Available online on: https://ijact.in

Date of Submission	23/02/2019
Date of Acceptance	31/07/2019
Date of Publication	06/08/2019
Page numbers	3242-3249(8 Pages)

Cite This Paper: Madhavi Devi B, Smriti agrawal, R Rajeshwar rao (2019). Dynamic Budget: Threshold Based Resource Reservation Technique, 8(7), COMPUSOFT, An International Journal of Advanced Computer Technology. ISSN: 2320-0790, PP. 3242-3249.

ISSN:2320-0790

COMPUSOFT
An International Journal of Advanced Computer Technology

DYNAMIC BUDGET-THRESHOLD BASED RESOURCE RESERVATION TECHNIQUE

Madhavi Devi Botlagunta*[1], Dr. Smriti Agrawal[2], Dr. R Rajeshwara Rao[3]

[1]Research scholar, Jawaharlal Nehru Technological University, Kakinada
[2]Associate Professor, Chaitanya Bharathi Institute of Technology, Hyderabad
[3]Professor, JNTUK University College of Engineering, Vizianagaram
*Corresponding Author

Abstract: The effective utilization of the resources is recognized as essential tool not only for the cost effectiveness of the system but also for the environment. Sharing of the resources improves its utilization but also creates contention among the processes using it. This contention may lead to frequent deadlocks. Many deadlock avoidance techniques exist for avoiding the deadlock while sharing the resources. However, all these technique suffer from two major limitations: 1) they assume that the resource requirement for each process is known in advance; 2) overhead for decision making for resource granting to ensure that any future deadlock must be avoided. This paper proposes a Dynamic Budget for Threshold based Resource Reservation Technique for Deadlock Avoidance (DB-TRA) which extends the existing Threshold based Resource Reservation Technique for Deadlock Avoidance (TRA) [24] technique. The existing TRA address the second limitation of excessive overhead for resource granting by forward calculation to avoid deadlock. However, this TRA technique also suffers from the first limitation and assumes that the resource requirement of each process is known in advance. The proposed DB-TRA technique address this limitation of the deadlock avoidance techniques by proposing a dynamic budget which eliminates the need of prior knowledge of the resource requirement of any process, the budget is adjusted dynamically to cater the need of the process. The proposed technique uses the resource reservation as suggested in TRA for minimizing the overhead in granting of the requested resources. The simulation result shows that the proposed DB-TRA technique performs better than existing deadlock avoidance technique.

Keywords: Deadlock; Deadlock handling; Deadlock Prevention; Deadlock Avoidance; Resource allocation; Banker's Algorithm; Resource management.

I. INTRODUCTION

Modern computing systems composed of multiple resources to serve multiple processes. Each process performs a set of operations. Each operation may need one or more resources. The resources acquired in one operation may not be released before new resources are acquired for the subsequent operation. These resources must be shared for their effective utilization. Resource manager in traditional computing systems often assigns fixed number of resources to a process to satisfy its requirement. Such type of static resource allocation often results in the underutilization of resources. On the other hand, if the resources are shared then if they are not properly managed the system may end in a deadlock, where no process is able to complete.

Sharing of resources is not only important in computing systems but also in real world. Some of the real world examples of shared resources are construction of building by contractor, transportation systems, manufacturing systems, engineering tools, real time systems and controlled systems, etc. For example [22], a project of constructing villas can be considered as a system, where the contractor constructs for various clients. In this system construction of each villa is a process. Each of these processes can be divided into operations (foundation, construction, interiors) requiring resources as money and time to complete. The contractor has limited initial amount to start and promises a time frame to deliver the completed villa (along with the interiors). The client pays back the construction cost only upon the completion of villa with its interiors. Thus, if the contractor uses all his initial amount for constructing the foundation for all the villas then he may not be left with any amount for further construction and interiors. The client will pay him back only on completion of the interiors hence, this can lead to a deadlock. On the other hand, if he finishes one villa at a time the resources (money) in his account will be underutilized and the time for completion of the project will also increase. Thus, it is crucial to manage the resources for their effective utilization.

This paper intends to develop a resource management technique that will manage the resources in an efficient way and also avoid deadlock. The proposed technique reduces the overhead by pre-estimating the resource requirements for a process and avoids the deadlock.

II. RELATED WORK

Coffman et. al. [3] and Holt [6] formalize the problem of deadlocks in term of Graph-theoretic model and identified the conditions for deadlock occurrence. Holt's deadlock model was transformed into a finite state automaton with the final states corresponding to deadlock by Nutt [7].

Broadly, there are four strategies to handle deadlocks; ignorance, detection, avoidance, prevention. The easiest solution is ignoring deadlock when it does not lead to critical situation for specific application. Deadlock detection methods try to detect and resolve them once they occur. A deadlock avoidance (DA) method avoids deadlocks in advance. The Deadlock Prevention (DP) techniques ensure that any one of the four essential conditions for deadlock does not occur.

Deadlock prevention affects the process itself and may not be possible to implement in all the systems. The deadlock avoidance techniques perform forward calculations based on the prior knowledge of the resource requirement for each process. One of the most popular deadlock avoidance techniques was developed by Dijkstra [4] as Banker's Algorithm for a single resource type. Habermann [5] extended this Banker's Algorithm to handle multiple resource types. Banker's algorithm assumes prior knowledge of the maximum resource requirement for every process in the system at any given time. Devillers [8] assumed that the future resource requirement for each process is represented as flowchart. Based on this flowchart the operating system and the process behave as if they are playing a game to find the deadlock and deadlock avoidance states. A dynamic algorithm is presented by Fontao [9] for DA. Frailey [10] presented a DA algorithm implemented on MACE operating system, of CDC 6500 at Purde University. Dixit and Khuteta[33]also assume prior knowledge of the resource requirement for all processes. They permit the resource requirement to change at run time for avoiding deadlock. They stack the processes as per the remaining Needs. Kawadkar et. al.[34]re-examined the Banker's algorithm to consider the processes in the waiting state. They prioritized the waiting processes based on the resources they were holding and the resources they further needed. They still assumed that the resources requirement of each task is known is advance. Youming Li [28] modified the Banker's algorithm making it 'n' times faster than the original. In this method a permutation matrix was generated by counting sort for each resource type. The process with highest position in the sorted sequence vector was greedily selected. The storage requirement for permutation matrix was high with respect of memory. Further, the advance knowledge of the resource requirement was also essential. Huang et.al. [40] proposed resource management for broadcasting wireless systems with limited frame length using a modification of Banker's algorithm. As in Banker's algorithm a safety judgment is made for effective allocation results. In case the system is not in safe condition, adjustments were suggested get to a more appropriate allocation result. Yin et. al. [42] addressed the multithreaded programs to avoid deadlock by using lock automation. They suggested to combining offline static analysis and runtime execution control. They created a control flow graph of a program, and then translated it into a formal model which was then analysed for to detect potential deadlocks. The deadlock avoidance was performed run time.

Pyla and Varadarajan [30] examined the need of deadlock handling in a multi-thread environment. They proposed Sammati for deadlock detection and recovery in a POSIX multi-threaded program. They suggested recovering from a deadlock by rollback once it is detected. Gwad et.al. [35] proposed a deadlock detection algorithm for threads. It indicated the deadlock initiator thread in advance. The resource allocation problem in grid systems is examined by Zhang et. al. [36]. Based on atomic transaction they present a fast co-allocation approach for the resources. Authors [37, 42] suggested detecting deadlock in a multi-processing system. However, they assumed that each request must be for a single unit of a resource. Cahit [38] presented a deadlock detection method using an undirected graph with labels as 0 or 1. It assumes that the deadlock would occur only if a cycle with alternating 0s and 1s is formed. Shiu et. al. [39] presented a hardware unit for deadlock detection. Xiao and Lee [27] used multiple units in parallel for deadlock detection leading to a run time complexity of O(log2(min(m,n))).All the deadlock detection techniques

detect deadlock after its occurrence and have as overhead in terms of deadlock recovery as well. In deadlock recovery some or all the processes are either rolled back and/or the resources are pre-empted. These steps for deadlock recovery may not be possible to implement in all systems Authors [23, 24, 25, 26] proposed deadlock handling by performing resource reservation where subsets of the resources are reserved by the system and the remaining resources are granted without any further checks. The reserved resources can be used by a process only if it is likely to complete and relinquish all the resources allocated to it. These techniques have low time complexity making them very efficient as compared to all the above reported techniques. However, they still suffer with one of the lacunas of the deadlock avoidance strategies, that is, prior knowledge of the resource requirement. The proposed technique aims to overcome this limitation by providing a budget that can adjust and still ensure deadlock free system.

III. SYSTEM MODEL

Dynamic resource allocation scheme assigns computing resources to processes based on demand as and when needed. Existing work has mainly concentrated on the deadlock avoidance techniques. They assume that the execution time and resource requirement for each process is known in advance. This assumption is unrealistic at times or has considerable overhead in real time scenarios. Hence, this paper presents a deadlock free dynamic budget-threshold based resource management technique. The main goal is to dynamically allocate the resources to the processes based on their budget calculated through threshold to improve resource utilization and increase the performance of the system.

The system is consists of 'm' resources; R_1, R_2, $R_3 \dots R_m$, with α_1, $\alpha_2, \dots \alpha_m$ instances of each type.The resources must be shared by $'n'$ independent processes, P_1, P_2, $P_3 \dots P_n$. Various data structures used are as described below:

- ***Actual Resource (Actual[i][j]):***It is a two-dimensional array of size $n \times m$. It is the accurate number of resources requested by a process during its execution. If Actual[i][j] equal k, then process $P_i$has requested k instances of resources type R_j so far. It is used to initializes the budget if a process is aborted.
- ***Request (Request[i][j]):*** It is a two-dimensional array of size $n \times m$. A process can be viewed as sub-processes. These sub-processes within the process $P_i$while executing request for resources. If Request[i][j] = k, it implies k instances of resource type R_j are requested by the process $P_i$for it's further execution.
- ***Allocation (Allocation[i][j]):*** The resources once granted to a process for its execution are said to be allocated to it. Allocation[i][j] is two dimensional array $n \times m$, specifying the number of instances of a resource held by a process for its execution. That is, if Allocation[i][j] = k, then process $P_i$is permitted to use k instances of resources type R_j for it's exclusive use during it's execution.
- ***Threshold (Threshold[i]):*** It is an array of m elements, with Threshold[j]=⌈⌊Request[i][j] ∀i=1,2…n⌋, 0⌉. It is least instance of a resource needed by one of the processes. That is, if threshold[j] =k, then k instances of resource type R_j are sufficient for at least one process $P_i$to complete its execution.
- ***Reserve Pool (Reserve[j]):*** An 'm' element array, where Reserve[j]=Threshold[j], these resources are used by a process to complete and avoid deadlock.
- ***Available Pool (Available[j]):*** All the resources in the system are divided into a reserve and available pool. The available pool contains all resources that are both unreserved resources and not allocated to any process. Thus, if Available[j] = k, then 'k' unreserved instances of resource type R_j are unallocated to any process and thus, are available for possible allocation.
- ***Budget (Budget[i][j]):*** A two dimensional array$n \times m$, defining the resources anticipated to be needed by a process. If Budget [i][j] = k, then process $P_i$is allotted a budget of 'k' instances of resources R_j in its life time. However, this Budget [i] is changed dynamically with the process P_i resource request; Budget[i][j] =Request[i][j] + Threshold[j] + Allocation[i][j].
- ***Need (Need[i][j]):*** It is two-dimensional array $n \times m$, quantifying the remaining resources a process is expected to request. That is, if Need[i][j] = k, then k more instances of resource $R_j$are likely to be requested to complete process P_i's execution. Mathematically,
  $Need[i][j] = Budget[i][j] - Allocation[i][j]$.
- ***Throughput:*** is the number of processes completing per unit of time.
- ***Turnaround time:*** is the overall time a process takes from the time it is submitted to the time it completes, it includes the waiting time of the process for the CPU or the I/O and execution

Safety Sequence [4]: "It is a sequence of process $\langle P_i, P_{(i+1)}, P_{(i+2)} \dots P_n \rangle$ which is considered to be safe if for each process, with the current allocation the Needs of all the processes can still be satisfied by the currently available resources plus resources held by all processes prior to it in the sequence". Safety Sequence may not be unique and can be estimated by algorithm suggested in [4].

Safe State [4]: "The system for which the Safety Sequence exists indicating Budget of each process can be satisfied is said to be in Safe State. That is, no deadlock is expected in the future as long as the present state of the system remains unaltered".

Unsafe State[4]:"The system which is not in Safe State is considered to be in Unsafe State, indicating that the system is heading towards a deadlock".

IV. Dynamic Budget for Threshold based Resource Reservation Technique for Deadlock Avoidance (DB-TRA)

The present work aims to propose a new deadlock management technique for systems where no prior knowledge about number of processes or their behavior is available. This contrasts with the existing deadlock Avoidance Techniques [24, 26, 28] where number of processes is assumed to be fixed and each process must declare its maximum resource requirement. This estimation of the maximum resource requirement must be made before the process starts executing and has an overhead. Further, the existing techniques do not allow a process to modify its resource requirement once declared at its inception.

Present work extends Deadlock Avoidance technique namely Threshold Based Resource Allocation Technique (TRA) [24] to overcome its limitation. The TRA technique assumes that the overall resource requirement (Max[i][j]) for every process is known prior to its execution. This technique has low overhead as it reduces the safety sequence estimation overhead for deadlock avoidance by reserving a portion of the resources. The deadlock is avoided by granting these reserved resources. The estimation of the resources to be reserved is based on a threshold of Threshold[j]=⌈⌊Need[i][j] ∀i=1,2…n⌋, 0⌉ where Need[i][j]= Max[i][j]-Allocation[i][j]. TRA endeavored that at least one process P has sufficient resources for its completion. Once this process P, completes it can relinquish all the resources it was holding. TRA considers only the Available Pool to take the decision of granting the resources incurring minimal overhead. That is, if sufficient resources are available in the Available pool the process request for resources is granted without any further checks. However, in case the Available pool does not contain sufficient resources then the reserve pool resources could be used such that the all the resources this process will ever need (i.e. Need[i]) to complete are granted. On completion this process will release all the resources allocated to it. Thus, the deadlock is avoided.

The assumption that the resource requirement is known in advance is serious limitation of most of the deadlock avoidance techniques as this estimation has considerable overhead. The present work proposes a dynamic budget (DB-TRA) technique to overcome the limitation of TRA technique of assuming that the maximum (Max[i][j]) resources required by each process is known in advance.

The DA-TRA proposes an initial budget of resources to be given to each process. This budget is initialized with no prior knowledge of the process resource requirement rather on the current request made by it. However, this budget is updated to accommodate the resource requests made by a process. The safety sequence test ensures that the system is deadlock free with the proposed budget. Thus, every time the process makes a new resource request within the budget allotment is done as per the TRA technique [24] using available resources and reservation pool. However, if the requested resources are beyond the budget granted to a process then the budget is modified to accommodate the new request. To ensure the deadlock Free State, the safety sequence is estimated. If the system is safe the requested resources are granted. However, if the safety sequence does not exist the process state is stored (check pointed) and queued in the pending state with its budget updated to the actual resource request. This process may be aborted to prevent the hold and wait condition which may lead to a deadlock. However, this can be postponed till some process requests for the resources held by this pending process. This will reduce the repercussions of over budgeting.

The following example illustrates the working of the proposed technique.

Example 1: Consider a system with four resource types {R1, R2, R3,R4} with instances {8,13,11,10} respectively. At time T0 processes arrive with their resource requests as shown in table 1. The resource actual requirement to complete is not known though. The budget for each process is initialized using Budget[i][j]=Request[i][j] + Threshold[j] + Allocation[i][j] where Request[i][j] is the resources requested by each process Pi, Threshold[j] is initialized as ⌈⌊Request[i][j] ∀i=1,2…n⌋, 0⌉ and Allocation[i]={0}. Thus, Threshold = {1, 2, 1, 2} and the budgets are estimated as shown in the table 1.

All the requests are granted and Allocation, Request and Need matrixes are updated accordingly. Eventually, the resources remaining are {3, 6, 5, 4} from which {1, 2, 1, 2} will be in the Reservation pool and remaining {2, 4, 4, 2} will be in the Available pool. The Safety Sequence is {P1,

Table 1: Process arrival at time T0

	Actual requirement (not known)				Allocation				Request at time T0				Budget at time T0 (allocation+request+threshold)				Need at time T0			
	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4
P1	0	0	1	0	0	0	0	0	0	0	0	0	1	2	1	2	1	2	1	2
P2	2	7	5	0	0	0	0	0	0	2	1	0	1	4	2	2	1	4	2	2
P3	6	6	5	2	0	0	0	0	4	3	1	2	5	5	2	4	5	5	2	4
P4	4	3	5	4	0	0	0	0	1	0	2	4	2	2	3	6	2	2	3	6
P5	0	6	5	0	0	0	0	0	0	2	2	0	1	4	3	2	1	4	3	2
Before Allocation : Available Pool ={7, 11, 10, 8}													Reserve Pool ={1, 2, 1, 2}							
After Allocation : Available Pool ={2, 4, 4, 2}													Reserve Pool ={1, 2, 1, 2}							

Table 2: Process arrival at time T1 when process P2 makes the request (0, 3, 2, 0)

	Actual requirement (not known)				Allocation				Request at time T1				Budget at time T0 (allocation+request+threshold)				Need at time T0			
	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4
P1	0	0	1	0	0	0	0	0	0	0	0	0	1	2	1	2	1	2	1	2
P2	2	7	5	0	0	2	1	0	0	3	2	0	1	7	4	2	1	5	3	2
P3	6	6	5	2	4	3	1	2	0	0	0	0	5	5	2	4	1	2	1	2
P4	4	3	5	4	1	0	2	4	0	0	0	0	2	2	3	6	1	2	1	2
P5	0	6	5	0	0	2	2	0	0	0	0	0	1	4	3	2	1	2	1	2
Before Allocation : Available Pool ={2, 4, 4, 2}													Reserve Pool ={1, 2, 1, 2}							
After Allocation : Available Pool = { 2, 1, 2, 2}													Reserve Pool ={1, 2, 1, 2}							

P2, P4, P3, P5}. Thus, the system is in safe state.

Suppose at time T1, process P2 with Need of {1, 2, 1, 2}, requests for {0, 3, 2, 0} resource instances. In other words, P2 requests for resource R2 and R3 more than its presumed budget. However, the proposed DB-TRA technique allows a process to modify its Budget as Budget[2][j]=Request[2][j] + Threshold[j] + Allocation[2][j]. In this case the Budget of process P2 will be modified from {1,4, 2, 2} (from table 1) to {1,7,4,2}(in table 2) which in turns modifies the Need. However, before the request can be granted the safety sequence is estimated as {P1, P2, P3, P4, P5} based on which the request is granted. Thus, the system is in safe state at time T1.

Suppose at time T2, Process P4 requests for {1, 0, 1, 2} resource instances whose Need is {1, 2, 1, 2}. At this time (Request<=Need) all the requested resources are within the Budget already allocated. The existing TRA technique [24] is used to take the decision and the safety sequence is not estimated. The requested resources {1, 0, 1, 2}are available in the Available Pool and are thus granted. The safety sequence is not needed for deadlock avoidance in TRA technique, however to prove that the system is in deadlock free the safety sequence is estimated as {P1, P2, P4, P3, P5}.

The proposed DB-TRA technique can be stated in the form of an algorithm as follows:

Algorithm: Safety Sequence Algorithm *can be found at [4]**:***

Algorithm: Dynamic Budget for Threshold based Resource Reservation Technique for Deadlock Avoidance (DB-TRA):

// Input: Available[j]=[α_j]; {System has 'm' resource types, i.e.,R_1, R_2, R_3 ... R_m, with α_1, α_2, ... α_m instances of each type.}

Begin

1. **Initialize Threshold[j] =Reserve[j]=**⌈⌊Request[i][j] ∀i=1,2…n⌋, 0⌉∀ j=1, 2, …m
2. **Initialize Budget [i][j]= Allocation [i][j] = Need[i][j]= {0}**∀i=1,2…n**and** ∀ j=1, 2, …m *//initially no budget or resources are allocated to any process*
3. **Available[j]= Available[j]-Reserve[j]** ∀ j=1, 2, …m
4. **For (Request[i][j] by a process P_i)**

Do

a. If Request[i][j]>Need[i][j] *for any j = 1, 2, ... m // indicating that the estimated budget is not sufficient for this process and must be modified*

I. ***Budget[i][j] = Allocation[i][j] + Request[i][j]+ Threshold[i][j]*** *//Budget is what a process already holding plus what it wants more and some resources that it might want in future.*

II. ***If Safety Sequence exists****// indicates that this new Budget will not lead to a deadlock*

i. *Accept process* ***P_i***

ii. ***If(Request[i][j]<=Available[i][j])***

a. ***Allocate(Allocation, Request, Available, 0)*** *// Allocate the requested amount from the Available pool only*

else

b. ***Allocate(Allocation, Need, Available, Reserve)****// Allocate all the resources as per the budget and expect this process to complete*

iii. ***Request****[i][j] ={0}*

Else

iv. *Save the status and insert the process Pi in pending queue*

v. ***Budget[i][j] =Allocation[i][j] + Request[i][j]***//preventing starvation due to over budgeting

Else

b. *If* **(Request*[i][j]<=Need[i][j] && Request[i][j]<=Available[i][j])*** *//indicates that the process requests resources within its budget and they are also available in the system*

I. ***Allocate(Allocation, Request, Available, 0)*** *// Allocate the requested amount from the Available pool only*

II. ***Request[i][j] ={0}***

Else *//indicating that the processes has requested resources within its budget but they are not available in the Available queue and the resources from the Reserve pool must be used*

I. ***Allocate(Allocation, Need, Available, Reserve)***

II. ***Request[i][j] ={0}***

5. **When a Process completes and releases resources.**

End

Allocate(Allocation, To_Allocate, Available, Reserve)

// Allocates the resources from the Available or Available and Reserve Pool

Begin

1. ***For (j=1 to m)***
   - ***a. Available[j]=Available[j] + Reserve[j] // merge the two pools***
   - ***b. Available[j]=Available[j] – To_Allocate[i][j]***
   - ***c. Allocation[i][j]= Allocation[i][j] + To_Allocate[i][j]***
   - ***d. Need[i][j] = Budget[i][j] – Allocation[i][j]***
   - ***e. Reserve[j]= min(Available[j], Threshold[j])***
   - ***f. Available[j]=Available[j]-Reserve[j]***

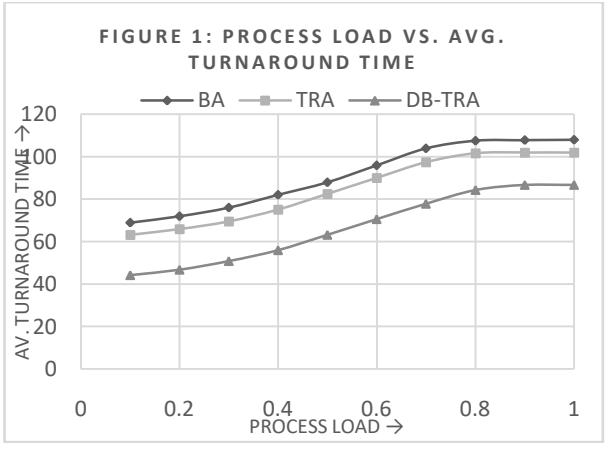
End

V. SIMULATION AND RESULTS

The process sets were synthesized and simulations are performed on them to assess the proposed: Dynamic Budget for Threshold based Resource Reservation Technique for Deadlock Avoidance (DB-TRA). Comparisons are done with the existing Banker's Algorithm (BA) and Threshold based Resource Allocation (TRA) techniques.

The comparison is done based on the Average Turnaround time which is the average of the times the processes take to complete from the time they were submitted for execution. A resource pool with up to 10 resource types was created with 0 to 20 instances for each resource type was generated randomly. Processes (1 to 100) were generated with random execution time and resource requirement.

Both existing Banker's Algorithm as well as Threshold based Resource Allocation TRA assumes that the resources required by a process are known in advance. This resource requirement estimation overhead is also taken into account in the simulations performed. As stated in the literature review section above there are multiple techniques [21] for performing this estimation, the simplest and most frequently used is Stop Watch Method. Present simulations also use this method.



The effect of process load on the average turnaround time can be seen in the figure 1.The increase in process load implies more processes in the system, hence higher contention for the resources. Higher contention may lead to more frequent deadlocks if not managed properly. It is observed that the average turnaround time increases for all the techniques as the load increases. DB-TRA eliminates the overhead for estimating the resources required by all the processes in the system. Further, DB-TRA also uses a resource reservation pool which reduces the overhead of safety sequence test for decision on granting resource allocation. The overhead saved prevents process accumulation. Lower accumulation of process in turn reduces the wait time of a process and also distributes the resource request. Hence, the proposed DB-TRA technique facilitates faster completion of the processes reducing the turnaround time as indicated in the figure 1.

VI. CONCLUSION

With the advancement in technology, the resources are shared extensively among processes. This sharing improves resource utilization but also creates contention among the processes using it. The resource contention may lead to frequent the deadlocks. Many effective deadlock avoidance techniques exist for avoiding the deadlock, but they are not popular as they have considerable overhead for 1) estimating the resource requirement of a process and 2) forward calculation for avoiding deadlocks in future. An existing Threshold based Resource Reservation Technique for Deadlock Avoidance (TRA) [24] suggests reserving a pool of resources for avoiding deadlock. They show considerable reduction in the overhead by eliminating the need for looking forward if a deadlock is likely in future. However, the TRA technique still suffers from the overhead for estimating the resource required by a process.

This paper proposed a Dynamic Budget for Threshold based Resource Reservation Technique for Deadlock Avoidance (DB-TRA) which extends this existing Threshold based Resource Reservation Technique for Deadlock Avoidance (TRA) [24] technique. The proposed DB-TRA technique suggest a budget which eliminates the need of prior knowledge of the resource requirement of any process, the budget is adjusted dynamically to cater the need of the process. The proposed technique uses the resource reservation as suggested in TRA for minimizing the overhead in granting of the requested resources. The simulation results shows that the proposed DB-TRA technique perform better than existing deadlock avoidance as well as TRA technique[24].

VII. REFERENCES

[1]. Havender, J.W., Avoiding Deadlock in Multitasking Systems, Ibm Systems Journal, 7(2):74–84(1968).

[2]. Dijkstra, E.W., Cooperating Sequential Processes, in Programming Languages, Genuys F. Editor, London, Academic Press, 1965.

[3]. Coffman, E.G., Elphick, M.J, Shoshani, A., Systems Deadlocks, Acm Computer Surveys, 3(2):67-78(1971).

[4]. Dijkstra, E.W., The Structure Of The Multiprogramming System, Communication Of The Acm, 26(1):341-346 (1968).

[5]. Habermann, A.N., Prevention of System Deadlocks, Communications of the ACM, 12(7):373– 377(1969).
[6]. Holt, R.C., Some Deadlock Properties of Computer Systems, ACM Computing Surveys, 4(3):179-196 (1972).
[7]. Nutt, G.J., Some Application of Finite State Automata Theory to the Deadlock Problem, Technical Report CU-CS-017-73, Department of Computer Scince, University of Colorado at Boulder, Colorado, 1973.
[8]. Devillers, R., Game Interpretation of the Deadlock Avoidance Problem, Communication of the ACM, 20(10):741-745(1979).
[9]. Fontao, R.O., A Concurrent Algorithm for Avoiding Deadlocks, Proceedings of the Third ACM Symposium on Operating Systems Principles, 72–79 (1971).
[10]. Frailey, D.J., A Practical Approach to Managing Resources and Avoiding Deadlock, Communications of the ACM, 16(5):323–329(1973).
[11]. Ghaffari, A., Rezg, N., XIe, X.L., Design of a live and maximally permissive Petri net controller using the theory of regions. IEEE Transactions on Robotics and Automation,19(1):137-142 (2003).
[12]. Huang, Y.S., Jeng, M.D., Xie, X.L., Chung, S.L., Deadlock prevention policy based on Petri nets and siphons. International Journal of Production Research, vol.39 (2):283- 305 (2001).
[13]. Huang, Y.S, Jeng, M.D., Xie, X.L, Chung, D.H, Siphon-based deadlock prevention policy for flexible manufacturing systems, IEEE Trans. System, Man, Cybern. A, System, Humans,36(6):1248-1256 (2006).
[14]. Li, Z.W, Zhou, M.C., Elementary siphons of Petri nets and their application to deadlock prevention in flexible manufacturing systems, IEEE Trans. on System, Man, and Cybern., part A, 34:38-51(2004).
[15]. Park, J., Reveliotis, S.A., Deadlock avoidance in sequential resource allocation systems with multiple resource acquisitions and flexible routings. IEEE Transactions on Automatic Control,46(10):1572-1583 (2001).
[16]. Tricas, F., Garcia Valles, F., Colom, J.M., Ezpeleta, J.,An iterative method for deadlock prevention in FMSs, 5th Workshop Discrete Event System, R. Boel and G. Stremersch, Eds., Ghent, Belgium,139-148(2000).
[17]. Tricas, F., Garcia-Valles, F., Colom, J.M., Ezpeleta, J.,A Petri net structure-based deadlock prevention solution for sequential resource allocation systems, IEEE International Conference Robot. Autom., Barcelona, Spain, 271-277 (2005).
[18]. Uzam, M., An optimal deadlock prevention policy for flexible manufacturing systems using Petri net models with resources and the theory of regions. International Journal of Advanced Manufacturing Technology, 19(3):192-208(2002).
[19]. Uzam, M., Zhou, M.C., Iterative synthesis of Petri net based deadlock prevention policy for flexible manufacturing systems. In Proc. IEEE International Conference on Systems, Man, and Cybernetics, 4260-4265(2004).
[20]. Xie, X.L., Jeng, M.D., 1999. ERCN-merged nets and their analysis using siphons, IEEE Trans. Robot. Autom., 15(4):692-703(1999).
[21]. David B. Stewart, “Measuring Execution Time and Real-Time Performance” in Embedded Systems Conference, Boston, September 2006.
[22]. Kees van Hee, Alexander Serebrenik, Natalia Sidorova Marc Voorhoeve, Jan van der Wal.,” Scheduling-free resource management” in Science Direct, Data & Knowledge Engineering 61 (2007) 59–75
[23]. Smriti Agrawal, B Madhavi Devi, Ch. Srinivasulu, "A Total Need Based Resource Reservation Technique For Effective Resource Management”, in Int'I Journal of Computer Applications, Volume 67, April 2013.
[24]. B Madhavi Devi, Smriti Agrawal, Ch. Srinivasulu, "An Efficient Resource Allocation Technique for Uni-Processor System", in Int' Journal of Advances in Engg & Tech (IJAET) Vol. 6 II, March 2013.
[25]. B Madhavi Devi, Smriti Agrawal, Rajeshwar Rao, “EFFECTIVE RESOURCE MANAGEMENT TECHNIQUES USING RESERVATION POOL” in IEEE International Conference on Recent Advances & Innovations in Engineering" (ICRAIE-2014), May 09-11, 2014. IEEE Conference Record # 33681.
[26]. Shubham Kumar and Saravanan Chandran, “Modified Execution Time based Resource Reservation (METRR) Algorithm”, presented at ICBIM 2016, 9-11, January 2016, ISBN: 978-1-5090-1228-2, NIT Durgapur, INDIA.
[27]. Xiang Xiao, Jaehwan John Lee.: A parallel multi-unit resource deadlock detection algorithm with O(log2(min(m, n))) overall run-time complexity. J. Parallel Distrib. Comput. 71. pp. 938–954. (2011)
[28]. Youming L, “ A Modified Banker’s Algorithm”, in Springer Innovations and Advances in Computer, Information, Systems Sciences, and Engineering pp 277-2819(2012)
[29]. Hongwei Wang ; Jinfeng Tian; Mingqi Li ; Weixin Mu; Xiangchuan Gao,” Banker's algorithm based resource allocation in next generation broadcasting wireless systems(2015) . IEEE Proc Int’l Conf Communications and Networking in China
[30]. H. K. Pyla and S. Varadarajan. Avoiding deadlock avoidance. PACT ’10, pages 75–86, New York, NY, USA, 2010. ACM
[31]. Operating systems: design and implementation, Andrew s. tanenbaum, prentice hall, 2006.
[32]. Aida.O.Abd EI-Gwad, Ahmed.I.Saleh, Mai.M.Abd-EIRazik. "A Novel Scheduling Strategy for an Efficient Deadlock Detection" IEEE. 2009
[33]. Kshipra Dixit, Ajay Khuteta, “A Dynamic and Improved Implementation of Banker’s Algorithm” in International Journal on Recent and Innovation Trends in Computing and Communication,2017, ISSN: 2321-8169 Volume: 5 Issue: 8 pg 45 – 49

[34]. Pankaj Kawadkar, Shiv Prasad, Amiya Dhar Dwivedi, "Deadlock Avoidance based on Banker's Algorithm for Waiting State Processes" in International Journal of Innovative Science and Modern Engineering (IJISME) ISSN: 2319-6386, Volume-2 Issue-12, November 2014

[35]. Aida. O. Abd El-Gwad, Ahmed. I. Saleh, Mai. M. Abd-ElRazik. "A Novel Scheduling Strategy for an Efficient Deadlock Detection" IEEE. 2009

[36]. Zhang, C., Liu, Y., Zhang, T., Zha, Y., Huang, K., "A Deadlock Prevention Approach based on Atomic Transaction for Resource Co-allocation", First International Conference on Semantics, Knowledge and Grid (SKG'05), 37-39(2005).

[37]. Kim, J. Koh,"An O(1) Time Deadlock Detection Scheme in Single Unit and Single Request Multiprocess System", Proc. IEEE Region 10 Conf. (TENCON '91), pp. 219-223. (1991)

[38]. Cahit, "Deadlock Detection Using (0, 1)-Labelling of Resource Allocation Graphs", IEEE Proc. Computers and Digital Techniques, pp. 68-72. (1998)

[39]. Shiu, P. Y. Tan, Mooney, "A Novel Parallel Deadlock Detection Algorithm and Architecture", Proc.Int'l Conf. Hardware Software Codesign (CODES '01), pp. 73-78. (2001)

[40]. Huang, Y.S., Lin, J.H., Hsu, C.N., "Comparison of deadlock prevention policies in FMS based on Petri nets siphons", In Proc. IEEE International Conference on Systems, Man, and Cybernetics, 4867-4872(2004).

[41]. Kim, J., "Algorithmic Approach on Deadlock Detection for Enhanced Parallelism in Multiprocessing Systems". Proc. Second AIZU Int'l Symp. Parallel Algorithms/Architecture Synthesis (PAS '97),pp. 233-238. (1997)

[42]. Yin,W., Stephane, L., Terence, K., Manjunath, K., Scott, M., "The Theory of Deadlock Avoidance via Discrete Control", 36th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages, PoPL,202- 214(2009).

[43]. Zhishuo Zheng , Deyu Qi, Mincong Yu, XinyangWang ,Naqin Zhou, Yang Shen, and Jing Guo, "Optimizing Job Coscheduling by Adaptive

[44]. Deadlock-Free Scheduler, Hindawi Mathematical Problems in Engineering, Volume 2018, Article ID 1438792, 18 pages.