第 19 卷第 2 期　　广西科学院学报　　Vol. 19, No. 2
2003 年 5 月　　Journal of Guangxi Academy of Sciences　　May 2003

一种基于并行技术的死锁检测算法
An Algorithm of Deadlock Detection Based on Concurrent Technology

陈　岚
Chen Lan

（广东肇庆学院计算机科学系　肇庆　526061）
(Dept. of Comp. Sci, Guangdong Zhaoqing Univ., Zhaoqing, 526061)

摘要　在介绍基于资源分配图的、传统的死锁检测算法基础上，提出一种新的基于并行技术的死锁检测算法，并用 1 个实例说明该算法的执行过程。新的死锁检测算法是基于矩阵表示方法，在最坏情况下，运行时间复杂度是 O (min (m,n))，其中 m 和 n 分别是进程和资源的数量。新的死锁检测算法与传统的算法相比，执行时间大大减少，需要内存也比较小，系统能够很好地检测死锁的发生，并且释放占有资源。

关键词　操作系统　死锁　检测算法　矩阵表示　并行处理
中图法分类号　TP316

Abstract　A new algorithm of deadlock detection based on concurrent technology is given with citing of an executive processes, which was introduced by the traditional algorithm based on the use of the resource allocation graph. The worst case cost of the new algorithm, which based on the expression of matrix, is O (min (m,n)), m is the number of processes and n is the number of resource. Compared with the traditional one, the new algorithm's decreasing in execute time greatly, and lesser in memory requirements. The system could detected the deadlock and released the possessive resource effectively.

Key words　operation system, deadlock, detection algorithm, matrix express, concurrent operation

1　操作系统中的死锁问题

在多道程序系统中，为了最大限度地利用计算机系统的资源，操作系统采用动态分配各类资源的策略。然而，系统中不少资源是互斥地使用的，若对资源分配不当，则可能造成进程间由于竞争资源而相互制约以至于无法继续运行的局面，人们把这种局面称为死锁[1]。一般情况下，我们把系统死锁的形式描述为一组并发进程 X_1、X_2、X_3、…X_n，它们共享资源 Y_1、Y_2、Y_3、…Y_m（$n>0$、$m>0$、$n>m$）。其中，每个 X_i（$1\leqslant i\leqslant n$）拥有资源 Y_j（$1\leqslant j\leqslant m$），直到再没

2003-03-16 收稿，2003-04-26 修回。

有其它剩余资源。同时，各 X_i 又在不释放 Y_j 的情况下，要求得到 $Y_k(1 \leqslant k \leqslant m, k \neq j)$，从而造成资源的互相保持和互相等待。在这种情况下，若无外力驱动，这些并发进程将永远不能再向前推进而陷入永久等待状态。

从上可知，产生死锁的根本原因是系统中共享资源和资源不足。由于对资源的共享而引起对资源的竞争，或进程在运行过程中请求和释放资源的顺序不当，导致进程陷入死锁。一般对死锁的解决方法包括预防、避免、检测和恢复。

死锁预防是在系统为进程分配资源时施加限制条件，死锁避免是在系统为进程分配资源时进行检测[2]。若系统为进程分配资源时，未采取任何限制性措施，则系统必须提供检测和解除死锁的手段，这需要提供一种算法，利用保存的有关资源请求和分配的信息，来检测系统是否已进入死锁状态。设立一个检测进程（平时处于睡眠状态）周期性地被唤醒去检查系统中有无进程陷入死锁。人们通常采用化简资源分配图的方法来检测系统是否处于安全状态。

2　传统的死锁检测算法

2.1　资源分配图[3,4]

资源分配图 $G=(V,E)$，其中顶点集 $V=P \cup R$。P 是进程集合，$P=\{p_1,p_2,\cdots,p_m\}$，p_i 表示系统中的第 i 个进程，在图中用“○”表示，R 是资源类集合，$R=\{r_1,r_2,\cdots,r_n\}$，r_i 表示系统中的第 i 类资源，在图中用“□”表示，r_i 类资源若为 n 个，则在“□”中画 n 个圆点。E 是边的集合，E 有两类边，一类是请求边 (p_i,r_j)，表示进程 p_i 请求一个单位的 r_j 类型的资源，另一类是分配边 (r_j,p_i)，表示进程 p_i 占有一个单位的 r_j 类型的资源。

资源分配图在系统中是随时间变化的。当进程 p_i 请求 r_j 时，将一条请求边 (p_i,r_j) 加在图中。若此请求被满足（分给 p_i 一个 r_j 类的资源），则将这个请求边改成分配边 (r_j,p_i)。当进程 p_i 释放一个 r_j 时，便去掉分配边 (r_j,p_i)。

2.2　基于资源分配图的一个传统算法

有了资源分配图，可以利用对资源分配图加以简化的方法，来检测系统处于 S 状态时，是否存在死锁现象。简化方法如下。

(1) 在资源分配图中找出一个既不阻塞又不独立的进程结点 p_i。在顺利的情况下，p_i 在请求后获得所需资源而被执行，直至运行完毕，再释放其所占有的全部资源。这相当于消去 p_i 所有的请求边和分配边，使之成为独立的结点。如图 1 中，将 p_1 的 2 个分配边和 1 个请求边消去，使之成为图 2 所示的情况。

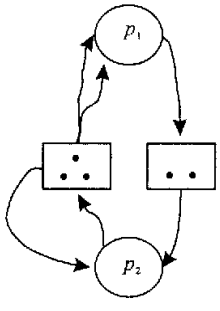

图 1　资源分配图

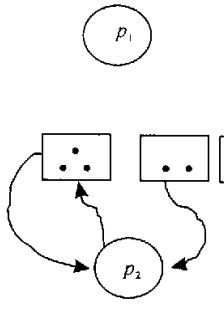

图 2　资源分配图的简化过程

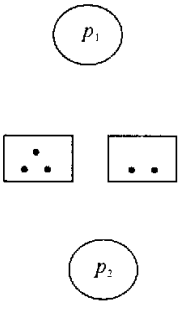

图 3　资源分配图的简化结果

(2) p_1 释放资源后，便可使 p_2 获得资源而继续运行，直至 p_2 完成后又释放出它所占有的全部资源，进而形成图 3 所示的情况。

(3) 进行一系列的简化后，如果能消去所有的边，使所有的进程都成为独立的结点，则称

该图是可完全简化的。如果不能通过任何措施使该图简化，则该图是不可完全简化的。如果 S 状态的资源分配图是不可完全简化的，则 S 为死锁状态，这就是死锁定理[1]。

这种基于资源分配图的实现算法在最坏的情况下，时间复杂度是 $O(m \times n)$，其中 m 和 n 分别是进程和资源的数目。

3 一种新的死锁检测算法及其分析

新算法的提出是基于矩阵表示的，下面首先介绍死锁检测问题的矩阵表示，然后提出算法的一些重要特征。本算法基于并行处理技术，能够同时处理多 requests/grands 问题，并且能够检测多重死锁，可以很显著地提高性能。

3.1 死锁检测问题的矩阵表示

在图论中，任何有向图都能够用邻接矩阵表示[2,5]。因此资源分配图也可以用邻接矩阵表示。它有两种类型的边：资源请求边，由进程 p_i 指向资源 q_j，它表示进程 p_i 请求 1 个单位的 r_j；资源分配边，由资源 q_j 指向进程 p_i，它表示把 1 个单位的资源 r_j 分配给了进程 p_i。为了区分不同类型的边，在邻接矩阵中，用 3 种不同的值来表示，见表 1。表 1 中最左的一列表示进程（p_1、p_2、$p_3 \cdots p_i \cdots p_m$）标识列，最上一行是资源（$q_1$、$q_2$、$q_3 \cdots q_j$，$\cdots q_n$）标识行。如果资源分配图中存在资源请求边 (p_i, q_j)，矩阵中相应的值定义为 r；如果有资源分配边 (q_i, p_j)，矩阵中相应的值定义为 g；否则值为 0。

所以，任何资源分配图 $RAG(V,E)$，V 代表节点，E 代表边，它的邻接矩阵可定义如下：

$M = [m_{ij}]^{m \times n} (1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n)$，$m$ 代表进程的数目，n 代表资源数目。

$m_{ij} \in \{r, g, 0\}$

如果存在 $(p_i, q_j) \in E$，则 $m_{ij} = r$，

如果存在 $(q_i, p_j) \in E$，则 $m_{ij} = g$，

否则 $m_{ij} = 0$.

此矩阵提供了 1 个分配和请求的模板。值得注意的是每个资源最多有 1 个资源分配，也就是说一列最多只有 1 个 g 值，而 1 个进程的需求数目则没有限制。

如果系统中存在死锁现象，在资源分配图至少有 1 个循环链，即有一系列的边 $\varepsilon = \{(p_{i1}, q_{j1}), (q_{j1}, p_{i2}), \cdots, (p_{ik}, q_{jk}), (q_{jk}, p_{i(k+1)}), \cdots, (p_{is}, q_{js}), (q_{js}, p_{i(s+1)})\}, \varepsilon \in E$。如果映射到矩阵中，则此循环可表示为 $\delta = \{m_{i1j1}, m_{i2j1}, \cdots, m_{ikjk}, m_{i(k+1)jk}, m_{i(k+1)j(k+1)}, m_{isjs}, \cdots m_{i1js}\}$，其中 $m_{i1j1}, m_{i2j2}, \cdots, m_{isjs}$ 均为资源需求，$m_{i2j1}, m_{i3j2}, \cdots, m_{i1js}$ 均为资源分配。因此，系统中我们可以通过邻接矩阵来检测死锁。

3.2 新的检测算法的实现

新的死锁检测算法基于邻接矩阵表示，并根据以下情况进行逐步精简矩阵。

(i) 全为 0 的 1 行或 1 列；

(ii) 源点（1 行有 1 个或者多个 r 但是没有 g 值，或者 1 列有 1 个 g 值但是没有 r 值）；

(iii) 汇点（1 行有多个 g 值但是没有 r 值，或者 1 列有 1 个或多个 r 值，但没有 g 值）。

反复执行以上步骤，直到矩阵不能再简化为止。如果此时矩阵中仍有非 0 元素，说明有

表 1 1 个给定系统的矩阵表示

	q_1	q_2	……	q_n
p_1	g	r	……	0
p_2	r	g	……	0
……	……	……	……	……
p_m	0	r	……	g

死锁存在，否则不存在死锁。

算法实现：Parallel Deadlock Detection

Step 0：initialization

$M=[M_{ij}]^{m\times n}$where $m_{ij}\in\{r, g, 0\}$ $(1\leqslant i\leqslant m, 1\leqslant j\leqslant n)$

$\Lambda=\{m_{ij}|m_{ij}\in M, m_{ij}\neq 0\}$;

Note that Λ is a set consisting initially of all the non-zero entries in the matrix M

Step 1：remove all the sinks and sources

Do

{

Reducible=0;

For each column:

If $(\ni m_{ij}\in\Lambda|\forall_k, k\neq i, m_{kj}\in\{m_{ij}, 0\})$　∋—存在

{

Λcolumn=Λ−$\{m_{ij}|j=1, 2, 3, \cdots, m\}$,

Reducible=1;

}

else {}

For each row:

If $(\ni m_{ij}\in\Lambda|\forall k, k\neq j, m_{ik}\in\{m_{ij}, 0\})$　∋—代表存在

{

Arow=Λ−$\{m_{ij}|j=1, 2, 3, \cdots, n\}$,

Reducible=1;

}

else {}

Λ=Λcolumn∩Arow;

} Until (reducible=0);

Step 2：Detect Deadlock

If $(\Lambda\neq\Phi)$, then deadlock exists

If $(\Lambda=\Phi)$, then no deadlock exists

如果矩阵是可简化的，此并行算法的每次叠代都至少有 1 次简化。对于 m 行 n 列的矩阵，算法所做的操作次数显然关于 m 或 n 是线性的，它最多由 min (m,n) 次叠代来完成死锁检测，算法在最坏的情况下，时间复杂度为 O (min (m,n))。下面的例子演示出此算法的流程。

例 1　2 个进程和 3 单位资源。

本例中有 2 个进程：*DSP* 和 *VSP*，分别定为 p_1 和 p_2。设备有 3 个单位资源：*ICP*，*PCI* 和 *WI*，分别定为 q_1，q_2，q_3，如图 4 所示。

例 1 的矩阵表示如表 2 所示。表 2 中先扫描列，矩阵的第 1 列和第 2 列都包含 g 值和 r 值，不可约。第 3 列只有 g 值，可被约。矩阵的每一行均有 g 和 r，不可约，接下来继续叠代以上过程，经过 1 次化简后的矩阵，第 1、2 列仍有 g 值和 r 值，列不可约。同时扫描行，同样每一行都有 g 和 r，不可约。所以通过此算法检测进行了 2 次叠代，发现 1 个死锁。

若删除分配边(q_2,p_2),此时矩阵表示如表3所示。扫描列,第1列包含有g和r,不可约,而第2列和第3列均可约,因为第2列只有1个r值没有g值,第3列只有1个g值没有r值。同时扫描行,第1行第2行均不可约,因为都有g和r值存在。所以第1次叠代过程可被化简。算法中的step1重复执行删除第2和第3列。接下来进行第2次叠代过程,第1列不可约,不过第1行和第2行均可约,step1再次被执行,此时矩阵为空。算法step2执行,检测到没有死锁。

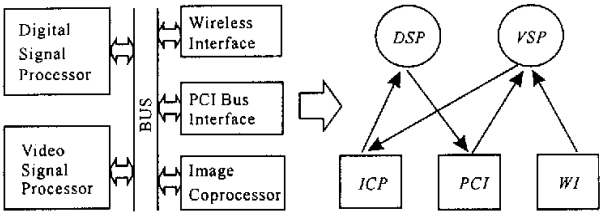

图4 2个进程和3个单位资源系统

表2 2个进程和3个单位资源系统的矩阵表示

	$q_1(ICP)$	$q_2(PCI)$	$q_3(WI)$
$p_1(DSP)$	g	r	0
$p_2(VSP)$	r	g	g

表3 2个进程和3个单位资源系统中删除1条分配边的矩阵表示

	$q_1(ICP)$	$q_2(PCI)$	$q_3(WI)$
$p_1(DSP)$	g	r	0
$p_2(VSP)$	r	0	g

4 结束语

基于邻接矩阵的死锁检测算法提出了矩阵表示方法,时间复杂度为O(min(m,n)),并且是基于并行技术的,比一些传统算法有了明显改进。运行时需要内存也比较小,系统能够很好地检测死锁的发生,并且释放占有资源。

参考文献

1 汤子瀛,哲凤屏,汤小丹.计算机操作系统.西安:西安电子科技大学出版社,2000.119~130.

2 Ju Gyun Kim. Algorithmic approach on deadlock detection for enhanced parallelism in multiprocessing systems. Aizu International Symposium on Parallel Algorithms Architecture Synthesis, IEEE, Piscataway, NJ(USA), 1997, 233~238.

3 朱丽莉,焦素云,周立娟.基于资源分配图的死锁检测算法的改进.情报科学,2000,5:13~16.

4 周 兵.对于避免死锁的安全算法的改进.微机发展,2001,3.

5 袁蒲佳,龙玉国,杨微徽.数据结构.武汉:华中理工大学出版社,1999.102~110.

(责任编辑:邓大玉)