Application Prospect of Deadlock Detection and Removal Technology in Modern Operating System

Shukun Wu[a], Zhongdong Wang[b,*], Zihan Yu[c], Xiangxin Li[d], Jianhao Lin[e], Zhensong Chen[f]

School of Mathematics & Computer Science, Guangxi Science & Technology Normal University, Laibin, China
[a]wuskctrl@outlook.com,[b]1647402833@qq.com,[c]yuzihaniot@outlook.com, [d]lixiangxincxk@outlook.com, [e]19907805060@163.com, [f]czs13198200904919@outlook.com
**Corresponding author*

Abstract: *Deadlock refers to the phenomenon that the system cannot work normally due to the interaction of more than two processes. In the modern operating system, the complexity, application difficulty and the range of participants of the system are getting higher and higher, so the detection and removal technology of deadlock is becoming more and more important. This paper reviews the basic technology, application status and shortcomings of deadlock detection and removal, and makes a preliminary prediction for the future development.*

Keywords: *Deadlock, Detection, Cancellation, Overview, Expectation*

1. Introduction

A deadlock is a situation where, between multiple processes (or threads), each process waits for another process to release resources and cause all processes to proceed. This often happens when multiple processes compete with each other for limited resources. In the deadlock state, no one process can continue execution, because each process is waiting for resources occupied by other processes that cannot be released. The causes of deadlock are shown in Fig. 1.

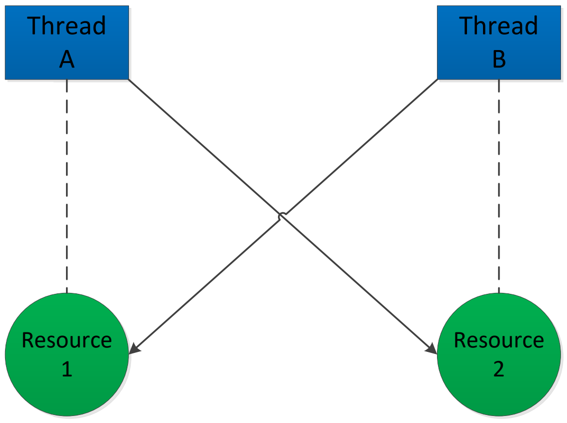

Fig. 1. Schematic diagram of deadlock causes

2. Deadlock detection and remove the latest development at home and abroad

The often used method of deadlock detection and removal is the banker algorithm. Since this algorithm has been proposed, many people have studied it and improved it.

Chavan and other researchers have proposed a new banker algorithm, the greedy algorithm, which can effectively monitor all the data and information and optimize the overall operational efficiency based on the increasing amount of data and information. With the map, we can determine a safe state at a moment. We will decide according to all available resources, and we will only continue if these available resources reach the first moment, otherwise we will terminate this inspection. When every step in the map can obtain the necessary resources, it means that the system has reached the security standard and can be used normally; however, if a step is found unable to obtain the necessary resources, then the system has lost the security and therefore cannot accept the use of such resources.

The security algorithm of the Banker's algorithm has been greatly improved by Zuo Wanli and his team. These improvements include: first, when the Finish vector is initialized as True, it is considered to have enough information, and second, when a process is seeking the necessary resources that are below the maximum allowable value of the system, it fails to continue running.

Zuo wanli uses the optimized security check algorithm to greatly reduce the unnecessary inspection, thus greatly improving the efficiency of the banker algorithm. In addition, Kshipra et al. also developed a dynamic banker algorithm, which can adjust the use of resources at any time according to the actual situation, thus effectively preventing the system from being in an unsafe state. The dynamic banker algorithm can quickly obtain the information of various resources in the system without any initialization, thus realizing automatic scheduling.

After further research, the improved algorithm of Chavan still needs to be improved. Among them, it is not reasonable to only use the amount of resource consumption as the ranking standard, resulting in the inability to obtain accurate information. To this end, Chavan's Sub-I adopts a new technology jointly created by Zuo Wanli and Kshipra, which greatly reduces the cycle of security detection and greatly improves the implementation efficiency of bankers' algorithm.

One big change: the Banker algorithm's system starts to have zero resources, but it can be adjusted to the user's request. For example, the user can decide whether to use the algorithm by detecting the system's resources. In this way, the running speed of the algorithm can be adjusted according to the user's request, thus improving the efficiency of the algorithm.

When Need [i] <=Available, process P is free to allocate resources to meet its needs. By introducing P, we can greatly improve the security of the algorithm. When Finish [i] =True is initialized as True, we can ensure that process P does not have any hazard information, thus avoiding interference from security checks.[1]

There are many similarities between banker algorithm and deadlock detection algorithm of computer operating system. The comparison diagram of banker algorithm and operating system is shown in Fig. 2.

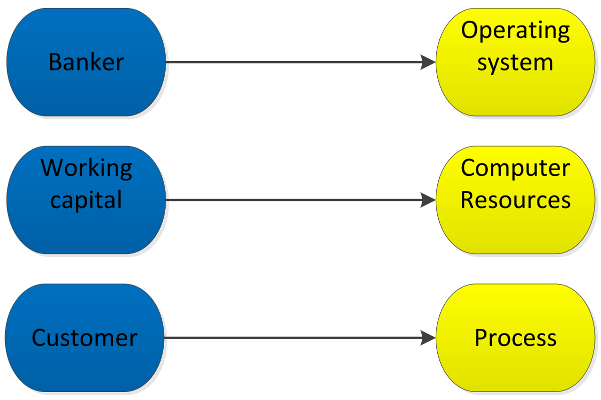

Fig. 2. Comparison diagram of banker algorithm and operating system

3. Deadlock detection and location of detection and release in the operating system

3.1. The role of deadlock detection and release of the deadlock in the operating system

Deadlock is a system problem caused by competitive behavior between multiple processes that makes these processes unable to continue, thus creating a stalemate that continues even without external interference, thus bringing the entire system to a standstill. The purpose of detection and release is to avoid the occurrence of deadlock, so as to release the system resources, reasonably allocate resources, so that the process or thread can run normally. In the process of execution, two or more threads wait for each other due to the competition for resources. The two threads can not advance without external force.[2]

3.2. The position of deadlock detection and release of the deadlock in the operating system

The detection and release of deadlocks is usually a part of the operating system kernel, which is the core module in the operating system used to handle resource allocation and process scheduling. Once the deadlock is removed, the system can return to normal operation. However, these methods usually bring some system overhead and performance loss, so the system should try to prevent the occurrence of deadlock, and avoid frequent deadlock detection and release. In the operating system, the detection and release of deadlock is usually handled by the process scheduler or resource manager. When a process requests a resource, the scheduler or resource manager checks for potential deadlock situations and takes action to avoid or remove them. The detection of deadlock is usually based on some algorithms, such as waiting graph algorithm or resource allocation graph algorithm. By analyzing the resource dependencies between different processes and the waiting state, these algorithms can effectively detect whether there is a cyclic waiting, and thus determine whether the deadlock problem occurs. When the system finds a deadlock, it can solve problems in many ways, such as using resources, restarting operations, or terminating the process. The specific release strategy depends on the design and application scenario of the operating system. The detection and release of deadlock is a very important part of the operating system, which can ensure the stability and reliability of the system, and avoid the process from falling into the deadlock state due to resource competition.[3]

4. The specific workflow of deadlock detection and removal

4.1. Understand the conditions of deadlock before deadlock detection and releas

1) Mutual exclusion condition

The exclusive use of allocated resources, that is, a resource is occupied by only one process during a period of time. If other processes request the resource, the requester can only wait until the process that owns the resource is released.

2) Request and Hold conditions

It means that a process has maintained at least one resource but has made a new resource request that has been occupied by other processes, when the request process is blocked but keeps the other resources it has acquired.

3) Non-deprivation condition

It refers to the resources obtained by the process. Before they are used up, they cannot be deprived and can only be released by themselves when they are used up.

4) Ring road waiting condition

When a deadlock occurs, there must be a process.

The specific conditions under which a deadlock occurs are shown in Fig. 3.

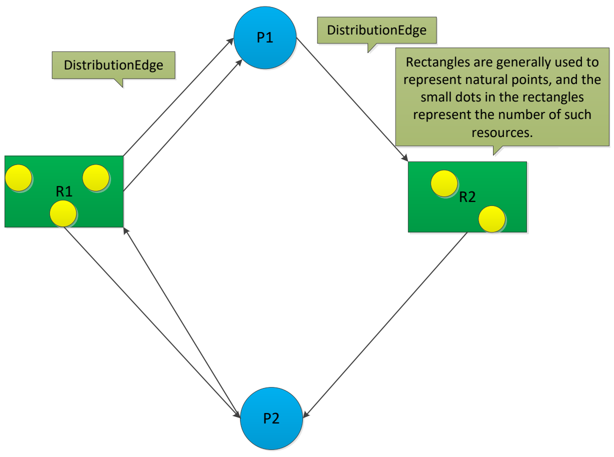
DistributionEdge
P1
DistributionEdge
Rectangles are generally used to represent natural points, and the small dots in the rectangles represent the number of such resources.
R1
R2
P2

Fig. 3. Schematic diagram of conditions for deadlock occurrence

4.2. Workflow for the detection and removal of deadlocks

Build a resource distribution graph to detect a deadlock, and a deadlock may exist if there is a loop, i. e., a loop waiting situation, in the resource allocation graph. By looking at the resource distribution map, we found some unrestricted, unhindered processes P. If we can handle various tasks normally, then it can easily meet our needs, and use the remaining resources to solve the problem. In turn, if we are not able to handle the task correctly, we have to remove P from the resource map and make it a separate node. Delete the two allocation edges and one request edge for the process P.

By repeating this process, the process P continuously releases its resources so that it can continue to operate until its resources are exhausted.

Through a series of simplified operations, you can delete all the edges in the graph, so that each process becomes a separate vertex, thus achieving the goal of removing the deadlock.[4]

The resource distribution diagram for the deadlock detection build is shown in Fig. 4.

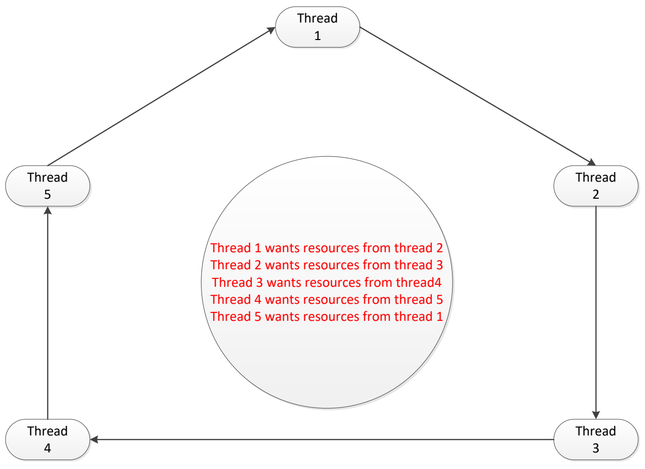
Thread 1
Thread 2
Thread 3
Thread 4
Thread 5
Thread 1 wants resources from thread 2
Thread 2 wants resources from thread 3
Thread 3 wants resources from thread4
Thread 4 wants resources from thread 5
Thread 5 wants resources from thread 1

Fig. 4. Resource distribution diagram Resource distribution diagram

5. The deficiency of the detection and removal of the deadlock

5.1. The cost of deadlock detection

The deadlock detection needs to regularly check whether there is a deadlock in the system. This increases the system overhead and may cause lower performance.

5.2. Unable to prevent the deadlock

The deadlock detection can only be tested and removed after the deadlock has occurred, and cannot prevent the occurrence of the deadlock. This means that in some cases, the system may be affected before the deadlock.

5.3. The complexity of unlocking deadlocks

Deadlocks need to be careful to ensure that it does not lead to abnormal behavior or data loss in other processes. If the deadlock removal method is not correct, it may lead to more serious problems.

5.4. Resource waste

When the deadlock occurs, the process in the system may be blocked and cannot continue to execute. This leads to a waste of resources because they cannot be used by other processes.

5.5. It is difficult to determine the optimal release strategy

The deadlock release strategy needs to be selected according to the specific situation, and may need to consider the factors such as process priority and resource requirements. Determining the optimal unwinding strategy may not be easy and may require complex analysis and decision making.

Although the detection and release method of deadlock can help solve the problem of deadlock, they also have some shortcomings. In order to better handle the deadlock problem, the operating system usually uses a combination of multiple deadlock prevention and avoidance strategies to reduce the possibility of deadlock occurrence.[5]

6. Improvement suggestions and development prospects for the detection and removal of deadlock

Advanced intelligent algorithms, such as reinforcement learning, can effectively detect and solve the deadlock problem. In addition, you can also adjust the resource allocation strategy in real time to effectively reduce the chance of deadlock occurrence.

By adopting parallel computing technology, we can greatly improve the deadlock detection algorithm, thus greatly improving its efficiency and accuracy.

By applying machine learning technology to deadlock detection and release, the performance and reliability of the system can be greatly improved. For example, deep learning algorithms can be used to make accurate predictions of areas where deadlock may occur.

Special attention should be paid to the safety of the system when conducting deadlock detection and lifting operation. In order to avoid the release of sensitive resources or data leakage, the relevant safety regulations should be strictly followed to ensure the safety of the system.

With the rapid development of distributed system, the deadlock problem is becoming increasingly complicated. Future research will focus on how to effectively detect and resolve deadlock to provide more possibilities for distributed systems. The cost of deadlock detection: the deadlock detection needs to regularly check whether there is a deadlock in the system. This increases the system overhead and may cause lower performance.

In real-time systems, in order to meet the strict requirements of response time, we must develop an algorithm that can quickly and accurately detect the deadlock. The detection and removal of deadlock is a challenging and opportunistic research field, and future development will require more advanced intelligent technologies to achieve this goal

7. Conclusion

This paper aims to explore the detection and release method of deadlock, and analyze it from different angles to obtain better results. After systematic research, we find that the current deadlock detection and removal technology has made significant progress, but it also faces many challenges and problems. Although existing deadlock detection algorithms can effectively detect anomalies in large-scale systems, it is still necessary to deeply investigate more effective algorithms and data structures and how to optimize deadlock strategies to achieve optimal system performance and resource utilization. With the continuous progress of distributed systems and cloud computing technology, the deadlock problem has become more and more complex, so effective measures must be taken to deeply explore and solve them.

With the development of technology, traditional deadlock detection methods, such as wait graph and resource allocation graph, are simple and intuitive, but they are not efficient in large-scale systems. In order to improve the efficiency of detection, some optimization techniques based on algorithm and data structure, such as hash table detection and matrix detection, have become effective ways to solve this problem. Machine learning technology has been widely used in deadlock detection, which can help the system to learn and predict the behavior, and thus improve the accuracy and efficiency of detection.

With the development of technology, some common deadlocking strategies, such as resource preemption, rollback operation and resource allocation adjustment, can effectively solve problems, but they may also bring about system performance reduction or waste of resources. Therefore, future research should focus on how to achieve deadlocks more effectively while maintaining system performance.

Future research will focus on the combination of machine learning and artificial intelligence technology to develop more intelligent and adaptive deadlock detection and release technologies; moreover, it will study deadlock problems in distributed systems and propose effective detection and removal strategies; finally, effective preventive precautions will be explored to effectively reduce the occurrence of deadlock by optimizing system design and resource allocation.

Deadlock detection and removal technology is playing an increasingly important role in today's society. Although some achievements have been made, there are still many challenges to be solved. Therefore, future research must be constantly explored and innovated to improve the reliability and efficiency of the system.

Acknowledgment

This project is supported by the "Youth Research Project of Guangxi Science & Technology Normal University (GXKS2022QN020)" in "Research Fund Project of Guangxi Science & Technology Normal University"

References

[1] Song Dan, Li Yadong. Improvement and implementation of banker algorithm [J]. Computer Age, 2021 (06): 64-67
[2] Fan Chengya. Detection and release of deadlocks in the operating system [J]. Journal of Zhengzhou Institute of Light Industry, 1993,8 (3): 53-55.
[3] Dong Jian, Yang Xiaozong, Cheng Xin, et al. fault-tolerant general deadlock model detection algorithm for distributed system [J]. Computer Research and Development, 2007,44 (5): 798-805.
[4] Li Tao, Zhao Hongsheng. Pway optimization of mobile robots based on the evolutionary ant colony algorithm [J]. Control and Decision-making, 2023,38 (3): 612-620.
[5] Xu Jihua, Zhu Xiaojuan. A collaborative decision algorithm based on multi-agent reinforcement learning [Z]. Journal of Ningxia Normal University, 2023,44 (4): 71-79.