第250页 7.3 即时编译器 7.3.1 概述

RednaxelaFX (Script Ahead, Code Behind)
在读 HotSpot实战

章节名：7.3 即时编译器 7.3.1 概述
页码：第250页 2014-04-27 04:57:56

> 通过-Xcomp选项则可以让虚拟机以编译模式运行。
>
> 引自 7.3 即时编译器 7.3.1 概述

很多人会误解-Xcomp的实际作用，而书中也只有这么一句而已。
我之前一篇笔记里有举例子说明HotSpot就算在-Xcomp模式下仍然可能（而且非常可能）用解释器来执行方法的：http://rednaxelafx.iteye.com/blog/1038324

> HotSpot虚拟机中常见的即时编译器包括客户端编译器和服务器端编译器。它们也分别被称为C1编译器和C2编译器。C1编译器能够做一些快速的优化；而C2编译器所做的优化会耗费更多的时间，但能够产生更高效的代码。从JDK6开始HotSpot加入了多级编译器，解释器可以和C1、C2编译器一起协同运行，在JDK7 -server模式下默认启用多级编译器。
> 清单7-16定义了虚拟机中的编译等级，具体如下所示。
> ■ 第0级：CompLevel_none，采用解释器解释执行，不采集性能监控数据，可以升级到第1级。
> ■ 第1级：CompLevel_simple，采用C1编译器，会把热点代码迅速的编译成本地代码，如果需要可以采集性能数据。
> ■ 第2级：CompLevel_limited_profile，采用C2编译器，进行更耗时的优化，甚至可能根据第1级采集的性能数据采取基金的优化措施。
> ■ 第3级：CompLevel_full_profile，采用C1编译器，采集性能数据进行优化措施（level+MDO）
> ■ 第4级：CompLevel_full_optimization，采用C2编译器，进行完全的优化。
>
> 引自 7.3 即时编译器 7.3.1 概述

书中第250页这段文字不幸几乎每句都有错…orz

后面第251页也有一句：

> 编译策略甚至也支持对第0级展开分析，若C1编译器在以十分缓慢的速度生成第3级代码时，而C2队列又很小，则完全可以选择对解释器展开性能分析。
>
> 引自 7.3 即时编译器 7.3.1 概述

结合这句，书里对第0层的说明基本正确。在多层编译模式下，HotSpot的解释器确实只是选择性对方法做性能分析。书中提到了其中一个条件，而还有一些别的条件会触发解释器做性能分析，例如第3层C1编译失败但第4层C2仍然有机会编译该方法的话。

第1层是用C1编译，但是不做任何profiling所以不采集任何性能数据。一个Java方法一旦进入这一层执行基本上就一直停留在此了；只能通过逆优化回到第0层的解释器里，而无法向更高层迁移。用这层编译得到的代码跟在Client VM里的

C1编译的基本一致；不过在JDK8的C1里新实现的消除数据边界检查和循环不变量外提优化只在Client VM的C1才做，而在Server VM的多层编译模式下不做，也就是说这个第1层比Client VM里的C1会稍微少做一点点优化。

第2层不是用C2而是用C1编译。它不使用任何性能数据来引导优化。它只是比第1层多了方法入口和循环计数器的代码而已，并且可以升级到更高层。

第3层是用C1编译，但它也不使用任何性能数据来引导优化。其实HotSpot VM的C1是故意比较保守不使用性能数据来优化代码的，以便代码更稳定。这一层编译在第2层的基础上增加了采集更细致的性能数据的代码，包括空指针检查、条件分支、虚方法调用、类型检查等。最近还新加了参数类型和返回值类型profiling。同时这一层编译会禁用部分优化，例如消除条件表达式和合并基本块这俩都不做。采集性能数据的额外开销和禁用部分优化带来的性能下降使得第3层会比第2层显著的慢，不过这还是比解释器采集性能数据要快多了。

第4层是用C2编译做完全的优化没错。

另外就是多层编译模式在JDK8的HotSpot Server VM才默认开启，在JDK7是没有默认开启的喔。原本是计划要在JDK7就默认开启，但Oracle内部做性能测试的时候发现在一些重要的测试中开启多层编译反而变慢了，所以推迟了。而JDK8用了个治标不治本的方式解决了那几个测试中性能下降的问题，所以默认就打开了。大家能猜出性能下降的原因不？>_<

我之前也有写过两篇笔记提到了HotSpot的多层编译系统：
http://hllvm.group.iteye.com/group/topic/39806#post-260654
http://rednaxelafx.iteye.com/blog/1022095

引用我的笔记里的：
在多层编译模式下，HotSpot Server VM（此时应该叫做HotSpot Tiered VM了）会同时用上解释器、C1和C2：
Tier 0：CompLevel_none，解释器
Tier 1：CompLevel_simple，C1的正常编译（没有任何profiling）。这跟在Client VM里C1的工作模式几乎一样。进入该层的方法无法继续升级到更高层，除非先逆优化到解释器里。
Tier 2：CompLevel_limited_profile，C1带基本profiling的编译（有方法调用和循环计数器，跟解释器用的计数器的位置和作用一样）。可以升级到更高层。
Tier 3：CompLevel_full_profile，C1带所有profiling（包括空指针检查、条件分支、虚方法调用、类型检查等。最近还新加了参数类型和返回值类型profiling），同时禁用少量优化。可以升级到更高层。
Tier 4：CompLevel_full_optimization，C2编译。