IN THE UNITED STATES DISTRICT COURT

FOR THE DISTRICT OF DELAWARE

ORACLE CORPORATION and ORACLE U.S.A., INC.,	)	
Plaintiffs,	)	
v.	)	Civ. No. 06-414-SLR
PARALLEL NETWORKS, LLP,	)	
Defendant.	)	

Mary B. Graham, Esquire and James W. Parrett, Jr., Esquire of Morris, Nichols, Arsht & Tunnell LLP, Wilmington, Delaware. Counsel for Plaintiffs. Of Counsel: James G. Gilliland, Esquire, Theodore T. Herhold, Esquire, Joseph A. Greco, Esquire, Robert J. Artuz, Esquire, and Eric M. Hutchins, Esquire of Townsend and Townsend and Crew LLP, Palo Alto, California; Dorian Daley, Esquire, Peggy E. Bruggman, Esquire, and Matthew M. Sarboraria, Esquire of Oracle Corporation & Oracle U.S.A., Inc., Redwood Shores, California.

Richard L. Horwitz, Esquire and David E. Moore, Esquire of Potter Anderson & Corroon LLP, Wilmington, Delaware. Counsel for Defendant. Of Counsel: Harry J. Roper, Esquire, George S. Bosy, Esquire, Aaron A. Barlow, Esquire, Patrick L. Patras, Esquire, David R. Bennett, Esquire, Benjamin E. Bradford, Esquire, Paul D. Margolis, Esquire and Emily C. Johnson, Esquire of Jenner & Block, Chicago, Illinois.

MEMORANDUM OPINION

Dated: December 4, 2008
Wilmington, Delaware

[illegible]
ROBINSON, District Judge

I. INTRODUCTION

On June 30, 2006, Oracle Corporation and Oracle U.S.A. Inc. (collectively, "Oracle" or "plaintiffs") filed this action for declaratory judgment against EpicRealm Licensing, L.P.[1] (D.I. 1) Defendant, a patent licensing firm, is owner and assignee of United States Patent Numbers 5,894,554 ("the '554 patent") and 6,415,335 ("the '335 patent"), which are directed to a system for creating and managing custom web sites. (*Id.*; D.I. 10 at 16) Plaintiffs seek a judgment that they do not infringe the '554 or '335 patent, and that both patents are invalid and/or unenforceable. (*Id.*; D.I. 339) Currently pending before the court are plaintiffs' motions for summary judgment of noninfringement (D.I. 204), invalidity (D.I. 206), no willful infringement (D.I. 208) and to exclude defendant from asserting damages based on plaintiffs' foreign sales (D.I. 212). Also before the court are defendant's motions for partial summary judgment of literal infringement (D.I. 223) and that plaintiffs' prior art references do not anticipate (D.I. 216).

II. BACKGROUND

A. The Parties and Litigation History

Plaintiffs manufacture, sell and license software products for customers to use in

[1]EpicRealm Licensing, L.P. ("EpicRealm") is a patent licensing firm headquartered in Richardson, Texas; it does not produce or sell any products. EpicRealm assigned all right, title, and interest in the patents in suit to Parallel Networks LLP in August 2007. The court granted EpicRealm's motion to substitute parties on September 29, 2008. (D.I. 355) For simplicity's sake, the court will refer to one "defendant" throughout its opinion.

conjunction with the delivery of dynamic web pages.[2]

Defendant previously brought several actions for infringement of the ‘554 and ‘335 patents in the United States District Court for the Eastern District of Texas.[3] That litigation was consolidated in November 2005 (hereinafter, “the Texas litigation”). Plaintiffs were not named in the Texas litigation. An Oracle customer, Safelite Group, Inc. (“Safelite”), was named as a defendant. Safelite asserted counterclaims that the ‘554 and ‘335 patents are invalid, and filed a third party complaint against plaintiffs for indemnification. Defendant and Safelite settled the Texas litigation and filed a stipulation of dismissal with the court on June 26, 2006. (D.I. 282, ex. 22) A stipulation of dismissal was also filed with respect to Safelite’s third party complaint against plaintiffs. On June 29, 2006, the court entered orders dismissing both complaints. (D.I. 1 at ¶¶ 26-27)

Plaintiffs brought their declaratory judgment suit in this court on June 30, 2006. (*Id.*) In the complaint, plaintiffs allege that, in a letter to Clark Consulting, Inc. (a party to the Texas litigation), defendant stated that Clark was required to provide discovery regarding Clark’s use of software proprietary to plaintiffs. (*Id.* at ¶ 24) Plaintiffs also claim that defendant demanded and received discovery from Safelite regarding its use of Oracle software. (*Id.* at ¶ 25) Defendant moved to transfer venue and consolidate

[2]Oracle Corporation is a software manufacturer incorporated and organized under the laws of Delaware with a principal place of business in Redwood Shores, California. (D.I. 1) Oracle USA, Inc., a Colorado corporation, is a wholly owned subsidiary of Oracle Corporation. (*Id.*)

[3]*epicRealm Licensing, LP v. Franklin Covey Co. et al*, 2:05-CV-356; *epicRealm, Licensing, LP v. Speedera Networks, Inc.*, 2:05-CV-150; *epicRealm Licensing, LP v. Autoflex Leasing, Inc. et al*, 2:05-CV-163.

with the Texas litigation. This court denied defendant's motions on March 26, 2007. (D.I. 21)

Defendant thereafter answered the complaint on May 3, 2007, in which it admitted an actual controversy exists between the parties for jurisdictional purposes, admitted that it sought discovery from Clark, but denied that it requested discovery specifically relating to Safelite's use of Oracle software. (D.I. 25 at ¶ 25) Defendant also brought a counterclaim of patent infringement. (*Id.*) Plaintiffs amended their complaint on October 15, 2007, to add a claim that the '554 and '335 patents are unenforceable due to inequitable conduct. (D.I. 369) Discovery is now closed, and trial is currently scheduled to commence January 12, 2009. (D.I. 29)

B. Technological Background and the Patents-at-Issue

The basic three-tiered architecture of the internet includes what is known as a desktop tier, an intermediate tier, and an enterprise tier. The desktop tier is composed of a client program (web browser, such as Microsoft Internet Explorer®) located on a user's desktop computer, which sends and receives requests for information over the internet. The intermediate tier comprises one or more web servers, which receive and process user requests and return completed web pages to the client for viewing. The enterprise tier is synonymous with data services; it comprises one or more back-end database servers which store the information used to make web pages.

Formerly, most web sites provided only "static" web pages, or pages whose content was not subject to change. When a web client (a computer with a web browser) identified a web site, the browser program connected to the web, and the web server operating the web site received the request and retrieved the specific file requested by the web client – no file modification occurred. Over time, web sites began to provide dynamic web pages, *i.e.*, web pages that are generated anew in response to a specific request of the web client. To generate dynamic web pages, the Common Gateway Interface ("CGI") was developed. CGI is a protocol for identifying a command, running it, and returning output from a web server. Once created, a CGI application does not have to be modified to retrieve new data and generate a dynamic page; it does so automatically.

The processing of dynamic web pages requires more processor time, memory, and/or other system resources than is the case with static web pages. As the number of users' dynamic web page requests increased, so too did the demand on web server resources, resulting in slowed response time, failure to provide the requested content, or the crashing of the web server. The tools that generate CGI applications do not solve these problems.

The patents-in-suit disclose systems for efficiently managing dynamic web page generation requests. The architecture of the patented system is depicted in figure 4 of the patents.[4] First, a web client initiates a request for a static or dynamic web page. ('554 patent, col. 4, ll. 55-57) The request is routed to a web server. (*Id.* at l. 57) Instead of the web server processing the request, an "interceptor" intercepts the request and routes it to a "dispatcher." (*Id.* at ll. 58-60) The dispatcher identifies one or more "page servers," or a server connected to the data source. (*Id.*, col. 5, ll. 37-39)

The dispatcher maintains a variety of information on each page server to select

[4]The '335 patent was issued from a continuation application claiming priority to the '554 patent; therefore, both patents share the same specification and filing date.

the appropriate page server. (*Id.*, col. 5, ll. 54-59) The patents provide several scenarios in which the dispatcher selects a page server. The first is "connection caching," whereby a dispatcher determines that a particular page server "has access to the requisite data" in the data source. (*Id.*, col. 5, ll. 60-67) Alternatively, the dispatcher may determine that a particular page server "already has the necessary data cached in the page server's page cache[[5]]," even though another page server may also be logged into the appropriate data source, and select the server containing the cached data. (*Id.*, col. 6, ll. 1-11) Lastly, the dispatcher may determine that multiple page servers are logged into the appropriate data source, in which case the dispatcher will select the "least busy" page server. (*Id.*, col. 6, ll. 12-19) This "load balancing" can "significantly increase performance at a busy web site." (*Id.*)

The patents provide that, while a page server is processing the request for data retrieval, the web server is free to concurrently process other web client requests, promoting web site efficiency. (*Id.*, col. 6, ll. 21-27) The page server dynamically generates a web page in response to the web client request, and the web page is either transmitted back to the web client or stored on a machine that is accessible to the web server for later retrieval. (*Id.* at col. 6, ll. 27-31)

Defendant asserts that plaintiffs infringe claims 1-5 and 7-11 of the '554 patent and claims 2 and 16 of the '335 patent. The asserted independent claims of the '554 patent read as follows:

1. A computer-implemented method for managing a dynamic Web page

[5]Generally, a block of memory for temporary storage of data likely to be used again. Web caches store previous responses from web servers, such as web pages.

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.

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.

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.

Claims 2 of the ‘335 patent depends from claim 1. Those claims read as follows:

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.

Asserted claim 16 of the '335 patent depends from claim 15, as follows:

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.

C. Accused Products

Defendant asserts that the following Oracle products infringe the patents-in-suit: (1) the Oracle Web Cache Products, beginning in November 2000 with Release 1.0.2 and all subsequent releases ("the Web Cache products"); (2) the Oracle Application Server Products beginning in April 2003 with Release 10gR1 (9.0.4) and all subsequent releases ("the Application Server products"); (3) the Oracle Database Products with Real Application Clusters ("RAC") beginning in May 2005 with Release 10gR2 (10.2.0.1.0) for JDBC and all subsequent releases and beginning in October 2007 with Release 11g (11.1) for OCI and all subsequent releases ("the Database products").[6] These products will be discussed in more detail *infra* in the context of the parties' infringement/noninfringement arguments.

III. STANDARD OF REVIEW

A court shall grant summary judgment only if "the pleadings, depositions, answers to interrogatories, and admissions on file, together with the affidavits, if any, show that there is no genuine issue as to any material fact and that the moving party is entitled to judgment as a matter of law." Fed. R. Civ. P. 56(c). The moving party bears the burden of proving that no genuine issue of material fact exists. *See Matsushita Elec. Indus. Co. v. Zenith Radio Corp.*, 475 U.S. 574, 586 n.10 (1986). "Facts that could alter the outcome are 'material,' and disputes are 'genuine' if evidence exists from which a rational person could conclude that the position of the person with the burden of proof on the disputed issue is correct." *Horowitz v. Fed. Kemper Life Assurance Co.*, 57 F.3d 300, 302 n.1 (3d Cir. 1995) (internal citations omitted). If the moving party has demonstrated an absence of material fact, the nonmoving party then "must come forward with 'specific facts showing that there is a genuine issue for trial.'" *Matsushita*, 475 U.S. at 587 (quoting Fed. R. Civ. P. 56(e)). The court will "view the underlying facts

[6]Defendant contends that the Web Cache products and Application Server products infringe every asserted claim, while the Database products allegedly infringe all asserted claims except claims 4 and 5 of the '554 patent.

and all reasonable inferences therefrom in the light most favorable to the party opposing the motion." *Pa. Coal Ass'n v. Babbitt*, 63 F.3d 231, 236 (3d Cir. 1995). The mere existence of some evidence in support of the nonmoving party, however, will not be sufficient for denial of a motion for summary judgment; there must be enough evidence to enable a jury reasonably to find for the nonmoving party on that issue. *See Anderson v. Liberty Lobby, Inc.*, 477 U.S. 242, 249 (1986). If the nonmoving party fails to make a sufficient showing on an essential element of its case with respect to which it has the burden of proof, the moving party is entitled to judgment as a matter of law. *See Celotex Corp. v. Catrett*, 477 U.S. 317, 322 (1986).

IV. DISCUSSION

A. Infringement

Defendant moves for partial summary judgment of infringement of the '554 patent, arguing that the accused products infringe because they literally meet every limitation of claim 11 of the '554 patent.[7] (D.I. 224 at 1) Plaintiffs move for summary

[7]Defendant fails to create a genuine issue of material fact as to whether the accused products infringe under the doctrine of equivalents. Defendant's expert Dr. David Finkel's supplemental report in support of a doctrine of equivalents theory is untimely, as it was submitted after summary judgment briefing was complete. Thus, defendant's argument for infringement under the doctrine of equivalents rests entirely on one paragraph in Finkel's second declaration stating that "[t]he Accused Oracle Products infringe because the differences between the Accused Oracle Products and the asserted claims are insubstantial from the perspective of a person of ordinary skill in the relevant art." (D.I. 273 at Exh. 5, ¶ 36) Defendant cites *Optical Disc Corp. v. Del Mar Avionics*, 208 F.3d 1324, 1336 (Fed. Cir. 2000), for the proposition that Finkel's conclusion is sufficient to create a genuine issue of material fact regarding plaintiff's infringement by equivalents. (D.I. 275 at 38-39) However, the expert in *Optical Disc* supported his conclusion regarding infringement by equivalents with a limitation by limitation comparison of the patent and the accused product and a detailed "function-way-result" analysis. *Optical Disc*, 208 F.3d at 1336. Defendant fails to show that Finkel's conclusion is similarly supported. In addition, the court notes that, even if

judgment of noninfringement of both the '554 and the '335 patents, arguing that the accused products do not infringe because they do not literally meet the "intercepting," "releasing," and "dispatcher" limitations common to the asserted claims of both patents. (D.I. 204 at 1-2) The parties do not dispute the physical characteristics of the accused products. (D.I. 204 at 1, ¶ 2; D.I. 224 at 1) Accordingly, the question of whether the accused products literally infringe the patents-in-suit turns on the court's claim construction and may be resolved on summary judgment. *See Gen. Mills, Inc. v. Hunt-Wesson, Inc.*, 103 F.3d 978, 983 (Fed. Cir. 1997).

1. The Accused Products

a. Web Cache products

Web Cache is a cache server, which is a software program designed to maintain a cache, or local store, of frequently used web pages. (D.I. 270 at ¶ 28) Web Cache sits in front of a web server and receives a web client's request for content before the web server does. (*Id.* at ¶ 28) If Web Cache has the requested content in its cache (a "cache hit"), Web Cache returns the requested content to the web client. (*Id.*) Cache hits are handled completely by Web Cache. (*Id.* at ¶ 30) If Web Cache does not have the requested content in its cache (a "cache miss"), Web Cache sends the web client's

---

Finkel's supplemental report had been timely filed, its doctrine of equivalents analysis seems to be nothing more than a restatement of defendant's literal infringement arguments in the "function-way-result" pattern and, thus, is not sufficiently particularized to create a genuine issue of material fact. Accordingly, Finkel's conclusion is insufficient to create a material factual dispute. *See Zelinski v. Brunswick Corp.*, 185 F.3d 1311, 1317 (Fed. Cir. 1999) (affirming district court's grant of summary judgment where only evidence on infringement under doctrine of equivalents was conclusory statement of patentee's expert); *Network Commerce, Inc. v. Microsoft Corp.*, 422 F.3d 1353, 1363 (Fed. Cir. 2005) (evidence supporting infringement by equivalents must be particularized to raise a genuine issue of material fact).

request to a web server for processing. (*Id.* ¶ 28) The web servers sitting behind Web Cache that originate new content in the event of a cache miss are called "origin servers." (*Id.*) Web Cache's purpose is to cache frequently requested content in order to reduce the load on the origin servers. (*Id.* at ¶ 32)

Web Cache performs its caching function by assigning each received web request to a "fiber" within the Web Cache program. (*Id.*) Each fiber is an independent unit of execution. (*Id.*) Web Cache fibers that connect to web clients and search the cache for requested content are called "Front End" fibers. (*Id.*) Web Cache fibers that connect to an origin server and add new content to the cache are called "Back End" fibers. (*Id.*)

When Web Cache receives a request from a web client, Web Cache first creates a new Front End fiber to handle that request. (*Id.* at ¶ 33)[8] The Front End fiber then compares the Universal Resource Locator ("URL")[9] of the request to the URLs of previously-requested content stored in the cache. (*Id.* at ¶ 34) If the comparison yields a cache hit, the Front End fiber returns the requested content to the web client and the processing is complete. (*Id.*) Web Cache then, unless commanded otherwise,

[8]Users may configure Web Cache for a maximum number of Front End fibers. (D.I. 270 at ¶ 33) The default maximum is 700, which means that, with its default configuration, Web Cache can support up to 700 simultaneous web client requests. (*Id.*) If all 700 Front End fibers are occupied – either processing a request or waiting for a requested web page from an origin server – then Web Cache cannot process any additional requests until one of the 700 Front End fibers completes a request. (*Id.*)

[9]A URL is a unique identifier, or address, that defines the location of a file on the web or any other internet facility. (D.I. 270 at ¶ 28)

destroys the Front End fiber.[10] (*Id.*)

If the URL comparison yields a cache miss, then Web Cache creates a Back End fiber, which communicates the URL of the request to an origin server and then waits for the origin server to process the request and return the content. (*Id.* at ¶ 35) The Front End and Back End fibers wait until either the origin server returns the requested content or a time-out error occurs. (*Id.*) Once the origin server locates the requested content, it returns the content to the web client via the Front End and Back End fibers, and the content is inserted into the cache so that Web Cache can satisfy future requests for the same content without involving an origin server. (*Id.* at ¶ 36) Web Cache then destroys the Front End and Back End fibers. (*Id.*)

Web Cache can be configured in several different ways. (*Id.* at ¶ 30) Users can configure a system to use a single Web Cache in front of multiple origin servers, with Web Cache maintaining a cache of content requested from each origin server and distributing cache misses among the multiple origin servers. (*Id.* at ¶ 31)

During normal operation, Web Cache receives TCP acknowledgments[11] from

---

[10]Web Cache does not normally keep the Front End fiber alive to handle a second request. (D.I. 270 at ¶ 37) The only exception is if the web client sends further requests over the same connection by means of a feature of version 1.1 of the HTTP protocol known as the "Keep-Alive" command. (*Id.*) Where the "Keep-Alive" command is invoked, the Front End fiber will process each of the additional requests from that web client one at a time before being destroyed. (*Id.*)

[11]The internet protocol suite (commonly known as "TCP/IP") is the set of communications protocols that implement the protocol stack on which the internet and most commercial networks run. (D.I. 270 at ¶ 59) It is named for its two most important protocols: the Transmission Control Protocol ("TCP") and the Internet Protocol ("IP"). (*Id.*) The internet protocol suite can be viewed as a set of layers, each one solving a set of problems involving data transmission and serving upper layer protocols based on using services from lower layers. (*Id.* at ¶ 60) Upper layers rely on lower layer

Oracle HTTP Server ("OHS") (*id.* at ¶ 58), which is the web server component of the Application Server products to be discussed hereafter (*id.* at ¶ 75).

---

protocols to translate data into forms that can eventually be physically transmitted. (*Id.*) The four TCI/IP layers, listed lowest to highest, are the network access layer, the network layer, the transport layer, and the application layer. (*Id.*)

The application layer consists of all the processes that involve user interaction. (*Id.* at ¶ 61) The applications within the layer determine the presentation and control the session. For example, applications such as web browsers (e.g., Microsoft Internet Explorer, Mozilla Firefox) primarily use Hypertext Transfer Protocol ("HTTP") to communicate with servers running websites. Other applications in the layer use other protocols, including Simple Mail Transfer Protocol ("SMTP") and Post Office Protocol ("POP") for e-mail and File Transfer Protocol ("FTP") for the transfer of files to and from various computers. (*Id.*)

There are two transport layer protocols: TCP and the User Datagram Protocol ("UDP"). (Id.) TCP guarantees that information is received as it was sent, whereas UDP does not. (*Id.*)

The network layer isolates the upper layer protocols from the details of the underlying network and manages the connections across the network. (*Id.*) IP is normally described as the network layer. (*Id.*) All upper and lower layer communications travel through IP as they are passed through the TCP/IP protocol stack. (*Id.*)

The network access layer consists of the data-link layer protocols that define the use of the electrical signals for data transmission, and the physical layer protocols that define the physical connection itself. (*Id.*)

TCP/IP's four-layer structure is built as information is passed down from applications to the network access layer. (Id. at ¶ 62) When data is sent, each layer treats the information received from the layer above as data and adds control information, or a header, to the front of the data. (*Id.*) This process is called encapsulation. (*Id.*) When data is received, the process reverses, as each layer removes its header before passing the data to the layer above. (*Id.*)

This structure allows software developers to write software providing for an application layer protocol, such as HTTP, without providing for implementation of TCP or other lower layer protocols. (*Id.* at ¶ 65) This is possible because implementation of lower layer protocols can be done by other software, such as a computer's operating system. (*Id.*)

During TCP communication between two computers, the operating system of the receiving computer will, upon receiving a TCP segment, transmit a transport layer acknowledgment ("ACK") message. (*Id.* at ¶ 66) If the sending computer does not receive the ACK message within a certain time, it will resend the TCP segment. (*Id.*) Once the sending cornputer receives the ACK message, it discards the received packets from the TCP memory buffer, which frees some of the memory allocated to the TCP buffer by the operating system. (*Id.* at ¶¶ 66, 68)

b. Application Server products

The Application Server products contain multiple software programs, such as OHS and Oracle Containers For Java ("OC4J"). (*Id.* at ¶ 75) OHS is the web server component of the Application Server products. (*Id.*) The OHS software program contains several built-in functions, including the HTTP Listener and a collection of modules. (*Id.*) The HTTP Listener receives incoming web client requests and passes them to the appropriate processing module. (*Id.* at ¶ 76) The modules perform various functions related to the processing of web client requests. (*Id.* at ¶ 77)

In the default configuration, there are 256 instances, or copies, of the OHS program. (*Id.* at ¶ 81) Where Web Cache is not present or where a web client's request is a cache miss, one of the instances of OHS accepts the web client's request for processing. (*Id.*) The OHS instance processes the request by calling each module in the order in which the modules were loaded into memory when the OHS program was first started. (*Id.* at ¶ 82) When it calls a module, the OHS instance compares the URL of the request against a list of types of web page content to determine whether the request is the type of request that the module is designed to process. (*Id.*) Once the OHS instance identifies the correct module to use, the OHS instance processes the request using that module. (*Id.*)

If the OHS instance calls all of the modules and no module is capable of processing the request, the OHS instance will attempt to satisfy the request by accessing the content from the computer's local storage, *e.g.*, files on the computer's hard drive. (*Id.* at ¶ 83) If the OHS instance cannot satisfy the request from accessing the local storage, then the OHS instance will send an error message to the web client indicating that the requested content was not found. (*Id.*) Once the OHS instance either sends the requested web page content or an error message, the OHS instance is done processing the request. (*Id.*)

One of OHS's modules, mod_oc4j, enables OHS to communicate with OC4J. (*Id.* at ¶ 78) OC4J is designed to contain a user's Java-based software applications. (*Id.* at ¶ 80) A user using OC4J would design a Java-based software application and then use OC4J to run that application when it is requested by a web client. (*Id.*) When a web client requests content that requires processing by a Java-based software application, and that request is not handled by Web Cache, an OHS instance uses mod_oc4j to route those requests to an OC4J program. (*Id.* at ¶ 79) The OHS instance then waits until OC4J returns the completed request. (*Id.* at ¶ 84) Once OC4J has returned the requested content to the OHS instance, the OHS instance sends the requested content to the web client. (*Id.*) Because OHS instances can process only one request at a time, requests made while an OHS instance is engaged in responding to a request – either processing the request itself or waiting for some other component to process the request – must be handled by some other OHS instance. (*Id.* at ¶¶ 84-85)

Users of the Application Server products can change some of the configuration details concerning how mod_oc4j communicates with OC4J instances.[12] (*Id.* at ¶ 86) As an initial matter, users must configure OHS with mod_oc4j. (*Id.*) Having done that, where users have also configured the Application Server product to have more than one OC4J instance, mod_oc4j will load balance requests among the multiple OC4J instances. (*Id.*) Users can configure mod_oc4j to perform one of eight different load balancing policies: Random, Round Robin, Random with Local Affinity, Round Robin with Local Affinity, Random using Routing Weight, Round Robin using Routing Weight, Metrics Based, and Metric Based with Local Affinity. (*Id.*)

Round Robin is the default load balancing configuration. (*Id.* at ¶ 87) Round Robin is based on a simple sequential algorithm. (*Id.* at ¶ 89) Mod_oc4j maintains a list of the OC4J instances and sends requests to those OC4J instances according to their listed order. (*Id.*) If the OC4J instance slated to handle the next request is still busy processing a previously-received request, then mod_oc4j will skip over it and send the request to the next available OC4J instance on the list. (*Id.*)

Metrics Based and Metric Based with Local Affinity are the only two load balancing configurations that enable mod_oc4j to route requests based on metrics from OC4J instances. (*Id.* at ¶ 87) To use either of these load balancing configurations, the

---

[12]OC4J instances are started and managed by Oracle Process Manager and Notification Server ("OPMN"). (D.I. 270 at ¶ 86)

user must reconfigure mod_oc4j. (*Id.*)

c. Database products

The Database products consist primarily of a Relational Database Management System ("RDBMS"). (*Id.* at ¶ 126) An RDBMS is a package of software programs that control organization, storage, management, and retrieval of data in a database based on a relational model. (*Id.*) An RDBMS primarily services requests for data stored in the database. (*Id.*) Those requests are typically made in the form of database queries, which are requests for data that are made in a language that the database management system can understand.[13] (*Id.*)

The Database products do not store web pages. (*Id.* at ¶¶ 127-128) Data retrieved from the Database product may be displayed in a web browser as part of a web page, but the web page is typically constructed by a separate application (e.g., OC4J or mod_plsql) or by a web client's browser. (*Id.* at ¶ 127) For example, where the Database product is working in conjunction with the Application Server product, a web client request seeking Java-based content will pass from OHS to OC4J using mod_oc4j. (*Id.* at ¶ 128) OC4J can retrieve data from the Database product via an application programming interface ("API") called JDBC and then construct a web page incorporating the database data. (*Id.*)

Application servers and other database clients can connect to and query the Database products in several ways. (*Id.* at ¶ 129) API is one such way. (*Id.*) API is not a separate executable software program; rather, it is a set of standardized function

[13]An example of such language is the Structured Query Language, or "SQL." (D.I. 270 at ¶ 126.)

calls that permit a software program to issue high-level commands through a library that is linked into the program. (*Id.*) JDBC is an industry standard set of APIs that provide the interface for connecting from Java applications to relational databases, like the Database products, to issue database queries. (*Id.* at ¶ 130) C and C++ applications can also connect to the Database products using APIs called Oracle Call Interface ("OCI") and the Oracle C++ Call Interface ("OCCI"), respectively. (*Id.* at ¶ 131) For example, OCI consists of a set of C-language software APIs that provide a low-level interface to the database. (*Id.*) OCI has APIs for administering the database (including system start-up and shutdown) and for using PL/SQL or SQL to query, access, and manipulate data. (*Id.*) Both JDBC and OCI are libraries that enable OHS to issue commands to the Database products. (*Id.*) Both the Application Server products and the Database products contain software code for these libraries. (*Id.*)

Oracle Real Application Clusters ("RAC") is a database option in which a single database is accessed by multiple instances on multiple computers, or "nodes." (*Id.* at ¶ 132) A typical RAC instance groups multiple database instances running on multiple nodes. (*Id.*) These nodes communicate with each other and share a common pool of disks. These disks house all of the data files that comprise the database. (*Id.*) A RAC database instance can generate a web page. (*Id.* at ¶ 139)

2. Establishing Infringement

To establish direct infringement, "every limitation set forth in a claim must be found in an accused product, exactly." *Southwall Tech., Inc. v. Cardinal IG Co.*, 54 F.3d 1570, 1575 (Fed. Cir. 1995). "If any claim limitation is absent from the accused device, there is no literal infringement as a matter of law." *Bayer AG v. Elan Pharm. Research Corp.*, 212 F.3d 1241, 1247 (Fed. Cir. 2000). Significant to the case at bar, if an accused product does not infringe an independent claim, it also does not infringe any claim depending thereon. *Wahpeton Canvas Co. v. Frontier, Inc.*, 870 F.2d 1546, 1553 (Fed. Cir. 1989).

To establish indirect infringement, defendant has available two theories: active inducement of infringement and contributory infringement. *See* 35 U.S.C. § 271(b) & (c). To establish active inducement of infringement, defendant must show that plaintiffs "knew or should have known [their] actions would induce actual infringements." *DSU Med. Corp. v. JMS Co., Ltd.*, 471 F.3d 1293, 1306 (Fed. Cir. 2006). To establish contributory infringement, defendant must show that plaintiffs sell "a component of a patented machine . . . knowing the same to be especially made or especially adapted for use in an infringement of such patent, and not a staple article or commodity of commerce suitable for substantial noninfringing use." *Golden Blount, Inc. v. Robert H. Peterson Co.*, 365 F.3d 1054, 1061 (Fed. Cir. 2004) (quoting 35 U.S.C. § 271(c)). Liability under either theory, however, depends on defendant having first shown direct infringement. *Joy Technologies, Inc. v. Flakt, Inc.*, 6 F.3d 770, 774 (Fed. Cir. 1993).

3. Direct Infringement

The court's infringement analysis begins and ends with the "releasing" limitation. This is so because the "releasing" limitation is common to the asserted claims of both patents, and its absence from the accused products is dispositive of both plaintiffs' and defendant's infringement motions. In other words, because the court does not find, on the record at bar, a genuine issue of material fact with respect to whether the accused products literally meet the "releasing" limitation, the accused products do not infringe.

The court construes the "releasing" limitation as follows: "said page server receiving said request and **releasing said Web server to process other requests**" means "**freeing the Web server to process other requests**."[14] Defendant presents two basic infringement paradigms through which it argues that the accused products meet the "releasing" limitation. One paradigm treats Web Cache as the "Web server" and OHS (the web server software program within the Application Server products) as the "page server." The other paradigm treats OHS as the "Web server" and either OC4J or a RAC database instance as the "page server." Defendant argues that, in both paradigms, the "page server" frees the "Web server" to process other requests.

Defendant posits two theories for "releasing" that it applies to both paradigms. Defendant's first theory ("the delegation theory") is that a "page server" processing a request frees the "Web server" to process other requests because the "Web server" no longer has the duty to process that request. (D.I. 224 at 19, 26, 34) Put another way, once a "Web server" delegates the processing of a request to a "page server," the "Web server" is then freed to process other requests. (D.I. 324 at 11-12) Defendant's second theory ("the ACK theory") is that, by acknowledging receipt of a request by sending an

[14]This is the "releasing" limitation as it appears in claims 1 and 11 of the '554 patent and claim 1 of the '335 patent. The "releasing" limitation also appears in claim 9 of the '554 and claim 15 of the '335 patent, though instead of referencing the release of "said Web server," those claims reference the release of, respectively, "said second computer system" and "said HTTP-compliant device." For brevity's sake, the court analyzes the "releasing" limitation as it appears in claims 1 and 11 of the '554 patent and claim 1 of the '335 patent, though its analysis applies with equal force to the "releasing" limitation as it appears elsewhere in the patents-in-suit.

ACK message, a "page server" frees the "Web server" from having to further process the request and enables it to instead process other requests. (D.I. 224 at 20, 26-27, 34) Put another way, the "Web server" devotes resources to a request until the "page server" acknowledges via an ACK message that the "page server" will process the request; if the "Web server" receives the ACK message, the "Web server" then knows that it can stop devoting resources to processing that request and can devote resources to processing other requests. (D.I. 324 at 12-13) The evidence to which the parties cite, however, shows that the accused products do not function consistent with these theories.

a. "Releasing" where Web Cache is the "Web Server"

Where Web Cache is considered to be the "Web server" and OHS the "page server," defendant argues that Web Cache is freed to process other requests by either OHS's processing of a request or its sending of an ACK message to Web Cache. (D.I. 224 at 34) However, Web Cache's source code does not support defendant's arguments.

Web Cache processes requests using a finite number of Front End and Back End fibers. For each request it receives, Web Cache creates a Front End fiber and, in the event of a cache miss, a Back End fiber. Web Cache functions as a "Web server" only through these fibers.

The fibers for each request function independently from the fibers for any other request. Web Cache may process several requests simultaneously using different Front End and Back End fibers but, as plaintiffs' expert, Dr. Paul C. Clark, explains from the source code, no set of fibers may ever be tasked to more than one request at a time. Freeing Web Cache to process other requests, then, would require OHS (the "page server") to free the set of fibers assigned to a particular request to process other requests. Web Cache's source code, however, establishes that the Front End and Back End fibers assigned to each request always wait – that is, they do not process any other request – until OHS has processed and returned the request or a time-out occurs. Other Web Cache fibers were never tasked with processing that particular request in the first instance and so cannot be "freed" by OHS's processing of that request. Forwarding, or "delegating," the request to OHS, then, does not free Web Cache to process other requests.

Neither does OHS sending an ACK message to Web Cache free it to process other requests. Plaintiffs and defendant agree that OHS's sending of an ACK message frees TCP buffer memory in Web Cache. Plaintiffs, however, present evidence that freeing TCP buffer memory has no effect on processing other requests. First, releasing TCP buffer memory does not affect web server functionality and so does not free Web Cache to process other requests. Second, even if an ACK message could affect web server functionality, it does so only for that fiber tasked with processing that request; because the fibers function independently and only ever process one request at a time, this means that receiving an ACK message from OHS has no effect on Web Cache's ability to process other requests.

Defendant in rebuttal does not cite to anything beyond Finkel's declaration and its own claim construction analysis, neither of which presents a detailed analysis of Web Cache's function. The court finds that this is insufficient to create a genuine issue of material fact with respect to whether the accused products meet the "releasing" limitation where Web Cache is considered to be the "Web server."

b. "Releasing" where OHS is the "Web Server"

Similar to its Web Cache argument, defendant argues that, where OHS is considered to be the "Web server" and either OC4J or a RAC database instance the "page server," OHS is freed to process other requests by either OC4J or the RAC database instance processing of a request or sending an ACK message to Web Cache. (D.I. 224 at 19-20, 26) However, as with Web Cache, OHS's source code does not support defendant's arguments.

OHS's instances receive and process requests. OHS functions as a "Web server" only through these instances, which function independently. OHS's source code reveals that each OHS instance can process only one request at a time. Thus, freeing OHS to process other requests would require the "page server" to free the OHS instance assigned to a particular request to process other requests. The source code also reveals that, where an OHS instance forwards a request to OC4J or a RAC database instance, the OHS instance is unavailable to process another request until it returns the completed request or an error message. Forwarding, or "delegating," the request to the "page server" does not free the OHS instance to process other requests.

Neither does the "page server" sending an ACK message to OHS free it to process other requests. Here again, plaintiffs and defendant agree that the "page server" sending an ACK message frees TCP buffer memory in OHS. However, as with Web Cache, plaintiffs present evidence that freeing TCP buffer memory has no effect on processing other requests. First, releasing TCP buffer memory does not allow an OHS instance to process other requests; as the source code reveals, an OHS instance only ever processes one request at a time and always waits for the "page server," irrespective of whether an ACK message frees its TCP buffer memory while it waits. Second, OHS instances have their own individual TCP buffers and do not share TCP connections. Thus, one OHS instance receiving an ACK message cannot free another OHS instance to process other requests.

Defendant cites Clark's deposition testimony and the testimony of plaintiffs' expert, Dr. Michael Ian Shamos, for the proposition that receipt of ACK messages frees resources. Even accepting that as true, defendant cites no evidence suggesting that the freeing of resources equates to the freeing of an OHS instance to process additional requests.[15] Defendant otherwise in rebuttal does not cite to anything beyond Finkel's declaration and its own claim construction analysis, neither of which presents a detailed analysis of OHS's function. The court finds that defendant's evidence is insufficient to create a genuine issue of material fact with respect to whether the accused products meet the "releasing" limitation where OHS is considered to be the "Web server."

3. Indirect Infringement

Because plaintiffs do not directly infringe the patents-in-suit, plaintiffs cannot be

[15]Defendant also argues that plaintiffs do not dispute that the "Web server" – Web Cache or OHS – is released, but only that components of the "Web server" are released. (D.I. 324 at 15) Defendant, however, cites nothing in the record to suggest that either Web Cache or OHS functions as a "Web server" in any form other than a collection of fibers or instances, respectively. Defendant thus fails to support its argument that Web Cache and OHS can themselves be released independent of their fibers or instances.

found to indirectly infringe the patents-in-suit.[16]

B. Validity

1. Collateral Estoppel

As a threshold matter, defendant asserts that collateral estoppel prevents plaintiffs from asserting invalidity in this case based upon the Texas litigation.

> The party seeking to invoke collateral estoppel bears the burden, *Dici v. Pennsylvania,* 91 F.3d 542, 548 (3d Cir.1996), of showing the following four elements are present: (1) the previous determination was necessary to the decision; (2) the identical issue was previously litigated; (3) the issue was actually decided in a decision that was final, valid, and on the merits; and (4) the party being precluded from relitigating the issue was adequately represented in the previous action.

Novartis Pharmaceuticals Corp. v. Abbott Laboratories, 375 F.3d 1328, 1333 (Fed. Cir. 2004) (quoting *Hawksbill Sea Turtle v. FEMA*, 126 F.3d 461, 475 (3d Cir.1997)).

In the Texas litigation, Safelite asserted invalidity counterclaims against defendant, subsequently filed a third party complaint against plaintiffs for indemnification and, shortly thereafter, settled the case. Pursuant to the stipulation of dismissal filed in the Texas litigation, Safelite agreed to dismiss all of its claims against defendant with prejudice. Plaintiffs contributed a third of the cost of Safelite's settlement with defendant. (D.I. 281 at 10-11) The parties dispute, however, whether plaintiffs' payment to Safelite was pursuant to an indemnity agreement, in other words, whether plaintiffs were in privity with Safelite when its invalidity claim was settled and

[16]Because the court grants plaintiffs' motion for summary judgment of non-infringement (D.I. 204) and denies defendant's motion for partial summary judgment of literal infringement (D.I. 223), plaintiffs' motions for partial summary judgment of no willful infringement (D.I. 208) and excluding damages evidence based on their foreign sales (D.I. 212) are denied as moot.

dismissed.[17]

The court need not address this particular issue, insofar as it is clear on all other relevant grounds that collateral estoppel is inappropriate in this case. Defendant does not, and cannot, argue that the validity of the ‘554 and/or ‘335 patents was “previously litigated,” or that any “determination” on validity was ever made by a judge or jury. The fact that the court ordered a dismissal with prejudice in the Texas litigation, pursuant to the parties’ stipulation, has no bearing on whether “the merits” of the validity issues were ever “decided” in a final decision for collateral estoppel purposes. Defendant’s argument to the contrary warrants no further comment.[18]

2. Anticipation

Plaintiffs move for summary judgment that the following references anticipate the ‘554 and ‘335 patents: (1) the Oracle WebServer 1.0 product[19] (“Oracle 1.0”); (2) the

[17]Defendant argues that the fact that plaintiffs objected on work product grounds to counsel’s deposition questions regarding the purpose of plaintiffs’ payment to Safelite evidences that the payment was made “to satisfy its indemnity obligations.” (D.I. 281 at 11) Plaintiffs do not dispute that they paid Safelite, but argue that they never agreed to indemnify Safelite, and a settlement payment cannot be used to demonstrate indemnification liability. (D.I. 320 at 5)

[18]In view of the court’s finding that defendant’s collateral estoppel argument has no merit, the court need not reach plaintiffs’ motion to strike defendant’s newly-iterated collateral estoppel defense. (D.I. 371 at ¶ 39) The court notes, however, that defendant’s amended answer was filed October 24, 2008 – well past the court’s imposed deadline for motions to amend pleadings – and was unaccompanied by a motion for leave to amend. Notwithstanding, plaintiffs’ motion (D.I. 387) is denied as moot.

[19]With respect to the references identified as “products,” the parties have described the “products” through use of multiple documentary references (e.g., the Oracle Web Server 1.0 “product” is described through use of the 1.0 User’s Guide and Reference 503) rather than through the use of source code. The court notes, without deciding, whether this is appropriate vis a vis an anticipation analysis.

Oracle WebServer 2.0 product ("Oracle 2.0"); (3) U.S. Patent No. 6,249,241 to Popp et al. ("Popp"); (4) Lagoze et al., Dienst: "Implementation and Reference Manual," Cornell University Technical Report TR-95-1514 (May 5, 1995) ("Deinst"); and (5) any of the foregoing in combination with Garland et. al, "Implementing Distributed Server Groups for the World Wide Web," Carnegie Mellon University, Technical Report CMU-CS-95-114 (January 25, 1995) ("Garland"). Defendant's motion for partial summary judgment of no anticipation is not in alignment with plaintiffs' motion. Defendant moves for judgment that the following references describing Oracle 1.0 and Oracle 2.0 do not, themselves, anticipate the claims: (1) "Oracle WebServer User's Guide, Release 1.0, Part No. A34986-2" (the "1.0 User's Guide"[20]); (2) "The Oracle WebServer: A Technical Discussion" (hereinafter, "Reference 503"); (3) "Understanding SQL*Net," Release 2.2, part no. A32090-1 ("SQL*Net 2.2"); (4) "Understanding SQL*Net," Release 2.3, part no. A42484-1 ("SQL*Net 2.3"); all of which are relevant to Oracle 1.0, and (5) "Oracle WebServer 2.0, Technical Note, March 1996" (hereinafter, "Reference 605"); and (6) the "Oracle WebServer User's Guide Release 2.0 Production, Part No. A23646-1 (the "2.0 User's Guide"), which describe Oracle 2.0.[21]

a. Anticipation standards

Under 35 U.S.C. § 102(a), "a person shall be entitled to a patent unless the

[20]The parties refer to this document as "Reference 153."

[21]The parties have, unnecessarily, submitted multiple copies of the relevant prior art. For ease of reference, the court will cite the following copies of the prior art throughout its discussion: 1.0 User's Guide (D.I. 214, ex. 40); Reference 503 (*id.*, ex. 41); Reference 605 (*id.*, ex. 44); SQL*Net 2.2 (*id.*, ex. 49); SQL*Net 2.3 (*id.*, ex. 50); 2.0 User's Guide (*id.*, ex. 42).

invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the application for patent."

A claim is anticipated only if each and every limitation as set forth in the claim is found, either expressly or inherently described, in a single prior art reference. *Verdegaal Bros., Inc. v. Union Oil Co.*, 814 F.2d 628, 631 (Fed. Cir. 1987).

> [A]nticipation requires that each limitation of a claim must be found in a single reference. Although [the Federal Circuit has] permitted the use of additional references to confirm the contents of the allegedly anticipating reference, . . . we have made clear that anticipation does not permit an additional reference to supply a missing claim limitation.

Teleflex, Inc. v. Ficosa North America Corp., 299 F.3d 1313, 1335 (Fed. Cir. 2002). That is, additional references may be used only to shed light on what a prior art reference would have meant to those skilled in the art at that time, not for a specific teaching, as this would be indicative of an attempt to improperly "combine the teachings of the references to build an anticipation." *Studiengesellschaft Kohle, m.b.H. v. Dart Industries, Inc.*, 726 F.2d 724, 727 (Fed. Cir. 1984).

A single prior art reference may expressly anticipate a claim where the reference explicitly discloses each and every claim limitation. However, the prior art need not be *ipsissimis verbis* (i.e., use identical words as those recited in the claims) to be expressly anticipating. *Structural Rubber Prods. Co. v. Park Rubber Co.*, 749 F.2d 707, 716 (Fed. Cir. 1984).

A single prior art reference also may anticipate a claim where one of ordinary skill in the art would have understood each and every claim limitation to have been disclosed inherently in the reference. *Continental Can Co. USA Inc. v. Monsanto Co.*, 948 F.2d 1264, 1268 (Fed. Cir. 1991). The Federal Circuit has explained that an inherent limitation is one that is necessarily present and not one that may be established by probabilities or possibilities. *Id.* That is, "the mere fact that a certain thing may result from a given set of circumstances is not sufficient." *Id.* "[I]nherency operates to anticipate entire inventions as well as single limitations within an invention." *Schering Corp. v. Geneva Pharms. Inc.*, 339 F.3d 1373, 1380 (Fed. Cir. 2003). The recognition of an inherent limitation by a person of ordinary skill in the art before the critical date is not required to establish inherent anticipation. *Id.* at 1377.

An anticipation inquiry involves two steps. First, the court must construe the claims of the patent in suit as a matter of law. *Key Pharms. v. Hercon Lab. Corp.*, 161 F.3d 709, 714 (Fed. Cir. 1998). Second, the finder of fact must compare the construed claims against the prior art to determine whether the prior art discloses the claimed invention. *Id.*

b. The references's availability as prior art

i. The patents' priority date

At issue in both motions is whether defendant is entitled to a priority date of earlier than April 23, 1996, the patents' filing date. Defendant argues that plaintiffs cannot prove that any of their asserted references antedate its invention. Defendant claims that the inventions were conceived in August 1995 and reduced to practice as early as November 1995. More specifically, defendant claims that "Virtuoso," later renamed the "ForeSite Application Server," was the first commercial embodiment of the claimed inventions. (D.I. 218 at 21) Plaintiffs contend that defendant has offered no proof that Virtuoso embodied all of the elements of the claimed inventions. (D.I. 267 at 18)

Defendant's evidence regarding prior invention may be summarized as follows: (1) Keith Lowery, named co-inventor of the '554 and '335 patents, testified at his deposition that he had all of the pieces of the invention working in experimental form sometime between August and October 1995 (D.I. 222, ex. 24 at 53); (2) Lowery sent a draft of a conceptual description of a product to a Bob West on September 15, 1995 (D.I. 221, ex. 17 at EPIC000172)[22]; (3) on October 15, 1995, InfoSpinner, Inc., a company formed to develop the invention, published a press release entitled "InfoSpinner Status" which defendant asserts discusses the development of the claimed inventions (D.I. 221, ex. 15); (3) Lowery testified that he demonstrated a prototype of the application server product to Software AG in January 1996 (D.I. 222, ex. 24 at 113); and (4) on February 28, 1996, InfoSpinner signed a distribution agreement with Software AG for Virtuoso (*Id.*, ex. 18). All of these events precede the patents' April 23, 1996 priority date.

Plaintiffs assert that defendant is not entitled to an earlier priority of invention. Plaintiffs point to several pieces of evidence in support of their position: (1) named co-inventor Andrew Levine testified that the InfoSpinner press release did not disclose a

---

[22]This product was called "WebSpinner," and was described as a tool for the creation of documents "on-the-fly" using dynamic data. (D.I. 221, ex. 17 at EPIC000168) Defendant states that Lowery's initial product was "Virtuoso" (D.I. 219 at 21), therefore, the relationship (if any) between WebSpinner and Vitruoso is unclear to the court.

dispatcher and page servers[23] (D.I. 272, ex. 77 at 104-05); and (2) Shamos declared that he has reviewed defendant's proffered evidence and does "not believe such evidence supports an invention date earlier than the filing date . . . For example, none of the documents . . . discloses the claimed step of the page server 'releasing' the Web server, which all the asserted claims require." (D.I. 271 at ¶ 46 n.3). (D.I. 267 at 17-18) Shamos's statement is highly conclusory and, taken alone, would not withstand summary judgment. Levine's testimony, however, suffices to raise a genuine issue of material fact with respect to whether the claimed invention was conceived prior to April 1996.

ii. Public availability of Oracle 1.0 and Oracle 2.0

Defendant asserts that neither Oracle 1.0 or Oracle 2.0 are prior art because Oracle cannot prove that the products were known or used by others in this country. (D.I. 219 at 19-20) Katrina Montinola, plaintiffs' Director of Engineering in the Network Products Division from 1990 to 1996, submitted a declaration in which she states that Oracle 1.0 was functional by the end of July 1995, released to the public in beta form on September 25, 1995 and was commercially released in October 1995. (D.I. 268 at ¶¶ 5-6, 8) In support, Montinola references a press release dated September 25, 1995, stating that Oracle 1.0 ("Web Server") had been released (in beta form) the week prior at the "International Oracle User Group conference" (*id.*; D.I. 214, ex. 53), and a copy of her presentation on Oracle 1.0 given at that conference (D.I. 214, ex. 41). She also cites an October 30, 1995 press release stating that Oracle 1.0 was "formally launched"

[23]Levine claims that he, not Lowery, conceived of the dispatcher and page server functionality of the claimed inventions. (D.I. 272, ex. 77 at 110-13)

to the public at large on that date. (*Id.*, ex. 55)

Montinola also states that Oracle 2.0 was disclosed to the public in February 1996. (*Id.* at ¶ 13) In support, she relies upon a press release dated February 21, 1996, stating that plaintiffs unveiled Oracle 2.0 at "Internet Expo '96" that same day. (*Id.*; D.I. 214, ex. 58) The foregoing establishes that defendant is not entitled to judgment that plaintiffs cannot prove that Oracle 1.0 or Oracle 2.0 are statutory prior art.

iii. Publication date of the relevant documentary references

Several of the documentary references at issue are undated or contain only the year of copyright on their face. In her declaration, Montinola provides the following publication dates for the references at issue.

(1) the 1.0 User's Guide: October 30, 1995

(2) Reference 503: September 25, 1995[24]

(3) SQL*Net 2.2: August 1995

(4) SQL*Net 2.3: March 1996[25]

(5) Reference 605: March 1996

(6) the 2.0 User's Guide: April 1996

(D.I. 268 at ¶¶ 6, 7, 9, 16)

---

[24] Montinola dates Reference 503 a week after the beta release of Oracle 1.0. Contrary to defendant's assertion, this statement does not contradict her deposition testimony, in which she reflected that Reference 503 was written in connection with a September 1995 presentation and which she believed was made public, although she could not be "one hundred percent sure." (D.I. 222, ex. 25 at 126)

[25]Montinola states that the SQL*Net 2.2 and SQL*Net 2.3 dates provided coincide with the public release of the products they describe. (D.I. 268 at ¶¶ 6, 16)

Montinola's proffered date for the 1.0 User's Guide coincides with the formal launch of Oracle 1.0, as described by the references discussed *supra*. The proffered date for the 2.0 User's Guide actually post-dates the February 1996 public "unveiling" of Oracle 2.0; Montinola states that Oracle 2.0 was commercially released no later than April 1996. (*Id.* at ¶ 15) The court is satisfied that Montinola's proffered testimony creates an issue of fact with respect to the publication dates of the references at issue in this case. Defendant is not entitled to judgment that plaintiffs cannot prove that Oracle 1.0 or Oracle 2.0 are statutory prior art.

c. Oracle 1.0

i. Background

According to the 1.0 User's Guide, Oracle 1.0 operated in the following manner. A user's web browser sent a request to the "Oracle Web Listener," or "a commercial quality HTTP server that services document requests." (D.I. 214, ex. 40 at ORCL000022) The Web Listener could handle a large amount of simultaneous requests. (*Id.*) First, the Web Listener "determines whether th[e] request is for a static document or a dynamic document. If the request is for a static document, the Web Listener sends the file and the associated type information directly to the client. If the request is for a dynamic document, it is created 'on the fly' by a program invoked by the Web Listener, in compliance with [CGI]." (*Id.*) This program was called the "Oracle Web Agent." Once the Web Listener starts the Web Agent, the Web Listener "makes several environment variables available to the Web Agent in compliance with the CGI specification." (*Id.* at ORCL000115)

> The Web Agent then connects to the "Oracle7 Server."
>
> In order to connect to an Oracle 7 server, the Web Agent requires certain information, such as which server to connect to and what username and password to use. This information is stored . . . as part of a Web Agent service. The Web Agent determines which service to use by parsing the . . . environment variable . . . set up by the Web Listener[.]

(*Id.*) The Oracle7 Server executes the PL/SQL[26] procedure that was specified as part of the request. (*Id.*)

ii. "Dispatching"

The parties' primary disagreement with respect to Oracle 1.0 and Oracle 2.0 is whether either performed "dispatching" as required by the claims. By its order of the same date, the court has construed "dispatching said request to said page server" as that limitation appears in the asserted claims as "[a]nalyzing a request to make an informed selection as to which page server should process the request based on a variety of information (both static and dynamic), and sending the request to the selected page server."

In describing Oracle 1.0, plaintiffs rely primarily on the 1.0 User's Guide, which they also assert discloses "dispatching" as required by the claims. However, plaintiffs also rely upon SQL*Net 2.2 and SQL*Net 2.3 in support for their argument that Oracle 1.0 performs "dispatching" using SQL*Net.[27] (D.I. 207 at 13; D.I. 257 at ¶¶ 50-52) According to plaintiffs, the Web Agent is the dispatcher in Oracle 1.0. It is a "separate

[26]Procedural Language/Structured Query Language. PL/SQL is one of three languages embedded in the Oracle Database. The other two are SQL and Java.

[27]Plaintiffs also rely on SQL*Net 2.2 in arguing that Oracle 1.0 anticipates claim 4 of the '554 patent, requiring a "connection cache." (D.I. 207 at 15)

program" invoked by the Web Listener when a request for a dynamic document is received. (D.I. 207 at 12) Plaintiffs cite the declaration of Shamos, who relies upon the following passage from the 1.0 User's Guide:

> The Oracle Web Agent is a program that is invoked by the Oracle Web Listener when a request for a database procedure is received. It handles the details of making a connection to the Oracle7 Server. A Web Agent will connect to a single Oracle7 Server, using a specific database username and password which are specified as part of a Web Agent service. To connect to different servers, or different schemas on the same server, multiple Web Agent services may be configured on a single Oracle WebServer. . . .

(D.I. 207 at 12; D.I. 257 at ¶ 48, citing D.I. 214, ex. 40 at ORCL000023) Additionally, the 1.0 User's Guide provides that "[w]hen a request from a Web browser comes in, the Web Listener will extract the service name that is embedded in the URL and find out which parameters to use by reading the owa.cfg [configuration] file." (D.I. 214, ex. 40 at ORCL000117) Shamos states that this configuration file is dynamic information used to make an informed choice of which Web Agent service is to be invoked to process the request. (D.I. 257 at ¶ 48)

Defendant argues that the Web Agent does not make an "informed selection" because there is only one Oracle7 (page) server that is capable of processing a given request; the Web Agent makes the "only" selection it can possibly make.[28] (D.I. 281 at

[28]Defendant argued in its papers that Oracle 1.0 cannot anticipate the claims, insofar as: "(1) the Web Agent does not use information indicating which page server can more efficiently process a request; [and] (2) the configuration file used by the Web Agent is not 'dynamic information'[.]" (D.I. 323 at 6; D.I. 273 at ¶ 43 ("Dynamic information does not refer to information that can be established prior to the execution of the program . . . [it] can only be determined during runtime execution.")) These arguments are foreclosed by the court's claim construction, insofar as the court does not limit the "dispatching" limitation to an informed selection of which page server "can more efficiently process the request," and the court has also declined to limit the nature of the information used to make the "informed selection" to only dynamic information.

16; D.I. 273 at ¶ 41) Shamos admits that an "informed selection" cannot occur absent multiple page servers.[29] Plaintiffs point out, however, that the 1.0 User's Guide states that the Web Agent can connect to one of a variety of different database servers:

> A Web Agent will connect to a single Oracle7 server, using a specific database username and password which are specified as part of a Web Agent service. **To connect to different servers**, or different schemas on the same server, multiple Web Agent services may be configured on a single Oracle Web Server.

(D.I. 214, ex. 40 at ORCL000023) (emphasis added) The more technical section of the 1.0 User's Guide, that explaining precisely "[h]ow the Oracle Web Agent [w]orks," provides:

> 1. The Web Agent connects to the Oracle7 server. In order to connect to an Oracle7 server, the Web Agent requires certain information, such as **which server to connect to** and what username to use. This information is stored in the owa.cfg file . . . as part of a Web Agent Service.

(*Id.* at ORCL000114) (emphasis added)

Plaintiffs state, without further support, that the foregoing demonstrates that Oracle 1.0 chooses from more than one page server and, therefore, meets the "dispatching" limitation. (D.I. 320 at 16) In its own motion, defendant argues that the 1.0 User's Guide discloses only one server, also without citing any testimony in support. (D.I. 219 at 24; D.I. 323 at 6-7) Absent any direct evidence regarding how a person of ordinary skill in the art would interpret the foregoing disclosure and, more importantly, absent any indication of a lack of a genuine issue of material fact on this point, the court

[29]Shamos specifically stated in his expert report that "[w]ith only a single page server, there cannot be any 'selection' at all, let alone an informed one. . . It is my opinion that systems having only a single page server cannot satisfy the limitation of 'dispatching'." (D.I. 239 at ¶¶ 103-04)

declines to grant judgment in favor of either party with respect to the 1.0 User's Guide.[30]

iii. "Dispatching": plaintiffs' alternate theory

Plaintiffs have an alternative theory on why the "dispatching" limitation is met. The 1.0 User's Guide states that SQL*Net may optionally be used between the Web Agent and the Oracle7 server.[31] (D.I. 214, ex. 40 at ORCL000022) Shamos relies on SQL*Net 2.2 to describe what he terms "SQL*Net dispatching." As described by SQL*Net 2.2, a "Network Listener" receives a request and has several options on how to establish connections and route the request. (D.I. 214, ex. 49 at ORCL0180664, fig. A-9) Shamos opines that "dispatching" also occurred using the method of load balancing described in SQL*Net 2.3, which describes a later version of SQL*Net. (D.I. 257 at ¶ 52; D.I. 214, ex. 50 at ORCL01808646-47)

[30]Defendant asserts that, for the same reasons as explained with respect to the 1.0 User's Guide, Reference 503 does not disclose dispatching. (D.I. 219 at 25) In their opposition papers, plaintiffs merely state that defendant admits that "Reference 503 discloses *many* of the same features of [Oracle 1.0] which [the 1.0 User's Guide] discloses." (D.I. 267 at 28) Plaintiffs do not move for summary judgment of anticipation based upon Reference 503. Absent any indication of a triable issue with respect to Reference 503, the court grants defendant's motion of no anticipation with respect to this reference.

[31]The 1.0 User's Guide further provides:

> To connect to the Oracle7 Server, the Web Agent needs the following information to be specified in the Web Agent service: • username • password • ORACLE_HOME • ORACLE_SID (for local databases only) • SQL*Net V2 Service Name or Connect String (for remote databases only). The Administration Utility allows the administrator to display, create, modify or delete a Web Agent service. With the Web Agent Creation form, you do not need to modify the configuration file for the Oracle Web Agent service (owa.cfg) directly.

(D.I. 214, ex. 40 at ORCL000117-19)

The court finds that plaintiffs' use of multiple references to describe the purported "dispatching" function of the Web Agent of Oracle 1.0 falls into the category of conduct prohibited by *Studiengesellschaft Kohle*. 726 F.2d at 727. The references to SQL*Net present in the 1.0 User's Guide are limited and general. Plaintiffs do not offer SQL*Net 2.2 (or SQL*Net 2.3) to shed light on SQL*Net generally, but to make the argument that a specific function of SQL*Net constitutes "dispatching" in and of itself. Plaintiffs may certainly use the discussion of SQL*Net in the 1.0 User's Guide in a motivation to combine argument under 35 U.S.C. § 103, but it may not broaden anticipation so as to reach their alternative argument regarding "dispatching" in this manner. For this reason, the court grants defendant's motion for partial summary judgment of no anticipation based on the 1.0 User's Guide in combination with SQL*Net 2.2 or SQL*Net 2.3. Because plaintiffs do not assert that either SQL*Net 2.2 or SQL*Net 2.3 are, in and of themselves, anticipatory references, the court also grants defendant's motion of no anticipation by either SQL*Net 2.2 or SQL*Net 2.3.

d. Oracle 2.0

With respect to Oracle 2.0, plaintiffs rely interchangeably upon the 2.0 User's Guide and Reference 605 in support of their anticipation arguments.[32] Nowhere in their papers do plaintiffs clearly assert that either reference discloses all of the claimed limitations. Plaintiffs predominantly discuss the 2.0 User's Guide in their arguments. Plaintiffs cite both references for their purported disclosures of "dispatching." (D.I. 207

[32]Plaintiffs also cite the publication "Oracle WebServer Release 2.0 – The Universal Information Server for Business Applications (1996)" generally to demonstrate the architecture of Oracle 2.0. (D.I. 207 at 19, citing D.I. 214, ex. 43, fig. 2)

at 21) In their own motion, plaintiffs rely solely on Reference 605 for the "intercepting" limitation (*id.*, citing D.I. 214, ex. 44; D.I. 257 at ¶¶ 86-87), but cite a portion of the 2.0 User's Guide in response to defendant's motion (D.I. 267 at 36, citing D.I. 214, ex. 42 at ORCL01816643).

There is no indication that plaintiffs are attempting to use one reference to broadly shed light on the disclosure of the other, such as may constitute a limited exception to the prohibition against using multiple references for anticipation. Plaintiffs' motion for summary judgment of anticipation based on Oracle 2.0 must be denied for this fundamental reason. The court will turn to defendant's motion for partial summary judgment that the 2.0 User's Guide and Reference 605 do not anticipate the claims at issue.

i. Background

As described by Reference 605, Oracle 2.0 comprises a Web Listener, Web Request Broker, including a Web Request Broker Dispatcher ("WRB Dispatcher") and Web Request Broker Engines ("WRBXs"), and the WebServer SDK. The Web Listener delivers incoming requests to the Web Request Broker. (D.I. 214, ex. 44 at ORCL000763) The WRB Dispatcher "decide[s] what type of object is being requested" so that it may dispatch the request to the appropriate WRBX. (*Id.* at ORCL000766) "The WRB Dispatcher performs dynamic load balancing between multiple instances of a WRB Service, and the Webmaster can configure each WRB Service to have a minimum and maximum number of instances." (*Id.*) The WRB Dispatcher examines a WRB configuration file in order to decide what type of object is being requested. (*Id.*) If no suitable WRBX is found, the request is passed back to the Web Listener for standard processing. (*Id.*)

ii. "Dispatching"

Defendant argues that neither the 2.0 User's Guide or Reference 605 anticipate the asserted claims because neither disclose "dispatching." That is, the references "only disclose that the WRB Dispatcher narrows the universe of WRBXs capable of processing a request based on: (1) the request; (2) a configuration file; and (3) whether a WRBX is already processing another request," and do not "disclose how the WRB Dispatcher chooses between the WRBXs capable of processing the request that are not already processing another request." (D.I. 219 at 34) Put another way, "a random selection would not be an 'informed selection.'" (*Id.* at 35)

Plaintiffs admit that, when two or more WRBXs are free at once, the WRB Dispatcher "selects a free WRBX based on the order in which the WRBXs were originally assigned requests." (D.I. 267 at 38) Plaintiffs claim, however, that "[t]his order can obviously change dynamically" because, as disclosed by the 2.0 User's Guide, the WRB Dispatcher "creates new WRBXs as needed." (*Id.*; D.I. 214, ex. 42 at ORCL01816653) Additionally, plaintiffs argue that "regardless of how [the WRB Dispatcher] makes its final selection it will still have relied on the three pieces of dynamic information" discussed above: (1) the request; (2) a configuration file; and (3) whether a WRBX is busy. (D.I. 267 at 38; D.I. 271 at ¶ 95)

The court agrees with defendant that the 2.0 User's Guide does not actually disclose how these pieces of information are used to select a WRBX, whether existing or newly-created. Plaintiffs point to no such information. (*See id.* ("regardless of how [the WRB Dispatcher] makes its final selection . . .")) Shamos cites Reference 605's statement that the "WRB Dispatcher performs dynamic load balancing"; even in this reference, there is no citation regarding precisely how WRBXs use the so-called "dynamic information" referenced above. (D.I. 257 at ¶ 89, citing D.I. 214, ex. 44 at ORCL0000766) The 2.0 User's Guide's discussion regarding the creation of and connection to new WRBXs is equally deficient. (*Id.*, citing D.I. 214, ex. 42 at ORCL01816653) Plaintiffs' argument is an inherency argument missing the requisite evidence that their proffered interpretation of either disclosure is more than a "probabilit[y] or "possibilit[y]."[33] *See Continental Can Co. USA, Inc.*, 948 F.2d at 1269. The court grants defendant's motion for partial summary judgment that neither the 2.0 User's Guide nor Reference 605 anticipate the asserted claims.[34]

e. Popp

Plaintiffs move for summary judgment that the asserted claims are invalid as anticipated by Popp and also by Dienst. The only issue disputed by the parties with respect to both references is the "dispatching" limitation of the asserted claims.

Plaintiffs assert that the CGI interface program disclosed in Popp performs

---

[33]*See Continental Can Co. USA, Inc. v. Monsanto Co.*, 948 F.2d 1264, 1269 (Fed. Cir. 1991) ("Inherency, however, may not be established by probabilities or possibilities. The mere fact that a certain thing may result from a given set of circumstances is not sufficient. If, however, the disclosure is sufficient to show that the natural result flowing from the operation as taught would result in the performance of the questioned function, it seems to be well settled that the disclosure should be regarded as sufficient.") (citation omitted).

[34]The court need not address defendant's arguments regarding the "intercepting" limitation.

"dispatching." In support, plaintiffs rely on Shamos, whose declaration provides that the "CGI Messenger 210 or Web Objects Server 912," as described in Popp, "examines the incoming request and dispatches it based on dynamic information maintained about the applications 214 (the page servers)[.]" (D.I. 257 at ¶ 126[35]) "Accordingly, CGI interface

---

[35]Shamos identifies the following excerpts of Popp as those describing "dispatching":

> To service the client request, CGIMessenger 210 communicates with HTTP server 206 to obtain information regarding the client request. For example, CGIMessenger 210 obtains any information that accompanied the user request such as form and/or application names and/or user input. The form names can include the name of the form that was submitted to initiate the client request as well as a return form.
> The application name identifies the application that services the client request. If an application is specified, CGIMessenger 210 transmits the client request and corresponding information transmitted from HTTP Server 206 to application 214. Application 214 can execute on the same or different server as CGIMessenger 210 and/or HTTP Server 206, for example. Application 214 executes an interaction flow to satisfy the user request. Application 214 can access an external data source such as database 224. Database 224 can be resident on the same server as application 214. Alternatively, database 224 can be resident on a separate server (e.g., a separate database server).
> * * *
> FIG. 5A provides an example of a process flow executed by the Web Objects server to manage applications. A function of the Web Objects server is to dispatch a request to an application that processes the request.
> At step 502, the request is received by the Web Objects server. Using the information contained in the request, the Web Objects server identifies the application that will process the request. At step 504, the Web Objects server determines whether the application is executing either on the same or a different server, for example. If the application is not currently loaded, the Web Objects server initiates the application at step 508 and processing continues at step 506. If the application is executing, processing continues to step 506. At step 506, the request is forwarded to the application.

program 322 dispatches requests [under either party's construction] . . . because it examines requests to make an informed selection of which page server (application server 214) should process requests based on dynamic information maintained about page servers." (*Id.*) Additionally, plaintiffs point out that Finkel, defendant's expert, testified that Popp meets the "dispatching" limitation. (D.I. 207 at 30, citing D.I. 213, ex. 25 at 100[36])

In opposition to plaintiffs' motion, defendant asserts that Popp's CGI inferface

---

(D.I. 214, ex. 45 at col. 7, ll. 16-35 and col. 27 ll. 39-55)

[36]Finkel apparently was given a copy of an "agreed claim construction for dispatching," and the following exchange occurred:

Q. "Examining a request to make an informed selection" is present, correct?
A. I'm not sure it makes a selection. Yeah, I guess it makes a selection. I'm not sure it makes a selection. It just goes to the one application that can process it, and I think selection – to do the selection you have to do the dynamic information.
Q. Well–
A. Examining the request.
Q. That portion of the claim construction says, "Examining a request to make an informed selection of which page server should process the request." And then it has a semicolon in the construction. Do you see that?
A. Yes.
Q. That portion of dispatching is performed by the '291 Popp patent, correct?
A. Yes.
Q. And the portion of dispatching that says "sending the request to the selected page server" is also present in the '291 Popp patent, correct?
A. Yes.
Q. So the only part that is not present in your opinion is the part between the two semicolons, ["]based on dynamic information maintained about page servers, the dynamic information indicating which page server can more efficiently process the request["], correct?
A. Yes.

(D.I. 213, ex. 25 at 100-01) As discussed previously, the latter arguments are foreclosed by the court's claim construction.

program does not make an "informed selection" of which page server should process a request because only one application exists which could possibly process each request. (D.I. 281 at 31) Finkel states the same in his second declaration filed in opposition to plaintiffs' motion, without further comment or support. (D.I. 273 at ¶ 59)

In their reply papers, plaintiffs argue that Popp discloses that "a CGI interface program can dispatch requests to one of multiple applications (*i.e.*, page servers) located on the same or different server machines." (D.I. 320 at 10) Popp discloses that the CGI interface program "is used to route requests to applications." (D.I. 214, ex. 45 at col. 7, ll. 5-7) The portion of Popp cited by plaintiffs states merely that "[t]he self-contained modules can be shared by one or more Web pages in a single application and/or across multiple applications executing on application server 316 (or any other server)." (D.I. 320 at 10, citing D.I. 214, ex. 45 at col. 4, ll. 30-34) Plaintiffs point to no expert opinion on this issue. (*Id.*)

As noted previously, Shamos admits that "systems having only a single page server cannot satisfy the limitation of 'dispatching'." (D.I. 239 at ¶¶ 103-04) The portion of Shamos's declaration discussing Popp only generally references multiple page servers; he does not explain or describe how multiple "applications" reside on more than one server, as plaintiffs assert. (D.I. 257 at ¶ 126 ("applications 214 (the page servers)"; "which page server (application server 214)") Plaintiffs' best evidence can be characterized as impeachment testimony of Finkel. This testimony went to the "ultimate question" of whether Popp disclosed "dispatching"; it does not make plaintiffs' case in the first instance – that is, explain how Popp's disclosure would be interpreted by a person of ordinary skill in the art as disclosing that more than one page server is available to receive a request. In view of the lack of evidence in this regard, the court denies plaintiffs' motion for summary judgment.[37]

f. Dienst

The court reaches the same conclusion with respect to Dienst. Dienst provides access to a digital library over the web. The digital works are contained on servers called "Dienst servers," containing a document repository wherein "documents from a given publisher must all be stored in a single repository." (D.I. 214, ex. 46 at MAC000280)

> Each server, in a set of interoperating Dienst servers, is capable of responding to a request for any document in the distributed collection. The respective server uses the publisher in the docid of the requested document and the mapping tables downloaded from the meta server to route the document request to the server that acts as the repository for the publisher. This routing is transparent to the user.

(*Id.*, ex. 46 at MAC000281) According to Shamos, "dispatching" occurs when one Dienst server, armed with "dynamic information," routes a request to another Dienst server. (D.I. 257 at ¶ 161) Defendant argues that Dienst discloses only one server that can be accessed to retrieve the works of a specific publisher; an "informed choice" cannot occur when only one option is available. (D.I. 281 at 35) In Finkel's words, "when the alleged 'dispatcher' selects a Dienst server to process a request, it does not make a selection, but rather locates the only Dienst server that contains that publisher's works." (D.I. 273 at ¶ 66) Finkel's opinion is sufficient to preclude summary judgment for plaintiffs on this issue.

---

[37]Defendant did not move for summary judgment of no anticipation based on Popp or Dienst. (D.I. 219)

3. Obviousness

a. Standards

"A patent may not be obtained . . . if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art." 35 U.S.C. § 103(a). Obviousness is a question of law, which depends on several underlying factual inquiries.

> Under § 103, the scope and content of the prior art are to be determined; differences between the prior art and the claims at issue are to be ascertained; and the level of ordinary skill in the pertinent art resolved. Against this background the obviousness or nonobviousness of the subject matter is determined. Such secondary considerations as commercial success, long felt but unsolved needs, failure of others, etc., might be utilized to give light to the circumstances surrounding the origin of the subject matter sought to be patented.

KSR Int'l Co. v. Teleflex Inc., 127 S. Ct. 1727, 1734 (2007) (quoting *Graham v. John Deere Co.*, 383 U.S. 1, 17-18 (1966)). "Because patents are presumed to be valid, *see* 35 U.S.C. § 282, an alleged infringer seeking to invalidate a patent on obviousness grounds must establish its obviousness by facts supported by clear and convincing evidence." *Kao Corp. v. Unilever U.S., Inc.*, 441 F.3d 963, 968 (Fed. Cir. 2006) (citation omitted).

"[A] patent composed of several elements is not proved obvious merely by demonstrating that each of its elements was, independently, known in the prior art." *KSR*, 127 S. Ct. at 1741. Likewise, a defendant asserting obviousness in view of a combination of references has the burden to show, by clear and convincing evidence, that a person of ordinary skill in the relevant field had a reason to combine the elements in the manner claimed. *Id.* at 1741-42. The Supreme Court has emphasized the need for courts to value "common sense" over "rigid preventative rules" in determining whether a motivation to combine existed. *Id.* at 1742-43. "[A]ny need or problem known in the field of endeavor at the time of invention and addressed by the patent can provide a reason for combining the elements in the manner claimed." *Id.* at 1742.

In addition to showing that a person of ordinary skill in the art would have had reason to attempt to make the composition or device, or carry out the claimed process, defendant must also demonstrate by clear and convincing evidence that "such a person would have had a reasonable expectation of success in doing so." *PharmaStem Therapeutics, Inc. v. ViaCell, Inc.*, 491 F.3d 1342, 1360 (Fed. Cir. 2007).

b. Plaintiffs' contention

Plaintiffs seek judgment that the asserted claims are invalid in view of Oracle 1.0, Oracle 2.0, Popp, or Dienst in view of Garland.[38] In its "Introduction," Garland acknowledges that the "exponential growth" of the internet has often resulted in a "prohibitively high" traffic load for single-server networks. (D.I. 214, ex. 48 at ORCL01818181) "Service provider[s] need[ ] more computing power to meet the demands of the clients requesting the service." (*Id.*)

> A fairly obvious solution to this problem is to implement a distributed Web server which will spread incoming request load among several machines. There is, however, a significant problem with this approach. The [HTTP] protocol used by

[38]Garland appears to be a student thesis for the School of Computer Science of Carnegie Mellon University in Pittsburgh, Pennsylvania. On its face, Garland provides a Carnegie Mellon University publication date of January 25, 1995, rendering it statutory prior art pursuant to 35 U.S.C. § 102(b). Garland also provides a second publication date by the National Technical Information Service of March 20, 1995. (D.I. 214, ex. 48)

> Web clients and servers identifies a resource on the Web using a valid hostname. Consequently, a distributed server approach needs to present a single hostname to maintain compatibility with existing Web clients. We can alleviate this problem by extending the concept of distributed process groups to the Web server level. . . .
>
> By supporting distributed Web server groups, we can provide a mechanism for smoothly scaling the server capacity of Web service providers. The use of server groups can reduce the latency of serving a request provided there is sufficient available network bandwidth to accommodate higher request throughput. Our Web server group implementation can efficiently balance request load, and is transparent to Web clients which support the HTTP/1.0 protocol.

(*Id.*) The system architecture described by Garland is illustrated in figure 5 of that reference, and is described as follows.

> There is a special server (the ***dispatcher***) which coordinates and controls the server group. Requests intended for the server group are received by the dispatcher.
>
> Attached to the dispatcher is a collection of ***member*** servers. These servers are the actual repository of data and fulfill requests made of the group.
>
> The dispatcher is the single entity representing the group to the outside world. Incoming requests are received by the dispatcher, and they are redirected to one of the member servers for actual processing. It is also the task of the dispatcher to balance the load placed upon the various members under its control. In essence, it tries to always dispatch incoming requests to the last loaded member.

(*Id.*, ex. 48 at ORCL01818183) (emphasis in original)

Garland also discusses "load balancing" by the dispatcher.

> The task of the dispatcher is to distribute incoming requests to members of the server group. However, simply doing this without making informed decisions about ***where*** to send requests would not be very desirable. We want the dispatcher to distribute requests in such a way as to balance the load placed on the various member servers. This requires two things: the dispatcher must have some knowledge of the load on members, and there must be some scheduling policy used to select which member to dispatch the next request to.

(*Id.*, ex. 48 at ORCL01818185) (emphasis in original) Garland proceeds to discuss the manner in which load may be measured and the periodic transfer of load information to the dispatcher. The dispatcher "uses a simple scheduling policy," keeping members in "a doubly-linked priority queue, sorted by increasing load. Thus, dispatching a request simply involves looking at the head of the queue and returning that member address to the requesting client." (*Id.*)

According to Shamos, "Garland is agnostic as to the type of processing done by the member servers, which can include dynamic Web page generation." (D.I. 257 at ¶ 189) Notwithstanding, Garland's introduction provides an express motivation to the distributed Web server architecture and load balancing techniques of Garland with any system suffering the drawbacks of high traffic volume. (*Id.* at ¶ 190) Shamos states that Garland also provides a motivation to combine load balancing techniques with more modern server architectures because such a combination is provided in Garland's figure 5 and discussion of its system architecture; additionally, the bibliography of Garland "cites to a variety of load balancing prior art references that describe load balancing techniques in various system and server architectures that are not necessarily in the field of Internet technology." (*Id.*) Shamos concludes that a combination of Garland and any of the alleged anticipatory references "would result in the above prior art references having the dispatching capabilities of Garland, namely the use of dynamic load balancing techniques to dispatch Web requests from a Web server to one or more page servers." (*Id.* at ¶ 191)

Defendant argues that Garland does not disclose "dispatching," rendering plaintiffs' asserted combination deficient as a matter of law. (D.I. 281 at 37) In support, defendant cites only to Finkel's second declaration, which states that "Oracle has not proved by clear and convincing evidence that Garland in combination with any of Oracle's four other references would have rendered obvious to one of ordinary skill in the art any of the twelve asserted claims at the time of the respective inventions." (*Id.*, citing D.I. 273 at ¶ 73) Defendant also argues that plaintiffs' motion must be denied because plaintiffs failed to consider the objective indicia of obviousness pursuant to *Graham v. John Deere Co.*, 383 U.S. 1, 17-18 (1966) (hereinafter, "the *Graham* factors").

c. Discussion

Generally, the party opposing summary judgment "must point to an evidentiary conflict created on the record at least by a counter statement of a fact or facts set forth in detail in an affidavit by a knowledgeable [person]." *Barmag Barmer Maschinenfabrik AG v. Murata Mach., Ltd.*, 731 F.2d 831, 836 (Fed. Cir. 1984). *Compare Stoller v. Ford Motor Co.*, Nos. Civ. A. 89-1493, 89-1494, 1991 WL 5889, *2 (Fed. Cir. Jan. 25, 1991) (unpublished) ("Opposing a summary judgment motion based on anticipation by and obviousness over a reference, however, does not, for purposes of preserving a right of appeal, require more than that the opposing party assert, **with substantiation**, that the prior art references cited by the movant do not disclose or render obvious the invention of the patent. The court can determine what the prior art references disclose without the aid of extrinsic facts.") (emphasis added) (citation omitted). Finkel's conclusory statement of non-obviousness does not delineate any particular questions of fact to be tried.

As explained by the *KSR* Court, however,

> [i]n considering summary judgment on [obviousness] the district court can and should take into account expert testimony, which may resolve or keep open certain questions of fact. That is not the end of the issue, however. The ultimate judgment of obviousness is a legal determination. Where . . . the content of the prior art, the scope of the patent claim, and the level of ordinary skill in the art are not in material dispute, and the obviousness of the claim is apparent in light of these factors, summary judgment is appropriate.

127 S. Ct. at 1745-46. Pursuant to *KSR*, therefore, the court looks to the record to determine, despite the lack of sufficient rebuttal testimony by Finkel, whether the obviousness of the asserted claims is “apparent” based on other facts of record.

The parties do not dispute the level of ordinary skill in the art in their papers.[39] The only dispute regarding the content of the asserted obviousness references is whether Garland discloses “dispatching”; as stated previously, defendant did not substantiate its argument in this regard. On its face, Garland discloses the dispatching and load balancing concepts which are missing from the remaining references. Further, as noted by Shamos, Garland provides a motivation to combine by stating that its architecture may be utilized to solve high volumes of server traffic.

There appears to be no evidence of record, however, regarding whether a person of ordinary skill in the art, posessing Garland, either Oracle 1.0, Oracle 2.0, Popp, or Dienst, and a motivation to combine them, would have had a reasonable expectation of success. Plaintiffs do not detail how such cornbinations would be made and how they would operate, and these details are not readily gleaned from the

[39] Defendant implies that Shamos and Finkel have advanced different opinions on the level of ordinary skill in the art, but defendant cites only to a paragraph of Finkel’s second declaration stating that Garland does not disclose “dispatching.” (D.I. 281 at 38, citing D.I. 273 at ¶ 71)

references themselves. Shamos's declaration states only that "[s]uch a combination would result in the above prior art references having the dispatching capabilities of Garland, namely the use of dynamic load balancing techniques to dispatch Web requests from a Web server to one or more page servers." (D.I. 257 at ¶ 191)

In its reply papers, plaintiffs state that "[t]he Garland dispatcher's ability to monitor the load on the various member servers and select the least loaded server to process the request can easily be combined with Popp's CGI interface program (the dispatcher)." (D.I. 320 at 12, citing D.I. 257 at ¶¶ 191-91) The portion of Shamos's declaration cited, however, does not specifically mirror this conclusion, or provide any details regarding such a combination. Defendant points out that Garland discloses using redirection. (D.I. 214, ex. 42 at ORCL01818183, Fig. 5) Plaintiffs assert that this distinction is "irrelevant because Popp already discloses sending the request **directly** from the CGI interface program to the page server." (D.I. 320 at 12, citing D.I. 257 at ¶ 126) (emphasis in original) Presumably, plaintiffs argue that a person of ordinary skill in the art, having a motivation to combine these particular references, would favor this aspect of Popp over Garland; plaintiffs, however, point to no justification for this assertion.

The lack of record evidence regarding the *Graham* factors buttresses the court's decision to deny plaintiff's motion. Most frequently, evidence of objective indicia of non-obviousness is presented to rebut a prima facie showing on obviousness; counter-evidence on secondary considerations (objective indicia of obviousness) is not a required part of a prima facie case. *See KSR*, 127 S. Ct. at 1734 ("Such secondary considerations . . . might be utilized . . . ."). Obviousness, however, is ultimately a legal question grounded upon factual determinations, which include objective indicia of obviousness. *See Eisai Co. Ltd. v. Dr. Reddy's Laboratories, Ltd.*, 533 F.3d 1353, 1356 (Fed. Cir. 2008). A district court must make *Graham* findings prior to invalidating a patent for obviousness, regardless of whether couched in terms of "*Graham*" or otherwise. *See Ruiz v. A.B. Chance Co.,* 234 F.3d 654, 664 (Fed. Cir. 2000).[40] The court notes at this juncture that plaintiffs devoted less than two pages of its opening brief to obviousness. (D.I. 207 at 38-40) Plaintiffs could not have seriously expected to present "clear and convincing" evidence of obviousness in such a manner. The court is not in the position to make the type of detailed findings mandated by the Federal Circuit when patents, carrying presumptions of validity, are invalidated on summary judgment. Put another way, plaintiffs have not met their high burden on the record at bar.[41]

V. CONCLUSION

For the aforementioned reasons, the court grants plaintiffs' motion for summary

[40]The purpose of the requirement for express *Graham* findings is to avoid the use of hindsight and to present sufficient "express and necessarily implied findings" for appellate review. *See Ruiz*, 234 F.3d at 664 (citing *Loctite Corp. v. Ultraseal Ltd.*, 781 F.2d 861 (Fed. Cir. 1985), overruled on other grounds by *Nobelpharma AB v. Implant Innovations, Inc.*, 141 F.3d 1059 (Fed. Cir. 1998)).

The court is not convinced that software patents, grounded in source code, are so "technologically complex" so as to lessen the necessity of a detailed discussion regarding the *Graham* factors. *Id.*

[41]The court notes that the validity of the '554 and '335 patents has not been previously adjudicated. There is no indication that any of plaintiffs' asserted obviousness references, including Garland, were before the examiners – they do not appear on the face of either patent, and defendant has not asserted that plaintiffs' obviousness arguments were considered, but rejected, by the patent office. As noted previously, even had these same arguments been presented in the Texas litigation (and there is no argument that they were), they were not substantively addressed by that court. Both patents are currently under reexamination. The court does not find non-final actions on reexamination persuasive authority.

judgment of non-infringement (D.I. 204) and denies as moot plaintiffs' motion for summary judgment of no willful infringement (D.I. 208). Plaintiffs' motion to exclude defendant from asserting damages based on plaintiffs' foreign sales is denied as moot. (D.I. 212) Defendant's motion for partial summary judgment of literal infringement is denied. (D.I. 223) The court denies plaintiffs' motion for summary judgment of invalidity. (D.I. 206) The court grants-in-part and denies-in-part defendant's motion for summary judgment of no anticipation. (D.I. 216) An appropriate order shall issue.