What does Oop Maps means in Hotspot VM exactly

Asked 10 years, 6 months ago Modified 4 years, 3 months ago Viewed 4k times

16

I read from some documents that Hotspot VM utilizes a data structure called Oop Maps to manage all OOPs in VM. My question is that when does this Oop Map data structure generated? At compile time or runtime? Any further detailed documents regarding to this will be more than welcomed. Thank you guys.

java jvm jvm-hotspot

Share Improve this question Follow

asked Sep 25, 2014 at 2:56
Bill Randerson
1,088 1 13 29

1 Answer

Sorted by: Highest score (default)

32

OopMap is a structure that records where object references (OOPs) are located on the Java stack. Its primary purpose is to find GC roots on Java stacks and to update the references whenever objects are moved within the Heap.

There are three kinds of OopMaps:

1. OopMaps for **interpreted methods**. They are computed lazily, i.e. when GC happens, by analyzing bytecode flow. The best reference is the source code (with lots of comments), see generateOopMap.cpp. InterpreterOopMaps are stored in OopMapCache.
2. OopMaps for **JIT-compiled methods**. They are generated during JIT-compilation and kept along with the compiled code so that VM can quickly find by instruction address the stack locations and the registers where the object references are held.
3. OopMaps for generated **shared runtime stubs**. These maps are constructed manually by the developers - authors of these runtime stubs.

During GC JVM walks through all thread stacks. Each stack is parsed as a stream of stack frames. The frames are either interpreted or compiled or stubs. Interpreted frames contain information about Java method and bci (bytecode index). OopMapCache helps to find an OopMap corresponding to the given method and bci. The method of a compiled frame is discovered by instruction address lookup.

Share Improve this answer Follow

edited Sep 15, 2018 at 11:43

answered Sep 25, 2014 at 22:56
apangin
98.9k 11 210 267

That's pretty cool answer, but I want to know more if not inappropriate: 1) OopMaps for interpreted methods. They are computed lazily, i.e. when GC happens, by analyzing bytecode flow. >>>>>>>> Where does these information stored in JVM? Any detailed reference about how they're computed? Or simply source code files in OpenJDK? 2) So when GC happens, and thread goes to safepoint, then JVM can simply find all the GC root set for current thread by checking Oop Maps in current safe point. Is the correct? Thanks a lot.
– Bill Randerson Sep 26, 2014 at 3:25

3 @BillRanderson 1) The best reference is the source code (well documented BTW): generateOopMap.cpp. InterpreterOopMap are stored in OopMapCache. – apangin Sep 26, 2014 at 17:57

2 @BillRanderson 2) During GC JVM walks through all thread stacks. Each stack is parsed as a stream of stack frames. The frames are either interpreted or compiled or stubs. Interpreted frames contain information about Java method and bci (bytecode index). OopMapCache helps to find a OopMap corresponding to the given method and bci. The method of a compiled frame is discovered by instruction address lookup. Compiled methods contain OopMaps along with the code. – apangin Sep 26, 2014 at 18:06

1 Thanks a lot for your elaborate explanation. Even though I cannot fully comprehend the process for now, I will figure it out later. – Bill Randerson Sep 27, 2014 at 2:54