多线程故障分析及解决方法

刘玉璇[1]　宫云战[1]　杨朝红[2]
(1. 北京邮电大学计算机科学与技术学院, 北京　100876; 2. 装甲兵工程学院信息工程系, 北京　100072)

摘　要: 多线程技术在近年来得到了广泛的发展和应用,对其的测试分析具有重要的意义。本文提出了针对多线程的一些故障模式,介绍了现有的对多线程故障分析的几种方法,并对他们进行比较,以利于以后新的检测工具的生成,促进软件测试在多线程技术方面的发展。

关键词: 多线程; 故障模式; 静态分析
中图分类号: TP311.5

引　言

近年来并发多线程的程序设计技术得到了广泛的发展和应用,从多种并发多任务的程序设计语言,到各种采用了线程技术的系统,美国电气与电子工程师学会(IEEE)也推出了有关多线程程序设计的标准 OSIX1003.4a。特别是在各个流行操作系统中,采用了线程作为基本的调度单位,其应用程序编程接口 API 中也提供了优选线程操作的用户程序接口,所有这些都无疑会促进多线程技术的发展和应用。

但与此同时,多线程编程也带来了很多特定的优化和检错问题,尤其是数据竞争和死锁问题。粗粒度的保守锁机制容易写出正确的代码,但是会带来不必要的锁操作或是锁住不需要保护的数据而降低了并发度,进而导致性能较低。另一方面,高度优化的细粒度锁机制有很好的性能表现,但是这种程序可能会缺乏稳定性,容易出错,比如保护不够就会引起数据竞争,当同步涉及到多个锁时由于获得锁的顺序而引起死锁问题。本文提出了针对多线程的一些故障模式,介绍了现有的对多线程故障分析的几种方法,并将它们进行了对比。

1　多线程的故障模式

通过对大量多线程程序例子的研究,本文提出了以下几种多线程故障模式。

1.1　不一致同步

被多线程访问的可变状态通常需要同步保护,常见的方法就是用对象锁保护这个对象可变域的访问,域的访问被包含在同步方法或者同步块中,进去同步区域必须获得接受对象的锁,如果对对象域的访问没有取得对象锁就构成了不一致同步。

1.2　无条件等待

当进入一个同步块时,等待操作会立刻无条件的执行,对 wait() 方法的调用不是在条件控制流中,线程所等待的条件已经发生,这个线程将无条件不确定地等待下去。

1.3　和锁有关的操作

(1) 一个对象占有锁想要请求另外的锁,而另一个线程正好占用这个锁却在请求第一个线程锁拥有的锁,因而产生了死锁。

(2) 声明了一个 Lock() 的对象并且上了锁之后,如果在方法中没有任何地方释放该锁的话,那别的线程就用不到这个锁了,所以最后需要在 finally 语句块中进行释放。

(3) 声明了一个 Lock() 的对象并且上了锁之后,如果在 try 内部发生异常,而在所有异常路径当中都没有释放锁,则需要在 finally 语句块中显式释放。

1.4　其他一些故障模式

(1) 双重验证即是指对一个对象进行了两次判断其是否为空,但这种方式在多线程应用中也会引起错误,过程中可能有多个线程取到了没有完成的实例,并用这个实例作出某些操作。

(2) 无论是继承了 Thread 还是实现了

收稿日期: 2007-05-15
基金项目: 国家"863"计划(2006AA01Z184)
第一作者: 女,1983 年生,硕士生
E-mail: xuan9806@gmail.com

Runnable的线程都应该定义一个run()方法,没有任何操作的空的run方法只能是浪费时间。

(3) 程序要启动的时候正确应该是Thread.start(),而不是直接调用run()方法。

2 多线程故障分析的方法

理想的程序分析具有可靠性和完备性:可靠性指的是检测工具在程序有错误时必须报告错误;完备性指的是检测工具只报告真实错误,没有假错误。文献[1]显示对没有过程和递归的rendezvous形式的同步程序,如果考虑所有可能的交互和执行路径,同步分析是个NP-complete问题。文献[2]对带过程程序的扩展分析显示任何上下文敏感、同步敏感的分析是不可判定的。因此,实际的并发分析工具都是可能覆盖所有的错误但是会有假错误的近似算法。下面就将分别介绍多线程故障分析的几种方法,它们在使用难易、分析精度、性能和覆盖率上各有特色。

2.1 动态分析

所谓动态分析,就是通过运行软件来检验软件的动态行为和运行结果的正确性。在运行期针对确定的可执行路径检测,拥有变量和别名的准确信息,因而是一种精确的检测方法。但是多数动态方法对检测对象往往不加筛选就做插装,监测代码会占用很多的运行时间,尤其是对于多线程并发程序,无法完全模拟所有可能发生的情况,动态分析由于了解运行值可以在分析成本不高的情况下得到准确的分析结果,但是它的分析结果往往依赖特定的输入、调度和执行路径,动态方法固有地只能检测特定执行,容易出现更多的漏报,一般的动态方法都是不可靠分析,有的动态分析也会产生假错误。而且动态分析看到的是问题本身而非征兆。

2.2 静态分析

静态分析不编译运行程序,而是通过对程序源代码进行分析以发现其中的错误。静态分析的目标不是证明程序完全正确,而是在程序运行之前尽可能多的发现其中隐含的错误,提高程序的可靠性和健壮性,提高软件的生产率。下面分别介绍了几种不同的静态分析方法。

2.2.1 冗余同步操作的静态删除算法 同步操作是多线程程序一个较大的开销,对于多处理器系统,它意味着流水线的断流、相关联的Cache缺失以及总线的锁定,所有线程必须等待同步结束。因此冗余同步的删除就成为一个重要的优化问题。该算法扩展了传统的Escape分析,跟踪了Escape对象在不同线程中的流动;设计了高精度的上下文敏感和流敏感的别名分析,充分考虑控制流结构和线程交互时序对对象逃逸状态的影响。

该算法基本思想为首先扫描程序并构造线程间调用图(ICG)和控制流图(CFG)。统计程序的线程个数,如果是单线程程序则删除所有同步操作。然后利用调用图和控制流图为Escape对象扩展访问的线程和同步信息。在ICG上自底向上分别对单个过程作数据流分析,随着语句的执行发生变化,当过程中出现函数调用语句时,信息就开始自底向上传播,然后结合当前方法的上下文信息在当前方法中继续做过程内部分析。

2.2.2 静态数据竞争检测算法 文献[3]提出了一种自动精确的数据竞争静态检测算法,其基本思想为将竞争问题分解为跨线程的控制流分析、以全局对象为中心的访问事件搜集以及时序关系的约束求解。分析过程应用了流敏感、上下文敏感别名分析,充分考虑了时序关系对竞争问题的影响,静态搜集了交互原语形成的访问事件发生序,提出了以对象为中心的检测,结合Escape分析缩小了检测范围,提高了分析效率。

该算法首先扫描整个程序构造线程间调用图(ICG),控制流图(CFG)和时序约束图(TCG),分析程序是否是单线程程序,如果是单线程程序则退出分析,否则对ICG和CFG自顶向下做深度优先遍历,搜集每个程序点的锁对象集合并记录在当前方法中,然后为各个线程节点收集全局量的访问信息,最后按照竞争定义对每对线程节点记录的全局变量的访问事件做出竞争判断。

2.2.3 静态竞争的OUG算法 文献[4]提出了一种静态竞争的OUG算法。该算法以对象为中心,构造了复杂的OUG(object use graph)。抽象不同线程对同一对象的访问事件,每个全局对象都必须各自对所有方法的CFG做符号执行,删除和该对象无关的操作后才能得到自己的OUG,也就是相当于别名信息和时序约束抽象的综合。

2.2.4 基于锁集合的动态数据竞争检测方法 文献[5]提出了一种基于锁集合的动态数据竞争检测方法。多线程并发程序中的数据竞争发生在两个或多个线程访问了同一个内存位置而没有时序限制,并且至少有一个是写操作。竞争有时是共享数据和

通讯的一种方式,但多数情况下属于程序错误。存在竞争错误的程序在相同的输入和相同的同步执行顺序下也可能出现不确定的执行结果。这种算法基本思想是通过用户提供指导来获得每个区间读写共享变量的集合,在存储一致性模型的框架模型基础上,通过定义增强发生序,在程序中找到许多读写数据竞争,从理论上保证了数据竞争检测的正确性。

2.2.5　模型检测　文献[6]介绍了模型检测的方法。模型检测是一种验证有限状态的并发系统的方法,它在硬件设计、通信协议、实时进程的验证方面取得过很大的成功,由于使用了巧妙的数据结构(BDD),模型检测的效率得到了极大的提高。其基本思想是对于有限状态的系统构造状态机或者有向图等抽线模型,再对模型进行遍历以验证系统的某一性质。

模型检测的难点在于如何避免状态空间爆炸。为此人们提出了多种方法。其中符号化模型检测方法是将抽象模型中的状态转换为逻辑公式,然后判定公式的可满足性。还可将模型转化成自动机,同时将需要检查的公式转换为一个等价的自动机,再将此自动机取补。这两个自动机的积构成了一个新的自动机,则判定模型是否具有某一属性的问题就转换成检查这个新自动机能接受的语言是否为空。

2.2.6　基于规则的检查　文献[7]提出了基于规则检查的方法。在面向不同应用的程序中,常常隐含着各种不同的编程规则,例如要求在使用某一共享变量时最受"使用前先加锁,使用后解锁"的规则,所以本文提出了基于规则对多线程程序进行分析的方法。采用基于规则进行分析的系统的结构是:首先由一个规则处理器处理规则,将其转换为分析器能够接受的内部表示,然后再将其应用于程序的分析。

此算法的具体实现为:读入源程序后首先获取词法和语法规则,定义抽象语法树的结构,编写相应的语法规则,进行词法和语法分析。在分析抽象语法树后得到一些辅助分析信息如:控制流图、定义使用链、相关函数的位置、相关变量的位置等。然后针对各种模式错误的分析算法,分析错误的模式,给出合适的错误模式的描述,给出分析结果。具体过程参见图1。

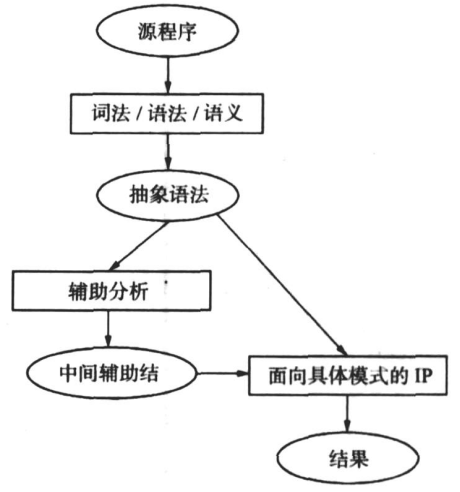

图1　基于规则的检查算法流程图

Fig. 1　The flow chart of the checking algorithm basing on rules

从表1可以看出静态分析中这几种方法的优缺点。

表1　几种静态分析方法的比较

Table 1　Comparison of the static analysis algorithms

方法	优点	缺点
冗余同步操作的静态删除算法	同步删除率高于现有的分析工具。可以在有效的时间内对中小规模的程序取得精确的分析结果	由于流敏感、上下文敏感分析固有的指数级时间和空间复杂度,这种精确分析很难扩展到大程序分析
静态数据竞争检测算法	可以在有效的时间内对中小规模的程序取得精确分析结果,精确度超过了现有算法	测试程序规模有限,需要进一步对实际应用规模的程序进行测试分析,没有提供区分竞争错误和良性竞争的机制
静态竞争的 OUG 算法	开销小于静态数据竞争检测算法,分析效率很高	形成很多重复的控制流信息和数据流信息,精确度较低
基于锁集合的动态数据竞争检测方法	算法应用的平均减速比达到了 3.14	采用用户指导,增加了用户负担. 应当采用代码插入技术来改进,使得无需用户指导,易于用户使用
模型检测	可以分析各类属性,更善于发现系统的复杂行为	分析成本昂贵,很难适用到大规模的实际应用
基于规则的检查	能够依据不同的规则对不同的系统就行分析,发现大规模程序中的潜在错误	受到规则描述机制的局限,只能分析特定类型的错误

3 结论

多线程存在多种故障模式并已有一些解决方法,且分别具有优缺点。目前基于故障的测试工具具有很好的发展前景,随着人们对软件故障认识的不断深入,多线程的故障模型也会越来越完备,并更加符合实际,我们还有更多的问题需要研究,包括故障模型的准确程度,测试的准确程度和测试的自动化程度。下一步工作将更详细地研究多线程的故障模式,并可对多种分析方法进行结合,从而生成新的测试工具。

参考文献:

[1] TAYLOR R. Complexity of analyzing the synchronization structure of concurrent programs[J]. Acta Informatica, 1983, 19: 57 - 84.

[2] RAMALINGAM G. Context-sensitive synchronization sensitive analysis is undecidable[J].

[3] 吴萍, 陈逸云, 张健. 多线程数据竞争的静态检测[J]. 计算机研究与发展, 2006, 43(2): 329 - 335.

[4] 章隆兵, 张福新, 吴少刚, 等. 基于锁集合的动态数据竞争检测方法[J]. 计算机学报, 2003, 26(10): 1217 - 1223.

[5] von PRAUN C, GROSS T R. Static conflict analysis for multi-threaded object-oriented programs [M]. ACM SIGPLAN notices, 2003.

[6] 戎玫, 张广泉. 模型检测新技术研究[J]. 计算机科学, 2003, 30(5):101 - 104.

[7] 杨宇, 张健. 程序静态分析技术与工具[J]. 计算机科学, 2004, 31(2):171 - 174.

The fault pattern analysis and arithmetic of multi-thread program

LIU YuXuan[1] GONG YunZhan[1] YANG ZhaoHong[2]

(1. College of Computer Science and Technology, Beijing University of Posts and Telecommunications, Beijing 100876;
2. Department of Information Engineering, Academy of Armored Force Engineering, Beijing 100072, China)

Abstract: Multi-thread technology has got extensive development in applications for recent years, it is important to get a full analysis of this technology. This paper introduces basic fault patterns of multi-thread program, and then summarizes several algorithms for multi-thread program analysis along with comparison of their advantages and limits. These multi-thread program analysis benefit a lot for the formulation of the new testing tools, and will promote the development of software testing in multi-thread technology field.

Key words: multi-thread; fault pattern; static analysis