a dispatcher, and dispatching said request to said page server said page server receiving said request and releasing said second computer system to process other requests, said page server processing means processing said request and dynamically generating a Web page in response to said request, said Web page including data dynamically retrieved from said one or more data sources.

10. The networked system in claim **9** wherein said router in said second computer system includes:

an interceptor intercepting said request at said second computer system and routing said request; and

a dispatcher receiving said routed request from said interceptor and dispatching said request to said page server.

11. A machine readable medium having stored thereon data representing sequences of instructions, which when executed by a computer system, cause said computer system to perform the steps of:

routing a dynamic Web page generation request from a Web server to a page server, said page server receiving said request and releasing said Web server to process other requests wherein said routing step further includes the steps of intercepting said request at said Web server, routing said request from said Web server to a dispatcher, and dispatching said request to said page server;

processing said request, said processing being performed by said page server while said Web server concurrently processes said other requests; and

dynamically generating a Web page, said Web page including data retrieved from one or more data sources.

* * * * *

US006415335B1

(12) **United States Patent**
Lowery et al.

(10) **Patent No.: US 6,415,335 B1**
(45) **Date of Patent: *Jul. 2, 2002**

(54) **SYSTEM AND METHOD FOR MANAGING DYNAMIC WEB PAGE GENERATION REQUESTS**

(75) Inventors: **Keith Lowery**, Richardson; **Andrew B. Levine**, Plano; **Ronald L. Howell**, Rowlett, all of TX (US)

(73) Assignee: **epicRealm Operating Inc.**, Richardson, TX (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **09/234,048**

(22) Filed: **Jan. 19, 1999**

Related U.S. Application Data

(62) Division of application No. 08/636,477, filed on Apr. 23, 1996, now Pat. No. 5,894,554.

(51) **Int. Cl.**[7] **G06F 13/14**; G06F 13/20

(52) **U.S. Cl.** **710/5**; 710/7; 709/219; 709/223; 709/238

(58) **Field of Search** 709/238, 223, 709/219; 710/5, 7, 20–21

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,866,706	A		9/1989	Christophersen et al. .. 370/85.7
5,341,499	A		8/1994	Doragh 395/700
5,392,400	A		2/1995	Berkowitz et al. 395/200
5,404,522	A		4/1995	Carmon et al. 395/650
5,404,523	A		4/1995	DellaFera et al. 395/650
5,404,527	A	*	4/1995	Irwin et al. 395/700
5,452,460	A		9/1995	Distelberg et al. 395/700
5,532,838	A		7/1996	Barbari 358/400
5,701,463	A	*	12/1997	Malcolm 395/610
5,751,956	A		5/1998	Kirsch 395/200.33
5,752,246	A	*	5/1998	Rogers et al. 707/10
5,754,772	A	*	5/1998	Leaf 395/200.33
5,761,673	A	*	6/1998	Bookman et al. 707/104
5,774,660	A		6/1998	Brendel et al. 395/200.31
5,774,668	A		6/1998	Choquier et al. 395/200.53

OTHER PUBLICATIONS

Hoffner, 'Inter–operability and distributed application platform design', Web URL:http://www.ansa.co.uk/, 1995, pp. 342–356.*

Mourad et al, 'Scalable Web Server Architectures', IEEE, Jun. 1997, pp. 12–16.*

Dias et al, 'A Scalable and Highly Available Web Server', IEEE, 1996, pp. 85–92.*

'Single System Image and Load Balancing for Network Access to a Loosely Coupled Complex', IBM TDB, vol. 34, Feb. 1992, pp. 464–467.*

Dias, Daniel M., et al.; A Scalable and Highly Available Web Server; IBM Research Division; T.J. Watson Research Center; 7 pages.

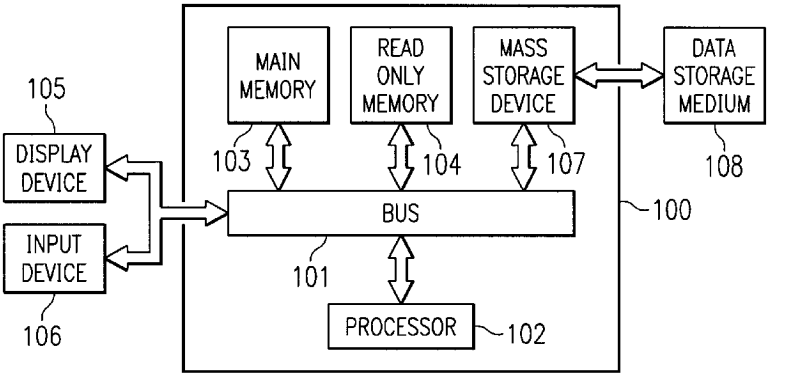
(List continued on next page.)

Primary Examiner—Jeffrey Gaffin
Assistant Examiner—Rehana Perveen
(74) *Attorney, Agent, or Firm*—Baker Botts L.L.P.

(57) **ABSTRACT**

The present invention teaches a method and apparatus for creating and managing custom Web sites. Specifically, one embodiment of the present invention claims a computer-implemented method for managing a dynamic Web page generation request to a Web server, the computer-implemented method comprising the steps of routing the request from the Web server to a page server, the page server receiving the request and releasing the Web server to process other requests, processing the request, the processing being performed by the page server concurrently with the Web server, as the Web server processes the other requests, and dynamically generating a Web page in response to the request, the Web page including data dynamically retrieved from one or more data sources.

29 Claims, 4 Drawing Sheets



OTHER PUBLICATIONS

Andresen, Daniel, Et Al.; Scalability Issues for High Performance Digital Libraries on the World Wide Web; Department of Computer Science; University of California at Santa Barbara; 10 pages.

Andresen, Daniel, Et Al.; SWEB: Towards a Scalable World Wide Web Server on Multicomputers; Department of Computer Science; University of California at Santa Barbara; 7 pages.

Holmedahl, Vegard; Et Al.; Cooperative Caching of Dynamic Content on a Distributed Web Server; Department of Computer Science; University of California at Santa Barbara; 8 pages.

Overson, Nicole; NeXT Ships WebObjects—On Time—As Promised; Deja.com: NeXT Ships WebObjects—On Time—As Promishttp://X28..deja.com/=dnc/ST_m=ps...EXT=927585438. 1744765032&hitnum=33.

International Search Report; 7 pages; dated Aug. 21, 1997.

Birman, Kenneth P. and van Renesse, Robbert; Software for Reliable Networks; Scientific American; May 1996; pp. 64–69.

"Beyond the Web: Excavating the Real World Via Mosaic"; Goldberg et al.; Second International WWW Conference; Oct. 17, 1994.

* cited by examiner

FIG. 1

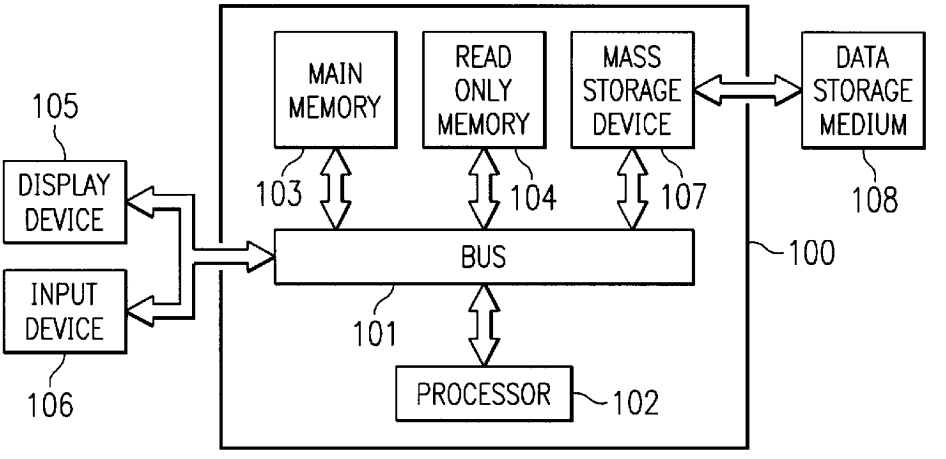


FIG. 2
(PRIOR ART)

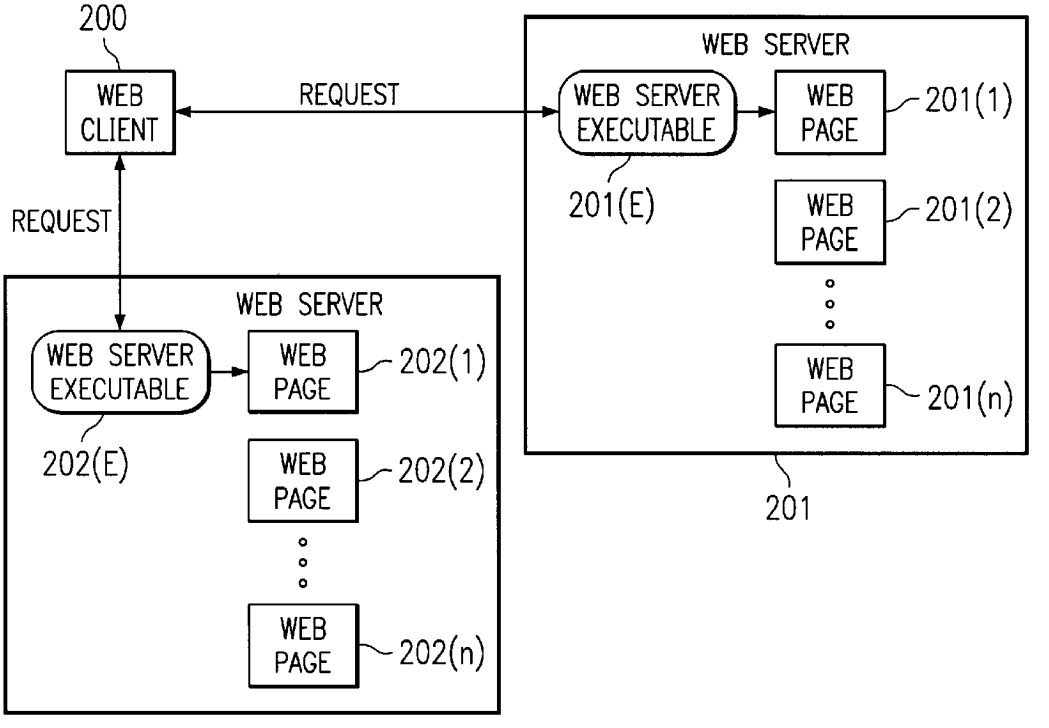
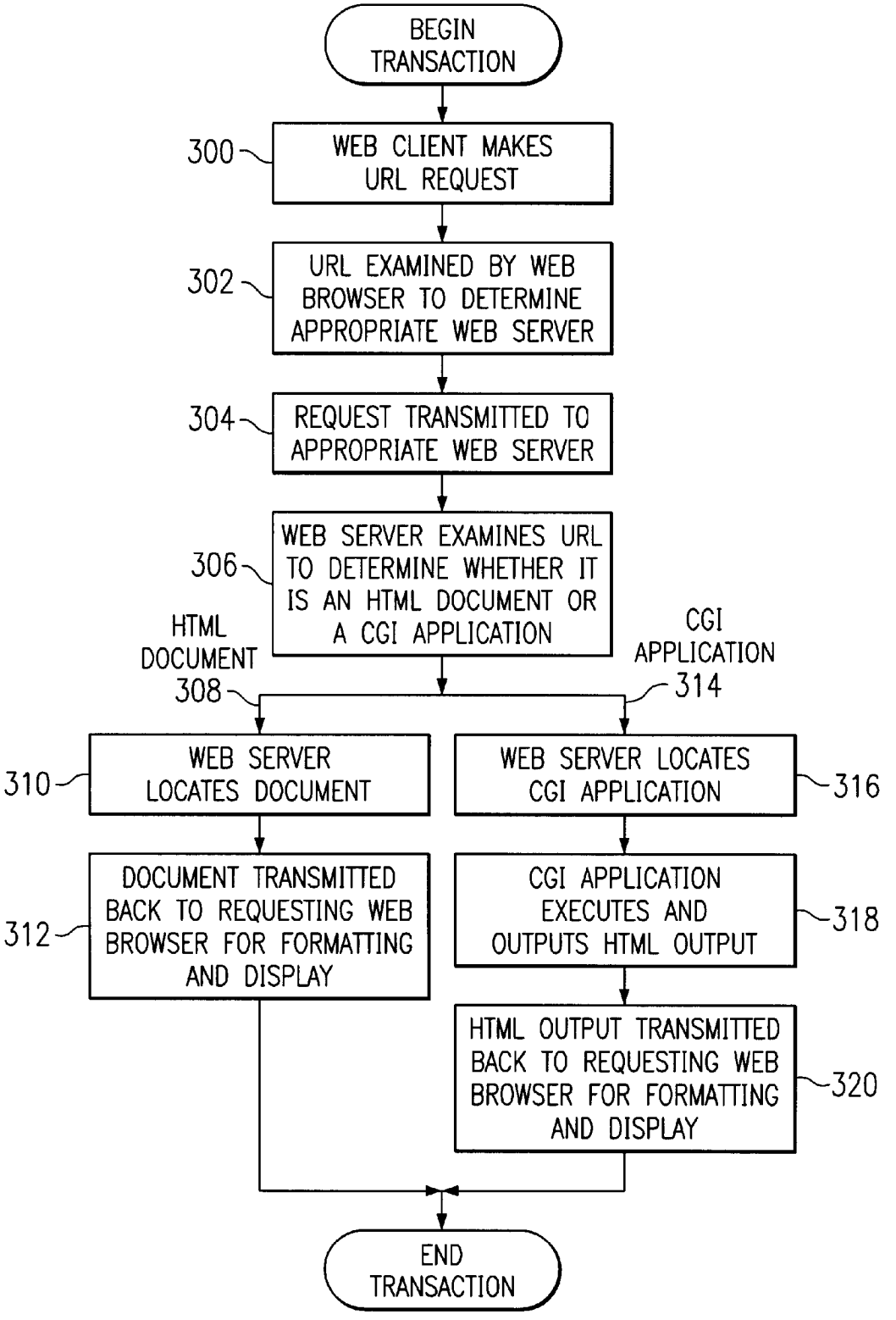


FIG. 3
(PRIOR ART)



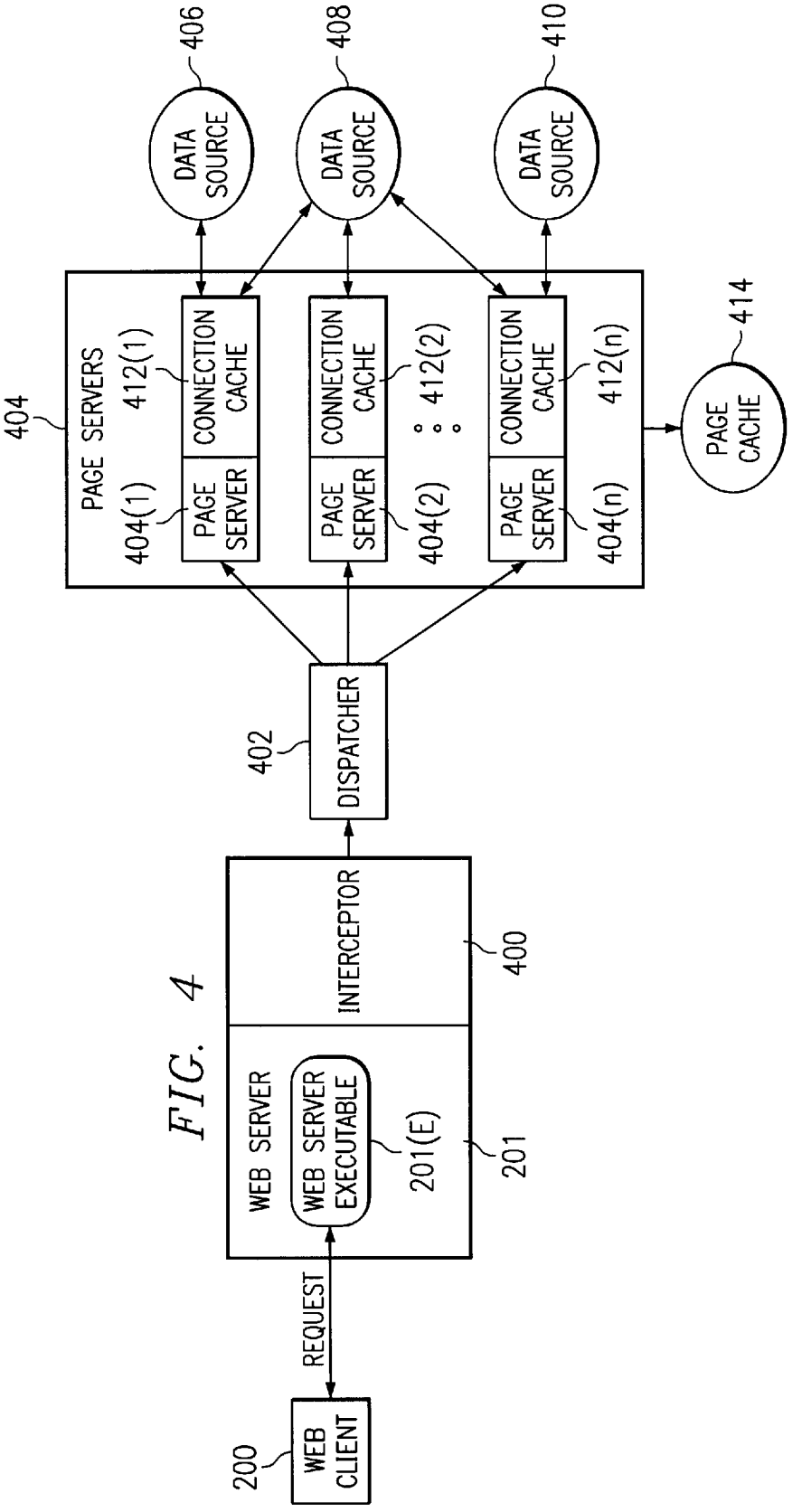
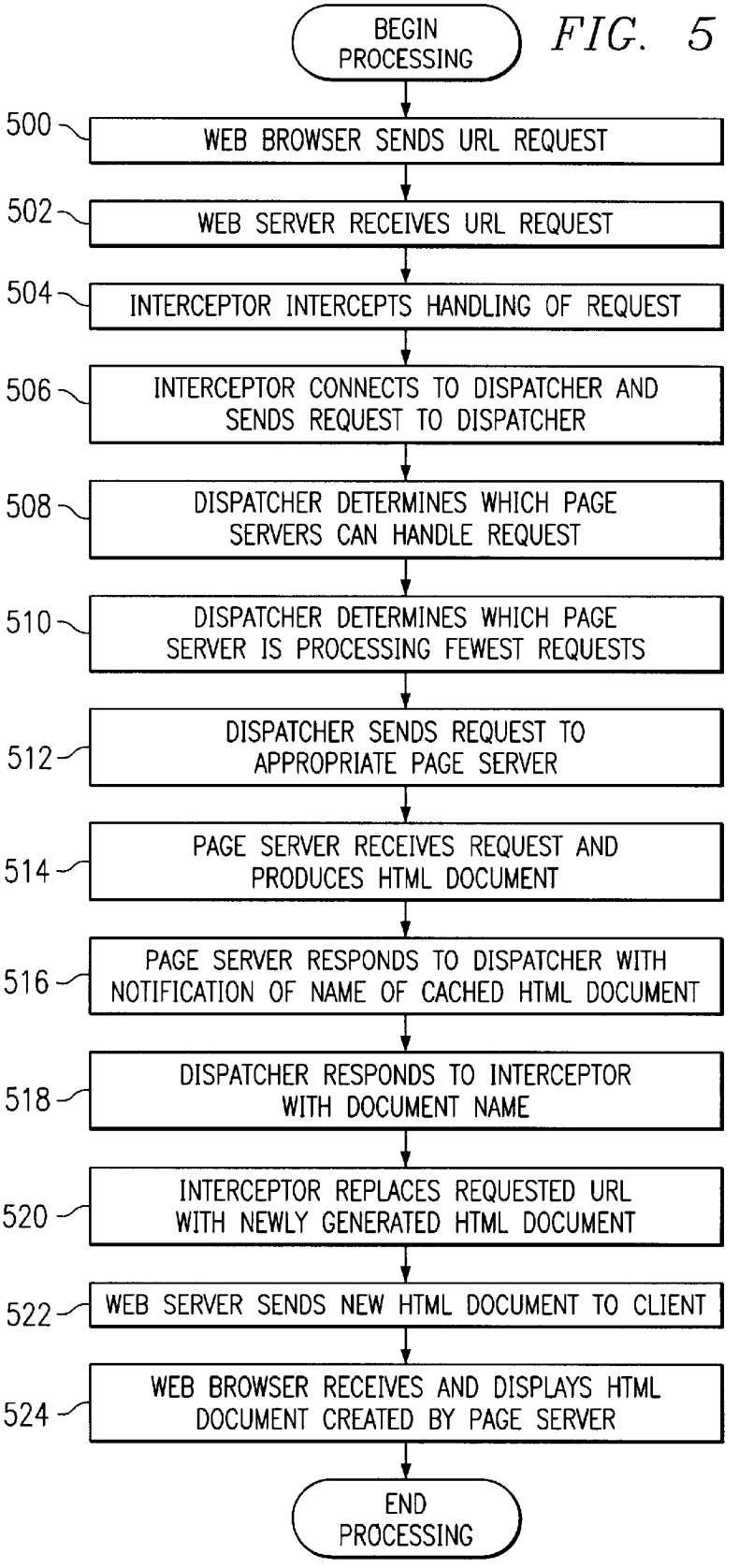

406
408
410
DATA SOURCE
DATA SOURCE
DATA SOURCE
404
PAGE SERVERS
412(1)
CONNECTION CACHE
PAGE SERVER
404(1)
CONNECTION CACHE
PAGE SERVER
404(2)
412(2)
CONNECTION CACHE
PAGE SERVER
404(n)
412(n)
414
PAGE CACHE
402
DISPATCHER
INTERCEPTOR
400
WEB SERVER
WEB SERVER EXECUTABLE
201(E)
201
REQUEST
200
WEB CLIENT

FIG. 4

FIG. 5


BEGIN PROCESSING
500
WEB BROWSER SENDS URL REQUEST
502
WEB SERVER RECEIVES URL REQUEST
504
INTERCEPTOR INTERCEPTS HANDLING OF REQUEST
506
INTERCEPTOR CONNECTS TO DISPATCHER AND SENDS REQUEST TO DISPATCHER
508
DISPATCHER DETERMINES WHICH PAGE SERVERS CAN HANDLE REQUEST
510
DISPATCHER DETERMINES WHICH PAGE SERVER IS PROCESSING FEWEST REQUESTS
512
DISPATCHER SENDS REQUEST TO APPROPRIATE PAGE SERVER
514
PAGE SERVER RECEIVES REQUEST AND PRODUCES HTML DOCUMENT
516
PAGE SERVER RESPONDS TO DISPATCHER WITH NOTIFICATION OF NAME OF CACHED HTML DOCUMENT
518
DISPATCHER RESPONDS TO INTERCEPTOR WITH DOCUMENT NAME
520
INTERCEPTOR REPLACES REQUESTED URL WITH NEWLY GENERATED HTML DOCUMENT
522
WEB SERVER SENDS NEW HTML DOCUMENT TO CLIENT
524
WEB BROWSER RECEIVES AND DISPLAYS HTML DOCUMENT CREATED BY PAGE SERVER
END PROCESSING

SYSTEM AND METHOD FOR MANAGING DYNAMIC WEB PAGE GENERATION REQUESTS

This application is a division of Ser. No. 08/636,477, filed Apr. 23, 1996, now U.S. Pat. No. 5,894,554.

FIELD OF THE INVENTION

The present invention relates to the field of Internet technology. Specifically, the present invention relates to the creation and management of custom World Wide Web sites.

DESCRIPTION OF RELATED ART

The World Wide Web (the Web) represents all of the computers on the Internet that offer users access to information on the Internet via interactive documents or Web pages. These Web pages contain hypertext links that are used to connect any combination of graphics, audio, video and text, in a non-linear, non-sequential manner. Hypertext links are created using a special software language known as HyperText Mark-Up Language (HTML).

Once created, Web pages reside on the Web, on Web servers or Web sites. A Web site can contain numerous Web pages. Web client machines running Web browsers can access these Web pages at Web sites via a communications protocol known as HyperText Transport Protocol (HTTP). Web browsers are software interfaces that run on World Wide Web clients to allow access to Web sites via a simple user interface. A Web browser allows a Web client to request a particular Web page from a Web site by specifying a Uniform Resource Locator (URL). A URL is a Web address that identifies the Web page and its location on the Web. When the appropriate Web site receives the URL, the Web page corresponding to the requested URL is located, and if required, HTML output is generated. The HTML output is then sent via HTTP to the client for formatting on the client's screen.

Although Web pages and Web sites are extremely simple to create, the proliferation of Web sites on the Internet highlighted a number of problems. The scope and ability of a Web page designer to change the content of the Web page was limited by the static nature of Web pages. Once created, a Web page remained static until it was manually modified. This in turn limited the ability of Web site managers to effectively manage their Web sites.

The Common Gateway Interface (CGI) standard was developed to resolve the problem of allowing dynamic content to be included in Web pages. CGI "calls" or procedures enable applications to generate dynamically created HTML output, thus creating Web pages with dynamic content. Once created, these CGI applications do not have to be modified in order to retrieve "new" or dynamic data. Instead, when the Web page is invoked, CGI "calls" or procedures are used to dynamically retrieve the necessary data and to generate a Web page.

CGI applications also enhanced the ability of Web site administrators to manage Web sites. Administrators no longer have to constantly update static Web pages. A number of vendors have developed tools for CGI based development, to address the issue of dynamic Web page generation. Companies like Spider™ and Bluestone™, for example, have each created development tools for CGI-based Web page development. Another company, Haht Software™, has developed a Web page generation tool that uses a BASIC-like scripting language, instead of a CGI scripting language.

Tools that generate CGI applications do not, however, resolve the problem of managing numerous Web pages and requests at a Web site. For example, a single company may maintain hundreds of Web pages at their Web site. Current Web server architecture also does not allow the Web server to efficiently manage the Web page and process Web client requests. Managing these hundreds of Web pages in a coherent manner and processing all requests for access to the Web pages is thus a difficult task. Existing development tools are limited in their capabilities to facilitate dynamic Web page generation, and do not address the issue of managing Web requests or Web sites.

SUMMARY OF THE INVENTION

It is therefore an object of the present invention to provide a method and apparatus for creating and managing custom Web sites. Specifically, the present invention claims a method and apparatus for managing dynamic web page generation requests.

In one embodiment, the present invention claims a computer-implemented method for managing a dynamic Web page generation request to a Web server, the computer-implemented method comprising the steps of routing the request from the Web server to a page server, the page server receiving the request and releasing the Web server to process other requests, processing the request, the processing being performed by the page server concurrently with the Web server, as the Web server processes the other requests, and dynamically generating a Web page in response to the request, the Web page including data dynamically retrieved from one or more data sources. Other embodiments also include connection caches to the one or more data sources, page caches for each page server, and custom HTML extension templates for configuring the Web page.

Other objects, features and advantages of the present invention will be apparent from the accompanying drawings and from the detailed description.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** illustrates a typical computer system in which the present invention operates.

FIG. **2** illustrates a typical prior art Web server environment.

FIG. **3** illustrates a typical prior art Web server environment in the form of a flow diagram.

FIG. **4** illustrates one embodiment of the presently claimed invention.

FIG. **5** illustrates the processing of a Web browser request in the form of a flow diagram, according to one embodiment of the presently claimed invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The present invention relates to a method and apparatus for creating and managing custom Web sites. In the following detailed description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent to one of ordinary skill in the art, however, that these specific details need not be used to practice the present invention. In other instances, well-known structures, interfaces and processes have not been shown in detail in order not to unnecessarily obscure the present invention.

FIG. **1** illustrates a typical computer system **100** in which the present invention operates. The preferred embodiment of

the present invention is implemented on an IBM™ Personal Computer manufactured by IBM Corporation of Armonk, New York. An alternate embodiment may be implemented on an RS/6000™ Workstation manufactured by IBM Corporation of Armonk, New York. It will be apparent to those of ordinary skill in the art that other computer system architectures may also be employed.

In general, such computer systems as illustrated by FIG. **1** comprise a bus **101** for communicating information, a processor **102** coupled with the bus **101** for processing information, main memory **103** coupled with the bus **101** for storing information and instructions for the processor **102**, a read-only memory **104** coupled with the bus **101** for storing static information and instructions for the processor **102**, a display device **105** coupled with the bus **101** for displaying information for a computer user, an input device **106** coupled with the bus **101** for communicating information and command selections to the processor **102**, and a mass storage device **107**, such as a magnetic disk and associated disk drive, coupled with the bus **101** for storing information and instructions. A data storage medium **108** containing digital information is configured to operate with mass storage device **107** to allow processor **102** access to the digital information on data storage medium **108** via bus **101**.

Processor **102** may be any of a wide variety of general purpose processors or microprocessors such as the Pentium™ microprocessor manufactured by Intel™ Corporation or the RS/6000™ processor manufactured by IBM Corporation. It will be apparent to those of ordinary skill in the art, however, that other varieties of processors may also be used in a particular computer system. Display device **105** may be a liquid crystal device, cathode ray tube (CRT), or other suitable display device. Mass storage device **107** may be a conventional hard disk drive, floppy disk drive, CD-ROM drive, or other magnetic or optical data storage device for reading and writing information stored on a hard disk, a floppy disk, a CD-ROM a magnetic tape, or other magnetic or optical data storage medium. Data storage medium **108** may be a hard disk, a floppy disk, a CD-ROM, a magnetic tape, or other magnetic or optical data storage medium.

In general, processor **102** retrieves processing instructions and data from a data storage medium **108** using mass storage device **107** and downloads this information into random access memory **103** for execution. Processor **102**, then executes an instruction stream from random access memory **103** or read-only memory **104**. Command selections and information input at input device **106** are used to direct the flow of instructions executed by processor **102**. Equivalent input device **106** may also be a pointing device such as a conventional mouse or trackball device. The results of this processing execution are then displayed on display device **105**.

The preferred embodiment of the present invention is implemented as a software module, which may be executed on a computer system such as computer system **100** in a conventional manner. Using well known techniques, the application software of the preferred embodiment is stored on data storage medium **108** and subsequently loaded into and executed within computer system **100**. Once initiated, the software of the preferred embodiment operates in the manner described below.

FIG. **2** illustrates a typical prior art Web server environment. Web client **200** can make URL requests to Web server **201** or Web server **202**. Web servers **201** and **202** include Web server executables, **201**(E) and **202**(E) respectively, that perform the processing of Web client requests. Each Web server may have a number of Web pages **201(1)**–(n) and **202(1)**–(n). Depending on the URL specified by the Web client **200**, the request may be routed by either Web server executable **201**(E) to Web page **201 (1)**, for example, or from Web server executable **202**(E) to Web page **202 (1)**. Web client **200** can continue making URL requests to retrieve other Web pages. Web client **200** can also use hyperlinks within each Web page to "jump" to other Web pages or to other locations within the same Web page.

FIG. **3** illustrates this prior art Web server environment in the form of a flow diagram. In processing block **300**, the Web client makes a URL request. This URL request is examined by the Web browser to determine the appropriate Web server to route the request to in processing block **302**. In processing block **304** the request is then transmitted from the Web browser to the appropriate Web server, and in processing block **306** the Web server executable examines the URL to determine whether it is a HTML document or a CGI application. If the request is for an HTML document **308**, then the Web server executable locates the document in processing block **310**. The document is then transmitted back through the requesting Web browser for formatting and display in processing block **312**.

If the URL request is for a CGI application **314**, however, the Web server executable locates the CGI application in processing block **316**. The CGI application then executes and outputs HTML output in processing block **318** and finally, the HTML output is transmitted back to requesting Web browser for formatting and display in processing block **320**.

This prior art Web server environment does not, however, provide any mechanism for managing the Web requests or the Web sites. As Web sites grow, and as the number of Web clients and requests increase, Web site management becomes a crucial need.

For example, a large Web site may receive thousands of requests or "hits" in a single day. Current Web servers process each of these requests on a single machine, namely the Web server machine. Although these machines may be running "multi-threaded" operating systems that allow transactions to be processed by independent "threads," all the threads are nevertheless on a single machine, sharing a processor. As such, the Web executable thread may hand off a request to a processing thread, but both threads will still have to be handled by the processor on the Web server machine. When numerous requests are being simultaneously processed by multiple threads on a single machine, the Web server can slow down significantly and become highly inefficient. The claimed invention addresses this need by utilizing a partitioned architecture to facilitate the creation and management of custom Web sites and servers.

FIG. **4** illustrates one embodiment of the presently claimed invention. Web client **200** issues a URL request that is processed to determined proper routing. In this embodiment, the request is routed to Web server **201**. Instead of Web server executable **201**(E) processing the URL request, however, Interceptor **400** intercepts the request and routes it to Dispatcher **402**. In one embodiment, Interceptor **400** resides on the Web server machine as an extension to Web server **201**. This embodiment is appropriate for Web servers such as Netsite™ from Netscape, that support such extensions. A number of public domain Web servers, such as NCSA™ from the National Center for Supercomputing Applications at the University of Illinois, Urbana-Champaign, however, do not provide support for

this type of extension. Thus, in an alternate embodiment, Interceptor **400** is an independent module, connected via an "intermediate program" to Web server **201**. This intermediate program can be a simple CGI application program that connects Interceptor **400** to Web server **201**. Alternate intermediate programs the perform the same functionality can also be implemented.

In one embodiment of the invention, Dispatcher **402** resides on a different machine than Web server **201**. This embodiment overcomes the limitation described above, in prior art Web servers, wherein all processing is performed by the processor on a single machine. By routing the request to Dispatcher **402** residing on a different machine than the Web server executable **201**(E), the request can then be processed by a different processor than the Web server executable **201**(E). Web server executable **201**(E) is thus free to continue servicing client requests on Web server **201** while the request is processed "off-line," at the machine on which Dispatcher **402** resides.

Dispatcher **402** can, however, also reside on the same machine as the Web server. The Web site administrator has the option of configuring Dispatcher **402** on the same machine as Web server **201**, taking into account a variety of factors pertinent to a particular Web site, such as the size of the Web site, the number of Web pages and the number of hits at the Web site. Although this embodiment will not enjoy the advantage described above, namely off-loading the processing of Web requests from the Web server machine, the embodiment does allow flexibility for a small Web site to grow. For example, a small Web site administrator can use a single machine for both Dispatcher **402** and Web server **201** initially, then off-load Dispatcher **402** onto a separate machine as the Web site grows. The Web site can thus take advantage of other features of the present invention regardless of whether the site has separate machines configured as Web servers and dispatchers.

Dispatcher **402** receives the intercepted request and then dispatches the request to one of a number of Page servers **404 (1)**–(n). For example, if Page server **404 (1)** receives the dispatched request, it processes the request and retrieves the data from an appropriate data source, such as data source **406**, data source **408**, or data source **410**. Data sources, as used in the present application, include databases, spreadsheets, files and any other type of data repository. Page server **404 (1)** can retrieve data from more than one data source and incorporate the data from these multiple data sources in a single Web page.

In one embodiment, each Page server **404(1)**–(n) resides on a separate machine on the network to distribute the processing of the request. Dispatcher **402** maintains a variety of information regarding each Page server on the network, and dispatches requests based on this information. For example, Dispatcher **402** retains dynamic information regarding the data sources that any given Page server can access. Dispatcher **402** thus examines a particular request and determines which Page servers can service the URL request. Dispatcher **402** then hands off the request to the appropriate Page server.

For example, if the URL request requires financial data from data source **408**, dispatcher **402** will first examine an information list. Dispatcher **402** may determine that Page server **404(3)**, for example, has access to the requisite data in data source **408**. Dispatcher **402** will thus route the URL request to Page server **404(3)**. This "connection caching" functionality is described in more detail below, under the heading "Performance." Alternately, Dispatcher **402** also has the ability to determine whether a particular Page server already has the necessary data cached in the Page server's page cache (described in more detail below, under the heading "Performance"). Dispatcher **402** may thus determine that Page server **404(1)** and **404(2)** are both logged into Data source **408**, but that Page server **404(2)** has the financial information already cached in Page server **404(2)**'s page cache. In this case, Dispatcher **402** will route the URL request to Page server **404(2)** to more efficiently process the request.

Finally, Dispatcher **402** may determine that a number or all Page servers **404(1)**–(n) are logged into Data source **408**. In this scenario, Dispatcher **402** can examine the number of requests that each Page server is servicing and route the request to the least busy page server. This "load balancing" capability can significantly increase performance at a busy Web site and is discussed in more detail below, under the heading "Scalability".

If, for example, Page server **404(2)**, receives the request, Page server **404(2)** will process the request. While Page server **404(2)** is processing the request, Web server executable **201**(E) can concurrently process other Web client requests. This partitioned architecture thus allows both Page server **404(2)** and Web server executable **201**(E) to. simultaneously process different requests, thus increasing the efficiency of the Web site. Page server **404(2)** dynamically generates a Web page in response to the Web client request, and the dynamic Web page is then either transmitted back to requesting Web client **200** or stored on a machine that is accessible to Web server **201**, for later retrieval.

One embodiment of the claimed invention also provides a Web page designer with HTML extensions, or "dyna" tags. These dyna tags provide customized HTML functionality to a Web page designer, to allow the designer to build customized HTML templates that specify the source and placement of retrieved data. For example, in one embodiment, a "dynatext" HTML extension tag specifies a data source and a column name to allow the HTML template to identify the data source to log into and the column name from which to retrieve data. Alternatively, "dyna-anchor" tags allow the designer to build hyperlink queries while "dynablock" tags provide the designer with the ability to iterate through blocks of data. Page servers use these HTML templates to create dynamic Web pages. Then, as described above, these dynamic Web pages are either transmitted back to requesting Web client **200** or stored on a machine that is accessible to Web server **201**, for later retrieval.

The presently claimed invention provides numerous advantages over prior art Web servers, including advantages in the areas of performance, security, extensibility and scalability

Performance

One embodiment of the claimed invention utilizes connection caching and page caching to improve performance. Each Page server can be configured to maintain a cache of connections to numerous data sources. For example, as illustrated in FIG. **4**, Page server **404(1)** can retrieve data from data source **406**, data source **408** or data source **410**. Page server **404(1)** can maintain connection cache **412(1)**, containing connections to each of data source **406**, data source **408** and data source **410**, thus eliminating connect times from the Page servers to those data sources.

Additionally, another embodiment of the present invention supports the caching of finished Web pages, to optimize the performance of the data source being utilized. This "page

caching" feature, illustrated in FIG. **4** as Page cache **414**, allows the Web site administrator to optimize the performance of data sources by caching Web pages that are repeatedly accessed. Once the Web page is cached, subsequent requests or "hits" will utilize the cached Web page rather than re-accessing the data source. This can radically improve the performance of the data source.

Security

The present invention allows the Web site administrator to utilize multiple levels of security to manage the Web site. In one embodiment, the Page server can utilize all standard encryption and site security features provided by the Web server. In another embodiment, the Page server can be configured to bypass connection caches **412(1)**–(n), described above, for a particular data source and to require entry of a user-supplied identification and password for the particular data source the user is trying to access.

Additionally, another embodiment of the presently claimed invention requires no real-time access of data sources. The Web page caching ability, described above, enables additional security for those sites that want to publish non-interactive content from internal information systems, but do not want real-time Internet accessibility to those internal information systems. In this instance, the Page server can act as a "replication and staging agent" and create Web pages in batches, rather than in real-time. These "replicated" Web pages are then "staged" for access at a later time, and access to the Web pages in this scenario is possible even if the Page server and dispatcher are not present later.

In yet another embodiment, the Page server can make a single pass through a Web library, and compile a Web site that exists in the traditional form of separately available files. A Web library is a collection of related Web books and Web pages. More specifically, the Web library is a hierarchical organization of Web document templates, together with all the associated data source information. Information about an entire Web site is thus contained in a single physical file, thus simplifying the problem of deploying Web sites across multiple Page servers. The process of deploying the Web site in this embodiment is essentially a simple copy of a single file.

Extensibility

One embodiment of the present invention provides the Web site administrator with Object Linking and Embedding (OLE) 2.0 extensions to extend the page creation process. These OLE 2.0 extensions also allow information submitted over the Web to be processed with user-supplied functionality. Utilizing development tools such as Visual Basic, Visual C++ or PowerBuilder that support the creation of OLE 2.0 automation, the Web site administrator can add features and modify the behavior of the Page servers described above. This extensibility allows one embodiment of the claimed invention to be incorporated with existing technology to develop an infinite number of custom web servers.

For example, OLE 2.0 extensions allow a Web site administrator to encapsulate existing business rules in an OLE 2.0 automation interface, to be accessed over the Web. One example of a business rule is the steps involved in the payoff on an installment or mortgage loan. The payoff may involve, for example, taking into account the current balance, the date and the interest accrued since the last payment. Most organizations already have this type of business rule implemented using various applications, such as Visual Basic for client-server environments, or CICS programs on mainframes. If these applications are OLE 2.0 compliant, the Page server "dynaobject" HTML extension tag can be used to encapsulated the application in an OLE 2.0 automation interface. The Page server is thus extensible, and can incorporate the existing application with the new Page server functionality.

Scalability

One embodiment of the claimed invention allows "plug and play" scalability. As described above, referring to FIG. **4**, Dispatcher **402** maintains information about all the Page servers configured to be serviced by Dispatcher **402**. Any number of Page servers can thus be "plugged" into the configuration illustrated in FIG. **4**, and the Page servers will be instantly activated as the information is dynamically updated in Dispatcher **402**. The Web site administrator can thus manage the overhead of each Page server and modify each Page server's load, as necessary, to improve performance. In this manner, each Page server will cooperate with other Page servers within a multi-server environment. Dispatcher **402** can examine the load on each Page server and route new requests according to each Page server's available resources. This "load-balancing" across multiple Page servers can significantly increase a Web site's performance.

FIG. **5** illustrates the processing of a Web browser request in the form of a flow diagram, according to one embodiment of the presently claimed invention. A Web browser sends a URL request to a Web server in processing block **500**. In processing block **502**, the Web server receives the URL request, and an interceptor then intercepts the handling of the request in processing block **504**. The interceptor connects to a dispatcher and sends the URL request to the dispatcher in processing block **506**. In processing block **508**, the dispatcher determines which Page servers can handle the request. The dispatcher also determines which Page server is processing the fewest requests in processing block **510**, and in processing block **512**, the dispatcher sends the URL request to an appropriate Page server. The Page server receives the request and produces an HTML document in processing block **514**. The Page server then responds to the dispatcher with notification of the name of the cached HTML document in processing block **516**. In processing block **518**, the dispatcher responds to the interceptor with the document name, and the interceptor then replaces the requested URL with the newly generated HTML document in processing block **520**. The Web server then sends the new HTML document to the requesting client in processing block **522**. Finally, the Web browser receives and displays the HTML document created by the Page server at processing block **524**.

Thus, a method and apparatus for creating and managing custom Web sites is disclosed. These specific arrangements and methods described herein are merely illustrative of the principles of the present invention. Numerous modifications in form and detail may be made by those of ordinary skill in the art without departing from the scope of the present invention. Although this invention has been shown in relation to a particular preferred embodiment, it should not be considered so limited. Rather, the present invention is limited only by the scope of the appended claims.

We claim:

1. A computer-implemented method for managing a dynamic Web page generation request to a Web server, said computer-implemented method comprising the steps of:

routing a request from a Web server to a page server, said page server receiving said request and releasing said

Web server to process other requests wherein said routing step further includes the steps of:

intercepting said request at said Web server and routing said request to said page server;

processing said request, said processing being performed by said page server while said Web server concurrently processes said other requests; and

dynamically generating a Web page in response to said request, said Web page including data dynamically retrieved from one or more data sources.

2. The computer-implemented method in claim **1** wherein said step of routing said request includes the steps of:

routing said request from said Web server to a dispatcher; and

dispatching said request to said page server.

3. The computer-implemented method in claim **1** wherein said step of processing said request includes the step of identifying said one or more data sources from which to retrieve said data.

4. The computer-implemented method in claim **1** wherein said step of dynamically generating said Web page includes the step of dynamically retrieving said data from said one or more data sources.

5. The computer-implemented method in claim **1** wherein said step of processing said request includes the step of said page server maintaining a connection cache to said one or more data sources.

6. The computer-implemented method in claim **1** wherein said step of processing said request includes the step of logging into said one or more data sources.

7. The computer-implemented method in claim **1** wherein said step of dynamically generating said Web page includes the step of maintaining a page cache containing said Web page.

8. The computer-implemented method in claim **1** wherein said page server includes tag-based text templates for configuring said Web page.

9. The computer-implemented method in claim **8** wherein said step of processing said request further includes the step of inserting said-dynamically retrieved data from said one or more data sources into said tag-based text templates.

10. The computer-implemented method in claim **8** wherein at least one of said tag-based text templates drives a format of the data dynamically retrieved from said one or more data sources in response to said request.

11. The computer-implemented method in claim **8** wherein said tag-based text templates include HTML templates.

12. The computer-implemented method in claim **1** wherein said step of processing said request further includes the step of dynamically updating data at said one or more data sources.

13. The computer-implemented method in claim **1** wherein said step of processing said request further includes the step of processing an object handling extension.

14. The computer-implemented method in claim **13** wherein said object handling extension is an OLE extension.

15. A computer-implemented method comprising the steps of:

transferring a request from an HTTP-compliant device to a page server, said page server receiving said request and releasing said HTTP-compliant device to process other requests wherein said transferring step further includes the steps of:

intercepting said request at said HTTP-compliant device and transferring said request to said page server;

processing said request, said processing being performed by said page server while said HTTP-compliant device concurrently processes said other requests; and

dynamically generating a page in response to said request, said page including data dynamically retrieved from one or more data sources.

16. The computer-implemented method in claim **15** wherein said step of transferring said request includes the steps of:

transferring said request from said HTTP-compliant device to a dispatcher; and

dispatching said request to said page server.

17. The computer-implemented method in claim **15** wherein said step of processing said request includes the step of identifying said one or more data sources from which to retrieve said data.

18. The computer-implemented method in claim **15** wherein said step of dynamically generating said page includes the step of dynamically retrieving said data from said one or more data sources.

19. The computer-implemented method in claim **15** wherein said step of processing said request includes the step of said page server maintaining a connection cache to said one or more data sources.

20. The computer-implemented method in claim **15** wherein said step of processing said request includes the step of logging into said one or more data sources.

21. The computer-implemented method in claim **15** wherein said step of dynamically generating said page includes the step of maintaining a page cache containing said page.

22. The computer-implemented method in claim **15** wherein said page server includes tag-based text templates for configuring said page.

23. The computer-implemented method in claim **22** wherein said step of processing said request further includes the step of inserting said dynamically retrieved data from said one or more data sources into said tag-based text templates.

24. The computer-implemented method in claim **22** wherein at least one of said tag-based text templates drives a format of the data dynamically retrieved from said one or more data sources in response to said request.

25. The computer-implemented method in claim **22** wherein said tag-based text templates include HTML templates.

26. The computer-implemented method in claim **15** wherein said step of processing said request further includes the step of dynamically updating data at said one or more data sources.

27. The computer-implemented method in claim **15** wherein said step of processing said request further includes the step of processing an object handling extension.

28. The computer-implemented method in claim **27** wherein said object handling extension is an OLE extension.

29. A computer-implemented method comprising the steps of:

transferring a request from an HTTP-compliant device to a dispatcher;

maintaining dynamic information regarding data sources a given page server may access;

dispatching said request to an appropriate page server based on said request and based on said dynamic information, said page server receiving said request and

releasing said HTTP-compliant device to process other requests;

processing said request, said processing being performed by said page server while said HTTP-compliant device concurrently processes said other requests; and

dynamically generating a page in response to said request, said page including data dynamically retrieved from one or more data sources.

* * * * *

JS 44 (Rev. 3/99)

CIVIL COVER SHEET

The JS-44 civil cover sheet and the information contained herein neither replace nor supplement the filing and service of pleadings or other papers as required by law, except as provided by local rules of court. This form, approved by the Judicial Conference of the United States in September 1974, is required for the use of the Clerk of Court for the purpose of initiating the civil docket sheet. (SEE INSTRUCTIONS ON THE REVERSE OF THE FORM.)

I. (a) PLAINTIFFS

Oracle Corporation and Oracle U.S.A. Inc.

(b) COUNTY OF RESIDENCE OF FIRST LISTED PLAINTIFF _____
(EXCEPT IN U.S. PLAINTIFF CASES)

(c) ATTORNEYS (FIRM ADDRESS AND TELEPHONE NUMBER)
Mary B. Graham
Morris, Nichols, Arsht & Tunnell LLP
1201 N. Market Street; PO Box 1347
Wilmington, DE 19899-1347
302-658-9200

DEFENDANTS

epicRealm Licensing, LP

COUNTY OF RESIDENCE OF FIRST LISTED DEFENDANT _____
(IN U.S. PLAINTIFF CASES ONLY)
NOTE: IN LAND CONDEMNATION CASES, USE THE LOCATION OF THE TRACT OF LAND INVOLVED.

ATTORNEYS (IF KNOWN)

II. BASIS OF JURISDICTION (PLACE AN "X" IN ONE BOX ONLY)

- ☐ 1 U.S. Government Plaintiff
- ☒ 3 Federal Question (U.S. Government Not a Party)
- ☐ 2 U.S. Government Defendant
- ☐ 4 Diversity (Indicate Citizenship of Parties in Item III)

III. CITIZENSHIP OF PRINCIPAL PARTIES (PLACE AN "X" IN ONE BOX FOR PLAINTIFF AND ONE BOX FOR DEFENDANT)
(For Diversity Cases Only)

	PTF	DEF		PTF	DEF
Citizen of This State	☐ 1	☐ 1	Incorporated or Principal Place of Business in This State	☐ 4	☐ 4
Citizen of Another State	☐ 2	☐ 2	Incorporated and Principal Place of Business in Another State	☐ 5	☐ 5
Citizen or Subject of a Foreign Country	☐ 3	☐ 3	Foreign Nation	☐ 6	☐ 6

IV. NATURE OF SUIT (PLACE AN "X" IN ONE BOX ONLY)

CONTRACT
- ☐ 110 Insurance
- ☐ 120 Marine
- ☐ 130 Miller Act
- ☐ 140 Negotiable Instrument
- ☐ 150 Recovery of Overpayment & Enforcement of Judgement
- ☐ 151 Medicare Act
- ☐ 152 Recovery of Defaulted Student Loans (Excl Veterans)
- ☐ 153 Recovery of Overpayment of Veteran's Benefits
- ☐ 160 Stockholder Suits
- ☐ 190 Other Contract
- ☐ 195 Contract Product Liability

TORTS

PERSONAL INJURY
- ☐ 310 Airplane
- ☐ 315 Airplane Product Liability
- ☐ 320 Assault Libel & Slander
- ☐ 330 Federal Employers Liability
- ☐ 340 Marine
- ☐ 345 Marine Product Liability
- ☐ 350 Motor Vehicle
- ☐ 355 Motor Vehicle Product Liability
- ☐ 360 Other Personal Injury

PERSONAL INJURY
- ☐ 362 Personal Injury - Med Malpractice
- ☐ 365 Personal Injury - Product Liability
- ☐ 368 Asbestos Personal Injury Product Liability

PERSONAL PROPERTY
- ☐ 370 Other Fraud
- ☐ 371 Truth in Lending
- ☐ 380 Other Personal Property Damage
- ☐ 385 Property Damage Product Liability

REAL PROPERTY
- ☐ 210 Land Condemnation
- ☐ 220 Foreclosure
- ☐ 230 Rent Lease & Ejectment
- ☐ 240 Torts to Land
- ☐ 245 Tort Product Liability
- ☐ 290 All Other Real Property

CIVIL RIGHTS
- ☐ 441 Voting
- ☐ 442 Employment
- ☐ 443 Housing/ Accommodations
- ☐ 444 Welfare
- ☐ 440 Other Civil Rights

PRISONER PETITIONS
- ☐ 510 Motions to Vacate Sentence

HABEAS CORPUS:
- ☐ 530 General
- ☐ 535 Death Penalty
- ☐ 540 Mandamus & Other
- ☐ 550 Civil Rights
- ☐ 555 Prison Condition

FORFEITURE/PENALTY
- ☐ 610 Agriculture
- ☐ 620 Other Food & Drug
- ☐ 625 Drug Related Seizure of Property 21 USC 881
- ☐ 630 Liquor Laws
- ☐ 640 R R & Truck
- ☐ 650 Airline Regs
- ☐ 660 Occupational Safety/Health
- ☐ 690 Other

LABOR
- ☐ 710 Fair Labor Standards Act
- ☐ 720 Labor/Mgmt. Relations
- ☐ 730 Labor Mgmt. Reporting & Disclosure Act
- ☐ 740 Railway Labor Act
- ☐ 790 Other Labor Litigation
- ☐ 791 Empl Ret Inc Security Act

BANKRUPTCY
- ☐ 422 Appeal 28 USC 158
- ☐ 423 Withdrawal 28 USC 157

PROPERTY RIGHTS
- ☐ 820 Copyrights
- ☒ 830 Patent
- ☐ 840 Trademark

SOCIAL SECURITY
- ☐ 861 HIA (1395ff)
- ☐ 862 Black Lung (923)
- ☐ 863 DIWC/DIWW (405(g))
- ☐ 864 SSID Title XVI
- ☐ 865 RSI (405(g))

FEDERAL TAX SUITS
- ☐ 870 Taxes (U.S. Plaintiff or Defendant
- ☐ 871 IRS - Third Party 26 USC 7609

OTHER STATUTES
- ☐ 400 State Reapportionment
- ☐ 410 Antitrust
- ☐ 430 Banks or Banking
- ☐ 450 Commerce/ICC Rates/etc
- ☐ 460 Deportation
- ☐ 470 Racketeer Influenced and Corrupt Organizations
- ☐ 810 Selective Service
- ☐ 850 Securities/Commodities/ Exchange
- ☐ 875 Customer Challenge 12 USC 3410
- ☐ 891 Agricultural Acts
- ☐ 892 Economic Stabilization Act
- ☐ 893 Environmental Matters
- ☐ 894 Energy Allocation Act
- ☐ 895 Freedom of Information Act
- ☐ 900 Appeal of Fee Determination Under Equal Access to Justice
- ☐ 950 Constitutionality of State Statutes
- ☐ 890 Other Statutory Actions

V. ORIGIN (PLACE AN "X" IN ONE BOX ONLY)

☒ 1 Original Proceeding | ☐ 2 Removed From State Court | ☐ 3 Remanded From Appellate Court | ☐ 4 Reinstated or Reopened | ☐ 5 Transferred From another district (specify) ____ | ☐ 6 Multidistrict Litigation | ☐ 7 Appeal to District Judge from Magistrate Judgement

VI. CAUSE OF ACTION

(CITE THE U.S. CIVIL STATUTE UNDER WHICH YOU ARE FILING AND WRITE BRIEF STATEMENT OF CAUSE DO NOT CITE JURISDICTIONAL STATUTES UNLESS DIVERSITY)

Action for declaratory judgment of patent invalidity and noninfringement - 35 U.S.C. §§ 101 et seq.

VII. REQUESTED IN COMPLAINT

☐ CHECK IF THIS IS A CLASS ACTION UNDER F.R.C.P. 23

DEMAND $

CHECK YES only if demanded in complaint:
JURY DEMAND: ☐ YES ☒ NO

VIII. RELATED CASE(S) IF ANY (See Instructions)

JUDGE _____ DOCKET NUMBER _____

DATE 6/30/06

SIGNATURE OF ATTORNEY OF RECORD Mary B. Graham

FOR OFFICE USE ONLY

RECEIPT # _____ AMOUNT _____ APPLYING IFP _____ JUDGE _____ MAG. JUDGE _____

AO FORM 85 RECEIPT (REV. 9/04)

United States District Court for the District of Delaware

Civil Action No. 06-414

ACKNOWLEDGMENT OF RECEIPT FOR AO FORM 85

NOTICE OF AVAILABILITY OF A UNITED STATES MAGISTRATE JUDGE TO EXERCISE JURISDICTION

I HEREBY ACKNOWLEDGE RECEIPT OF 3 COPIES OF AO FORM 85.

6-30-06
(Date forms issued)

(Signature of Party or their Representative)

Luke Nozzi
(Printed name of Party or their Representative)

Note: Completed receipt will be filed in the Civil Action