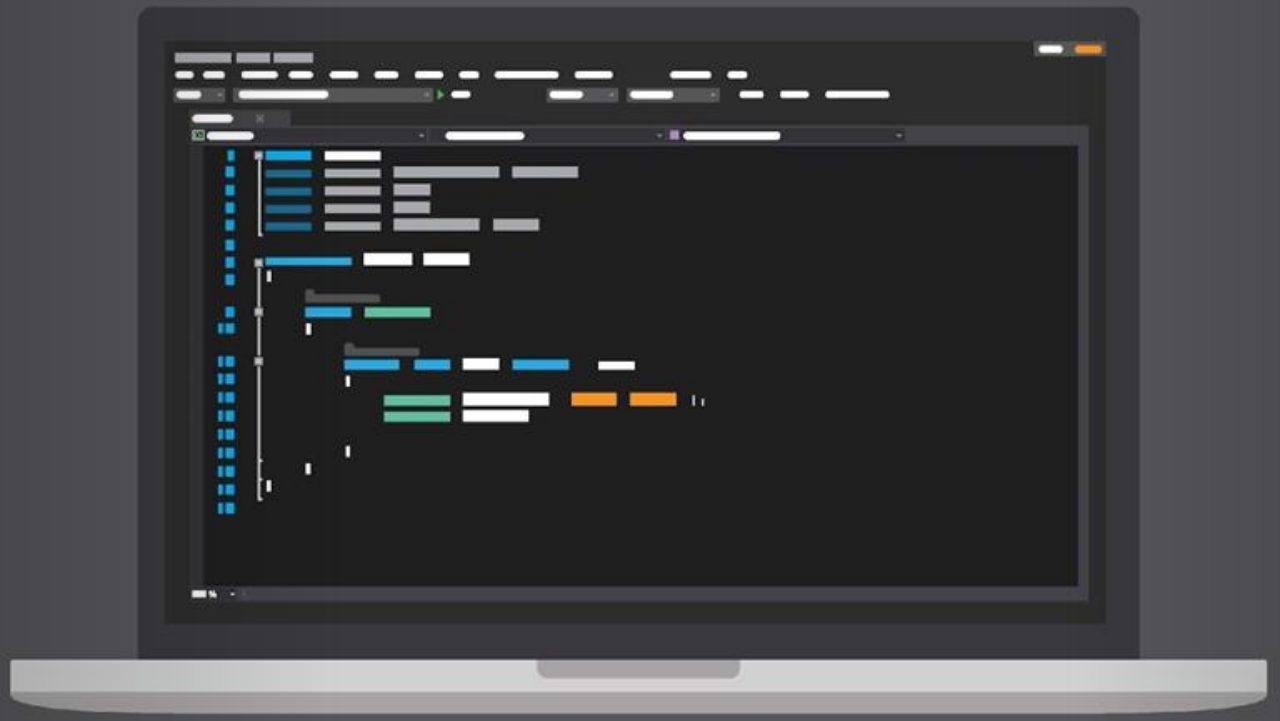
IMPROVED ROUND ROBIN-BASED SCHEDULING ALGORITHM



TECHNIQUES TO IMPROVE LINUX PROCESS SCHEDULING

AYUSH MANGUKIA, MOHAMMED IBRAHIM,
SOORYA GOLAMUDI, NISHANT KUMAR

INTRODUCTION

CPU or process Scheduling, which is an important part in any operating systems, allocates processes to the CPU in specific order to optimize some objective functions.

The efficiency of any operating system relies strongly on the scheduling algorithms used. Several scheduling algorithms exists. Among them, Round Robin (RR) is the most widely utilized algorithm.

RR has proved to be effective in several types of operating systems, such as time-sharing systems. This is due to the reasonable response time it gives.

However, it suffers from some shortcomings such as high average turnaround time, high average waiting time.

So, we devise an improved Round – Robin Algorithm.

LITERATURE SURVEY

You have the classical methods like First Come First Serve (FCFS), Shortest Job First (SJF), Shortest Remaining Time, Priority Scheduling and Round Robin Scheduling. The problem with FCFS is waiting time is too long if a big process comes in the Queue. Similarly, SJF causes starvation if short processes keep coming. Even Priority causes starvation when important processes keep coming and lower priority processes may be postponed indefinitely.

Even the most famous Round Robin Scheduling Algorithm has the drawback of causing high waiting time for each process if the Quantum is set too small. The quantum is set forehand and is generally not changed. There are many papers which have some improvements on the Round Robin Algorithm. One such improvement accredited by many Authors was to check if the Remaining Time of the process under execution is less than the Time Quantum, then the processor is re-assigned to the process for the remaining time. There was another Improvement which spoke about setting the time value of Time Quantum according to the Data Set. It used the approach of setting the Quantum value as the Twice the Average of Burst Times i.e. Two Times the Mean or set the Quantum Value as the middle most burst time in the Data Set i.e. Median.

Algorithm 1: Pseudo-Code

Result: Round Robin

1.Assign New Processes to Ready Queue

2.Initialize Quantum Value by calculating Mean of All Burst Time

3.**while** *all processes not completed* **do**

    Execute The First Process in ready_queue with calculated Time_Quantum;

    Calculate Remaining Time of Current Process;

    Send Process to end of Ready Queue;

    Go to Step 3;

end

4.Calculate Average_Waiting_Time, Average_Turn Around_Time, Average_Response_Time, Context_Switches

5.End

LITERATURE SURVEY THE ORIGINAL ROUND ROBIN ALGORITHM

THE PROPOSED ROUND ROBIN ALGORITHM

Algorithm 1: Pseudo-Code

Result: Improved Round Robin

1.Assign New Processes to Ready Queue

Quantum_Percentile = 0.85

2.Rearrange all Processes in Increasing Order of Burst Time

3.Initialize Quantum Value

4.**while** *all processes not completed* **do**

- Execute The First Process in ready_queue with calculated Time_Quantum;
- Calculate Remaining Time of Current Process;
- **if** *remaining_time* < = *Time_Quantum* **then**
  - Re-allocate CPU to Current Process for the remaining_time;
- **else**
  - Send Process to end of Ready Queue;
  - Go to Step 4;
- **end**
- **if** *New Process arrived* **then**
  - Sum_Burst = Sum of all Processes Burst Time;
  - Average_Burst = Sum_Burst / No. of Processes;
  - Time_Quantum = Quantum_Percentile x Average_Burst;
  - Go to Step 1;
- **else**
- **end**
- Select Next Process in Ready Queue;

end

5.Calculate Average Waiting Time, Average Turn Around Time, Average Response Time, Context Switches

6.End

COMPARISON (GANTT CHART) REGULAR ROUND ROBIN

Normal Round Robin:

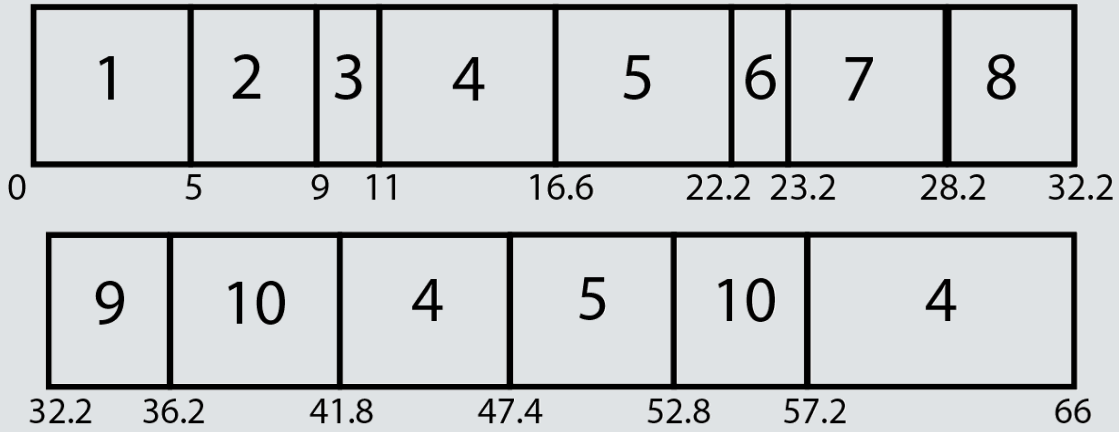
Quantum : 5.61 Seconds

Average Response Time: 14.771

Average Waiting Time: 21.893

Average Turn-Around Time: 28.493

Context Switches: 13



Process	Arrival Time	Burst Time
1	0	5
2	0	4
3	1	2
4	1	20
5	3	11
6	4	1
7	4	5
8	7	4
9	7	4
10	9	10

COMPARISON (GANTT CHART) IMPROVED ROUND ROBIN

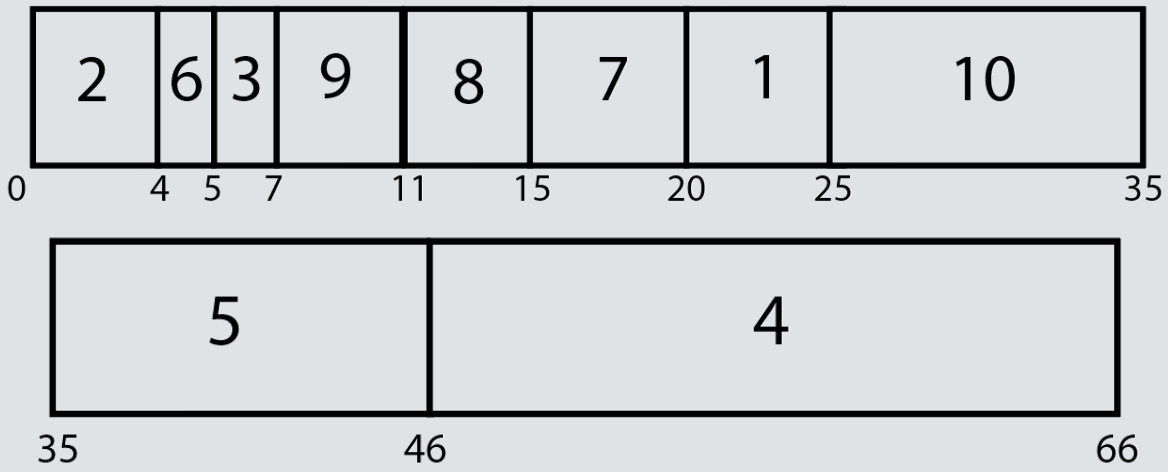
Improved Round Robin:

Average Response Time: 13.200

Average Waiting Time: 13.200

Average Turn-Around Time: 19.800

Context Switches: 9



Process	Arrival Time	Burst Time
1	0	5
2	0	4
3	1	2
4	1	20
5	3	11
6	4	1
7	4	5
8	7	4
9	7	4
10	9	10

IMPROVEMENTS WITH THE ALGORITHM

1. Using the Algorithm helps in reducing Average Waiting, Turn Around and Response Time as the Quantum Time is not fixed and is variable depending on the Tasks given to it.

2. Reduces number of context switches since the CPU is re-allocated to the current process if the remaining time is less than the time quantum.

RESULTS FOUND

For Dataset of Size 50, we find that a Quantum of Range 0.7 to 0.95 Produce Low Average Waiting and Turnaround Time with Context Switches almost equal to Size

For Dataset of Size 100, we find that a Quantum of Range 0.8 to 0.95 Produce Low Average Waiting and Turnaround Time with Context Switches almost equal to Size

For Dataset of Size 150, we find that a Quantum of Range 0.75 to 0.95 Produce Low Average Waiting and Turnaround Time with Context Switches almost equal to Size

For Dataset of Size 200, we find that a Quantum of Range 0.8 to 0.95 Produce Low Average Waiting and Turnaround Time with Context Switches almost equal to Size

RESULTS FOR QUANTUM VALUES

Quantum	N	AWT	ART	ATT	#CS
0.1	50	165.216	17.461	171.276	439
0.2	50	144.522	31.109	150.581	198
0.3	50	126.380	47.674	132.440	124
0.4	50	117.690	58.606	123.750	97
0.5	50	111.779	66.769	117.839	79
0.6	50	101.700	81.360	107.760	64
0.7	50	97.410	87.870	103.470	59
0.8	50	95.408	90.668	101.468	56
0.85	50	93.814	92.374	99.874	53
0.9	50	93.120	93.120	99.180	49

A NEW CONCEPT – PARALLEL PROCESSING INTO ROUND ROBIN

- Uses several systems or processors to run different processes in parallel
- Significantly lower values of time metrics and context switching even with 2/4 processors
- Better management of workload for the individual processors compared to regular parallel processing
- Better turn around times for the processes also compared to regular parallel processing

ROUND ROBIN WITH PARALLEL PROCESSING

Algorithm 1: Pseudo-Code

Result: Improved Round Robin with Parallel Processing

1.Assign New Processes to Ready Queue

Quantum_Percentile = 0.85

2.Rearrange all Processes in Increasing Order of Burst Time

3.Initialize Quantum Value

4.**while** *all processes not completed* **do**

    Select the Next Available Processor;

    Execute The First Process in ready_queue with calculated Time_Quantum using Selected Processor;

    Calculate Remaining Time of Current Process;

    if *remaining_time* < = *Time_Quantum* **then**

        Re-allocate CPU to Current Process for the remaining_time;

    else

        Send Process to end of Ready Queue;

        Go to Step 4;

    end

    if *New Process arrived* **then**

        Sum_Burst = Sum of all Processes Burst Time;

        Average_Burst = Sum_Burst/ No. of Processes;

        Time_Quantum = 0.85 x Average_Burst;

        Go to Step 1;

    else

    end

    Select Next Process in Ready Queue;

end

5.Calculate Average_Waiting_Time, Average_Turn Around_Time, Average_Response_Time, Context_Switches

6.End

LITERATURE SURVEY COMPARISON (GANTT CHART) IMPROVED ROUND ROBIN (2 CORES)

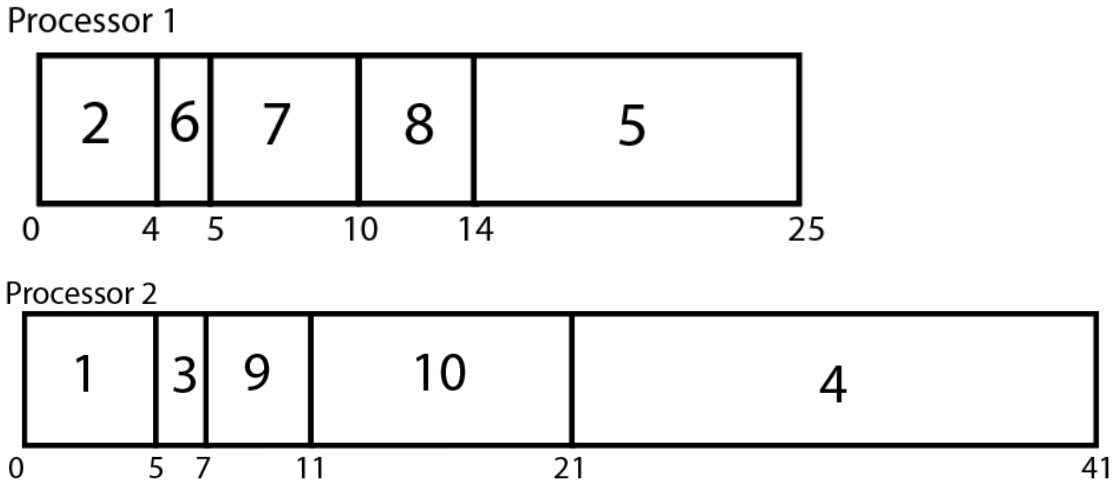
Improved RR (2 cores):

Average Response Time: 4.100

Average Waiting Time: 4.100

Average Turn-Around Time: 10.700

Context Switches: 8



Process	Arrival Time	Burst Time
1	0	5
2	0	4
3	1	2
4	1	20
5	3	11
6	4	1
7	4	5
8	7	4
9	7	4
10	9	10

LITERATURE SURVEY COMPARISON (GANTT CHART) IMPROVED ROUND ROBIN (4 CORES)

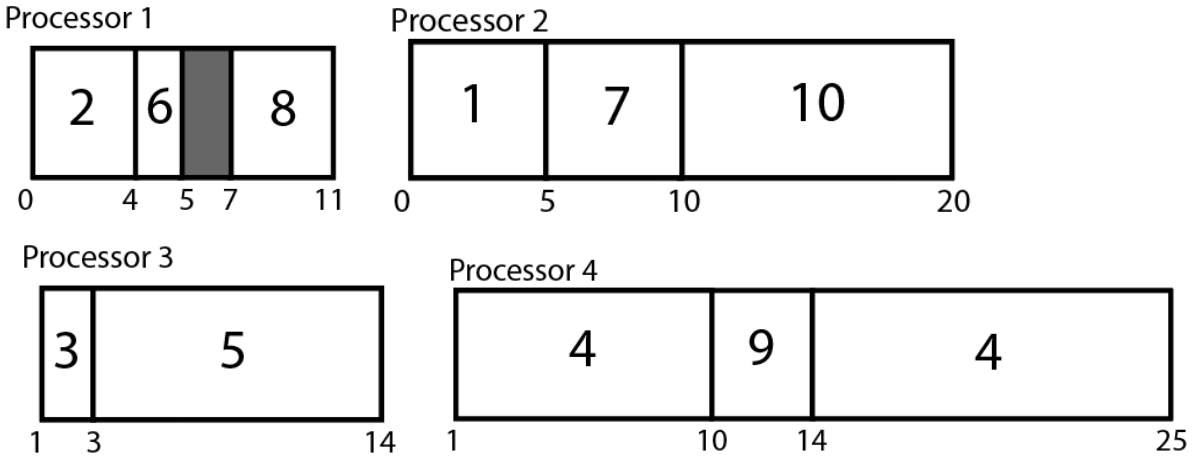
Improved RR (4 cores):

Average Response Time: 0.500

Average Waiting Time: 0.900

Average Turn-Around Time: 7.500

Context Switches: 6



Process	Arrival Time	Burst Time
1	0	5
2	0	4
3	1	2
4	1	20
5	3	11
6	4	1
7	4	5
8	7	4
9	7	4
10	9	10

IMPROVEMENTS USING PARALLEL PROCESSING

Method	N_Processes	Avg Response Time	Avg Waiting Time	Avg Turnaround Time	Context Switches
Round Robin	50	94.66864	176.18346	182.24346	82
Improved Round Robin	50	92.37462	93.81462	99.87462	53
Improved Round Robin Dual (2) Processor	50	43.24	43.24	49.3	48
Improved Round Robin Quad (4) Processor	50	21.2	21.14797	27.20797	49
Round Robin	70	117.5181714	230.4090286	235.8518857	120
Improved Round Robin	70	121.2	121.2	126.6428571	69
Improved Round Robin Dual (2) Processor	70	56.01428571	56.01428571	61.45714286	68
Improved Round Robin Quad (4) Processor	70	24.08571429	24.08571429	29.52857143	66
Round Robin	100	218.596035	424.161765	431.051765	173
Improved Round Robin	100	219.36	219.36	226.25	99
Improved Round Robin Dual (2) Processor	100	100.69	100.69	107.58	98
Improved Round Robin Quad (4) Processor	100	42.721415	42.66761	49.55761	97

IMPROVEMENTS USING PARALLEL PROCESSING

Method	N_Processes	Avg Response Time	Avg Waiting Time	Avg Turnaround Time	Context Switches
Round Robin	150	235.0491067	470.3493067	475.4759733	255
Improved Round Robin	150	239.78	239.78	244.9066667	149
Improved Round Robin Dual (2) Processor	150	106.0133333	106.0133333	111.14	148
Improved Round Robin Quad (4) Processor	150	40.78876333	40.76796667	45.89463333	147
Round Robin	200	328.83582	639.1362	644.4512	332
Improved Round Robin	200	329.36	329.36	334.675	199
Improved Round Robin Dual (2) Processor	200	147.21	147.21	152.525	198
Improved Round Robin Quad (4) Processor	200	58.005	58.005	63.32	196

REPOSITORY OF WORK DONE

GitHub Repo :
https://github.com/TheKillingAMD/OS-Algorithm-Improved-Round-Robin

Report :

https://www.overleaf.com/read/zmvmbshfxtpt