Java 程序优化与数据竞争检测的研究

王依多 *
WANG Yi-duo

摘 要 Java 程序语言由于具有一定的安全性和高效性，在不同平台都得到广泛应用。从服务端到移动电话平台的利用和优化，Java 程序的优化对系统的运行方式具有重要作用。所以为了增加该程序在利用方式中的较大需求，实现广泛的应用模式，在本文对 Java 程序进行优化设计，并对数据竞争检测方式进行研究。

关键词 Java；程序优化；数据；竞争检测

doi：10.3969/j.issn.1672-9528.2015.12.020

引言

在互联网平台建立方式中，要实现应用程序的跨平台要求，就要利用 Java 语言对编程进行解释，从而实现重要的应用手段。Java 编程语言在使用形式上一直具有较高的使用率，随着 Java 应用范围的不断扩大，提升该程序的性能方式成为主要的研究领域，特别是对程序语言在错误发出形式上实施检测的优化方式。

* 渤海大学信息科学与技术学院 计算机科学与技术（软件开发） 辽宁省锦州市 121000

1 静态与动态优化

1.1 Java 的即时编译优化

JIT 技术的出现，提升了 Java 程序的性能，改变了人们对 Java 程序性能差的想法。在这种动态编译执行下，Java 程序的优化工作主要体现在两方面。根据字节离线中的优化形式、 JIT 技术的优化运行，为了保护 Java 中字节码在平台中的无关性，就要对 Java 中产生的字节码空间进行优化。

程中也会出现各种性能问题。随着现代人对于计算机程序运行性能的要求越来越高，也使得程序性能测试技术得到了快速的发展。JPDA 平台就是一种快捷、简单的 Java 程序测试平台，传统的测试平台在测试过程中需要在程序段内加入标志程序，因此会引发各种各样的代码污染情况，导致程序运行性能测试受到明显的影响，测试结果与实际结果之间偏差较大，但 JPDA 则可以有效避免这一问题。JPDA 平台在发展过程中也逐渐完成并转化成为 TPTP 平台，该平台是以 Eclipse 为框架，进一步简化性能测试工作流程。而在 Java 程序性能检测过程中主要是利用相同代码的运行情况进行测试，其有助于检测结果之间的对比。本文所研究的测试代码主要是将 Obj 代码段放入 for 循环当中创建和循环外创建的程序运行性能，根据程序循环运行不同次数作为参照，分别测试了一千、五千、一万、一万五千次时两个程序运行的时间和内存的消耗情况，验证了当 Obj 对象被创建在 for 循环以外时 Java 程序运行的时间更少、消耗的内存资源更少。由此可知在对 Java 程序进行设计的过程中，应该充分利用内存回收机制，减少对象重复创建次数，有效降低重复内存资源的利用，使整个程序的运行性能得到提高，并保障虚拟机本身具有较为充足的运行内存。

结语

具备编程优势的 Java 语言在现代智能程序设计领域得到了较为广泛的应用，其简洁的设计语言使程序设计更加方便和快捷。但是，由于智能设备在运行过程中所占据的内存资源量较大，使得计算机或移动端设备资源相对缺乏，因此在设计 Java 程序时，应该以更低的资源消耗量、更高的程序运行效率为准则。JPDA 平台是目前测试 Java 程序性能的重要技术，其同样能够实现便捷的测试，无需加入单独的标志代码，对 Java 程序不会造成污染，进而有效提升了测试时的准确性和稳定性，因此在现代 Java 软件性能测试当中应进一步扩大 JPDA 平台的应用范围，并不断加深对该平台技术的改革，进而实现更加高效的 Java 程序性能测试。

参考文献

[1] 耿利祥，陈钱，钱惟贤．改进的联合概率数据关联算法（JPDA）对红外目标与诱饵的辨别 [J]．红外与激光工程，2013（02）：31-32.

[2] 胡勇．基于 JPDA 的 Java 软件性能测试 [J]．电子技术与软件工程，2013（15）：28-29.

[3] 肖云鹏，等．基于 Java 的嵌入式软件性能改进 [J]．重庆邮电大学学报（自然科学版），2014（02）：102-103.

（收稿日期：2015-12-23）

化。但为了保护 Java 字节间与平台产生的无关性，在有限空间表现形式上，就要提升性能速度。因为 Java 在动态语言上具有较大特性，所以说，Java 程序的实现主要在 JIT 技术运行下进行优化的。因为 JIT 技术在编译策略中能够实现选择性编译，能够对部分代码实施动态性优化，根据整个程序之间的路径代码重新选择，然后实现动态性的权衡方式；能够实现反馈优化，利用插桩中的相关信息进行优化、指导，在函数内联、对代码进行布局、对指令进行调度等多种技术进行优化，从而保证 JIT 编译器的级别优化。

1.2 Java 的静态编译优化

与及时编译相比，静态编译对时间没有较大影响，利用计算资源能够实现较大丰富性。所以在进行程序的分析与优化期间，就要对性能产生的效益进行优化。在比较高的性能计算应用中，与静态编译具有较大关系，在优化过程中，不仅能体现更大的展开空间，实现多种过程的分析程度，还能提升程序化的性能水平。对于 Java 静态编译器来说，它不支持动态性的载入方式，在优化期间能对所有对象进行。静态编译器在表现形式上具有多种优势 [1]。静态编译器能生成可执行文件，方便程序的调试过程。特别是编译器在自身上出现错误时，利用 JVM 来调试更加困难。静态编译器产生的执行文件能对 Java 程序中相关的知识产权进行保护，特别是字节码容易被翻译。静态编辑器在运行之前，能对程序成分利用并优化，例如在开销比较大的过程中进行优化，由于 JIT 对编译时间比较敏感，所以最简单的优化方式要利用轻量级来进行。保证编译在开销、性能效益之间的平衡性，从而实现合理的优化形式。静态编译器在 Java 程序中启动的时间比较快。可执行文件占用的时间也比较小，由于 JVM 有很多功能模块，本身具有较大内存，所以在资源储存形式上，就要在嵌入式系统中实现较大的竞争优势。它能对 Java 程序中程序在分析以及检测上实现更大的安全性，能够体现静态编译器的扩展性和有效性。

2 Java 的特性优化

2.1 虚函数调用恢复

在 Java 程序语言中，抽象是它发展中的一种主要特性，它能对运行期间的对象产生的动态性进行调用并选择。虚函数的使用增加了程序编写的方便性，但增加了编译器在分析上的难度。虚函数能对同一类成员函数进行重载等。如果一个基类指针、引用在一个对象中继承时，就要调用继承类中的函数版本。对 Java 虚函数进行调用分析期间，首先，对虚函数中的调用信息进行识别，由于 Java 不能进行指针操作，所以就不能进行指针调用，在虚函数调用方式上就要利用间接性的虚函数进行调用。然后根据编译过程中存在的类型表分析对象的初始化状态。为了在分析中实现更大的准确性，就要在各个阶段中构建相关的层次关系，根据在编译过程中产生的类型表，找出具有的继承关系，并根据继承关系中的相关信息建立一个完整的数据结构进行保存 [2]。在处理期间，调用点是虚函数发挥调用方式利用的，当一个初始化类型中没有一个子类，该函数在调用期间，就要很据保守解析策略进行分析。如果在这种类型表现形式上判断出该对象为实际类型，首先应根据对象确定出一定的初始化类型，然后根据相关函数进行调用。将已经调用的函数直接利用，并实施更新形式。

2.2 冗余同步删除

Java 能够实现语言的多种线程支持方式，程序员能够利用 Java 中已经提供的同步结构、原语对共享的数据进行安全操作等形式。但由于线程环境比较复杂，程序员为了保证编程的方便性，就会利用保守机制来实现同步操作，从而增加了大量的冗余同步操作。所以在这种操作方式中，为了删除线程中冗余的同步现象，提高程序在运行期间的性能，就要利用优化技术来实现。首先，对逃逸进行分析，它主要对每个对象中产生的逃逸状态进行分析。对于线程级的逃逸对象来说，它只对线程级的对象进行同步操作，其他对象执行冗余操作。但尽管其他线程在操作中没有被编写，但也能存在冗余同步现象，所以在分析转化过程中，就会对程序中的被同步对象的逃逸状态进行分析。在逃逸分析中，在实现方式中主要体现在两方面。一种是过程内分析，它能对每个函数中的语句进行分析，并记录对象中存在的逃逸信息。它是单个过程的分析形式，在每个过程中生成一个连接图，然后对连接图中的节点属性进行分析，从而对每个节点中的逃逸信息进行分析。一种是过程间分析，它主要利用调用图对函数的调用关系进行识别，从而分析出各个对象中的存在的逃逸信息。

2.3 对象内联

面向对象特性提高了 Java 中的编程效率和程序的复用性。但一个完整的应用程序在分解方式上形成了多个小类，并降低程序在运行期间的性能。在运行期间，分配出的大小对象在相互引用中不仅增加了对象区域的访问形式，在内存分散形式上，相关的小对象也降低了在程序出现中的局部性 [3]。对象内联方式的优化主要将相关的对相进行分配，然后在连续的内存空间中提高缓存的布局性，并利用相关地址计算出对象区域中访问次数。首先，利用程序中全局调用图，统计出对象域的基本属性，然后根据对象中的基本属性，识别出对象能够优化的保存区域，最后，对对象实现合并分配并进行相关区域的载入处理。在载入处理过程中，一般是利用载入折叠方式来实现的，需要对各个区域进行访问。当访问子对象区域期间，就要利用父对象中的首地址、偏移量进行计算，从而保证两个区域在载入、操作等形式上的合并以及操作等。

3 动态数据竞争检测

3.1 动态数据竞争检测技术

动态数据竞争检测是利用变量的获取、准确的消息，对访问、共享中发生的关系和信息在执行方式中实现路径中数据竞争错误。在检测策略中主要有三种方式。首先，对于锁集合检测技术来说，它主要将所有的共享内存消息进行锁定，当两个线程访问一个内存位置期间，在这种操作形式中并没有产生公共锁，在这种情况中说明两个访问事件存在的数据竞争是错误的。所以在这种访问期间，如果多个线程访问同一个共享位置，在访问期间就要获得同一个锁，尽管在多个线程上进行访问，也不会存在错误的数据竞争现象。为了减少检测存在的失误，就要对内存分为四个组成部分，设定内存访问位置集合、线程集合以及锁集合，根据每个、访问类型的不同进行读写操作。对于 Happens-before 的数据竞争检测，它主要在分布式系统中形成建立的，该工具在运行上主要体现在两方面，一种是对插桩代码中的相关指令、同步原语进行执行，一种是对全部的访问对象在原始数据信息中进行记录、维护，从而提高数据竞争的检测性能。

3.2 三层粒度的数据竞争检测

在 Java 中，对于模型对象来说创造了更大的严格性 [4]。只能对引用对象中的数据进行修改，首先为对象创建一个线程，在开始创建中就只能对创建的线程进行访问，如果该对象被引用到其他线程对象上去，引用的对象就从局部现象变为全局现象。如果访问了全局对象，就会导致数据竞争错误，所以在这种方式中就要对全局对象进行访问并监控。三层检测粒度不仅能对相关的区域对象进行检测，还能对所有对象中的数据进行检测。当一个对象从局部转变全局对象期间，检测器在这些全局对象中就会建设一定关系，然后将相关节点作为监控对象，如果对象局部的，就不需要进行处理；如果对象已经被监视，检测器就会通过读取等操作对潜在的数据竞争错误进行检测，如果发生错误提示信息，就要更改已经读取的历史信息；如果对象没有被监控，检测器就要为对象的拥有者进行监控，如果已经被监控，就要进行读取操作。

3.3 实现

首先，对向量时钟的实现方式，它主要能够维护 Happens-before 关系信息的中的数据结构。当同步指令在操作起价，向量时钟就会走动。在实现方式上一般是利用数组来完成的，因为每个数组都代表一个程序中的线程，所以在线程中，大量的冗余元素都要进行一定的内存开销。在竞争检测器中，为了能对全局对象中的元数据进行访问，对象数据结构被修改，就要对修改后的对象进行布局。对于垃圾回收模块，它能够保证程序员在修改形式上由于失误造成内存泄露等现象。垃圾收集器会在某个时间段内对运行空间中的垃圾进行处理，从而达到释放空间的目的 [5]。利用技术 GC 在程序中实现执行方式，对实时程序的实现具有重要意义。对于追踪式垃圾收集算法，它能够保障计数算法的不足现象。对于分代式垃圾收集算法，由于年强垃圾存在比较频繁，所以主要减少年轻垃圾的收集次数。对于对象拥有关系树，当一个对象从局部变为全局发展期间，会产生相关节点，竞争检测器会将这些节点作为全局状态，从而对这些对象进行构建、回收等形式。

3.4 性能测试

竞争检测器的测试数据，主要根据在运行期间的性能、检测精度来实现的，并能根据其他检测器进行比较分析。首先，对于运行性能的测试，在测试期间，每个程序的运行都要保证五遍，除去最大值和最小值，最后根据三次时间求出平均值。如果性能退化，表示每个检测粒度在运行期间会产生较大比率，出现的比率数越大，说明数据竞争检测器运行的性能就会比较差。主要由于数据竞争检测器中产生的插桩代码比较小，所以为了提高数据竞争器的性能，就要在时间消耗、访问内存信息中进行维护、处理。对于检测精度来说，它主要能为数据竞争检测器提供良好的检测依据。对于在对象层产生的粒度进行检测，由于竞争错误报告主要在对象层上，其他粒度的数据竞争检测都会出现在区域层来报告数据是否错误。

结论

Java 程序语言在不同平台发展中都得到有效利用，能够利用第一语言实现多线程的语言支持。随着多核处理器的不断发展和普及，要实现 Java 程序的优化和相关数据竞争的检测，就要对相关问题进行探讨，不仅要研究静态技术在编译上的优化，还要提高数据在竞争检测工具的应用性，从而实现技术领域的不断创新和发展。

参考文献：

[1] 宋东海，陈二虎．一种基于调用链的 Java 程序数据竞争静态检测算法 [J]．舰船电子工程，2013,12:53-57.

[2] 宋东海，贵可荣，张志祥．一种基于类的 Java 多线程程序数据竞争静态检测算法 [J]．计算机工程与科学，2014,02:233-237.

[3] 张昱，郝允允．Java 程序数据竞争的增量式检测 [J]．西安交通大学学报，2009,08:22-27+58.

[4] 杨哲憨．Java 语言的程序漏洞检测与诊断技术 [D]．复旦大学，2012.

[5] 魏鹏．多线程程序中数据竞争故障的动态检测技术研究 [D]．华中师范大学，2009.

（收稿日期：2015-12-23）