多线程并行程序数据竞争静态检测方法

陈 俊[1]，周宽久[2]，贾 敏[3]

（1. 贵州师范大学 教育科学学院，贵州 贵阳 550001；2. 大连理工大学 软件工程学院，辽宁 大连 116024；3. 北京电子科技职业学院 经济管理学院，北京 100029）

摘 要：通过使用词法分析和语法分析进行线程并行程序数据竞争静态检测，根据规则将源代码数据竞争相关信息映射到XML文件，解决多个线程程序因可以随机访问共享变量而导致运行结果不确定性，容易引发数据竞争等问题。通过XQuery查询语言获取XML映射模型中的数据访问信息，将信息存储于数据竞争表示层。通过竞争图生成算法建立数据竞争有向图模型，利用拓扑排序检测数据竞争的具体位置，获取所有竞争序列。实验结果表明，该检测方法能够准确快速地发现多线程程序中的数据竞争并在源文件中进行准确定位。

关键词：多线程程序；数据竞争；静态检测；XML映射模型；XQuery查询

中图法分类号：TP3 **文献标识号**：A **文章编号**：1000-7024（2017）05-1264-09

doi：10.16208/j.issn1000-7024.2017.05.027

Multi-thread parallel program data race static detection model

CHEN Jun[1], ZHOU Kuan-jiu[2], JIA Min[3]

(1. School of Education Science, Guizhou Normal University, Guiyang 550001, China; 2. School of Software Engineering, Dalian University of Technology, Dalian 116024, China; 3. School of Economics and Management, Beijing Electronic Science and Technology Vocational College, Beijing 100029, China)

Abstract: The lexical analysis and syntax analysis were used to carry out the Multi-thread parallel program data race static detection. According to the rules of data-race, the problems that all shared variables can be freely accessed in multi-thread programs, generating uncertainty of the running results, easily causing data-race, atomic violations and dead-locks to the program were solved. All shared variables access information was extracted from the XML mapping model by XQuery language and saved into data-race presentation layer. Data race directed graph model was constructed according to data-race graph, and the concrete data-race position was detected with topological sort algorithm to extract all data-race sequence. Experimental results show that the detection method can accurately and quickly find data-races in multi-thread programs and can accurately locate their positions in the source code file.

Key words: multi-thread program; data race; static detection; XML mapping model; XQuery query

0 引 言

多线程并行程序访问共享资源具有一定随机性，会带来数据竞争（data race）、原子性侵犯（atomicity violation）及死锁（dead lock）等问题，且故障难以再现。如何有效构建多线程数据竞争抽象模型，设计准确的数据竞争定位算法，辨别由此导致的多线程程序安全故障，已成为目前保证多线程并发程序可信性亟待解决的基本难题，并影响多核处理器在航天等关键领域的普及应用。多线程并行程序执行过程不确定性使得数据竞争检测一直是可信软件研究的难点。

在动态测试方面，现有面向多线程动态检测研究成果大多因为测试用例选取数量庞大和访问随机等问题而难以进行，所以目前大部分研究倾向于静态分析。Xiangyu Zhang等[1]采用一种双切片方法，准确性较高，但需要多次执行确定互补线程；陈意云提出一个以对象为中心，结

收稿日期：2015-09-01；**修订日期**：2017-03-02

基金项目：贵州省科学技术基金项目（黔科合协字师大LH字［2014］7040号）

作者简介：陈俊（1979-），男，贵州贵阳人，博士，副教授，研究方向为网络技术；周宽久（1966-），男，辽宁大连人，博士，教授，研究方向为软件工程；贾敏（1982-），女，北京人，硕士，研究方向为管理科学与工程。E-mail:82631710@qq.com

合 Escape 分析缩小检测范围的检测算法，并配合精确的别名分析模拟访问事件发生序，减少了传统数据竞争静态检测中误报的产生；Jacob Burnim 设计一个允许程序员编写程序状态断言框架，使非确定性问题转化为确定性问题，断言书写比较简单，能协助程序员开发正确的并行程序，但需要手动书写断言；奚宏生等提出一种基于隐 Markov 模型的多线程程序时序分析方法，通过建立多线程程序时序分析的隐 Markov 模型，使用 Baum-Welch 和前向算法仿真上下文对程序实际运行状态的影响[2]；Xiangyu Zhang 提出通过运行两个不同的线程调度序列收集执行过程信息，设计利用 dual slicing 算法分析竞争故障，该方法能够对大型并行线程缺陷做精确分析，但互补线程难以确定，需要多次执行；K. Sen 等[3,4]提出将导引测试技术应用于并行程序。在传统串行程序的基础上结合并行程序多线程的特点，通过给出新的调度方案，侦测数据竞争关系，从而实现对程序路径的全面覆盖，但该研究仅局限于 Java 程序，文中给出实例仅限于理想情况下的简单程序，同时并未给出测试效率以及与其它方法的横向比较，无法证明方法的正确性和有效性；文献［5］虽给出了多线程程序执行情况的建模，但对复杂多线程程序的验证耗时较长，且容易造成内存溢出。KONG De-Guang 等[6]提出了发现并行软件 bugs 的实用方法，但该方法没有在实际的并行软件测试集上运行；Qadeer 等提出了一种发现并行软件 bugs 的实用方法，它通过有界数量的上下文切换执行对程序进行分析；Lucas 通过上下文切换分析，对多线程程序交织执行情况进行形式化建模，使用 Lazy、SR、UW 这 3 种方法验证多线程程序正确性，并讨论了加锁、原子操作等语句多线程程序性质验证问题；Koushik 等基于通用的多线程程序可控制性设计原则，提出一种编程语言 SPL，扩展了传统 Java 序列的简化描述，并用异步原子性方法对多线程进行模型验证[7]；K. Sen 提出了一种测试算法，该算法不仅可以执行程序中的所有可执行路径，而且可以检测到所有的数据竞争和死锁状态。该算法使用导引执行方法来发现并行程序的不同因果结构。

1 静态检测数据竞争方法

相比数据竞争动态检测，数据竞争静态检测有很多优势，不需要实际生成测试用例，不需要实际运行程序，静态检测首先对输入的源代码进行解析，对源代码解析基于各种成熟的编译技术或者借助 GCC 编译工具，提取多线程数据竞争涉及到的所有代码信息，即共享数据访问信息[8]。在此基础上构建映射模型，即将共享数据访问信息转换成为映射模型，一方面能够存储访问信息，另一方面能对访问信息进行解释。本文采用 XML 表示映射模型，利用 XML 在数据存储和数据交互的便利，方便信息获取及处理。数据竞争检测：在 XML 映射模型的基础上，实现数据竞争检测，获取所有可能竞争序列[9]。该检测模块主要依据数据竞争检测算法进行，并通过拓扑排序获取数据竞争序列。针对本文 XML 映射模型，采用 XQuery 查询和分析 XML 映射模型。最后数据竞争定位分析，由于检测模块检测到的数据竞争信息均来自 XML 映射模型，需要将存在数据竞争的代码段在源代码中标注出来。系统结构层次如图 1 所示。

（1）数据竞争发现层构建：一种基于 XML 描述的数据竞争访问模型，降低了代码静态信息提取难度和冗余度，并且利用 XML 在数据描述和存储上的优势以及目前成熟的 XML 文件检索技术，将数据竞争信息以 XML 格式存储，保证信息获取的全面性与准确性，便于分析[10]。

根据多线程软件特点，构建抽象语法树，嵌入语义动作，通过静态分析提取信息；设计并发访问信息数据结构，存储标识符信息和语句信息；构建并发信息 XML 中间数据模型，形式化存储信息；设计并发信息检索方法，匹配相关节点，获取共享变量等与数据竞争相关的线程信息，构建线程访问变量表描述所有线程全局共享变量访问情况[11]。如图 2 所示。

（2）数据竞争表示层构建：数据竞争表示层设计分为数据竞争表示设计以及其相应的获取算法设计。数据竞争结构表示方法如图 3 所示，竞争变量用圆形描述，引出的黑色箭头代表该数据资源被多个线程共享，其中线程对共享变量的竞争用方框进行描述，每个方框描述线程对共享变量的一次访问，Num 描述线程标号，对应相应线程标识，线程标识定义为：＜域名，线程函数名＞，并将线程标号，线程标识对应存储。线程对数据的每次访问需记录访问线程标号 Num，访问位置行号 Line，该行号位置为线程函数开始的相对值，State 描述线程对变量的访问状态，分为 Read、Write、Unlock、Lock 这 4 种[12]。

数据竞争图获取算法主要基于数据竞争发现层，数据竞争发现层的数据结构描述源代码中每个线程相关联的共享变量信息，包括完整作用域名，读写访问，访问行号等。构建数据竞争图时，须解决如何获取多个线程可能的共享变量，最简单的办法就是通过对多个线程的相关共享变量进行遍历获取，需要多次循环，复杂度为 O(m＊n)，其中 m 标识源代码中线程的个数，n 描述与线程相关联共享变量的线性遍历。

（3）数据竞争状态分析层构建：数据竞争状态分析层主要构建数据竞争状态，形象直观地判断竞争资源在多线程间的数据竞争关系。为研究竞争资源在不同线程间调度情况，需要列举出竞争资源在不同线程间的全部调度序列。我们完成了一种基于拓扑排序方法的数据竞争状态图获取算法，为分析竞争资源在多线程间的调度情况，采用多个链表描述数据竞争关系。

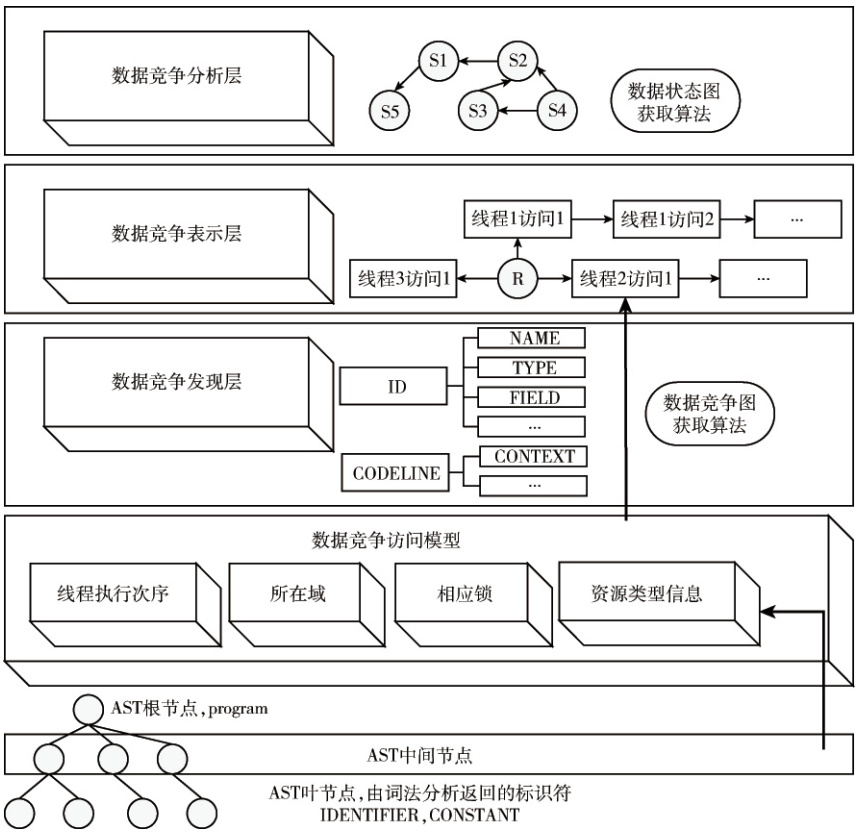

图 1 数据竞争方法系统结构层次

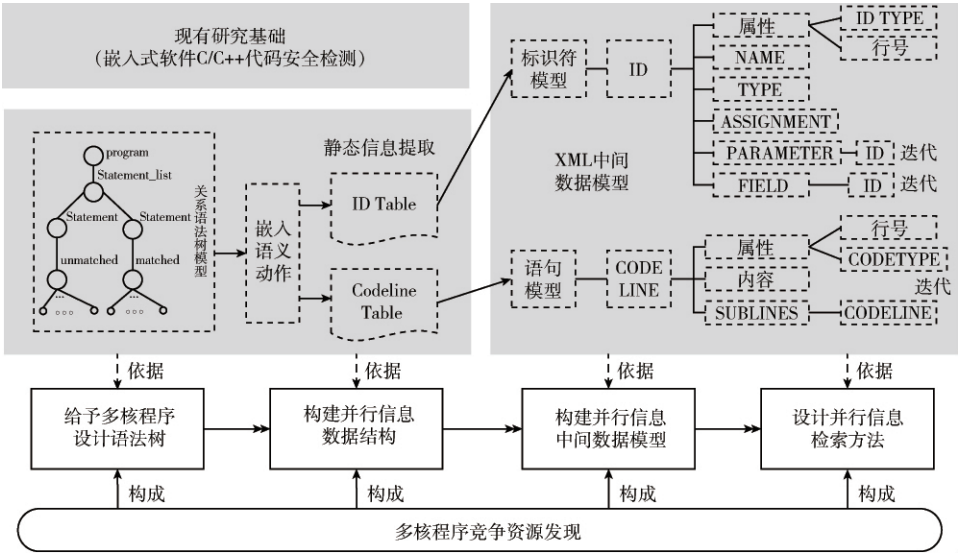

图 2 多线程数据竞争发现方法

在数据竞争状态的全部执行路径中，如若发现任意两个线程对同一竞争资源存在（Ti-读、Tj-写、Ti-读）或者（Ti-写、Tj-写、Ti-读）序列对时，即可推断待检测程序中存在潜在的数据竞争关系。图 4 为数据状态的获取过程。

基于静态代码分析的软件模型生成方法主要是从源代码中提取各类对象属性和对象间关系信息，用于构建各类模型，主要分为 3 大部分，如图 5 所示。首先通过词法分析获取粗粒度符号序列，语法分析获取代码行语句文本和结构信息；然后基于语法分析结果构建中间 XML 模型，分

离信息提取与信息分析过程，提高构建模型的可扩展性；最后设计查询过程，采用开放式的模型构建方法，只提供各类信息接口，构建用户需求的模型，基于指导的构建方式灵活性较高，有利于用户理解系统[13]。

语法分析产生结果并不适合直接进行信息归纳分析，缓存中的数据结构虽然暂存了所需信息，可以进行模型构建，但是扩展性较差，而且实现后构建过程无法进行人工干预，采用中间 XML 模型组织信息，利用 XQuery 语言设计查询方式，可以很好地解决这一问题。中间 XML 模型对最终模型构建产生直接影响，信息充足、便于检索是设计的基本原则，对已经成功识别的代码行，分类构建节点，记录其属性信息，并按照语句结构信息组织语句节点，充分保留源代码中对象属性和对象间关系信息，通过节点标识和属性可成功检索和提取所需信息。

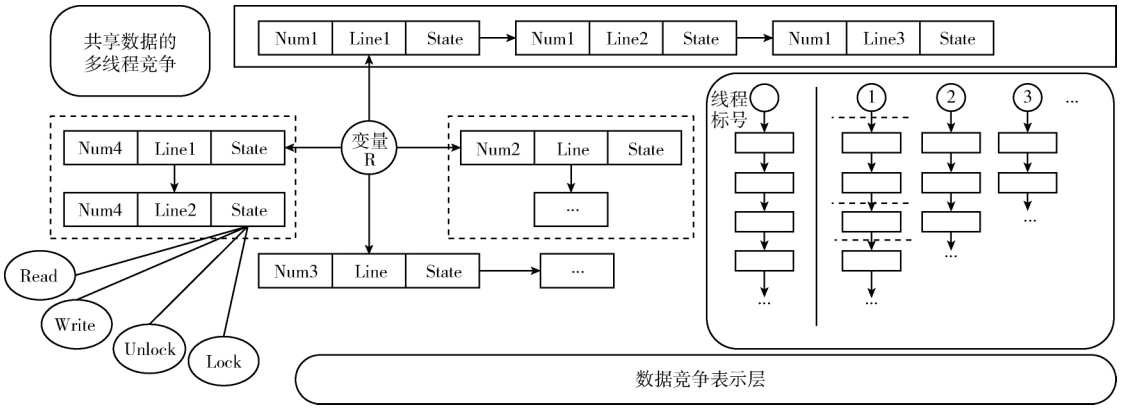

图 3　数据竞争

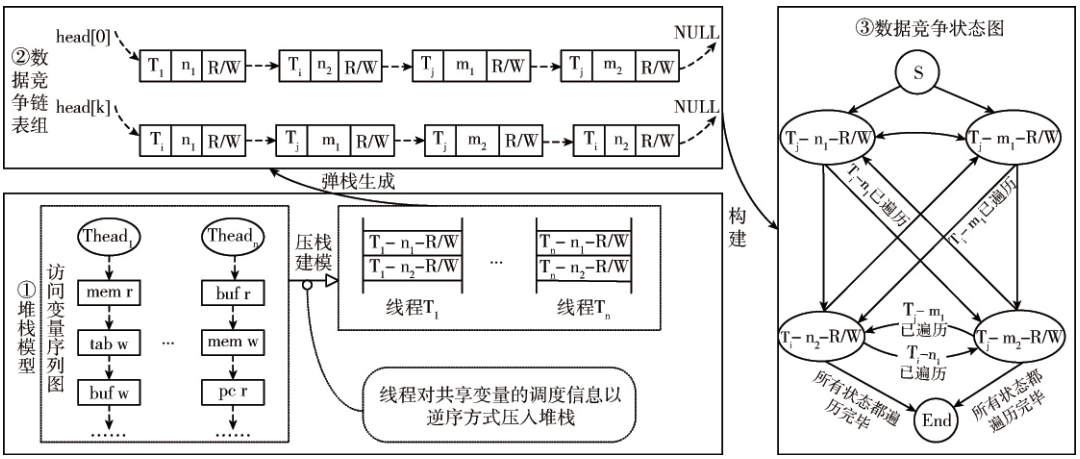

图 4　数据状态的获取

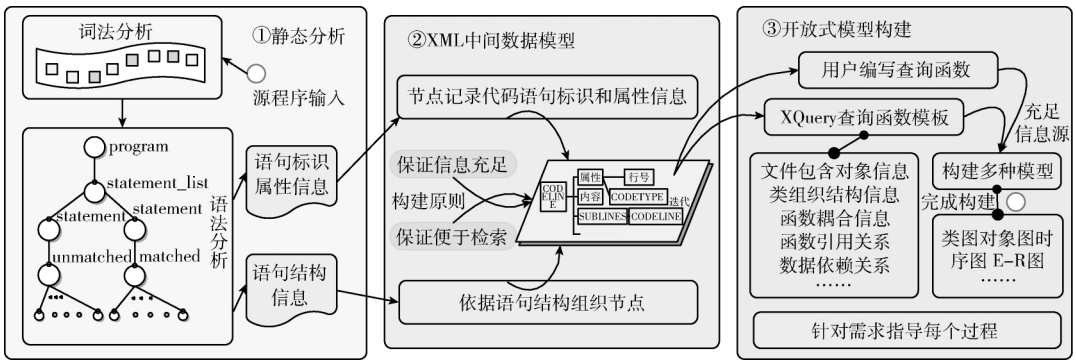

图 5　基于静态代码分析的软件模型生成方法

2 数据竞争信息获取及竞争有向图生成算法

2.1 构建线程访问数据表

为了便于分析，提高信息获取速度，利用 XQuery 查询语句从 XML 映射文件中获取所需信息，将所有线程全局共享变量访问情况构建为线程访问数据表，整个线程访问资源表由四层链表组成：第一层链表以所有线程标识符为表项，以字典排序方法对表项进行排序；第二层链表以每个线程标识符为表头，除表头外第一表项为访问数据总数目，以数据标识符起始字母分组，各个访问数据分组所含数据数目为后续 6 个表项，将访问数据数目作为表项，可以加快查询速度；第三层链表由每个访问数据分组所含数据标识符组成，以字典排序方法排序；第四层链表由每个数据标识符访问情况（行号＋访问类型）组成，以数据标识符为表头，以行号排序。线程访问数据表如图 6 所示。

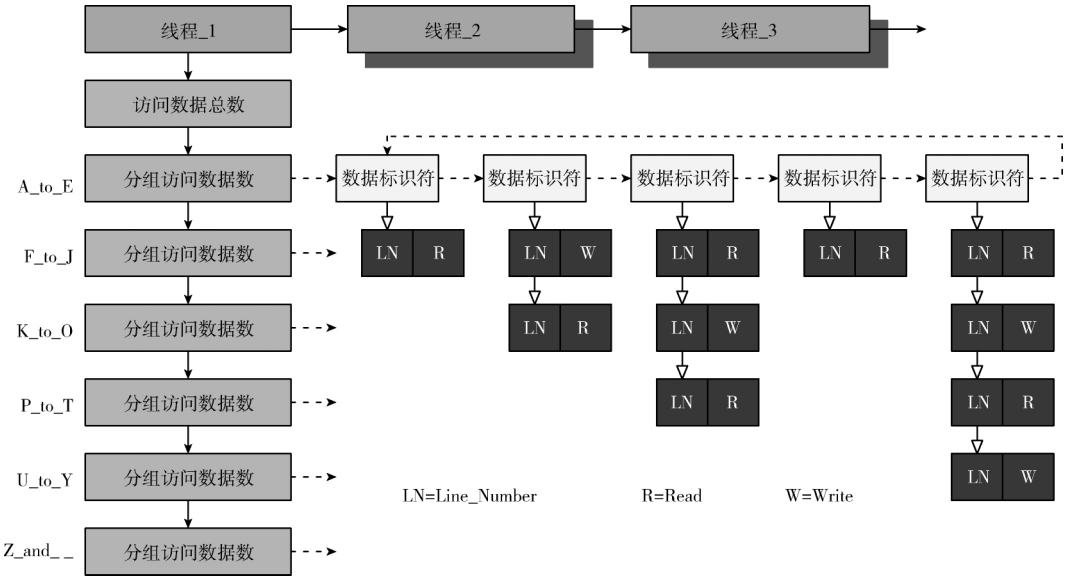

图 6 线程访问数据表

为了降低时间复杂度和空间复杂度，这里使用与字符串大小的比较规律对数据竞争数据结构进行处理，构建一种以线程为单元、兼顾竞争变量名长度的一种有序结构，并替代原有结构，释放其所在内存。

2.2 数据竞争有向图生成算法研究

（1）数据竞争有向图的定义：根据资源竞争访问模型结构特点，可以看出线程对某个变量的访问操作全部处于节点 LOCKSTART 和节点 LOCKEND 之间，因此可以将这两个节点中线程对该变量的访问信息抽象为一个树的节点；可以将访问前后关系抽象为节点间父子，兄弟关系。基于多线程程序的执行特点，一个树节点的父节点可能有多个，从而形成一种网状的树结构。利用 XQuery 查询语句，可以提取资源竞争访问模型中对该变量访问所有节点信息，抽象为一种特定的网状树结构，称之为数据竞争有向图。

定义 数据竞争有向图 $G=<S, E>$ 是一个自顶向下存在交叉的特定的树状图。从根节点到叶子节点顺序执行不存在回路。S 是树的节点，一个树节点可能有多个父亲节点。基于锁集[14]来定义其结构：

$S\rightarrow\{OPERATOR, LOCKINFO, POINTERS\}$；

$OPERATOR\rightarrow\{THREADID, LINE, OPTYPE\}$；

$THREADID\rightarrow(flead, name)$；

$LINE\rightarrow(flead, linenum)$；

$OPTYPE\rightarrow(R, W)$；

其中，$OPERATOR$ 为一个操作段，每个操作段对应于资源竞争访问模型中的 LOCKSTART 和 LOCKEND 两个节点之间的对变量访问操作信息。将其定义为一个三元组，包括线程标识符 $THREADID$，访问发生的位置信息 $LINE$ 以及访问形式 $OPTYPE$。$THREADID$ 定义为：域名＋线程函数名；$LINE$ 定义为：域名＋行号；$OPTYPE$ 访问形式分为 R 和 W。

$LOCKINFO$ 为操作段锁的信息，存储锁标识符；

$POINTERS$ 指针集，存储指向父节点和子节点指；

对于加锁的操作段，记该树节点为 LN，对于未加锁的操作段，记该树节点为 NN；

$S_0\rightarrow\{NULL, NULL, POINTERS\}$

S_0 是一个不存储 $OPERATOR$，$LOCKINFO$ 信息的 S 节点，作为树的根节点。

（2）数据竞争有向图生成算法：利用线程函数定义位置、函数调用位置和访问操作发生位置及作用域关系，在

节点 S_n 操作段中对变量的第一个和最后一个操作的行号进行修正（原 S_n 中信息不改变），修正后的值为线程函数开始的相对值，分别保存在 $L(n, l)$ 和 $L(n, f)$ 中。$IsSameThread(S_n, S_m)$ 判断 S_n 和 S_m 是否在同一线程。$Order(n)$ 记录 S_n 所在操作段 $ORDER$ 节点的值。$Level(n)$ 记录 S_n 锁空间的所在层次的值 S_n 与 S_m 发生序关系描述为

$$\begin{aligned} S_n \prec S_m \Leftrightarrow & (IsSameThread(S_n, S_m) == \\ & true \wedge L(n,l) > L(m,f)) \\ & \vee (IsSameThread(S_n, S_m) == \\ & false \wedge Order(n) < Order(m)) \\ & \vee (IsSameThread(S_n, S_m) == \\ & false \wedge (Order(n) == Order(m)) \\ & \wedge Level(n) < Level(m)) \end{aligned} \tag{1}$$

S_n 与 S_m 并发关系描述为

$$S_n \,||\, S_m \Leftrightarrow \neg \exists (S_n \prec S_m) \wedge \neg \exists (S_m \prec S_n) \tag{2}$$

为每一个被访问变量分别构建数据竞争有向图模型，基于数据竞争有向图结构定义及发生序判定分析，结合发生序和锁集，定义访问树生成算法。

构建完成的数据竞争有向图如图 7 所示。

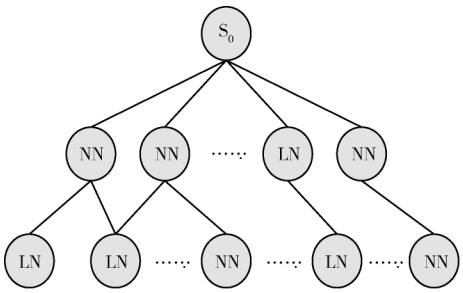

图 7 数据竞争有向图

2.3 数据竞争检测及定位算法研究

（1）数据竞争检测理论：文献［15］采用一种基于发生序和锁集的数据竞争检测方法。本文提出基于访问树 NET-TREE 模型，对任意两节点 S_n、S_m，$IsWrite(S_n)$ 表示节点 S_n 操作段中是否存在写操作，如果操作段中存在至少一个写操作，则为 $true$，否则为 $false$。$IsLock(S_n, S_m)$ 表示节点 S_n 和 S_m 是否有相同锁保护，如果 S_n 中 LOCKINFO 存储的锁标识符与 S_m 中相同，且都不为 NULL，则 $IsLock(S_n)$ 为 $true$，否则为 $false$。基于发生序和锁集的数据竞争静态检测方法对数据竞争有向图中任意两节点 S_n，S_m 是否存在数据竞争的检测，用式（3）表示如下

$$\begin{aligned} IsRace(S_n, S_m) \Leftrightarrow & S_n \;//\; S_m \quad \wedge (IsWrite(S_n) == \\ & true \quad \vee \quad IsWrite(S_m) == true) \\ & \wedge (IsLock(S_n, S_m) == false) \end{aligned} \tag{3}$$

（2）数据竞争定位相关结构体设计：利用反向重定位方法，在源代码中显示发生数据竞争的两个操作段位置[16]。检测发生数据竞争的两节点后，通过反向重定位方法将数据竞争的发生位置重定向到源代码文件中，反向重定位方法基于反向定位信息集合 $Relocation$ 构建[17]。$Relocation$ 结构描述如下：

```
Struct Relocation
{
globalVarId; //当前正处理的变量标识符（域名＋变量名）
NodeInfo1; //节点 Sn 中操作段的第一个操作和最后一个操作的位置信息；
NodeInfo2; //节点 Sm 中操作段的第一个操作和最后一个操作的位置信息；
RaceLists; //节点 Sn 和 Sm 所有可能的竞争序列；
};
```

（3）数据竞争检测算法研究：对于数据竞争有向图中的两个节点，如果有子孙关系，则顺序执行不存在数据竞争；如果没有子孙关系，但他们有相同锁保护，则不存在数据竞争，反之如果没有相同锁保护，则存在数据竞争关系[18]。再利用上述式（2），可以设计数据竞争检测算法。该算法输入数据竞争有向图，输出反向定位信息集 $Relocation$，并以数据竞争图中除根节点外的所有其它节点的集合开始，后将 S_n 和 S_m 操作段的访问位置信息及变量标志添加到 $Relocation$ 中，并将 S_n 和 S_m 中操作段的信息于栈数组 $Thread\_Stack[]$ 中，调用 topSort 获取两个节点间的竞争序列，否则 S_n 和 S_m 不存在数据竞争。

（4）基于拓扑排序的数据竞争序列获取算法：对于两个存在数据竞争的节点，节点内的访问操作顺序执行，而节点间的则随机执行，竞争数据在多个线程间调度执行序列可以看作是竞争数据在单线程执行序列基础上的一种拓扑排序，因此提出拓扑排序获取两竞争节点间访问操作所有可能的竞争序列的算法 topSort。TopSort 算法流程如图 8 所示。

通过获取 $Relocation$ 中变量访问位置信息，可反向定位到源程序相应位置，提示在源程序中数据竞争存在的两个操作段的位置，并列出两个竞争操作段中全部可能竞争序列，给出修改建议。算法输入为栈数组 $Thread\_Stack[]$，输出为所有可能的竞争序列 $RaceList$。检测竞争序列 $RaceList$ 的方法为，弹出栈首元素并压入排序队列，对刚列为的栈首元素恢复到栈首，后删除 $raceTempList$ 最后一个元素。

2.4 实 验

实验测试源代码取自文献［19］中 badcnt. c，该段代码为错误同步化的计数器程序。经过算法分析，可得到该源代码中 badcnt. c::cnt 变量的 NET-TREE 模型如图 9 所示。

表 1 中描述 badcnt. c 中基本的竞争关系，源代码中有两个线程对变量 badcnt. c::cnt 产生了数据竞争，线程 badcnt. c::count#1 和线程 badcnt. c::count 调用相同线程函数，因此加编号作为区分。实验所得的数据见表 1。

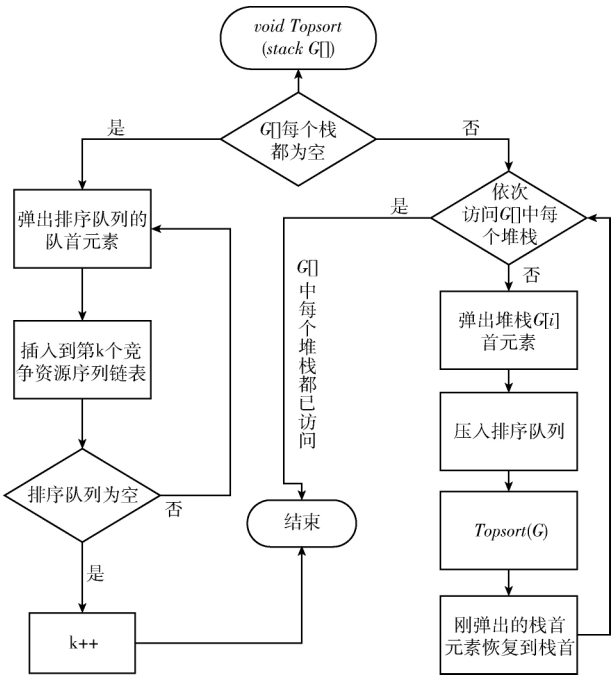

图 8 数据竞争序列获取算法流程

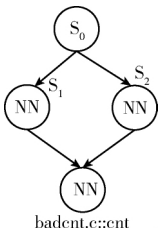

图 9 badcnt. c::cnt 变量的 NET-TREE 模型

表 1 竞争信息描述

编号	竞争变量	竞争节点 1(strat-end)	竞争节点 2(strat-end)	数目
1	badcnt. c::cnt	badcnt. c::count::30	badcnt. c::count#1::30	6
		badcnt. c::count::30	badcnt. c::count#1::30	

表 1 描述表 2 中每两个竞争节点所有可能竞争序列，对于表 2 中连两个表项描述表编号 1，对应竞争节点 1 与竞争节点 2 共产生可能的竞争序列数目为都 6，具体可能的竞争情况见表 2。描述竞争节点之间可能的竞争顺序关系，根据如下情况提示程序设计人员进行合理修改，以避免可能竞争序列产生，带来不可预期的执行结果。

表 2 局部竞争序列

描述表编号	竞争序列号	竞争序列执行顺序描述		
		线程名	访问位置(行号)	操作状态
1	1	badcnt. c_count	30	R
		badcnt. c_count	30	W
		badcnt. c_count#1	30	R
		badcnt. c_count#1	30	W
1	2	badcnt. c_count	30	R
		badcnt. c_count#1	30	R
		badcnt. c_count	30	W
		badcnt. c_count#1	30	W
1	3	badcnt. c_count	30	R
		badcnt. c_count#1	30	R
		badcnt. c_count#1	30	W
		badcnt. c_count	30	W
1	4	badcnt. c_count#1	30	R
		badcnt. c_count	30	R
		badcnt. c_count	30	W
		badcnt. c_count#1	30	W
1	5	badcnt. c_count#1	30	R
		badcnt. c_count	30	R
		badcnt. c_count#1	30	W
		badcnt. c_count	30	W
1	6	badcnt. c_count#1	30	R
		badcnt. c_count#1	30	W
		badcnt. c_count	30	R
		badcnt. c_count	30	W

3 检测系统实现原型

基于上述方法，我们设计实现了数据竞争静态检测原型系统 RaceChecker，其结构如图 10 所示。RaceChecker 系统应用于某型号列车运行监控系统的数据竞争检测。该列车运行监控系统在 windows server2003 运行，系统采用 VS2008 工具开发，代码约 10 万行，经过人工处理，去掉不重要非关键代码，提取其中 28 个关键线程的源代码由该 RaceChecker 系统处理，发现 4 个可疑 bug，交工程师确认，工程师确认其中的 2 个 bug。

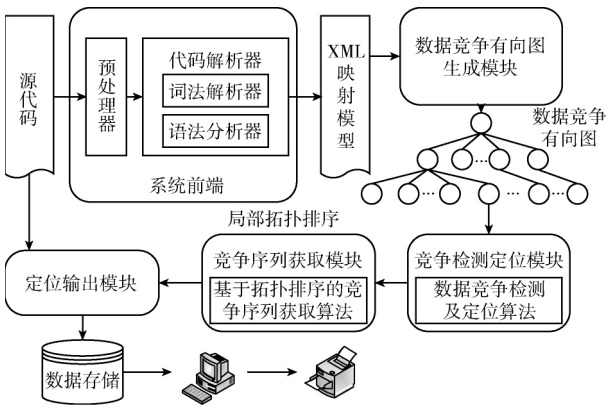

图 10 数据竞争静态检测系统原型

4 结束语

本文提出基于 XML 映射模型的静态分析方法、数据竞争检测及定位算法、基于拓扑排序的竞争序列获取算法，并设计实现了数据竞争静态检测原型系统 RaceChecker，旨在解决并行程序由于随即访问共享变量存在数据竞争和原子性侵犯问题。该系统目前已应用于某列车运行监控系统（10 万行）C 代码的数据竞争检测当中，成功发现多处 bug，实际运行结果表明该检测方法具有实用性及有效性。

参考文献：

[1] Weeratunge D, Zhang X, Sumner WN, et al. Analyzing concurrency bugs using dual slicing [C] //19th International Symposium on Software Testing & Analysis, 2010: 253-264.

[2] KONG Deguang, TAN Xiaobin, XI Hongsheng, et al. Analysis of multi thread program timing hidden Markov model [J]. Journal of Software, 2010, 21 (3): 461-472 (in Chinese). [孔德光, 谭小彬, 奚宏生, 等. 多线程程序时序分析的隐 Markov 模型 [J]. 软件学报, 2010, 21 (3): 461-472.]

[3] Sen K, Agha G. Concolic testing of multithreaded programs and its application to testing security protocols [C] //Computer Science Research and Tech Reports, 2011.

[4] Sen K, Viswanathan M. Model checking multithreaded programs with asynchronous atomic methods [G]. LNCS 4144: Proceedings of the 18th International Conference on Computer Aided Verification, 2010: 1-15.

[5] Lucas Cordeiro, Bernd Fischer. Verifying multi-threaded software using SMT-based context-bounded model checking [J]. ICSE, 2011, 5 (6): 331-340.

[6] Kong Deguang, Tan Xiaobin, Xi Hongsheng, et al. Hidden markov model for multi-thread programs time sequence analysis [J]. Journal of Software, 2010, 21 (3): 461-472.

[7] YANG Yu, ZHANG Jian. Program static analysis techniques and tools [J]. Computer Science, 2014, 31 (2): 171-174 (in Chinese). [杨宇, 张健. 程序静态分析技术与工具 [J]. 计算机科学, 2014, 31 (2): 171-174.]

[8] Chess B, Mcgraw G. Static analysis for security [J]. IEEE Security & Privacy, 2012, 2 (6): 32-35.

[9] Mcgraw G. Automated code review tools for security [J]. Computer, 2008, 41 (12): 108-111.

[10] LI Zhaojun, JIANG Fan. Defect semantic description language for static analysis [J]. Information Security and Communication Security, 2009 (10): 105-107 (in Chinese). [李朝君, 蒋凡. 用于静态分析的缺陷语义描述语言 [J]. 信息安全与通信保密, 2009 (10): 105-107.]

[11] Pu F, Lu WM. Preservation of liveness and deadlock-freeness in synchronous synthesis of Petri net systems [J]. Journal of Software, 2003, 14 (12): 1977-1988.

[12] GJB5369-2005 space model software C language security subset [S]. National Defense Science and Technology Industry Committee, 2005: 1-44 (in Chinese). [GJB5369-2005 航天型号软件 C 语言安全子集 [S]. 国防科学技术工业委员会, 2005: 1-44.]

[13] Samuel P, Harbison Guy L, Steele Jr C. C language reference manual [M]. XU Bo, transl. 5th ed. Beijing: Machinery Industry Press, 2008: 206-223 (in Chinese). [Samuel P, Harbison Guy L, Steele Jr C. C 语言参考手册 [M]. 徐波, 译. 5 版. 北京: 机械工业出版社, 2008: 206-223.]

[14] HU Xuan, LIU Bin. Concept and formal representation of code defect patterns [J]. Computer Engineering, 2010, 36 (7): 47-49 (in Chinese). [胡璇, 刘斌. 代码缺陷模式的概念及形式化表示 [J]. 计算机工程, 2010, 36 (7): 47-49.]

[15] LIANG Chengcai, ZHANG Daiyu, LIN Haijing. Synthesis of software defects [J]. Computer Engineering, 2012, 32 (19): 88-90 (in Chinese). [梁成才, 章代雨, 林海静. 软件缺陷的综合研究 [J]. 计算机工程, 2012, 32 (19): 88-90.]

[16] Buschmann F, Meunier R, Rohnert H, et al. Volume 1: A system of patterns [M] // Pattern-Oriented Software Architecture. Wiley Press, 1996.

[17] Viega J, Bloch JT, Kohno Y, et al. ITS4: A static vulnera-

bility scanner for C and C++ code [J]. Computer Security Applications, Acsac Conference, 2000: 257-267.

[18] LI Shaodong, XU Lei. A static detection method for BPEL data competition based on sequence and lock set [J]. Computer and Digital Engineering, 2010, 38 (8): 6-9 (in Chinese). [**李少东，许蕾. 一种基于发生序和锁集的** BPEL **数据竞争静态检测方法** [J]. **计算机与数字工程**, 2010, 38 (8): 6-9.]

[19] Bryant RE, O'Hallaron DR. Computer systems: A programmer's perspective [M]. HarperCollins, 1996.

(**上接第** 1251 **页**)

[3] Ma H, King I, Lyu M. Learning to recommend with explicit and implicit social relations [J]. Acm Transactions on Intelligent Systems & Technology, 2011, 2 (3): 29.

[4] Liu N, Qiang Y. EigenRank: A ranking-oriented approach to collaborative filtering [C] //International ACM SIGIR Conference on Research and Development in Information Retrieval, 2008: 83-90.

[5] SUN Jiankai, WANG Shuaiqiang, MA Jun. Weighted-Tau Rank: A ranking-oriented algorithm for collaborate filtering [J]. Journal of Chinese Information Processing, 2014, 28 (1): 33-40 (in Chinese). [**孙建凯，王帅强，马军**. Weighted-Tau Rank: **一种采用加权** Kendall Tau **的面向排序的协同过滤算法** [J]. **中文信息学报**, 2014, 28 (1): 33-40.]

[6] LI Mantian, WANG Jinlin, DENG Haojiang, et al. Ranking oriented algorithm for TOP-N recommendation [J]. Computer Simulation, 2013, 30 (5): 264-268 (in Chinese). [**李满天，王劲林，邓浩江，等. 一种面向排序的** TOP-N **推荐算法** [J]. **计算机仿真**, 2013, 30 (5): 264-268.]

[7] LI Gui, CHEN Shenghong, LI Zhengyu, et al. Recommendation algorithm with user temporal fusion [J]. 2014, 41 (6A): 394-399 (in Chinese). [**李贵，陈盛红，李征宇，等. 融合用户实效偏好的推荐算法** [J]. **计算机科学**, 2014, 41 (6A): 394-399.]

[8] WU Jun. Beauty of mathematics [M]. 2nd ed. Beijing: Posts & Telecom Press, 2014: 104-110 (in Chinese). [**吴军. 数学之美** [M]. 2 **版. 北京：人民邮电出版社**, 2014: 104-110.]

[9] Wang Shuaiqiang, Sun Jiankai, Byron JG, et al. VSRank: A novel framework for ranking-based collaborative filtering [J]. ACM Transaction on Intelligent Systems and Technology, 2014, 5 (3): 51: 1-24.

[10] SUN Jiankai. Research and implementation of ranking based personalized recommender algorithms [D]. Jinan: Shandong University, 2014 (in Chinese). [**孙建凯. 面向排序的个性化推荐算法研究与实现** [D]. **济南：山东大学**, 2014.]

[11] BO Jinjin. Baidu Wikipedia: Users' preferences [EB/OL]. [2016-02-10]. http://infq.lib.uh.edu/pacsl.html (in Chinese). [**波金金. 百度百科：用户偏好** [EB/OL]. [2016-02-10]. http://infq.lib.uh.edu/pacsl.html.]

[12] Clayton J, James A. Positive and negative opinion modeling: The influence of another's similarity and dissimilarity [J]. Journal of Personality and Social psychology, 2006, 90 (3): 440-452.

[13] Liu Y, Yang J. Improving ranking-based recommendation by social information and negative similarity [C] // Procedia Computer Science, 2015: 732-740.