Science World Journal Vol. 17(No 2) 2022
www.scienceworldjournal.org
ISSN: 1597-6343 (Online), ISSN: 2756-391X (Print)
Published by Faculty of Science, Kaduna State University

AN IMPROVED HALF LIFE VARIABLE QUANTUM TIME WITH MEAN TIME SLICE ROUND ROBIN CPU SCHEDULING (IMHLVQTRR)

Ashiru Simon, *Gabriel Lazarus Dams, Salome Danjuma

Department of Computer Science, Kaduna State University, Tafawa Balewa Way, Kaduna, Nigeria

*Corresponding Author Email Address: damsgabe@kasu.edu.ng

ABSTRACT
Round Robin (RR) CPU scheduling is a scheduling technique that allocate equal time slice known as quantum time (QT) to processes wanting to use the CPU. Processes are allocated the CPU in a circular manner in such a way that if QT is greater than or equal to a process' burst time, the process will run to completion otherwise the process will be interrupted and return to the tail of the ready queue for next round of execution. The average waiting time and turnaround time in a classical RR is higher when compared with First Come First Serve and Shortest Job First CPU scheduling algorithms. The existing technique Half Life Variable Quantum Time Round Robin (HLVQTRR) further increases the average waiting time and average turnaround time for the system. Researchers proposed a dynamic QT in order to improve the classical RR which has a static QT. In dynamic RR, there are more than one QT used for allocating time slot to processes as opposed to classical RR where a fix QT is used for the allocation. This research work is a proposal that modified HLVQTRR to be 'An Improved Half Life Variable Quantum Time with Mean Time Slice Round Robin CPU Scheduling (ImHLVQTRR)'. In the proposed technique, two quantum time (QT1 and QT2) is calculated. QT1 is the average of all the processes in the ready queue and it is constant while QT2 is the half of each process burst and it changes depending on which process is in execution. The proposed approach was developed and simulated using python programming language. Using python programing language, the proposed approach was developed and the system was simulated and compared against the classical RR and HLVQTRR. The result showed that the proposed technique (ImHLVQTRR) minimized average waiting time, average turnaround time and number of context switching by 1292.087, 1292.089 and 27.40 time units respectively against the existing technique (HLVQTRR) and the classical RR.

Keywords: CPU Scheduling, Operating System, Dynamic Round Robin, Quantum Time, Waiting Time.

INTRODUCTION
One of the major resources used in the computer is the Central Processing Unit popularly known as CPU. There are multiple request from processes to use the CPU but it can only attend to each request one at a time. Since CPU is a scarce resource, processes must compete to acquire it. This limited resource is managed by the Operating system (OS). The main functions of an OS is resource allocation and program management. The execution of user programs is the OS major concern(Silberschatz et al., 2005). The user programs will change to processes as soon as they are loaded into the main memory for execution. It is the responsibility of the OS to decide the process that will make use the CPU through a technique known as CPU scheduling. The OS decides the order of execution through the scheduler (Khokhar and Kaushik, 2017). The scheduler uses some scheduling algorithms to allocate CPU to processes ready for execution. This is done by assigning time slot to respective processes is known as CPU scheduling (Mody and Mirkar, 2019). This technique is necessary in managing resource allocation in the system (Danjuma et al., 2021). CPU scheduling sometimes may be preemptive or non-preemptive. In a preemptive scheduling, a process may involuntarily relinquish its resource (CPU) before completion of its execution. This is possible as a result of timer interrupt or a process with a higher priority request to use the CPU. On the other hand, in a non-preemptive scheduling, a process relinquishes its resource (CPU) voluntarily not by force. A process may voluntarily release a resource if it run to completion or the process may be in need of a different resource such as input/output operation.

Types of CPU Scheduling Algorithms
Among the several types of CPU scheduling algorithms, the main types are briefly discussed in this section according to Omar et al. (2021), which are: First Come First Serve (FCFS), Shortest Job First (SJF), Priority Scheduling and Round Robin (RR). Each of these techniques have their advantages and disadvantages for processes allocation.
In FCFS algorithm, the first process that arrive will be assign the CPU and then the next process and so on. FCFS is very simple and easy to implement but it may increase the overall waiting time if processes that arrive earlier have larger CPU bursts. In the case of SJF algorithm, a process with the shortest burst time is executed first and then next and so on. This algorithm gives the best average waiting time when compare with other algorithms. However, SJF may lead to starvation. If processes with shorter bursts keep coming into the system, the processes with larger bursts will have to wait indefinitely. In priority scheduling, processes are executed base on their priority levels. The one with the highest priority is executed first and then the next once and so on. As for Round Robin CPU scheduling, processes are allocated the CPU in a circular manner such that each process is given equal time slot known as quantum time (QT) for execution. If QT is greater than or equal to a process' burst, the process will run to completion otherwise the process will be interrupted and return to the tail of the ready queue for next round of execution. One of the most frequently used CPU scheduling algorithms is RR (Paul et al., 2019). The advantage of RR is the equal time slice given to each process in the system. The disadvantage of RR is the cost of context switching added to the system since some process that did not complete their execution will have to return for some subsequent rounds of execution. As a result of this, the average waiting time and turnaround time in a classical RR is always higher when compared with First Come First Serve and Shortest Job First CPU scheduling algorithms (Ashiru et al., 2014a).

Round Robin can be static or dynamic. A static RR uses a fixed single quantum time (QT) for all processes in the ready queue and in all rounds(Zouaoui et al., 2019). Fixed QT has some disadvantages (Fiad et al., 2020). Processes whose burst time are just small enough larger than QT will have return for execution in the next round. In dynamic RR, there are more than one QT use for allocating time slot to processes. Dynamic QT allocation technique is more efficient than static QT allocation technique (Alazzam et al., 2019). Using some mathematical and logical problem solving technique variable QT can be obtain and use for CPU allocation.

Process States

A process is a program that is already executing. A process is defined as an active program (Ashiru et al., 2014b). Once a program is loaded into the memory for execution, it automatically changes to a process. This process can be in five different state in its life cycle – new, ready, running, terminated or waiting. If a process is just created, it will be in a new state. If it is waiting for the CPU, that process will be in a ready state. A process is in running state if it is currently executing. When a running process finished executing, it will be in a terminated state. A process that is using any other resource such as input/output device other than the CPU will be in a waiting state. Figure 1 is a diagram representing process states.

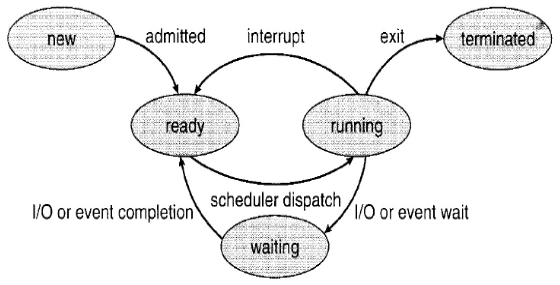


Figure 1: process state diagram
Source (Silberschatz et al., 2005)

Evaluation Parameters

The criteria used for evaluation are average waiting time (AWT), average turnaround time (ATAT) and number of context switching CS).

- **Waiting Time**: This is the total time taken by a process in the ready waiting for the CPU. This criteria should be minimized always. The lower the waiting, the better the performance of the system.
- **Turnaround Time**: It is the overall time taken by a process to finish executing. It is the time taken for a process waiting in the ready queue plus the time it takes executing. This criteria should be minimized.
- **Context Switching**: This is the time taking switching the CPU from one process to another. The CPU is idle at the point of context switching. This is pure overhead which need to be minimized.

Review of Related Literatures

Various research works on how to apply variable quantum time to RR CPU scheduling had been proposed. This is because the classical RR has a fix QT that cannot change throughout process execution. With dynamic QT, average waiting time, average turnaround time and number of context switching can be minimized. (Ashiru et al., 2014b) proposed a dynamic variable quantum time in which processes are allocated half of their QT in their first round. Each process remaining burst time will be used as the QT for each in the next round. In this approach, processes go for two rounds before it finish executing. For all process $P_i$with burst time, BT_i, QT will be$^{BT_i}/_2$. (Sohrawordi et al., 2019) proposed a dynamic RR where QT is the average of processes' bursts in the ready queue. The processes are sorted in ascending order such that the process with the smallest burst is executed first and then the next and so on. Given a process P where $P_i = \{P_1, P_2, P_3, \ldots, P_n\}$. The QT=$AVG(\sum_{i=0}^{n} P_i)$. (Khokhar and Kaushik, 2017) proposed a dynamic RR where QT is the average between the median and the mean of processes' bursts. Processes are arranged in an increasing order such that the process with smallest burst time are executed first and then followed by the next and so on. It is calculated as, QT= (mean + median)/2. (Mody and Mirkar, 2019) proposed a RR technique where CPU are allocated to processes dynamically using two factors – Smart Time Quantum (STQ) and Delta. STQ is the average difference between adjacent processes' bursts in the ready queue while Delta is (STQ)/2. The QT is given as STQ + Delta. (Paul et al., 2019) proposed variable quantum for RR CPU scheduling using three different QT under various scenario. In the first case, if the number of process in the ready queue is less than or equal to 4 then the average of processes' bursts will be used as their QT. That is QT=$AVG(\sum_{i=0}^{n} BT_i)$. In another case, if the number of processes is even, then the QT will be the average between the first process, the last process, the median process and the process just after the median process. That is, QT=$AVG\left(BT_i, BT_N, BT_{\frac{N}{2}}, BT_{\frac{N}{2}+1}\right)$. Also, in a situation where the number of process in the ready queue are odd, the QT will be the average between the first process, the last process, the process just after the median process and the process just before the media process. That is, QT=$AVG(BT_i, BT_N, BT_{\frac{N}{2}-1}, BT_{\frac{N}{2}+1})$. (Biswas et al., 2018) proposed a dynamic RR CPU scheduling where QT is the sum of the maximum difference between adjacent processes' bursts and the process with the least burst time. That is QT=$MAX + BT[0]$, where MAX= $(MAX, diff)$ and $diff = BT[i+1] - BT[i]$. Processes are sorted in ascending order and are executed from the process with the least burst time up to the process with the largest burst time.

Description of the Existing Technique (HLVQTRR)

Half Life Variable Quantum Time Round Robin (HLVQTRR) is a proposed RR CPU scheduling algorithm by (Ashiru et al., 2014b) where QT is calculated and allocated to processes dynamically. The QT is a variable one in which half of each process' burst was used as its time slice. That is, if a process P_i has a burst timeBT_i, its QT will be $\frac{BT_i}{2}$. For example, if the burst time of process P_i is 20 time unit, the QT of process P_i will be 20/2=10 time unit. This proposed technique ensured that each process must go for two rounds to complete its execution. The system will incurred the cost of performing context switching and CS is a pure system overhead because the CPU is idle at that time (Mishra and Rashid, 2014). This will lead to poor average waiting time and average turnaround time. The proposed technique will improve the average waiting time, average turnaround time and number of context switching.

Description of the Proposed Technique (IMHLVQTRR)
The proposed technique is a direct improvement of HLVQTRR CPU scheduling technique. In this research work, variable quantum is used when allocating time slice to processes in the ready queue. It assume that all processes are already waiting in ready to be assign CPU for execution. If the burst time of a process is less than or equal to the average burst time for all the processes in the ready queue, the QT for that process will be the average burst time for all the processes in the ready queue. Otherwise, if the process burst time is greater than the average burst time for all the processes, the QT will be half of its burst time. This technique just as the existing technique ensures that process execution cannot go beyond two rounds. In the first round, a process may run to completion or may execute half of its burst time while in the second round, those that executed half of their burst time will run to completion. That is, the system has two QT. the first QT (QT1) is the average burst of the processes in the ready queue and the second QT (QT2) is half of the burst time of each process. If the burst of a process is greater than half of the average burst time for all the processes in the ready queue, the QT for that process will be QT2 while those processes whose burst time are less or equal to half of the burst time of all the processes in the ready queue, their QT will be QT1. Those processes that will go for the second round, their burst time will automatically be adjusted to be their QT. For example, consider a set of processes P_i such that $P_i = \{P_1, P_2, P_3, \dots, P_n\}$ with their respective burst time$\{BT_1, BT_2, BT_3, \dots, BT_n\}$. For instance, if the average bursts $(BT_1\ to\ BT_n)$for the processes $P_i = \{P_1, P_2, P_3, \dots, P_n\}$is 45 and process P_1 and P_2 have their burst time to be 20 and 50 respectively. The QT for P_1 will be 45 while that of P_2 will be$\frac{BT_2}{2} = \frac{50}{2} = 25.$
The flowchart for the proposed technique (ImHLVQTRR) is shown in figure 2.

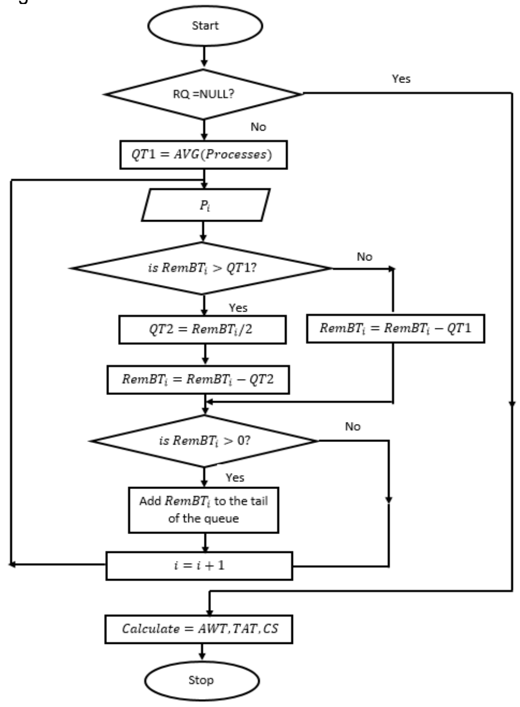


Figure 2: Flowchart for the proposed technique (ImHLVQTRR)

Illustration/Demonstration
Consider a set of five processes P1, P2, P3, P4, and P5 with their respective burst times 20, 45, 13, 65, 28 and 32 as shown in the table 1.

The Classical RR
Using the classical RR, the average burst time for all the processes in the ready queue is used as the QT. According to table 1, QT = (20+45+13+65+28+32)/6=34. This value is used to assign time slot to processes in the ready queue in each rounds.

Table 1: Rounds for Classical RR technique

Process	Round One		Round Two	
	Burst Time	Remaining Burst Time	Burst Time	Remaining Burst Time
P1	20	0		
P2	45	45-34=11	11	0
P3	13	0		
P4	65	65-34=31	31	0
P5	28	0		
P6	32	0		
	QT=Average =203/6=34			

Using table 1, the Gantt chart for the existing technique is generated as shown in chart I.

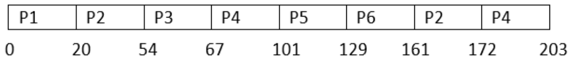


Chart I: Gantt chart for Classical RR

From chart I, the average waiting time, average turnaround time and number of context switching is calculated.
P1: 0, P2: 20 + (161-54) =127, P3: 54, P4: 67 + (172-101) =138, P5: 101, P6: 129
AWT= (0+127+54+138+101+129)/6 = 549/6=91.50
AWT= (0+20) + (45+127) + (13+54) + (65+138) + (28+101) + (32+129) = 710/6=118.33
Number of CS = 8

The Existing Technique (HLVQTRR)
Using the existing technique, half of each process burst is executed in the first round while the remaining burst will be executed in the second round with the assumption that Quantum Time (QT) is a whole number for ease of computation.

Table 2: Rounds for the existing technique (HLVQTRR)

Process	Round One			Round Two		
	Burst Time (BT_1)	Quantum Time ($QT_1 = BT_1/2$)	Remaining Burst Time ($RBT_1 = BT_1\text{-}QT_1$)	$BT_2 = RBT_2$	$QT_2 = BT_2$	RBT
P1	20	10	20-10=10	10	10	0
P2	45	22	45-22=23	23	23	0
P3	13	6	13-6=7	7	7	0
P4	65	32	65-32=33	33	33	0
P5	28	14	28-14=14	14	14	0
P6	32	16	32-16=16	16	16	0

Using table 2, the Gantt chart for the existing technique is generated as shown in chart II.

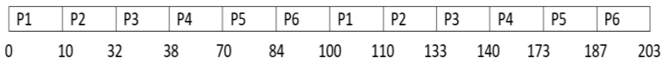


Chart II: Gantt chart for HLVQTRR

From chart II, we can calculate the average waiting time, average turnaround time and number of context switching.

P1: 0+ (100-10)=90, P2: 10+ (110-32)=88, P3: 32+(133-38)=127, P4: 38+(140-70)=108,
P5: 70+(173-84)=159, P6: 84+ (187-100) =171
AWT= (90+88+127+108+159+171)/6 = 743/6=123.83
AWT= (20+90) + (45+88)+ (13+127)+ (65+108) +(28+159)+ (32+171) = 946/6=157.66
Number of CS = 12

The Proposed Technique (ImHLVQTRR)
In the proposed technique, two quantum time (QT1 and QT2) is calculated. QT1 is the average of all the processes in the ready queue and it is constant while QT2 is the half of each process burst and it changes depending on which process is in execution. If process P_i is assigned the CPU, the scheduler will allocated time slot of QT1 to P_i if its burst time is greater than QT1, otherwise the system will allocate QT2 as the time slot for process P_i. That is, if the burst time of process P_i is less than or equal to the average burst time for all the processes in the ready queue, the process will execute half of its burst time only, otherwise it will use the average burst time for all the processes in the ready queue as its quantum time.

Table 3: Rounds for the proposed technique (ImHLVQTRR)

	Round One				Round Two		
Process	Burst Time	QT2	QT Used for P_i	Remaining Burst Time	Burst Time	QT Used for P_i	Remaining Burst Time
P1	20	20/2=10	34 (QT2)	0			
P2	45	45/2=22	22 (QT1)	45-22=23	23	23	0
P3	13	13/2=6	34 (QT2)	0			
P4	65	65/2=32	32 (QT1)	65-32=33	33	33	0
P5	28	28/2=14	34 (QT2)	0			
P6	32	32/2=16	34 (QT2)	0			
	QT_1= 203/6=34						

As shown in table 3, process P1, P3, P5, and P6 run to completion because their burst time is less than average burst time for the processes in the ready queue. Also, as it can be seen in, process P2 and P4 returned for the second round for execution since their burst time is greater than the average burst time for all the processes in the ready queue. The Gantt chart is represented in chart III.

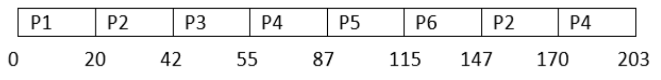
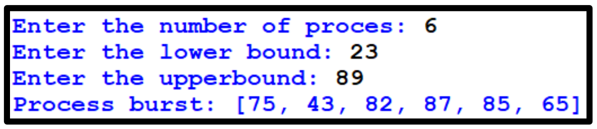


Chart III: Gantt chart for ImHLVQTRR

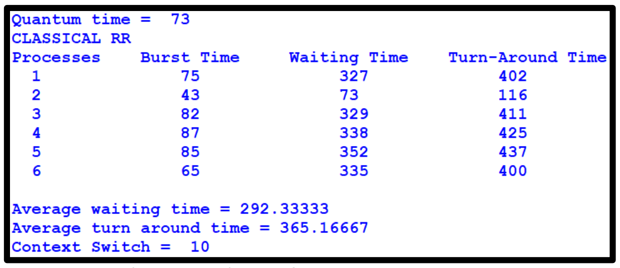
Using chart III, the average waiting time, average turnaround time and number of context switching can be calculated as:
P1: 0, P2: 20 + (147-42) =125, P3: 42, P4: 55 + (170-87) =138, P5: 87, P6: 115
AWT= (0+125+42+138+87+115)/6 = 507/6=84.50
AWT= (0+20) + (45+125) + (13+42) + (65+138) + (28+87) + (32+115) = 710/6=118.33
Number of CS = 8

PRESENTATION AND DISCUSSION OF RESULTS



```
Enter the number of proces: 6
Enter the lower bound: 23
Enter the upperbound: 89
Process burst: [75, 43, 82, 87, 85, 65]
```

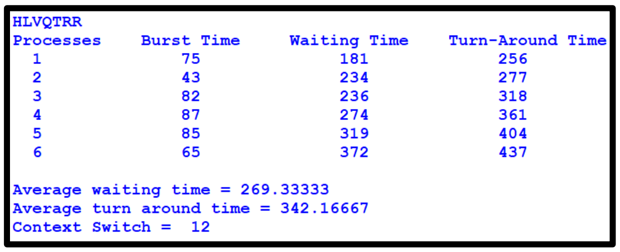
Figure 3: Inputs and randomly generated processes bursts times



```
Quantum time =  73
CLASSICAL RR
Processes     Burst Time     Waiting Time     Turn-Around Time
  1               75             327              402
  2               43              73              116
  3               82             329              411
  4               87             338              425
  5               85             352              437
  6               65             335              400

Average waiting time = 292.33333
Average turn around time = 365.16667
Context Switch =  10
```

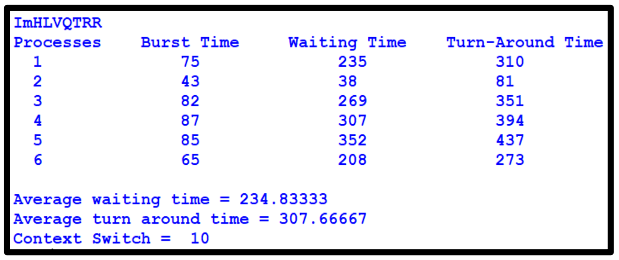
Figure 4: Interface result for the Classical RR



```
HLVQTRR
Processes     Burst Time     Waiting Time     Turn-Around Time
  1               75             181              256
  2               43             234              277
  3               82             236              318
  4               87             274              361
  5               85             319              404
  6               65             372              437

Average waiting time = 269.33333
Average turn around time = 342.16667
Context Switch =  12
```

Figure 5: Interface result for the HLVQTRR



```
ImHLVQTRR
Processes     Burst Time     Waiting Time     Turn-Around Time
  1               75             235              310
  2               43              38               81
  3               82             269              351
  4               87             307              394
  5               85             352              437
  6               65             208              273

Average waiting time = 234.83333
Average turn around time = 307.66667
Context Switch =  10
```

Figure 6: Interface result for the ImHLVQTRR

Figure 3 shows the burst time generated for 6 processes using number of processes, lower bound and upper bound as the input parameters. The interfaces in figure 4, figure 5 and figure 6 shows the results computed for classical RR, HLVQTRR and ImHVQTRR respectively. From the results, it is clear that the proposed technique (ImHLVQTRR) minimizes average waiting time, average turnaround time and number of context switching when compared with the existing technique (HLVQTRR). In the same vein, though the number of context switching is the same for the proposed technique and the classical RR yet the proposed technique provides better average waiting time and average turnaround time compared to the classical RR.

SIMULATION OUTPUT BETWEEN CLASSICAL RR, HLVQTRR and IMHLVQTRR

Table 4: Outputs of Simulation between Classical RR, HLVQTRR and ImHLVQTRR

nP	LB	UB	Classical RR			HLVQTRR			ImHLVQTRR		
			AWT	ATAT	CS	AWT	ATAT	CS	AWT	ATAT	CS
10	6	76	220.80	264.9	15	291.0	335.1	20	202.8	246.9	15
20	12	98	830.65	891.25	31	872.0	932.6	40	700.65	761.25	31
30	23	90	1073.7	1129.96	45	1221.26	1277.53	60	941.7	997.96	45
40	32	99	1917.62	1986.65	60	2039.02	2108.05	80	1640.65	1709.67	60
50	7	67	1123.74	1160.26	74	1323.3	1359.82	100	995.72	1032.24	74
60	9	123	2209.55	2271.3	90	2688.16	2749.91	120	1988.41	2050.16	90
70	32	243	6200.27	6340.74	107	7181.44	7321.91	140	5477.31	5617.78	107
80	40	304	8876.88	9045.31	121	10050.60	10219.02	160	7794.36	7962.78	121
90	33	321	10783.93	10966.06	137	12196.55	12378.68	180	9453.62	9635.75	137
100	53	432	14211.41	14439.27	146	17064.48	17292.34	200	12811.72	13039.58	146
Total Average			**4744.855**	**4849.57**	**82.6**	**5492.781**	**5597.496**	**110**	**4200.694**	**4305.407**	**82.6**

Table 4 shows the simulated result between Classical RR, HLVQTRR and ImHLVQTRR on the performance of the algorithms in terms of average waiting time (AWT), average turnaround time (ATAT) and the number of context switching (CS). In this table, nP represents number of process, LB represents lower bound while UB is upper bound. When the program is executed, it will require three parameters – nP, LB and UB to generate random variables between LB and UB equal to the number of processes entered. For example, nP=10, LB=6 and UB=76 will generate 10 random variables which are between 6 and 76. In each of these cases, AWT, ATAT and CS is calculated for all the three algorithms. From table 4, after 10 runs, the average total for the existing technique (HLVQTRR) in terms of AWT, ATAT and CS are 5492.781, 5597.496 and 110 time units while for the proposed technique (ImHLVQTRR) are 4200.694, 4305.407 and 82.6 time units. By taking the difference, the results show that the proposed technique (ImHLVQTRR) minimizes AWT, ATAT and CS by 1292.087, 1292.089 and 27.40 time units respectively when compared with the existing technique (HLVQTRR). As for the classical RR, the proposed technique provides better AWT and ATAT than the Classical RR even though the number of context switching is the same for both algorithms. The graph for the results are plotted and shown in figure 7, 8 and 9

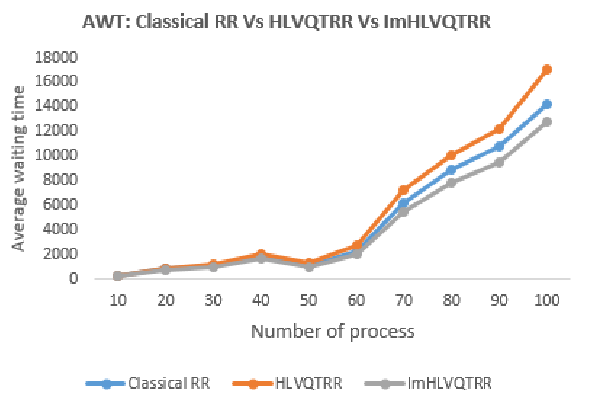


Figure 7: AWT: Classical RR Vs. HLVQTRR and ImHLVQTRR

Figure 7 shows the average waiting time between Classical RR Vs. HLVQTRR and ImHLVQTRR. In this presentation, the proposed technique provide the minimum average waiting time and then followed by the classical RR. HLVQTRR provide the worst waiting time.

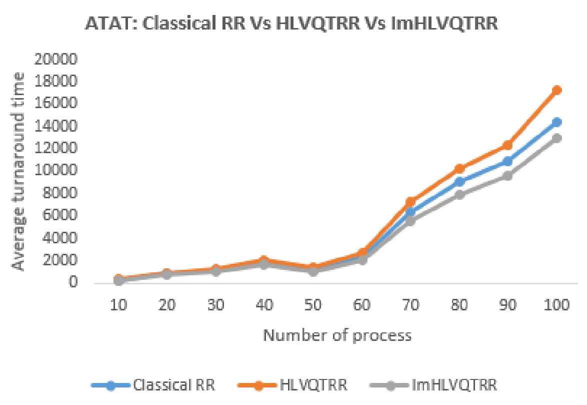


Figure 8: ATAT: Classical RR Vs. HLVQTRR and ImHLVQTRR

Figure 8 shows the result that compare the average turnaround time for the simulation between Classical RR, HLVQTRR and the proposed techniques (ImHLVQTRR). From this graph, the proposed technique gives the minimum average turnaround time and then followed by the classical RR. HLVQTRR produces the worst turnaround time.

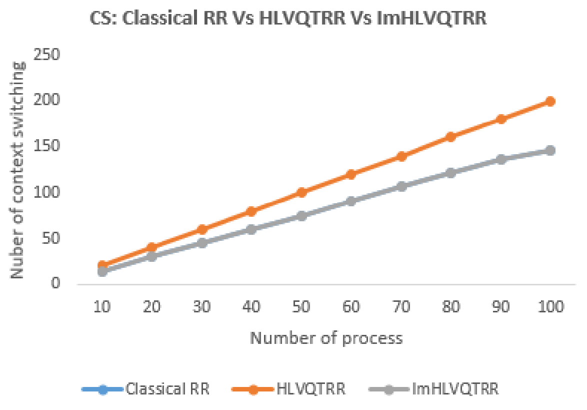


Figure 9: CS: Classical RR Vs. HLVQTRR and ImHLVQTRR

From Figure 9, the number of context switching is the same between the proposed technique (ImHLVQTRR) and the classical RR while HLVQTRR produces the worst context switching

Conclusion

The target of CPU scheduling algorithms is to minimize average waiting time, average turnaround time and number of context switching. The existing technique (HLVQTRR) proposed by (Ashiru et al., 2014b) further increased the average waiting time, average turnaround time and number of context switching as shown in table 4. In order to improve the existing technique, a modified version of HLVQTRR known as 'An Improved Half Life Variable Quantum Time with Mean Time Slice Round Robin CPU Scheduling (ImHLVQTRR)'was proposed. The simulation results showed that the average waiting time and average turnaround time is greatly improved when compared with the classical RR and the existing technique (HLVQTRR). More so, concerning the number of context switching, the proposed technique and the classical RR have equal number of switching. As for the existing technique regarding context switching, the proposed method minimized number of context switching more than the existing technique. This algorithm showed great promise in CPU scheduling.

REFERENCES

ALAZZAM, H., ALHENAWI, E. & AL-SAYYED, R. 2019. A Hybrid Job Scheduling Algorithm based on Tabu and Harmony Search Algorithms. *The Journal of Supercomputing*.

ASHIRU, S., ABDULLAHI, S. & SAHALU, J. 2014a. Dynamic Round Robin with Controlled Preemption (DRRCP) *International Journal of Computer Science Issues,* 11**,** pp.109-117.

ASHIRU, S., SALLEH, A. & SAHALU, J. 2014b. Half Life Variable Quantum Time Round Robin (HLVQTRR). *International Journal of Computer Science and Information Technologies,* 5**,** 7210-7217.

BISWAS, D., SAHA, S., FAISAL, R. H. & SAMSUDDOHA, M. 2018. An Improved Round Robin Scheduling Algorithm Based on Maximum Difference of Two Adjacent Processes. *Barishal University Journal Part 1,* 5 pp. 257-271.

DANJUMA, S., DAMS, G. L. & SIMON, A. 2021. Proposed Approach for Resource Allocation Management in Service Oriented Architecture (SOA) Environment. *Science World Journal* 16**,** pp. 357-362.

FIAD, A., MAAZA, Z. M. & BENDOUKHA, H. 2020. Improved Version of Round Robin Scheduling Algorithm Based on Analytic Model. *International Journal of Networked and Distributed Computing,* 8**,** pp. 195–202.

KHOKHAR, D. & KAUSHIK, A. 2017. Best Time Quantum Round Robin CPU Scheduling Algorithm. *International Journal of Scientific Engineering and Applied Science (IJSEAS),* 3**,** pp. 213-217.

MISHRA, M. K. & RASHID, F. 2014. An Improved Round Robin CPU Scheduling Algorithm with Varying Time Quantum. *International Journal of Computer Science, Engineering and Applications,* 4.

MODY, S. & MIRKAR, S. 2019. Smart Round Robin CPU Scheduling Algorithm For Operating Systems. *4th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT).*

*OMAR, H. K., JIHAD, K. H. & HUSSEIN, S. F. (2021). Comparative analysis of the essential CPU scheduling algorithms. Bulletin of Electrical Engineering and Informatics, Vol. 10(5)***,** *pp. 2742 - 2750.doi: 10.11591/eei.v10i5.2812*

PAUL, T., FAISAL, R. H. & SAMSUDDOHA, M. 2019. Improved Round Robin Scheduling Algorithm with Progressive Time Quantum. *International Journal of Computer Applications,* 178**,** pp. 30-36.

SILBERSCHATZ, A., GALVIN, P. B. & GAGNE, G. 2005. *Operating Systems Concepts,* USA, John Wiley and Sons.

SOHRAWORDI, M., ALI, U. A. M. E., UDDIN, M. P. & HOSSAIN, M. M. 2019. A Modified Round Robin CPU Scheduling Algorithm with Dynamic Time Quantum. *International Journal of Advanced Research,* 7**,** pp. 422-429.

ZOUAOUI, S., BOUSSAID, L. & MTIBAA, A. 2019. Improved Time Quantum Length Estimation for Round Robin Scheduling Algorithm using Neural Network. *Indonesian Journal of Electrical Engineering and Informatics (IJEEI),* 7 pp. 190-202.