程序执行时间的静态预估与可视化分析方法*

孙昌爱, 金茂忠, 刘 超, 靳若明

(北京航空航天大学 计算机科学与工程系,北京 100083)

An Approach to Static Prediction and Visual Analysis of Program Execution Time

SUN Chang-Ai+, JIN Mao-Zhong, LIU Chao, JIN Ruo-Ming

(Department of Computer Science and Engineering, Beijing University of Aeronautics and Astronautics, Beijing 100083, China)

+ Corresponding author: Phn: 86-10-82317641, E-mail: sca@sei.buaa.edu.cn

http://www.buaa.edu.cn

Received 2001-07-10; Accepted 2002-04-10

Sun CA, Jin MZ, Liu C, Jin RM. An approach to static prediction and visual analysis of program execution time. *Journal of Software*, 2003,14(1):68~75.

Abstract: An important issue of real-time software development is to analyze and predict the execution time of real-time software. A kind of visual prediction and analysis framework of the execution time of real-time software based on program flowchart is proposed in the paper. The key issues of implementing the framework are discussed in detail, including creating the mapping between intermediate code segment and statement line of source code, retrieving the time of any given program segment from the perspective of CPU cycles of goal machine instruct, calculating CPU cycles of statement lines of source code, point-to-point $WCET_C$ (worst case execution time calculated) analysis algorithm based on program flowchart, and transforming CPU cycle into physical time. Based on the framework, a practical tool has been developed to predicate and analyze visually the program execution time. Finally, conclusion and comparison between the work in this paper and others is given.

Key words: program execution time evaluation; worst case execution time; program flowchart; real-time software; software testing; software engineering

摘 要: 软件时间性能分析与评估技术是实时软件开发中的一个重要课题.提出了一种基于控制流程图的程序执行时间的可视化分析框架,研究了中间代码段与源程序中语句的对应关系的自动分析、源程序语句行的CPU周期数的提取和计算方法、基于控制流程图的点到点最大时间分析算法和CPU周期的绝对时间估计方法.设计并实现了一个实用的基于控制流程图的程序执行时间静态分析与评估工具.最后,对研究工作进行了相关比较和总结.

关键词: 程序执行时间预估;最大执行时间;程序控制流程图;实时软件;软件测试;软件工程

中图法分类号: TP311 **文献标识码**: A

* Supported by the National Natural Science Foundation of China under Grant No.60073005 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2001AA110244 (国家高技术研究发展计划)

第一作者简介: 孙昌爱(1974－),男,江苏盐城人,博士生,主要研究领域为软件测试,软件工程环境,软件体系结构,构件技术.

实时软件,特别是硬实时软件的正确性不仅依赖于程序的逻辑,还依赖于程序的时间特性[1],即某一任务必须在指定的时间范围内完成,否则会导致软件系统的失败.基本的时间特性指标有指定程序段的最大执行时间 $WCET_C$(worst case execution time calculated)和最小执行时间 $BCET_C$(best case execution time calculated).

在时间预估方法方面,文献[2]中提出一种以机器指令执行时间为依据、面向程序源代码的时间分析方案.程序执行时间预估的核心问题是语句行执行时间的提取[3].而机器指令周期又与诸多因素相关,如机器类型、指令体系结构、Caching、流水线技术等,因此必须研究机器指令周期计算模型[4~6].在基于语句行时间分析的时候,必须考虑语言层所产生的影响[7~9].

我们认为,程序执行时间的静态预估与分析应直观,即可视化、易于实现与修改,并具备一定的可信度.本文提出了一种基于程序控制流程图、分层的可视化时间分析方法,给出了一种可视化的时间分析与基于汇编代码层的时间提取相结合的统一框架,并讨论了其中若干关键技术.

1 基于控制流程图的程序执行时间预估分析的原则与框架

1.1 程序执行时间的预估分析的原则

程序执行时间的静态预估为实时软件开发人员在软件开发时提供了必要时间性能度量手段,因此必须具备足够的可信度,即满足如下两个规则:

(1) $\forall p.((WCET_C(p) \geq WCET_A(p)) \wedge (BCET_C(p) \leq BCET_A(p)))$.

这称为有效性规则.任何指定程序段 p 的预估最大执行时间 $WCET_C$ 应大于等于实际要求的最大执行时间 $WCET_A$,预估的最小执行时间 $BCET_C$ 应该小于等于实际要求的最小执行时间 $BCET_A$.

(2) $\forall p.((\frac{|WCET_C(p)-WCET_A(p)|}{WCET_A(p)} \leq \delta_{wlmt}) \wedge (\frac{|BCET_C(p)-BCET_A(p)|}{BCET_A(p)} \leq \delta_{blmt}))$.

这称为精确性规则,即要求最小执行时间 $BCET_A$ 和最大执行时间的预估 $WCET_C$ 误差分别小于一定范围的 δ_{blmt} 和 δ_{wlmt}.

在程序执行时间预估与分析的实现上,应该具备如下特点:易于实现并易于集成已有研究成果,如在改进机器描述模型后能得到更准确的语句行提取算法,因此实现上应构件化;时间分析应该尽可能地实现自动化,同时具备一定的适应性.

1.2 程序执行时间可视化预估分析原理

程序执行时间可视化分析原理如图 1 所示.

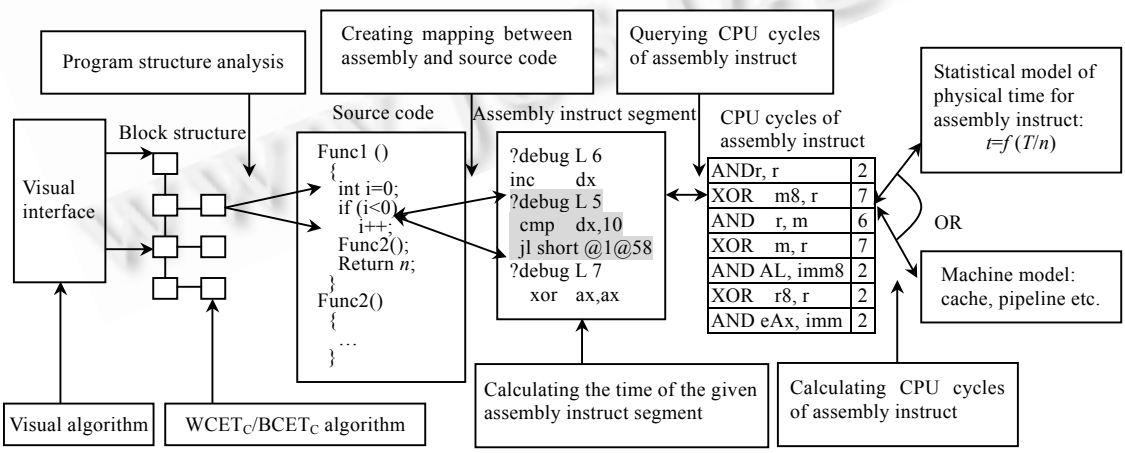

Fig.1 The principle of visual analysis of program execution time based on source code

图 1 基于源代码的程序执行时间的可视化分析原理

在图 1 中,可视化界面提供程序结构的执行时间显示与用户交互;可视化算法处理程序结构的布图;最大最

小时间分析算法分析指定程序段的最大/最小执行时间;程序结构分析采用编译前端处理技术得出程序代码与程序结构图中块的映射关系;汇编、高层语言关联分析维持汇编代码与高层语言之间的映射关系;汇编代码段结构分析与时间计算分析汇编代码的程序结构并分析其时间;汇编代码是指相应的高级程序段的对应的汇编代码段;汇编指令周期表为汇编指令的 CPU 周期数;汇编指令周期分析根据不同的计算模型得出汇编指令的周期数.

1.3 基于控制流程图的可视化时间分析框架

可视化时间分析层次型结构由汇编代码分析层、源代码分析层和可视化分析层构成,如图 2 所示.

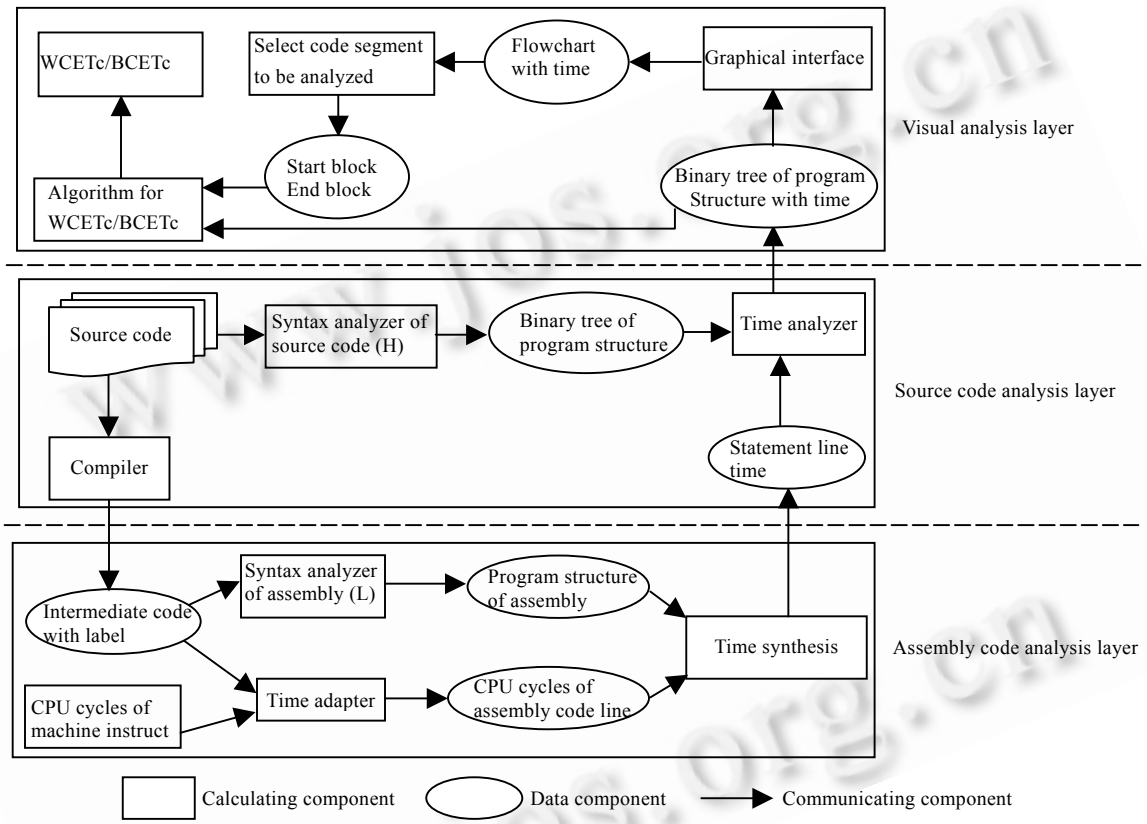

Fig.2 The framework of visual time analysis based on program flowchart

图 2 基于控制流程图的可视化时间分析框架

1.3.1 汇编代码分析层

汇编代码分析层主要任务为源代码分析层提供语句行时间数.其中,机器指令周期表是整个分析过程的基础.根据软件运行的目标环境,查看 CPU 指令格式及相应周期表,并建立指令周期索引格式,见表 1.

Table 1 The CPU cycles of machine instruct

表 1 机器指令周期表

Index	CPU cycles	Instruct	Index	CPU cycles	Instruct
…					
23/0	= 2	;AND r,r	30/1	= 7	;XOR m8,r
23/1	= 6	;AND r,m	31/1	= 7	;XOR m,r
24	= 2	;AND AL,imm8	32/0	= 2	;XOR r8,r
25	= 2	;AND eAx,imm	32/1	= 6	;XOR r8,m8
26	= 1	;ES:	33/0	= 2	;XOR r,r
27	= 4	;DAA	42	= 2	;INC eDX
28/0	= 2	;SUB r,r8	7C	= 7	;JL rel8
			83#7/0	= 2	;CMP r,imm8
…					

在建立指令周期索引表时,本文按指令类型进行分类和编排.机器指令周期表确保了程序执行时间分析与具体机器属性的分离,具备良好的通用性和适应性:可以适用于不同语言编写的实时软件;支持用户的反馈,即用户可以采用最适合于目标系统时间特性的评估模型来定义机器指令周期表,并可以根据实际使用效果和评估精度对此表进行修正.

1.3.2 源代码分析层

将由汇编代码分析层得到的源程序语句行时间数存放到中间数据库中,并供源代码层分析使用.通过源代码语法分析器(H)对源程序的分析得到了源代码的程序结构.对程序结构描述可以采用以块为节点的二叉树结构.在源代码的二叉树结构的基础上,根据中间代码层分析得到的语句行时间数,由时间合成分析器计算块结构中的基本时间数.

1.3.3 可视化分析层

在可视化分析层中,可视化布图根据源代码分析层得到带基本时间评估值的程序结构二叉树逆向生成一个带时间特性程序结构控制流程图[10],不仅反映了程序的基本控制逻辑,而且给出各个分支块的基本时间数.最小/最大时间分析算法不仅要进行可达性检查,还要进行最大执行时间和最小执行时间分析.

2 程序执行时间的预估计算与可视化分析

2.1 高级程序语言的语句行时间的分析算法

基于中间语言的时间提取的最大优点在于考虑了影响程序执行时间的硬件因素.为此,要建立源代码语言和中间语言(如汇编语言)之间的双向映射.将高级语言翻译成中间语言的任务可由高级语言编译器完成,只需在编译时添加适当的编译选项就可以生成带语句标号的机器代码,如图 3(b)所示.汇编代码分析器(L)分析带标号的中间代码程序,得到源代码语句所对应的汇编代码段[11].如源代码语句“for (;*I*<10;)”(如图 3(a)所示)在源程序中为第 5 行,其对应的汇编代码为图 3(b)中阴影部分的代码.在源语句翻译成带标号的汇编代码段时都有一个明显的特征,即汇编代码段以“?debug L xx”开始,而以“?debug L yy”结束.

```
1    int main()
2    {
3       int i;
4       i=1;
5       for(;i<10;)
6       i++;
7       return 0;
8    }
```

(a) The sample of source code
(a) 源程序代码样例

```
37                                  ;
38                                  ;     for (;i<10;)
39                                  ;       i++;
40                                  ;
41                                    ?debug L 6
42 0008   42                                inc   dx
43 0009                               @1@86:
44                                    ?debug L 5
45 0009   83 FA 0A                          cmp  dx,10
46 000C   7C FA                             jl    short @1@58
47                                  ;
48                                  ;     return 0;
49                                  ;
50                                    ?debug L 7
51 000E   33 C0                             xor   ax,ax
```

(b) The assembly code with statement label
(b) 带语句标号的机器代码

Fig 3
图 3

语句行时间提取的算法如图 4 所示.

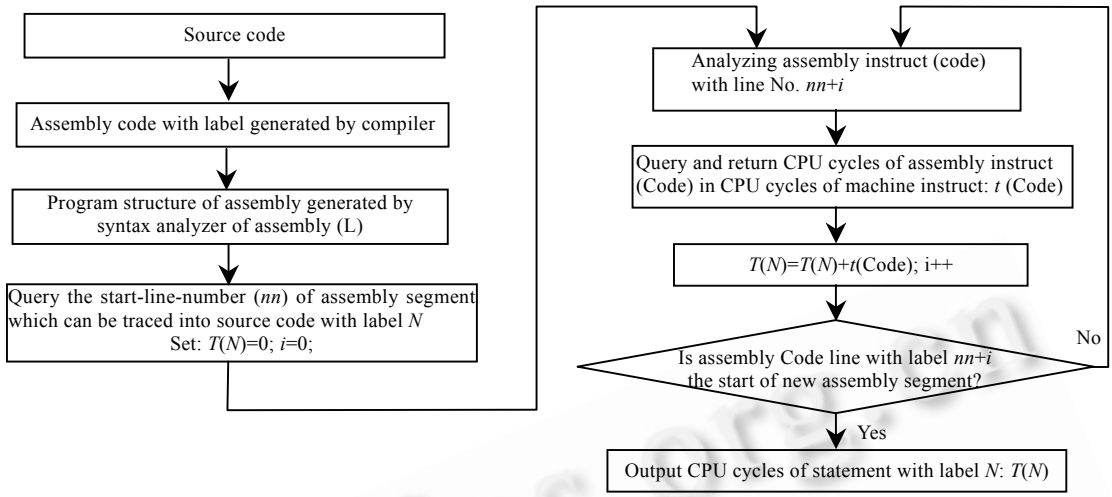

Fig.4 The retrieving algorithm of time of source code statement line
图 4 源程序语句行时间提取算法

2.2 指定程序段的最大/最小时间分析算法

2.2.1 基本概念

定义 1(基本块).

$$BasicBlock=\{Statement_{i,\cdots,j}\mid (i\le j)\wedge\neg\exists n.((n<i)\wedge(Statement_n\prec Statement_i)\wedge (type(Statement_n)\in EnumTypeSet))\wedge\neg\exists m.((j<m)\wedge(Statement_j\prec Statement_m)\wedge (type(Statement_m)\in EnumTypeSet))\wedge(\forall k.(i\le k\le j)\wedge(type(Statement_k)\notin EnumTypeSet))\}.$$

其中,$Statement_i$ 表示程序中的逻辑语句行 i,$Statement_{i,\cdots,j}$ 表示由逻辑语句 i 到 j 构成的逻辑语句行的段;$Statement_x\prec Statement_y$ 当且仅当 $Statement_x$ 执行后立即执行 $Statement_y$;$EnumTypeSet$ 这样几种特殊的跳转语句类型.

定义 2(特殊基本块). $SpecialBasicBlock=\{Statement_i\mid type(Statement_1)\in EnumTypeSet\}$.

定义 3(复合块). $CompositeBlock=\prod_{j=1}^{n}(\sum_{i=1}^{m}block_i)$.其中,$Block\in BasicBlock\cup SpecialBasicBlock$, $\sum_{i=1}^{m}$表示 1 到多个基本块或特殊基本块的集合,$\prod_{j=1}^{n}$表示对该块集合进行 1~n 次复合,对于特殊的语言复合,应该满足一定的语法.

定义 4(子块关系(Sub)). $Block_j$ 是 $Block_i$ 的子块(记 $Block_j=Sub(Block_i)$)当且仅当满足如下条件:

(1) $Block_j, block_i\in BasicBlock\cup SpecialBasicBlock\cup CompositeBlock$;

(2) 如果 $Block_i$ 执行可能引起 $Block_j$ 执行,而且 $Block_j$ 执行后继续 $Block_i$ 执行.

定义 5(顺接块关系(Next)). $Block_j$ 是 $Block_i$ 的顺接块(记 $Block_j=Next(Block_i)$)当且仅当满足如下条件:

(1) $Block_j, block_i\in BasicBlock\cup SpecialBasicBlock\cup CompositeBlock$;

(2) 如果 $Block_i$ 执行必然引起 $Block_j$ 执行,而且 $Block_j$ 执行后不会导致 $Block_i$ 执行.

2.2.2 源代码程序结构的二叉树表示和时间树

按照上述程序块的思想,通过源代码语法分析器(H)可以得到源代码程序结构的二叉树描述[12].时间树则是带时间属性的二叉树,即在二叉树中每个节点加入时间属性,是基于控制流程图的最大/最小时间分析基础.

2.2.3 可达性规则

程序执行时间预估分析一般分为点到点分析法和线性分析法.前者是指程序中任意指定两点确定的程序段的时间分析,而后者则逐个分析程序语句行时间.点到点时间分析首先要确定指定的程序段是否合理,即进行

可达性检查.在以块为节点的程序二叉树中,我们将可达性规则定义为:称 $Block_i$ 可到达 $Block_j$,则应满足如下条件之一:

(1) $Block_i=Block_j$;

(2) $Block_j=Next(\dots Next(Block_i))$;即 $Block_i$ 通过有限次$(1,\dots,n)$顺接块达到 $Block_j$.

2.2.4 最大时间分析算法

本文提出的最大/最小时间分析算法基于以块节点为单位的程序二叉树结构以及中间代码分析层得到语句行时间数集合.由于最小时间算法与最大时间算法是对称的,因此仅讨论最大时间算法.

输入:*StartBlock*,*EndBlock*,程序的二叉树结构,程序语句行时间数;

输出:最大时间数.

算法过程如下:

步骤 1. 可达性判断,当且仅当起始块等于终止块,或者起始块经过有限次的顺接块到达终止块时,则设可达性标志为 TRUE,否则设为 FALSE;

步骤 2. 如果可达性标志为 TRUE,则转步骤 3;否则退出,并给出不可达信息;

步骤 3. 按照先序遍历以块为节点的程序结构二叉树,并按照表 2 中的规则求得各种基本块节点和特殊基本块节点的基线时间数 *Time*;

步骤 4. 采用递归算法,并按照表 2 中的各种块的最大时间 Max*Time* 的计算规则,计算各种块的最大时间数 Max*Time*;

步骤 5. 按如下算法求得指定程序段的最大时间数:

$$\text{Max}Time=\sum_{i=0}^{n}Next^{i}(startBlock).\text{Max}Time,$$

其中 $Next^0(block)=block$, $Next^1(block)=block.NextBlock$, $Next^{i+1}(block)=Next(Next^i(Block))$.

步骤 6. 返回最大时间数 Max*Time*.

Table 2 The calculating rules for basic time and max time of different types of statements and blocks

表 2 各种语句类型及块节点的基线时间与最大时间计算规则

Statement type	Block type	*Time*	Max*Time*
Expr	BasicBlock	The total time of statements which is contained in the block	Max*Time*=*Time*=$\sum_{i=block.StartLine}^{block.EndLine}getLineTime(i)$ //*getlineTime*(*i*) to get the time of statement line No. *I*
If	SpecialBasiclBlock	The time of If statement line	Max*Time*=*Time*+Max(*block.subBlock*.Max*Time*, *Block.subblock.nextBlock*.Max*Time*)
Switch	SpecialBasiclBlock	The time of Switch statement line	Max*Time*=*Time*+Max(*CaseBlock*1.Max*Time*,…, *CaseBlock n*. Max*Time*)
Case	SpecialBasiclBlock	The time of Case statement line	Max*Time*=*Time*+*Block.subBlock*.Max*Time*;
Default	SpecialBasiclBlock	The time of Default statement line	Max*Time*=*Time*+*Block.subBlock*.Max*Time*;
Label	SpecialBasiclBlock	The time of Label statement line	Max*Time*=*Time*
Break	SpecialBasiclBlock	The time of Break statement line	Max*Time*=*Time*
Goto	SpecialBasiclBlock	The time of Goto statement line	Max*Time*=*Time*
Return	SpecialBasiclBlock	The time of Return statement line	Max*Time*=*Time*
Continue	SpecialBasiclBlock	The time of Continue statement line	Max*Time*=*Time*
For	SpecialBasiclBlock	The time of For statement line	Max*Time*=*block.ExecTimes**(*Time*+*block.subBlock*.Max*Time*)+*Time*
Do	SpecialBasiclBlock	The time of Do statement line	Max*Time*=*block.ExecTimes**(*Time*+*block.subBlock*.Max*Time*)
While	SpecialBasiclBlock	The time of While Statement line	Max*Time*=*block.ExecTimes**(*Time*+*block.subBlock*.Max*Time*)+*Time*
Compound	CompositeBlock	0	Max*Time*=*block.subBlock*.Max*Time*;

2.3 CPU周期的绝对时间估算

基于机器描述模型[5]的绝对时间估算方法实现起来较为困难,且各种因素的影响因子计算复杂.为此,本文提出了基于统计方法估算各种机器环境下的CPU周期的绝对时间,其方法如下:

(1) 选择有各种语句成分的C语言程序片段P_s,在P_s前后分别插入时间断点T_b和T_e,即$\{T_b\}P_s\{T_e\}$.为了保证记录的准确性,将P_s执行次数放大n倍,即$\{T'_b\}nP_s\{T'_e\}$,将由$\{T'_b\}$和$\{T'_e\}$确定的时间段除以系数n,可得P_s实际执行时间T_s(我们将此方法称为放大刻度方法),即$T_s=\frac{T'_e-T'_b}{n}$.

(2) 将P_s编译生成中间代码程序P_a,计算P_a的CPU周期数T_a(编译的方式必须同P_s编译生成可执行文件的模式一致).

(3) 对于任何高级语言程序段P_S经相同的编译模式生成机器代码P_m和中间代码P_a,如果将P_a翻译生成机器代码P'_m,则P_m和P'_m的时间特性相同.据此,计算T_s和T_a的关系f,得到$T_s=f(T_a)$.

3 程序执行时间静态预估与可视化分析工具

基于本文提出的可视化程序执行时间分析框架,实现了支持多种C/C++源程序的程序执行时间预估分析工具,如图5所示.流程图中的数字均表示基本块的执行时间,不同的块采用相应的字母标识,如B表示break语句,R表示return语句,while表示while语句,if表示if语句等等.选择起始块和终止块后(如图5中两箭头所示),进行可达性检查.程序中如果存在循环,在计算时则提醒用户给出循环次数,并执行时间最小/最大分析.如while循环次数设为10时,指定程序段的执行时间的CPU周期数最大为418(11++18+(18+4+9+7)*10+9),最小为348(11++18+(18+4+9)*10+9).

时间预估分析工具还支持线性时间分析,即在源程序中给出每个语句行的时间数,如图6所示.图中左栏第1列表示语句行号;第2行是时间数(可执行语句才有CPU周期数,而不可执行语句在编译时不存在相应的机器代码,图中标以#);右栏为源程序代码行.

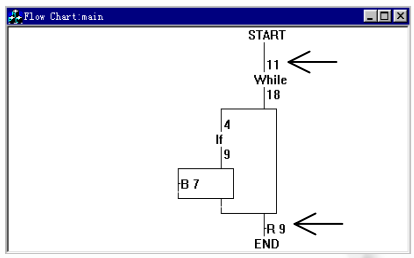

Fig.5 The program flowchart with time
图5 带时间数的控制流程图

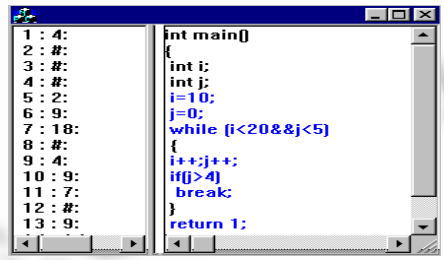

Fig.6 The view of source code with time
图6 有时间数的源程序视图

4 结束语

与已有工作相比,本文的研究工作有如下几个特点:(1) 程序最终执行要经过编译器或解释器生成机器代码,因此,基于汇编代码估计高级源程序中的语句行时间既易于实现,又不失精确性;(2) 由于高级语言存在着多样性,采用基于汇编代码的语句行时间分析方法在实现上可以有效地解决复用问题;(3) 在基于语句行的最大时间分析方面,本文提出并实现了基于控制流程图之上的可视化程序执行时间静态分析.与传统的工作[2]相比,将基于高层源代码层语句时间分析向前推进了一步;(4) 在将CPU周期转化为绝对时间方面,本文提出了采用统计方法来估算CPU周期的实际执行时间,从而避免了基于机器描述模型方案中的复杂性[3~6].实时软件的实时性需求使得在开发阶段就能准确地预估程序的执行时间成为迫切需要.本文研究了基于程序源代码的可视化时间预估分析技术,提出了一个基于程序控制流程图的可视化时间分析框架,并讨论了其中主要的关键技术.按照文中讨论的可视化分析方法,实现了一个实用的C/C++程序的时间预估分析工具,并已投入实际应用.

References:

[1] Sun CA, Jin RM, Liu C, *et al*. Testing technology of real-time and embedded software. Mini Micro Systems, 2000,21(9):920~924 (in Chinese with English abstract).

[2] Lo K, Christopher H, Emily R, *et al*. Timing constraint specification and analysis. Software-Practice and Experience, 1999,29(1): 77~98.

[3] Harmon MG, Baker TP, Whalley DB. A retargetable technique for predicting execution time. In：Proceedings of the 13th Real-Time Systems Symposium. 1992. 68~77. http://serel.cis.famu.edu/~harmon/rtss92.ps.

[4] Lim SS, Bae YH, Jang GT, *et al*. An accurate worst case timing analysis for RISC processor. IEEE Transactions on Software Engineering, 1995,21(7):593~604.

[5] Lim, SS, Min SL, Park CY, *et al*. An accurate instruction cache analysis technique for real-time system. In: Proceedings of the Workshop on Architectures for Real-Time Applications. 1994. 139~147. http://archi.snu.ac.kr/archive/rt-timing-analysis/AN-sslim-wart94.ps.

[6] Huang TY, Liu JWS, Hull D. Method for bounding the effect of DMA I/O interference on program execution time. In: Proceedings of the Real-Time Systems Symposium. 1996. 275~285. http://pertsserver.cs.uiuc.edu/members/dhull/.

[7] Healy C, Sjodin M, Rustagi V, *et al*. Bounding loop iterations for timing analysis. In: Proceedings of the Real-Time Technology and Applications. 1998. 12~21. http://user.it.uu.se/~mic/loopbound_hsrw.pdf.

[8] Ermedahl A, Gustafsson J. Deriving annotations for tight calculation of execution time. In: Proceedings of the Euro-Par'97. 1997. 1298~1307. http://www.docs.uu.se/astec/reports/reports/9704.ps.gz.

[9] Puschner P, Schedl A. Computing maximum task execution times—a graph-based approach. Real-Time Systems, 1997,13(1): 67~91.

[10] Sun CA, Liu C, Jin MZ. An effective wove algorithm for program graph. Journal of Beijing University of Aeronautics and Astronautics, 2000,26(6):1305~1309 (in Chinese with English abstract).

[11] Jin RM. Testing approach and tool for software with high reliability software [MS. Thesis]. Beijing: Beijing University of Aeronautics and Astronautics, 1999 (in Chinese with English abstract).

[12] Wu PC, Jin MZ. Static analyzer for C++ based on object-relations model. Journal of Beijing University of Aeronautics and Astronautics, 1997,23(1):105~110 (in Chinese with English abstract).

附中文参考文献:

[1] 孙昌爱,靳若明,刘超,金茂忠.实时嵌入式软件的测试技术.小型微型计算机系统,2000,21(9):920~924.

[10] 孙昌爱,刘超,金茂忠.一种有效的程序结构图的布图算法.北京航空航天大学学报, 2000,26(6):1305~1309.

[11] 靳若明.高可靠性软件的测试方法和工具研究[硕士学位论文].北京:北京航空航天大学,1999.

[12] 吴鹏程,金茂忠.基于对象关系模型的 C++程序静态分析器.北京航空航天大学学报,1997,23(1):105~110.