Ahead of Time (AOT) Compilation

Vaibhav Choudhary (@vaibhav_c)
Java Platforms Team
https://blogs.oracle.com/vaibhav
Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.
Agenda of the day ...

1. Java Compilers as of Today
2. Limitations of current compilers
3. Need of AOT
4. AOT - Deep dive and working
5. AOT - Future
Java Compilers as of Today
At Compile Time

Java Source [*.java] → javac → Java Bytecode [*.class]

At Runtime Time

ClassLoader Verifier

Java Interpreter

JIT Compiler (C1, C2)

Native Level (OS, Hardware)

Most of the code get optimized here
Runtime compilers

• Currently we have two flavors of runtime compiler
  – C1 (client side compiler)
  – C2 (server side compiler)
  – TieredCompilation (A combination of C1 and C2)

• Only on interpreter mode, use -Xint
Concerns with Runtime compilers

• Start-up time
  – Not preferred or often tweaked for ultra-low latency applications like Stock Market.

• Heavy on Resource
  – Resource consumption is high at runtime.

• Takes time is giving the peak performance
  – Goes via several level of optimization and de-optimization.
Need of static compiler - Welcome to AOT
Ahead of Time Compilation (AOT)

• Generation of machine code before runtime (*.so)
• Load pre-existing code at the runtime
• JIT compilation not need at runtime ?
• Native code can be shared across VM processes.

• jaotc --output HelloWorld.so HelloWorld.class
• java -XX:AOTLibrary=./HelloWorld.so HelloWorld
At Compile Time

Java Source [*.java]

javac

Java Bytecode [*.class]

jaotc

Shared Object [*.so]

At Runtime Time

ClassLoader Verifier

Java Interpreter

JIT Compiler (C1, C2)

Native Level (OS, Hardware)

Most of the code get optimized here
Current runtime compiler

- JVM Process1
  - Own Runtime
    - Separate Native Code

- JVM Process2
  - Own Runtime
    - Separate Native Code

- JVM Process3
  - Own Runtime
    - Separate Native Code

With AOT compiler

- JVM Process1
  - Own Runtime
  - shared object

- JVM Process2

- JVM Process3
AOT run (jaotc)

• jaotc --output HelloWorld.so HelloWorld.class
• java -XX:AOTLibrary=./HelloWorld.so HelloWorld

-bash-4.1$ ./java -XX:+PrintAOT -XX:AOTLibrary=./HelloWorld.so
HelloWorld
    15    1    loaded ./HelloWorld.so aot library
    153   1    aot[ 1] HelloWorld.<init>()V
Hello AOT

• jaotc --output base.so --module java.base
VM options are important

```bash
-bash-4.1$ ./java -XX:+UseConcMarkSweepGC -XX:+PrintAOT -XX:AOTLibrary=./HelloWorld.so HelloWorld

Java HotSpot(TM) 64-Bit Server VM warning: Option UseConcMarkSweepGC was deprecated in version 9.0 and will likely be removed in a future release. Shared file ./HelloWorld.so error: UseG1GC has different value 'true' from current 'false'

5     1     skipped ./HelloWorld.so  aot library

Hello AOT
```
AOT run (jaotc)

-bash-4.1$ ./jaotc --info --output HelloWorld.so HelloWorld.class
Compiling HelloWorld...
1 classes found (62 ms)
2 methods total, 2 methods to compile (6 ms)
Compiling with 2 threads

2 methods compiled, 0 methods failed (923 ms)
Parsing compiled code (2 ms)
Processing metadata (21 ms)
Preparing stubs binary (1 ms)
Preparing compiled binary (0 ms)
Creating binary: HelloWorld.o (17 ms)
Creating shared library: HelloWorld.so (26 ms)
Total time: 1886 ms
<table>
<thead>
<tr>
<th>Performance Type</th>
<th>Dynamic (JIT)</th>
<th>Static (AOT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start-up performance</td>
<td>Tunable, but not so good</td>
<td>Best</td>
</tr>
<tr>
<td>Steady-state performance</td>
<td>Best</td>
<td>Good</td>
</tr>
<tr>
<td>Interactive performance</td>
<td>Not so good</td>
<td>Good</td>
</tr>
<tr>
<td>Deterministic performance</td>
<td>Tunable, but not best</td>
<td>Best</td>
</tr>
</tbody>
</table>

Need of AOT

• Cloud Applications
• Embedded Devices
• UI centric application
• Mobile devices