Global Journal of Computer Science and Technology: H
Information & Technology
Volume 23 Issue 3 Version 1.0 Year 2023
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

# Optimized Round Robin CPU Scheduling for Critical Processes

By Debashish Barman, Biswajit Paul, Swastik Bhattacharya, Dr. Sourav De & Dr. Govind Prasad Arya

*Abstract-* An operating system serves as a fundamental component of any computer system. Scheduling lies at the core of operating system functionality, involving the arrangement of processes to execute in a well-defined manner. The primary goal of scheduling is to enhance system efficiency and speed. Several fundamental scheduling algorithms exist, including First Come First Serve (FCFS), Round Robin, Priority-Based Scheduling, and Shortest Job First (SJF). This thesis primarily focuses on the Round Robin Scheduling algorithm and seeks to address certain limitations associated with it.

One notable drawback of Round Robin Scheduling is the critical choice of the time quantum. If the time quantum is excessively large, the scheduling behavior closely resembles that of FCFS. Conversely, a smaller time quantum leads to a higher number of context switches. The central objective here is to overcome this limitation inherent to the traditional Round Robin scheduling algorithm, thereby maximizing CPU utilization and enhancing system efficiency.

*Keywords: CPU scheduling, round robin scheduling, priority scheduling, time quantum, waiting time, turnaround time.*

*GJCST-H Classification: ACM Code: D.4.1*

OPTIMIZEDROUNDROBINCPUSCHEDULINGFORCRITICALPROCESSES

*Strictly as per the compliance and regulations of:*

© 2023. Debashish Barman, Biswajit Paul, Swastik Bhattacharya, Dr. Sourav De & Dr. Govind Prasad Arya. This research/review article is distributed under the terms of the Attribution-NonCommercial NoDerivatives 4.0 International (CC BYNCND 4.0). You must give appropriate credit to authors and reference this article if parts of the article are reproduced in any manner. Applicable licensing terms are at https://creativecommons.org/licenses/by-nc-nd/4.0/.

# Optimized Round Robin CPU Scheduling for Critical Processes

Debashish Barman [α], Biswajit Paul [σ], Swastik Bhattacharya [ρ], Dr. Sourav De [Ѡ] & Dr. Govind Prasad Arya[¥]

*Abstract-* An operating system serves as a fundamental component of any computer system. Scheduling lies at the core of operating system functionality, involving the arrangement of processes to execute in a well-defined manner. The primary goal of scheduling is to enhance system efficiency and speed. Several fundamental scheduling algorithms exist, including First Come First Serve (FCFS), Round Robin, Priority-Based Scheduling, and Shortest Job First (SJF). This thesis primarily focuses on the Round Robin Scheduling algorithm and seeks to address certain limitations associated with it.

One notable drawback of Round Robin Scheduling is the critical choice of the time quantum. If the time quantum is excessively large, the scheduling behavior closely resembles that of FCFS. Conversely, a smaller time quantum leads to a higher number of context switches. The central objective here is to overcome this limitation inherent to the traditional Round Robin scheduling algorithm, thereby maximizing CPU utilization and enhancing system efficiency.

In this thesis, we propose an innovative algorithm that classifies processes into two categories: high-priority processes and low-priority processes. This novel scheme significantly reduces the average waiting time of high-priority processes, regardless of the presence of low-priority processes. The overall average waiting time varies based on the specific set of processes under consideration. Our analysis demonstrates that the proposed scheme consistently outperforms previously suggested methods, resulting in reduced average waiting times for the selected process sets.

*Keywords: CPU scheduling, round robin scheduling, priority scheduling, time quantum, waiting time, turnaround time.*

## I. Introduction

CPU scheduling is a fundamental practice in the realm of operating systems, orchestrating the execution of processes to efficiently utilize the CPU. This practice becomes necessary when a process must seize CPU control while another process is temporarily halted in a waiting state, typically due to resource unavailability, such as I/O operations. The primary objectives of CPU scheduling are to enhance system effectiveness, responsiveness, and fairness while maximizing CPU utilization.

*Author α σ ρ: Department of Computer Science, Cooch Behar Government Engineering College, Cooch Behar, West Bengal, India. e-mails: deba.barmanjpg@gmail.com, siswajitpaul23me@gmail, swastikbhattacharya6@gmail.com*
*Author Ѡ ¥: (Research Guide) Department of Computer Science & Engg 4Cooch Behar Government Engineering College. e-mail: govind.arya10@gmail.com*

Process scheduling, an integral component of multiprogramming operating systems, involves managing the transition of processes in and out of the CPU based on a specific strategy. These operating systems can load multiple processes into executable memory concurrently, allowing them to share the CPU through time multiplexing.

There are two principal categories of CPU scheduling algorithms: preemptive and non-preemptive. In preemptive scheduling, a process allocated to the CPU can be interrupted, and its running state may be changed to a waiting state. This approach is known for temporarily suspending logically runnable processes and is referred to as preemptive scheduling. However, frequent arrivals of high-priority processes in the ready queue can potentially lead to starvation for lower-priority processes. It's important to note that preemptive scheduling comes with the overhead of managing these process interruptions.

In contrast, non-preemptive scheduling ensures that once a process gains access to the CPU, it retains control until its execution is complete. The CPU cannot be forcibly taken away from the process until it finishes its execution. In this scenario, a process voluntarily releases the processor only after its task is done.

While various CPU scheduling algorithms exist, some common ones include First In First Out (FIFO), Shortest Job First (SJF), Priority Scheduling, and Round Robin CPU Scheduling. Each of these algorithms offers unique advantages and trade-offs in managing the CPU's allocation to processes.

## II. Literature Survey

In FCFS scheduling, jobs are executed in the order they arrive, following a "first come, first served" principle [1]. This algorithm can operate in both non-preemptive and preemptive modes depending on system requirements. It is easy to understand and implement, relying on a First-In-First-Out (FIFO) queue. However, FCFS suffers from the drawback of high average waiting times, limiting its overall performance.

Shortest Job First (SJF), also known as Shortest Job Next, prioritizes tasks based on their execution time [3]. It can function as both a preemptive and non-preemptive algorithm. SJF is particularly effective in reducing waiting times, making it a preferred choice in batch systems where CPU time requirements are known in advance. However, it is impractical for interactive systems where predicting CPU time is challenging.

Priority scheduling is a non-preemptive algorithm commonly used in batch systems [5]. Each process is assigned a priority, with the highest-priority process scheduled first, followed by processes of equal priority in a first-come-first-served manner. Priorities can be assigned based on memory, time, or other resource requirements.

Round Robin is a preemptive scheduling algorithm where each process is allocated a fixed time quantum for execution [8]. When a process's time quantum expires, it is preempted, and another process is allowed to execute for its allocated time period. Context switching is necessary to manage preempted processes effectively.

Multiple-level queues are a manual scheduling algorithm [15] that leverages various existing algorithms to categorize jobs based on common characteristics. Multiple queues are maintained for processes with similar attributes, each with its specific scheduling algorithm [8]. Priorities are assigned to each queue, enabling effective organization. For instance, OS-bound jobs can be grouped in one queue, while I/O-bound jobs reside in another. The Process Scheduler selects jobs from each queue based on the algorithm associated with that queue. Multi-level queue scheduling was developed for scenarios where processes naturally belong to different groups.

## III. Shortcomings of Existing Algorithm

We have evaluated the conventional Round Robin (RR) algorithm as our baseline scheduling approach. The RR algorithm is generally considered efficient because it ensures that all processes in the process set have an equal opportunity for execution. However, our research has identified that our system comprises both critical processes with high priority and normal (low-priority) processes. A significant limitation of the RR algorithm is its lack of consideration for process priorities, which we regard as a major drawback.

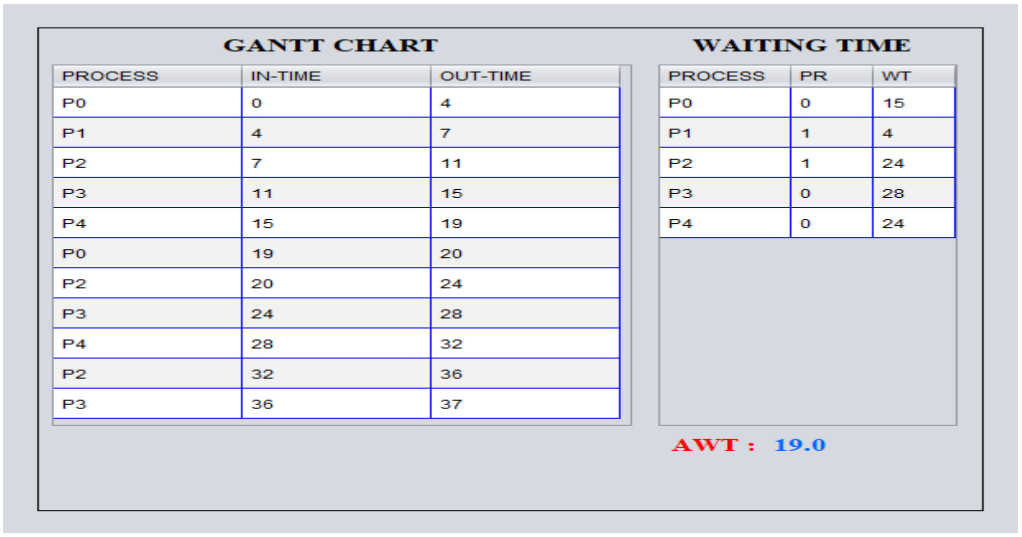
To address this limitation, we have proposed a novel methodology aimed at enhancing the RR algorithm's effectiveness.

Let's now consider the following set of processes with a fixed time quantum of 4.

*Table 1:* For the Existing Methodology, Processes in the Ready Queue

| Process Name | Priority | Burst Time |
|---|---|---|
| P0 | 0 | 5 |
| P1 | 1 | 3 |
| P2 | 1 | 12 |
| P3 | 0 | 9 |
| P4 | 0 | 8 |

Round Robin scheduling is known for its ability to ensure a fair chance for every process in the set to execute. Consequently, Figure 1 illustrates the Gantt chart and waiting times for the given set of processes.


GANTT CHART

| PROCESS | IN-TIME | OUT-TIME |
|---|---|---|
| P0 | 0 | 4 |
| P1 | 4 | 7 |
| P2 | 7 | 11 |
| P3 | 11 | 15 |
| P4 | 15 | 19 |
| P0 | 19 | 20 |
| P2 | 20 | 24 |
| P3 | 24 | 28 |
| P4 | 28 | 32 |
| P2 | 32 | 36 |
| P3 | 36 | 37 |

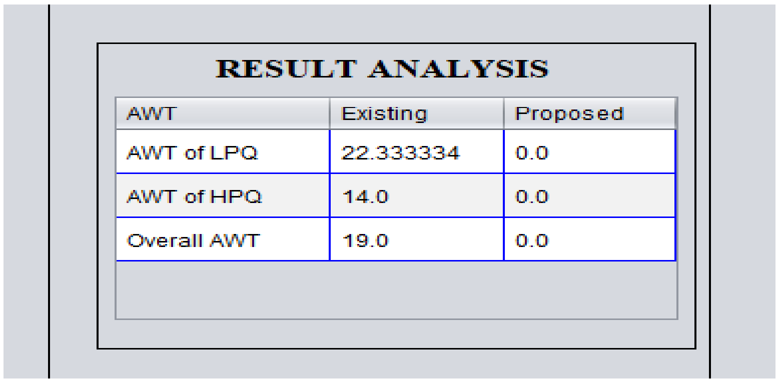
WAITING TIME

| PROCESS | PR | WT |
|---|---|---|
| P0 | 0 | 15 |
| P1 | 1 | 4 |
| P2 | 1 | 24 |
| P3 | 0 | 28 |
| P4 | 0 | 24 |

AWT : 19.0

*Figure 1:* Gantt chart of Existing Methodology

The average waiting time (AWT) for processes with both low and high priorities is presented in Figure 2 below.


RESULT ANALYSIS

| AWT | Existing | Proposed |
|---|---|---|
| AWT of LPQ | 22.333334 | 0.0 |
| AWT of HPQ | 14.0 | 0.0 |
| Overall AWT | 19.0 | 0.0 |

*Figure 2:* Waiting Time Analysis of Existing Methodology

## IV. Proposed Method

The Round Robin algorithm operates under the premise of treating all jobs with equal priority, executing processes one at a time for a specific duration known as the Time Quantum (TQ). A process can continue running until either its time quantum (TQ) is exhausted or it completes its CPU burst time. Within the system, processes have varying priorities, distinguishing between high-priority critical tasks, which demand immediate CPU attention, such as shutting down the computer due to overheating or issuing alerts for unauthorized access, and normal-priority processes, which encompass all other standard tasks.

## V. Proposed Algorithm

Our proposed algorithm is given below.

*Step 1:* Input process details, including the process name, priority, and burst time.

*Step 2:* Save the collected information in a queue labeled as "READYQ."

*Step 3:* Establish two distinct queues: "HIGHPQ" for high-priority processes and "LOWPQ" for regular-priority processes.

*Step 4:* Repeat steps 5 to 11 until the remaining CPU burst times for processes in both "HIGHPQ" and "LOWPQ" reach zero.

*Step 5:* Choose the next process from "HIGHPQ" or "LOWPQ" alternatively, with the initial selection favoring "HIGHPQ" to give higher-priority tasks precedence.

*Step 6:* If the selected process has a remaining CPU burst time greater than or equal to the time quantum, proceed to step 7; otherwise, go to step 8.

*Step 7:* Execute the chosen process for the duration of the time quantum.

*Step 8:* Continue executing the selected process until its remaining burst time reaches zero.

*Step 9:* Update the remaining CPU burst time of the corresponding process in the respective queue.

*Step 10:* Record the process's IN-TIME and OUT-TIME in a table known as the GANTTCHART.

*Step 11:* If the previous process was selected from "HIGHPQ," switch the next turn to "LOWPQ," and vice versa.

In this study, I have introduced an approach that ensures high-priority processes receive precedence in execution. The methodology I've suggested involves granting alternating opportunities to both high and low priority processes. It begins by selecting a process from the high-priority queue, followed by the selection of the next process from the low-priority queue. The following steps outline the proposed methodology.

HIGHPQ- This queue contains the processes of high priority.

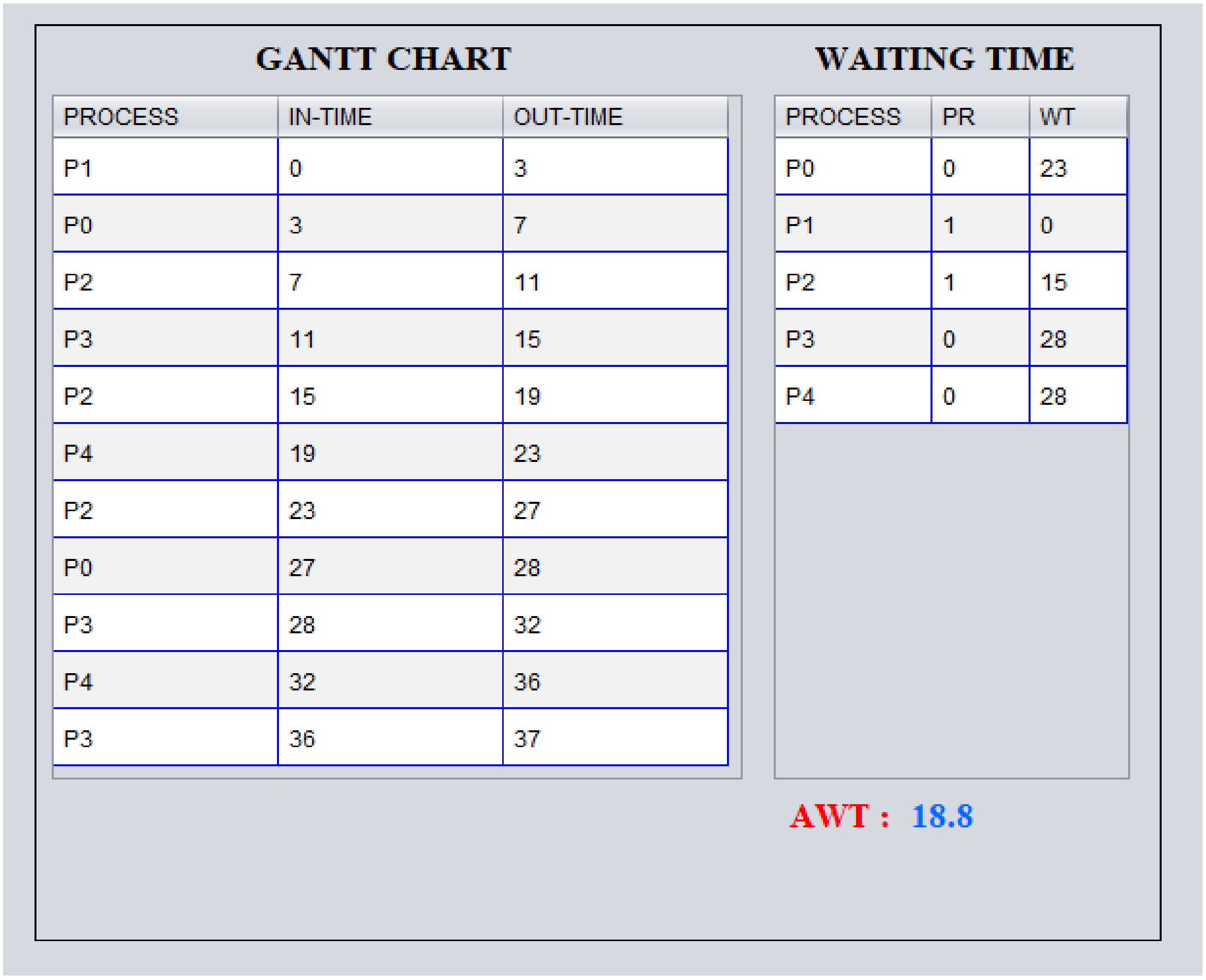
| Process Name | Priority | Burst Time |
|---|---|---|
| P1 | 1 | 3 |
| P2 | 1 | 12 |

LOWPQ- This queue contains the processes of low priority.

| Process Name | Priority | Burst Time |
|---|---|---|
| P0 | 0 | 5 |
| P3 | 0 | 9 |
| P4 | 0 | 8 |

Below, in Figure 3, you can observe the Gantt chart and waiting times for the processes listed in Table 1, using a time quantum of 4.

© 2023 Global Journals



GANTT CHART

| PROCESS | IN-TIME | OUT-TIME |
|---|---|---|
| P1 | 0 | 3 |
| P0 | 3 | 7 |
| P2 | 7 | 11 |
| P3 | 11 | 15 |
| P2 | 15 | 19 |
| P4 | 19 | 23 |
| P2 | 23 | 27 |
| P0 | 27 | 28 |
| P3 | 28 | 32 |
| P4 | 32 | 36 |
| P3 | 36 | 37 |

WAITING TIME

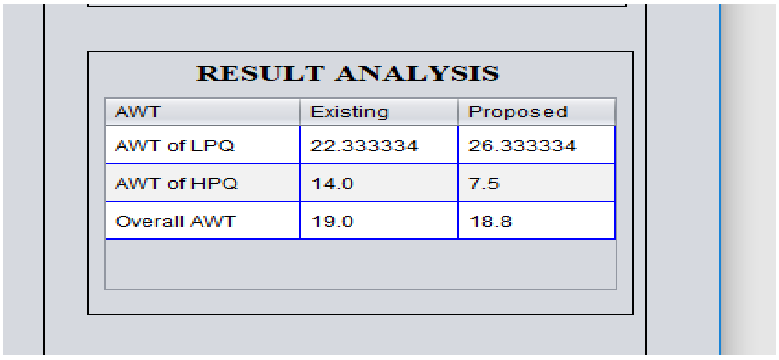
| PROCESS | PR | WT |
|---|---|---|
| P0 | 0 | 23 |
| P1 | 1 | 0 |
| P2 | 1 | 15 |
| P3 | 0 | 28 |
| P4 | 0 | 28 |

AWT : 18.8

*Figure 3:* Working of Proposed Methodology

## VI. Result and Analysis

The figure below illustrates the application of the proposed algorithm, resulting in an average waiting time for high-priority processes of approximately 7.5. This value is nearly half of the average waiting time observed when using the existing algorithm. Furthermore, the overall waiting time for the process set is significantly reduced through the implementation of the proposed algorithm.



RESULT ANALYSIS

| AWT | Existing | Proposed |
|---|---|---|
| AWT of LPQ | 22.333334 | 26.333334 |
| AWT of HPQ | 14.0 | 7.5 |
| Overall AWT | 19.0 | 18.8 |

*Figure 4:* Result Analyses of Existing Vs Proposed Methodology

The same result can be analyzed using bar chart shown in figure 5.

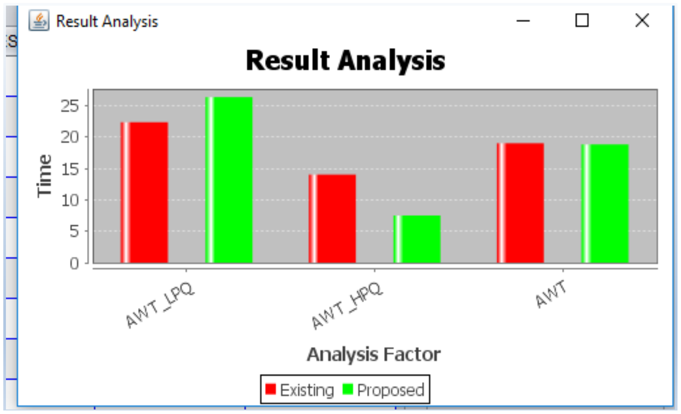


*Figure 5:* Result Analysis of Existing Vs Proposed Methodology Using Bar chart

## VII. Conclusion

In this study, I've maintained the core principle of traditional round-robin scheduling, which aims to ensure that all processes receive an equal opportunity to execute within a specific time quantum. The innovation lies in the strategic placement of high-priority processes at the rear of the ready queue, preventing them from being excessively delayed by late arrivals. The proposed approach is expected to reduce the average waiting time for high-priority processes, but it may lead to an increase in the average waiting time for normal priority processes. The overall average waiting time for all processes within the ready queue may exhibit improvement or remain unchanged, contingent on the specific mix of processes.

Although the proposed algorithm demonstrates enhanced performance for high-priority processes, there remains an ongoing drive for continued improvement. In the future, these results could potentially be refined by introducing variable time quantum strategies. Furthermore, optimizing the algorithm's execution can be accomplished by leveraging more efficient data structures.

## References Références Referencias

1. Sanjay Kumar Panda and Saurav Kumar Bhoi, "An Effective Round Robin Algorithm using Min-Max Dispersion Measure", International Journal on Computer Science and Engineering, 4 (1), pp. 45-53, January 2012.
2. Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, "Operating System Concepts", Sixth Edition.
3. Rakesh Mohanty, H. S. Behera, Khusbu Patwari, Monisha Dash, "Design and Performance Evaluation of a New Proposed Shortest Remaining Burst Round Robin (SRBRR) Scheduling Algorithm", Proc. of International Symposium on Computer Engineering & Technology 2010, Vol 17, pp. 126-137, 2010 .
4. Abdulrazak Abdulrahim, Salisu Aliyu, Ahmad M Mustapha & Saleh E. Abdullahi, (2014) "An Additional Improvement in Round Robin (AAIRR) CPU Scheduling Algorithm", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 4, Issue 2, pp 601-610.
5. Improvised Round Robin (CPU) Scheduling Algorithm. Sirohi, Abhishek; Pratap, Aseem; Aggarwal, Mayank // International Journal of Computer Applications;Aug2014, Vol. 99, p40.
6. Pallab Banerjee, Probal Banerjee, Shweta Sonali Dhal, "Comparative Performance Analysis of Average Max Round Robin Scheduling Algorithm (AMRR) using Dynamic Time Quantum with Round Robin Scheduling Algorithm usingStatic Time Quanmtum", International Journal of Innovative Technology and Exploring Engineering, 1(3), pp. 56-62,August 2012.
7. P.Surendra Varma, "A Finest Time Quantum for Improving Shortest Remaining Burst Round Robin (SRBRR) Algorithm", Journal of Global Research in Computer Science, 4 (3), pp. 10-15, March 2013.
8. Two Queue based Round Robin Scheduling Algorithm for CPU Scheduling. Jindal, Srishty; Grover, Priyanka // International Journal of Computer Applications;Nov2014, Vol. 105 Issue 1-18, p21.
9. A 2LFQ Scheduling with Dynamic Time Quantum using Mean Average. Lenka, Rakesh K.; Ranjan, Prabhat // International Journal of Computer Applications;6/1/2012, Vol. 47, p15.
10. Raman, Dr. Pradeep Kumar Mittal, "An Efficient Dynamic Round Robin CPU Scheduling Algorithm (EDRR)", International Journal of Advanced Research in Computer Science and Software Engineering, 4 (5), pp. 907-910, May 2014.
11. Silberschatz, A., P.B. Galvin and G. Gagne, Operating Systems Concepts. 7th Edn., John Wiley and Sons, USA., ISBN:13: 978-0471694663, pp. 944.

© 2023 Global Journals

12. An Effective Round Robin Algorithm using Min-Max Dispersion Measure. Panda, Sanjaya Kumar; Bhoi, Sourav Kumar // International Journal on Computer Science & Engineering; Jan2012, Vol. 4 Issue 1, p45.
13. Designing Various CPU Scheduling Techniques using SCILAB. Saini, Mona // International Journal of Computer Science & Information Technolo; 2014, Vol. 5 Issue 3, p2918.
14. Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of the Now Running Processes. Matarneh, Rami j. // Ameri- can Journal of Applied Sciences;2009, Vol. 6 Issue 10, p1831.
15. R. J. Matarneh, "Seif-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of the Now Running Proceses", American Journal of Applied Sciences, 6(10), pp. 1831-1837, 2009.
16. H. S. Behera, R. Mohanty, and D. Nayak, "A New Proposed Dynamic Quantum with Re-Adjusted Round Robin Scheduling Algorithm and Its Performance Analysis", International Journal of Computer Applications, 5 (5), pp. 10-15, August 2010.
17. E. O. Oyetunji, A. E. Oluleye," Performance Assessment of Some CPU Scheduling Algori-thms", Research Journal of Information Techno-logy,1(1): pp 22-26, 2009
18. Ajit Singh, Priyanka Goyal, Sahil Batra," An Optimized Round Robin Scheduling Algorithm for CPU Scheduling", (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 07, 2383-2385, 2010.
19. Rami J. Matarneh."Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of Now Running Processes", American J. of Applied Sciences 6 (10):1831-1837, 2009.
20. Abdulrazak Abdulrahim, Saleh E. Abdullahi & Junaidu B. Sahalu, (2014) "A New Improved Round Robin (NIRR) CPU Scheduling Algorithm", International Journal of Computer Applications, Vol. 90, No. 4, pp 27-33.
21. Sourav Kumar Bhoi, Sanjaya Kumar Panda, Debashee Tarai, "Enhancing cpu performance using subcontrary mean dynamic round robin (smdrr) scheduling algorithm" ,JGRCS, Volume 2, No. 12, December 2011, pp.17-21.
22. Rakesh Mohanty, H. S. Behera, Khusbu Patwari, Monisha Dash, "Design and Performance Evaluation of a New Proposed Shortest Remaining Burst Round Robin (SRBRR) Scheduling Algorithm", International Symposium on Computer Engineering & Technology (ISCET), Vol 17, 2010
23. Ishwari Singh Rajput," A Priority based Round Robin CPU Scheduling Algorithm for Real Time Systems", (IJIET)International Journal of Innovations in Engineering and Technology Vol. 1 Issue 3 Oct 2012.
24. Manish Kumar Mishra & Abdul Kadir Khan, (2012) "An Improved Round Robin CPU Scheduling Algorithm", Journal of Global Research in Computer Science, Vol. 3, No. 6, pp 64-69.
25. P.Surendra Varma , "A FINEST TIME QUANTUM FOR IMPROVING SHORTEST REMAINING BURST ROUND ROBIN (SRBRR) ALGORITHM" Journal of Global Research in Computer Science, 4 (3), March 2013, 10-15.
26. Rakesh Kumar Yadav, Abhishek K Mishra, Navin Prakash, Himanshu Sharma," An Improved Round Robin Scheduling Algorithm for CPU Scheduling", (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 04, 1064-1066, 2010.

© 2023 Global Journals