IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

ORACLE CORPORATION and ORACLE U.S.A. INC., Plaintiffs/Counterclaim Defendants, v. EPICREALM LICENSING, LP, Defendant/Counterclaim Plaintiff.	C.A. No. 06-414-SLR **JURY TRIAL DEMANDED** **PUBLIC VERSION**

EPICREALM'S MEMORANDUM IN SUPPORT OF ITS MOTION FOR PARTIAL SUMMARY JUDGMENT OF LITERAL INFRINGEMENT

OF COUNSEL:

Harry J. Roper
George S. Bosy
Aaron A. Barlow
Patrick L. Patras
David R. Bennett
Paul D. Margolis
Benjamin J. Bradford
Emily C. Johnson
JENNER & BLOCK
330 N. Wabash Avenue
Chicago, IL 60611-7603
Tel: (312) 923-8305

Richard L. Horwitz (#2246)
David E. Moore (#3983)
POTTER ANDERSON & CORROON LLP
Hercules Plaza, 6th Floor
1313 N. Market Street
Wilmington, DE 19899
Tel: (302) 984-6000
rhorwitz@potteranderson.com
dmoore@potteranderson.com

Attorneys for Defendant/Counterclaim Plaintiff epicRealm Licensing, LP

Dated: July 31, 2008
876827 /31393 / Oracle

Public Version Dated: August 7, 2008

TABLE OF CONTENTS

I. NATURE AND STAGE OF THE PROCEEDINGS 1

II. SUMMARY OF ARGUMENT 1

III. CONCISE STATEMENT OF FACTS 1

A. Background Of The Technology 2

B. The '554 And '335 Patents-In-Suit 4

C. Relevant Oracle Products 8

1. Oracle Adds Metric-Based Load Balancing To The Oracle Application Server Products Pursuant To Customer Demands. 9

2. Oracle Adds Metric-Based Load Balancing To The Oracle Database Products. 11

3. Oracle Adds Metric-Based Load Balancing To The Oracle Web Cache Products 13

4. Oracle's Distribution Of The Accused Oracle Products 14

5. Oracle Touts The Advantages Of Its Metric-Based Load Balancing In The Accused Oracle Products 14

IV. ARGUMENT 15

A. Applicable Legal Principles 15

1. Summary Judgment 15

2. Infringement 16

B. EpicRealm Should Be Granted Summary Judgment That Oracle Literally Infringes Claim 11 Of The '554 Patent With Respect To The Accused Oracle Products. 16

1. The Accused Oracle Products Meet the Preamble Of Claim 11 Of The '554 Patent. 17

2. The Accused Oracle Application Server Products Literally Infringe Claim 11 Of The '554 Patent. 17

a) "routing a dynamic Web page generation request from a Web server to a page server, said page server receiving said request" 18

b) "and releasing said Web server to process other requests"....19

c) "wherein said routing step further includes the steps of intercepting said request at said Web server," 20

d) "routing said request from said Web server to a dispatcher, and dispatching said request to said page server;"....21

e) "processing said request, said processing being performed by said page server while said Web server concurrently processes said other requests; and"....22

f) "dynamically generating a Web page, said Web page including data retrieved from one or more data sources." 23

3. The Accused Oracle Database Products Literally Infringe Claim 11 Of The '554 Patent....23

a) "routing a dynamic Web page generation request from a Web server to a page server, said page server receiving said request" 25

b) "and releasing said Web server to process other requests"....26

c) "wherein said routing step further includes the steps of intercepting said request at said Web server," 27

d) "routing said request from said Web server to a dispatcher, and dispatching said request to said page server;"....27

e) "processing said request, said processing being performed by said page server while said Web server concurrently processes said other requests; and"....30

f) "dynamically generating a Web page, said Web page including data retrieved from one or more data sources." 31

4. The Accused Oracle Web Cache Products Literally Infringe Claim 11 Of The '554 Patent....31

a) "routing a dynamic Web page generation request from a Web server to a page server, said page server receiving said request" 34

b) "and releasing said Web server to process other requests".....34

c) "wherein said routing step further includes the steps of intercepting said request at said Web server,".....35

d) "routing said request from said Web server to a dispatcher, and dispatching said request to said page server;".....35

e) "processing said request, said processing being performed by said page server while said Web server concurrently processes said other requests; and".....37

f) "dynamically generating a Web page, said Web page including data retrieved from one or more data sources.".....37

V. CONCLUSION.....38

TABLE OF AUTHORITIES

Cases

Anderson v. Liberty Lobby, Inc.,
477 U.S. 242 (1986) 15, 16

Callaway Golf Co. v. Acushnet Co.,
523 F. Supp. 2d 388 (D. Del. 2007) 16

Cybor Corp. v. FAS Techs., Inc.,
138 F.3d 1448 (Fed. Cir. 1998) 16

Horowitz v. Fed. Kemper Life Assurance Co.,
57 F.3d 300 (3d Cir. 1995) 15

Markman v. Westview Instruments, Inc.,
52 F.3d 967 (Fed. Cir. 1995) 16

Matsushita Elec. Indus. Co., Ltd. v. Zenith Radio Corp.,
475 U.S. 574 (1986) 15

Southwall Techs., Inc. v. Cardinal IG Co.,
54 F.3d 1570 (Fed. Cir. 1995) 16

Rules

Fed. R. Civ. P. 56(c) 15

Defendant epicRealm Licensing, LP ("epicRealm") moves this Court for partial summary judgment that Oracle literally infringes Claim 11 of epicRealm's '554 Patent with respect to the accused Oracle products.

I. NATURE AND STAGE OF THE PROCEEDINGS

This patent action was filed in 2006 and is now scheduled for a two-week jury trial commencing January 12, 2009. Pursuant to the Stipulated Amendment to Joint Discovery Plan and Scheduling Order (D.I. 191) entered by the Court on July 18, 2008, the deadline for filing summary judgment opening briefs is July 31, 2008.

II. SUMMARY OF ARGUMENT

EpicRealm should be granted summary judgment that Oracle literally infringes Claim 11 of the '554 Patent with respect to the Accused Oracle Web Cache Products, the Accused Oracle Application Server Products and the Accused Oracle Database Products (as defined herein). The operation of the Accused Oracle Products is undisputed. Applying either party's claim construction leaves no doubt that the Accused Oracle Products literally infringe Claim 11 of the '554 Patent as a matter of law.

III. CONCISE STATEMENT OF FACTS

This case involves two epicRealm patents: United States Patent Nos. 5,894,554 ("the '554 Patent") (Exh. 1) (see appendix filed herewith) and 6,415,335 ("the '335 Patent") (Exh. 2). The '335 Patent is a continuation of the '554 Patent. EpicRealm 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 claims"). The present motion is limited to Oracle's literal infringement of Claim 11 of the '554 Patent.

EpicRealm has accused Oracle of literally infringing Claim 11 of the '554 Patent with respect to the following products: (1) the Oracle Web Cache Products beginning in November

2000 with Release 1.0.2 and all subsequent releases (the "Accused Oracle Web Cache Products"); (2) the Oracle Application Server Products beginning in April 2003 with Release 10gR1 (9.0.4) and all subsequent releases (the "Accused Oracle Application Server Products"); and (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 "Accused Oracle Database Products") (all three collectively, the "Accused Oracle Products").

A. Background Of The Technology

The patented invention can best be understood in the context of the prior art and the prior art problems that the patents solved. This prior art included the then existing World Wide Web (the "Web") that was used by a "Web client" (*e.g.*, a computer running a Web browser) desiring to access information on a Web site through the retrieval of Web pages. (Exh. 1, '554 Patent at col. 1, ll. 14-17.) In this system, the "Web client" used a "Web browser" (*e.g.*, Microsoft Internet Explorer or other software capable of requesting and displaying Web content) to identify the Web site and the requested information; the Web browser connected to the Web, allowing access to the Web site. (*Id.* at col. 1, ll. 22-37.) The Web server operating the Web site received the request and generated a response which was returned to the Web browser. As explained in the patents-in-suit, the prior art used a particular software language to facilitate this process called "HTML" (HyperText Markup Language). The prior art also used a protocol for certain Web communications called "HTTP" (Hypertext Transport Protocol).

Initially, most Web sites provided only "static" Web pages. Typically, static Web pages are files stored on the Web server that remain the same until they are manually modified. (Declaration of Dr. David Finkel ("Finkel Decl."), ¶4 filed herewith.) When a static Web page is

requested, the Web server retrieves the specific file requested by the Web client and returns that file to the Web client without modifying the file. (*Id.*)

As the Web developed, Web sites began to provide dynamic Web pages, *i.e.*, Web pages that are generated only after they are requested in response to the specific request of the Web client. (Finkel Decl. ¶5.) In order to generate dynamic Web pages, the Common Gateway Interface ("CGI") was developed to facilitate the generation of dynamic Web pages by a Web server, as were various "tools" that facilitated the use of CGI. (Exh. 1, '554 Patent at col. 1, ll. 47-61.) The processing of dynamic Web pages is more resource intensive than is the case with static Web pages, *e.g.*, requiring more processor time, memory and/or other resources. (Exh. 3, Shamos Rpt. ¶29.) As the number of dynamic Web page requests increased, many issues arose based on this increased demand for Web server resources, including slower response time, the failure to provide the requested Web content, or even the complete failure ("crashing") of the Web server. (Finkel Decl. ¶5.)

The patents-in-suit summarize some of the problems with the management of dynamic Web page requests by the prior art, including CGI:

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

(Exh. 1, '554 Patent at col. 2, ll. 1-12.)

REDACTED

REDACTED

The prior art attempted to solve these problems by adding hardware (*i.e.*, computers or routers) to a Web site to handle the increase in requests for dynamic Web pages. (Finkel Decl. ¶5.) This approach was costly and cumbersome. (Finkel Decl. ¶5.) As explained below, the patents-in-suit solved the problem by minimizing the need for adding costly and cumbersome hardware.

B. The '554 And '335 Patents-In-Suit

The patents-in-suit generally relate to systems for efficiently managing dynamic Web page generation requests. The patents-in-suit solve the aforementioned problems associated with the management of dynamic Web page requests in a very different way from the prior art attempts to solve those problems (*e.g.*, CGI and related development tools). An example of the "architecture" of an embodiment of the patented invention is shown in Figure 4 of the patents-in-suit:

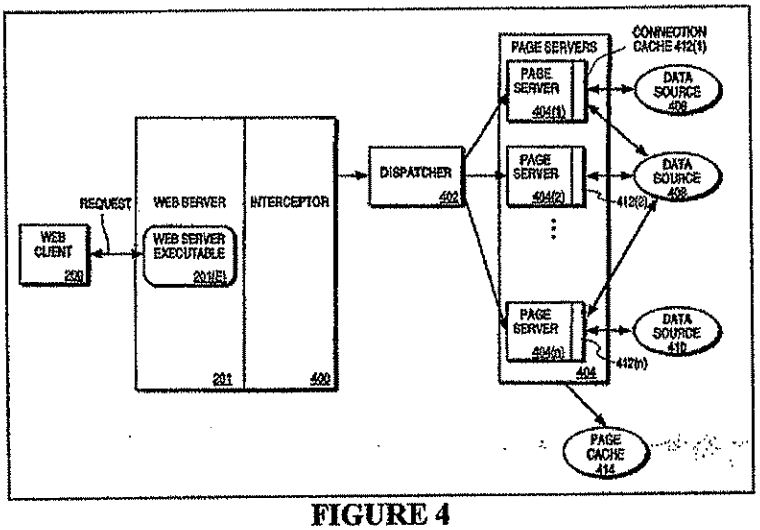


FIGURE 4

(Exh. 1, '554 Patent at Fig. 4.) In the above architecture, starting from the left of the diagram, the Web client (200), which again may simply be a computer running Microsoft Internet Explorer, initiates the request for a static or dynamic Web page, and the request is sent over the Web to the Web server (201) for processing. (*Id.* at col. 4, ll. 55-57.) The Web server may itself process the request and send a Web page back to the Web client. Not all requests are handled by the Web server, according to the patent. For a request that will not be processed at the Web server, "Interceptor 400 intercepts the request and routes it to the Dispatcher 402." (*Id.* at col. 4, ll. 58-60; *see also* col. 5, ll. 37-39.) For example, if the request is for a static Web page, *i.e.*, one that is relatively easy to locate and retrieve, the request may be processed by the Web server itself—in other words, the static page is located by the Web server and then sent back to the Web client. If, however, the request is for a dynamic Web page, the Web page to be sent back may be created by one of a number of "page servers" (404) shown to the right in the figure.

In the above illustration, three page servers are shown. However, there could be fewer or more than three page servers depending on the configuration of the Web site. (*See id.* at col. 5, ll. 37-39.) When the request is received at the Web server, the Web server initially has the duty to process the request. However, when a request is intercepted and routed from the Web server

to a page server, that page server assumes the processing duties for that request. In this way, the Web server is "released" to perform other tasks. (*See id.* at col. 5, ll. 8-19, col. 6, ll. 20-24.)

Coming back to Figure 4 above, once a request is intercepted, it is sent to the dispatcher (402). The function of the dispatcher is to select a page server that can more efficiently process the request. First, the dispatcher examines the request to determine the subset of page servers that are capable of processing this type of request. (*See, e.g., id.* at col. 6, ll. 12-13.) Second, the dispatcher makes an informed selection as to which page server in the subset ***can more efficiently process*** the request based on dynamic information maintained about the page servers. (*See, e.g., id.* at col. 6, ll. 14-19.) In that way, the dispatcher can take advantage of the fact that one page server in the subset of capable page servers can more efficiently process the request than the other page servers that can process it. (*Id.* at col. 5, l. 50-col. 6, l. 19.) In this example, the dispatcher greatly increases the efficiency of processing Web page requests. As disclosed in the patents-in-suit, this type of "load balancing" of dynamic Web page requests can significantly increase the performance at a busy Web site. (Exh. 1, '554 Patent, col. 6, ll. 16-19.)

As just noted, the selection made by the dispatcher as to which page server can more efficiently process the request is based on dynamic information maintained about page servers. (*Id.* at col. 5, ll. 50-55.) In the context of the patented invention, dynamic information about the page servers has several qualities. It is important to recognize that dynamic information is information that can only be established during the execution of a program. (Finkel Decl. ¶6.) In other words, dynamic information is not information that can be established prior to the execution of the program. (*Id.*) In addition, dynamic information in the context of the patent must be information maintained about page servers. (Exh. 1, '554 Patent at col. 5, ll. 50-55.)

This dynamic information maintained about page servers must indicate which page server can more efficiently process the request. (*Id.* at col. 5, l. 50-col. 6, l. 19.)

Three examples given in the patents-in-suit illustrate the type of dynamic information that allows the dispatcher to determine which page server can more efficiently process the request. As a first example, the dynamic information may identify a page server that has the least processing load (*e.g.*, the number of requests that the page server is servicing) as compared to other page servers, a page server that can process the request more quickly than the other page servers that can process it. (*Id.* at col. 6, ll. 14-19.) This is a type of information that cannot be known until the program is executing because the number of requests being serviced cannot be determined until the page server software is running. The patents-in-suit specifically describe this type of load balancing capability as one that can "significantly increase performance at a busy Web site." (*Id.* at col. 6, ll. 16-19.)

In a second example in the patent, the dynamic information may identify a particular page server that is already logged into a data source, a page server that can access the requested data more quickly than a page server that would need to log into the data source. (*Id.* at col. 6, ll. 5-13.) Again, this is the type of information that cannot be known until the software is running because the page server cannot log into a data source until the page server software is running.

As a final example, the dynamic information may identify a page server that has already retrieved the requested content in a previous request, a page server that therefore does not need to spend time to obtain the requested content from the data source. (*Id.* at col. 6, ll. 1-3.) Yet again, this type of information cannot be known until the page server software is running.

The summary of the invention set forth in the patents-in-suit explains how the patented invention functions:

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

(Exh. 1, '554 Patent at col. 2, ll. 20-31.)

The patented invention provides many advantages for a Web site host. The patented invention decreases the time necessary to process requests by allowing for dispatching and the subsequent routing to a page server as discussed above. (*Id.* at col. 6, ll. 14-19.) In addition, the patented invention improves performance because it facilitates the scaling up of the Web site to handle more requests as the Web traffic grows. (*Id.* at col. 8, ll. 11-24.) "Scalability" refers to the ability to increase the processing capability of a Web site, *e.g.*, the ability to add page servers as needed. (*Id.*; Finkel Decl. ¶7.)

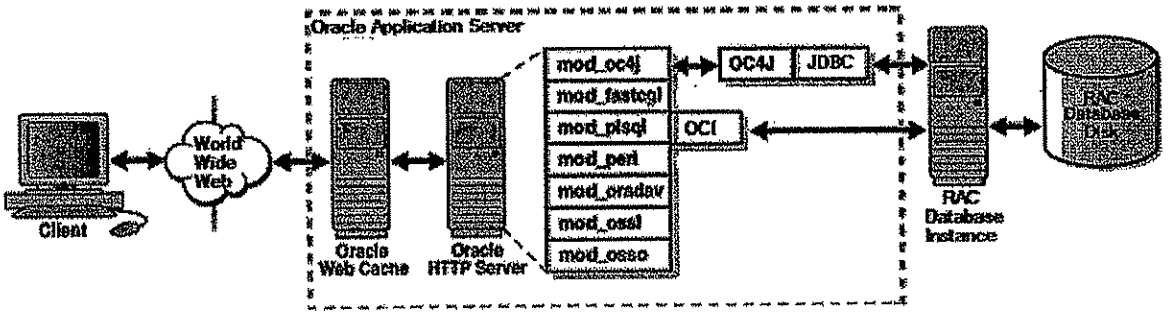
EpicRealm filed the application for what became the '554 Patent on April 23, 1996. On January 19, 1999, epicRealm filed the patent application leading to the '335 Patent. The '554 Patent issued on April 13, 1999, making public its disclosure of dispatching a dynamic Web page request using dynamic information maintained about page servers that indicates which page server can more efficiently process the request. After that public disclosure, Oracle, for the first time, began using such dispatching (which it sometimes refers to as "metric-based load balancing") for dynamic Web page requests in all three of the Accused Oracle Products.

C. Relevant Oracle Products

Over the years, Oracle has made many attempts at developing products for generating dynamic Web pages. Oracle aborted many such attempts, prior to invoking the technology of the

patents-in-suit. Oracle's aborted attempts are the allegedly anticipatory Oracle References that are the subject of epicRealm's Motion for Partial Summary Judgment That The Oracle References Do Not Anticipate, filed contemporaneously herewith.

Starting in 2000, just over a year after the issuance of the '554 Patent, Oracle added metric-based load balancing of dynamic web page requests to its products. Below is a schematic diagram (modified from an Oracle document) depicting the major components relevant to Oracle's infringement. This diagram provides an overview of how the Accused Oracle Web Cache Products, the Accused Oracle Application Server Products and the Accused Database Products can be used together. For ease of illustration, only one of each component is shown, even though there may be several of certain components in a typical installation. As shown, Oracle Web Cache is now part of Oracle Application Server. The parties, however, have referred to epicRealm's three infringement accusations as relating to: (1) Application Server; (2) Web Cache; and (3) Database. EpicRealm maintains that convention herein.



1. Oracle Adds Metric-Based Load Balancing To The Oracle Application Server Products Pursuant To Customer Demands.

Oracle Application Server ("OAS") is a software program that can be used to operate a Web site. OAS includes Oracle HTTP Server ("OHS"), which is a Web server based on the open source Apache HTTP Server, version 1.3.28. (Exh. 7, Clark Rpt. 28.) Oracle Application Server also includes various "mods" (short for modules). These mods perform various functions

related to the processing of Web page requests and allow OHS to integrate with other components of Oracle Application Server. (*Id.*) For example, mod_perl enables a Perl Application to respond to a request. (*Id.* at 28-29.) Most of the mods included in Oracle Application Server are from Apache, although Oracle has added some of its own mods including mod_oc4j (discussed below) and mod_plsql. Mod_plsql is used to respond to certain requests that will be processed by Oracle Database. (Exh. 8 at 1-6.)

REDACTED Oracle introduced a container for Java called Oracle Container For J2EE ("OC4J"). (Exh. 30, Nelson Dep. 68:17-69:3.) When it did so, it also instituted a new Oracle module called mod_oc4j to route requests to an OC4J. (*Id.*) OC4J is a container that includes and runs Java servlets, Java Server Pages ("JSPs") and Enterprise Java Beans ("EJBs"), all of which are used to generate dynamic Web pages. (Exh. 9 at ORCL00338431.)

REDACTED

Metric-based load balancing allowed mod_oc4j to route dynamic Web page requests by considering dynamic information about the OC4Js that were capable of processing the request. Thus, for example, mod_oc4j now could take into account the fact that one OC4J was very busy and likely to be slower to respond. In that case, mod_oc4j would be less likely to route the request to the very busy OC4J. REDACTED

REDACTED

2. Oracle Adds Metric-Based Load Balancing To The Oracle Database Products.

Oracle also added metric-based load balancing to its database products. Oracle Database is a complex set of software programs to control the organization, storage, management, and retrieval of data. (Exh. 7, Clark Rpt. 46.) The primary purpose of Oracle Database is to service client requests (or "database queries"). (*Id.*) Oracle Database responds to a client request by retrieving data from its database and returning it the client. (*Id.*) In addition to the standard relational database capabilities, Oracle Database includes the ability to receive, process and respond to dynamic Web page requests. (*See, e.g.*, Exh. 8 at 1-6.) Specifically, Oracle Database can receive a dynamic Web page request and return a dynamic Web page through its XML DB functionality, PL/SQL Web Application functionality, and PL/SQL Server Page functionality. (*Id.*; Exh. 12 at 1-17, 3-70 to 3-71.)

Oracle Real Application Cluster ("RAC") is a database configuration that allows a single database to be accessed by multiple database instances on multiple computers or "nodes." (Exh. 7, Clark Rpt. 49.) A RAC instance is the processes running on a computer that can access and modify the data. (Exh. 13 at 1-1.) A database node is a computer that is running one or more database instances. (*Id.* at 1-1 to 1-2.) The data for RAC is stored on a common pool of disks that can be accessed by the RAC instances. (Exh. 7, Clark Rpt. 49.) As an example, when a request for data, *e.g.*, a dynamic Web page request, is sent to a RAC database, the request is assigned to a particular RAC instance that processes the request, retrieves any necessary data from the common pool of disks, and then returns a response, *e.g.*, a dynamic Web page.

Recently, Oracle added metric-based load balancing to the RAC database via a new feature that Oracle calls runtime load balancing.

REDACTED

A work request can be a dynamic Web page request. (Finkel Decl. ¶12.) Thus, runtime load balancing can direct dynamic Web page requests to the RAC instances that can more efficiently process the requests.

To implement runtime load balancing, RAC collects information, including workload information, about the RAC instances and then posts notifications advising how work requests should be distributed among the RAC instances. (Exh. 13 at 4-11 to 4-12.) The advice provided by RAC on how to distribute work requests is the Load Balancing Advisory, and the advice is posted through "Fast Application Notification" ("FAN") events. (Exh. 7, Clark Rpt. 50.) As an example, a RAC database may calculate that work requests should now be distributed as follows: 30% to RAC instance #1, 20% to RAC instance #2, and 50% to RAC instance #3. In runtime load balancing, OCI and JDBC distribute the work requests based on the Load Balancing Advisory posted in the FAN events.

3. Oracle Adds Metric-Based Load Balancing To The Oracle Web Cache Products.

Oracle Web Cache is a software program designed to reduce the processing resources consumed by a Web site by saving copies of Web pages in memory. (Exh. 7, Clark Rpt. 9-10.) To respond to increased demand on a Web site, Oracle Web Cache can be installed in front of the Oracle HTTP Server. (*Id.* at 9.) When Oracle Web Cache is installed, a Web page request is first received by Oracle Web Cache, rather than Oracle HTTP Server.

Oracle Web Cache reduces the burden placed on a Web site by maintaining a cache, or a local store, of frequently requested Web page content, including entire Web pages, in its memory. (*Id.*) When Oracle Web Cache receives a Web page request, it checks its cache to determine whether it has a copy of the requested Web page in memory. (*Id.* at 10.) If Oracle Web Cache has the requested Web page in memory (a "cache hit"), it returns the Web page to the Web client. (*Id.*) If Oracle Web Cache does not have the requested Web page in memory (a "cache miss"), the request is forwarded on to an "application server" (*e.g.*, Oracle Application Server) for processing. (*Id.*) After the application server generates a response to the request, the newly created content is then stored in the Oracle Web Cache memory and used to respond to future requests.

REDACTED Oracle added metric-based load balancing of dynamic Web page requests to its Web Cache by routing "cache misses" using what Oracle called "weighted available capacity." Oracle defines "weighted available capacity" as:

(Capacity - Load) / Capacity

where:

- Capacity is the maximum number of concurrent connections that the [application server] can accept
- Load is the number of connections currently in use

(Exh. 11 at ORCL00513921; Exh. 30, Nelson Dep. 60:9-61:15.) The "weighted available capacity" is dynamic information maintained about the application servers. (Finkel Decl. ¶11.) The "weighted available capacity" is dynamic information because it cannot be determined until runtime. (*Id.*) When the "weighted available capacity" for each application server is different, Oracle Web Cache sends the request to the application server with the most weighted available capacity. (Exh. 11 at ORCL00513921; Exh. 30, Nelson Dep. 62:1-64:15.)

4. Oracle's Distribution Of The Accused Oracle Products

REDACTED

5. Oracle Touts The Advantages Of Its Metric-Based Load Balancing In The Accused Oracle Products.

REDACTED

At Oracle Open World, Oracle presented its metric-based load balancing, telling its customers that using metric-based load balancing was "the best way not to fall," *i.e.*,

the best way to prevent a Web site from crashing. (Exh. 18 at ORCL01050248-64; Exh. 31, Surber Dep. 98:16-103:10.) The presentation highlighted the metric-based load balancing in the Accused Oracle Application Server Products, the Accused Oracle Database Products and the Accused Oracle Web Cache Products, even stating that metric-based load balancing used in the Accused Oracle Application Server Products and in the Accused Oracle Database Products had the highest possible level of criticality. (*Id.*)

IV. ARGUMENT

EpicRealm should be granted partial summary judgment that Oracle has literally infringed Claim 11 of the '554 Patent with respect to the Accused Oracle Application Server Products, the Accused Oracle Database Products and the Accused Oracle Web Cache Products.

A. Applicable Legal Principles

The law on summary judgment and literal infringement is well settled.

1. Summary Judgment

Summary judgment should be granted if the "pleadings, the discovery and disclosure materials on file, and any affidavits show that there is no genuine issue as to any material fact and that the movant is entitled to judgment as a matter of law." Fed. R. Civ. P. 56(c). The moving party bears the burden to prove that there are no genuine issues of material fact. *Matsushita Elec. Indus. Co., Ltd. v. Zenith Radio Corp.*, 475 U.S. 574, 586 n.10 (1986). A fact that can "alter the outcome [is] material." *Horowitz v. Fed. Kemper Life Assurance Co.*, 57 F.3d 300, 302 n.1 (3d Cir. 1995). A fact issue is "genuine...if the evidence is such that a reasonable jury could return a verdict for the nonmoving party." *Anderson v. Liberty Lobby, Inc.*, 477 U.S. 242, 248 (1986). "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."

Callaway Golf Co. v. Acushnet Co., 523 F. Supp. 2d 388, 394 (D. Del. 2007) (citing *Anderson*, 477 U.S. at 249).

2. Infringement

An infringement determination requires a two-step analysis. *First*, the court must construe the asserted claims so as to ascertain their meaning and scope. *Second*, the claims as construed must be compared to the accused product. *See Markman v. Westview Instruments, Inc.*, 52 F.3d 967, 976 (Fed. Cir. 1995), *aff'd*, 517 U.S. 370 (1996). Claim construction is a question of law, while infringement is a question of fact. *See Cybor Corp. v. FAS Techs., Inc.*, 138 F.3d 1448, 1454 (Fed. Cir. 1998) (*en banc*). To establish 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). Here, summary judgment of infringement should be granted because there are no genuine issues of material fact as to how the Accused Oracle Products work and Oracle literally infringes Claim 11 as a matter of law under either party's claim constructions.

B. EpicRealm Should Be Granted Summary Judgment That Oracle Literally Infringes Claim 11 Of The '554 Patent With Respect To The Accused Oracle Products.

Oracle literally infringes Claim 11 of the '554 Patent by making, using, selling, offering for sale and/or importing in the United States: (1) the Accused Oracle Application Server Products; (2) the Accused Oracle Database Products; and (3) the Accused Oracle Web Cache Products. Claim 11 of the '554 Patent recites:

> 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.

(Exh. 1, '554 Patent at col. 10, ll. 24-40.)

1. The Accused Oracle Products Meet the Preamble Of Claim 11 Of The '554 Patent.

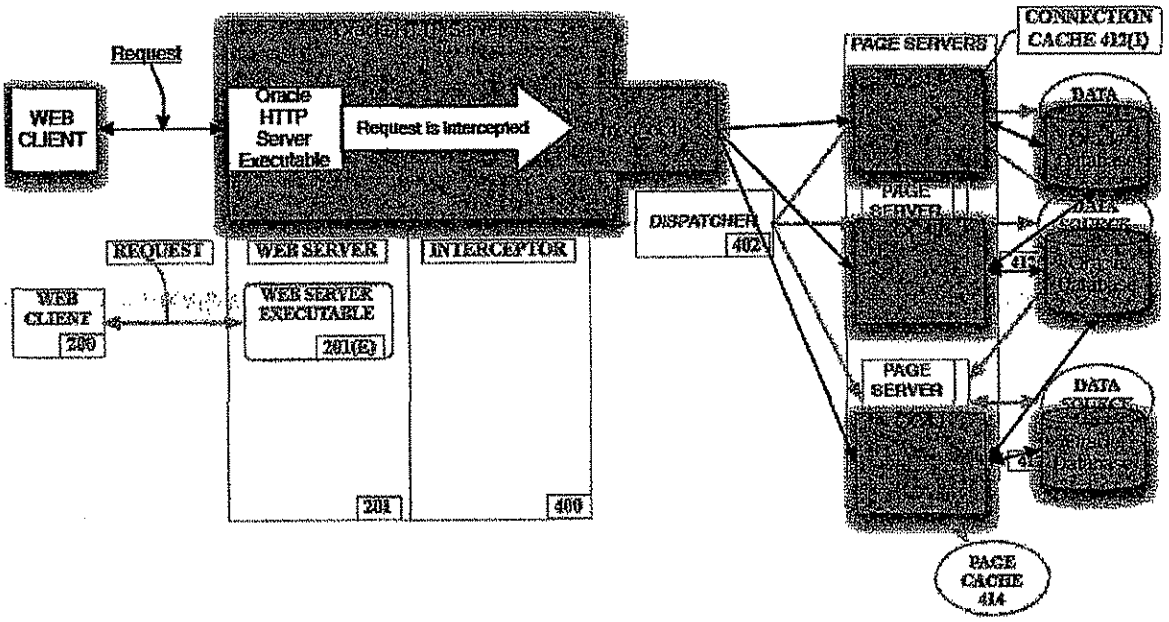
REDACTED

The remainder of the infringement proofs will treat the body of Claim 11 separately for each of the three Accused Oracle Products and under both parties' constructions.

2. The Accused Oracle Application Server Products Literally Infringe Claim 11 Of The '554 Patent.

Oracle literally infringes Claim 11 of the '554 Patent under either party's construction by making, using, selling, offering for sale and/or importing the Accused Oracle Application Server

Products. Below is a schematic diagram summarizing epicRealm's infringement accusations on the Accused Oracle Application Server Products overlaid on Figure 4 of the '554 Patent.



a) "routing a dynamic Web page generation request from a Web server to a page server, said page server receiving said request"

The Accused Oracle Application Server Products cause a computer system to perform this step under both parties' constructions.[1] The Accused Oracle Application Server Products route dynamic Web page generation requests under both parties' constructions. (Finkel Decl. ¶14.) In addition, the Accused Oracle Application Server Products meet the "Web server" limitation under either party's construction because the Oracle HTTP Server is HTTP-compliant server software that receives Web page requests and returns Web pages in response to the requests. (Finkel Decl. ¶15.) Moreover, the Accused Oracle Application Server Products also meet the "page server" limitation because OC4J is page-generating software that generates dynamic Web pages and can run on a processor separate from that of the "Web server." (Finkel

[1] The parties' proposed constructions are referenced throughout and also submitted in the supporting appendix as Exhibit 10.

Decl. ¶16.) Further, the "page server" can receive a request because, for example, it can generate a dynamic Web page in response to the request. (Finkel Decl. ¶16.)

Thus, the Accused Oracle Application Server Products route a dynamic Web page generation request from a "Web server" (Oracle HTTP Server) to a "page server" (OC4J), and the "page server" receives the request under either party's constructions as a matter of law. (Exh. 19 at 2-2 to 2-5.)

b) "and releasing said Web server to process other requests"

The Accused Oracle Application Server Products meet this limitation under either party's construction. The Accused Oracle Application Server Products include a "Web server" (Oracle HTTP Server). EpicRealm contends that "releasing" means "freeing." Oracle contends that "releasing" means "said page server receiving said request and said page server performing an act (separate from merely receiving the request) to free the Web server to process other requests." The Accused Oracle Application Server Products meet the "releasing" limitation under both parties' constructions for the same two reasons.

First, the "Web server" (Oracle HTTP Server) is freed by the "page server" (OC4J) processing the request. When the "Web server" receives a request, it is assigned the task, duty or obligation to process the request (*e.g.*, generate a Web page). (Finkel Decl. ¶17.) Upon identifying that the request can be satisfied by a "page server," the request is routed from the "Web server" to a "page server." (*Id.*) When the "page server" receives the request, it processes the request, thus freeing the "Web server" of the task, duty or obligation to satisfy the request. (*Id.*) As required under Oracle's construction, the act, separate from merely receiving the request by the "page server" to free the "Web server," is the processing of the request.

As a non-technical analogy, when an employer accepts work from a client, the employer is assigned the task, duty or obligation to do the work for that client. (Finkel Decl. ¶18.)

However, the employer is freed to do work for other clients when the employer assigns the work to an employee. (*Id.*) Thus, the fact that a "page server" processes the dynamic Web page request means that the "Web server" does not need to process it, causing the "Web server" to be released or freed to process other requests.

Second, the "Web server" (Oracle HTTP Server) is freed by the "page server" (OC4J) acknowledging receipt of a request.

REDACTED

This acknowledgement tells the sender that its request has been transmitted successfully; the receipt of this acknowledgement allows the "Web server" to release resources associated with this request. (*Id.*; Finkel Decl. ¶19.)

REDACTED

Continuing the non-technical analogy from above, but for the employee's acceptance of an assignment, the employer would be obligated to do the work or to assign it to another employee. (Finkel Decl. ¶20.) Thus, the "page server" performs the act of sending the acknowledgement, which is an act separate from merely receiving a request, that frees the "Web server" to process other requests. Therefore, the Accused Oracle Application Server Products meet this limitation under either party's construction as a matter of law.

c) "wherein said routing step further includes the steps of intercepting said request at said Web server,"

EpicRealm contends that "intercepting said request at said Web server" means "intercepting the handling of a request at a Web server," while Oracle proposes "receiving a

request at the Web server machine and diverting the request before the Web server executable can process the request."

The Accused Oracle Application Server Products meet this limitation under either party's construction. This limitation is met under epicRealm's construction because the "Web server" (Oracle HTTP Server) can handle some requests on its own while other requests, such as dynamic Web page generation requests, are intercepted at the "Web server" and, instead, routed to a "dispatcher" for processing by a "page server." (Finkel Decl. ¶21.) Thus, in the Accused Oracle Application Server Products, the handling of a dynamic Web page generation request can be intercepted at the "Web server." This limitation is also met under Oracle's construction because the Accused Oracle Application Server Products receive a dynamic Web page generation request at the "Web server" machine and divert the request before the "Web server" executable can process the request; instead, the request is sent to a "page server" (OC4J) for processing. (*Id.*) Therefore, the Accused Oracle Application Server Products meet this limitation under either party's construction as a matter of law.

d) "routing said request from said Web server to a dispatcher, and dispatching said request to said page server;"

EpicRealm contends that no construction is necessary for "dispatcher." Oracle contends that "dispatcher" means "a software program for determining which page server should be used to process a dynamic web page generation request." EpicRealm contends that "dispatching" means "examining a request to make an informed selection of which page server should process the request based on dynamic information maintained about page servers, the dynamic information indicating which page server can more efficiently process the request, and sending the request to the selected page server." Oracle contends that "dispatching [said request to said

page server]" means "analyzing a request to make an informed selection of which page server should process the request, and sending the request to that page server."

The Accused Oracle Application Server Products route the request from the "Web server" (Oracle HTTP Server) to a "dispatcher" (mod_oc4j). Mod_oc4j can dispatch the request to the "page server" (OC4J) using metric-based load balancing.

REDACTED

Mod_oc4j can then use one of its two metric-based load balancing algorithms and, based on dynamic information maintained about page servers indicating which page server can more efficiently process the request (for example, response time), mod_oc4j sends the request to the selected page server. Under Oracle's "dispatching" construction, mod_oc4j analyzes the request to make an informed selection of which page server should process the request and sends the request to that page server. (*Id.*; Exh. 7, Clark Rpt. 30-31.) Therefore, the Accused Oracle Application Server Products meet this limitation under either party's construction as a matter of law.

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

Under either party's constructions, the "page server" (OC4J) can process a request concurrently with the "Web server" (Oracle HTTP Server) processing other requests. (Finkel Decl. ¶22.) Therefore, the Accused Oracle Application Server Products meet this limitation under either party's construction as a matter of law.

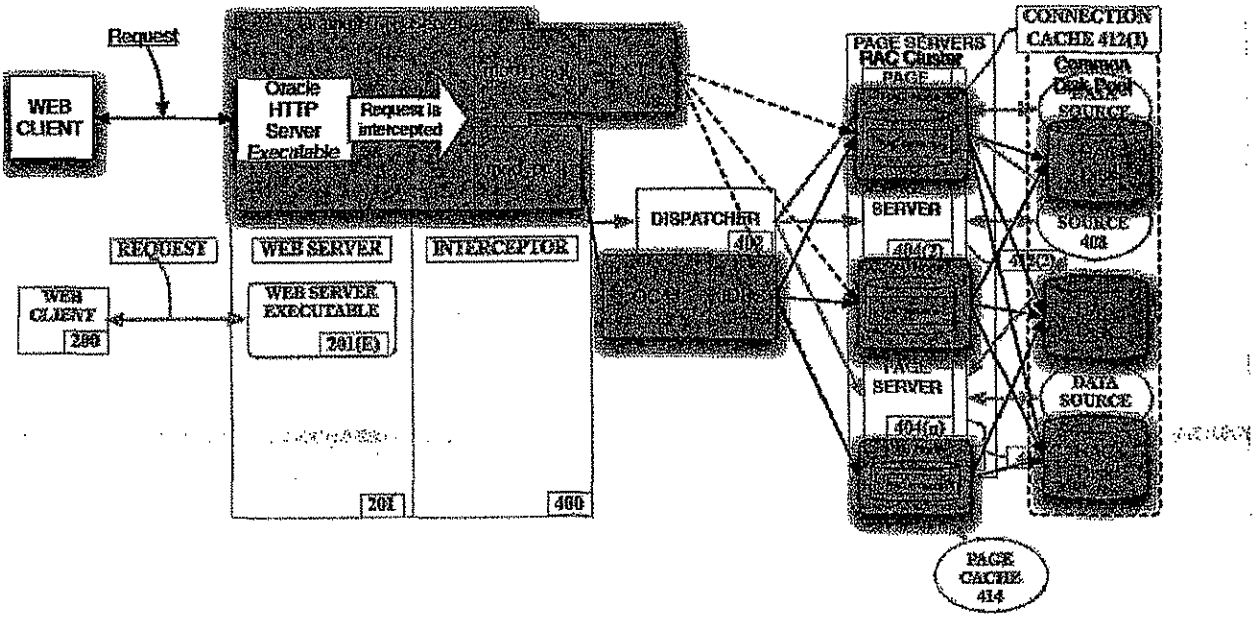
f) "dynamically generating a Web page, said Web page including data retrieved from one or more data sources."

The Accused Oracle Application Server Products can retrieve data from one or more data sources, *e.g.*, Oracle Database, under either party's construction. (Finkel Decl. ¶23.) The "page server" (OC4J) can dynamically generate a Web page including data retrieved from the one or more data sources (*e.g.*, Oracle Database). (*Id.*) Therefore, the Accused Oracle Application Server Products meet this limitation under either party's construction as a matter of law.

Therefore, the Court should grant partial summary judgment that the Accused Oracle Application Server Products literally infringe Claim 11 of the '554 Patent as a matter of law.

3. The Accused Oracle Database Products Literally Infringe Claim 11 Of The '554 Patent.

Oracle literally infringes Claim 11 of the '554 Patent by making, using, selling, offering for sale and/or importing the Accused Oracle Database Products. A further description of the Accused Oracle Database Products is helpful to understand the infringement issues. Below is a schematic diagram summarizing their infringement of Claim 11 overlaid on Figure 4 of the '554 Patent:



In order for a client (*e.g.*, Oracle HTTP Server) to connect with an Oracle RAC Database, the client can use an application programming interface ("API") (commonly referred to as a "database driver"). (Exh. 7, Clark Rpt. 47.) At least two APIs provided by Oracle are relevant here: Oracle Call Interface ("OCI") and Java Database Connectivity ("JDBC"). OCI allows applications to access a database using the C programming language. (*Id.*) Similarly, JDBC is typically used to connect Java applications, *e.g.*, OC4J, to databases, *e.g.*, Oracle database. (*Id.*) As an example, the OCI API allows a client (*e.g.*, Oracle HTTP Server) to send PL/SQL requests to the database. (*Id.*)

When clients and applications using OCI connect to a RAC database, they can use "session pooling" with runtime load balancing. (*Id.* at 50.) A session pool is a collection of stateless connections to a database, *e.g.*, a RAC database. (Exh. 21 at 9-7.) The sessions in the pool are reused saving the overhead of repeatedly creating and closing sessions. (Exh. 22 at ORCL01310470.) When session pooling and runtime load balancing are enabled, a work request can be serviced when OCI selects a session from its session pool based on the advice provided

by RAC through FAN events. (Exh. 7, Clark Rpt. 50.) In short, OCI runtime load balancing "is basically routing work requests to sessions in a session pool which have been selected as being able to best serve the work." (*Id.*)

Similarly, JDBC implements runtime load balancing to route work requests based on the advice provided by RAC through FAN events. (*Id.*) JDBC maintains an implicit connection cache of connections to an Oracle database, *e.g.*, a RAC database. (*Id.*) The JDBC connection cache is analogous to the OCI session pool. When runtime load balancing is enabled for JDBC, a work request can be serviced when JDBC selects a connection from its implicit connection cache based on the advice provided by RAC through FAN events. (*Id.*) Thus, as with OCI, JDBC enables the routing of work requests to connections in its implicit connection cache that can best serve the work request. (Exh. 23 at ORCL01054850.)

In addition to merely retrieving data from the common pool of disks, a RAC instance can generate a dynamic Web page in at least three ways: (1) operate XML DB (Exh. 12 at 1-17, 3-70 to 3-71); (2) execute a PL/SQL Web Application (Exh. 8, Figure 11-1); (3) execute a PL/SQL Server Page (Exh. 8 at 12-1).

As demonstrated below, the Accused Oracle Database Products literally infringe Claim 11 under either party's constructions.

a) "routing a dynamic Web page generation request from a Web server to a page server, said page server receiving said request"

The Accused Oracle Database Products can use any commercially available Web server including Oracle HTTP Server. (Finkel Decl. ¶24.) As explained above for the Accused Oracle Application Server Products, Oracle HTTP Server meets either party's construction for "Web Server" and receives Web page generation requests, which meet either party's construction for "request." (Finkel Decl. ¶15.) In addition, under either party's construction, the Accused Oracle

Database Products include a "page server" (a RAC instance). (Finkel Decl. ¶25.) The Accused Oracle Database Products can route a dynamic Web page generation request from the "Web server" to a "page server" in at least two ways.[2] *First*, as explained for the Accused Oracle Application Server Products, a request can be routed from Oracle HTTP Server to OC4J, which can then route it to a RAC instance using JDBC. (Exh. 7, Clark Rpt. 46.) *Second*,

REDACTED

In both cases, the RAC instance receives the request because, as explained above, it generates a response. Thus, in both cases, the Accused Oracle Database Products meet this limitation under either party's construction as a matter of law.

b) "and releasing said Web server to process other requests"

The "page server" (a RAC instance) releases the "Web server" (*e.g.*, Oracle HTTP Server) to process other requests under each party's construction in the same two independent ways as discussed for the Accused Oracle Application Server Products. *First*, the "Web server" offloads the request to the "page server," thereby freeing the "Web server" of the task, duty or obligation of responding to the request. *Second*, the "page server" acknowledges receipt of the request over a Reliable Network Protocol, such as TCP. When a Reliable Network Protocol acknowledgement ("ACK message") is received, the "Web server" can free resources that it otherwise would have had to use had the "Web server" not received the ACK message that the "page server" had received the request. For example, the "Web server" will resend a request if the page server does not acknowledge receipt of the request, thereby consuming additional

[2] There are several other ways in which Oracle HTTP Server can route a request to a RAC instance, however, only two are listed here.

resources. (Exh. 25, Clark Dep. 219:18-220:7.) Therefore, the Accused Oracle Database Products meet the "releasing" limitation under either party's construction as a matter of law.

c) "wherein said routing step further includes the steps of intercepting said request at said Web server,"

The Accused Oracle Database Products meet this limitation under either party's construction. As explained above for the Accused Oracle Application Server Products, requests are intercepted at the "Web server" (*e.g.*, Oracle HTTP Server) under epicRealm's construction because the handling of a dynamic Web page generation request can be intercepted at an Oracle HTTP Server. (Finkel Decl. ¶21.) As also explained above for the Accused Oracle Application Server Products, requests are intercepted at the "Web server" under Oracle's construction because a dynamic Web page generation request is received by Oracle HTTP Server and diverted before the "Web server" executable can process it as the request is instead sent to a "page server" (a RAC instance) for processing. (*Id*; Exh. 19 at 2-2 to 2-15.) Therefore, the Accused Oracle Database Products meet this limitation under either party's construction as a matter of law.

d) "routing said request from said Web server to a dispatcher, and dispatching said request to said page server;"

The Accused Oracle Database Products perform the step of "routing said request from said Web server to a dispatcher" under either party's construction. The "dispatcher" is OCI or JDBC performing runtime load balancing.

REDACTED

OCI and JDBC route the request to the RAC instance that can best service the request. Thus, OCI and JDBC are software programs that determine which "page server" (RAC instance) should be used to process the dynamic Web page generation request.

The Accused Oracle Database Products perform the "dispatching" limitation under either party's construction when OCI or JDBC routes a request using runtime load balancing. Both parties' constructions for "dispatching" require examining or analyzing a request to make an informed selection of which "page server" should process a request. Both OCI and JDBC examine a request during runtime load balancing.

In OCI, a request is examined in two ways as part of the "dispatching" process. *First*, OCI examines the request for "tagging" information. Oracle describes the tagging of a session in a session pool as follows:

> Some of the sessions in the pool may be 'tagged' with certain properties. For instance, a user may request for a default session, set certain attributes on it, then label it or 'tag' it and return in to the pool. That user, or some other user, can require a session with the same attributes, and thus request for a session with the same tag. There may be several sessions in the pool with the same tag. The 'tag' on a session can be changed or reset.

(Exh. 21 at 9-7.) Thus, when OCI selects a session from its session pool in order to route a request, the collection of sessions is narrowed based on tag information, and then, OCI, using runtime load balancing, selects the session that can best serve the request. (*Id.* at 9-8, 9-11.) The tag information used to narrow the session pool can come from examining a request. (Finkel Decl. ¶26.)

Second, OCI examines requests for authentication information. An OCI session pool can contain heterogeneous sessions. Heterogeneous sessions have "different security attributes and privileges." (Exh. 21 at 9-8.) Thus, when runtime load balancing is used with heterogeneous sessions, OCI can examine the request to first narrow the collection of sessions based on security attributes and privileges, and then OCI selects the session from the narrowed collection that can best serve the request. The security attributes and privileges can come from the request. (Finkel Decl. ¶27.) Therefore, in both ways, OCI examines or analyzes a request.

JDBC also examines requests in two ways as part of the "dispatching" process. JDBC distinguishes between connections in the JDBC implicit connection cache based on both log-in credentials and striping as explained by Oracle's witness:

REDACTED

Thus, when JDBC determines which connection should be used to route a request to a RAC instance using runtime load balancing, it goes through the following steps:

REDACTED

Therefore, in both ways, JDBC examines or analyzes the request as part of the "dispatching" process.

In addition, both JDBC and OCI make an "informed selection of which page server should process the request" when they select the best RAC instance to process the request based on information provided by RAC. As explained above, RAC collects information on all of its

instances and provides advice to JDBC and OCI on how to best route work requests. When JDBC and OCI implement runtime load balancing, they select the best instance to service a work request based on that advice. (Exh. 7, Clark Rpt. 50; Exh. 23 at ORCL01054850.) Thus, the selection of a RAC instance by JDBC and OCI is an informed selection of which page server should process a request.

After JDBC or OCI selects the best RAC instance to service the request, the request is routed to that RAC instance. Thus, the Accused Oracle Database Products also meet the requirement of both EpicRealm's and Oracle's construction that the request be sent to the selected "page server."

Finally, epicRealm's construction also requires that the informed selection be based on dynamic information maintained about "page servers" indicating which "page server" can more efficiently process the request. In the Accused Oracle Database Products, such dynamic information is the information relied upon by JDBC and OCI for runtime load balancing. Such information can be, *e.g.*, service time or throughput. (Exh. 13 at 4-12.) The information is "dynamic" because it cannot be generated prior to runtime. This information is maintained about page servers (RAC instances) and indicates which page server can more efficiently process the request. Thus, the Accused Oracle Database Products meet the "dispatching" limitation under either party's construction as a matter of law.

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

Under either party's constructions, the "page server" (a RAC instance) can process a request concurrently with the "Web server" (*e.g.*, Oracle HTTP Server) processing other requests. (Finkel Decl. ¶29.) Therefore, the Accused Oracle Database Products meet this limitation under either party's construction as a matter of law.

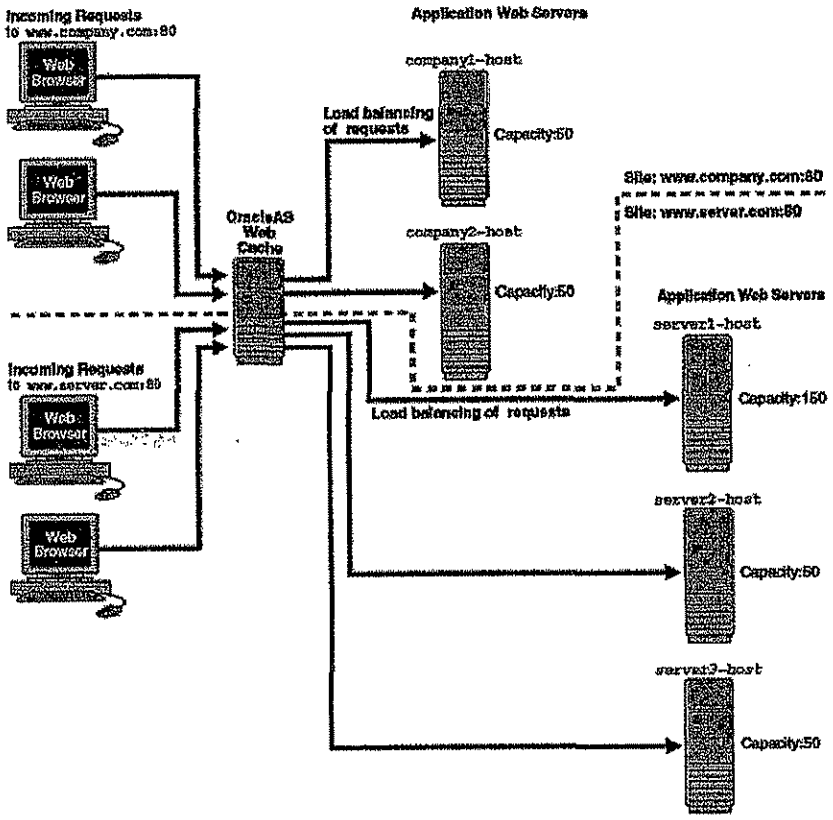
f) "dynamically generating a Web page, said Web page including data retrieved from one or more data sources."

The common pools of disks that can be accessed by the RAC instances meet either party's construction of "data sources." (Finkel Decl. ¶30.) When the RAC instance generates a dynamic "Web page," it can retrieve data from one or more "data sources" and include that data in the dynamically generated "Web page." (*Id.*) Thus, the Accused Oracle Database Products meet this limitation under both parties' constructions as a matter of law.

As shown above, the Accused Oracle Database Products literally meet every limitation of Claim 11 of the '554 Patent under either party's construction. Therefore, epicRealm should be granted partial summary judgment that the Accused Oracle Database Products literally infringe Claim 11 of the '554 Patent.

4. The Accused Oracle Web Cache Products Literally Infringe Claim 11 Of The '554 Patent.

Oracle literally infringes Claim 11 of the '554 Patent by making, using, selling, offering for sale and/or importing the Accused Oracle Web Cache Products. When there is a cache miss, Oracle Web Cache distributes the request to an Application Server for processing as shown in the diagram below:



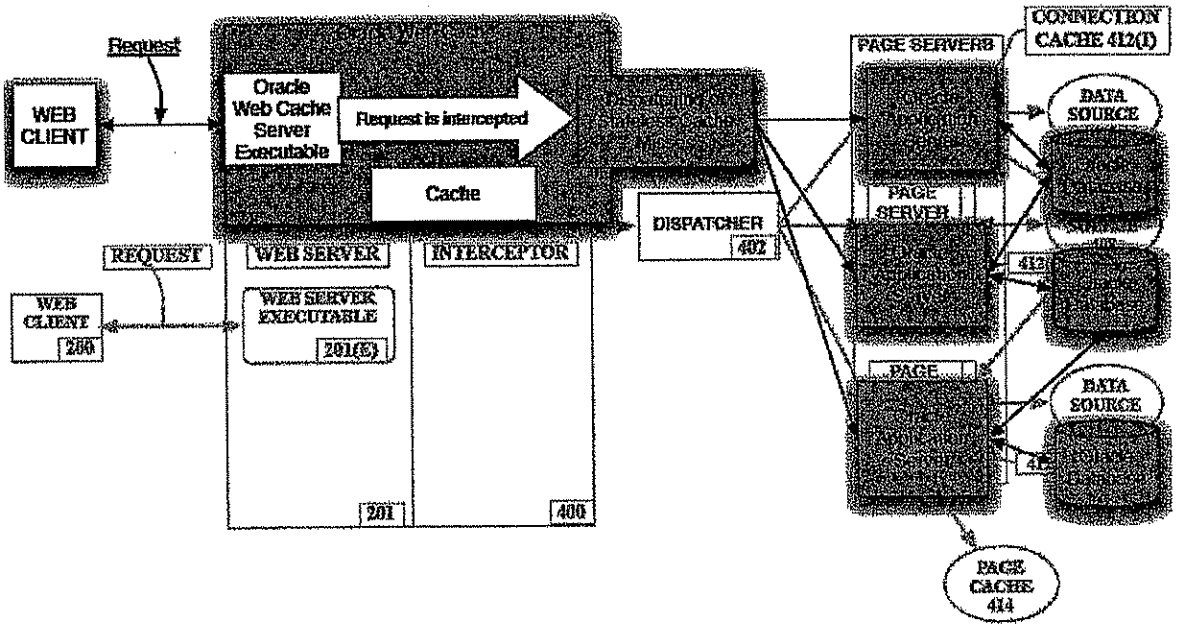
(Exh. 11 at 1-14.) When determining which Application Server should process a request, Oracle Web Cache first examines the request to determine if it is a stateless request, *i.e.*, whether an Application Server is maintaining information about the client sending the request. (*Id.* at 1-16 to 1-17.) Stateful requests are routed to the same Application Server that processed a previous request from that client. (*Id.*)

Next, the Web Cache determines which Application Servers are capable of processing the request by examining the request and comparing the information against a configuration file. (*Id.* at 1-8 to 1-16.) For example, in the diagram above, the top two Application Web Servers process requests sent to www.company.com:80, while the bottom three Application Web Servers process requests sent to www.server.com:80. The diagram provides an example of how Web Cache distributes requests according to weighted available capacity. In the diagram, the bottom three servers have capacities of 150, 50, and 50 respectively. If their current loads were, *e.g.*, 75,

40 and 5, their weighted available capacities would be 50%, 20%, and 90%, respectively. Thus, the next "cache miss" request will be routed to Application Server 3 because it has the highest weighted available capacity (90%).

When an Application Server receives a request, it processes it and returns a response to the Web Cache. Oracle Web Cache then stores the response in its cache and returns the response to the client. In the Accused Oracle Web Cache Products, the application servers can be Oracle Application Servers which, as explained previously, can respond to dynamic Web page requests in a variety of ways, including using OC4J, Perl, or CGI.

The Accused Oracle Web Cache Products literally infringe Claim 11 under either party's constructions. Below is a schematic diagram summarizing the infringement of Claim 11 by the Accused Oracle Web Cache Products overlaid on Figure 4 of the '554 Patent:



a) "routing a dynamic Web page generation request from a Web server to a page server, said page server receiving said request"

The Accused Oracle Web Cache Products cause a computer system to perform this step under both parties' constructions. The Accused Oracle Web Cache Products route dynamic Web page generation requests under both parties' constructions. (Finkel Decl. ¶31.) In addition, the Accused Oracle Web Cache Products meet the "Web server" limitation under either party's construction because Oracle Web Cache is HTTP-compliant server software that receives Web page requests and returns Web pages in response to the requests. (*Id.*) Moreover, the Accused Oracle Web Cache Products also meet the "page server" limitation because, as explained above, Oracle Application Server (the "page server") includes page-generating software that generates dynamic Web pages and can run on a processor separate from that of the Oracle Web Cache. (*Id.*) Further, the "page server" can receive a request because, for example, it can generate a dynamic Web page in response to the request. (*Id.*)

b) "and releasing said Web server to process other requests"

The "page server" (Oracle Application Server) releases the "Web server" (Oracle Web Cache) to process other requests under each party's construction in the same two independent ways as discussed for the Accused Oracle Application Server Products. *First*, the "Web server" offloads the request to the "page server," thereby freeing the "Web server" of the task, duty or obligation of responding to the request. *Second*, the "page server" acknowledges receipt of the request over a Reliable Network Protocol, such as TCP. When the ACK message is received, the "Web server" can free resources that it otherwise would have had to use had it not received the ACK message that the "page server" received the request.

REDACTED

Therefore, the Accused

Oracle Web Cache Products meet the "releasing" limitation under either party's construction as a matter of law.

c) "wherein said routing step further includes the steps of intercepting said request at said Web server,"

The Accused Oracle Web Cache Products meet this limitation under either party's construction. Under epicRealm's construction, the handling of a dynamic Web page generation request can be intercepted at the "Web server" because Oracle Web Cache handles some requests on its own, *e.g.* retrieving pre-existing content from its cache, while other requests, "cache misses," are intercepted and, instead, routed to a "page server" (*e.g.*, Oracle Application Server) for processing. (Finkel Decl. ¶32.) Similarly, the "Web server" receives a dynamic Web page generation request at the "Web server" machine and diverts the request before the "Web server" executable can process the request because the request is instead routed to a "page server" for processing. (*Id.*) Therefore, the Accused Oracle Web Cache Products meet this limitation under either party's construction as a matter of law.

d) "routing said request from said Web server to a dispatcher, and dispatching said request to said page server;"

The Accused Oracle Web Cache Products meet this limitation under either party's construction. The "dispatcher" in the Accused Oracle Web Cache products is the portion of Oracle Web Cache that determines which "page server" (Oracle Application Server) should be used to process a request. As explained above, Oracle Web Cache routes the requests to the Oracle Application Server with the greatest weighted available capacity. Thus, Oracle Web Cache is a software program that determines which "page server" should be used to process the dynamic Web page generation request.

The Accused Oracle Web Cache Products perform the "dispatching" limitation under either party's construction. The Accused Oracle Web Cache Products perform "dispatching"

when Oracle Web Cache routes a request according to the weighted available capacity of the Oracle Application Servers. Both parties' constructions for "dispatching" require examining or analyzing a request to make an informed selection of which "page server" should process a request. Oracle Web Cache examines or analyzes a request in two ways.

First, as explained above, Oracle Web Cache examines a request to determine if it is a stateful or a stateless request. (Exh. 11 at 1-16 to 1-17.) Stateful requests are routed to the Application Server that previously processed a request from the same client. (*Id.*) Stateless requests are distributed based on the weighted available capacity of the Application Servers. (Exh. 11 at ORCL00513921.)

Second, as also explained above, Oracle Web Cache examines a request to determine which Application Servers can process the request. As shown in the diagram above, the top two Application Servers process requests sent to www.company.com:80, while the bottom three Application Servers process requests sent to www.server.com:80. Thus, for example, Oracle Web Cache examines the requests to determine if the request should be distributed among the top two Application Servers or the bottom three Application Servers.

In addition, Oracle Web Cache makes an "informed selection of which page server should process a request" when it routes a request to the Application Server with the greatest weighted available capacity. Importantly, the "load" is dynamic information maintained by Web Cache that represents the total number of requests sent by the Oracle Web Cache that are currently being processed by the Application Server. (Exh. 11 at ORCL00513921.) Thus, the weighted available capacity also is dynamic information because it cannot be generated prior to runtime. (Finkel Decl. ¶33.) Therefore, Web Cache makes an informed selection of which

"page server" should process a request because Oracle Web Cache bases its selection on the weighted available capacity of the "page servers."

Finally, epicRealm's construction also requires that the "informed selection" be based on dynamic information maintained about "page servers" indicating which "page server" can more efficiently process the request. Here, that dynamic information is the weighted available capacity of the "page servers." This information is maintained about "page servers" and indicates which "page server" can more efficiently process the request. Thus, the Accused Oracle Web Cache Products meet this limitation under either party's construction as a matter of law.

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

Under either party's constructions, the "page server" (*e.g.*, Oracle Application Server) can process a request concurrently with the "Web server" (Oracle Web Cache) processing other requests. (Finkel Decl. ¶34.) Therefore, the Accused Oracle Web Cache Products meet this limitation under either party's construction as a matter of law.

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

The "data sources" in the Accused Oracle Web Cache Products include Oracle Database or any other data source available to Oracle Application Server. As explained above, when Oracle Application Server generates a dynamic Web page, it can retrieve data from one or more data sources and include that data in the dynamically generated Web page. (Finkel Decl. ¶¶23, 35.) Thus, the Accused Oracle Web Cache Products meet this limitation under both parties' constructions as a matter of law.

The Accused Oracle Web Cache Products literally meet every limitation of Claim 11 of the '554 Patent under either party's constructions. Therefore, epicRealm should be granted partial summary judgment that the Accused Oracle Web Cache Products literally infringe Claim 11 of the '554 Patent.

V. CONCLUSION

For the foregoing reasons, the Court should grant epicRealm partial summary judgment that the Accused Oracle Products literally infringe Claim 11 of the '554 Patent.

POTTER ANDERSON & CORROON LLP

OF COUNSEL:

Harry J. Roper
George S. Bosy
Aaron A. Barlow
Patrick L. Patras
David R. Bennett
Paul D. Margolis
Benjamin J. Bradford
Emily C. Johnson
JENNER & BLOCK
330 N. Wabash Avenue
Chicago, IL 60611-7603
Tel: (312) 923-8305

By: */s/ David E. Moore*
Richard L. Horwitz (#2246)
David E. Moore (#3983)
Hercules Plaza, 6th Floor
1313 N. Market Street
Wilmington, DE 19899
Tel: (302) 984-6000
rhorwitz@potteranderson.com
dmoore@potteranderson.com

Attorneys for Defendant/Counterclaim Plaintiff epicRealm Licensing, LP

Dated: July 31, 2008
876827 /31393 / Oracle

Public Version Dated: August 7, 2008

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

CERTIFICATE OF SERVICE

I, David E. Moore, hereby certify that on August 7, 2008, the attached document was electronically filed with the Clerk of the Court using CM/ECF which will send notification to the registered attorney(s) of record that the document has been filed and is available for viewing and downloading.

I further certify that on August 7, 2008, the attached document was Electronically Mailed to the following person(s):

Mary B. Graham
Morris, Nichols, Arsht & Tunnell LLP
1201 North Market Street
P.O. Box 1347
Wilmington, DE 19899
MGraham@MNAT.com

James G. Gilliland
Igor Shoiket
Townsend and Townsend and Crew LLP
Two Embarcadero Center
Eighth Floor
San Francisco, CA 94111-3834
OracleEpicrealm@townsend.com

Theodore T. Herhold
Robert J. Artuz
Eric M. Hutchins
Eric A. Mercer
Joseph A. Greco
Nitin Gupta
Townsend and Townsend and Crew LLP
379 Lytton Avenue
Palo Alto, CA 94301
OracleEpicrealm@townsend.com

Chad E. King
Townsend and Townsend and Crew LLP
1200 Seventeenth Street
Suite 2700
Denver, CO 80202
OracleEpicrealm@townsend.com

/s/ David E. Moore
Richard L. Horwitz
David E. Moore
Potter Anderson & Corroon LLP
Hercules Plaza – Sixth Floor
1313 North Market Street
P.O. Box 951
Wilmington, DE 19899-0951
(302) 984-6000
rhorwitz@potteranderson.com
dmoore@potteranderson.com