# 介绍

JDK 9引入了用于支持对Class文件和模块进行提前编译的工具Jaotc，以减少程序的启动时间和到达全速性能的预热时间，但由于这项功能必须针对特定物理机器和目标虚拟机的运行参数来使用，加之限制太多，Java开发人员对此了解、使用普遍比较少，本节我们将用Jaotc来编译Java SE的基础库（java.base模块），以改善本机Java环境的执行效率。

# 提前编译HelloWorld

使用自己编译的openjdk11

Jaotc的基本使用过程：

|  |
| --- |
| jaotc [选项] [名称 | 清单] |

* 名称：Java类或jar文件
* 清单：用“冒号:”分隔的类名称，模块，JAR文件或包含类文件的目录的列表

|  |
| --- |
| $ javac HelloWorld.java  $ java HelloWorld  hello world  $ jaotc --output libHelloWorld.so HelloWorld.class |

然后就生成了一个名为libHelloWorld.so的库，我们可以使用Linux的ldd命令来确认这是否是一个静态链接库，使用mn命令来确认其中是否包含了HelloWorld的构造函数和main()方法的入口信息，操作如下：

|  |
| --- |
| $ ldd libHelloWorld.so  statically linked  $ nm libHelloWorld.so  ……  0000000000003000 T A.text  0000000000003020 t HelloWorld.<init>()V  0000000000003120 t HelloWorld.main([Ljava/lang/String;)V |

现在就可以使用这个静态链接库而不是Class文件来输出HelloWorld了：

|  |
| --- |
| $ java -XX:AOTLibrary=./libHelloWorld.so HelloWorld  hello world |

# 提前编译JavaSE基础库

java.base包含的代码数量庞大，虽然其中绝大部分内容现在都能被Jaotc的提前编译所支持了，但总还有那么几个“刺头”会导致编译异常。因此我们要建立一个编译命令文件来排除这些目前还不支持提前编译的方法，将此文件取名为java.base-list.txt，其具体内容如下：

|  |
| --- |
| # jaotc: java.lang.StackOverflowError  exclude sun.util.resources.LocaleNames.getContents()[[Ljava/lang/Object;  exclude sun.util.resources.TimeZoneNames.getContents()[[Ljava/lang/Object;  exclude sun.util.resources.cldr.LocaleNames.getContents()[[Ljava/lang/Object;  exclude sun.util.resources..\*.LocaleNames\_.\*.getContents\(\)\[\[Ljava/lang/Object;  exclude sun.util.resources..\*.LocaleNames\_.\*\_.\*.getContents\(\)\[\[Ljava/lang/Object;  exclude sun.util.resources..\*.TimeZoneNames\_.\*.getContents\(\)\[\[Ljava/lang/Object;  exclude sun.util.resources..\*.TimeZoneNames\_.\*\_.\*.getContents\(\)\[\[Ljava/lang/Object;  # java.lang.Error: Trampoline must not be defined by the bootstrap classloader  exclude sun.reflect.misc.Trampoline.<clinit>()V  exclude sun.reflect.misc.Trampoline.invoke(Ljava/lang/reflect/Method;Ljava/lang/Object;[Ljava/lang/Object;)Ljava  # JVM asserts  exclude com.sun.crypto.provider.AESWrapCipher.engineUnwrap([BLjava/lang/String;I)Ljava/security/Key;  exclude sun.security.ssl.\*  exclude sun.net.RegisteredDomain.<clinit>()V  # Huge methods  exclude jdk.internal.module.SystemModules.descriptors()[Ljava/lang/module/ModuleDescriptor; |

开始进行提前编译了，使用的命令如下所示：

|  |
| --- |
| jaotc -J-XX:+UseCompressedOops -J-XX:+UseG1GC -J-Xmx4g --compile-for-tiered --info --compile-commands java.base-list.txt --output libjava.base-coop.so --module java.base |

Jaotc用了-J参数传递与目标虚拟机相关的运行时参数，这些运行时信息与编译的结果是直接相关的，编译后的静态链接库只能支持运行在相同参数的虚拟机之上，如果需要支持多种虚拟机运行参数（譬如采用不同垃圾收集器、是否开启压缩指针等）的话，可以花点时间为每一种可能用到的参数组合编译出对应的静态链接库。此外，由于Jaotc是基于Graal编译器开发的，所以现在ZGC和Shenandoah收集器还不支持Graal编译器，自然它们在Jaotc上也是无法使用的。事实上，目前Jaotc只支持G1和Parallel（PS+PS Old）两种垃圾收集器。使用Jaotc编译java.base模块的输出结果如下所示：

|  |
| --- |
| Compiling libjava.base-coop.so...  6206 classes found (5140 ms)  55847 methods total, 49514 methods to compile (3163 ms)  Compiling with 2 threads  ................................................................  49514 methods compiled, 0 methods failed (437626 ms)  Parsing compiled code (2015 ms)  Processing metadata (40077 ms)  Preparing stubs binary (2 ms)  Preparing compiled binary (314 ms)  Creating binary: libjava.base-coop.o (13902 ms)  Creating shared library: libjava.base-coop.so (7266 ms)  Total time: 537367 ms |

编译JDK 11的java.base大约花了9分钟的时间，生成的libjava.base-coop.o库大小为366MB。

用-XX：AOTLibrary来指定链接库位置运行HelloWorld

|  |
| --- |
| $ java -XX:AOTLibrary=./libjava.base-coop.so,./libHelloWorld.so HelloWorld  hello world |

还可以使用-XX:+PrintAOT参数来确认哪些方法使用了提前编译的版本，从输出信息中可以看到，如果不使用提前编译版本的java.base模块，就只有HelloWord的构造函数和main()方法是提前编译版本的：

|  |
| --- |
| $ java -XX:+PrintAOT -XX:AOTLibrary=./libHelloWorld.so HelloWorld  7 1 loaded ./libHelloWorld.so aot library  498 1 aot[ 1] HelloWorld.<init>()V  499 2 aot[ 1] HelloWorld.main([Ljava/lang/String;)V  hello world |

但如果加入libjava.base-coop.so，那使用到的几乎所有的标准Java SE API都是被提前编译好的，输出如下：

|  |
| --- |
| $ java -XX:+PrintAOT -XX:AOTLibrary=./libjava.base-coop.so,./libHelloWorld.so HelloWorld |

目前状态的Jaotc还有许多需要完善的地方，仍难以直接编译SpringBoot、MyBatis这些常见的第三方工具库，甚至在众多Java标准模块中，能比较顺利编译的也只有java.base模块而已。不过随着Graal编译器的逐渐成熟，相信Jaotc前途还是可期的。