Self-Adjusted-Round-Robin Scheduling with Improved Dynamic Quantum Selection Strategy

Arjun Singh Saud
Asst. Professor ,
Central Department of Computer Science and IT
Tribhuvan University, Kathmandu, Nepal
Email: arjunsaud@cdcsit.edu.np

Abstract: *Performance of Round Robin (RR) scheduler depends upon quantum size. Smaller quantum causes larger context switch overhead and larger quantum size causes RR policy to be degenerated to First-Come-First-Serve policy. Self Adjusted Round Robin (SARR) is an RR based scheduling policy with dynamic-time-quantum. This research paper modifies the quantum selection strategy of SARR policy in case of even number processes and experimentally verifies that improved SARR policy gives better performance than SARR policy by using reduced quantum size without increasing the number of context switches.*

Key Words: *Scheduling Algorithms, Dynamic Quantum, Process Scheduling, RR Scheduler, SARR Scheduler*

1. INTRODUCTION:

Scheduling is the fundamental concept in operating systems. In multitasking and multiprogramming environment it is necessary to choose a process among the number of processes present in the job pool for execution. Allocation of CPU to the processes is done by scheduler, which is decided by some scheduling algorithms. First-Come-First-Serve (FCFS), Shortest-Job-First (SJF), Priority & Round Robin (RR) are different types of scheduling algorithms. Among them, RR is the most popular preemptive scheduling algorithm. In non-preemption, CPU is assigned to a process until its execution is completed. But in preemption, running process is forced to release the CPU by another process [1].

RR (Round Robin) scheduler is the most popular algorithm due to its fairness and starvation free nature, which is achieved by using the time quantum. As the time quantum is static, it causes less context switching in case of larger time quantum and high context switching in case of smaller time quantum. Large number of context switches creates high system overhead and fewer context switches causes RR scheduler to be degenerated to FCFS scheduling policy. Thus, the performance of the system solely depends upon the choice of optimal time quantum, which is dynamic in nature [2]. There are varieties of techniques that had been tried for this achievement, among these techniques the SARR (Self-Adjustment-Round-Robin) is also an approach based on dynamic-time-quantum [3]. The idea of this approach is to make the time quantum repeatedly adjusted according to the burst time of the now-running processes. In this approach the median based strategy is used for finding optimum time quantum to reduce the performance parameters like number of context switches, average waiting time (AWT), average turnaround time (ATAT) etc.

2. PROBLEM DEFINITION

Finding dynamic quantum based on median is widely researched topic. All of them use some variants of following equation:

$$Q = \overline{X} = \begin{cases} Y_{(N+1)/2} & If\ N\ is\ Odd \\ \frac{1}{2}(Y_{\frac{N}{2}} + Y_{1+N/2}) & If\ N\ is\ even \end{cases}$$

In above equation there is no rationale behind choosing $Q = 1/2(Y_{n/2} + (Y_{1+N/2})$ in case of even number of processes. Since, no process has burst time between $Y_{n/2}$ and $Y_{1+n/2}$. Therefore it is worthless to select $Q = 1/2(Y_{n/2} + (Y_{1+N/2})$. Choosing this only increases quantum size without reducing number of context switches. Thus the major research question is: What happens if $Q = Y_{n/2}$ is selected as quantum for even number of processes? This research is focused on answering this research question.

3. OBJECTIVE

The main objectives of this research work are:

a. To modify the dynamic quantum selection policy in SARR algorithm such that quantum size can be reduced without increasing number of context switches and without hunting other performance parameters such as average waiting time, average turnaround time etc.
b. To compare SARR and improved SARR scheduling algorithms.

4. SARR SCHEDULING POLICY

Self-Adjusted Round Robin (SARR) is based on a dynamic quantum approach. The idea of this approach is to make the time quantum repeatedly adjusted according to the burst time of the now running processes. When a new process loaded to be executed the operating system tests the status of the specified program which can be either 1 or 0. When the status equals to zero this means that the process is either being executed for the first time or it has been modified or updated since the last analysis. In this case the operating system assign a counter to find the burst time of the process and continues executing the processes in the ready queue on the current round including the new arrival process using the current time quantum Q, otherwise and when status is equal to 1, then the operating system recalculates the time quantum Q depending on the remaining burst time of all ready processes including the new arrival process. It has found through the experience that the optimal time quantum can be presented by the median for the set of processes in the ready queue [4].

The pseudocode of this policy is given below:

- Loop Until ready queue! = null
- If new process pi arrived then
  – If status of pi status =0 then
  – Assign new counter Ci for this process
  – End if
- End if
- Calculate New_TQ by using all processes in ready queue with Status=1 as below

$$\text{New\_TQ}=\begin{cases} Y_{(N+1)/2} & If\ N\ is\ Odd \\ \frac{1}{2}(Y_{\frac{N}{2}}+Y_{1+N/2}) & If\ N\ is\ even \end{cases}$$

- For i=1 to n loop
  – Execute process pi for new time quantum (New_TQ)
  – If pi terminated normally and status of pi =0
  – Then Save counter ci of process pi
  – Set status of pi =1
  – End if
- End for
- Repeat

5. IMPROVED SARR POLICY

This is same as self-adjusted RR algorithm besides that its quantum selection strategy is slightly different in case of even number of processes. In this method time quantum is calculated using the median when the number of processes are odd and when the number of process are even then lower term of the median (i.e. $Y_{n/2}$) is selected as the quantum. The pseudocode of this policy is presented below:

- Loop Until ready queue! = null
- If new process pi arrived then
  – If status of pi status =0 then
  – Assign new counter Ci for this process
  – End if
- End if
- Calculate New_TQ by using all processes in ready queue with Status=1 as below

$$\text{New\_TQ}=\begin{cases} Y_{(N+1)/2} & If\ N\ is\ Odd \\ Y_{\frac{N}{2}} & If\ N\ is\ even \end{cases}$$

- For i=1 to n loop
  – Execute process pi for new time quantum (New_TQ)
  – If pi terminated normally and status of pi =0
  – Then Save counter ci of process pi
  – Set status of pi =1
  – End if
- End for
- Repeat

6. RELATED WORK:

A lot of attempts were done to find a solution for the minimal turnaround time, waiting time and to reduce overhead of extra context switches in round robin algorithm. Regardless of the different methodologies used in these attempts, Most of them are based on fixed time quantum and in recent years some of the works are also based on dynamic time quantum.

Self-Adjustment- Round-Robin (SARR) policy based on dynamic-time-quantum was designed to solve all problems associated with static quantum in a practical, simple and applicable manner [3]. In this research authors presents a solution to the time quantum problem by making the operating system adjusting the time quantum according to the burst time of the existing set of processes in the ready queue. In this paper the optimal time quantum can be presented by the median for the set of processes in the ready queue, if the median less than 25 then its value must be modified to 25 to avoid the overhead of the context switch.

Another scheduling algorithm called Improved Round Robin (IRR) orders processes and then finds optimal time quantum by using median based method [5]. Average Max Round Robin algorithm (AMRR) selects quantum by using mean of the summation of the average and maximum burst time [6]. Min Max Round Robin (MMRR) approach takes as the range of the CPU burst time of all the processes as quantum. If the min max time quantum is less than 25, then its value must be modified to 25 to avoid the overhead of the context switch [7].

Another approach of finding time quantum of Round Robin CPU scheduling algorithm in general computing system is by using integer programming. This approach solve equations that decides value that is neither to large nor too small such that every process has reasonable response time and the throughput of the system is not decreased due to reduction in unnecessarily context switches [8]. This method developed a changing the time quantum in each round over the cyclic queue. This method is called as Changeable Time Quantum (CTQ) techniques. Non-linear mathematical model can also be used for optimizing the time quantum value in RR scheduling algorithm [9].

Virtual Time Round-Robin (VTRR) combines fair queuing algorithms with the RR scheduling algorithm [10]. RR algorithm is optimized by using the concept of fuzzy rules [11] and the approach is named as Fuzzy Rule Based Round-Robin CPU Scheduling (FRRCS). This paper showed that the algorithm reduces the average waiting time. Genetic algorithm is also used to determine optimum time quantum value in Round Robin CPU scheduling algorithm to minimize the average waiting time of the processes [12].

7. EXPERIMENTAL FRAMEWORK:

To evaluate the algorithms considered in this research, three different test cases are designed. In all test cases number of processes ranges from 20-200 with interval size 20.

a. **Test Case I:** In this test case jobs with relatively small burst time is taken. Here burst time of jobs varies between 1 to100.
b. **Test Case II:** In this test case jobs with high burst time is taken, where burst time of jobs varies from 500 to 1000.
c. **Test Case III:** And in third test case jobs with average burst time is considered, where burst time varies from 100 to 500. All burst times are randomly generated numbers.

8. DATA COLLECTION AND ANALYSIS:

All data collected in this research is primary data. Burst time of processes generated for above mentioned test cases, which are given as input to the simulated SARR and improved SARR scheduling policy and result is collected from the traces generated by the algorithms.

Table 1: Result for Test Case I

	Awt		Atat	
No. of Processes	**SARR**	**ISARR**	**SARR**	**ISARR**
20	450.73	445.75	501	495.86
40	1032.9	1027.5	1087.8	1082.4
60	1268.3	1263.9	1316.1	1311.7
80	1936.2	1932.4	1987.5	1983.7
100	2371.3	2369.0	2421.8	2419.4
120	2965.5	2962.1	3017.4	3014.0
140	3298.4	3293	3348.5	3343.1
160	3787.5	3787.5	3838.3	3838.3
180	4256.5	4256.5	4306.9	4306.8
200	4637.5	4637.5	4687.2	4687.2

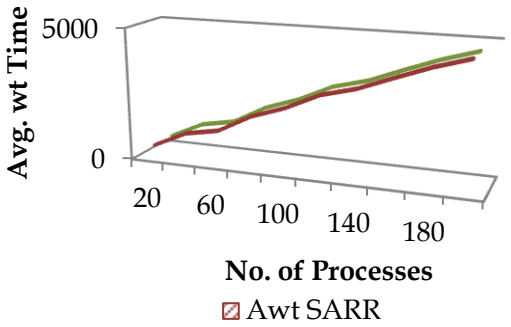

Figure 1: Graph for Average Waiting Time

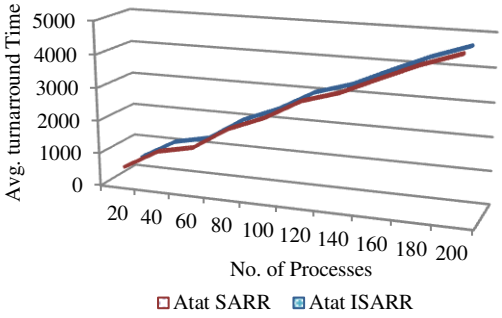

Figure 2: Graph for AverageTurnarround Time

From above graph it is clear that average waiting time of improved SARR is 1.1% lower and average turnaround time is 1% lower than SARR policy.

Table 2: Result for Test Case II

	Awt		Atat	
No. of Processes	**SARR**	**ISARR**	**SARR**	**ISARR**
20	7800.8	7755.1	8531.05	8485.4
40	166214	16605.1	17373.1	17356.8
60	24225.6	24220.1	24945.8	24940.3
80	33306.8	33280.4	34050.1	34023.7
100	41632.1	41603.7	42368.3	42339.9
120	50011.6	49969.2	50750.7	50708.3
140	59847.1	59833.5	60597.4	60583.8
160	69651.8	69621.1	70411.1	70380.3
180	76915.6	76860.9	77662.4	77607.7
200	87555.2	87527.3	88318.6	88290.6

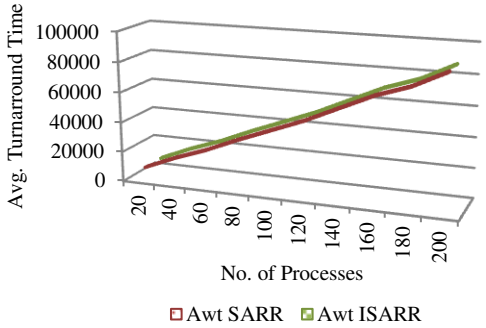

Figure 3: Graph for Average Waiting Time

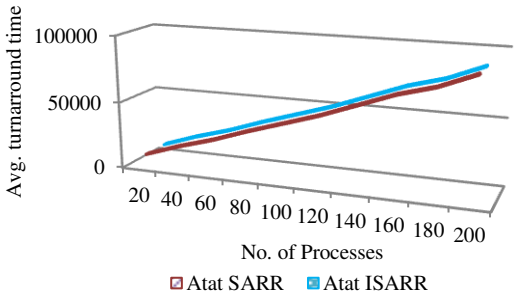

Figure 4: Graph for Average Turnarround Time

Above graph shows that average waiting time and average turnaround time of improved SARR scheduling policy is slightly better than SARR policy. Average waiting time increases up to 0.58% and average turnaround time increases up to 0.53%.

Table 3: Result for Test Case III

	Awt		Atat	
No. of Processes	SARR	ISARR	SARR	ISARR
20	3001.05	2997.95	3307	3303.9
40	5955	5945.5	6242.62	6233.12
60	9922.8	9875.96	10238.4	10191.5
80	11577.7	11573.9	11862.9	11859.1
100	14217.6	14150.3	14496.7	14429.4
120	18876.1	18847.6	19177.8	19149.3
140	21113.4	21067.8	21404.2	21358.5
160	25247.6	25232.4	25551.9	25536.7

180	29008.6	28954.1	29316.7	29262.1
200	33758.9	33715.8	34074.7	34031.6

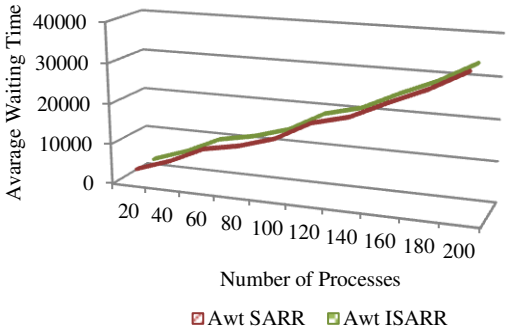

Figure 5: Graph for Average Waiting Time

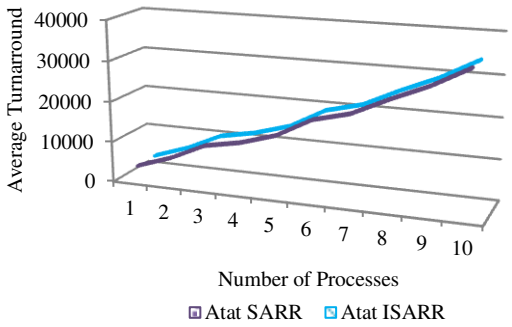

Figure 6: Graph for Average Turnarround Time

From above graph we can see that average turnaround time and average turnaround time of improved SARR increases up to 0.46% and 0.48% respectively.

9. CONCLUSION:

SARR scheduling policy selects $Q = 1/2(Y_{n/2} + (Y_{1+N/2})$ as quantum in case of even number of processes. This research modifies quantum selection strategy of SARR policy in case of even number of processes. The adopted strategy selects lower term of the median as quantum (i.e. $Q = y_{n/2}$)

This strategy gives same number of context switches with reduced quantum size, since there is no process that has burst time between $Y_{n/2}$ and $Y_{1+n/2}$. Reducing the quantum size without increasing context switch overhead is main achievement of this research. Besides this, this paper experimentally verifies that improved SARR policy beats SARR policy in many cases. Average waiting time and average turnaround time of improved SARR policy is better or at least not lower (in some cases) than SARR policy. Thus, this research is able to reduce the quantum size along with improving other performance parameters slightly.

REFERENCES:

1. A.S Tanenbaun (2008), Modern Operating Systems, 3rd Edition, Prentice Hall of India.
2. A. Ghishing (2009), Analysis of the Varying Time Quantum Round Robin Scheduling, Masters Thesis, CDCSIT TU.
3. R. J. Matarneh (2009), Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of the now Running Processes, American Journal of Applied Sciences.
4. H. S. Behera, R. Mohanty and D. Nayak (2010), A New Proposed Dynamic Quantum with Re-Adjusted Round Robin Scheduling Algorithm and its performance analysis, International Journal of computer Applications .
5. D. Nayak, S. K. Malla, and D. Debadarshini (2012), Improved Round Robin Scheduling using Dynamic Time Quantum, International Journal of Computer Applications.
6. P. Benerjee, and S.S. Dhal (2012), Comparitive performance analysis of Average Max Round Robin Scheduling Algorithm (AMRR) using Dynamic TimeQuantum with Round Robin Scheduling Algorithm using static Time Quantum, International Journal of Innovative Technology and Exploring Engineering (IJITEE).
7. S.K. Panda and S. K. Bhoi (2011), An Effective Round Robin Algorithm using Min Max Dispersion Measure, International Journal of Computer Science and Engineering.
8. S. M. Mostafa, S. Z. Rida, and H. S. Hamad (2010), Finding time quantum of Round Robin CPU scheduling in general computing systems using integer programming, International Journal of Research and Review in Applied Science.
9. S. Saeidi and H. A. Baktash (2011), Determining the Optimum Time Quantum Value in Round Robin Process Scheduling Method, Information Technology of Computer Science.
10. J. Nieh, Ch. Vaill and H. Zhong (2001), Virtual-Time Round-Robin: An O(1) Proportional Share Scheduler, Proceeding of the 2001 USENIX Annual Technical Conference, USA.
11. M. H. Zahedi, M. Ghazizade and M. Naghibzade (2008)., Fuzzy Round Robin CPU Scheduling (FRRCS) Algorithm, Advances in Computer and Information Sciences and Engineering.
12. M. U. Siregar (2012), A New Approach to CPU Scheduling Algorithm: Genetic Round Robin, International Journal of Computer Applications.