Interpreters and JIT Compilers: Questions

Jeremy Singer

Dec 2018

1. What is the main advantage of a bytecode interpreted language over a compiled language?

   (a) easier to port to new platform

   (b) higher performance application code

   (c) lower memory footprint

   (d) all of the above

2. In what way is a stack-based bytecode scheme more efficient than a register-based bytecode?

   (a) instructions are more compact

   (b) code is more compressible

   (c) no assumptions about CPU register file

   (d) all of the above

3. Name a stack-based interpreter virtual machine system.

4. Identify the missing initial C keyword that generates an appropriate pattern for a stack interpreter.

```
_____ (*instruction_pointer++) {

    case ADD:
      stack[sp-1] = stack[sp-1]+stack[sp];
      sp--;
      break;

    case SUB:
      ...

    case JMP:
      offset = (int)(*instruction_pointer);
      instruction_pointer += offset;
```

```
        break;

    ...
}
```

(a) while

(b) if

(c) switch

(d) for

5. At what granularity does an interpreted bytecode VM operate?

   (a) one method at a time

   (b) one basic block at a time

   (c) one instruction at a time

6. Which of the following is true for a bytecode interpreter?

   (a) *easy* to develop, *low* startup overhead, *poor* application performance

   (b) *hard* to develop, *high* startup overhead, *good* application performance

   (c) *easy* to develop, *high* startup overhead, *good* application performance

7. How might a bytecode interpreter become faster as it executes an application?

   (a) it saves previously translated bytecode sequences to avoid re-interpretation

   (b) the interpreter code is loaded into the CPU cache, reducing memory access time

   (c) interpreters do not get faster over time

8. Why is it expensive to just-in-time compile a bytecode program to native code at runtime?

9. What is the Pareto Principle for *hot* code execution?

10. Why does a typical adaptive compiler like OpenJDK HotSpot have several optimization levels?

    (a) it takes time to identify hot code at runtime

    (b) there are different tradeoffs for different kinds of code

    (c) the system has evolved over time, so there are various legacy compilers

11. How does a runtime compiler identify hot code?

    (a) stack sampling

    (b) back-edge counters

(c) hardware performance counters

(d) all of the above

12. In a multicore system, how might we reduce pause time for JIT compilers?

    (a) run compiler threads in the background

    (b) perform concurrent garbage collection

    (c) parallelize application workloads

    (d) all of the above

13. The latest version of ART (Android runtime) uses ahead-of-time (AOT) compilation for Dex bytecode. What are the characteristics of AOT compilation?

    (a) - longer app install time, - app code requires more space, + faster app execution

    (b) + shorter app install time, - app code requires more space, + faster app execution

    (c) - longer app install time, + app code requires less space, - slower app execution