IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

ORACLE CORPORATION and ORACLE U.S.A. INC.,	)	
Plaintiffs/Counterdefendants,	)	
v.	)	C.A. No. 06-414 (SLR)
EPICREALM LICENSING, LP,	)	
Defendant/Counterclaimant.	)	

ORACLE'S BRIEF IN SUPPORT OF MOTION FOR SUMMARY JUDGMENT OF NONINFRINGEMENT

MORRIS, NICHOLS, ARSHT & TUNNELL LLP
Mary B. Graham (#2256)
James W. Parrett, Jr. (#4292)
1201 North Market Street
P.O. Box 1347
Wilmington, DE 19899
mgraham@mnat.com
jparrett@mnat.com
(302) 658-9200

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

OF COUNSEL:

James G. Gilliland
Theodore T. Herhold
Joseph A. Greco
Robert J. Artuz
Eric M. Hutchins
TOWNSEND AND TOWNSEND AND CREW LLP
379 Lytton Avenue
Palo Alto, CA 94301
(650) 326-2400

Dorian Daley
Peggy E. Bruggman
Matthew M. Sarboraria
ORACLE CORPORATION
ORACLE U.S.A., INC.
500 Oracle Parkway
Redwood Shores, CA 94065

DATED: July 31, 2008

TABLE OF CONTENTS

Page

I. NATURE AND STAGE OF THE PROCEEDINGS 1

II. SUMMARY OF ARGUMENT 1

III. STATEMENT OF FACTS 4

A. The Patented Technology 4

B. Oracle's Accused Products 6

1. Web Cache 6

2. Internet Application Server 8

3. Database 11

IV. LEGAL STANDARDS 12

A. Summary Judgment Standard 12

B. Direct Infringement Standard 13

C. Indirect Infringement Standard 14

V. ARGUMENT 14

A. Oracle And Its Customers Do Not Directly Infringe 15

1. The "Web Server" In Oracle's Products Is Not "Released" By A "Page Server" 15

a) Oracle Web Cache Is Never "Released" To Concurrently Process Additional Requests 15

b) Oracle HTTP Server (OHS) Is Never "Released" To Concurrently Process Additional Requests 24

2. Oracle's Products Do Not "Intercept" A Request For A Dynamic Web Page At A "Web Server" 27

a) Oracle Web Cache Does Not "Intercept" As Required By Each Asserted Claim 27

b) Oracle HTTP Server (OHS) Does Not "Intercept" As Required By Each Asserted Claim 30

3. Oracle's Products Do Not "Dispatch" A Request From A "Dispatcher" To A "Page Server" 31

a) Oracle Web Cache Lacks A "Dispatcher" 31

b) Oracle Internet Application Server Lacks A "Dispatcher" 33

c) Oracle Database Lacks a "Dispatcher" 33

d) Oracle Database Does Not "Dispatch" A Request From A "Dispatcher" To A "Page Server" 34

4. Oracle's Use Of The Oracle Products To Operate Its Own Websites Does Not Directly Infringe 35

B. Oracle Does Not Indirectly Infringe 35

1. The Absence Of Direct Infringement Precludes Contributory Infringement And Inducement Of Infringement 35

2. Oracle's Products Have Substantial Noninfringing Uses 36

3. Oracle Did Not Have Notice Of The Patents Prior To 2006 37

C. Oracle Does Not Infringe Under The Doctrine Of Equivalents 37

1. EpicRealm Waived Any Right To Assert Infringement Under The Doctrine Of Equivalents 37

2. EpicRealm's Untimely Allegations Under The Doctrine Of Equivalents Fail As A Matter Of Law 38

VI. CONCLUSION 39

TABLE OF AUTHORITIES

Page

Cases

Arthur A. Collins, Inc. v. Northern Telecom Ltd.,
216 F.3d 1042 (Fed. Cir. 2000) 14

Bayer AG v. Elan Pharm. Research Corp.,
212 F.3d 1241 (Fed. Cir. 2000) 13

Celotex Corp. v. Catrett,
477 U.S. 317 (1986)..... 13

DSU Medical Corp. v. JMS Co., Ltd.,
471 F.3d 1293 (Fed. Cir. 2006) (en banc) 14, 37

General Mills, Inc. v. Hunt-Wesson, Inc.,
103 F.3d 978 (Fed. Cir. 1997) 13, 24, 30, 31, 33

Golden Blount v. Robert H. Peterson Co.,
365 F.3d 1054 (Fed. Cir. 1994) 14

Kahn v. Gen. Motors Corp.,
135 F.3d 1472 (Fed. Cir. 1998) 12, 13

Matsushita Elec. Indus. Co. v. Zenith Radio Corp.,
475 U.S. 574 (1986)..... 13

Network Commerce, Inc. v. Microsoft Corp.,
422 F.3d 1353 (Fed. Cir. 2005) 38

Panduit Corp. v. Hellermanntyton Corp.,
451 F.3d 819 (Fed. Cir. 2006) 38

Process Control Corp. v. Hydreclaim Corp.,
190 F.3d 1350 (Fed. Cir. 1999) 23

TechSearch, LLC v. Intel Corp.,
286 F.3d 1360 (Fed. Cir. 2002) 13

Trell v. Marlee Elecs. Corp.,
912 F.2d 1443 (Fed. Cir. 1990) 14, 37

Wolverine World Wide Inc. v. Nike, Inc.,
38 F.3d 1192 (Fed. Cir. 1994) 13

Statutes

35 U.S.C. § 271(b) 14

35 U.S.C. § 271(c) 14

Rules

Fed. R. Civ. P. 56(b) 12

Fed. R. Civ. P. 56(e) 13

Plaintiffs and Counterdefendants Oracle Corporation and Oracle U.S.A. Inc. (jointly "Oracle") submit this brief in support of Oracle's Motion for Summary Judgment of Noninfringement.

I. NATURE AND STAGE OF THE PROCEEDINGS

Oracle incorporates by reference its summary of the Nature and Stage of the Proceedings set forth in its Claim Construction Brief, filed herewith.[1]

II. SUMMARY OF ARGUMENT

1. The Court should grant Oracle summary judgment of noninfringement on all the asserted claims of both patents-in-suit.

2. There is no genuine issue as to any material fact concerning the structure or functions of Oracle's accused software products: (a) Internet Application Server ("IAS"), and (b) Oracle Database operating in a Real Application Cluster ("RAC") configuration. The question of literal infringement collapses, therefore, into claim construction and should be decided on summary judgment.

3. EpicRealm asserts three different infringement theories in which the *same* parts of Oracle's products often are alleged to satisfy *different* elements of the claims. All three theories fail as a matter of law.

---

[1] In addition to the present brief, Oracle also files herewith: 1) Oracle's Opening Claim Construction Brief; 2) Oracle's Motion for Summary Judgment of Invalidity; 3) Oracle's Motion for Partial Summary Judgment of No Willful Infringement; 4) Oracle's Motion to Exclude Expert Report and Testimony of EpicRealm's Damages Expert, Michael J. Wagner; and 5) Oracle's Motion to Exclude Damages Based on Foreign Sales. Exhibits in support of Oracle's motions are contained in the Appendix of Exhibits in Support of Oracle's Opening Claim Construction Brief and Motions Filed July 31, 2008, filed herewith. Evidentiary support for each of the exhibits is contained in the accompanying declarations in support of Oracle's motions.

4. In one theory, epicRealm alleges that IAS infringes because Oracle's Web Cache is the "Web server" (and the "second computer system" and the "HTTP-compliant device") of the asserted claims.[2] Web Cache is an optional cache product that sits in front of Oracle HTTP Server ("OHS"), which is itself a Web server, but which epicRealm characterizes as part of a "page server" for this theory. This theory fails as a matter of law because the following limitations of all the claims are lacking:

- The alleged "page server" of IAS (which includes OHS), which receives requests for dynamic Web pages from Web Cache, does not "release" Web Cache to process other requests;
- Neither Web Cache nor any part of IAS "intercepts" requests for dynamic Web pages before Web Cache can process them;
- IAS does not have a "dispatcher"; epicRealm's assertion that Web Cache is both the Web server and the dispatcher is wrong as a matter of law because the claims require that the request be "routed" from the Web server to the dispatcher.

5. In a second theory, epicRealm ignores Web Cache and alleges that OHS, which epicRealm alleged was part of the "page server" in the prior theory, is the "Web server" and the "dispatcher." This theory fails as a matter of law for similar reasons:

- The "page server" under this theory (the OC4J server that is part of IAS) does not "release" OHS to process other requests;
- Neither OHS nor any part of IAS "intercepts" requests for dynamic Web pages before OHS can process them;
- IAS does not have a "dispatcher"; epicRealm's assertion that OHS is both the Web server and the dispatcher is wrong as a matter of law because the claims require that the request

[2] Some claims refer to the part of the system that receives request for Web pages as a "Web server," while other claims use the term "second computer system" or "HTTP-compliant device." The parties' proposed constructions for all these terms is the same. Thus, they can all be thought of as "Web servers" for purposes of this motion.

be "routed" from the Web server to the dispatcher.

6. In a third theory, epicRealm again ignores Web Cache, alleges that OHS is the "Web server," though not also the dispatcher, and alleges that one instance of an Oracle RAC Database is the page server. This theory fails as a matter of law for similar reasons and an additional reason:

- The "page server" under this theory (RAC database instance) does not "release" OHS to process other requests;
- Neither OHS nor any part of IAS or a RAC database instance "intercepts" requests for dynamic Web pages before OHS can process them;
- The JDBC and OCI interfaces to the RAC database are part of the OHS Web server program and cannot be "dispatchers," as epicRealm alleges under this theory, because they are not separate programs from the Web server and the claims require that the request be "routed" from the Web server to the dispatcher;
- The JDBC and OCI interfaces do not perform "dispatching" because they never "examine" or "analyze" requests for dynamic Web pages, as required under both parties' claim constructions.

7. EpicRealm did not disclose a doctrine of equivalents theory in its final contentions or in its expert's infringement report. EpicRealm's attempt to raise a doctrine of equivalents theory on the "releasing" element—in its expert's rebuttal report on invalidity—is improper and should not be considered. In any event, it fails to raise a genuine issue of material fact.

8. Oracle does not contributorily infringe or induce infringement of the patents-in-suit. Oracle's products cannot be used to directly infringe any claim. Even under epicRealm's theories, the products have substantial noninfringing uses. EpicRealm has no factual proof that Oracle customers are using the products to infringe, even under epicRealm's theories. Oracle did not have actual notice of the patents-in-suit until January 2006, and could not have indirectly infringed prior to that date.

III. STATEMENT OF FACTS[3]

A. The Patented Technology

The patents-in-suit share a common specification and relate to structures and methods for managing dynamic Web page requests by computers. The patent specification describes the prior art Web server environment as lacking "any mechanism for managing the Web requests or the Web sites." A1, Col.4:33-34.[4] The patent further states that "[a]lthough these [Web server] machines may be running 'multi-threaded' operating systems that allow transactions to be processed by independent 'threads,' all the threads are nevertheless on a single machine, sharing a processor." Col. 4:41-45. The specification points out that "[w]hen numerous requests are being simultaneously processed by multiple threads on a single machine, the Web server can slow down significantly and become highly inefficient." Col. 4:48-51.

The claimed invention seeks to address this perceived problem by utilizing a "partitioned architecture." Col. 6:24-27. As shown in Figure 4 of the patent, this partitioned architecture has several components, including a Web server (201), an "Interceptor"[5] (400), a "Dispatcher" (402), and one or more "page servers" (404(1)-(n)):

---

[3] The Statement of Facts in Oracle's opening brief on Claim Construction contains background on the parties and general facts of the case. This motion sets forth more detailed facts concerning the patents-in-suit and the accused products.

[4] Appendix Exhibits are contained in Oracle's Appendix of Exhibits in Support of Oracle's Opening Claim Construction Brief and Motions Filed on July 31, 2008, filed concurrently herewith and are referred to by number as "A__."

[5] All of the asserted claims require "intercepting" a request, but only '554 claim 10 requires an "interceptor." The other claims require, however that the "intercepting" be done "at the Web server" or at a similar component (such as "second computer system" or "HTTP-compliant device") that performs Web server functions.

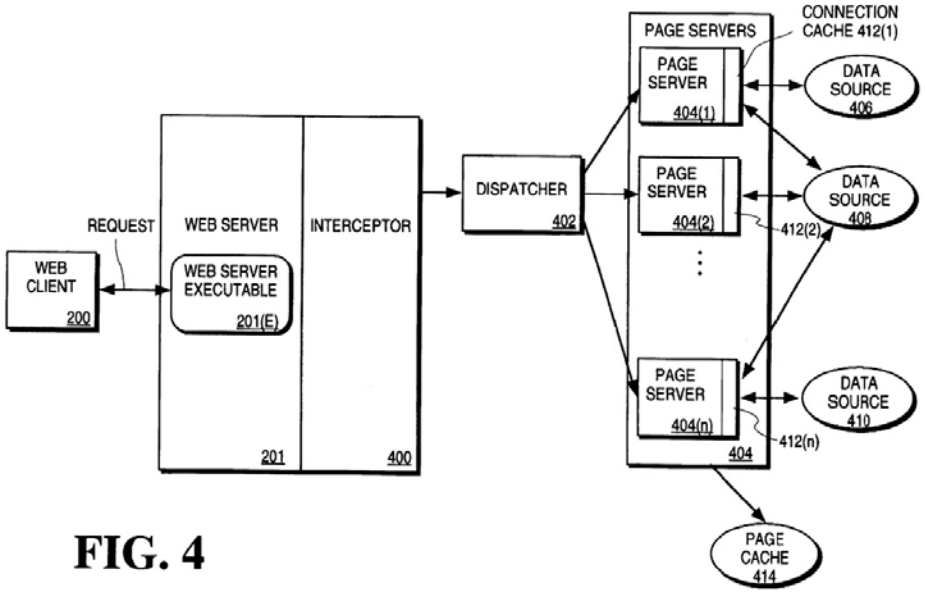


FIG. 4

Claim 1 of the '554 patent is exemplary of the asserted claims:

> 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.

Claim 1 describes the method in which the components of the partitioned architecture shown in Figure 4 are used. A Web client initiates a request for dynamic Web content. The Interceptor (or other component "at the Web server") diverts the request, which would otherwise be processed by the Web server, and routes the request to the Dispatcher. Col. 4:58-60. The Dispatcher receives the intercepted request and dispatches it to one of the page servers for

processing. The page server receives the request from the Dispatcher and releases the Web server so that it may concurrently process other requests. After releasing the Web server, the page server processes the request, while the Web server, having been released by the page server, concurrently processes other requests. The page server processes the request in its possession by dynamically generating a Web page, in response to the request, based on data obtained from a data source (Fig. 4, items 406, 408, 410, 414).

By using the page servers to release the Web server to concurrently process other requests, the claimed invention seeks to address the perceived inefficiency of the prior art Web servers that rely on multiple processing "threads."

B. Oracle's Accused Products

1. Web Cache

Web Cache is a software program designed to maintain a cache, or a local store, of frequently accessed Web pages. Without Web Cache, a request made by a Web client (*e.g.*, a browser, such as Microsoft's Internet Explorer) is sent directly to another software program, Oracle HTTP Server ("OHS"), which is a Web server. Web Cache, however, sits in front of OHS, the Web server, and receives the Web client's request for a Web page before OHS does. Nevertheless, in one of epicRealm's infringement theories, epicRealm argues that Web Cache is the "Web server" of the claims, and that OHS is not a Web server, but rather part of the "page server." The facts about Web Cache's construction and operation described in this section form the basis for Oracle's noninfringement arguments as to these allegations.

Web Cache is able to cache both static and dynamic content. Declaration of Dr. Paul C. Clark in Support of Oracle's Opening Claim Construction Brief and Motion for Summary Judgment of Noninfringement ("Clark Decl.") ¶52. If Web Cache has the requested content in its cache (a "cache hit") then the Web Cache simply returns the requested content to the Web

client, eliminating the need to repeatedly process identical requests through the Web server and database, and thus reducing the load on the Web server sitting behind the Web Cache. Clark Decl. ¶51. If the requested content is not cached (a "cache miss"), Web Cache sends the request to a Web server for processing. *Id.* The Web servers sitting behind Web Cache, to which the request is sent in the event of a cache miss, are called "origin servers." *Id.*

Web Cache is designed as a single multi-threaded program, in which each Web request is delegated to a "fiber" or "thread" within the Web Cache program. Clark Decl. ¶55. Web Cache is further divided into "Front End" fibers that connect to Web clients and perform cache lookups, and "Back End" fibers that connect to the origin server and add content to the cache. *Id.* The importance of Web Cache's "multi-threaded" or "multi-fibered" program is that, as will be explained later, neither Web Cache, nor any fiber or thread in Web Cache, is "released" by a "page server" to concurrently process other requests.

Front End fibers are created on-the-fly upon receipt of a request, up to a user-configured maximum number of fibers. Clark Decl. ¶56. For each new Web request received from a Web client—whether static or dynamic—a Front End fiber compares the request to the previously requested Web pages it has stored within its cache. Clark Decl. ¶¶57,86-89. In the event of a cache hit, the Front End fiber returns the Web page to the Web client. In the event of a cache miss, the Web Cache program will create a Back End fiber that sends the request to the OHS software (the origin server). Clark Decl. ¶58. The Front End and Back End fibers will then wait until either the origin server returns the requested Web page, or a time-out error occurs. *Id.*

Once OHS returns the requested content, the Front End and Back End fibers return the content to the Web client and insert the content into the cache so that future requests for the same content can be satisfied by the Web Cache without involving OHS. This accomplishes Web

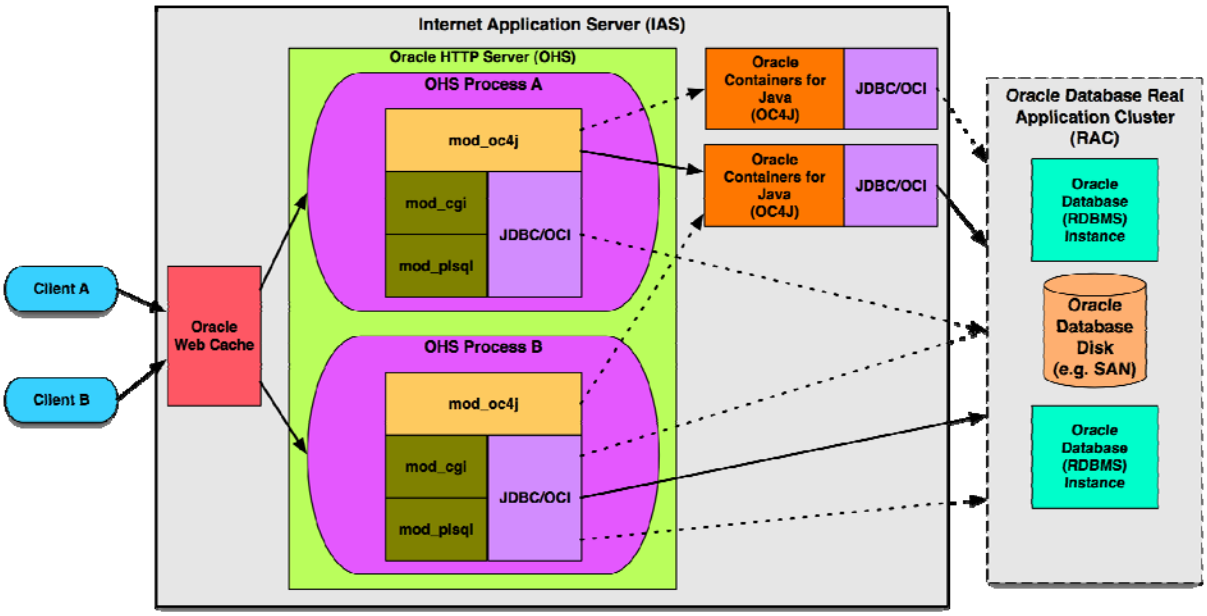
Cache's intended purpose to reduce the load on the computer running OHS. A fiber can handle only one request at a time, and is never free to concurrently process another request when waiting for requested content from OHS. Clark Decl. ¶¶58-60;64-66.

2. Internet Application Server

Oracle's Internet Application Server ("IAS") is a large "enterprise" software package that includes numerous features and functions, many of which are unrelated to the Web page technology at issue in this case. IAS includes the OHS and Oracle Containers For Java ("OC4J") software programs, which are involved in responding to Web page requests. OHS is the Web server component of IAS, and consists of several functions implemented within the same software executable. An "executable" (for a single-threaded program such as OHS) is a single instance of a computer program (a series of instructions that carry out an algorithmic solution to a task) that is executed sequentially by a computer system. Clark Decl. ¶95. The OHS executable includes the HTTP Listener and a collection of modules.

The HTTP Listener receives incoming requests and then begins processing them using the modules. Some of these modules are standard "Apache" Web server modules developed by the open-source software community, while some are Oracle-developed modules specific to IAS components. Each module is designed to process certain request types. For example, mod_plsql is designed to process requests that require execution of a PL/SQL (Structured Query Language) query to a database. The module mod_oc4j is designed to process requests that require execution of an application in the Java computer language. Clark Decl. ¶¶96-99. OC4J is a separate executable process which OHS, using mod_oc4j, employs to run Web-based Java programs. Clark Decl. ¶100.

As shown in the following figure prepared by Dr. Clark, OHS uses multiple executable processes, where each new request is handled by a separate process. Clark Decl. ¶¶101-105.



When a Web client sends a request to the IP address of the computer running OHS, that request is assigned by the computer's operating system to one of the OHS processes running on that computer. In the default configuration, there are 256 OHS processes running at any given time. Clark Decl. ¶101. Once a particular OHS process has received the request, that OHS process begins processing the request. As will be further explained below, neither OHS nor any of these processes is ever "released" by any component epicRealm labels a "page server" to concurrently process other requests.

The OHS process begins processing the request by calling each module, in the order in which they were loaded into memory when the OHS processes were first started. Clark Decl. ¶102. Importantly, no request for a dynamic Web page is ever "intercepted" before the OHS processes it; OHS processes *all* requests in the same way. Each module, when called, looks at the request to determine whether it is the type of request that the module is configured to process. *Id.* If the OHS process gets through calling all of the modules and no module has the capability to process the request, the OHS process will attempt to satisfy the request from the static content

available to it on the computer's local storage (the file system on the computer's hard drive). Clark Decl. ¶103. If there is no file in local storage that satisfies the request, then OHS will send an error message to the Web client that the requested content was not found. *Id.* When the OHS process sends such an error message, the OHS process performs no further processing of the request. Only when the OHS process sends either the requested Web page content or an error message to the Web client is the OHS process done processing the request and able to handle another request. *Id.*

In the case of requests for dynamic Web content using a Java-based application, mod_oc4j is the module that would typically handle the request. The OHS process invokes the mod_oc4j function, which sends the request to an OC4J process, and then waits until OC4J returns the completed request. Clark Decl. ¶104. When OC4J completes its processing of the request and sends the requested Web content back to the OHS process via mod_oc4j, the OHS process stops waiting, and finishes processing the request by sending the requested content back to the Web client that initiated the request.

If another Web client sends a second request when this particular OHS process is waiting, that second request is assigned by the operating system to one of the *other* OHS processes for the very reason that the prior process is never "released" to concurrently process other requests. Clark Decl. ¶¶105,112-114. If, hypothetically, all of the OHS processes (256 by default) were busy processing requests or waiting for content from an OC4J process, then the Web server will be unable to process additional requests until one of the occupied OHS processes completes processing and returns a completed request to the Web client. Clark Decl. ¶114. Again, none of the processes can process another request until processing of a prior request is completed.

When more than one OC4J process is available, mod_oc4j will load balance requests across those multiple OC4J processes. Users can configure mod_oc4j to perform one of eight different load balancing policies: (1) Random, (2) Round Robin, (3) Random with Local Affinity, (4) Round Robin with Local Affinity, (5) Random using Routing Weight, (6) Round Robin using Routing Weight, (7) Metrics Based, and (8) Metric Based with Local Affinity. Clark Decl. ¶¶106-108. EpicRealm contends that a user practices the method and system claims of the patents-in-suit only when using the last two types of "metrics based" load balancing, and *admits* that none of the other types of load balancing can be used to practice the claimed inventions because they do not meet the "dispatching" limitation. A27, at pp. 13-14.

IAS's default load balancing policy is Round Robin, based on a simple sequential algorithm, that epicRealm admits does not meet the "dispatching" requirement of the claims. Clark Decl. ¶108. Few, if any, Oracle customers use the metrics-based load balancing algorithms which epicRealm contends are necessary to infringement of the claims. A21, at p. 137:4-11.

3. Database

Oracle Database primarily consists of a Relational Database Management System ("RDBMS"). An RDBMS is a complex set of software programs that controls the organization, storage, management, and retrieval of data in a database (a structured collection of records or data) based on a relational model. The primary role of an RDBMS is to service requests for data stored in the database. Clark Decl. ¶137.

EpicRealm's infringement theory regarding Oracle Database depends on Oracle customers' use of a special configuration of Oracle Database called Real Application Clusters ("RAC"). A27, at p. 3. RAC is a database option in which multiple computers, or "nodes," access a common database. Clark Decl. ¶143. In other words, it can be thought of as a giving a

user the ability to access multiple "copies" or "instances" (through multiple "nodes") of the same database.

EpicRealm's infringement theory also requires that the RAC Database be used in combination with the use of what is called JDBC and OCI connection caching/pooling and runtime load balancing.[6] A27, at p. 10 & Exhs. G,H. JDBC (Java Database Connectivity) and OCI (Oracle Call Interface) are application programming interfaces ("APIs") that application servers and other Database clients can use to connect to an Oracle Database. Clark Decl. ¶¶140-142. JDBC and OCI are software libraries that are linked into, and part of, the OHS executable. Clark Decl. ¶151.

The JDBC and OCI interfaces are not separate programs from the Web server. Nor do JDBC and OCI transmit requests for Web pages; they transmit database queries. Clark Decl. ¶¶137-138,151. Oracle Database is primarily used as a data source, not as a means to generate Web pages. *Id.*

IV. LEGAL STANDARDS

A. Summary Judgment Standard

Summary judgment shall be granted where the moving party "shows 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(b). EpicRealm, as the holder of the asserted patents, bears the burden of proving infringement. *Kahn v. Gen. Motors Corp.*, 135 F.3d 1472, 1476-77 (Fed. Cir. 1998). To obtain summary judgment, Oracle need only show "an absence of evidence to support the

---

[6] As with "metrics based" load balancing in IAS, few Oracle customers use JDBC and OCI connection caching/pooling and runtime load balancing, which epicRealm contends are necessary to infringement of the claims.

nonmoving party's case." *Celotex Corp. v. Catrett*, 477 U.S. 317, 325 (1986). EpicRealm, as the nonmoving party, must "come forward with 'specific facts showing that there is a genuine issue for trial'" in order to successfully oppose this motion. *Matsushita Elec. Indus. Co. v. Zenith Radio Corp.*, 475 U.S. 574, 587 (1986) (quoting Fed. R. Civ. P. 56(e)).

B. Direct Infringement Standard

A determination of infringement requires a two-step analysis. The court first construes each asserted patent claim as a matter of law and then determines whether each and every limitation of each such claim is found in the accused product. *Kahn*, 135 F.3d at 1476. The first step, claim construction, is a matter of law, while the second step is a question of fact. *TechSearch, LLC v. Intel Corp.*, 286 F.3d 1360, 1369 (Fed. Cir. 2002). "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). Where there is no material dispute of fact over the physical characteristics of the accused product, *e.g.*, its structures or functions, "the question of literal infringement collapses into claim construction and is amenable to summary judgment." *General Mills, Inc. v. Hunt-Wesson, Inc.*, 103 F.3d 978, 983 (Fed. Cir. 1997).[7] Where an independent patent claim is not infringed, it is axiomatic that any claims depending from that claim are also not infringed. *Wolverine World Wide Inc. v. Nike, Inc.*, 38 F.3d 1192, 1199 (Fed. Cir. 1994).

---

[7] Infringement under the doctrine of equivalents is not at issue in this lawsuit because epicRealm elected not to timely assert any theory of infringement under the doctrine of equivalents. *See* Section V.C., *infra*. EpicRealm's attempt to introduce a belated, conclusory theory in its expert's *invalidity rebuttal report* cannot defeat summary judgment. *Id.*

C. Indirect Infringement Standard

The two forms of indirect infringement within the United States are active inducement pursuant to 35 U.S.C. § 271(b), and contributory infringement pursuant to § 271(c). Both theories of indirect infringement require that direct infringement by another be proven as a predicate. *Arthur A. Collins, Inc. v. Northern Telecom Ltd.*, 216 F.3d 1042, 1048-49 (Fed. Cir. 2000). Contributory infringement requires that the accused infringer sell a "component" of a patented invention "knowing the same to be especially made or especially adapted for use in an infringement of such patent." *Trell v. Marlee Elecs. Corp.*, 912 F.2d 1443 (Fed. Cir. 1990). There can be no contributory infringement, however, if the component is either not "a material part of the invention," or has a "substantial non-infringing use." *Golden Blount v. Robert H. Peterson Co.*, 365 F.3d 1054, 1061 (Fed. Cir. 1994). The patentee has the burden of proving that the component sold has no "substantial non-infringing use." *Id.* Inducement of infringement requires that the accused infringer "knew or should have known his actions would induce actual infringements." *DSU Medical Corp. v. JMS Co., Ltd.*, 471 F.3d 1293, 1306 (Fed. Cir. 2006) (en banc). (quoting *Manville Sales Corp. v. Paramount Systems, Inc.*, 917 F.2d 544, 553 (Fed. Cir. 1990).

V. ARGUMENT

EpicRealm has three different infringement theories, in which the *same* parts of Oracle's accused software allegedly satisfy *different* elements of the asserted claims. These theories are described in the Summary of Argument, ¶¶4-6. The reasons why each theory fails are discussed below with respect to the claim elements that are lacking. In its first theory, epicRealm labels Web Cache as both the "Web server" and the "dispatcher," and labels OHS as part of the "page server." *See* Section V.A.1(a), 2(a), and 3(a), *infra*. In a second theory, epicRealm labels OHS as the "Web server," part of OHS (mod_oc4j) as the "dispatcher," and OC4J as the "page server."

See Section V.A.1(b), 2(b), and 3(b), *infra*. In a third theory, epicRealm labels OHS as the "Web server," the JDBC and OCI interfaces as the "dispatcher," and a RAC Database instance as the "page server." *See* Section V.A.1(b), 2(b), and 3(c)-(d), *infra*.

A. Oracle And Its Customers Do Not Directly Infringe

1. The "Web Server" In Oracle's Products Is Not "Released" By A "Page Server"

a) Oracle Web Cache Is Never "Released" To Concurrently Process Additional Requests

EpicRealm's first theory fails because the "releasing" limitation is absent under either party's claim construction of that term. EpicRealm alleges that Web Cache is the required "Web server" (and "second computer system" and "HTTP-compliant device") of the asserted claims. Therefore, whatever epicRealm labels a "page server" must both receive a request from Web Cache *and* must "release" Web Cache to concurrently process *other* requests while the "page server" processes that first request. *See*, *e.g.*, '554 claim 1, which provides in pertinent part:

> 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 . . . processing said request, said processing being performed by said page server while said Web server concurrently processes said other requests

Cols. 8:66-9:1; 9:6-8. Web Cache is never "released," by any "page server" or otherwise, to process other requests under either party's claim construction. Oracle construes this required act of releasing as "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." EpicRealm proposes that "releasing" be construed to mean "freeing."

EpicRealm attempts to apply both parties' claim constructions to argue that Web Cache is "released" as required by the claim. First, epicRealm offers an "implicit releasing" theory, whereby it argues that under its construction ("freeing"), the "page server" can "release" Web

Cache to process other requests merely by receiving the first request and processing it. Second, epicRealm offers an "explicit releasing" theory, whereby it argues that under either party's construction, the Web server is "released" by what is called a TCP acknowledgment or "ACK" sent by the operating system. Both theories are wrong: Web Cache does not meet this required limitation of the asserted claims under either party's construction.

The parties also proffer slightly different constructions of the term "Web server." However, while the parties constructions are not identical they are in accord that "Web server" may refer to "software, or a machine running such software, that receives Web page requests and returns Web pages in response to the requests."[8] Again, however, neither construction saves epicRealm from judgment of noninfringement, because Oracle's Web Cache does not meet the releasing limitation under either definition of "Web server."

Implicit Releasing

EpicRealm's implicit releasing theory asserts that because a "page server"[9] may process a given dynamic Web page request, the Web server is released or freed to process other requests. A27, at p. 4 & Exh. E, at p. 4. As Oracle argues in its claim construction brief, this effectively reads the "releasing" limitation out of the claims, because the claims already require the page server to receive and process the request. Unless the term "releasing" means something more

---

[8] Oracle's construction is: "HTTP-compliant server software, or a machine running such software, that receives Web page requests and returns Web pages in response to the requests." EpicRealm's construction is: "Software, or a machine having software, that receives Web page requests and returns Web pages in response to the requests."

[9] EpicRealm's infringement theory in which Web Cache is the alleged "Web server" alleges that the "page server" is a conglomeration of Oracle's OHS (which is itself a Web server), the OHS mod_oc4j, and whatever OC4J server that mod_oc4j may route a request to for processing.

than fulfilling those requirements, the term is superfluous. Nevertheless, even under this convoluted claim construction, and irrespective of whether the Web server is "software" or a "machine running that software," Oracle does not infringe.

Oracle's expert, Dr. Clark, reviewed the Web Cache source code relevant to epicRealm's releasing theories. This code establishes that the ability or inability of Web Cache—either the Web Cache software fibers themselves or the computer server machine on which they reside—to concurrently process additional requests is not affected in any way by the sending of a request to an alleged page server. Clark Decl. ¶55-66. Hence, even if epicRealm's claim construction of releasing ("freeing") were adopted and it were applied to require merely "implicit releasing," epicRealm's theory that Web Cache is "released" by a "page server" fails.

If the "Web server" is the Web Cache software, there is no infringement because the Web Cache software that receives a Web page request and returns a Web page in response to that request—a Web Cache fiber—is never released to concurrently process another request. *Id.* That fiber is either busy processing the request, or idly waiting for that request to be completed at the OHS (Oracle HTTP Server). *Id.* The Web Cache executable can create multiple fibers to process multiple requests, but each fiber can process only one request at a time—just as in the multi-threaded prior art servers noted and criticized in the specification of epicRealm's patents. *Id.* The fact that other fibers are available to concurrently process other requests is irrelevant, because none of those fibers were ever occupied with the first request; they could not be "released" from something that was not occupying them in the first place and they certainly were not "released" by the "page server's" receipt of a request from a completely different fiber.

If the "Web server" is the "machine having software" and the claims could somehow be satisfied if the "machine" were "released," even if the software were not, epicRealm's "implicit

releasing" theory still fails. The computer (machine) running the Web Cache executable software is either free or not free to handle additional requests irrespective of whether any given request is sent to a "page server" for further processing. The maximum number of simultaneous requests the Web Cache software running on a machine can accept at any given time is 700, in the default configuration. *Id.* That is, the software can create up to 700 fibers at one time. The system administrator can change the default from 700 to some other number and then restart the Web Cache with the new configuration, but regardless of the settings used, when Web Cache is turned on there is an established upper bound on the number of requests.

In the situation where the Web Cache software running on the machine is handling fewer than 700 requests, the fact that any of those requests may be sent to an OHS (part of the alleged "page server") does not aid or affect Web Cache's ability to concurrently process additional requests. When the next Web client sends a request, there will be an available Web Cache fiber ready to receive it, and this is true regardless of whether any of the other requests have been sent to alleged page servers or not. *Id.* In this situation, therefore, there is simply no reasonable way to argue that "the machine" running the Web Cache software has been "released" by a "page server" receiving and processing a request from one of the Web Cache fibers.

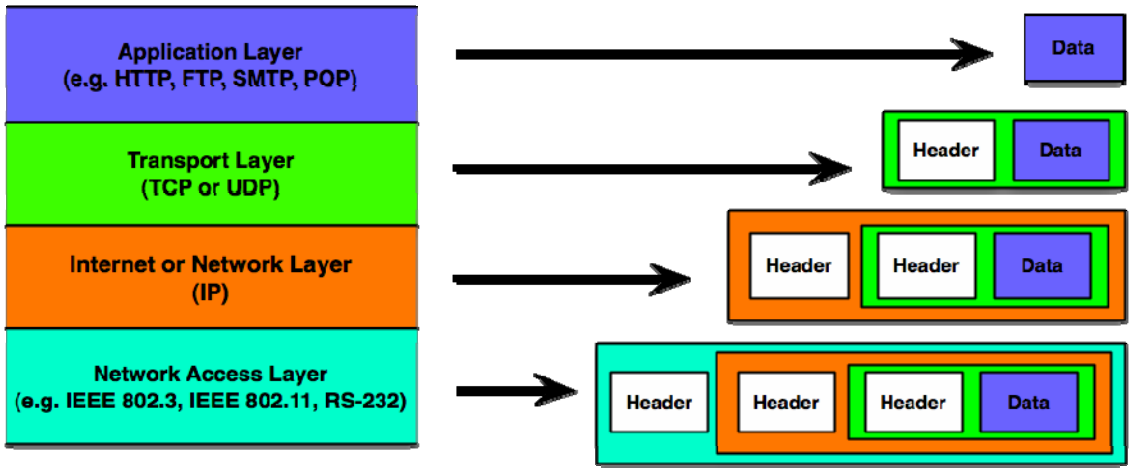
In a hypothetical situation where the server running Web Cache has 700 pending requests (or whatever the maximum configured limit is), the fact that any of those requests at the Web Cache may be sent to an OHS (again, part of the alleged "page server") does not aid or affect Web Cache's ability to concurrently process additional requests, because when the next Web client sends a request, there is no Web Cache fiber ready to receive it. *Id.* No new requests can be processed until one of the 700 Web Cache fibers completes its pending request and returns a Web page to the corresponding Web client. This is true regardless of whether any of the other

requests have been sent to alleged page servers. *Id.* It is Web Cache's return of requested content to a Web client when a request is completed—not the sending of a pending request to an OHS—that permits processing of other requests. Processing the next request after *completion* of the previous request is, by definition, not the concurrent processing required by the asserted claims. Again, "the machine" running the Web Cache software has not been "released" by a "page server" receiving and processing a request from one of the Web Cache fibers.

Since the ability or inability of Web Cache—either the Web Cache fibers themselves or the computer machine on which they reside—to concurrently process additional requests is not affected in any way by the sending of a request to an alleged page server, epicRealm's implicit releasing theory fails.

Explicit Releasing

EpicRealm argues, alternatively, that under either party's claim construction, an alleged page server "releases" Web Cache by sending a "TCP acknowledgement" to Web Cache. EpicRealm argues that the TCP acknowledgement "frees resources" on Web Cache, and that this alleged "freeing of resources," in turn, permits Web Cache to process additional requests. A27, Ex. E, at pp. 5-6. As with implicit releasing, regardless of whether "software" or "a machine having software" is the "Web server," epicRealm's theory fails. The following illustration provided by Dr. Clark depicts the standard TCP/IP communication model:



A software program operating at the application layer, such as a Web server (using the HTTP protocol) sends a communication by invoking the TCP transport layer present in the computer operating system.[10] Clark Decl. ¶¶68-75. The TCP layer encapsulates the HTTP communication with a TCP header that contains the necessary information to deliver the data via TCP. *Id.* The transport layer then sends the communication packet to the IP protocol, which again encapsulates the data with its own header, and so forth. *Id.*

The receiving computer works in the opposite fashion: The IP layer uses the IP header information and then discards it, sending the TCP-encapsulated data to the transport layer, where TCP uses the TCP header information, discards it, and then sends the HTTP communication to the Web application on the receiving end. *Id.* During TCP communication between two computers, the operating system of the receiving computer will transmit an acknowledgement (ACK) message when it receives a TCP segment. *Id.* If the ACK for that TCP segment is not received by the operating system of the sending computer within a certain time, the operating

[10] The operating system of a computer is the software that allows software at the application layer to function. Access to the Transport, Internet, and Network Access layers in the diagram above is supplied by the operating system.

system of the sending computer will resend that TCP segment. *Id.* Once received, the operating system of the sending computer discards the received packets from the TCP memory buffer. As discussed below, TCP ACK messages have no effect on, and are transparent to, the software layer where the Web server functionality resides. *Id.*

If the "Web server" is the Web Cache software, epicRealm's theory fails because, as shown above, fibers are never "released" by TCP acknowledgements to concurrently process another request. A fiber processing a request first checks to see if it can satisfy the request from the cache. In the event of a cache miss, the fiber will send the request to an OHS for further processing. Any number of TCP acknowledgements can be sent and received while the Web Cache fiber is waiting in this manner, and none of these TCP acknowledgements will permit that fiber to stop waiting and begin processing other requests. Clark Decl. ¶¶76-81. That Web Cache fiber will idly wait until it receives the requested content from the OHS. The fiber cannot concurrently process another request while the OHS (part of the alleged "page server") is processing the first request, as required by the "releasing" step of the asserted claims. *Id.*

Again, the Web Cache executable can create multiple fibers to process multiple requests, but each fiber can process only one request at a time—just as in the multi-threaded prior art servers criticized in the specification of epicRealm's patents. Clark Decl. ¶60. The fact that other fibers are available to concurrently process other requests is irrelevant, because none of those fibers were ever occupied with the first request, and could not be "released" from something that was not occupying them in the first place. Moreover, because each Web Cache fiber has its own TCP buffer, TCP acknowledgements related to one fiber and request have no effect on the operation of the other fibers and their processing of requests. Clark Decl. ¶79.

Thus a Web Cache fiber is never able to concurrently process additional requests as required by the "releasing" limitation of the asserted claims.

If the "Web server" is the "machine having software," epicRealm's "explicit releasing" theory based on the TCP acknowledgement also fails. The computer running the Web Cache executable software is either free or not free to handle additional requests irrespective of any TCP acknowledgements received. Clark Decl. ¶¶76-81.

Again, the default maximum number of simultaneous requests the Web Cache machine can accept at any given time is 700. In the situation where the Web Cache software running on the machine is handling fewer than 700 requests, the TCP acknowledgements do not affect Web Cache's ability to concurrently process additional requests: when the next Web client sends a request, there will be an available Web Cache fiber ready to receive it, regardless of the number of TCP acknowledgements received. *Id.* The "machine" running the Web Cache software has in no way been "released" by any TCP acknowledgement.

In the situation where the server running Web Cache has 700 pending requests (or whatever the maximum configured limit is), the fact that any number of TCP acknowledgements may be received does not affect Web Cache's ability to concurrently process additional requests. When the next Web client sends a request, there is no Web Cache fiber ready to process it. *Id.* No new requests can be processed until one of the 700 Web Cache fibers completes its pending request and returns a Web page to the corresponding Web client. This is true regardless of the number of TCP acknowledgements received. *Id.* It is the Web Cache's return of requested content to a Web client when a request is completed—not the receipt of TCP acknowledgements—that permits processing of additional requests, and processing the next

request after completion of the previous request is not the concurrent processing required by the asserted claims.

Dr. Finkel's assertion that "resources" are freed by TCP acknowledgements is a red herring. The claims require that the Web server be freed *to process other requests*—not merely that some "resources" be freed. *Process Control Corp. v. Hydreclaim Corp.*, 190 F.3d 1350, 1356-57 (Fed. Cir. 1999) (when claims are susceptible to only one reasonable construction, court must construe claims as the patentee drafted them). There is no factual dispute that when a computer's operating system receives a TCP acknowledgement, the operating system frees some of the memory allocated to the TCP buffer. Clark Decl. ¶75-81. But freeing "resources" in this way does not correlate with freeing the Web server to concurrently process additional requests. *Id.* As Dr. Clark verified, each Web Cache fiber has its own TCP buffer. Hence, even in the extremely unlikely scenario—contrary to normal operation—where one Web Cache fiber's TCP buffer reached the maximum size permitted by the computer's operating system because it did not receive any TCP acknowledgements, this failure will have no effect on the operation of the other Web Cache fibers. The remaining Web Cache fibers will function normally. Whether or not a TCP acknowledgement was sent to "clear" that buffer has no effect whatsoever on the other fibers and neither they, nor the machine on which they are running, could in any sense be "released" by a TCP acknowledgement. The affected Web Cache fiber (with the hypothetically filled TCP buffer) will either wait until the requested content is received from the OHS or until it times out and returns an error message to the Web client, just as if the TCP buffer were not full. *Id.* Again, whether or not a TCP acknowledgement was sent to "clear" the buffer could not in any sense "release" that fiber, or the machine on which it is running, to process other requests.

Dr. Clark's analysis of the Web Cache source code establishes that Web Cache's processing of requests simply does not practice "releasing" by an alleged page server. EpicRealm's expert, Dr. Finkel, did not review any of this source code to determine whether Web Cache was "released" as required by the claims:

> [MR. HERHOLD]: Q. Did you look at any code pertaining to other features of either Web Cache, iAS, or Database?
>
> MR. PATRAS: Object to the form.
>
> BY THE WITNESS [DR. FINKEL]: A. I don't believe so.
>
> Q. What about code that pertained to any intercepting that went on as you talk about it in your report?
>
> A. I don't recall looking at any.
>
> Q. How about any code that pertained to the releasing as you talk about it in your report?
>
> MR. PATRAS: Object to the form. Lack of foundation.
>
> A. I don't recall looking at any of that either.

A23, at pp. 82:14-83:4. Nor do the experts dispute how the TCP communication standard works. The ability or inability of Web Cache—either the Web Cache fibers themselves or the computer server machine on which they reside—to concurrently process additional requests is not affected in any way by the receipt of TCP acknowledgements. Like its theory of "implicit" releasing, epicRealm's explicit releasing theory fails. Summary judgment of noninfringement is, therefore, appropriate. *General Mills,* 103 F.3d at 983.

b) Oracle HTTP Server (OHS) Is Never "Released" To Concurrently Process Additional Requests

EpicRealm's two other infringement theories both allege that OHS, rather than Web Cache, is the "Web server" (and "second computer system" and "HTTP-compliant device") of the claims. The alleged "page server" in each of these two theories differs. As noted above, under epicRealm's second theory, the OC4J server of Internet Application Server ("IAS") is the alleged "page server." Under epicRealm's third theory, an instance of an Oracle RAC database is

the alleged "page server." Under both theories, OHS, the "Web server," must be "released" by the alleged "page server" to concurrently process additional requests as required by the asserted claims. While Web Cache does not figure into either of these alternative theories, epicRealm's infringement arguments as to why OHS is "released" by a page server are quite similar to its arguments as to why Web Cache is "released," and they fail for similar reasons.

Dr. Clark reviewed the relevant source code for IAS, including OHS, while Dr. Finkel, epicRealm's expert did not. That code establishes that OHS—whether you consider it to be the individual OHS processes or the machine running those processes—is never released by OC4J or by an instance of a RAC database to concurrently process other requests.

An OHS can have 256 separate processes running at a time (and can be configured to have a different number). In the situation where the OHS software running on a machine is handling fewer than 256 pending requests, neither the sending of any given request to an OC4J process or to a RAC instance, nor the receipt of any number of TCP acknowledgements, affects in any way OHS's ability to concurrently process additional requests. When the next Web client sends a request, there will be an available OHS process ready to receive it, regardless of the number of requests sent to OC4J or to a RAC instance and regardless of the number of TCP acknowledgements received. Clark Decl. ¶¶120-126. EpicRealm's expert, Dr. Finkel, does not dispute this:

> [MR. HERHOLD] Q. Is it your testimony that the Web server in that situation [Oracle HTTP Server] is unable to take on additional requests until the memory is freed up, as you've described it?
>
> A. No.
>
> MR. PATRAS: Objection. Incomplete hypothetical.
>
> BY THE WITNESS:
>
> A. No.
>
> Q. So the Web server continues to be able to take on additional requests regardless of whether the ACK message is sent back or not?

MR. PATRAS: Same objection.
BY THE WITNESS:
A. That's correct.

A23, at pp. 269:19-270:7. Dr. Clark's review of the source code indisputably demonstrates that when any individual OHS process sends a request to an OC4J process or to a RAC instance for further processing, that OHS process will idly wait until the OC4J process or the RAC instance returns the requested content. Clark Decl. ¶¶112-126. At no time prior to the completion of that first request, will the waiting OHS process be released or free to concurrently process another request.

If OHS hypothetically had 256 pending requests (or whatever the maximum configured limit is), neither the sending of any given request to an OC4J process or a RAC database instance, nor the receipt of any number of TCP acknowledgements, would affect in any way OHS's ability to concurrently process additional request. When the next Web client sends a request, there is no OHS process ready to process it. *Id.* No new requests can be processed until one of the 256 OHS processes completes its pending request and returns a Web page to the corresponding Web client. *Id.* It is OHS's return of requested content to the Web client when the request is completed—not the sending of the request to OC4J or to a RAC database instance, or the receipt of TCP acknowledgements—that permits processing of additional requests. Processing the next request after completion of the previous request is not the concurrent processing required by the asserted claims.

Moreover, epicRealm's specious "freeing of resources" argument fares no better here than it did for Web Cache. The freeing of "resources" on the Web server is irrelevant because the freeing of those resources does not permit OHS to concurrently process additional requests. OHS is never "released" and therefore neither IAS nor Oracle Database with a RAC

configuration can infringe. As with the fibers in Web Cache, each OHS process is allocated its own TCP buffer. Clark Decl. ¶123. The receipt (or failure to receive) TCP acknowledgements by one OHS process has no effect on other OHS processes, and the affected OHS process (with the hypothetically filled TCP buffer) will either wait until the requested content is received from the OC4J process or until it times out and returns an error message to the Web client, just as if the TCP buffer were not full. Clark Decl. ¶¶124-125.

2. Oracle's Products Do Not "Intercept" A Request For A Dynamic Web Page At A "Web Server"

a) Oracle Web Cache Does Not "Intercept" As Required By Each Asserted Claim

Web Cache is the alleged "Web server" (and "second computer system" and "HTTP-compliant device") in epicRealm's first infringement theory. Web Cache does not, however, practice the required step of "intercepting" under either party's claim construction. Oracle contends that intercepting is "receiving a request at the Web server machine and diverting the request before the Web server executable can process the request." EpicRealm contends that intercepting is "intercepting the handling of a request at a Web server." As more fully explained in Oracle's opening claim construction brief, both the "diverting" in Oracle's construction and the "intercepting" in epicRealm's construction require that the Web server of the patent distinguish requests for dynamic content from other requests and "process" or "handle" them differently from, for example, requests for static content. In his deposition, epicRealm's expert, Dr. Finkel, confirmed that intercepting requires that dynamic requests be intercepted:

> Q. Do you have some understanding that the patent describes that the intercepting takes place with respect to requests for dynamic information as opposed to static information?
>
> A. So Claim 1 says, for example, A computer implemented method for managing a dynamic Web page request.
>
> Q. Is that the answer to the question? The question is: Does the patent make some distinction between requests for dynamic

> information as opposed to requests for static information as far as the intercepting is concerned?
>
> MR. PATRAS: Objection. Vague. Lack of foundation.
>
> BY THE WITNESS: A. So in Claim 1, for example, is says, Routing said requests, receiving said requests; further steps, intercepting said requests. So in that claim, intercepting refers to intercepting dynamic Web page generation requests.
>
> Q. So you understood when it referred said requests as being a request for dynamic information?
>
> A. That's correct.

A23, at pp. 138:14-139:12. Because 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 under either party's construction.

Web Cache Does Not Intercept Under Oracle's Proposed Construction

Dr. Finkel describes epicRealm's "intercepting" theory under Oracle's proposed construction as follows:

> Logic associated with Oracle Web Cache instructs the intercepting of a request and routes it to a dispatcher. 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 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). Thus, the Accused Oracle Web Cache Products meet Oracle's construction of this limitation because at the time that the request is intercepted at the "Web server" . . . the "Web server" has not yet satisfied the request with pre-existing, cached or static content.

A27, Ex. E, at p. 7 (emphasis added). Dr. Finkel's characterization of Web Cache's operation is misleading. Web Cache's processing of a request never depends on whether the request is for static or dynamic content. Web Cache does not "intercept" because it never "diverts" the processing of requests for dynamic Web pages. Irrespective of any characteristic of the request, Web Cache always processes the request in the same way: by performing its primary purpose of checking for the requested content in its cache. It does not matter whether the content requested

is for a dynamic Web page or not. Clark Decl. ¶86. For example, a request may be for an article in an encyclopedia, *i.e.*, "static" content, or for a stock quote during trading hours, *i.e.*, dynamic content. Web Cache caches both static and dynamic content. For example, even though the stock price will be changing over time, Web Cache will store the last quote, and until erased from the cache, will satisfy the request with the stored quote. Regardless of whether the request is for static or dynamic content, therefore, Web Cache will always go through the same algorithm: check the cache, return content when there's a cache hit, send the request to the origin server when there's a cache miss. There is no diversion of requests before the Web Cache executable can process them, and therefore Web Cache does not intercept any requests.

Web Cache Does Not Intercept Under epicRealm's Proposed Construction

Dr. Finkel describes "intercepting" under epicRealm's proposed construction as follows:

> Logic associated with Oracle Web Cache instructs the intercepting of a request and routes it to a dispatcher. 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).

A21, Ex. E, at p. 6 (emphasis added). As discussed in Oracle's claim construction brief, epicRealm's construction of "intercepting" is circular, because it repeats the term to be construed. But "intercepting" must require some "diverting" of dynamic Web page requests to a page server before the Web server processes them. EpicRealm's original proposed construction of "intercepting" in the Texas Action recognized this, defining "intercepting" as "stopping, deflecting, or interrupting the processing of a request." A32, at p. 8. Moreover, as shown in Oracle's claim construction brief, epicRealm argued to the Patent Office that "merely routing" a dynamic Web page cannot constitute intercepting. The processing of dynamic requests must be treated differently than other requests if "intercepting" is to make any sense at all.

As stated above, Web Cache does not process requests differently based upon the content of a request. Neither requests for static content nor requests for dynamic content are ever diverted from Web Cache's normal processing. The "mere routing" of cache misses to an origin server cannot be "intercepting." Based on the undisputed facts as to Web Cache's functioning, it does not perform the "intercepting" step as a matter of law, and the Court should grant Oracle summary judgment of noninfringement for this reason. *General Mills*, 103 F.3d at 983.

b) Oracle HTTP Server (OHS) Does Not "Intercept" As Required By Each Asserted Claim

EpicRealm's two alternative infringement theories directed to Oracle Internet Application Server ("IAS") and Oracle Database identify OHS as the "Web server" (and "second computer system" and "HTTP-compliant device"). EpicRealm's "intercepting" theory for OHS is similar to its theory for Web Cache, and fails for the similar reasons, under both parties' claim constructions.

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. Clark Decl. ¶¶130-133. The OHS's determination of which module will process the request is not based on the requested content, *e.g.*, whether the request is for dynamic, as opposed to static, content; rather, it is determined by the nature of the Web application that needs to be executed. *Id.* If the request calls for generation of content—static or dynamic—via execution of a Java application, then mod_oc4j is selected. If the request calls for execution of a Perl program, then mod_perl will be selected, and so forth. OHS will always go through the same iterative process of invoking these modules in order to process each request, regardless of the requested content. No requests are "intercepted" or "diverted." *Id.*

As shown in Oracle's opening claim construction brief, epicRealm distinguished its claimed invention from the prior art Leaf patent to overcome the examiner's rejection of epicRealm's claims. OHS processes requests in a fashion similar to the processing disclosed in Leaf, in which the Web server would use part of the request URL and the MIME type of the requested content to determine which "transaction gateway" would process the request. EpicRealm distinguished this manner of processing requests in order to obtain allowance of the patent. In particular, epicRealm told the Patent Office that this type of processing is not "intercepting" because it is "merely routing" the request to the transaction gateway. Similarly, in OHS there is no diversion of requests before the OHS executable can process them, regardless of whether they are routed to either of the alternative alleged "page servers"—OC4J in epicRealm's IAS infringement theory and an instance of an Oracle RAC database in epicRealm's database theory. Based on the undisputed facts as to OHS's functioning, it does not perform the "intercepting" step as a matter of law, and the Court should grant Oracle summary judgment of noninfringement for this reason. *General Mills*, 103 F.3d at 983.

3. Oracle's Products Do Not "Dispatch" A Request From A "Dispatcher" To A "Page Server"

a) Oracle Web Cache Lacks A "Dispatcher"

EpicRealm's infringement theory based on Web Cache fails for the additional reason that Web Cache does not contain the required "dispatcher" of all the asserted claims. EpicRealm contends that no construction of "dispatcher" is necessary, while Oracle contends that a "dispatcher" is "a software program for determining which page server should be used to process a dynamic Web page generation request." All of the asserted claims require that a dynamic Web page generation request be routed from a Web server to a dispatcher. *See, e.g.*, Col. 9:4-5 (Claim 1). Dr. Clark explained the patent's requirement that the Web server route to a dispatcher:

> [MR. PATRAS] Q. Why do you contend that the request does not come from a dispatcher?
>
> [DR. CLARK] A. Because there's no separate dispatcher on the OHS machine.
>
> Q. Any other reason?
>
> A. Sure. I mean, if you don't have a dispatcher, you can't have dispatching. And the dispatcher has to be capable of being routed to from the Web server after the dynamic request is intercepted. It's not there.
>
> Q. So the reason why you believe that there is no dispatcher is because what Dr. Finkel has pointed to as the dispatcher is not on a separate machine from the Web server; is that correct?
>
> A. No. It's not separate from the Web server executable itself. You must have the capability to route to the dispatcher after there is an intercepting. And what you're pointing to as the dispatcher is part of the Web server executable, and you can't route to yourself. There is no routing.

A26, at pp. 252:24-253:18. The report of epicRealm's expert, Dr. Finkel, contains no mention of a separate dispatcher in Web Cache. EpicRealm's infringement theory asserts that Web Cache is both the Web server and the dispatcher of the patent, i.e., that the Web Cache is dispatching requests to itself. Clark Decl. ¶¶91-94. As Dr. Clark notes, this position contradicts both the plain language of the patent claims (which require routing a request *from the Web server to the dispatcher*) and common sense (if the Web server *is* the dispatcher, there would be no routing *to* the dispatcher). Clark Decl. ¶¶91-94; A26, at pp. 252:24-253:18.

Further, Dr. Clark's analysis of Web Cache source code confirms that the "Web server" functionality, *i.e.*, receiving a request and checking its cache, and the alleged "dispatching" functionality of sending the requests to the origin server, are all performed by fibers within the same Web Cache executable program. Clark Decl. ¶55. In the event of a cache miss, the Web Cache fiber routes the request to an OHS, part of the alleged page server—not to any Dispatcher. Based on the undisputed facts of Web Cache's code and functionality, it does not contain the

required "dispatcher" as a matter of law, and the Court should grant Oracle summary judgment of noninfringement for this reason. *General Mills*, 103 F.3d at 983.

b) Oracle Internet Application Server Lacks A "Dispatcher"

EpicRealm's final infringement contentions, and the report of its expert, Dr. Finkel, contain no mention of a dispatcher for Internet Application Server which is not merely part of the Web server, and thus fail for the same reasons as in Web Cache. EpicRealm asserts that OHS is the Web server and mod_oc4j is the dispatcher of the patent claims. However, Dr. Clark's source code analysis confirms—and epicRealm does not dispute—that mod_oc4j is part of the OHS executable. While a customer can choose whether to install or uninstall the mod_oc4j code, when the code is present, it is contained entirely within the OHS executable. Clark Decl. ¶¶134-136. The OHS process cannot be both the Web server and the dispatcher because it cannot "route" requests to itself. In the event of a request for Java-based content, the OHS process invokes mod_oc4j to route the request to an OC4J process, the alleged page server—not to any dispatcher. Without a dispatcher as required by the asserted claims, OHS does not infringe epicRealm's patents as matter of law, and the Court should grant Oracle summary judgment of noninfringement.

c) Oracle Database Lacks a "Dispatcher"

EpicRealm's infringement theory for Oracle Database identifies OHS as the alleged Web server and JDBC and OCI as the "dispatcher." *See* discussion of JDBC and OCI in Section III.B.3, *supra*. But this too must fail. Dr. Clark's analysis confirms that these APIs, when installed and used, are not separate executable processes. Clark Decl. ¶¶149-151. An API, or Application Programming Interface, is not a separate executable process: it is a set of standardized function calls that permit a software program to issue high-level commands through a library that is linked into the program. Clark Decl. ¶¶140-142. JDBC and OCI are simply a

library that permits OHS to issue commands to the Oracle Database. *Id.* There is software code located on both the OHS and the Database that implements this library, but that code resides in the OHS and the Database, not in any separate process. *Id.* Thus, JDBC and OCI cannot be "routed to" within the meaning of the asserted claims. Without a Dispatcher, OHS does not infringe epicRealm's patents as matter of law, and the Court should grant Oracle summary judgment of noninfringement.

d) Oracle Database Does Not "Dispatch" A Request From A "Dispatcher" To A "Page Server"

Regardless of whether JDBC or OCI could qualify as "dispatchers," Oracle Database does not "dispatch" requests because neither JDBC nor OCI "examine" or "analyze" a Web request, which is a required element of dispatching under both parties' proposed constructions. Oracle contends that "dispatching" is "analyzing a request to make an informed selection of which page server should process the request, and sending the request to that page server." EpicRealm's proposed construction for "dispatching" uses the phrase "examining a request to make an informed selection." Thus, the parties agree that "dispatching" requires the analyzing or examination of the dynamic Web page generation request of the patent. For this reason, Oracle Database does not practice the "dispatching" limitation of the asserted claims under either party's proposed constructions.

Dr. Clark reviewed the relevant source code sections relating to how JDBC and OCI forward requests to RAC instances. His analysis confirmed that neither JDBC nor OCI examines a request to make its decision. Clark Decl. ¶¶152-167. JDBC and OCI simply execute the appropriate connection method and a RAC instance is selected, irrespective of the content of the Web page request. *Id.* None of the methods of establishing a connection to a RAC instance require the passing of a dynamic Web page request. *Id.* Since the request is never passed to

JDBC or OCI, they are incapable of analyzing or examining the request to make an informed selection of which RAC instance with which to establish the connection. *Id.* Accordingly, JDBC and OCI lack the "analyzing" or "examining" of a request necessary to meet the "dispatching" limitation of the asserted claims as a matter of law and the Court should grant Oracle summary judgment for this additional reason.

4. Oracle's Use Of The Oracle Products To Operate Its Own Websites Does Not Directly Infringe

EpicRealm's expert, Dr. Finkel, opines that Oracle infringes the asserted claims by operating the Oracle.com Website because the system that runs the site allegedly uses Web Cache to load balance dynamic Web page requests across two or more origin servers. A27, at p. 10; A20, at pp. 54:9-60:7. First, for the reasons stated above in Section V.A(1)-(3), Oracle's use of Web Cache or any of the other accused products does not and cannot infringe the asserted claims. Second, Oracle's Website engineer, Mark Clark, never testified that Oracle.com used Web Cache to load balance requests across origin servers. EpicRealm has provided no evidence to support its assertion.

B. Oracle Does Not Indirectly Infringe

1. The Absence Of Direct Infringement Precludes Contributory Infringement And Inducement Of Infringement

As a matter of law, both contributory infringement and inducement of infringement require that direct infringement by another be proven as a predicate. *See* Section IV.C, *supra*. Oracle's accused products are incapable of direct infringement because they lack the "releasing," "intercepting," "dispatcher," and "dispatching" limitations, as discussed above. The use of the accused products, therefore, cannot directly infringe, and epicRealm's contributory infringement and inducement of infringement theories fail.

Even assuming, *arguendo*, that direct infringement is possible, epicRealm has not provided any evidence that Oracle's customers use the accused products in an allegedly infringing configuration. In their default configurations, none of the products infringe because they do not practice the dynamic load balancing on which epicRealm's infringement theories rely for the "dispatching" requirement of the claims. Neither of epicRealm's experts, Dr. Finkel or Mr. Wagner, investigated whether Oracle customers actually use this dynamic load balancing. Dr. Clark spoke with several Oracle engineers, who confirmed Dr. Clark's opinion that dynamic load balancing is very rarely implemented because its minimal benefits are almost always outweighed by the costs associated with the complexity of dynamic load balancing. A26, pp. 128:7-129:8; A21, p. 137:4-11. EpicRealm cannot simply assume that Oracle's customers stray from the default configuration and use dynamic load balancing, and they have provided no evidence of actual infringing use. Summary judgment of no contributory infringement and no inducement of infringement is therefore proper.

2. Oracle's Products Have Substantial Noninfringing Uses

EpicRealm cannot meet its burden of proving that Oracle's accused products have no substantial noninfringing uses. Oracle cannot, therefore, be liable for contributory infringement. *See* Section IV.C. Each of Oracle's accused products are capable of substantial non-infringing uses, as discussed in the Clark Declaration, ¶¶169-174. For example, the products all can be used without the "dynamic load balancing" on which epicRealm relies to meet the "dispatching" requirement of the claims. *Id.*

Further, customers can use both Internet Application Server and Oracle Database in various systems that do not perform the management of dynamic Web page requests such as handling requests for e-mail, handling of requests for USENET newsgroups, handling of requests

for static Web content, such as text, images, sound, and video, and the handling of requests by means of communications protocols other than HTTP. Clark Decl. ¶173.

Additionally, Oracle Database has numerous substantial uses in which the client does not request Web page content. Oracle Database is frequently employed for such uses, and none of these uses would infringe epicRealm' patents. Clark Decl. ¶174.

3. Oracle Did Not Have Notice Of The Patents Prior To 2006

In order to prevail on contributory infringement, epicRealm must show that Oracle had knowledge that the product is "especially made or especially adapted for use in an infringement of [a] patent." *Trell v. Marlee Elecs. Corp.*, 912 F.2d 1443 (Fed. Cir. 1990). In order to prevail on inducement of infringement, epicRealm must prove that Oracle "'knew or should have known [its] actions would induce actual infringements.'" *DSU*, 471 F.3d at 1306 (quoting *Manville*, 917 F.2d at 553). As discussed in Oracle's Motion for Summary Judgment of No Willful Infringement, Oracle did not have actual notice of epicRealm's patents until it became involved in the Safelite action in January 2006. Therefore, even in the event that epicRealm is able to meet the remaining elements of contributory infringement, that infringement can only run from January 2006.

C. Oracle Does Not Infringe Under The Doctrine Of Equivalents

1. EpicRealm Waived Any Right To Assert Infringement Under The Doctrine Of Equivalents

EpicRealm raised its doctrine of equivalents infringement theory for the first time in Dr. Finkel's *rebuttal* report on *invalidity*, after fact discovery and the service of Oracle's noninfringement expert report. EpicRealm neglected to offer any equivalents theory in its responses to Oracle's interrogatory regarding infringement theories, in its final infringement contentions, or in its expert's report on infringement. EpicRealm's use of its rebuttal report on

invalidity to introduce a new theory of *infringement* at this late stage is improper, and unduly prejudices Oracle. Oracle therefore asks this Court to preclude epicRealm from asserting its new theory of infringement under the Doctrine of Equivalents.

2. EpicRealm's Untimely Allegations Under The Doctrine Of Equivalents Fail As A Matter Of Law

While the Court should not consider epicRealm's untimely equivalents theory, it is insufficient as a matter of law in any event. The entirety of epicRealm's equivalents theory is contained in the last five pages of Dr. Finkel's rebuttal report on invalidity. Dr. Finkel opines that his theory under the doctrine of equivalents is based on his understanding that Oracle's invalidity expert, Dr. Shamos, offered new claim constructions that differed from those upon which the parties had heretofore relied. A28, ¶¶ 233-236. But even if Dr. Shamos offered new constructions in his report, which he did not, Oracle has not adopted or proposed those constructions. Moreover, Dr. Finkel conducted no analysis under the doctrine of equivalents. He merely recites conclusory statements that the function, way, and result achieved by the accused products is equivalent to those of the invention, without explaining why this is so. *Id.*, ¶¶ 238, 241. These conclusory opinions are insufficient to create a material factual dispute. *Network Commerce, Inc. v. Microsoft Corp.*, 422 F.3d 1353, 1363 (Fed. Cir. 2005) (conclusory statements by expert witness that are generalized and do not provide "particular testimony and linking argument on a limitation-by-limitation basis" are insufficient to defeat a motion for summary judgment of no infringement under the doctrine of equivalents). Dr. Finkel's opinion reads the "releasing" limitation out of the claims and improperly vitiates that element. *Panduit Corp. v. Hellermanntyton Corp.*, 451 F.3d 819, 830 (Fed. Cir. 2006). These conclusory opinions are insufficient to create a material factual dispute. Summary judgment that Oracle's products do not infringe epicRealm's patents under the Doctrine of Equivalents is therefore proper.

VI. CONCLUSION

The source code and other undisputed material facts demonstrate that epicRealm cannot meet its burden of showing that Oracle's accused products meet each and every limitation of the asserted claims under either party's claim constructions. Summary judgment of noninfringement is therefore proper and should be granted.

MORRIS, NICHOLS, ARSHT & TUNNELL LLP

/s/ James W. Parrett, Jr.

Mary B. Graham (#2256)
James W. Parrett, Jr. (#4292)
1201 North Market Street
P.O. Box 1347
Wilmington, DE 19899
mgraham@mnat.com
jparrett@mnat.com
(302) 658-9200

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

OF COUNSEL:

James G. Gilliland
Theodore T. Herhold
Joseph A. Greco
Robert J. Artuz
Eric M. Hutchins
TOWNSEND AND TOWNSEND AND CREW LLP
379 Lytton Avenue
Palo Alto, CA 94301
(650) 326-2400

Dorian Daley
Peggy E. Bruggman
Matthew M. Sarboraria
ORACLE CORPORATION
ORACLE U.S.A., INC.
500 Oracle Parkway
Redwood Shores, CA 94065

2432451

61439009 v2

CERTIFICATE OF SERVICE

I hereby certify that on July 31, 2008, I caused the foregoing to be electronically filed with the Clerk of the Court using CM/ECF which will send electronic notification of such filing to the following:

Richard L. Horwitz
David Ellis Moore
POTTER ANDERSON & CORROON, LLP

Additionally, I hereby certify that true and correct copies of the foregoing were caused to be served on July 31, 2008 upon the following individuals in the manner indicated:

BY E-MAIL AND HAND DELIVERY

Richard L. Horwitz
David Ellis Moore
POTTER ANDERSON & CORROON, LLP
1313 N. Market St., Hercules Plaza, 6th Floor
P.O. Box 951
Wilmington, DE 19899-0951

rhorwitz@potteranderson.com
dmoore@potteranderson.com

BY E-MAIL

George S. Bosy
JENNER & BLOCK
330 N. Wabash Avenue
Chicago, IL 60611-7603

gbosy@jenner.com

/s/ James W. Parrett, Jr.
James W. Parrett, Jr. (#4292)

61439009 v2