주문일정관리에서 교착상태 회피를 위한 은행원 알고리즘의 적용

오지강 ㅇ,이욱진

한양대학교 컴퓨터공학과

hhhwwwuuu@hanyang.ac.kr, scottlee@hanyang.ac.kr

The Application of Banker's Algorithm in Order Scheduling Management for Deadlock Avoidance

Zhiqiang Wu○, Scott Uk-Jin Lee

Department of Computer Science & Engineering, Hanyang University

Abstract

Order Scheduling in product supply chain is an important activity to adequately manage cash flow for supplier. However, in current order scheduling, deadlock may appear when managing multiple supply chain flows simultaneously. This paper proposes a deadlock avoidance method for order scheduling in product supply chain system by adopting Banker's algorithm which is a commonly used in operating systems for deadlock avoidance. After thorough analysis and comparisons, Banker's algorithm is utilized to manage order scheduling in product supply chain instead of managing memories in operating systems. As a result, order scheduling can be managed effectively with dramatically less chances of deadlock when handling multiple orders. To verify the proposed method, SPIN is used to simulate and verify the correctness.

1. INTRODUCTION

In operating systems, many concurrent processes are executed simultaneously where many scheduling issues such as deadlock, data race and starvation exist. In order to prevent a deadlock for resource allocation, Operating System (OS) provides adequate mechanism called Banker's algorithm [1]. The deadlock not only exists in OS, but also in Order Scheduling Management (OSM) of product supply chain. In OSM, a deadlock can frequently occur when managing supply and demand between supplier and retailers. Supplier needs to reasonably arrange sequence of order to satisfy retailers. Scheduling management for orders is critical in product supply chain to ensure every retailer to receive commodities from a common supplier. The order contracts between supplier and retailers can specify various payment options such as before or after the delivery of commodities or even several installments. With such a variation in money flow, a supplier has to manage not only the cost for manufacturing commodities but also other operational cost such as management fee, equipment maintenance and storage charge. These situations may cause supplier to reach insufficient money flow to produce commodities, resulting in a low productivity or even a state of financial deficit. Hence, the risk of capital chain rupture for supplier increases with the increase in number of orders or amounts of commodities. Without a careful scheduling of orders, resources allocation of supplier may result in a deadlock situation. Therefore, efficient algorithm is required to avoid such deadlock situations leading to capital chain rupture in OSM.

In order to avoid capital chain rupture in OSM and to effectively manage the money flow, it is necessary to adequately allocate requirements of commodities to ensure satisfaction of retailers. We propose an adoption of Banker's algorithm in OSM to avoid possible deadlock situations. After thorough analysis and comparisons, Banker's algorithm is utilized to manage order scheduling in OSM instead of managing memories in OS. Then, the proposed deadlock avoidance is verified with the finite state model checker called SPIN [2]. The model will be written in Promela language to verify the correctness of system.

The rest of the paper is organized as follows. Section 2 discusses Banker's algorithm for avoidance of a deadlock in OS. The implementation of Banker's algorithm in OSM and verification with SPIN are described in section 3. Finally, section 4 concludes the paper.

2. RELATED WORK

2.1 Deadlock avoidance with Banker's Algorithm

A deadlock is a situation where number of processes result in a circular waiting [1]. It occurs due to inadequate resource allocations. A deadlock will increase the cost of system resources and decrease their utilization rate. There are three existing methodologies to allocate resources in multiprocessing system for avoiding a deadlock; deadlock prevention, deadlock avoidance and deadlock detection [3]. Banker's Algorithm which is used in OS for deadlock avoidance was developed by Dijkstra [4], then extended to handle multiple resources types by Haberman[5]. Banker's algorithm specifies distinct two states which occurs during resource allocation. The definition of each state is shown as following:

- Safe state: contains at least one resource allocation sequence that can make all processes finish safely at the current state.
- Unsafe state: contains resource allocation sequences which always result in a deadlock at the current state.

Suppose a system with *n* processes *(P1, P2, …, Pn)* and *m* resources *(R1, R2,…, Rm)*.

- The matrix of the maximum demand is formulated as *Max(i,j)=k*. It defines that the process Pi needs *k-Rj* resources at maximum.
- The matrix of allocation is formulated as *Allocation(i,j)=k*. It defines that the process *Pi* has already acquired *k-Rj* resources.
- The matrix of need is formulated as *Need(i,j)=k*. It defines that the process *Pi* still needs $k-Rj$ resources to complete.
- The vector of availability is formulated as *Availability(i)=k*. It defines that *k-i* resources are available currently. *Availability'* defines a status that the process returns the all resource to system.
- The relationships between the above three matrices are formulated as below:

$$Need(i,j) = Max(i,k) - Allocation(i,j)$$

$$Availability'(1,...m)=Availability(1,...m)+Allocation(i,1,..m)$$

According to the algorithm of safe detection, system will successfully detect deadlock causing unsafe states. If the system is in safe state, the allocated resources will be executed. Otherwise, the process *Pi* will wait for the required resources.

2.2 Comparison with OSM and OS

In this section, we analyze and compare OSM with OS to successfully adopt Banker's algorithm since they both deals with processing multiple tasks concurrently.

Table 1, describes comparison between OS and OSM. In both OS and OSM, risk of deadlock exists when allocating resources. In OS, resources are adequately allocated to processes to avoid deadlock. But, in OSM, supplier arranges orders of retailers to avoid risk of capital chain rupture. In OS, system resources are allocated to each process and they are returned to system when process is completed. But, in OSM, commodities are allocated to retailers and retailers make payment to supplier when the requests are fully satisfied.

In these concurrent systems, two systems deals with difference entity types, but they have a common goal to avoid deadlocks from incorrect scheduling.

Comparison Criteria	OS	OSM
Object to be Scheduled	Processes	Orders
Deadlock Avoidance Policy	Banker's algorithm	Scheduling algorithm
Resources	Memory, CPU, etc.	Money, Commodities

Table. 1. Comparison between OS and OSM

3. ADOPTION OF BANKER'S ALGORITHM IN OSM

3.1 Modeling OSM

In order to adopt Banker's algorithm, OSM is modeled accordingly. A single order in OSM is modeled as a process and multiple commodities is modeled as resources that retailers need. If scheduling in OSM is inadequate, it will cause insufficient money flow of supplier and create a risk of capital chain rupture. We such a deadlock, supplier cannot produce commodities for current order and retailers enters an unsafe state of indefinite waiting.

Since original resources (commodities) are not return but payment is made from retailers to supplier in OSM, safe state is defined as a state with sufficient money flow for the next production. A safety allocation sequence in OSM can be acquired by Banker's algorithm where the money flow should remain in a safe state.

3.2 Implementation of Banker's Algorithm

The ideal approach is simply to ensure that the

supplier acquires sufficient money flow by always remaining in a safe state. Safe state in OSM enable supplier to avoid deadlock situation by acquiring sufficient money flow. Resource recycling process in OS is applicable in OSM where the transactions between retailers and supplier take similar roles. For this reason, the Banker's algorithm from OS can be directly implemented in OSM. An implementation of Banker's algorithm would require $O(m)$ time and $O(n+m)$ space to process a request or release, where m is the number of resource types [6].

Suppose that $Request_i(1,...,m)$ is vector of requested resources for the process Pi. When retailers make a request to supplier, the system will detect the state of order scheduling management as below:

1) If $Request_i(1,...,m) \geqq Need(i,1,...,m)$, it is illegal request since requested commodities exceeds maximum quantity of commodities in the contract. Otherwise, move forward to the next step.
2) If $Request_i(1,...,m) \geqq Availability(1,...,m)$, it represents that the supplier cannot satisfy the current request of retailer and causes deadlock. Otherwise, move forward to the next step.
3) Supplier tries to allocate commodities for requests of retailers. Then, the relationship should be updated:

$$Availability'(x) = Availability'(x) - Request_i(x)$$
$$Allocation'(i,x) = Allocation(i,x) + Request_i(x)$$
$$Need'(i,x) = Need(i,x) - Request_i(x)$$

Through the above procedural method of deadlock detection for OSM, the supplier is able to safely perform current state order scheduling.

3.3 Verification with SPIN

The safety sequences of order scheduling for OSM can be obtained by Banker's algorithm. In order to verify the correctness of the safety sequences SPIN model checker is used. SPIN is an efficient verification tool for multi-threaded software system [7]. To verify the correctness of the proposed methodology, it is written in Promela language. Safe detection method of Banker's algorithm is used to simulate this model for current request of orders. The pseudo code is a detection algorithm for detecting safe state using in SPIN.

When the model above is simulated with SPIN, no errors were resulted indicating the correctness of the proposed safety sequences. Hence, it proves that the correct safety sequences can be obtained in OSM by adopting Banker's algorithm.

Detection of safe state

```
Initial: work[m]    //available commodities, m is number of types
         finish[n]= false   //the status of finished orders
         num = 0 // counter
1.    work[1 to m] equals currently available commodities
2.    for i∈n do
3.      for j∈m do
4.        if Need[i,j]≦work[j] and !finish[i] then
5.           num++
6.        end
7.        if num == m then
8.           work[j] <- work[j] + Allocation[i,j]
9.           finish[i] <- true
10.       end
11.     end
12.   end
13.   if finish[1,...,n] == TRUE then Safe state;
       else Unsafe state
```

4. CONCLUSION

In this paper, an effective deadlock avoidance methodology for OSM is proposed by adopting Banker's algorithm from OS. Initially, the differences between OS and OSM are analysis and compare. Then OSM is modeled adequately and Banker's algorithm is adopted for order scheduling. Finally, the correctness of the proposed methodology is verified using SPIN. With the proposed methodology, suppliers can better manage orders and prevent capital chain ruptures by avoiding deadlock in order scheduling. It also enables suppliers to increase their money flow.

REFERENCES

[1] X. Tang, "Computer Operating System," Xidian University, Xi'an, Shannxi, China, 2014.

[2] G. J. Holzmann, "The SPIN model checker: Primer and reference manual," Addison Wesley, Boston, MA, USA, 2004.

[3] G. Hou, "Deep analysis of banker algorithm," Journal of Weifang University, vol. 6, pp. 46-48, March 2006.

[4] E. W. Dijkstra, "Cooperating sequential processes," Technical Report EWD-123, Technological University, Eindhoven, The Netherlands, 1965.

[5] A. N. Haberman, "Prevention of system deadlocks," Communication of the ACM, vol. 12, No. 7, pp. 373-385, July 1969.

[6] F. Belik, "Deadlock avoidance with a modified banker's algorithm," BIT Numerical Mathematics, vol. 27, No. 3, pp. 290-305, 1987.

[7] H. Lin, W. Zhang, "Model checking: Theories, Techniques and Applications,"Acta Electronica Sinica, vol. 30, No. 12, pp. 1907-1912, December, 2002.