IN THE UNITED STATES DISTRICT COURT

FOR THE DISTRICT OF DELAWARE

ORACLE CORPORATION and ORACLE AMERICA, INC.,	)	
Plaintiffs,	)	
v.	)	Civ. No. 06-414-SLR
****PARALLEL NETWORKS, LLC**,	)	
Defendant.	)	

Mary B. Graham, 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.

John W. Shaw, Esquire and Karen E. Keller, Esquire of Young Conaway Stargatt & Taylor, Wilmington, Delaware. Counsel for Defendant. Of Counsel: George S. Bosy, Esquire and David R. Bennett, Esquire of Bosy & Bennett, Chicago, Illinois; Patrick L. Patras, Esquire and Matthew J. Canna, Esquire of Hinshaw & Culbertson LLP, Chicago, Illinois; and Darryl J. Adams, Esquire and Kevin J. Meek, Esquire of Baker Botts LLP, Austin, Texas.

****AMENDED MEMORANDUM OPINION**

Dated: April 29, 2011
Wilmington, Delaware

ROBINSON, District Judge

I. INTRODUCTION

On June 30, 2006, Oracle Corporation and Oracle U.S.A. Inc. (collectively, "Oracle")[1] filed this action for declaratory judgment against Parallel Networks, LLC's ("Parallel"'s) predecessor-in-interest, EpicRealm Licensing, L.P. ("EpicRealm"), a patent licensing firm. (D.I. 1) EpicRealm assigned all right, title, and interest in the patents-in-suit to Parallel in August 2007. The court granted EpicRealm's motion to substitute parties on September 29, 2008. (D.I. 355) Oracle seeks judgment that it does not infringe, and that the patents-in-suit, U.S. Patent Nos. 5,894,554 ("the '554 patent") and 6,415,335 ("the '335 patent"), are invalid and/or unenforceable due to inequitable conduct. (D.I. 369) The '554 and '335 patents are directed to a system for creating and managing custom web sites.

On December 4, 2008, the court granted Oracle's motion for summary judgment of noninfringement on both the '554 and '335 patents on the ground that the accused products did not meet the "releasing" limitation of the claims. (D.I. 400) Alternate non-infringement arguments were not reached (i.e., "intercepting" and "dispatching" limitations and indirect infringement). The court also granted summary judgment of no anticipation as to several (but not all) asserted prior art references based on the dispatching limitation of the ****claims. The court denied Oracle's motion for summary judgment of obviousness.** (*Id.*) The Federal Circuit subsequently vacated the court's noninfringement ruling on the basis that a reasonable jury could find that the

---

[1]On April 19, 2011, the parties stipulated to substitute Oracle America, Inc. for Oracle U.S.A., Inc. and to amend the caption to reflect this substitution. (D.I. 450) The court continues to refer to plaintiffs collectively as "Oracle."

accused devices satisfy the "releasing" limitation based upon the court's construction, with which it did not take issue. The Court did not consider the alternate infringement arguments. *See Oracle Corp. v. Parallel Networks, LLC*, 325 Fed. Appx. 36, 40-41 (Fed. Cir. Apr. 28, 2010). On remand from the Federal Circuit, the court will address noninfringement based on the "intercepting" and "dispatching" limitations, indirect infringement, and literal infringement of claim 11 of the '554 patent.

II. BACKGROUND

A. The Parties and Litigation History

Oracle manufactures, sells and licenses software products for customers to use in conjunction with the delivery of dynamic web pages.[2] Parallel 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"). Oracle was 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 Oracle for indemnification. Parallel and Safelite settled the Texas litigation and filed a stipulation of dismissal with the court on June 26,

[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 U.S.A., Inc., a Colorado corporation, was a wholly owned subsidiary of Oracle Corporation. (*Id.*) On February 15, Oracle U.S.A., Inc. merged with and into Sun Microsystems, Inc. and was renamed Oracle America, Inc. (D.I. 450)

[3]*epicRealm Licensing, LP v. Franklin Covey Co. et al*, Civ. No. 2:05-CV-356; *epicRealm, Licensing, LP v. Speedera Networks, Inc.*, Civ. No. 2:05-CV-150; *epicRealm Licensing, LP v. Autoflex Leasing, Inc. et al*, Civ. No. 2:05-CV-163.

2006. (D.I. 282, ex. 22) A stipulation of dismissal was also filed with respect to Safelite's third party complaint against Oracle. On June 29, 2006, the court entered orders dismissing both complaints. (D.I. 1 at ¶¶ 26-27)

Oracle brought its declaratory judgment suit in this court on June 30, 2006. (*Id.*) In the complaint, Oracle alleges that, in a letter to Clark Consulting, Inc. (a party to the Texas litigation), Parallel stated that Clark was required to provide discovery regarding Clark's use of software proprietary to Oracle. (*Id.* at ¶ 24) Oracle also claims that Parallel demanded and received discovery from Safelite regarding its use of Oracle software. (*Id.* at ¶ 25) Parallel moved to transfer venue and consolidate with the Texas litigation. This court denied Parallel's motions on March 26, 2007. (D.I. 21)

Parallel 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) Parallel also brought a counterclaim of patent infringement. (*Id.*) Oracle amended its complaint on October 15, 2007, to add a claim that the '554 and '335 patents are unenforceable due to inequitable conduct. (D.I. 369) As discussed above, the court granted Oracle's motion for summary judgment of noninfringement based on the "releasing" limitation, and did not reach arguments regarding noninfringement based on the "intercepting" and "dispatching" limitations, nor on indirect infringement. (D.I. 400) The court denied Parallel's motion for partial summary judgment of infringement based on the "releasing" limitation. (*Id.*) The Federal Circuit vacated this court's ruling of noninfringement, holding that the accused products could be found to infringe, based on this court's

construction and a hardware scenario (freeing processor cycles) that had not been considered. (D.I. 422 at 9) Discovery has been closed for almost three years and trial is currently scheduled to commence May 9, 2011. (D.I. 176; D.I. 439)

B. Technological Background and the Patents-in-Suit

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 that may be 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 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" and, even though another page server may also be logged into the appropriate data source, it selects the server containing the cached

[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.

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)

Parallel asserts that Oracle infringes 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 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.

Claim 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

Parallel 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").[5] These products will be discussed in more detail *infra* in the context of the parties' infringement/noninfringement arguments.

[5]Parallel 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.

III. STANDARD OF REVIEW

A. Summary Judgment

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 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).

B. Infringement

"Determining whether a patent claim is infringed requires a two-step analysis: 'First, the claim must be properly construed to determine its scope and meaning. Second, the claim as properly construed must be compared to the accused device or process.'" *Nike Inc. v. Wolverine World Wide, Inc.*, 43 F.3d 644, 646 (Fed. Cir. 1994) (*quoting Carroll Touch, Inc. v. Electro Mechanical Sys.*, 15 F.3d 1573, 1576 (Fed. Cir. 1993)). To prove direct infringement, the plaintiff must establish, by a preponderance of the evidence, that one or more claims of the patent read on the accused device literally or under the doctrine of equivalents. *See Advanced Cardiovascular Sys., Inc. v. Scimed Life Sys., Inc.*, 261 F.3d 1329, 1336 (Fed. Cir. 2001). To establish literal 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 prove infringement by the doctrine of equivalents, a patentee must provide "particularized testimony and linking argument" as to the "insubstantiality of the differences" between the claimed invention and the accused product, or with respect to

the function/way/result test. *See Texas Instruments Inc. v. Cypress Semiconductor Corp.*, 90 F.3d 1558, 1567 (Fed. Cir. 1996).

To establish indirect infringement, a patent owner 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, a patent owner must show that an accused infringer "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, a patent owner must show that an accused infringer sells "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 the patent owner having first shown direct infringement. *Joy Technologies, Inc. v. Flakt, Inc.*, 6 F.3d 770, 774 (Fed. Cir. 1993).

IV. DISCUSSION

Parallel 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.[6] (D.I. 224 at 1) Oracle moves for summary judgment of

[6]Parallel fails to create a genuine issue of material fact as to whether the accused products infringe under the doctrine of equivalents. Parallel'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, Parallel'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

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)

A. The Accused Products

The parties do not dispute the physical characteristics of the accused products. (D.I. 204 at 1, ¶ 2; D.I. 224 at 1)

1. 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) Web Cache caches both static and dynamic web pages.

---

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, ex. 5 at ¶ 36) Parallel 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 Oracle'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. Parallel fails to show that Finkel's conclusion is similarly supported. In addition, the court notes that, even if Finkel's supplemental report had been timely filed, his doctrine of equivalents analysis is merely a restatement of Parallel's literal infringement arguments in the "function-way-result" pattern and, thus, 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 of 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).

(D.I. 217, ex. "Clark" at ¶ 52) In order to avoid sending outdated dynamic content to a Web client, Web Cache uses an algorithm to periodically flush dynamic content from the cache, forcing the next request for that dynamic content to be handled by the web server again. (*Id.*) If Web Cache has the requested content in its cache (a "cache hit"), Web Cache returns the requested content to the web client. (D.I. 270 at ¶ 28) 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 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*.) 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)[7] The Front End fiber then

[7]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*.)

compares the Universal Resource Locator ("URL")[8] 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, destroys the Front End fiber.[9] (*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*.)

2. Application Server products

The Application Server products contain multiple software programs, such as Oracle HTTP Server ("OHS") and Oracle Containers For Java ("OC4J"). (*Id*. at ¶ 75)

---

[8]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)

[9]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*.)

OHS is the web server component of the Application Server products. (*Id*.) OHS is based on the Apache Web server, a Web server software product developed by the open-source software community and distributed for free over the internet. (*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 HTTP Listener is based on an Apache HTTP listener. (*Id.*) The modules perform various functions related to the processing of web client requests. (*Id*. at ¶ 77) Many of the standard Apache modules, such as mod_cgi, mod_fastcgi, mod_perl, and mod_php are included as part of OHS. (Id. at ¶¶ 77, 82) These modules are developed and maintained by third-parties within the open-source community. (*Id.* at ¶ 82) OHS also includes several modules such as mod_oc4j and mod_plsql, developed by Oracle, that are specific to the Application Server products. (*Id.* at ¶ 77, 82)

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.[10] (*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*.)

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

---

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

3. 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.[11] (*Id*.)

The Database products do not store web pages. (*Id*. at ¶¶ 127-28) Data retrieved from a 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 a 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 a 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 calls that permit a software program to issue high-level commands through a library that

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

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*.) An RAC database instance can generate a web page. (*Id*. at ¶ 139)

B. Literal Infringement

Oracle presents three arguments against literal infringement based on the "releasing," "intercepting" and "dispatching" limitations shared by all asserted claims. The Federal Circuit, as noted above, has held that summary judgment of noninfringement cannot rest on consideration of the "releasing" limitation. Therefore,

this court will proceed to consider literal infringement only vis-a-vis the remaining two limitations.

1. "Intercepting"

In its memorandum order of December 4, 2008, the court construed the limitation "[i]ntercepting said request at said Web server"[12] to mean "[d]iverting the handling of said request before the request is processed by the Web server." (D.I. 399 at ¶ 2) The term "said request" is a reference to preceding claim language specifying a request for a dynamic web page.

Parallel presents two basic infringement paradigms through which it argues that the accused products meet the "intercepting" 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" ("Web Cache paradigm").[13] (D.I. 224 at 35) The other paradigm treats OHS as the "Web server" and either OC4J or a RAC database instance as the "page server" ("OHS paradigm"). (D.I. 224 at 20-21; D.I. 275 at 16-17) Parallel argues that, in both paradigms, the accused products meet the "intercepting" limitation.

---

[12]The court similarly construed this term when referring to an "HTTP-compliant device" and a "second computer system," as used in other asserted claims.

[13] As Dr. Finkel reported:

> The intercepting occurs when a determination is made that the "Web server" (for example, Oracle Web Cache) will not satisfy the request itself with pre-existing, cached, or static content, but rather, the request will be satisfied with dynamic content by a "page server" (for example, Oracle Application Server including Oracle HTTP Server).

(D.I. 275 at 21)

a. Web Cache paradigm

Where Web Cache is considered to be the "Web server" and OHS the "page server," Parallel argues that Web Cache diverts the request (for a dynamic web page) before the Web Cache executable can process the request because the request is intercepted and routed to OHS instead of being processed by Web Cache. (D.I. 224 at 35) Parallel further argues that Web Cache meets the "intercepting" limitation because "[Web Cache] processes some but not all requests." (D.I. 275 at 20)

Oracle argues that the asserted claims require that "the 'Web server' of the patents distinguish requests for dynamic content from other requests and 'process' or 'handle' them differently from, for example, requests for static content." (D.I. 205 at 27) In support of this argument, Oracle points to the deposition testimony of Parallel's expert on infringement, Dr. Finkel: "[I]ntercepting refers to intercepting dynamic Web page generation requests." (D.I. 214, ex. A23 at 138:14-139:12) Oracle argues that "[b]ecause Web Cache performs its 'processing' or 'handling' of each request in the same way regardless of the type of content requested - static or dynamic - it does not perform the 'intercepting' of the asserted claims . . . ." (*Id.* at 28) Oracle points to the file history of the '335 application, where "intercepting" was distinguished from "mere

routing" to overcome a rejection,[14] and further argues that "cache misses" are merely routed to an origin server (OHS in this instance). (*Id.* at 29-30)

In sum, Oracle argues that meeting the "intercepting" limitation depends on the criterion for diverting requests from being processed by the Web server, and that the criterion must be whether the request is for a static or dynamic web page. Parallel, in essence, argues that meeting the "intercepting" limitation is not dependent on the criterion used to divert requests, and that as long as some dynamic web page requests are diverted, the limitation is met.

b. OHS paradigm

Similar to its Web Cache argument, Parallel argues that, under the OHS paradigm, the limitation is met because OHS "can handle some requests on its own while other requests, such as dynamic Web page generation requests, are intercepted" and then routed to a "page server". (D.I. 224 at 21) Parallel further argues that the limitation is met because the "Web server" machine (OHS) diverts the request before the "Web server" executable can process the request. (*Id.*)

---

[14] Claims 17-45 stand rejected under U.S.C. § 102(e) as being anticipated by Leaf, U.S. Pat. No. 5,754,772 ("Leaf"). Applicants respectfully traverse this rejection for at least the following reasons. Claim 17 recites, in part, "intercepting said request at said Web server and routing said request to said page server". Leaf does not teach or suggest "intercepting said request". Instead, Leaf teaches that the web server routes the request directly to the transaction gateway client. Leaf, col. 4, lines 55-57. Leaf does not teach or suggest "intercepting said request at said Web server" because merely routing a request from a web server to the transaction gateway does not involve interception.

(D.I. 214, ex. A4 at EPIC000497-98)

Oracle argues that "OHS does not 'intercept' because it never diverts requests from their normal processing based on the request content. OHS processes all requests by invoking its modules, one at a time, and then using the appropriate module to process the request." (D.I. 205 at 30) (*citing* D.I. 217, ex. "Clark" at ¶¶ 130-33) Oracle asserts that OHS selects the module to handle the request based on the nature of the request, with Java applications handled by mod_oc4j and Perl applications handled by mod_perl. (*Id.*)

c. Conclusion

Oracle's noninfringement arguments directed to the "intercepting" limitation are drawn to its proposed claim construction. Oracle argues that intercepting requires diversion of dynamic Web page requests based solely on the distinction between the static and dynamic nature of the requests. This argument fails under the court's broader construction that requires only that the request be diverted before the request is processed by the Web server.[15]

As to literal infringement of claim 11 of the '554 patent, Web Cache uses an unspecified algorithm to purge the cache of dynamic content that is outdated. Web Cache creates a Front End fiber to process each request, static or dynamic. The Front End fiber then searches the cache for the requested content. There is a genuine issue of material fact as to the algorithm used to purge the cache, which ultimately determines whether a request will be diverted. A reasonable jury could find that

[15]The court notes that broader construction of claims, while tending to encompass more accused products, also increases the range of prior art which may be used to prove invalidity.

creating Front End fibers, searching the cache or purging of dynamic content constitutes processing of the request before it is diverted to an origin server. Therefore, the court denies Parallel's motion for partial summary judgment of infringement of claim 11 of the '554 patent as to the Web Cache paradigm.

Under the OHS paradigm, there is a genuine issue of material fact pertaining to the order in which modules are loaded and, therefore, the order of their execution. OHS includes standard Apache modules such as mod_CGI. Each module is executed, in turn, until one handles the request, if possible. The order in which modules are loaded into memory determines the order in which they are executed. If mod_CGI is executed before another module that would divert the request, it may process the request before it is diverted. This is precisely the condition that the invention attempts to avoid. Further, a reasonable jury could find that execution of each module, in turn, constitutes processing by OHS before diverting a request to a "page server." Therefore, the court denies Parallel's motion for partial summary judgment of claim 11 of the '554 patent as to the OHS paradigm.

2. "Dispatching"

In its memorandum order of December 4, 2008, the court construed the limitation "[d]ispatching said request to said page server" to mean "analyzing a request to make an informed selection of 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." Again, the term "request" refers to a dynamic web page request. Parallel presents the same two infringement paradigms for the "dispatching" limitation as for "intercepting."

a. Web Cache paradigm

Parallel asserts that Web Cache routes requests to the OHS server with the greatest weighted available capacity and argues, therefore, that "Web Cache is a software program that determines which 'page server' should be used to process the dynamic Web page generation request." Parallel presents two theories under which Web Cache "examines or analyzes a request." First , Parallel asserts that Web Cache examines a request to determine if it is a stateful or a stateless request.[16] (D.I. 224 at 36) (*citing* D.I. 225, ex. 11 at ORCL00513924-25) While stateful requests are routed to the same OHS server that originally handled the request, stateless requests are distributed based on the weighted available capacity of the OHS server. (*Id.*) (*citing* D.I. 225, ex. 11 at ORCL00513921) Second, Parallel asserts that Web Cache examines a request to determine which OHS servers can process the request based on the URL

---

[16] OracleAS Web Cache supports applications that use a session ID or session cookie to bind user sessions to a given origin server in order to maintain state for a period of time. To utilize the session binding feature, the origin server itself must maintain state, that is, it must be stateful. An application binds user sessions by including session data in the HTTP header or body it sends to a client in such a way that the client is forced to include it with its next request. This data is transferred between the origin server and the client through OracleAS Web Cache either with an embedded URL parameter or through a cookie, which is a text string that is sent to and stored on the client. OracleAS Web Cache does not process the value of the parameter or cookie, it simply passes the information back and forth between the origin server and the client.

(D.I. 225, ex. 11 at ORCL00513924) To configure Web Cache to use session binding, the name of the session parameter or session cookie must be specified. (*Id.* at ORCL00513925)

and a configuration file, and then load balances among those found to be eligible. (*Id.* at 32, 36)

Oracle argues that Web Cache lacks a "dispatcher" and, therefore, cannot dispatch. (D.I. 205 at 31-32) According to Oracle, the Web Cache executable itself determines whether to fetch the page from the cache ("cache hit") or route the request to an OHS server ("cache miss"), and the plain language of the claims requires that the dispatcher be separate from the "Web server" since requests are routed to the dispatcher. Further, Oracle argues that Web Cache routes cache misses directly to an OHS server, the "page server" in this paradigm, and not to a dispatcher. (*Id.* at 32)

b. OHS paradigm

Under the OHS paradigm, Parallel argues that OHS (as "Web server") routes requests to mod_oc4j (the "dispatcher") which further routes the request, using metric-based load balancing, to an OC4J instance (the "page server"). (D.I. 224 at 21-22) Parallel asserts that "mod_oc4j examines the request and then looks in a configuration file to determine which 'page servers' (OC4Js) are capable of processing this type of dynamic Web page request;" mod_oc4j can then use one of two metric-based load balancing algorithms in determining which page server should receive the request. (*Id.* at 22)

Oracle counters, in a manner similar to the Web Cache argument, that mod_oc4j is a part of the OHS executable itself[17] and, therefore, cannot be a "dispatcher." (D.I.

[17]Oracle points out that mod_oc4j is a configurable option, and when customers elect to use its functionality, it is present only as part of the OHS executable. (D.I. 205 at 33)

205 at 33) Again Oracle argues that as part of the executable, OHS cannot route requests to mod_oc4j, and further that OHS, using mod_oc4j, routes requests to an OC4J process, not to a "dispatcher." (*Id.*) Oracle's arguments regarding the second version of the OHS paradigm, where JDBC and OCI act as the "dispatcher," follow similar lines; these APIs are a part of the OHS executable and, therefore, cannot represent a "dispatcher." (*Id.*) Oracle further argues that "because neither JDBC nor OCI 'examine' or 'analyze' a Web request," they do not "dispatch." (*Id.* at 34) Parallel responds that OCI examines session tag and security attribute parameters of the request before using runtime load balancing to select the session that can best serve the request." (D.I. 275 at 19)

c. Conclusion

Oracle's arguments are again framed in the context of its proposed construction. Oracle argues that dispatching must be performed by a software executable separate from the "Web server." In the case of the Web Cache paradigm, Oracle argues that there is no "dispatcher" separate from the Web Cache executable. As to the OHS paradigm, Oracle argues that the mod_oc4j module is a part of the OHS executable. The court's construction of the "dispatching" limitation, however, does not require a separate executable. Therefore, Oracle's motion for summary judgment in this regard is denied.

C. Indirect Infringement

Oracle seeks summary judgment that it does not indirectly infringe the patents-in-suit, either by inducement or contributorily. (D.I. 204; D.I. 205 at 35) Either form of

indirect infringement requires a predicate showing of direct infringement.[18] *Joy Technologies,* 6 F.3d at 774. Oracle argues that, even if direct infringement is possible, Parallel has not produced any evidence that Oracle's customers use the accused products in an infringing configuration. (D.I. 205 at 36) Oracle asserts that the default configuration of the accused products does not practice the dynamic load balancing requirement of the "dispatching" limitation, and neither of Parallel's experts investigated whether Oracle's customers actually implemented this feature. (*Id.*) Oracle further argues, as to contributory infringement, that each of its accused products have significant noninfringing uses; Parallel, having the burden to prove otherwise, has failed to do so. (*Id.*)

Parallel argues that it has provided sufficient circumstantial evidence of direct infringement by Oracle customers, showing that Oracle instructs its customers "how to use the Accused Oracle Products in an infringing manner" and that Oracle "freely and openly strongly encourages them to do so." (D.I. 275 at 29) As an example, Parallel points to a presentation at "Oracle World," a conference attended by "tens of thousands" of Oracle customers and potential customers. (*Id.*) Parallel asserts that "Oracle explained [at Oracle World] that the infringing metric-based load balancing was 'the best way not to fall,' i.e., the best way to prevent system crashes." (*Id.* at 29-30) Parallel further points to a "2004 'Best Practices' document, [wherein] Oracle stated that Oracle Web Cache 'customers should use the built-in load balancing functionality in

[18]Both forms of indirect infringement also require knowledge on the part of the accused infringer. The parties do not dispute this issue. Oracle argues that the timing of its notice supporting the knowledge requirement affects the period of any alleged infringement. (D.I. 205 at 37) The court does not reach this argument here.

OracleAS Web Cache to distribute cache miss traffic over the application Web server farm.'" (*Id.* at 30) Parallel also asserts that it has provided direct evidence in the form of deposition testimony and emails that admit to certain customers' use of metric-based load balancing configurations. (*Id.* at 32) The court finds that Parallel has proffered sufficient circumstantial and direct evidence to withstand summary judgment as to this point, thereby resolving the issue of induced infringement.

Contributory infringement may not be found, however, if the accused product is either not "a material part of the invention," or has a "substantial noninfringing use." *Golden Blount,* 365 F.3d at 1061. Referring to 35 U.S.C. § 271(c), Parallel argues that "[t]he accused Oracle products, specialized computer software, are not the type of staple articles of commerce intended to fit [this] narrow exception". (*Id.* at 33) "The word 'staple' means 'a commodity that is produced regularly or in large quantities [especially] for a wholesale market.'" (*Id.*) (*citing Bliss & Laughlin Indus., Inc. v. Bil-Jax, Inc.*, 356 F. Supp. 577, 580 (D. Ohio 1972)) A reasonable jury could find that the accused products do not meet the definition of a staple article of commerce. Therefore, the court denies Oracle's motion for summary judgment of noninfringement as to contributory infringement.

V. CONCLUSION

For the aforementioned reasons, the court denies Oracle's motion for summary judgment of noninfringement (D.I. 204). Parallel's motion for partial summary judgment of literal infringement (D.I. 223) is denied. An appropriate order will issue.