64位环境中的Java(下) PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/145/2021_2022_64_E4_BD_8D_E7_8E_AF_E5_A2_c104_145478.htm 内存管理器也有必须面对的问题。虽然具有较大的堆对性能有益，但为了使堆管理算法伸缩自如，这也加重了JVM的负担。虽然始终都有堆碎片的问题，但在堆较大时，这一问题就很严重了。编译器和优化器 为了实现快速启动，许多JVM选择在开始时首先解释Java字节码，在随后运行时再对这些字节码进行编译。然而，JRockit首先使用JIT（Just In Time）编译器编译代码。虽然启动时间稍长，但这样可以使应用程序能够从一开始就提高性能。为了实现快速启动，WebLogic JRockit不使用所有可能的编译器优化。虽然使用所有编译器优化可能会在应用程序执行的初始阶段获得较高性能，但在启动时间上的额外延长也被认为是不必要的。 从应用程序性能的角度考虑，使用所有优化去编译所有方法也是不必要的，因为编译时间也是应用程序执行时间的一部分。因此，不仅WebLogic JRockit不会在启动时完全优化所有方法，而且在整个应用程序运行期间，也会保留大量的方法不被优化。WebLogic JRockit仅选择改进后能够最大限度地提高应用程序性能的函数，然后仅对这一少部分方法进行优化。 WebLogic JRockit有两个各不相同但可以协同操作的代码生成器：JIT编译器和优化编译器。如图1所示。大多数方法只能遍历图表的左半边。某些选择方法将会利用优化编译器。图1. BEA WebLogic JRockit有两条代码编译途径。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com