

基于 Petri 网的多线程程序死锁检测

黄 理^{a,b,c}, 顾乃杰^{a,b,c}, 曹华雄^{a,b,c}

(中国科学技术大学 a. 计算机科学与技术学院; b. 安徽省计算与通信软件重点实验室; c. 先进技术研究院, 合肥 230027)

摘 要: 针对并发程序中死锁检测困难的问题, 基于 Petri 网对多线程程序进行死锁检测。定义抽象描述多线程程序中锁操作的 Petri 网模型, 在现有基于混合整数规划 (MIP) 的死锁检测算法基础上, 提出改进的 MIP 算法检测该模型中是否存在死锁。实验结果表明, 改进 MIP 算法能够检测到 Petri 网模型中的死锁, 与标志矩阵算法和可达图相比, 处理大规模多线程程序时计算效率更高。

关键词: 多线程程序; 锁操作; Petri 网; 混合整数规划; 死锁检测

中文引用格式: 黄 理, 顾乃杰, 曹华雄. 基于 Petri 网的多线程程序死锁检测[J]. 计算机工程, 2016, 42(4): 1-6.

英文引用格式: Huang Li, Gu Naijie, Cao Huaxiong. Deadlock Detection in Multi-threaded Program Based on Petri Net[J]. Computer Engineering, 2016, 42(4): 1-6.

Deadlock Detection in Multi-threaded Program Based on Petri Net

HUANG Li^{a,b,c}, GU Naijie^{a,b,c}, CAO Huaxiong^{a,b,c}

(a. School of Computer Science and Technology; b. Anhui Province Key Laboratory of Computing and Communication Software; c. Institute of Advanced Technology, University of Science and Technology of China, Hefei 230027, China)

[Abstract] Deadlock detection is in general difficult in concurrent program. Aiming at this problem, this paper proposes a method of deadlock detection in multi-threaded program based on Petri net. It defines a Petri net model which describes the lock operation in multi-threaded program. Based on the current deadlock detection algorithm using Mixed Integer Programming (MIP), it proposes an improved MIP algorithm to detect whether the model exists deadlock. Experimental results show that the improved MIP algorithm can detect the deadlock in Petri net model. It has higher computational efficiency when dealing with larger programs, compared with flag matrix algorithm and reachability graph.

[Key words] multi-threaded program; lock operation; Petri net; Mixed Integer Programming (MIP); deadlock detection

DOI: 10.3969/j.issn.1000-3428.2016.04.001

1 概述

随着多核处理器的快速发展, 多线程模型成为主流的并发编程模型, 但是, 由于并发行为的不确定性, 使得多线程程序的测试和开发相对困难, 其中最典型的问题就是死锁。文献[1]通过统计分析得出接近 30% 的并发问题与死锁有关的结论, 并且一旦出现死锁问题, 程序开发人员很难调试和修正。因此, 如何检测潜在死锁对于多线程程序的开发具有重要意义。

目前被广泛使用的死锁检测方法主要包括 2 类: 动态死锁检测和静态死锁检测。动态死锁检测需要程序运行才能检测, 它不能提前发现程序中可能的死锁, 而静态死锁解决了这个问题。目前静态死锁

检测的方法主要有: (1) 基于数据流分析技术检测并发程序中的死锁^[2]; (2) 模型检验方法^[3-4], 该方法的缺点是模型建模复杂, 难以手动构造; (3) 借助 Petri 网^[5]建模并通过虹吸、活性等性质来检测死锁。文献[6]基于 Petri 网对并程序通信死锁的检测和预防进行了研究。文献[7]利用 Petri 网对 Pthread 多线程程序进行建模, 并通过标志矩阵算法^[8]寻找 Petri 网的虹吸来检测死锁, 但检测结果准确度不高, 原因是标志矩阵算法没有考虑 Petri 网的初始标识, 不能体现网的动态执行过程。而活性体现了网的动态执行特点, 因此, 目前主要通过分析 Petri 网的活性来检测对应多线程程序的死锁^[9-10]。现有的活性检测方法主要有 2 种: (1) 基于可达图分析方法^[11], 通过求解所有使网死锁的死标识来检测网的活性,

基金项目: 安徽省自然科学基金资助项目“基于 GPU 集群的深度神经网络并行部署和优化策略研究”(1408085MKL06)。

作者简介: 黄 理 (1989 -), 男, 硕士研究生, 主研方向为并行计算; 顾乃杰, 教授、博士生导师; 曹华雄, 博士研究生。

收稿日期: 2015-03-31 **修回日期:** 2015-05-20 **E-mail:** lee123@mail.ustc.edu.cn

但该方法时间复杂度高,不适合分析规模较大的程序;(2)结合标识和虹吸来分析网的活性,进而检测系统的死锁。文献[12]在给定一个标识的情况下使用混合整数规划(Mixed Integer Programming, MIP)方法求解一个为空的极大虹吸,以检测结构有界 Petri 网是否存在死锁,但该方法未考虑连接弧权重大于 1 的情况。文献[13-14]定义 Gadara 网建模多线程程序,并通过求使网不活的死标识来检测死锁,但 Gadara 网只支持互斥锁。

基于以上的分析,本文针对 Pthread 多线程,定义一种锁操作的 Petri 网(Lock Petri Net, LOCK-PN)模型来描述多线程程序的并发操作,在基于 MIP 的死锁检测算法基础上,提出一种改进的 MIP 算法求解使 LOCK-PN 模型不活的死标识,以检测对应多线程程序是否存在死锁,并通过实例和多线程程序来验证该算法的可行性。

2 研究背景

2.1 Petri 网

一个 Petri 网^[5] Σ 由五元组 (P, T, F, W, M_0) 构成,其中, P 表示库所集; T 表示变迁集; $F \subseteq (P \times T) \cup (T \times P)$ 表示连接弧; $W: F \rightarrow \{1, 2, \dots\}$ 表示连接弧的权函数; M_0 表示由所有库所标记集合构成的初始标识。对于一个变迁 t , ${}^{\cdot}t(\cdot)$ 表示 t 的输入(输出)库所集合。若 $\forall t \in T$, 有 $|{}^{\cdot}t| = |\cdot t| = 1$, 则称网 Σ 是一个状态机。如果对任何 $f \in F$ 有 $W(f) = 1$, 则 Σ 是一个原型 Petri 网。

网 Σ 的标识 M 是一个映射 $M: P \rightarrow \{0, 1, \dots\}$, 连接弧 (p, t) 在 M 下是使能的, 当且仅当 $M(p) \geq W(p, t)$ 。变迁 t 在 M 下是使能的, 当且仅当对 $\forall p \in {}^{\cdot}t, M(p) \geq W(p, t)$ 。在任何标识 M 下可能有若干个变迁是使能的, 其中任何一个变迁使能后得到新的标识 M' , 即为 $M[>M'$, 其中, “[>”表示 M' 从 M 可达。从 M_0 可达的一切标识集记为 $R(M_0)$ 。如果在一个标识 M 下, 任何一个变迁都不使能, 则称 M 为死标识。如果存在死标识, 那么网 Σ 死锁。

网 Σ 的结构可以用关联矩阵 $A = [a_{ij}]_{n \times m}$ 表示, 其中, n 和 m 分别表示变迁和库所的总个数; $a_{ij} = W(t_i, p_j) - W(p_j, t_i)$ 。对于任何 $M \in R(M_0)$, $M = M_0 + A^T X$ 表示网 Σ 的状态方程, 其中, X 表示从 M_0 到 M 所经历的变迁序列。

如果对 $\forall t \in T$ 和 $\forall M \in R(M_0)$ 都存在 $M' \in R(M)$, 使得 t 在 M' 下使能, 那么称网 Σ 是活的。如果对所有的 $\forall t \in T$ 存在 $M \in R(M_0)$, t 在 M 下使能, 那么称网 Σ 是半活的。如果存在非空集合 $S, S \subseteq P$ 且 ${}^{\cdot}S \subseteq S^{\cdot}$, 那么 S 为网 Σ 的虹吸。一个 P-半流 Y 是一个 P-维向量, 满足 $\forall p \in P, Y(p) \rightarrow \{0,$

$1, \dots\}, \forall p \in P, Y(p) \rightarrow \{0, 1, 2, \dots\}$ 且 $Y^T A = 0$, $\|Y\| = \{p \in P | Y(p) \neq 0\}$ 表示 Y 的支集。

2.2 多线程程序的 Petri 网建模

多线程程序的特点是在主线程(main 线程)创建后, 根据需要创建一个或者多个子线程, 并且这些子线程之间是平等关系。主线程与子线程之间的关系主要有 2 种: 可会合(joinable)和相分离(detached), 前者表示在主线程继续运行前等待某个子线程执行完毕, 后者表示主线程与子线程之间是平等关系。为了方便讨论, 本文将主线程和子线程之间按照相分离关系来处理, 这样使得所有线程之间是平等关系, 并且每个线程有独立的控制流结构, 线程之间只有通过同步操作进行交互。相应地, 对多线程程序进行 Petri 网建模的过程可以分为 2 个部分:

(1) 确定线程的入口函数, 并对每个线程的控制结构进行建模^[7], 其中库所表示程序的状态, 变迁表示程序的执行, 并且每个变迁只有一个输入库所和输出库所, 使得每个线程对应的 Petri 网是一个状态机。

(2) 对线程的同步操作进行建模。Posix 线程提供的同步原语主要有锁、条件变量和信号量, 本文主要讨论锁实现的同步操作。Posix 线程提供的锁类型有互斥锁、自旋锁和读写锁, 这 3 种锁的建模方式如图 1 所示。

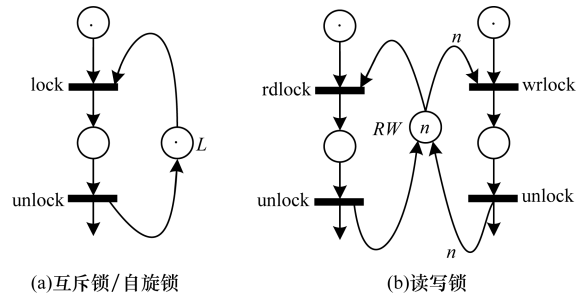


图 1 互斥锁、自旋锁和读写锁的 Petri 网建模方式

图 1 中的加锁和解锁操作是配对的。其中, L 表示互斥锁或自旋锁库所; RW 表示读写锁库所。 L 的初始标记为 1, 表示只有一个线程拥有这个锁, 而 RW 的初始标记为 n , 表示开始时可以加锁的次数。 RW 到变迁 $wrlock$ 的连接弧权重为 n , 表示一次只有一个线程可以占有写模式的读写锁。 L 和 RW 统称为资源库所。

3 LOCK-PN 模型的定义和性质

根据 2.2 节的建模过程, 定义一个描述多线程程序的 LOCK-PN 模型, 记为 $\mathcal{N}_T$, 网 $\mathcal{N}_T = (P, T, F, W, M_0)$, 并且对 $I_N = \{1, 2, \dots, n\}$ (n 表示线程数) 满足以下条件:

(1) $P = P_0 \cup P_S \cup P_R$, 其中, $P_0 = \bigcup_{i \in I_N} P_{0_i}$; P_S

$= \cup_{i \in I_N} P_{S_i}, P_{S_i} \neq \emptyset$, 且对 $\forall i \neq j, P_{S_i} \cap P_{S_j} = \emptyset; P_R = \{r_1, r_2, \dots, r_k\}, k > 0$ 。

(2) $T = \cup_{i \in I_N} T_i, T_i \neq \emptyset, \forall i \neq j, T_i \cap T_j = \emptyset$ 。

(3) $\forall i \in I_N$, 由 $\{p_{0_i}\} \cup P_{S_i} \cup T_i$ 构成的子网 $\mathcal{N}_i$ 是个状态机。

(4) $W = W_A \cup W_R$, 其中, $W_R: (P_R \times T) \cup (T \times P_R) \rightarrow N; W_A: ((P_A \cup P_0) \times T) \cup (T \times (P_A \cup P_0)) \rightarrow \{0\}$ 。

(5) $\forall r \in P_R$, 存在唯一的极小 P-半流 Y_r , 满足: $\{r\} = \|Y_r\| \cap P_R, P_0 \cap \|Y_r\| = \emptyset, Y_r(r) = 1$ 。此外, $P_S = \cup_{r \in P_R} (\|Y_r\| \setminus P_R)$ 。

(6) $\forall p \in P_S$, 如果 $|p^\cdot| > 1$, 那么 $\forall t \in p^\cdot, t \cap P_R = \emptyset$ 。

(7) 标识 M_0 满足: $\forall p \in P_S, M_0(p) = 0; \forall r \in P_R, M_0(r) \geq 1; \forall p_0 \in P_0, M_0(p_0) \geq 1$ 。

条件(1)表示网 $\mathcal{N}_T$ 的库所集合, 包括资源库所集合 P_R 、所有线程中的库所集合 P_S 和线程的开始库所集合 P_0 。一个线程从开始库所开始运行, 如果这个线程正常结束, 那么标记返回到开始库所, 并且任何 2 个线程之间的库所集合是独立的。

条件(2)表示 $\mathcal{N}_T$ 中所有变迁的集合, 任何 2 个线程之间的变迁集合是独立的。

条件(3)表示如果不考虑资源库所和相关的连接弧, 每个线程形成的子网 $\mathcal{N}_i$ 是个状态机。

条件(4)表示对网 $\mathcal{N}_T$ 相关连接弧的权重约束, 根据定义可得网 $\mathcal{N}_T$ 不是一个普通 Petri 网。

条件(5)表示加解锁操作是配对的, 并且由于资源库所中初始标记的值总是正整数, 由此可得 $Y_r(r) = 1$ 。

条件(6)规定了程序中跳转分支所属变迁 t 的前集一定不包含资源库所。

条件(7)规定了网 $\mathcal{N}_T$ 的初始标识, 其中除了开始库所和资源库所外, 其余库所都不含标记。

性质 1 网 $\mathcal{N}_T$ 是半活的。

证明: 根据网 $\mathcal{N}_T$ 定义的条件(3)和条件(7)可得, 对于网 $\mathcal{N}_T$ 中任何变迁 t , 存在一条从开始库所到 t 的变迁使能序列, 使得 t 最终使能, 因此, t 是半活。 $\forall t \in T$, 满足 t 是半活, 因此, $\mathcal{N}_T$ 是半活的, 得证。

性质 2 如果网 $\mathcal{N}_T$ 不活, 那么至少存在一个标识 $M \in R(M_0)$ 和一个子网 $\mathcal{N}_i, t \in T_i$ 在标识 M 下是死变迁, 并且满足 $M(p_{0_i}) < M_0(p_{0_i})$ 。

证明: 假设这个结果不成立, 则表明 M_0 从 M 可达, 而由性质 1 可得 $\mathcal{N}_i$ 是半活的, 此时变迁 t 在标识 M 下使能, 即 t 不是死变迁, 与前提矛盾, 得证。

推论 如果网 $\mathcal{N}_T$ 存在死变迁 t , 那么存在资源库所 $r \in P_R$ 和标识 $M \in R(M_0)$ 满足 $|t^\cdot| > 1, t \cap r \neq \emptyset$, 连接弧 (r, t) 在 $M' \in R(M)$ 下不使能导致变迁 t

是死的。

证明: 假设死变迁 t 的发生与任何资源库所 $r \in P_R$ 无关, 根据网 $\mathcal{N}_T$ 中子网 $\mathcal{N}_i$ 是状态机结构可得 t 在任何标识 $M \in R(M_0)$ 是使能的, 即 t 是活的, 与前提矛盾, 得证。

4 基于 LOCK-PN 的死锁检测

4.1 基于 MIP 的死锁检测算法

文献[12]针对结构有界的普通 Petri 网系统, 首先使用 MIP 方法检测系统是否存在死锁。该方法描述如下: 给定一个虹吸 S , 引入 2 个指示器, 如式(1)所示。

$$v_p = 1 \{p \notin S\}, z_t = 1 \{t \notin S^\cdot\} \quad (1)$$

根据虹吸 S 的定义, 有 $v_p = 0 \Rightarrow z_t = 0, \forall t \in p^\cdot; z_t = 1 \Rightarrow v_p = 1, \forall p \in t^\cdot$, 进而得到性质 3。

性质 3 在标识 M 下, 存在一个极大的虹吸 S 使得网 Σ 死锁, 当且仅当 $G(M) < |P|$, 其中:

$$G(M) = \min \sum v_p \quad (2)$$

满足:

$$z_t \geq \sum_{p \in t^\cdot} v_p - |t^\cdot| + 1, \forall t \in T \quad (3)$$

$$v_p \geq z_t, \forall (t, p) \in F \quad (4)$$

$$M = M_0 + A^T X, M \geq 0, X \geq 0 \quad (5)$$

$$v_p \geq M(p)/SB(p), \forall p \in P \quad (6)$$

在式(6)中, $SB(p) = \max \{M(p) \mid M = M_0 + A^T X, M \geq 0, X \geq 0\}$ 表示库所 p 的上界。如果式(2)有解, 求解得到的虹吸 $S = \{p \in P \mid v_p = 0\}$ 使得网 Σ 死锁。为了下文描述方便, 称通过式(2)求解虹吸 S 为 MIP 算法。

4.2 基于 LOCK-PN 的死锁检测

MIP 算法对检测 LOCK-PN 模型中的死锁存在 2 个不足: (1) 约束条件没有考虑到网中连接弧的权重大于 1 的情况; (2) 约束条件过多, 对求解规模较大程序的时间复杂度较高。基于这两点, 本文在 MIP 算法的基础上, 添加决策变量和约束条件以支持对第(1)个不足的讨论; 同时充分利用网 $\mathcal{N}_T$ 的结构特点, 将指示器 z_t 转换为对所有与变迁 t 相连的库所和连接弧的讨论, 以减少约束条件数量。

首先在式(1)的基础上, 修改 v_p 的含义, 同时引入符号 f_{pt} 和 $\bar{M}(p)$, 其定义如下:

$$v_p = \begin{cases} 1 & \forall t \in p^\cdot, M(p) \geq W(p, t) \\ 0 & \text{else} \end{cases} \quad (7)$$

$$f_{pt} = \begin{cases} 1 & M(p) \geq W(p, t) \\ 0 & \text{else} \end{cases} \quad (8)$$

$$\bar{M}(p) = \begin{cases} \bar{M}(p) & p \notin P_0 \\ 0 & p \in P_0 \end{cases} \quad (9)$$

式(7)中 v_p 表示只要任何一个弧 (p, t) 不使能, 那么 $v_p = 0$, 否则 $v_p = 1$; 式(8)中 f_{pt} 为连接弧 (p, t) 是否使能的标记; 式(9)中 $\bar{M}$ 为不考虑开始库所的标记。引入这 2 个标记可以更好地刻画网 $\mathcal{N}_T$ 运行时的状态。在此基础上给出以下引理, 其证明思路同文献[13]类似。

引理 给定一个网 $\mathcal{N}_T$, 它是活的, 当且仅当网 $\mathcal{N}_T$ 不存在死标识 $\bar{M}$ 和虹吸 S , 其中 S 是在标识 $\bar{M}$ 下所有 $v_p = 0$ 的库所构成的, 并且至少包括一个资源库所。

证明:

(必要性) 假设网 $\mathcal{N}_T$ 不活, 根据性质 2 可得, 网 $\mathcal{N}_T$ 中存在一个标识 $M' \in R(M_0)$ 和至少一个子网 $\mathcal{N}_i, t \in T$ 在 M' 下是死变迁, 并且 t 在 $R(M')$ 下都是死的, 则对剩余不存在死变迁的子网而言, 从标识 M' 开始, 存在变迁序列使得这些子网标记返回到开始库所。设此时网 $\mathcal{N}_T$ 的标识为 $M, M \in R(M')$, 根据 $\bar{M}$ 的定义, 网 $\mathcal{N}_T$ 在标识 $\bar{M}$ 下死锁, 并且根据虹吸的定义可得, 在标识 $\bar{M}$ 下所有 $v_p = 0$ 的库所构成一个虹吸 S 。另外根据上文推论可得 S 中包括资源库所, 与前提矛盾, 得证。

(充分性) 假设网 $\mathcal{N}_T$ 中存在死标识 $\bar{M}$, 并且标识 $\bar{M}$ 下所有 $v_p = 0$ 的库所构成的虹吸 S 至少包括一个资源库所 r 。考虑 $t \in r^\cdot$, 连接弧 (r, t) 不使能, 根据虹吸的定义, t 永远不使能。如果进一步引入开始库所的标记将 $\bar{M}$ 转换为原来的 M , 根据网 $\mathcal{N}_T$ 定义可得, 对于开始库所的输出变迁 $t_1 \in P_0^\cdot$, 满足 $r \notin t_1^\cdot$, 这表明在标识 M 下 r 仍然不会获得标记, 因此, t 是死变迁, 从而可得网 $\mathcal{N}_T$ 不是活的, 与前提矛盾, 得证。

上述引理说明可以通过求解死标识 $\bar{M}$, 在标识 $\bar{M}$ 下网中所有 $v_p = 0$ 的库所集合构成虹吸 S , S 使得网 $\mathcal{N}_T$ 不活。根据网 $\mathcal{N}_T$ 的结构, 可将网 $\mathcal{N}_T$ 中的变迁 t 分为如图 2 所示的 2 种形式。

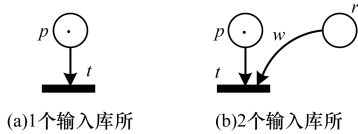


图 2 网 $\mathcal{N}_T$ 中变迁的分类

具体分析如下:

(1) 图 2(a) 中 t 有 1 个输入库所, 如果 t 不使能, 那么根据上述讨论可得 $v_p = \bar{M}(p) = 0$ 。

(2) 图 2(b) 中 t 有 2 个输入库所, 如果 t 不使能, 那么至少有 1 个输入连接弧不使能。

(3) 存在某个死变迁 $t \in T, t$ 的输入库所包含 p 和 $r \in P_R, p = {}^\cdot t \cap P_S$, 有 $M(p) \geq W(p, t), v_p = 1, f_{rt} = 0$, 表示连接弧 (r, t) 不使能。

(4) 由于读写锁、互斥锁引起的死锁是以循环等待的形式出现, 因此有: 网 $\mathcal{N}_T$ 至少有 2 个资源库所满足 $v_p = 0$ 。

上述 4 个结论恰好可以作为 MIP 方法的约束条件。基于这些认知, 本文提出一种用于求解死标识 $\bar{M}$ 的 MIP- $\mathcal{N}_T$ 算法, 该算法的目标方程和约束条件如下:

$$\min: \sum_{p \in P_S} v_p \quad (10)$$

满足:

$$\forall p \in P_0, v_p = 0 \quad (11)$$

$$\sum_{p \in P_S} v_p \geq 2 \quad (12)$$

$$\sum_{p \in P_R} v_p \leq |P_R| - 2 \quad (13)$$

对于 $|{}^\cdot t| > 1, \forall p \in P_S$, 有:

$$\bar{M}(p) = v_p = 0, p \in {}^\cdot t, |{}^\cdot t| = 1 \quad (14)$$

$$v_p + f_{rt} \leq |{}^\cdot t| - 1, |{}^\cdot t| > 1, r^\cdot \cap t \neq \emptyset \quad (15)$$

$$v_p \leq \bar{M}(p) \quad (16)$$

对于 $\forall p \in P_R, \forall t \in p^\cdot$ 且 $W(p, t) > 0$, 有:

$$f_{pt} \geq \frac{M(p) - W(p, t) + 1}{SB(p)} \quad (17)$$

$$f_{pt} \geq v_p \quad (18)$$

$$\sum_{t \in p^\cdot} f_{pt} - |p^\cdot| + 1 \leq v_p \quad (19)$$

$$v_p, f_{pt} \in \{0, 1\} \quad (20)$$

$$M = M_0 + A^T X, M \geq 0, X \geq 0 \quad (21)$$

$$\bar{M}(p) = \begin{cases} M(p) & p \notin P_0 \\ 0 & p \in P_0 \end{cases} \quad (22)$$

式(10)为目标函数, 表示最少的线程数产生死锁。

式(11)是根据 $\bar{M}$ 的定义得到。

式(12)、式(13)是结论(3)的体现。

式(14)、式(15)是结论(1)和结论(2)的体现。

式(16): 根据式(7)和式(9)可得对 $\forall p \in P_S$, $\bar{M}(p) = M(p)$ 。若 $v_p = 1$, 则 $\bar{M}(p) \geq 1$; 若 $v_p = 0$, 则 $\bar{M}(p) \geq 1$ 。因此 $v_p \leq \bar{M}(p)$, 式(16)成立。

式(17): 对 $\forall r \in P_R, t \in r^\cdot$, 如果弧 (r, t) 使能, 则 $f_{rt} = 1$, 并且对 $M(r) \geq W(r, t)$, 有 $M(r) - W(r, t) + 1 \geq 1$, 因此, 式(17)可满足; 如果弧 (r, t) 不使能, 那么 $f_{rt} = 0$, 式(15)可满足。其中, $SB(r)$ 表示资源库所 r 的上界, 一般来说, 对互斥锁而言, $SB(r)$ 等于 1, 对读写锁来说, $SB(r)$ 表示开始时读者可加锁的次数。

式(18)、式(19): 对任何资源库所 r , 如果任何一个弧 (r, t) 不使能, 那么 $v_r = 0$, 式(18)可满足; 如果 r 的标记使得弧 (r, t) 都可以使能, 那么式(19)可满足。

式(20) ~ 式(22): 式(20)用于限定 v_p 和 f_{pt} 的值; 式(21)用于确定状态空间; 式(22)表示 $\bar{M}$ 和 M 的关系。

如果式(10)有解, 则根据引理 1 的证明过程可知求解得到的 $\bar{M}$ 导致网 $\mathcal{N}_T$ 不是活的; 如果式(10)无解, 说明网 $\mathcal{N}_T$ 不存在死标识 $\bar{M}$, 由 4.2 节引理可得网 $\mathcal{N}_T$ 是活的。

5 实验与结果分析

本文使用 C 语言和求解线性规划问题的开源工具 GLPK^[15] 编写相应的建模和死锁检测算法代码。首先通过案例分析描述算法的改进效果; 然后利用 MIP- $\mathcal{N}_T$ 算法和现有的死锁检测算法检测真实程序中的死锁。以此表明本文提出的 MIP- $\mathcal{N}_T$ 算法是可行和高效的。

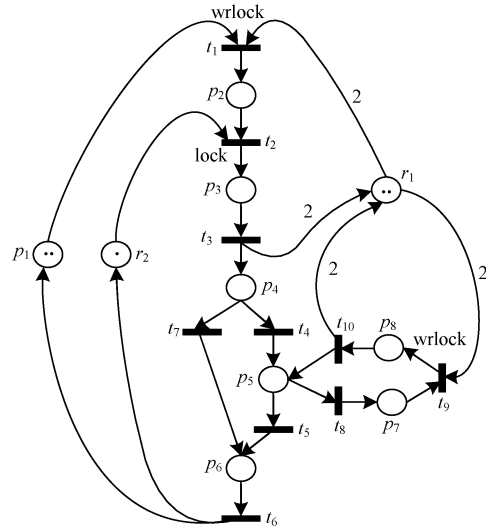
实验环境为: CPU 为 Intel (R) Xeon (TM) @ 2.10 GHz, 内存 4 GB, 操作系统: Redhat 4.8.2, 内核版本为 3.11.10。

5.1 案例分析

线程 T1 和线程 T2 独立执行如下的程序:

```
Program( void ) { ...
pthread_rwlock_wrlock( r1 );
pthread_mutex_lock( r2 );
pthread_rwlock_wunlock( r1 );
if ( ... ) {
    for ( ... ) {
        pthread_rwlock_wrlock( r1 );
        pthread_rwlock_wunlock( r1 );
    }
}
pthread_mutex_lock( r2 );
}
```

上述代码为一个简化的 Linux 多线程程序 (仅包括锁操作语句、选择和循环语句), 该程序包括 2 个线程, 并且这 2 个线程的入口函数都为 Program, 线程间通过读写锁 r_1 和互斥锁 r_2 实现同步。对 Program 进行建模得到如图 3 所示的 Petri 网 $\mathcal{N}_T$, 其中, 开始库所 p_1 的初始标记为 2, 表示线程 T1 和 T2 同时执行; r_1 表示读写锁库所, 初始标记为 2, r_2 表示互斥锁库所。如果 T1 依次激发变迁 t_1, t_2, t_3, t_4, t_8 , T2 激发变迁 t_1 后, 网 $\mathcal{N}_T$ 的标识为 $M = p_2 + p_7$, 此时 M 是一个死标识, 在标识 M 下 p_1 代表的网 $\mathcal{N}_T$ 不能执行, 网 $\mathcal{N}_T$ 对应的多线程程序死锁。对网 $\mathcal{N}_T$ 分别利用 MIP 算法和 MIP- $\mathcal{N}_T$ 算法求解, 得到结果如表 1 所示。



入口函数和锁变量,并对线程的控制流结构进行建模;然后依次分析 2 个和 3 个线程,对每次待分析的 $m(2 \leq m \leq 3)$ 个线程,组合这 m 个线程的入口函数,包括对同一个入口函数的线程进行组合(除了主线程外),并建立这 m 个线程对应的网 $\mathcal{N}_T$;最后通过 MIP- $\mathcal{N}_T$ 算法、标识矩阵算法和可达图检测网 $\mathcal{N}_T$ 中是否存在死锁。实验结果如表 3 和表 4 所示。表 3 中检测的平均时间表示算法计算所有组合的线程所消耗的平均时间;表 4 中数字表示是否检测到死锁(1 表示检测到;0 表示未检测到)。

表 3 死锁检测的平均时间 s

程序	本文算法	标志矩阵算法	可达图
DP	0.00	0.003	0.004
Apache	6.28	342.343	>1 512
OpenLDAP	9.57	523.237	>1 398
Pfscan	0.00	9.748	12.341

表 4 死锁检测结果对比

程序	本文算法	标志矩阵算法	可达图
DP	1	1	1
Apache	0	0	0
OpenLDAP	1	1	0
Pfscan	0	0	0

由表 3 和表 4 可得,在检测出同样结果的前提下,MIP- $\mathcal{N}_T$ 算法消耗的平均时间比其他 2 个算法少,并且这种优势在规模较大的程序下体现得越明显。其中,可达图由于受到内存的限制,在分析程序 Apache 和 OpenLDAP 超过 20 min 后仍未检测出死锁,而标识矩阵算法的计算复杂度与关联矩阵中变迁和库所的数量有关,对于规模较大的 Apache 和 OpenLDAP 程序,其建模后得到的变迁和库所较多,因此,时间复杂度相对较高。由上述分析可知,相比其他死锁检测算法,本文提出的 MIP- $\mathcal{N}_T$ 算法更适合处理规模较大的多线程程序。

6 结束语

本文定义一种描述多线程程序锁操作的 Petri 网模型,在现有 MIP 算法的基础上,提出 MIP- $\mathcal{N}_T$ 算法检测 LOCK-PN 模型是否存在死锁。应用结果表明,该算法能检测到 LOCK-PN 模型中的死锁,与现有基于 Petri 网的死锁检测算法相比,时间复杂度更低。由于本文只针对锁类型的死锁进行检测,因此下一步工作将研究多线程程序中与条件变量、信号量相关的 Petri 网建模和死锁检测。

参考文献

- [1] Lu Shan, Park S, Seo E. Learning from Mistakes: A Comprehensive Study on Real World Concurrency Bug Characteristic [J]. ACM SIGOPS Operating Systems Review, 2008, 42(2): 329-339.
- [2] Kahlon V, Sankaranarayanan S, Gupta A. Static Analysis for Concurrent Programs with Applications to Data Race Detection [J]. International Journal on Software Tools for Technology Transfer, 2013, 15(4): 321-336.
- [3] Bensalem S, Griesmayer A, Legay A, et al. Efficient Deadlock Detection for Concurrent Systems [C]// Proceedings of the 9th IEEE/ACM International Conference on Formal Methods and Models for Codesign. Washington D. C., USA: IEEE Press, 2011: 119-129.
- [4] Sun Jun, Yang Liu, Jin Songdong. Modeling and Verifying Hierarchical Real-time Systems Using Stateful Timed CSP [J]. ACM Transactions on Software Engineering and Methodology, 2013, 22(1): 1-29.
- [5] 袁崇义. Petri 网原理与应用 [M]. 北京: 电子工业出版社, 2005.
- [6] 崔焕庆, 刘 强. 基于 Petri 网并行程序通信死锁的检测和预防 [J]. 计算机工程, 2008, 34(23): 50-52.
- [7] Wang Yin, Lafortune S, Kelly T. The Theory of Deadlock Avoidance via Discrete Control [J]. ACM SIGPLAN Notices, 2009, 44(1): 252-263.
- [8] Boer E R, Murata T. Generating Basis Siphons and Traps of Petri Nets Using the Sign Incidence Matrix [J]. IEEE Transactions on Circuits and Systems, 1994, 41(4): 266-271.
- [9] Liao Hongwei, Wang Yin, Stanley J. Eliminating Concurrency Bugs in Multithreaded Software: A New Approach Based on Discrete Event Control [J]. IEEE Transactions on Control Systems Technology, 2013, 21(6): 2067-2082.
- [10] 朱承丞, 董利达. 一种多线程软件并发漏洞检测方法 [J]. 西安电子科技大学学报, 2015, 42(2): 167-173.
- [11] 林 英. 多核软件形式化建模、验证和性能评价研究方法 [D]. 昆明: 云南大学, 2013.
- [12] Feng Chu, Xie Xiaolan. Deadlock Analysis of Petri Nets Using Siphons and Mathematical Programming [J]. IEEE Transactions on Robotics and Automatic, 1997, 13(6): 793-804.
- [13] Liao Hongwei, Wang Yin, Cho H K, et al. Concurrency Bugs in Multithreaded Software: Modeling and Analysis Using Petri Nets [J]. Discrete Event Dynamic Systems, 2013, 21(6): 2067-2082.
- [14] Liao Hongwei, Lafortune S, Reveliotis S, et al. Optimal Liveness-enforcing Control of a Class of Petri Nets Arising in Multithreaded Software [J]. IEEE Transactions on Automatic Control, 2013, 58(5): 1123-1138.
- [15] GLPK (GNU Linear Programming Kit) [EB/OL]. (2012-06-23). <http://www.gnu.org/software/glpk/>.

编辑 金胡考