WCET分析:建立实时系统可靠运行的技术*)

金永贤　赵建民　陈会羽
(浙江师范大学计算机科学研究所　浙江 金华 321004)

摘　要　实时系统开发必须强调时间的重要性,为了保证系统安全运行,需要验证系统是否在时限内完成各个任务,因此,当设计和验证实时系统时,了解运行在系统中代码的最坏执行时间(WCET)是非常重要的。WCET静态分析(简称WCET分析)计算实时程序最坏执行时间的上界,而上界被用来为应用程序的任务分配正确的CPU时间,它们也是可调度分析工具的输入,因此,WCET分析是可靠建立实时系统安全正确运行的基础。介绍了WCET分析的概念,指出了传统测量存在的缺陷,剖析了WCET分析研究的关键技术,探讨了目前存在的问题和今后的发展方向。

关键词　可预估性,WCET分析,程序流分析,上界,Flow facts

实时系统设计时必须满足环境的时间要求,每个实时任务必须满足时限,即任务必须在截止期之前完成,否则实时系统会失败。衡量实时系统及时性的最重要的参数是内部执行代码(任务)的最坏执行时间(worst case execution time,简称WCET),而WCET分析目的是在程序用于系统之前,提供有关程序最坏可能执行时间信息,因此,WCET分析对于保证实时程序质量和确保实时系统可靠运行都是非常有意义的。一个实时程序的WCET取决于可能的程序流(如循环、子程序调用等)、特定的目标硬件特性(如caches、pipelines等)和输入参数的可能值。

WCET分析概念[1~4]在十多年前被提出来,关于WCET分析最早的一篇论文由Kligerman[1]所写,这篇论文讨论了为了计算实时任务执行时间所需对程序语言(这种语言叫Real Time Euclid)的限制。最近几年,越来越多的学者从事WCET分析研究,从2001年以来每年都要召开有关WCET分析研究的国际会议,相应的WCET分析方法的研究和工具的开发已经变成了一个活跃的研究领域。本文首先介绍了WCET分析的概念,指出了传统测量存在的局限性,剖析了WCET分析研究的相关技术,探讨了目前存在的问题和今后的发展方向。由于篇幅有限,有些内容作了简化处理,有兴趣的读者可以参考本文所列的参考文献。

1 基本概念

·控制流路径(control flow path ,CFP):CFP是指具有执行约束的程序的可能执行路径集合,它是描述存在于程序执行中不同可能执行路径的集合。

·Flow facts:描述程序可能的控制流路径的信息,用这些流信息可以查找通过程序的最长执行路径。Flow facts可以通过程序结构和语义抽取。由于大部分情况程序执行行为(或状态)还取决于外部输入数据,因此这种内含的flow facts不足以计算WCET,为此,要提供附加的flow facts,这种附加的flow facts一般通过注释给出,这些注释可以包含在源代码内或通过数据文件和用户交互输入等。

·程序分析(program analysis):测定程序在没有执行前的运行特性,广泛使用的一种程序分析理论是抽象解释[5]。抽象解释就是用抽象值代替具体值完成程序的计算分析。使用抽象值代替具体值的第一个理由是可计算性,即强调在有限时间内获得计算结果。第二,获得描述所有输入的计算结果。

·WCET[6]:指实时程序(任务)执行时间的最大可能值。

·WCET分析:计算已知应用程序代码执行时间的上界,这里每条代码的执行时间可以定义为执行这条代码所消耗的处理机时间。关于WCET分析概念还要强调几点:①WCET分析计算WCET的上界,仅仅是一个估计值,不是一个精确值(当然分析结果越精确越好,这是WCET分析追求的目标);② 计算代码的WCET上界与应用环境有关。不同的应用上下文,代码的执行路径不同(实时系统对于不同的输入数据可能做出不同的反应),因而,计算结果不同;③ WCET分析与硬件平台有关。因此,WCET分析必须模拟目标硬件特征。

2 WCET分析目标

产生于WCET分析的最坏情况执行时间上界决定了系统设计者和调度工具必须为各个任务执行

*)本文工作得到计算机应用重点学科资助。金永贤　硕士,副教授,主要研究方向为实时系统软硬件协同设计、可靠性设计。

所预留的时间总量。为了使分析计算得到的WCET能够用来正确设计和验证实时系统,WCET分析应满足下列要求:

(1)有效性(valid),提供的WCET上界安全,即不能过低估计最坏执行时间。

(2)有用性(useful),计算的WCET上界尽量紧或精确,WCET分析结果越悲观(pessimism),给每个任务预留的CPU时间就越多,浪费资源的概率越大。为了使WCET分析结果有用,不能过高估计最坏执行时间。

3 动态测试方法存在的局限性

动态测试也可以获得实时程序的执行时间界,但测量结果与下列几个因素有关。①处理器初始状态影响测量值;②取决于输入数据的控制转移指令产生多执行路径;③地址取决于输入数据的存储器访问会产生不同的处理器状态。这些因素导致测试具有不同结果,而且数量呈指数级。因此,通常的WCET测试只能提供不十分可靠的WCET的下界,即这种方法只能使获得的执行时间和实际WCET的偏差在一定的范围内。目前,研究人员提出pWCET(probabilistic WCET)概念的目的是希望通过统计方法能够使WCET过低估计的风险降低到一定的限度或低于系统构件的失败率。

如果最坏情况的处理器初始状态能设置,则在以下几种情况下,测试可以获取可靠的WCET界:①小输入空间,如果输入参数值范围很小,穷举输入空间是可行的;②顺序代码,如果控制转移指令不取决于输入数据(执行路径可预估),则仅仅需要一次测试。

从以上分析可知,通过测试获得可靠的WCET上界,仅仅适用于控制流路径变化少的程序。

4 WCET分析研究的关键技术

已知特定的应用环境,实时程序的WCET取决于下列因素:(1)实时程序执行的各种可能路径(序列);(2)所有路径(序列)中,每个路径(序列)的每个动作(指令)所需时间。

由于不可判定性(undecidability),已经被证明,在通常情况下,通过静态分析计算程序的WCET是不可能的(等同于停机问题),因此,WCET分析经常使用附加流信息(即flow facts)来查找通过程序的最长执行路径。可以在不同的语言层完成WCET分析,但为了获得可靠的WCET,最好在目标代码层完成,而程序员易于在源代码层提供有关代码可行(不可行)路径信息,因此,WCET分析应把源代码提供的路径信息转换成目标代码表示。因此,WCET分析主要和下面这些技术有关。

4.1 程序流分析技术

程序流分析目的是判定程序可能流,不考虑指令或基本块的具体执行时间。程序执行期间,各个动作潜在的执行序列由程序源代码描述。实时程序设计语言结构的控制语义定义语句的可能执行序列,通常,用这些信息来自动给程序的WCET定界是不够的。研究人员通过对执行路径特性的分析,提出各种办法来描述WCET分析所需的flow facts。他们寻找程序注释(annotation)方法(注释机制用来定义最大循环范围、决定互斥路径等),来充分描述程序的执行频率和执行序列以及它们的相关性[1,3,7]。

例如,文[8]介绍通过语义分析来进行路径分析和应用符号执行(即用抽象的符号表示程序中变量的值)来识别假路径,它使用循环界和函数递归界作为flow facts注释。由于没有考虑输入变量的注释和仅使用基于变量和常量简单关系的粗糙值域,该方法准确性有限。

程序流分析结果提供有关哪些函数被调用、循环多少次、IF语句是否有相关性等信息。

4.2 路径信息转换(flow facts transformation)技术

在进行WCET分析时,将源代码说明的路径信息转化成机器代码表示是非常重要的。编译器,尤其是优化编译器,通过重新安排代码和执行转换以获得高效代码和达到优化目标,代码结构的这些变化,使得源代码表示的路径信息在机器代码中识别变得很困难。如何在目标代码中识别源代码说明的flow facts是当前WCET分析研究的热点。

目前基本上有两种途径:其一让编译器在进行代码转换时,同时转换路径信息[9-11]。在某些情况下(如high cost, 等),调整编译器或重写编译器是不可接受或不可能的,因此,所采用的方法最好是编译器无关[12],或者仅仅需要编译器有限的支持[13];其二是设法自动从机器代码或汇编代码推导出路径信息,这种方法有些困难和局限性,因为自动路径分析问题通常具有不可判定性。

4.3 执行时间建模技术

目的是建立目标硬件结构的执行时间模型。对于高性能的现代处理器需要特定的flow facts(先前指令执行情况)来精确建立应用环境相关的时间模型。由于现代处理器越来越多地应用在安全关键的实时系统中,因此具有硬件加速(speedup)特征的处理器执行时间建模是当前活跃的研究课题,也是难点所在。

高性能的实时系统不仅要利用缓冲器,而且要利用流水线处理器,其复杂性取决于目标硬件特性,在先进的计算机体系结构中,一条指令的执行时间

不仅取决于所执行的操作和操作数,而且取决于指令执行时硬件状态,因此,为了获得理想分析的结果,这一步要结合缓冲器分析[14~17]、流水线分析[16,17]等。

例如,Kim 等人[14]通过模拟数据高速缓冲器机制,大大改善了分析结果。他们目的是减少由于动态数据装载和存储指令对 WCET 分析结果的不利影响。首先,他们应用全局的数据流分析以避免动态错误分类的数据存取。其次,通过考虑控制流信息,以数据相关性分析(data - dependency analysis)来减少动态装载和存储所产生的不利影响。

Healy[16]等人介绍了模拟指令高速缓冲器的静态高速缓冲仿真(static cache simulation)法。这种方法通过分析程序可能的控制流和对每次存取指令的缓冲器行为进行静态分类,如始终命中,始终失败等,最后,时间分析器使用这些分类信息来计算执行时间上界。

4.4 计算 WCET 方法

计算阶段目的是根据程序流分析、执行时间建模分析,计算实时程序的 WCET 估计值。计算通常采用下面几种方法:基于路径、基于树和基于 IPET(implicit path enumeration technique)或者采用混合方法。

(1)基于路径[16,18,19]:通过计算程序不同路径的执行时间,找出最长执行时间的路径产生 WCET。

(2)基于树[3,4,20]:也称为时间方案[4]。它是由 Shaw 等人先提出来,这是一种计算 WCET 形式化的方法。这种方法将执行时间的计算规则应用于不同的程序结构(如 simple, if - then - else, loop, 等)和结构序列。通过自底向上遍历代表程序的语法分析树产生 WCET。这种方法概念上简单,计算成本低,但是,在处理流信息时较困难,因为计算局限于简单语句,不能解决语句的相关性。

(3)基于 IPET(IPET - based)[15,21,22]:程序流和基本块执行时间用代数和逻辑约束来表示,即程序被变换成 IPET 的约束集合,通过满足约束的最大化目标函数(即 ILP, integer linear programming)来计算 WCET。IPET 约束可表示复杂的程序流信息,允许处理全局 flow facts,这种方法是目前最流行的 WCET 计算方法。

5 存在的问题及对策探讨

提出 WCET 分析概念已有十多年了,在此期间,许多不同的 WCET 分析方法和方案被提出来,针对这些方法的分析工具实现和使用都比较复杂。前面已经分析过,实时程序的 WCET 取决于应用程序执行的各种可能路径和所有路径中每个路径的每个动作所需时间。很明显,这两个因素不仅决定代码的 WCET,而且决定了 WCET 分析的复杂性。其中,动作(指令)的可能路径取决于解决这个问题的算法和编译器编译期间处理的代码;每个动作(指令)所需的时间取决于动作执行的机器(硬件)特征和结构。下面分别讨论这两个因素对 WCET 分析复杂性的影响。

5.1 软件优化

实时程序设计者使用在非实时应用中已证明有效的算法和程序设计技术。在非实时应用中,大多数算法速度优化是追求的主要目标,时间可预估(测)不是努力的主要目标。在大多数情况中,非实时算法执行的动作是基于输入数据的,然而,输入数据相关控制判断(Input - data dependent control decisions)引起程序在不同的路径上执行具有不同的执行时间,因此,WCET 分析需要考虑的不同情况很多,从而导致 WCET 分析变得很复杂。为此在软件上要设法摆脱数据相关性影响,通过降低输入数据的相关性,以减少 WCET 分析需要考虑的路径数目,从而也降低 WCET 分析的复杂性。根据这个思路,我们可以采用"单一路径"方案[23],这种方案能产生时间可预估的程序。它的中心思想是产生性能和输入数据无关的程序,因而始终在一条并且只有一条可能的路径上执行。单一路径程序可以通过代码变换原理,以消除程序中数据相关分支语句。

5.2 硬件加速

这主要体现在分级(层)存储体系结构中,如高速缓冲器。指令或数据装入(保存在)小容量的快速存取的高速缓冲器,这样可以提高数据和指令的存取速度。哪些数据被装入、保存和取代缓冲器内容,通过启发式推测(heuristics speculation)完成,即推测将来哪些数据项被存取,使用推测判断机制导致内存存取时间变化。各种特定内存存取时间,取决于先前在高速缓冲器的操作状态,内存存取时间的变化和实际内存存取时间和过去状态的相关性导致 WCET 分析变得复杂。

与非实时应用相反,(强)实时应用主要要求时间可预估,因此,支持 WCET 分析的正确的硬件设计目标应该是时间可预估性。为了达到这个目的,通过使用对高速缓冲器内容实施绝对控制的内存分级体系,而不是依靠推测。如预取策略(prefetching strategies)使高速缓冲器内容和存取时间易于预测控制。

结束语 这里指出今后 WCET 分析研究可能的发展方向,仅供参考。(1)WCET 分析计算结果的可视化是一个方向;(2)如何更好地测定先进处理器特征的作用,即解决具有先进体系结构处理器执行时间模型问题;(3)研究与语言及硬件平台无关的可移植的 WCET 分析框架,即 WCET 分析的可移植性研究;(4)如何将静态分析(WCET 分析、调度分析)和动态测量结合在一起,即研究混合处理方

法,以提高 WCET 分析的可信度。

参考文献

1 Kligerman E, Stoyenko A. Real - Time Euclid: A language for reliable real - time systems. IEEE Transactions on Software Engineering, 1986, SE - 12(9):941 ~ 949

2 Mok A, Amerasinghe P, Chen M, Tantisirivat K. Evaluating tight execution time bounds of programs by annotations. In: proc. 6th IEEE workshop on real - time operating system and software, 1989.74 ~ 80

3 Puschner P, Koza C. Calculating the maximum execution time of real - time programsReal - time systems, 1989,1(2):159 ~ 176

4 Shaw A. C Reasoning about time in higher - level language software. IEEE transactions on software engineering, 1989, SE - 15(7):875 ~ 889

5 Cousot P, Cousot R. Abstract Interpretation: A unified lattice model for static analysis of programs by construction or approximation of fix points. In: Proc. of the 4th ACM Symposium on Principles of Programming Languages, 1977.238 ~ 252

6 邹勇,李明树,王青.开放式实时系统的调度理论与方法分析.软件学报,2003,14(1):83 ~ 90

7 Park C Y. Predictable program execution time by analyzing static and dynamic program paths. Real - time systems, 1993,5(1):31 ~ 62

8 Alterbernd P . On the false path problem in hard real - time programs. In: ProcEuromicro Workshop on Real - Time Systems. L'Aquila, Italy, 1996

9 Pospischil G, Puschner P, Vrchoticky A. Developing real - time tasks with predictable timing. IEEE software, 1992,9(5):35 ~ 44

10 Puschner P. A Tool for high - level language analysis of worst - case execution times. In: proc. Euromicro workshop on real - time system, 1998.130 ~ 137

11 Vrchoticky A. Compilation support for fine - grained execution time analysis. In: proc. of the ACM SIGPLAN workshop on language, compilers, and tools for real - time system, 1994

12 Puschner P Worst - case execution time analysis at low cost. Control Engineering Practice, 1998,6(1):129 ~ 135

13 Engblom J, Ermedahl A, Altenbernd P. Facilitating worst - case execution times analysis for optimized codeIn: ProcEuromicro Workshop on Real - Time Systems, 1998.146 ~ 153

14 Kim S, Min S L. Efficient Worst Case Timing Analysis of Data Caching. In: Proc. of the IEEE Real - Time Technology and Applications Symposium, 1996.230 ~ 240

15 Li Y S, Malik S, Wolfe A. Cache modeling for real - time software: beyond direct mapped instruction caches. In: proc17th IEEE real - time system symposium, 1996

16 Healy C A, Arnold R D, Mueller F. Bounding Pipeline and instruction cache performance. IEEE Transaction on Computers, 1999:48(1):53 ~ 70

17 Stappert F. Predictable pipelining and caching behaviour of hard real - time programs. In: procEuromicro workshop on real - time system, 1997.80 ~ 86

18 Stappert F, Altenbernd P. Complete worst - case execution time analysis of straight - line hard real - time programs. Journal of Systems Architecture, 2000,46(4):339 ~ 355

19 Theiling H, Ferdinand C, Wilhelm R. Fast and precise WCET prediction by separated cache and path analysis. Journal of Real time Systems, 2000,18:157 ~ 179

20 Colin A, Puaut I. Worst case execution time analysis for a processor with branch prediction. Journal of Real time Systems, 2000,18:249 ~ 274

21 Ottosson G, SjAodin M. Worst - case execution time analysis for modern hardware architectures. In: Proc. ACM SIGPLAN Workshop on Languages, Compilers and Tools for Real - Time Systems (LCT - RTS'97), June 1997

22 Engblom J, Ermedahl A. Modeling Complex Flow for Worst - Case Execution - Time Analysis. In: Proc. 21th IEEE Real - Time Systems Symposium (RTSS'00), 2000

23 Puschner P, Burns A. Wiring temporal predictable code. In: Proc.7th IEEE international workshop on object - oriented real - time dependable systems, 2002.85 ~ 91

(上接第 33 页)

构。举例来说,如果实例代码的结构类似于"ABCD",其中 A 表示施教对象代码,B 表示框架主题代码,C 表示模型种类代码,D 表示实例序号,则关于某个主题 t_{ij}的 FMI 教学活动 FMI(t_{ij})的检索就非常简单明了,只需要在 I(e)中提取代码为"AB * *"的实例子集即可。

结束语 学高为师,生正为范,在教学领域,由于学科不同,施教对象不同,因此可采用的教学模式也千差万别。让我们积极探索,潜心研究,以科学务实的教学活动适应新形势下的高等教育,培养出更多的新世纪创新人才。

参考文献

1 黄剑飞.VR 技术与教学课件[J].工程图学学报,2001(增刊):285 ~ 287

2 张恩华.任务驱动教学法初探[J].郑州铁路职业技术学院学报,2003,15(3):79 ~ 83

3 邵玉琳.论计算机硬件维修教学[J].电子计算机与外部设备,2000,24(3):54

4 刘娜.《计算机维修与维护》课的教学改革[J].机械职业教育,2003(3):37 ~ 39

5 袁保社."微机维修与维护"课教学内容探讨[J].高等理科教育,2002(5):68 ~ 69

6 王金岗.计算机维修实训教学的尝试[J].现代技能开发,2002(7):91 ~ 92

7 张夷人.微机组装与维护[M].北京:清华大学出版社,2003

8 杨志翔.微型计算机故障诊断与处理[M].重庆:重庆大学出版社,2002

9 丁唐.最新计算机组装与维修教程[M].北京:冶金工业出版社,2000

10 封滟彦,聂明德,王世高.最新电脑采购与组装完全手册[M].重庆:电脑报社,2002

11 梁和.微机组装与维修[M].北京:清华大学出版社,2002

12 佟伟光.微型机组装与维护实用教程[M].北京:高等教育出版社,2003