Jupiter Publications Consortium

Vol. 1(2), pp. 91-95, 28 Sep, 2015
ISSN Print : 2454-1435 © 2015 IJRMMAE
ISSN Online : 2454-1443 © 2015 IJRMMAE
http://www.ijrmmae.in

International Journal of Research in Mechanical, Mechatronics and Automobile Engineering

A Review on the Implementation of Deadlock Prevention using Banker's Algorithm

C N Ravi, Dr. N Chinnaiyan
Associate Professor, Information Science and Engineering Department, New Horizon College of Engineering, Bangalore, India,
ravicn@newhorizonindia.edu

S.Rajeswari, Shwetha K S, Baswaraju Swathi
Assistant Professor, Information Science and Engineering Department, New Horizon College of Engineering, Bangalore, India,
rajeswaris@newhorizonindia.edu

Received 10, September 2015 | Accepted 21, September 2015

Abstract:— The deadlock avoidance procedure, Banker's algorithm, was developed for computer operating systems, an environment where very little information regarding the future resource requirements of executing processes is known. Also, information on the maximum resource claims for each of the regions can be extracted prior to process execution. By inserting operating system calls when entering a new region for each process at runtime, and applying the original banker's algorithm for deadlock avoidance, this method has the potential to achieve better resource utilization because information on the "localized approximate maximum claims" is used for testing system safety.

Keywords:—Banker's Algorithm, Deadlock, Deadlock avoidance, Workflow Scheduling.

I. INTRODUCTION

Deadlock-free operation is an important operational requirement in flexible manufacturing systems (FMSs). In general, deadlock is the situation in which there exists a set of concurrent processes with each process in the set awaiting an event that can be caused only by another process in the set (Silbershatz and Peterson, 1991). A ubiquitous problem in discrete event systems, deadlock results from various aspects of systems operations, such as resource allocation and communications (Holt, 1972)[1]. In an FMS, deadlock is caused by imprudent allocation of buffer space, tooling, and material handling equipment (Cho, Kumaran, and Wysk, 1995).

The FMS controller, therefore, must incorporate some strategy for handling deadlocks; otherwise continuing system operation cannot be guaranteed. The deadlock phenomenon has been studied extensively in computer operating systems[2]. In these systems, executing processes compete for computing resources such as I/O channels, disk space, and memory.

In contrast to computer operating systems, a part in an FMS visits a predictable sequence of machines for processing, the part route. At each machine, the part requires a certain set of resources (buffer space, cutting tools, etc.) to complete its processing before moving on to the next machine. The objective of this paper is to be focus on various resources used in bankers algorithms[3].

1. Detection and recovery. Every time a resource is requested or released, a check is made to see if any deadlocks exist. If exist, some policy will be adopted to recover the system from deadlock.

Jupiter Publications Consortium

Vol. 1(2), pp. 91-95, 28 Sep, 2015
ISSN Print : 2454-1435 © 2015 IJRMMAE
ISSN Online : 2454-1443 © 2015 IJRMMAE
http://www.ijrmmae.in

International Journal of Research in Mechanical, Mechatronics and Automobile Engineering

2. Deadlock prevention, by structurally negating one of the four required deadlock conditions. deadlocks will be impossible[4].

3. Dynamic avoidance by careful resource allocation: deadlock avoidance[5]. Make judicious choices to assure that the deadlock point is never reached.

2. Data Structures Available:

Vector of length m # instances of each resource type available in system If available [j] = k, there are k instances of resource type Rj available.

Max:

• n x m matrix.

• Maximum # of instances of each resource each process can request.

• If Max [i,j] = k, then process Pi may request at most k instances of resource type Rj.

Allocation:

• n x m matrix

• # instances of each resource type allocated to each process

• If Allocation[i,j] = k then Pi is currently allocated k instances of Rj..

Need:

1. n x m matrix

2. # instances of each resource type each process may need more of

3. If Need[i,j] = k, then Pi may need k more instances of Rj to complete its task

4. Need [i,j] = Max[i,j] – Allocation [i,j].

Safety Algorithm:

1. Let Work and Finish be vectors of length m and n, respectively. Initialize: Work = Available

2. Finish [i] = false for i = 0, 1,2, …, n-1.

3. Find an i such that both:

(a) Finish [i] = false

(b) Needi ≤ Work

If no such i exists, go to step 4.

3. Work = Work + Allocationi

Finish[i] = true go to step 2.

4. If Finish [i] == true for all i, then the system is in a safe state.

Vol. 1(2), pp. 91-95, 28 Sep, 2015
ISSN Print : 2454-1435 © 2015 IJRMMAE
ISSN Online : 2454-1443 © 2015 IJRMMAE
http://www.ijrmmae.in

International Journal of Research in Mechanical, Mechatronics and Automobile Engineering

3. Resource-Request Algorithm for Process Pi :

Requesti = request vector for process Pi.

If Requesti [j] = k then process Pi wants k instances of resource type Rj. When resource request made by Pi, the following occurs:

1. If Requesti ≤ Needi go to step 2. Otherwise, raise error condition, since process has exceeded its maximum claim.

2. If Requesti ≤ Available, go to step 3. Otherwise Pi must wait, since resources are not available.

3. Pretend to allocate requested resources to Pi by modifying the state as follows: Available = Available - Requesti;

Allocationi= Allocationi + Requesti;

Needi = Needi – Requesti;

If safe the resources are allocated to Pi.

If unsafe Pi must wait, and the hold resource allocation state is restored

4. EXAMPLE OF BANKER'S ALGORITHM:

5 processes P1 through P5 3 resource types:

A: 10 instances

B: 5 instances

C: 7 instances

Resource-allocation state at time T0:

	Allocation	Max	Available
	A B C	A B C	A B C
P0	0 1 0	7 5 3	3 3 2
P1	2 0 0	3 2 2	
P2	3 0 2	9 0 2	
P3	2 1 1	2 2 2	
P4	0 0 2	4 3 3	

The content of the matrix, Need is defined to be Max – Allocation.

Need
A B C
P0 7 4 3

Vol. 1(2), pp. 91-95, 28 Sep, 2015
ISSN Print : 2454-1435 © 2015 IJRMMAE
ISSN Online : 2454-1443 © 2015 IJRMMAE
http://www.ijrmmae.in

International Journal of Research in Mechanical, Mechatronics and Automobile Engineering

P1 1 2 2

P2 6 0 0

P3 0 1 1

P4 4 3 1

The system is in a safe state since the sequence < P1, P3, P0, P2, P4> satisfies safety criteria.

Example (cont.):

Now P1 requests 1 instance of A and 2 instances of C: Request1 (1, 0, 2)

1.) Check that Request1 ≤ Available (1,0,2) ≤ (3,3,2) _ true (can immediately grant request)

2.) Now see if system is in safe state:

	Allocation	**Need**	**Available**
	A B C	A B C	A B C
P0	0 1 0	7 4 3	2 3 0
P1	3 0 2	0 2 0	
P2	3 0 1	6 0 0	
P3	2 1 1	0 1 1	
P4	0 0 2	4 3 1	

Executing safety algorithm shows that sequence <P1, P3, P4, P0, P2> satisfies safety requirement.

5. CONCLUSION

In this paper, we proposed an extension of the banker's algorithm for deadlock avoidance. Assuming that the control flow of the resource-related calls of each process forms a rooted tree, we proposed a quadratic-time algorithm which decomposes these trees into regions and computes the associated maximum resource claims, the extended banker's algorithm has the potential to improve the resource utilization while incurring low runtime overhead.

6. REFERENCES

[1] The Application and Evaluation of Banker's Algorithm for Deadlock-Free Buffer Space Allocation in Flexible Manufacturing Systems. The International Journal of Flexible Manufacturing Systems, 10 (1998) Kluwer Academic Publishers, Boston.

[2] AnjuBala, Dr.Inderveer Chana, A Survey of Various Workflow Scheduling Algorithms in Cloud Environment.2nd National Conference on Information and Communication Technology (NCICT) 2011. Proceedings published in International Journal of Computer Applications® (IJCA)

Vol. 1(2), pp. 91-95, 28 Sep, 2015
ISSN Print : 2454-1435 © 2015 IJRMMAE
ISSN Online : 2454-1443 © 2015 IJRMMAE
http://www.ijrmmae.in

International Journal of Research in Mechanical, Mechatronics and Automobile Engineering

[3] Sheau-DongLang, An Extended Banker's Algorithm for Deadlock Avoidance. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 25, NO. 3, MAY/JUNE 1999.

[4] Algoritmo del bankero.pdf

[5] https://www.computer.org/csdl/trans/ts/1999/03/e0428.html