How to check the jitted Java method parameters from the JVM core dump through assembly code?

Asked 2 years, 11 months ago Active 2 years, 11 months ago Viewed 250 times

0

1

I want to get the Assembly code for the jitted code to get the parameters value based on the Java calling convention. Suppose the JVM is hotspot, the platform is Linux 64 bit , and we have the following caller and calle, and I want to check the parameters passed to callee from the JVM core dump.

```
protected void caller( ) {
callee(1,"123", 123,1);

}

protected void callee(int a,String b, Integer c,Object d) {
 Thread.sleep(11111111);
}
```

Based on the following Java calling convention, we know we can get the parameters from the Registers, such as up to 6 first integer arguments are passed in registers: rsi, rdx, rcx, r8, r9, rdihttp://hg.openjdk.java.net/jdk8u/jdk8u/hotspot/file/b4bdf3484720/src/cpu/x86/vm/assembler_x86.hpp#l91

For the c/c++ method, and we can use the gdb just through the way by printting call stack by the command backtrace , then frame N(N is the thread number), then x/20i $pc-64 to get the assembly code, and we can get the value from the related frame context Register. However the Java method call stack can not be printed from gdb, and we don't know the frame number, then we cannot use the same way like c/c++ to get the assembly code,so how to check the assembly code for the Java jitted method from the core dump?

PS, Someone mentioned the PrintOptoAssembly, however I need the Assembly code to get the parameters value from the registers by calling convention(such as by backtrace , then frame N , then x/20i $pc-64 through gdb) not just the Assembly code only.

java jvm gdb

edited Jan 18 '17 at 10:59

asked Jan 18 '17 at 9:23
YuFeng Shen
961 7 23

No, I know the PrintOptoAssembly , however PrintOptoAssembly just print the Assembly code, but it cannot switch into the special frame to get the register value like gdb frame N and x/20i $pc-64 . – YuFeng Shen Jan 18 '17 at 10:54

i think hotspot runs with the same calling convention as C code. Have you tried with `-XX:+PreserveFramePointer` ? – the8472 Jan 18 '17 at 13:54

Thank you the8472 , however this is helpful for c/c++ hotspot code, however what I asked is for Java code. – YuFeng Shen Jan 19 '17 at 2:37

1 Answer

2

You won't see Java frames with gdb `backtrace` command. However, you don't need to extract VM structures from a coredump manually - there are better options.

1. HotSpot Serviceability Agent

Serviceability Agent is an instrument designed specially for analyzing memory of a Java process or a coredump. It has Java API available in `sa-jdi.jar` supplied with a standard JDK package.

Here is an example that prints extended Java stacktraces wtih local variable info. It can also parse coredumps.

2. HotSpot debug functions

Debug builds of HotSpot JVM include special debugging functions that can be called from gdb. E.g.

- `psf()` print stack frames;
- `pfl()` print frame layout;
- `disnm(intptr_t addr)` disassemble compiled Java method at given address;
- `pp(intptr_t addr)` print Java object at given address;
- etc. See other commands in debug.cpp.

These functions work while debugging an active process; not suitable for coredumps though.

BTW, a quick guide to building debug version of JVM by yourself.

edited May 23 '17 at 12:08
Community ♦
1 1

answered Jan 18 '17 at 22:25
apangin
61.6k 9 130 164

Thank you apangin, we can get local variable from the coredump by SA, however I want to know is if we can get the method parameter values by SA? We know based on the Java calling convention, some method parameters would be stored in the registers, however the register value would be changed in different method frame context, so we need the frame N to switch to the special frame to get the register value for the C++/c code.I afraid if the SA can get the register value at a special frame? It would be so great if there is an example for getting the parameter values from coredump by the SA. – YuFeng Shen Jan 19 '17 at 2:34

BTW, I need to get the parameters value for the coredump. – YuFeng Shen Jan 19 '17 at 2:49

@Hermas Method parameters **are** local variables, and like regular variables they can change. Generally it is not possible to get parameter values as they were at the moment of method call, however you can often deduce them from the values of local variables with the help of disassembled method code. – apangin Jan 19 '17 at 15:59

Using SA you can iterate over JavaVFrames, and each VFrame has its own RegisterMap that describes where frame values are located and where registers are saved. – apangin Jan 19 '17 at 16:04

Apangin ,can you kind help to point out how to use the SA to get the disassembled method code(for the jitted method) for the core dump (suppose we don't use the PrintAssembly at startup) ? – YuFeng Shen Jan 20 '17 at 3:24

@Hermas This example already prints addresses of compiled methods (NMethods). Now you just need to call `Disassembler.decode` on these NMethods. – apangin Jan 20 '17 at 15:55

@ apangin, very appreciate your expert knowledge on JVM, thanks a lot. – YuFeng Shen Jan 22 '17 at 2:07