JMTrace 工具使用的算法简介

在这个工具中，我采用的实现方式是基于字节码层面的。具体而言，采用了 ASM 框架来对字节码进行插桩。使用了 JVM 提供的 java.lang.Instrument 接口。
具体而言，在这个编程作业中任务实际上就是尝试在运行时记录对对象的 field 读写以及对数组元素进行读写。那么在字节码层面，相当于我们就是要对以下相关指令进行处理：
1. getstatic getfield
2. putstatic putfield
3. *aload
4. *astore

简单理解 JVM 内存模型的话，我们的任务可以这样表述：在以上指令出现时，出于对 javac 实现的信任，我们实际上就可以假设当时栈中有字节码手册中所规定的合规内容，那么我们就可以在这种已知信息的基础上通过 JVM 提供的栈数据操作指令，在栈上准备好我们需要记录信息的副本作为函数传入参数，并且调用一个用于信息记录的静态函数“消耗”掉我们在栈中额外写入的内容，这样就可以做到“无副作用”的插桩了。

若以 getfield 为例：
此命令运行时栈变化是：..., objectref → ..., value
那么字节码 getfield 隐含了类型“W”，同时 pid 可以在记录函数中通过调用系统 API 得到，标识读写对象的唯一 id，这里简单的使用对象、对象对应类对象、对象读写的 field 名称三者共同通过一个 hash 函数提供, objectref 中实际上可以获知类对象从而可知名称, 具体 field 名称也在 getfield 中可知，由此，我们所需的信息就已经全部获取。

从原理上而言，所有六条字节码都采用了以上模式来处理。相对有技巧性的实际上是写入指令中写入值的干扰：在 JVM 实现中，long 类型与 double 类型和其他类型都不一样，在栈中占用两倍的空间，另外就是*store 字节码中我们所需信息在栈顶下二三位，这使得栈操作相对比较有趣。

*store 栈操作：
首先是一般的类型（不是 long 或 double）
*store 栈上结果（消耗三个元素）
..., arrayref, index, value → ...
..., arrayref, index, value
DUP_X2（将 value 往栈深处处理）
..., value, arrayref, index, value
POP（除去 value）
..., value, arrayref, index
DUP2_X1（将栈顶复制后插入 value 之下恢复现场，同时准备好 static invoke）
..., arrayref, index, value, 【arrayref, index】（【】框住的就是函数消耗掉的）

类似的，稍作改变，就可以将其变为适配宽 value 结果：
..., arrayref, index, valueW
DUP2_X2

..., valueW, arrayref, index, valueW
POP2
..., valueW, arrayref, index
DUP2_X2
..., arrayref, index, valueW,【 arrayref, index】

另外，我也在信息记录方面做了一定的优化，考虑到此工具运行后，JVM 中用户程序每一次相关字节码运行都会调用相关 log 方法, 每次在 log 方法中拼凑输出字符串显然相当耗时，那么这里我们采用不即时输出，而将 4 个关键内容保存在内存中的策略。这样另一方面，程序 io 就不是向着终端或被重定向至某一个文件，而是写入内存了，这样工具的效率显然更高。
工具设置了 addShutdownHook，当 JVM 退出时才正式开始输出。
但这个策略也有问题，就是运行时需要额外占用空间，且程序不退出，结果就仍在内存中，相对更容易丢失数据。但就我个人使用 TamiFlex 以及在个人毕业设计中的经验，认为在动态分析时运行时效率的提高相对更重要一些。

参考资料

1. The Java Virtual Machine Instruction Set
https://docs.oracle.com/javase/specs/jvms/se11/html/jvms-6.html
2. ASM 9.2 javadoc https://asm.ow2.io/javadoc/
3. https://stackoverflow.com/questions/57398474/is-there-a-way-to-swap-long-or-double-and-reference-values-on-jvm-stack
4. TamiFlex https://github.com/secure-software-engineering/tamiflex
5. 毕业设计 DyDy