International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 5 Issue: 11 157 – 163

Implementation of Banker's Algorithm Using Dynamic Modified Approach

Mrinal Gaur
M.Tech Scholar,
CGI Bharatpur
E-Mail: iammrinnal@gmail.com

Dushyant Singh
Assistant Professor,
CGI Bharatpur

Abstract: Banker's algorithm referred to as resource allocation and deadlock avoidance algorithm that checks for the safety by simulating the allocation of predetermined maximum possible of resources and makes the system into s-state by checking the possible deadlock conditions for all other pending processes.

It needs to know how much of each resource a process could possibly request. Number of processes are static in algorithm, but in most of system processes varies dynamically and no additional process will be started while it is in execution. The number of resources are not allow to go down while it is in execution.

In this research an approach for Dynamic Banker's algorithm is proposed which allows the number of resources to be changed at runtime that prevents the system to fall in unsafe state. It also give details about all the resources and processes that which one require resources and in what quantity. This also allocates the resource automatically to the stopped process for the execution and will always give the appropriate safe sequence for the given processes.

*****

1. Introduction

1.1 Deadlock

A set of processes is deadlocked if each process in the set is waiting for an event that only another process in the set can cause. Usually the event is release of a currently held resource. None of the processes can run and release resources.

In an operating system, a deadlock occurs when a process or thread enters a waiting state because a requested system resource is held by another waiting process, which in turn is waiting for another resource held by another waiting process. If a process is unable to change its state indefinitely because the resources requested by it are being used by another waiting process, then the system is said to be in a deadlock [1].

1.1.1 Conditions for Deadlock

Deadlock can arise if four conditions hold simultaneously

(i) **Mutual exclusion condition:** Only one process at a time can use a resource (non-shareable resource). Each resource is assigned to a process or is available.

(ii) **Hold and wait condition:** A process holding at least one resource can request for additional resources.

(iii) **No preemption condition:** A resource can be released only voluntarily by the process holding it. That is previously granted resources cannot be forcibly taken away.

(iv) **Circular wait condition:** There exists a set {P0,P1,…,P0} of waiting processes such that P0 is waiting for a resource that is held by P1, P1 is waiting for a resource that is held by P2,…,Pn–1 is waiting for a resource that is held by Pn, and P0 is waiting for a resource that is held by P0.

1.2 Methods for Handling Deadlocks

Generally speaking there are three ways of handling deadlocks:

1. Deadlock prevention or avoidance - Do not allow the system to get into a deadlocked state.
2. Deadlock detection and recovery - Abort a process or preempt some resources when deadlocks are detected.

Ignore the problem all together - If deadlocks only occur once a year or so, it may be better to simply let them happen and reboot as necessary than to incur the constant overhead and system performance penalties associated with deadlock prevention or detection. This is the approach that both Windows and UNIX take.

1.3 Banker's Algorithm

For resource categories that contain more than one instance the resource-allocation graph method does not work, and more complex (and less efficient) methods must be chosen. The Banker's Algorithm gets its name because it is a method that bankers could use to assure that when they lend out resources they will still be able to satisfy all their clients. A banker won't loan out a little money to start building a house unless they are assured that they will later be able to loan out the rest of the money to finish the house.

When a process starts up, it must state in advance the maximum allocation of resources it may request, up to the amount available on the system.

When a request is made, the scheduler determines whether granting the request would leave the system in a safe state.

 ISSN: 2321-8169
157 – 163

If not, then the process must wait until the request can be granted safely.

The banker's algorithm relies on several key data structures: (where n is the number of processes and m is the number of resource categories.)

- Available[m] indicates how many resources are currently available of each type.
- Max[n] [m] indicates the maximum demand of each process of each resource.
- Allocation[n] [m] indicates the number of each resource category allocated to each process.
- Need[n] [m] indicates the remaining resources needed of each type for each process. Note that Need[i][j] = Max[i][j] - Allocation[i][j] for all i, j.)

For simplification of discussions, we make the following notations / observations:

- One row of the Need vector, Need[i], can be treated as a vector corresponding to the needs of process i, and similarly for Allocation and Max.

A vector X is considered to be <= a vector Y if X[i] <= Y[i] for all i.

1.4 **An Illustrative Example**

- Consider the following situation:

	Allocation	Max	Available	Need
	A B C	A B C	A B C	A B C
P_0	0 1 0	7 5 3	3 3 2	7 4 3
P_1	2 0 0	3 2 2		1 2 2
P_2	3 0 2	9 0 2		6 0 0
P_3	2 1 1	2 2 2		0 1 1
P_4	0 0 2	4 3 3		4 3 1

Figure 1.1: Process, Allocated Resources, Max Resources and Needed Resources

- And now consider what happens if process P1 requests 1 instance of A and 2 instances of C. (Request[1] = (1, 0, 2)

	Allocation	Need	Available
	A B C	A B C	A B C
P_0	0 1 0	7 4 3	2 3 0
P_1	3 0 2	0 2 0	
P_2	3 0 2	6 0 0	
P_3	2 1 1	0 1 1	
P_4	0 0 2	4 3 1	

Figure 1.2: Process, Allocated Resources, Needed Resources and Available Resource

What about requests of (3, 3, 0) by P4? Or (0, 2, 0) by P0? Can these be safely granted? Why or why not?

2. Literature Review

In 2014, Pankaj Kawadkar, Shiv Prasad, Amiya Dhar Dwivedi in their research "Deadlock Avoidance based on Banker's Algorithm for Waiting State Processes" proposed an algorithm for deadlock avoidance used for Waiting State processes.

They proposed that if process is going to waiting state then the consideration of number of allocated resources and/or number of instances as well as need of resources in order to select a waiting process for the execution will make Banker's Algorithm more efficient. But they didn't give any solution when the system is in unsafe state [15].

In 2013, Smriti Agrawal, Madhavi Devi Botlagunta and Chennupalli Srinivasulu in their research titled "A Total Need based Resource Reservation Technique for Effective Resource Management" and proposed an approach for Total Need Based Resource Reservation (TNRR) that suggests reserving some resources so as to ensure that at least one process will complete after it. The simulation results indicate that the frequency of deadlocks has reduced by approximately 75% for higher load (above 80%) as compared to the Deadlock Recovery technique, while for lower load it tends to be zero. The turnaround time of the TNRR is approximately 9% better than the existing Banker's algorithm. But in case of insufficient resources when there is no safe sequence is possible they didn't provide details for resources and processes that causes the deadlock if executed or no safe sequence [4].

In 1999, Sheau-Dong Lang in his research titled "An Extended Banker's Algorithm for Deadlock Avoidance" proposed an approach for safety in banker's algorithm assuming that the control flow of the resource-related calls of each process forms a rooted tree, they
proposed a quadratic-time algorithm which decomposes these trees into regions and computes the associated maximum resource claims, prior to process execution. This information is then used at runtime to test the system safety using the original banker's algorithm. But this approach was unable to recognize patterns of resource-related calls in real-time system and the practicality was also not proven [16].

3. Motivation

Many research has been done for the improvement of Banker's Algorithm. Most of the researchers has worked on the limitations of waiting time and resource allocation to improve the performance or minimizing the deadlocks. But if at the end when system is not in safe state and traditional Banker's algorithm cannot be applied then what? We do not have any information about the process or resources due to which the system was in unsafe state. This specific problem leads us towards this approach. Proposed approach gives the details about all the resources and processes who require resources in what quantity. This also allocates the resource automatically to the stopped process for the execution and

will always give the appropriate safe sequence for the given processes.

4. Proposed Approach

Banker's algorithm was originally designed to check whether the allocation of resources leave the system in safe state or not and if it is in safe state then it gives the safe sequence of processes and allocate the resources.

In banker's algorithm when, a new process enters the system, it must declare the maximum number of instances of each resource type that it may need. This number may not exceed the total number of resources in the system.

When a user requests a set of resources, the system must determine whether the allocation of these resources will leave the system in a safe state. If it will, the resources are allocated; otherwise, the process must wait until some other process releases enough resources.

Steps for Banker's Algorithm

1. Let Request[n][m] indicate the number of resources of each type currently requested by processes. If Request[i] > Need[i] for any process i, raise an error condition.
2. If Request[i] > Available for any process i, then that process must wait for resources to become available. Otherwise the process can continue to step 3.
3. Check to see if the request can be granted safely, by pretending it has been granted and then seeing if the resulting state is safe. If so, grant the request, and if not, then the process must wait until its request can be granted safely. The procedure for granting a request (or pretending to for testing purposes) is:
   a) Available = Available – Request
   b) Allocation = Allocation + Request
   c) Need = Need - Request

Several data structures must be maintained to implement the banker's algorithm. These data structures encode the state of the resource-allocation system.

In this research an approach for Dynamic Banker's algorithm is proposed which allows the number of resources to be changed at runtime that prevents the system to fall in unsafe state.

Proposed Algorithm

Proposed algorithm that provides a safe sequence. It depends upon need (i)=1,2…n,

Allocation (i) =1, 2…n and Available resources.

It works as follows:

After the Resource-Request-Algorithm if process is going to waiting state then these steps must be followed:

Step.1- Need (i) is compare to Need(i+1 to last).

Step.2- Take the process with minimum Need and maximum Allocation.

Step.3- If Available<Need (i) then Execute process. Set state is executed.

Step.4-Available=Available + Allocation (i)

Step.5- Repeat step 1 to 4 until all the process is going to executed state.

Whenever a process requests for instance of resource it checks the availability and demand of instance of resource.

If availability is more than demand then instance of resource is allocated to process according to the priority of which process free the most number of resources.

It checks for the needed availability to execute the process.

Return the information about the instance of resource needed to system along with associated process.

5. Performing Banker's Algorithm with Existing Approach

Here in the first experiment of this research, implementing the original Banker's Algorithm having 3 types of different resources and 5 processes. Here some resources are already allocated to the processes and there is also given the maximum demand and remaining needs of these processes. After implementing the Banker's Algorithm results are evaluated to determine the safe sequence.

Here in this experiment there are five processes (p0 to p4) are used and three types of resources are used with different number of instances as A-10, B-5 and C-7.

TABLE 5.1

AVAILABLE RESOURCES

Available		
A	B	C
10	5	7

Table 5.1 shows the number of resources available for the process allocations. These are having several instances of same type.

TABLE 5.2

RESOURCES ALLOCATIONS

	Allocations		
	A	B	C
p0	0	1	0
p1	2	0	0
p2	3	0	2
p3	2	1	1
p4	0	0	2

Table 5.2 displays the resources allocated to the number of processes with different types of instances. These resources are already allocated to the processes and processes are executing on these resources.

TABLE 5.3
MAX DEMAND

	Available		
	A	B	C
p0	7	5	3
p0	3	2	2
p2	9	0	2
p3	2	2	2
p4	11	3	3

Table 5.3 shows the max demand of the processes for the resources. It also shows that which resource is having in what number of instances of which type of resource. Max demand also containing the allocated resources so here it is must to calculate the remaining needs of the process for the execution of the Banker's Algorithm.

TABLE 5.4
REMAINING NEED

	Remaining Need		
	A	B	C
p0	7	4	3
p1	1	2	2
p2	6	0	0
p3	0	1	1
p4	11	3	1

Table 5.4 shows the remaining needs of the processes for the resources. It also shows that which resource is having in what number of instances of which type of resource. Remaining need is calculating through the max demand and allocation of the resources. The formula which is used to calculate the need is given below.

Need = maximum resources - currently allocated resources

Processes (possibly needed resources):

After calculating the need for the processes then need to check the execution of the processes that these are in safe state or not.

Now to check whether above state is safe sequence in which above requests can be fulfilled. After evaluating the Banker's Algorithm shows that it is not in the safe state because any of the process needs for more resources which are not available to fulfill the requirements. Here Banker's Algorithm does not show that particular process which needs more resources of what type. It just shows that it is not in the safe sequence, so it is very difficult to say that which resource is needed to which process so the problem can be solved.

A state is considered unsafe if it is not possible for all processes to completing executing. Since the system does not know when a process will finish termination, or how many resources it requested by then, the system assumes that all processes will not able to acquire their stated maximum resources and terminated. This is a reasonable assumption in most of the cases the system is not particularly concerned with how long each process runs. Also, if a process terminates without having its maximum resources it only makes it simpler on the system. A safe state is considered as to be the decision maker if it is going to process ready queue. Any state where such set exists is a safe state.

5.1 Performing Modified Banker's Algorithm with New Approach

Here in the second experiment of this research, implementing the modified Banker's Algorithm having 3 types of different resources and 5 processes. Here some resources are already allocated to the processes and there is also given the maximum demand and remaining needs of these processes. After implementing the modified Banker's Algorithm results are evaluated to determine the safe sequence.

Here in this experiment there are five processes (p0 to p4) are used and three types of resources are used with different number of instances as A-10, B-5 and C-7.

TABLE 5.5
AVAILABLE RESOURCES

Available		
A	B	C
10	5	7

Table 5.5 shows the number of resources available for the process allocations. These are having several instances of same type.

Available		
A	B	C
10	5	7

TABLE 5.6
RESOURCES ALLOCATIONS

Table 5.6 displays the resources allocated to the number of processes with different types of instances. These resources are already allocated to the processes and processes are executing on these resources.

	Allocation		
	A	B	C
p0	0	1	0
p1	2	0	0
p2	3	0	2
p3	2	1	1
p4	0	0	2

TABLE 5.7

MAX DEMAND

Max Demand			
	A	B	C
p0	7	5	3
p1	3	2	2
p2	9	0	2
p3	2	2	2
p4	11	3	3

Table 5.7 shows the max demand of the processes for the resources. It also shows that which resource is having in what number of instances of which type of resource. Max demand also containing the allocated resources so here it is must to calculate the remaining needs of the process for the execution of the Banker's Algorithm.

TABLE 5.8

REMAINING NEED

Remaining Need			
	A	B	C
p0	7	4	3
p1	1	2	2
p2	6	0	0
p3	0	1	1
p4	11	3	1

Table 5.8 shows the remaining needs of the processes for the resources. It also shows that which resource is having in what number of instances of which type of resource. Remaining need is calculating through the max demand and allocation of the resources. The formula which is used to calculate the need is given below.

Need = maximum resources - currently allocated resources

Processes (possibly needed resources):

After calculating the need for the processes then need to check the execution of the processes that these are in safe state or not.

Now to check whether above state is safe sequence in which above requests can be fulfilled. After evaluating the Modified Banker's Algorithm shows that which process needs what instance of resource in what quantity? After knowing that which instance of resources need to be added more, then it is very easy to add the more resources for the process. Here modified Banker's Algorithm shows that particular process which needs more required resources of what type. It also shows that it is not in the safe sequence, so it is very easy to find out & add which resource is required to the process so the problem can be solved by this approach.

TABLE 5.9

REQUIRE RESOURCES

Require Resources			
	A	B	C
p0	4	1	1

Table 5.9 displays the require resources of the process. It shows after calculating that which resources is not available for the process. Then it is easy to add more resources further and then calculate the new safe sequence. It ensure that it will always give the safe sequence which never lead to deadlock.

After adding more resources in the system then calculate the safe sequence as follow P0, p1, p2, p3, p4.

Finally evaluating the results of both experiments it is very clear to say that original Banker's algorithm gives only assumption that the execution of the processes are in safe state or not in safe state.

Banker's Algorithm does not show that particular process which needs more resources of what kind of type. It just shows that it is not in the safe sequence, so it is very difficult to say that which resource is needed to which process so the problem can be solved.

Modified Banker's Algorithm shows that which process needs what instance of resource in what quantity? After knowing that which instance of resources need to be added more, then it is very easy to add the more resources for the process. Here modified Banker's Algorithm shows that particular process which needs multiple required resources of what type. It also shows that it is not in the safe sequence, so it is very easy to add which resource is needed to the process so the problem can be solved by this approach.

6. Results

Performance is calculated through executing several examples which prove that new approach of Banker's Algorithm is capable to solve more number of problems than existing approach of Banker's Algorithm.

Here is result comparison of both algorithms.

TABLE 6.1

PERFORMANCE ANALYSIS

No. of Experiments	**Banker's Algorithm**	**New Approach**
20	7	16

Table 6.1 displayed the total 20 experiments were performed with different problems (set of process and resources) out of which Banker's algorithm was able to solve 6 problems, on the other hand improved banker's algorithm was able to solve 16 out of the 20 problems.

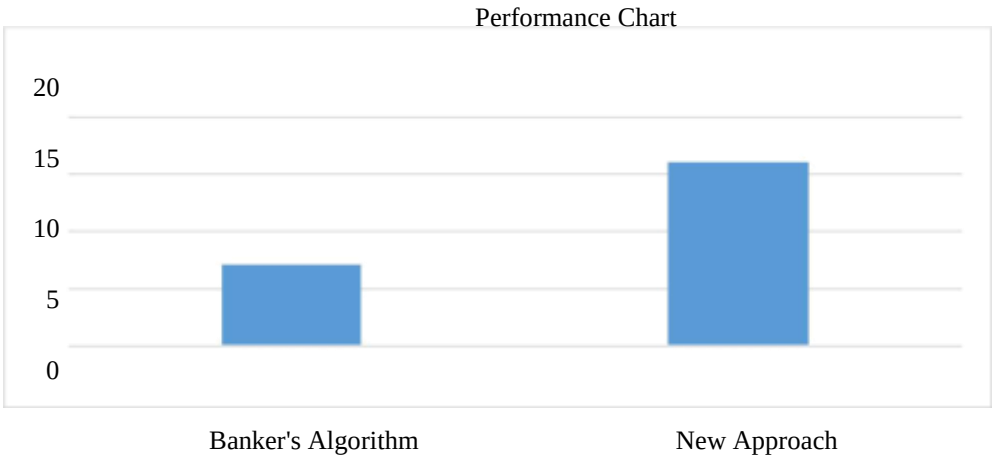

Figure 6.1 Performance Chart

Figure 6.1 defines that the total number of experiments were performed with different problems. Out of which traditional banker's approach was able to inform about no deadlock in 35% problems. The modified banker's algorithm was able to avoid deadlock in 80% of total experiments. Thus the performance of improved algorithm was 45% better than the banker's approach.

7. Conclusion & Future Work

7.1 Conclusion

This research shows the Banker's Algorithm working, problem in original algorithm to identify the reason for failing the process execution. Here Dynamic Banker's algorithm solve the existing problem of the original Banker's algorithm.

Results prove that modified Banker's Algorithm shows that particular process which needs more resources of what type. It also shows that it is in the safe sequence or not, so it is very easy to add which resource is needed to the process so the problem can be solved by this approach.

7.2 Future Scope

The present and future of this area is bright, and full of opportunities and great challenges as it processes high demands.
In future it can be used for the auto added process and killing the undesired process.

REFERENCES

[1] Goswami, Vaisla and Ajit Singh, "VGS Algorithm: An Efficient Deadlock Prevention Mechanism for Distributed Transactions using Pipeline Method" International Journal of Computer Applications (0975 – 8887) Volume 46–No.22, May 2012

[2] William Stallings, "Operating Systems: Internal and Design Principles", Fifth Edition, Pearson Publications, 2008.

[3] N. Ramasubramanian, Srinivas V.V., Chaitanya V, "Studies on Performance Aspects of Scheduling Algorithms on Multicore Platforms," International Journal of Advanced Research in Computer Science and Software Engineering, Vol 2, Issue 2, February 2012.

[4] Smriti Agrawal, Madhavi Devi Botlagunta and ChennupalliSrinivasulu, "A Total Need based Resource Reservation Technique for Effective Resource Management", International Journal of Computer Applications (0975 – 8887), Volume 68– No.18, April 2013

[5] H. S. Behera, RatikantaPattanayak, PriyabrataMallick, "An Improved Fuzzy-Based CPU Scheduling (IFCS) Algorithm for Real Time Systems," International Journal of Soft Computing and Engineering (IJSCE) (2231-2307), Volume-2, Issue-1, March 2012

[6] SarojHiranwal, Dr.K.C.Roy, "Adaptive Round Robin Scheduling using Shortest Burst Approach Based on Smart Time Slice," International Journal of Data Engineering (IJDE), Volume 2, Issue 3 2012.

[7] A. Silberschatz, P. B. Galvin, and G. Gagne, Operating Systems Concepts, 6th edition, Addison-Wesley, Reading, Mass, pp. 204, 243, 244, 266, 2002.

[8] G. Nutt, Operating Systems, a Modern Perspective, 2nd edition, Addison-Wesley, Reading, Mass, Pages.150-279, 2000.

[9] B Madhavi Devi, Smriti Agrawal, Ch. Srinivasulu, "An Efficient Resource Allocation Technique for Uni-Processor System" International Journal of Advances in Engineering & Technology (IJAET) Volume 6 Issue 1, March 1, 2013.

[10] H. Wu, W. Chin, and J. Jaffar, "An Efficient Distributed Deadlock Avoidance Algorithm for the AND Model," IEEE Trans. on Software Engineering, vol.28, no.1, pp. 18-29, Jan. 2002.

[11] W. Lin and D. Qi, "Research on Resource Self-Organizing Model for Cloud Computing," in Proc. IEEE

International Conference on Internet Technology and Applications, pp. 1–5, 2010.

[12] X. Nan, Y. He, and L. Guan, "Optimal resource allocation for multimedia cloud based on queuing model," in IEEE MMSP. pp. 1–6, Oct. 2010.

[13] Nan, Xiaoming," Optimal resource allocation for multimedia cloud in priority service scheme ", in IEEE International Symposium on Circuits and Systems (ISCAS), 2012.