US009507613B2

(12) **United States Patent**
Zhou et al.

(10) **Patent No.: US 9,507,613 B2**
(45) **Date of Patent: Nov. 29, 2016**

(54) **METHODS AND APPARATUS FOR DYNAMICALLY PRELOADING CLASSES**

(75) Inventors: **Jiangli Zhou**, San Ramon, CA (US); **Christopher Jordan Plummer**, San Martin, CA (US); **Dean Roy Ernest Long**, Chula Vista, CA (US)

(73) Assignee: **Oracle International Corporation**, Redwood City, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 740 days.

(21) Appl. No.: **13/436,476**

(22) Filed: **Mar. 30, 2012**

(65) **Prior Publication Data**

US 2013/0263123 A1 Oct. 3, 2013

(51) **Int. Cl.**

G06F 9/44	(2006.01)
G06F 9/455	(2006.01)
G06F 9/445	(2006.01)

(52) **U.S. Cl.**

CPC ***G06F 9/45504*** (2013.01); ***G06F 9/445*** (2013.01)

(58) **Field of Classification Search**

None

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,815,718 A * 9/1998 Tock G06F 9/443 712/E9.084
5,966,542 A 10/1999 Tock
6,223,346 B1 * 4/2001 Tock G06F 9/443 712/E9.084
6,470,494 B1 * 10/2002 Chan G06F 9/445 717/139
6,571,388 B1 * 5/2003 Venkatraman G06F 9/44557 717/108
6,658,492 B1 * 12/2003 Kawahara G06F 9/44573 717/159
6,745,386 B1 * 6/2004 Yellin G06F 9/44521 717/139
6,813,762 B1 * 11/2004 Plaxton G06F 9/44557 717/139
6,826,559 B1 * 11/2004 Ponte G06F 17/30867
6,996,813 B1 * 2/2006 Sokolov G06F 9/44521 717/139

(Continued)

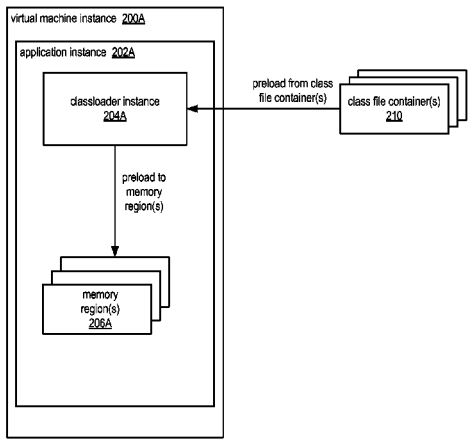
OTHER PUBLICATIONS

"Dalvik Bytecode Verifier Notes," 2008 The Android Open Source Project, downloaded from http://www.netmite.com/android/mydroid/dalvik/docs/verifier.html on Feb. 22, 2012, 2 pages.

(Continued)

Primary Examiner — Emerson Puente
Assistant Examiner — Mehran Kamran
(74) *Attorney, Agent, or Firm* — Robert C. Kowert; Meyertons, Hood, Kivlin, Kowert & Goetzel, P.C.

(57) **ABSTRACT**

A class preloading mechanism that dynamically preloads classes at runtime in a virtual machine (VM) environment. Data structures representing preloaded classes may be stored in a persistent module corresponding to a classloader. A persistent module can be directly mapped or copied into a memory region at runtime so that the classes may not have to be loaded from the class file container. The preloaded classes are not fully linked and resolved. When a classloader receives a class request, the classloader looks up the preloaded class in the memory region and completes linking and resolution of the class. Persistent modules may be pre-generated and, for example, installed with an application. Alternatively, a persistent module for a class file container may be generated at runtime by preloading classes from the class file container into a memory region and storing data structures representing the classes as a persistent module.

20 Claims, 9 Drawing Sheets