I hereby certify that this correspondence is being filed using EFS-Web addressed to: Mail Stop Inter Partes Reexam, Central Reexamination Unit, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the date shown below.

Dated: March 1, 2011 Signature: / Robert T. Neufeld /
Patent Attorney & Reg. No. 48,394

Docket No. 13557.105125
(PATENT)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Reexamination of:

Fresko

Patent No.: 7,426,720

Issue Date: September 16, 2008

For: SYSTEM AND METHOD FOR DYNAMIC PRELOADING OF CLASSES THROUGH MEMORY SPACE CLONING OF A MASTER RUNTIME SYSTEM PROCESS

Control No.: Not Yet Assigned

Examiner: Not Yet Assigned

Art Unit: Not Yet Assigned

REQUEST FOR *INTER PARTES* REEXAMINATION UNDER 37 C.F.R. § 1.915

Mail Stop Inter Partes Reexam
Attn: Central Reexamination Unit
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

Google Inc. (hereinafter, "Requester") submits, under the provisions of 37 C.F.R. § 1.902 *et seq.*, a Request for Reexamination (hereinafter, "Request") of claims 1-8, 10-17, and 19-22 of U.S. Patent No. 7,426,720 (hereinafter "the '720 patent") entitled "System and Method for Dynamic Preloading of Classes through Memory Space Cloning of a Master Runtime System Process," issued to Fresko on Sept. 16, 2008. The '720 patent is provided as Exhibit 1 to the Request.

In support of its request, Requester provides the following:

- The $8,800.00 fee for requesting *inter partes* reexamination set forth in 37 C.F.R. § 1.20(c)(2) (37 C.F.R. § 1.915(a));

- An identification of the patent by patent number and every claim for which reexamination is requested (37 C.F.R. § 1.915(b)(1));

- A citation of the patents and printed publications which are presented to provide a substantial new question of patentability (37 C.F.R. § 1.915(b)(2));

- A statement pointing out each substantial new question of patentability based on the cited patents and printed publications, and a detailed explanation of the pertinency and manner of applying the patents and printed publications to every claim for which reexamination is requested (37 C.F.R. § 1.915(b)(3));

- A copy of every patent or printed publication relied upon or referred to in paragraphs (b)(1) through (3) of 37 C.F.R. §1.915, accompanied by an English language translation of all the necessary and pertinent parts of any non-English language document (37 C.F.R. § 1.915(b)(4));

- A copy of the entire patent including the front face, drawings, and specification/claims (in double column format) for which reexamination is requested, and a copy of any disclaimer, certificate of correction, or reexamination certificate issued in the patent. All copies must have each page plainly written on only one side of a sheet of paper (37 C.F.R. § 1.915(b)(5));

- A certification by the third party requester that a copy of the request has been served in its entirety on the patent owner at the address provided for in 37 C.F.R. § 1.33(c). The name and address of the party served must be indicated. If service was not possible, a duplicate copy of the request must be supplied to the Office (37 C.F.R. § 1.915(b)(6));

- A certification by the third party requester that the estoppel provisions of 37 C.F.R. § 1.907 do not prohibit the inter partes reexamination (37 C.F.R. § 1.915(b)(7)); and

- A statement identifying the real party in interest to the extent necessary for a subsequent person filing an inter partes reexamination request to determine whether that person is a privy (37 C.F.R. § 1.915(b)(8)).

Pursuant to 35 U.S.C. § 303, the prior art references discussed in this Request raise "substantial new questions of patentability" with respect to claims 1-8, 10-17, and 19-22 of the '720 patent.

Pursuant to 37 C.F.R. §§ 1.915(b)(7) and (b)(8), Requester identifies the real party in interest to this third-party request as Google Inc. The Requester and real party in interest, Google Inc., further certifies that the estoppel provisions of 37 C.F.R. § 1.907 do not prohibit this *inter partes* reexamination.

TABLE OF CONTENTS

I. INTRODUCTION

Requester seeks reexamination of claims 1-8, 10-17, and 19-22 of the '720 patent (Exhibit 1) under 35 U.S.C. §§ 311-318 and 37 C.F.R. § 1.903 *et seq*. The application for the '720 patent was filed on December 22, 2003. The '720 patent is assigned to Sun Microsystems, Inc.

Substantial new questions of patentability exist with respect to claims 1-8, 10-17 and 19-22 of the '720 patent based on references and combinations laid out in detail below and not before the Patent Office during the original examination. The '720 patent is generally directed to a method and system for the "dynamic preloading of classes through memory space cloning of a master runtime system process." *See* '720 patent, Claim 1. The claims of the '720 patent were allowed as a result of 1) amendments made by the Applicant adding the limitation of "copy-on-write" to the claims after a final rejection and 2) the Applicant's associated arguments relying on this "copy-on-write" limitation to distinguish the Application from the prior art of record. *See* Letter to Examiner Junchun Wu (Dec. 11, 2007) (attached as Exhibit 2); Notice of Allowability (May 20, 2008) at 3 (attached as Exhibit 3).

In particular, the Applicant argued that Webb and Kuck, the then-applied prior art of record, failed to disclose "reducing memory usage for virtual machines by using copy-on-write cloning."

However this very "copy-on-write" limitation and all the other limitations of the allowed claims are disclosed in either an individual reference or a combination of references that were either not before or were not considered by the Examiner at the time of allowance. These references form the specific proposed rejections of this Request. Accordingly, this Request raises substantial new questions of patentability with respect to claims 1-8, 10-17 and 19-22 of the '720 patent.

Briefly, the specification of the '720 patent acknowledges that a well-known process cloning mechanism in the prior art was the fork( ) system call of the Unix and/or Linux operating systems: "an example of a process cloning mechanism suitable for use in the present invention is the fork () system call provided by the Unix or Linux operating systems, such as described in [the prior art reference] M. J. Bach, 'The Design of the Unix Operating System,' Ch. 7, Bell Tele. Labs., Inc. (1986)." *See* '720 patent at 4:54-5:6 (citing Chapter 7 of the APA-Bach reference). The '720 patent then describes a "copy on-write variant" of this fork( ) system call (as noted above, the claims were allowed as a result of amendments adding the limitation of "copy-on-write" after a final rejection): "[i]n a copy-on-write variant of the fork( ) system call, the operating system only copies references to the memory space and defers actually copying individual memory space segments until, and if, the child process attempts to modify the referenced data of the parent process context." *See id.* at 4:66-5:3.

Although the '720 patent incorporates by reference **only** Chapters 2 and 7 of the APA-Bach reference, which the Applicant relied on for discussion of the fork( ) system call. The APA-Bach reference was not listed in any Information Disclosure Statement nor otherwise considered by the Examiner during prosecution. Thus, the Examiner was not aware of the disclosure in Chapter 9 of the APA-Bach reference, which explicitly discloses the use of copy-on-write with the fork( ) system call:

> As explained in Section 7.1, the kernel duplicates every region of the parent process during the *fork* system call and attaches it to the child process. Traditionally, the kernel of a swapping system makes a physical copy of the parent's address space, usually a wasteful operation, because processes often call *exec* soon after the *fork* call and immediately free the memory just copied. On the System V paging system, the kernel avoids copying the page by manipulating the region tables, page table entries, and pfdata table entries: It simply increments the region reference count of shared regions . . . The page can now be referenced through both regions, which share the page until a process writes to it. The kernel then copies the page so that each region has a private version. **To do this, the**

> **kernel turns on the 'copy on write' bit for every page table entry in private regions of the parent and child processes during *fork*. If either process writes the page, it incurs a protection fault, and in handling the fault, the kernel makes a new copy of the page for the faulting process. The physical copying of the page is thus deferred until a process really needs it.**

See APA-Bach at 289-290 (emphasis added). Thus the prior art of record in view of APA-Bach fully discloses the claimed subject matter including copy-on-write process cloning—i.e., the alleged novelty of the claimed invention of the '720 patent.

In other words, Chapter 9 of the APA-Bach was not before the Examiner but is material to patentability. The Webb in view of Kuck rejection that the Examiner had already developed combined with Chapter 9 of the APA-Bach reference renders invalid each of the claims of the '720 patent, regardless of the Applicant's amendment to add only the copy-on-write limitation to the '720 patent claims. Because the APA-Bach reference clearly discloses the copy-on-write limitation with respect to process cloning, and because this limitation was the basis for the allowance of the '720 patent, the prior art of record (i.e., the Webb and Kuck references) in view of APA-Bach poses a substantial new question of patentability.

The Webb and Kuck in view of APA-Bach combination flows directly from the prosecution history. The Examiner's developed arguments strongly suggest that, were the Examiner aware of the disclosure of the APA-Bach reference, he would not have allowed the '720 patent to issue. But the APA-Bach reference is merely the first among many references disclosing the use of a copy-on-write process cloning mechanism, not only broadly in the context of virtual machines, but also specifically in the context of Java virtual machines.

For example, the Dike prior art reference anticipates each element of the '720 patent, including the copy-on-write limitation. Dike describes dynamic preloading of classes through memory space cloning of a master runtime process. *See e.g.*, Dike at §§ 2.2, 2.3. More

specifically, Dike describes the creation of a Linux virtual machine that runs on Linux, wherein the virtual machine systems are all cloned, using the fork( ) system call, onto the host system. Since all of the systems need access to the kernel data, Dike employs a shared data state between the processes, disclosing a copy-on-write mechanism to allow all of the processes to access the kernel data.

In other words, 1) Dike explicitly discloses that the underlying cloning process is the Linux fork( ) call, and 2) as explicitly referenced in Dike the Linux fork( ) cloning is optimized using copy-on-write. Thus, Dike explicitly discloses use of the fork( ) system call and the use of a copy-on-write process cloning mechanism, as well as all of the other elements of claims 1-8, 10-17 and 19-22 of the '720 patent.

Furthermore, a person of ordinary skill in the art reviewing Dike[1] would have recognized that the Linux fork( ) cloning is optimized using copy-on-write. This optimization is described in, e.g., the Steinberg prior art reference, which states that "[t]he created child process inherits copies of the parent process data space, heap and stack, which are copied on demand using a copy-on-write mechanism." *See* Steinberg at 21. Moreover, if one believes that Dike does not anticipate claims 1-8, 10-17 and 19-22 of the 720 patent, Dike in view of Steinberg teaches all the elements of these claims and thus render these claims unpatentable under 35 U.S.C § 103. One of ordinary skill in the art reading Dike's teaching of process cloning would have combined the teaching of Dike with the copy-on-write mechanism explicitly described in Steinberg because, among other reasons, copy-on-write is explicitly referenced in Dike as the default mechanism.

[1] *See* M.P.E.P. § 2131.01(III) (An extra reference or evidence can be used in an anticipation rejection to show an inherent characteristic of the concept taught by the primary reference).

In addition, the Bryant prior art reference discloses a method and apparatus useful for increasing the performance of Java programs by executing a fork command and creating a child java machine. Bryant discloses a Java server that "invokes the Java virtual machine and preloads all potentially needed object files during initialization of the Java virtual machine to speed up the actual execution of a particular Java application. The Java server accomplishes the execution of a particular Java application by forking itself and then having the child Java server run the Java class files in the already loaded Java virtual machine for the specific Java CGI-BIN script." *See* Bryant at 2:46-48. Bryant, in combination with APA-Bach's disclosure of the copy-on-write mechanism, teaches all the elements of claims 1-8, 10-17, and 19-22 of the '720 patent, including the purported point of novelty of reducing memory usage for virtual machines by using copy-on-write cloning. Likewise, the Traut prior art reference discloses the use of a copy-on-write process cloning mechanism while forking a virtual machine; the combination of Bryant in view of Traut therefore also poses a substantial new question of patentability. Furthermore, the combination of APA-Bach's copy-on-write disclosure with the object-oriented programming system of the Srinivasan reference, which explicitly uses fork( ) to spawn a worker process, poses a substantial new question of patentability.

Similarly, the Sexton prior art reference is directed to reducing memory usage for virtual machines by providing read-only access to shared data. *See* Sexton at 8:41-64. Sexton cites the Bugnion reference and the Johnson reference. The Bugnion reference discloses a Virtual Machine Monitor ("VMM") layer that "maintains copy-on-write disks." The Johnson reference "introduces copy on write storage." Thus Sexton in view of Bugnion or Johnson also poses a substantial new question of patentability.

Requester has identified at least eleven (11) prior art patents and printed publications that individually anticipate or, in combination, render obvious each of claims 1-8, 10-17, and 19-22 of the '720 patent and thus raise substantial new questions of patentability with respect to the '720 patent. The prior art patents and printed publications were either not cited to, nor considered by, the Examiner during prosecution of the '720 patent, or they were not considered in combination with the prior art patents and printed publications currently submitted, and they are not cumulative of information cited to, or considered by, the Examiner during prosecution of the '720 patent. These prior art patents and printed publications anticipate or render obvious each element of the '720 patent, including the purportedly novel (but in reality well-known) use of "copy-on-write" that was the basis for the '720 patent's allowability.

Accordingly, in view of these listed prior art references, Requester respectfully requests the issuance of an order for reexamination. Requester further respectfully requests that the Director provide an order of action dates to accompany the decision ordering reexamination of the '720 patent.

II. STATEMENT UNDER 37 C.F.R. § 1.915 (B)(3) POINTING OUT SUBSTANTIAL NEW QUESTIONS OF PATENTABILITY

The new prior art references not previously considered by the Examiner raise substantial new questions of the patentability of claims 1-8, 10-17, and 19-22 of the '720 patent. Section II.A provides an overview of the '720 patent and an interpretation of certain claim limitations of the '720 patent. Section II.B summarizes certain aspects of the law regarding reexamination. Section II.C summarizes the evidentiary standards applicable to reexamination. Section II.D provides a list of all prior art patents and printed publications relied upon in this Request. Section II.E provides a list of other supporting documents discussed in this Request. Section II.F provides a summary of pending litigation involving the '720 patent. Section II.G provides an

identification of the substantial new questions of patentability raised in this Request. Section II.H provides an overview of the substantial new questions of patentability raised in this Request.

A. Overview of the '720 Patent

The claims of the '720 patent relate to a method and system for the "dynamic preloading of classes through memory space cloning of a master runtime system process" relying in all the claims on a copy-on-write process cloning mechanism. *See* '720 patent, Claims 1, 10, and 20. Claims 1, 10, and 20 are independent claims.

Claim 1 recites:

> A system for dynamic preloading of classes through memory space cloning of a master runtime system process, comprising:
>
> A processor; A memory
>
> a class preloader to obtain a representation of at least one class from a source definition provided as object oriented program code;
>
> a master runtime system process to interpret and to instantiate the representation as a class definition in a memory space of the master runtime system process;
>
> a runtime environment to clone the memory space as a child runtime system process responsive to a process request and to execute the child runtime system process; and
>
> a copy on write process cloning mechanism to instantiate the child runtime system process by copying references to the memory space of the master runtime system process into a separate memory space for the child runtime system process, and to defer copying of the memory space of the master runtime system process until the child runtime system process needs to modify the referenced memory space of the master runtime system process.

Claim 10 recites:

> A method for dynamic preloading of classes through memory space cloning of a master runtime system process, comprising:
>
> executing a master runtime system process;
>
> obtaining a representation of at least one class from a source definition provided as object oriented program code;
>
> interpreting and instantiating the representation as a class definition in a memory space of the master runtime system process;
>
> and cloning the memory space as a child runtime system process responsive to a process request and executing the child runtime system process;
>
> wherein cloning the memory space as a child runtime system process involves instantiating the child runtime system process by copying references to the memory space of the master runtime system process into a separate memory space for the child runtime system process;
>
> and wherein copying references to the memory space of the master runtime system process defers copying of the memory space of the master runtime system process until the child runtime system process needs to modify the referenced memory space of the master runtime system process.

And Claim 20 recites:

> An apparatus for dynamic preloading of classes through memory space cloning of a master runtime system process, comprising:
>
> A processor; A memory means for executing a master runtime system process;
>
> means for obtaining a representation of at least one class from a source definition provided as object oriented program code;
>
> means for interpreting and means for instantiating the representation as a class definition in a memory space of the master runtime system process;
>
> and means for cloning the memory space as a child runtime system process responsive to a process request and means for executing the child runtime system process;
>
> wherein the means for cloning the memory space is configured to clone the memory space of a child runtime system process using a copy on write process cloning mechanism that instantiates the child runtime system process by copying references to the memory space of the master runtime system process into a separate memory space for the child runtime system process and that defers copying of the memory space of the master runtime system process until the child runtime system process needs to modify the referenced memory space of the master runtime system process.

The '720 patent recites the cloning of a master runtime process into one or more child runtime processes to support the spawning of multiple and independent isolated user

applications. *See* '720 patent 2:44-48. Each of the independent claims of the '720 patent rely on a copy-on-write process cloning mechanism to instantiate the child runtime system process. *See id.* at Claims 1, 10, and 20. The '720 patent describes the process cloning mechanism by incorporating by reference Chapter 7 of M.J. Bach's "The Design of The Unix Operating System," which describes the Unix Operating System fork( ) system call. *See id.* at 4:53-58. When implementing the fork( ) system call, use of the copy-on-write process allows for the operating system to create a copy of the context of the parent process into the memory space of the child process by copying only references to the memory space, deferring the actual copying of the individual memory space segments until, and if, the child process attempts to modify the referenced data of the parent process context. *See id.* at 4:63-5:3. The '720 patent lists the reduced impact on system memory as a primary benefit of the combination of a fork( ) system call with copy-on-write. *See* '720 patent at 3:21-40 (indicating that deferring copying by way of the copy-on-write system "avoids impacting application performance until, and if, the segment is required").

The '720 patent issued from U.S. Patent Application Serial No. 10/745,023, filed on December 22, 2003. The United States Patent & Trademark Office ("USPTO") issued a non-final rejection on April 27, 2007, then a final rejection on October 18, 2007. In both instances all claims of the application were rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 6,823,509 to Webb (hereinafter "Webb") in view of U.S. Application 2003/0088604 to Kuck, et al. (hereinafter "Kuck"). The Examiner found that it would have been obvious to one of ordinary skill in the art at the time the invention was made to modify Webb's teaching by adding a runtime environment to clone a parent memory space as a child runtime process responsive to a process request and to execute the child runtime process as taught by Kuck. *See*

Non-Final Rejection (Apr. 27, 2007) at 5 (attached as Exhibit 4). In response to these rejections, and in conjunction with the Applicant's filing of a Request for Continued Examination, the Applicant added the copy-on-write limitation to all independent claims. *See* Amendment after Final Rejection (Dec. 18, 2007) (attached as Exhibit 5). Specifically, Claim 1 and Claim 24 (issued as Claim 20) were amended to include a "copy-on-write process cloning mechanism." *See id.* Similarly, Claim 12 (issued as Claim 10) was amended to include a limitation such that "copying references to the memory space of the master runtime system process defers copying of the memory space of the master runtime system process until the child runtime system process needs to modify the referenced memory space of the master runtime system process." *See id.* Thus, each and every claim of the '720 patent was amended to include—and presently contains—a copy-on-write limitation.

In their letter to, and interview with, the Examiner, Applicants relied on the "copy-on-write" cloning mechanism to distinguish their Application from the prior art, including Webb and Kuck. *See* Letter to Examiner Junchun Wu. The Examiner agreed, noting that

> "the prior art of record fails to teach and/or suggest a runtime environment to clone the memory space as a child runtime system process responsive to a process request and to execute the child runtime system process and a copy-on-write process cloning mechanism to instantiate the child runtime system process by copying references to the memory space of the master runtime system process into a separate memory space for the child runtime system process, and to defer copying of the memory space of the master runtime system process until the child runtime system process needs to modify the referenced memory space of the master runtime system process as recited in the independent claims."

Notice of Allowability at 3. Thus, it is clear from the record that the patentability of the '720 patent hinged on the purported novelty of the copy-on-write limitation. As discussed below, however, application of a copy-on-write mechanism to the standard Unix or Linux fork( ) system call was well-known in the art prior to the filing of the application for the '720 patent. Indeed,

copy-on-write fork( ) was explicitly disclosed in the APA-Bach reference cited in the specification of the '720 patent – a disclosure that the Applicant failed to bring to the Examiner's attention during the prosecution of the '720 patent.

B. Aspects of the law governing reexamination

1. Citation of prior art

Any person at any time may file a request for reexamination by the Office of any claim of any patent on the basis of any prior art cited under the provisions of section 301." 35 U.S.C. § 302. Section 301 limits prior art to "patents or printed publications." 35 U.S.C. § 301.

MPEP 2128 classifies a reference as a printed publication if it is accessible to the public:

> A reference is proven to be a 'printed publication' 'upon a satisfactory showing that such *document* has been disseminated or otherwise made available to the extent that persons interested and ordinarily skilled in the subject matter or art, exercising reasonable diligence, can locate it.'

In re Wyer, 655 F.2d 221, 210 USPQ 790 (C.C.P.A. 1981) (quoting *I.C.E. Corp. v. Armco Steel Corp.*, 250 F. Supp. 738, 743, 148 USPQ 537, 540 (S.D.N.Y. 1966)).

2. "Old" prior art can raise a significant new question of patentability

The fact that a prior art reference was cited or even previously considered by an examiner does not preclude use of that reference to find a substantial new question of patentability. *See* 35 U.S.C. § 303(a); MPEP Section 2258.01; *see also In re Swanson*, 540 F.3d 1368, 1380-81 (Fed. Cir. 2008) (holding that consideration of a prior art reference in previous litigation and in an original examination does not preclude a finding of a SNQ based on the same prior art reference in reexamination).

A combination of such "old art" and art newly cited during the reexamination proceeding may raise a SNQ. *See* MPEP Section 2258.01. The Patent Office may even find a SNQ based exclusively on previously cited references.

> For example, a SNQ may be based solely on old art where the old art is being presented/viewed in a new light, or in a different way, as compared with its use in the earlier concluded examination(s), in view of a material new argument or interpretation presented in the request.

See id.

3. Obviousness standard under *KSR*

The Supreme Court recently relaxed the Federal Circuit's requirement of a "teaching/suggestion/motivation test," and instead held that "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." *KSR Int'l Co. v. Teleflex Inc. et al.*, 550 U.S. 398, 416 (2007). The Court noted that "[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation" of an existing system, then "§103(a) likely bars its patentability." *Id.* at 417. *KSR* also held that "if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious" if within his or her skill. *See id.*

On October 10, 2007, the U.S. Patent and Trademark Office ("USPTO") released Examination Guidelines for Determining Obviousness Under 35 U.S.C. 103(a) in View of the Supreme Court Decision in *KSR Int'l Co. v. Teleflex Inc.*, 72 Fed. Reg. 195 at 57526 (the "PTO Guidelines"). The PTO Guidelines adopt the rationales from the *KSR* decision for determining obviousness. One of the rationales is "'Obvious to Try' – Choosing from a Finite Number of Identified, Predictable Solutions, With a Reasonable Expectation of Success." To reject a claim on this basis, the PTO Guidelines note that pertinent factors to consider are whether "there had

been a finite number of identified, predictable potential solutions to the recognized need or problem," and "one of ordinary skill in the art could have pursued the known potential solutions with a reasonable expectation of success." *Id.* at 57532. The PTO Guidelines have been incorporated into the MPEP's examination guidelines for determining obviousness under 35 U.S.C. § 103. *See* MPEP 2141.

Additionally, the Federal Circuit has applied the *KSR* obviousness standard to combine multiple embodiments disclosed in a single prior art reference. *Boston Sci. Scimed, Inc. v. Cordis Corp.*, No. 2008-1073, 2009 U.S. App. LEXIS 588, at *24 (Fed. Cir. Jan. 15, 2009) (holding that a person of ordinary skill would have been motivated to combine one embodiment found in a patent reference with a second, separate embodiment found in the same patent reference).

4. Prior art references need not be enabling in an obviousness inquiry

Moreover, prior art references need not be enabling in the context of an obviousness inquiry. As stated in the MPEP:

> **35 U.S.C. 103(a) REJECTIONS AND USE OF INOPERATIVE PRIOR ART**
>
> "Even if a reference discloses an inoperative device, it is prior art for all that it teaches." *Beckman Instruments v. LKB Produkter AB*, 892 F.2d 1547, 1551, 13 USPQ2d 1301, 1304 (Fed. Cir. 1989). Therefore, "a non-enabling reference may qualify as prior art for the purpose of determining obviousness under 35 U.S.C. 103." *Symbol Techs. Inc. v. Opticon Inc.*, 935 F.2d 1569, 1578, 19 USPQ2d 1241, 1247 (Fed. Cir. 1991).

MPEP 2121.01; *see also* MPEP 2145; *Amgen Inc. v. Hoechst Marion Roussel, Inc.*, 314 F.3d 1313, 1357 (Fed. Cir. 2003) (holding that under 35 U.S.C. § 103, "a reference need not be enabled; it qualifies as prior art, regardless, for whatever is disclosed therein.") (citations to other cases omitted).

5. Claims of the patent are to be broadly construed

In a reexamination proceeding, claims are to be given their broadest construction consistent with the specification. *See In re Icon Health & Fitness, Inc.*, 496 F.3d 1374, 1379 (Fed. Cir. 2007) ("During reexamination, as with original examination, the PTO must give claims their broadest reasonable construction consistent with the specification.").

C. Evidentiary standards

If the prior art patents and printed publications raise a substantial question of patentability of at least one claim of the patent, then a substantial new question of patentability is present. *See* MPEP 2242. A prior art patent or printed publication raises a substantial question of patentability where there is a substantial likelihood that a reasonable examiner would consider the prior art patent or printed publication important in deciding whether or not the claim is patentable. *See id.*

D. Prior art patents and printed publications relied upon in this Request

In accordance with 37 C.F.R. § 1.915(b)(2), reexamination of claims 1-8, 10-17, and 19-22 of the '720 patent is requested in view of the prior art patents and printed publications listed below, which raise substantial new questions of patentability. This Request will demonstrate how claims 1-8, 10-17, and 19-22 of the '720 patent are anticipated and/or rendered obvious in view of the following prior art references:

1. U.S. Patent No. 6,823,509 to Webb, entitled "Virtual Machine with Reinitialization," published on Dec. 12, 2001, issued on Nov. 23, 2004 ("Webb"), provided as Exhibit 6.

2. U.S. Patent Publication No. 2003/0088604 A1 to Kuck, entitled "Process Attachable Virtual Machines," published on May 8, 2003 (hereinafter "Kuck"), provided as Exhibit 7.

3. M. J. Bach, The Design of the Unix Operating System, Bell Telephone Labs., Inc. (1986); excerpts ("APA-Bach"), provided as Exhibit 8.

4. Jeff Dike, "A user-mode port of the linux kernel," Proceeding ALS'00 Proceedings of the 4th annual Linux Showcase & Conference - Volume 4, USENIX Association Berkeley, CA, USA (2000) ("Dike"), provided as Exhibit 9.

5. Udo Steinberg, "Fiasco μ-Kernel User-Mode Port," Dresden University of Technology, Institute of System Architecture (Dec. 19, 2002) ("Steinberg"), provided as Exhibit 10.

6. U.S. Patent No. 6,405,367 to Bryant et al., entitled "Apparatus and Method for Increasing the Performance of Java Programs Running on a Server," issued on June 11, 2002 ("Bryant"), provided as Exhibit 11.

7. U.S. Patent Application Pub. No. 2004/001787 to Traut et al., entitled "Method for Forking or Migrating a Virtual Machine," filed on Jul. 11, 2002, published on Jan. 15, 2004, and issued as a United States Patent on Dec. 25, 2007 ("Traut"), provided as Exhibit 12.

8. Sriram Srinivasan, Advanced Perl Programming, O'Reilly & Associates, Inc. (August 1997) ("Srinivasan"), provided as Exhibit 13.

9. U.S. Patent No. 6,854,114 to Sexton, entitled "Using a virtual machine instance as the basic unit of user execution in a server environment," filed on Feb. 25, 2000, issued on Feb. 8, 2005 ("Sexton"), provided as Exhibit 14.

10. U.S. Patent No. 6,075,938 to Bugnion et al., entitled "Virtual Machine Monitors for Scalable Multiprocessors," filed on June 10, 1998, issued on June 13, 2000 ("Bugnion"), provided as Exhibit 15.

11. U.S. Patent No. 6,330,709 to Johnson et al., entitled "Virtual Machine Implementation for Shared Persistent Objects," filed on Feb. 25, 2000 , issued on Dec. 11, 2001 ("Johnson"), provided as Exhibit 16.

E. Supporting documents discussed in this Request

The following documents are provided to assist the Examiner in understanding the Request, including claim charts and references providing background information:

1. Claim Chart based on Webb and Kuck in view of APA-Bach, provided as Exhibit 17.

2. Claim Chart based on Dike in view of Steinberg, provided as Exhibit 18.

3. Claim Chart based on Bryant in view of APA-Bach, provided as Exhibit 19.

4. Claim Chart based on Bryant in view of Traut, provided as Exhibit 20.

5. Claim Chart based on Srinivasan in view of APA-Bach, provided as Exhibit 21.

6. Claim Chart based on Sexton in view of Bugnion, provided as Exhibit 22.

7. Claim Chart based on Sexton in view of Johnson, provided as Exhibit 23.

F. Current Litigation

The Requester is aware of at least one current litigation matter involving the '720 patent. On August 12, 2010, Oracle America, Inc. filed a complaint in the U.S. District Court for the Northern District of California alleging that Google, Inc. infringed the '720 patent. The case is styled *Oracle America, Inc. v. Google, Inc.*, Civil Action No.: 3:10-cv-03561 WHA. A Joint Case Management Statement for the case, dated November 18, 2010, provides for a claim construction hearing in the case to take place on April 20, 2011. Non-expert discovery will end on July 29, 2011. The deadline for filing dispositive motions is September 8, 2011.

G. Identification of Substantial New Questions of Patentability

In this Request, substantial new questions of patentability for claims 1-8, 10-17, and 19-22 of the '720 patent are identified in accordance with 37 C.F.R. § 1.915(b)(3) as follows:

1. Obviousness under 35 U.S.C. § 103(a) based on the Webb, Kuck, and APA-Bach references.

    a. Claims 1-8, 10-17, and 19-22 are unpatentable under 35 U.S.C. § 103(a) as rendered obvious by Webb in view of Kuck and further in view of APA-Bach.

2. Anticipation under 35 U.S.C. § 102(b) based on the Dike reference in view of Steinberg.

    a. Claims 1-8, 10-17, and 19-22 are unpatentable under 35 U.S.C. § 102(b) as anticipated by Dike in view of Steinberg.

3. Obviousness under 35 U.S.C. § 103(a) based on the Dike reference in view of Steinberg.

a. Claims 1-8, 10-17, and 19-22 are unpatentable under 35 U.S.C. § 103(a) as rendered obvious by Dike in view of Steinberg.

4. Obviousness under 35 U.S.C. § 103(a) based on the Bryant and APA-Bach references.

   a. Claims 1-8, 10-17, and 19-22 are unpatentable under 35 U.S.C. § 103(a) as rendered obvious by Bryant in view of APA-Bach.

5. Obviousness under 35 U.S.C. § 103(a) based on the Bryant and Traut references.

   a. Claims 1-8, 10-17, and 19-22 are unpatentable under 35 U.S.C. § 103(a) as rendered obvious by Bryant in view of Traut.

6. Obviousness under 35 U.S.C. § 103(a) based on the Srinivasan and APA-Bach references.

   a. Claims 1-8, 10-17, and 19-22 are unpatentable under 35 U.S.C. § 103(a) as rendered obvious by Srinivasan in view of APA-Bach.

7. Obviousness under 35 U.S.C. § 103(a) based on the Sexton and Bugnion references.

   a. Claims 1-8, 10-17, and 19-22 are unpatentable under 35 U.S.C. § 103(a) as rendered obvious by Sexton in view of Bugnion.

8. Obviousness under 35 U.S.C. § 103(a) based on the Sexton and Johnson references.

   a. Claims 1-8, 10-17, and 19-22 are unpatentable under 35 U.S.C. § 103(a) as rendered obvious by Sexton in view of Johnson.

H. Overview of Substantial New Questions of Patentability

Webb in view of Kuck and further in view of APA-Bach

The Examiner allowed the '720 patent to issue because, in his view, the combination of Webb and Kuck did not disclose the copy-on-write limitation that the Applicant added to each and every claim of the '720 patent. The APA-Bach reference, incorporated by reference in the '720 patent but neither included in an Information Disclosure Statement nor considered by the Examiner during prosecution, discloses the copy-on-write limitation and therefore poses a substantial new question of patentability.

Both Webb and Kuck are prior art to the '720 patent, consistent with the Examiner's findings during the prosecution of the '720 patent. *See* Non-Final Rejection at 3. The APA-Bach reference was published in 1986, and is prior art to the '720 patent under 35 U.S.C. § 102(b), given a priority date for the '720 patent of December 22, 2003. The APA-Bach reference was not considered by the Patent Office during the prosecution of the application that matured into the '720 patent, and therefore the combination of Webb, Kuck, and APA-Bach is not cumulative to the prior art considered by the Patent Office during the prosecution of the '720 patent.

As the Examiner discussed, Webb discloses a computer system including a memory and a processor. *See* Webb at 3:56-4:2. Webb implements a class pre-loader: "[a]t run-time objects are created as instantiations of these class files, and indeed the class files themselves are effectively loaded as objects." *See* Webb at 1:22–38. Webb's disclosure of a master runtime system process to interpret and to instantiate the representation as a class definition in a memory space of the master runtime system process is clear in Fig. 3, which Webb explains is the "initialization of a loaded class (step 350), which represents calling the static initialization method (or methods) of the class . . . this application must be performed once and only once before the first active use of a class" which then allows for "[t]he new application class [to load] the application classes into the JVM (step 410), and involves the steps shown in Fig. 3 for all the application classes." *See* Webb at 6:59-65 & 7:34-39.

Next, the Examiner relied on Kuck to disclose a runtime environment to clone the memory space as a child runtime system process responsive to a process request and to execute the child runtime system process. *See* Kuck at ¶¶ [0064]-[0065]. Thus the Examiner found that all of the claim elements of independent Claim 1, which were identical to the issued claim 1 save

for the addition of the copy-on-write limitation, were obvious under Webb in view of Kuck; the Examiner applied this same reasoning to the remaining independent claims. *See id.* at 6 & 10. The Examiner briefly addressed the copy-on-write limitation as recited in dependent claim 7, but did not cite the APA-Bach reference. *See id.* at 7. Later, upon the amendment appending the copy-on-write limitation to all independent claims, the Examiner allowed the patent to issue because "the prior art of record fails to teach and/or suggest . . . a copy-on-write process cloning mechanism . . . as recited in [the] independent claims." *See* Notice of Allowability at 2-3.

But the Examiner's rationale for allowance, based on the Applicant's arguments and representations to the United States Patent and Trademark Office, cannot comport with the disclosure of APA-Bach. APA-Bach or, more particularly, Chapter 7 of APA-Bach, was incorporated by reference within the Application but was neither disclosed in an Information Disclosure Statement nor considered by the Examiner during prosecution. One of ordinary skill in the art at the time of the invention, in possession of the APA-Bach reference, would need merely turn ahead to Chapter 9, where APA-Bach clearly and conspicuously discloses the exact copy-on-write limitation claimed by the '720 patent. Specifically, APA-Bach notes that "[t]he *copy-on-write* bit, used in the *fork* system call, indicates that the kernel must create a new copy of the page when a process modifies its contents." *See* APA-Bach at 287. And, even more specifically, APA-Bach discloses:

> As explained in Section 7.1, the kernel duplicates every region of the parent process during the fork system call and attaches it to the child process. Traditionally, the kernel of a swapping system makes a physical copy of the parent's address space, usually a wasteful operation, because processes often call exec soon after the fork call and immediately free the memory just copied. On the System V paging system, the kernel avoids copying the page by manipulating the region tables, page table entries, and pfdata table entries: It simply increments the region reference count of shared regions. . . . The page can now be referenced through both regions, which share the page until a process writes to it. **The kernel then copies the page so that each region has a private version. To do**

> **this, the kernel turns on the 'copy on write' bit for every page table entry in private regions of the parent and child processes during fork.** If either process writes the page, it incurs a protection fault, and in handling the fault, the kernel makes a new copy of the page for the faulting process. The physical copying of the page is thus deferred until a process really needs it."

See APA-Bach at 289–90 (emphasis added). Thus the Examiner's well-developed rejection based on Webb and Kuck—combined with the explicit copy-on-write disclosure of APA-Bach—renders obvious the claims of the '720 patent.

One of ordinary skill in the art would have many reasons to combine the Webb, Kuck, and APA-Bach references. As an initial matter, the Applicant has admitted in the '720 patent the relevance of the APA-Bach reference, which is directed to the implementation of the fork( ) system call in a UNIX environment. The Applicant did this by incorporating into the patent APA-Bach's discussion of the UNIX fork( ) system call. Just as the Applicant pointed to APA-Bach as an obvious source to look to for an explanation of the fork( ) system call, so to would one of ordinary skill in the art have looked to APA-Bach for a description of the fork( ) system call. In that text, they would also find the description of the copy-on-write capability that renders the '720 patent obvious. Even without the Applicant's admission of the applicability of the APA-Bach reference, one or ordinary skill in the art would, consistent with the Examiner's findings, look for ways to reduce system memory usage when instantiating or executing virtual machines. For example, *see* Kuck at ¶ [0065]: "Initialization (as well as execution) overhead can be further reduced through another optimization: storing type information (i.e., the runtime representation of loaded classes) in a section of shared memory that can be accessed by all the PAVMs. This technique can reduce the overhead for class loading, verification, and resolution incurred by each PAVM." Further in addition to the two examples above, and even more broadly, any programmer of ordinary skill in the art at the time of the invention, dealing with the

cloning of a master runtime process as disclosed by Webb in view of Kuck, would have been aware of and looked to utilize the strengths of the UNIX platform. This would include the UNIX copy-on-write functionality described in APA-Bach.

The Examiner, if aware of the APA-Bach reference, would have likely considered the teachings of Webb and Kuck in view of APA-Bach to be material in determining whether or not the claims of the '720 patent were patentable. As further detailed in the claim chart in Exhibit 17, the Webb, Kuck, and APA-Bach references render obvious claims 1-8, 10-17, and 19-22 of the '720 patent. For this reason, the Webb, Kuck, and APA-Bach references raise a substantial new question of patentability with respect to claims 1-8, 10-17, and 19-22 of the '720 patent.

Dike in view of Steinberg

Dike explicitly discloses the use of the fork( ) system call to clone virtual machine processes onto the host system, and allows the various processes access to the kernel data through a shared implementation of a copy-on-write data structure. Dike anticipates each element of the '720 patent, including the copy-on-write limitation.

Dike was published in 2000. Steinberg was published on Dec. 19, 2002. Since the priority date of the '720 patent is December 22, 2003, both the Dike reference and the Steinberg reference qualify as prior art under 35 U.S.C. § 102(b). Neither Dike nor Steinberg was in front of the Patent Office during the prosecution of the application that matured into the '720 patent nor is Dike in view of Steinberg cumulative to the prior art considered by the Patent Office during the prosecution of the '720 patent, at least because of its disclosure of copy-on-write in conjunction with cloning of system processes.

Dike is directed to the creation of a user space virtual machine that allows a Linux kernel to run in a set of Linux processes using simulated hardware. *See* Dike at Abstract. The

simulated hardware is constructed from services provided by the host kernel. *See id.* Dike creates an infrastructure that allows the Linux kernel to run just as it does on physical hardware. *See* Dike at § 3. In order to effect this, Dike discloses a method for intercepting system calls and executing them in the virtual terminal. *See id.* at § 2.1. This is necessary because the processes of the user space virtual machine will tap directly into the host kernel, a method necessary to virtualize system calls. *See id.*

Dike calls for the creation of "a new process in the host for each new process in the virtual machine." *See* Dike at § 2.3. This is accomplished by executing the "fork or clone" system call to create a process in the host corresponding to a process in the user space virtual machine, i.e., cloning the virtual machine process. *See id.* "Since the generic kernel arranged for the new address space to be a copy of the parent address space, and the new process has the same registers as the old one, except for the zero return value from the system call, it is a copy of its parent." *See id.* The underlying cloning process used in Dike is the Linux fork( ) call, which is optimized using copy-on-write, as evidenced by the teachings of Dike, *see e.g.*, the third to last paragraph of section 2.1.

Each of the cloned virtual machine processes (using the copy-on-write mechanism of the Linux fork( ) call) running on the host kernel will need to access the kernel data. Because the corresponding processes running on the virtual machine have separate addresses in the host system, the Dike disclosure suggests further creating a shared data segment from the copy-on-write segment containing the kernel data. *See, e.g, id.* ("This converts a copy-on-write segment of the address space into a shared segment."). This conversion allows for a system where "the kernel's memory is in each process address space." *See* Dike at § 5.1. The copy-on-write nature of the cloning process of the Linux fork( ) call used in Dike is evidenced by this discussion, i.e.,

Dike's approach to sharing *kernel* memory—which contains data such as device buffers—across processes where the kernel memory is initially stored in copy-on-write memory segments. *See* Dike at § 2.1 ("Each process within a virtual machine gets its own process in the host kernel. Even threads sharing an address space in the user-mode kernel will get different address spaces in the host. Even though each process gets its own address spaces, they must all share the kernel data. Unless something is done to prevent it, every process will get a separate, copy of the kernel data. So, what is done is that the data segment of the kernel is copied into a file, unmapped, and that file is mapped shared in its place. This converts a copy-on-write segment of the address space into a shared segment.").

The Steinberg reference further explains the copy-on-write mechanism of the Linux fork( ) system call that is disclosed by Dike. Specifically, Steinberg reiterates Dike's disclosure of copy-on-write in conjunction with the Linux fork( ) system call: "The normal means by which a Linux process can create new tasks is the fork system call. This system call creates an exact copy of the calling process, which then becomes the calling process' child process. The created child process inherits copies of the parent process data space, heap and stack, which are copied on demand using a copy-on-write mechanism." *See* Steinberg at 21; *see also* Steinberg at 16 (describing the Linux clone functionality).

A reasonable examiner would have considered the teachings of Dike to be material to the patentability of the claims of the '720 patent. As further detailed in the claim chart in Exhibit 18, the Dike reference anticipates claims 1-8, 10-17, and 19-22 of the '720 patent.

Also as further detailed in the claim chart in Exhibit 18, Dike in view of Steinberg renders obvious the claims of the '720 patent. It would have been obvious to one of ordinary skill in the art at the time of the invention to combine Dike's disclosure, directed to virtual

machines in a Linux environment and making using of a fork( ) system call, with Steinberg's disclosure of the Linux fork( ) system call and its explicit use of the copy-on-write mechanism. For this reason, the Dike reference raises a substantial new question of patentability with respect to claims 1-8, 10-17, and 19-22 of the '720 patent.

Bryant in view of Bach, Bryant in view of Traut

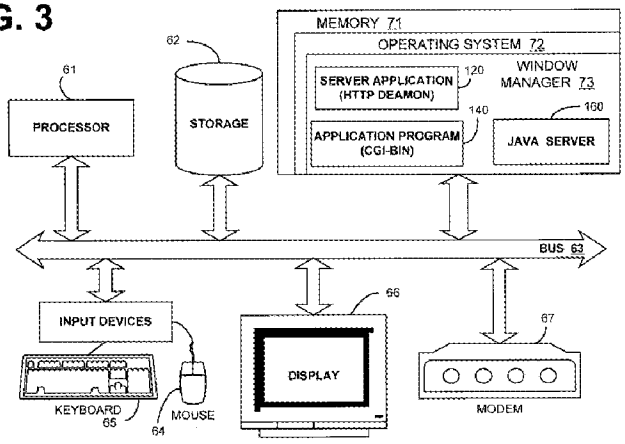
Bryant discloses a method and apparatus useful for increasing the performance of Java programs by executing a fork command and creating a child java machine. Bryant, in combination with the well-known copy-on-write process cloning mechanism, teaches all the elements of claims 1-8, 10-17, and 19-22 of the '720 patent, including the purported point of novelty of reducing memory usage for virtual machines by using copy-on-write cloning.

Bryant was filed on June 5, 1998, and issued as a U.S. Patent on June 11, 2002. The APA-Bach reference was published in 1986. Traut was filed on July 11, 2002, published as U.S. Publication No. 2004/0010787 on Jan. 15, 2004, and later issued as a U.S. Patent on December 25, 2007. Since the priority date of the '720 patent is December 22, 2003, each of the Bryant, APA-Bach, and Traut references qualify as prior art under 35 U.S.C. § 102(b). None of the above-listed references were in front of the Patent Office during the prosecution of the application that matured into the '720 patent nor are they cumulative to the prior art considered by the Patent Office during the prosecution of the '720 patent, at least because the Bryant reference pertains specifically and directly to Java technologies, and because each of APA-Bach, and Traut disclose a copy-on-write mechanism.

Bryant discloses a Java server that "invokes the Java virtual machine and preloads all potentially needed object files during initialization of the Java virtual machine to speed up the actual execution of a particular Java application. The Java server accomplishes the execution of

a particular Java application by forking itself and then having the child Java server run the Java class files in the already loaded Java virtual machine for the specific Java CGI-BIN script." *See* Bryant at 2:46-48. In Fig. 3, Bryant clearly discloses a processor 61 and a memory 71:

FIG. 3



And in Fig. 8, Bryant clearly discloses a class preloader to obtain a representation of at least one class from a source definition provided as object-oriented program code: "The process of the Java server 160 will now be discussed with respect to FIG. 8. First, the Java server 160 is initialized at step 161, and then waits to be called. The initialization step 161 includes starting the Java virtual machine and loading the standard class files needed for Java application execution." *See* Bryant at 6:66-7:4.

Bryant discloses a runtime environment to clone the memory space as a child runtime system process responsive to a process request and to execute the child runtime system process at 5:9-14 and 7:7-14, both in the context of improving the execution of Java-interpreted language execution in application software. *See id.* at 7:7-14 ("Immediately upon being called by the application program 140, the Java server 160 forks a process of the child server 180 with the pipe connection, thereby establishing communication with the application program 140, with Java server 160 and with the process of the child server 180 at step 163."). The disclosure of Bryant was directed primarily towards a means to streamline and accelerate the loading of large Java

scripts. *See id.* at 2:55-63. One of ordinary skill in the art at the time of the invention would have looked to the prevailing information available in the field pertaining to methods for streamlining the execution of Java code within parent and child Java Virtual Machines. In this case, the Bryant reference implements the fork( ) system call, as explained and explicitly recited in the Abstract. Here, the combination of Bryant with certain variants of Unix such as System V, or with a Linux system, or with any other UNIX system with copy-on-write, will inherently make use of the copy-on-write feature that is built into the software. And even where a system not already automatically including copy-on-write is used, a person of ordinary skill in the art would have been familiar with UNIX and, in keeping with Bryant's stated goal to "increase the performance of Java application execution," *see* Bryant at Abstract, would have looked to the copy-on-write feature prevalent in many UNIX systems at the time of the invention.

Bryant, when read in view of APA-Bach or Traut, renders obvious the limitation of using a copy-on-write process cloning mechanism to instantiate the child runtime system process by copying references to the memory space of the master runtime system process into a separate memory space for the child runtime system process, and to defer copying of the memory space of the master runtime system process until the child runtime system process needs to modify the referenced memory space of the master runtime system process.

The APA-Bach reference is particularly instructive, as described above. APA-Bach clearly and conspicuously discloses the exact copy-on-write limitation claimed by the '720 patent. Thus Bryant in view of APA-Bach renders obvious the claims of the '720 patent.

Bryant may also be combined with Traut, which specifically discloses methods to improve the speed of a forked virtual machine, including the use of "copy on write" "forking":

> Referring now to FIGS. 2 and 3, "forking" is a term used by UNIX programmers to describe the duplication of a UNIX process and its address space. Both the

> original process and the fork are then allowed to run as independent processes from the forking point. **The implementation of forking often involves a technique called "copy on write" in which case all memory pages in both address spaces are marked "write protected."** When either the original or the forked process writes to a page, a copy is made so that each process has its own copy. Pages that are not modified can continue to be shared between the two processes. **This technique not only saves on memory resources, but it also makes forking much faster than otherwise possible.**

See Traut at ¶ [0026] (emphasis added). A reasonable examiner would have considered the teachings of Bryant in view of APA-Bach to be material in determining whether or not the claims of the '720 patent were patentable. Further, a reasonable examiner would have considered the teachings of Bryant in view of Traut to be important in determining whether or not the claims of the '720 patent were patentable. As detailed in the claim chart in Exhibit 19, the Bryant and APA-Bach references render obvious claims 1-8, 10-17, and 19-22 of the '720 patent. And as detailed in the claim chart in Exhibit 20, the Bryant and Traut references render obvious claims 1-8, 10-17, and 19-22 of the '720 patent. For this reason, the Bryant, APA-Bach, and Traut references raise a substantial new question of patentability with respect to claims 1-8, 10-17, and 19-22 of the '720 patent.

Srinivasan in view of APA-Bach

Srinivasan is directed to a detailed discussion of the Perl programming language, and provides significant information regarding execution of fork commands, whereby "[t]he fork call results in two identical processes -- the parent and child -- starting from the statement following the fork." *See* Srinivasan at 195. Srinivasan, in combination with the well-known copy-on-write mechanism, teaches all the elements of claims 1-8, 10-17, and 19-22 of the '720 patent, including the purported point of novelty of reducing memory usage for virtual machines by using copy-on-write cloning.

Srinivasan was published in August of 1997. The APA-Bach reference was published in 1986. Since the priority date of the '720 patent is December 22, 2003, both the Srinivasan and APA-Bach references qualify as prior art under 35 U.S.C. § 102(b). Neither of the above-listed references were in front of the Patent Office during the prosecution of the application that matured into the '720 patent nor are they cumulative to the prior art considered by the Patent Office during the prosecution of the '720 patent, at least because the Srinivasan reference pertains specifically and directly to programming languages using the fork( ) system call, and because APA-Bach discloses the use of a copy-on-write mechanism to improve the efficiency of the fork( ) system call.

Srinivasan is a multi-chapter disclosure focused on the Perl programming language, and specifically contemplates the implementation of the fork( ) system call, "on Unix and similarly empowered systems." *See* Srinivasan at 194. Srinivasan describes the creation of a child process from a master runtime process: "The server process acts as a full-time receptionist: it blocks on `accept`, and when a connection request comes in, it spawns a child process and goes back to `accept`. The newly created child process meanwhile has a copy of its parent's environment and shares all open file descriptors." *See id.* Srinivasan discusses the effect of the fork( ) system call on the computer memory:

> On Unix, when a child process exits (or terminates abnormally), the system gets rid of the memory, files, and other resources associated with it. But it retains a small amount of information (the exit status if the child was able to execute `exit()`, or a termination status otherwise), just in case the parent uses `wait` or `waitpid` to enquire about this status. The terminated child process is also known as a *zombie* process, and it is always a good thing to remove it using `wait`; otherwise, the process tables keep filling up with junk.

See id. at 195. APA-Bach, as disclosed above with respect to the Bryant reference, explicitly discloses the copy-on-write limitation. Thus, one of ordinary skill in the art, assessing the fork( )

system call as described by Srinivasan and seeking to further improve the effect on the system memory, would look to the relevant art, and in doing so would find the copy-on-write mechanism disclosed by APA-Bach.

A reasonable examiner would have considered the teachings of Srinivasan in view of APA-Bach to be material to determining whether or not the claims of the '720 patent were patentable. As further detailed in the claim chart in Exhibit 21, the Srinivasan and APA-Bach references render obvious claims 1-8, 10-17, and 19-22 of the '720 patent. For this reason, the Srinivasan and APA-Bach references raise a substantial new question of patentability with respect to claims 1-8, 10-17, and 19-22 of the '720 patent.

Sexton in view of Bugnion, Sexton in view of Johnson

Like the claimed invention of the '720 patent, Sexton is also directed to the streamlining of Java programs by way of utilizing a memory space shared between Virtual Machines. The '720 patent claims a fork( ) system call, creating a memory space clone of a parent or master runtime process, in order to streamline the execution of process requests. In order to further streamline that process, the '720 patent applies a copy-on-write mechanism to allow the master and clone Virtual Machines to share a memory space, creating a full copy of the memory space only when necessary. Given the goal of reducing session memory by sharing data between multiple Virtual Machines, one of ordinary skill in the art at the time of the invention would look to combine the disclosure of Sexton with the well-known copy-on-write mechanism, thereby placing the artisan in possession of the invention. Further to that point, two references explicitly disclosing a copy-on-write mechanism within the context of a Java Virtual Machine are cited on the face of the Sexton patent: Bugnion and Johnson. The disclosure of Sexton coupled with the

basic knowledge available to one of ordinary skill in the art poses a substantial new question of patentability.

The Sexton reference was filed on Feb. 25, 2000 and issued as a patent on Feb. 8, 2005, and thus is prior art to the '720 patent under 35 U.S.C. § 102(e), given a priority date for the '720 patent of December 22, 2003. The Bugnion patent issued on Jun. 13, 2000, and the Johnson patent issued on Dec. 11, 2001; both the Bugnion and Johnson patents are prior art to the '720 patent under 35 U.S.C. § 102(b). Neither the Sexton nor the Bugnion nor the Johnson reference was cited to or considered by the Patent Office during the prosecution of the application that matured into the '720 patent. Further, Sexton in combination with either of Bugnion or Johnson is not cumulative to the prior art considered by the Patent Office during the prosecution of the '720 patent.

Sexton provides techniques for instantiating separate Java Virtual Machines in a way that can "reduce[e] startup costs and incremental memory requirements of the Java virtual machines instantiated by the server." *See* Sexton at 5:53-55. This streamlining of the Java process is accomplished by sharing certain data between the Virtual Machines. "Each VM instance has read-only access to the data that has been loaded into the shared state area, and therefore the VM instances do not contend with each other for access rights to that data. According to one embodiment, the shared state area is used to store loaded Java classes." *See* Sexton at 8:45-48. This allows the Java process to proceed on each Virtual Machine without making two copies of the data, which is the first step in—and, essentially, half way to—the implementation of a copy-on-write mechanism. In other words, "[t]he non-session-specific data for the class, including the methods, method table and fields, are not duplicated in the session memory for each VM instance. Rather, all VM instances share read-only access to a single instantiation of the class,

thus significantly reducing the memory requirements of VM instances (the per-session memory requirements)." *See* Sexton at 8:55-61. Thus, Sexton discloses a method to reduce the memory impact of creating child virtual machines by creating a shared memory session. This is essentially the same functionality disclosed in APA-Bach ("referring to the UNIX "copy-on-write" mechanism) and as claimed by the '720 patent. Sexton discloses a shared memory state among the virtual machines and, combined with the explicit disclosure of "copy-on-write" in either Bugnion or Johnson, renders the '720 patent obvious.

Bugnion discloses a Virtual Machine Monitor ("VMM") layer that "also maintains copy-on-write disks that allow Virtual Machines to transparently share main memory resources and disk storage resources, and performs dynamic page migration/replication that hides distributed characteristics of the physical memory resources from the operating systems. The VMM layer may also comprise a virtual memory resource interface to allow processes running on multiple virtual machines to share memory." *See* Bugnion at 6:29-36. Similarly, Johnson discloses a preferred embodiment that "introduces copy on write storage which provides access to static variables. Static variable as defined by a class file are stored at a particular address in SAS copy on write storage. When an instance of a class is created, any static variables defined for that class are created using the definitions stored in the class file in SAS. The present invention allows static variables to be shared among instances of a class running in a particular JVM. However, different JVMs do not access the same static variables. Each JVM has its own copies of any static variables defined for a class of which it has an instance." *See* Johnson at 18:34-44.

A reasonable examiner would have considered the teachings of Sexton material to determining whether or not the claims of the '720 patent were patentable. As further detailed in the claim chart in Exhibit 22, the Sexton and Bugnion references render obvious claims 1-8, 10-

17, and 19-22 of the '720 patent. And, as further detailed in the claim chart in Exhibit 23, the Sexton and Johnson references render obvious claims 1-8, 10-17, and 19-22 of the '720 patent. For this reason, the Sexton reference in combination with either Bugnion or Johnson raises a substantial new question of patentability with respect to claims 1-8, 10-17, and 19-22 of the '720 patent.

III. DETAILED EXPLANATION UNDER 37 C.F.R. § 1.915(B)(3) OF THE PERTINENCY AND MANNER OF APPLYING THE CITED PRIOR ART TO EVERY CLAIM FOR WHICH REEXAMINATION IS REQUESTED

The detailed explanation herein under 37 C.F.R. § 1.915(b)(3) comprises a summary of the reasons for unpatentability of the claims (set forth below) supported by detailed Claim Charts. This detailed explanation describes the pertinence and manner of applying the prior art references to the claims of the '720 patent.

A. Rejections of Claims

1. Claims 1-8, 10-17, and 19-22 are unpatentable under 35 U.S.C. § 103(a) as rendered obvious by Webb in view of Kuck and further in view of APA-Bach.

As discussed above, each of the Webb, Kuck, and APA-Bach references are prior art to the '720 patent. And as set forth in detail in the Claim Chart attached as Exhibit 17, Webb in view of Kuck and further in view of APA-Bach renders obvious each of claims 1-8, 10-17, and 19-22 of the '720 patent.

2. Claims 1-8, 10-17, and 19-22 are unpatentable under 35 U.S.C. § 102(b) as anticipated by Dike in view of Steinberg.

As discussed above, the Dike and Steinberg references are prior art to the '720 patent. And as set forth in detail in the Claim Chart attached as Exhibit 18, Dike anticipates each of claims 1-8, 10-17, and 19-22 of the '720 patent.

3. Claims 1-8, 10-17, and 19-22 are unpatentable under 35 U.S.C. § 103(a) as rendered obvious by Dike in view of Steinberg.

As discussed above, the Dike and Steinberg references are prior art to the '720 patent. And as set forth in detail in the Claim Chart attached as Exhibit 18, Dike in view of Steinberg renders obvious each of claims 1-8, 10-17, and 19-22 of the '720 patent.

4. **Claims 1-8, 10-17, and 19-22 are unpatentable under 35 U.S.C. § 103(a) as rendered obvious by Bryant in view of APA-Bach.**

As discussed above, each of the Bryant and APA-Bach references are prior art to the '720 patent. And as set forth in detail in the Claim Chart attached as Exhibit 19, Bryant in view of APA-Bach renders obvious each of claims 1-8, 10-17, and 19-22 of the '720 patent.

5. **Claims 1-8, 10-17, and 19-22 are unpatentable under 35 U.S.C. § 103(a) as rendered obvious by Bryant in view of Traut.**

As discussed above, each of the Bryant and Traut references are prior art to the '720 patent. And as set forth in detail in the Claim Chart attached as Exhibit 20, Bryant in view of Traut renders obvious each of claims 1-8, 10-17, and 19-22 of the '720 patent.

6. **Claims 1-8, 10-17, and 19-22 are unpatentable under 35 U.S.C. § 103(a) as rendered obvious by Srinivasan in view of APA-Bach**

As discussed above, each of the Srinivasan and APA-Bach references are prior art to the '720 patent. And as set forth in detail in the Claim Chart attached as Exhibit 21, Srinivasan in view of APA-Bach renders obvious each of claims 1-8, 10-17, and 19-22 of the '720 patent.

7. **Claims 1-8, 10-17, and 19-22 are unpatentable under 35 U.S.C. § 103(a) as rendered obvious by Sexton in view of Bugnion.**

As discussed above, each of the Sexton and Bugnion references are prior art to the '720 patent. And as set forth in detail in the Claim Chart attached as Exhibit 22, Sexton in view of Bugnion renders obvious each of claims 1-8, 10-17, and 19-22 of the '720 patent.

8. **Claims 1-8, 10-17, and 19-22 are unpatentable under 35 U.S.C. § 103(a) as rendered obvious by Sexton in view of Johnson.**

As discussed above, each of the Sexton and Johnson references are prior art to the '720 patent. And as set forth in detail in the Claim Chart attached as Exhibit 23, Sexton in view of Johnson renders obvious each of claims 1-8, 10-17, and 19-22 of the '720 patent.

IV. CONCLUSION

For the reasons provided herein, Requester respectfully submits that the prior art submitted herewith raises substantial new questions of patentability as to claims 1-8, 10-17, and 19-22 of the '720 patent because, as discussed above, claims 1-8, 10-17, and 19-22 of the '720 patent are either anticipated or rendered obvious in view of the prior art patents and printed publications discussed herein. Accordingly, reexamination of claims 1-8, 10-17, and 19-22 of the '720 patent is respectfully requested, finally rejecting these claims.

The undersigned further notes the standards set forth at 37 C.F.R. 1.903 wherein the reexamination Requester will be sent copies of Office actions issued during the reexamination proceedings as well as served (by the patent owner) with any document filed in the reexamination proceeding in accordance with 37 C.F.R. 1.248. (*See* MPEP § 2624.)

If the Patent Office determines that a fee and/or other relief is required, Requester petitions for any required relief including authorizing the Commissioner to charge the cost of such petitions and/or other fees due in connection with the filing of this document to **Deposit Account No. 11-0980** referencing Docket No. 13557.105125.

As identified in the attached Certificate of Service and in accordance with 37 C.F.R. §§ 1.33(c) and 1.915(b)(6), a copy of the present request is being served to the address of the attorney or agent of record.

March 1, 2011

Respectfully submitted,

By / Robert T. Neufeld/
Robert T. Neufeld
Patent Attorney
Registration No. 48,394
KING & SPALDING LLP
1180 Peachtree Street
Atlanta, Georgia 30309