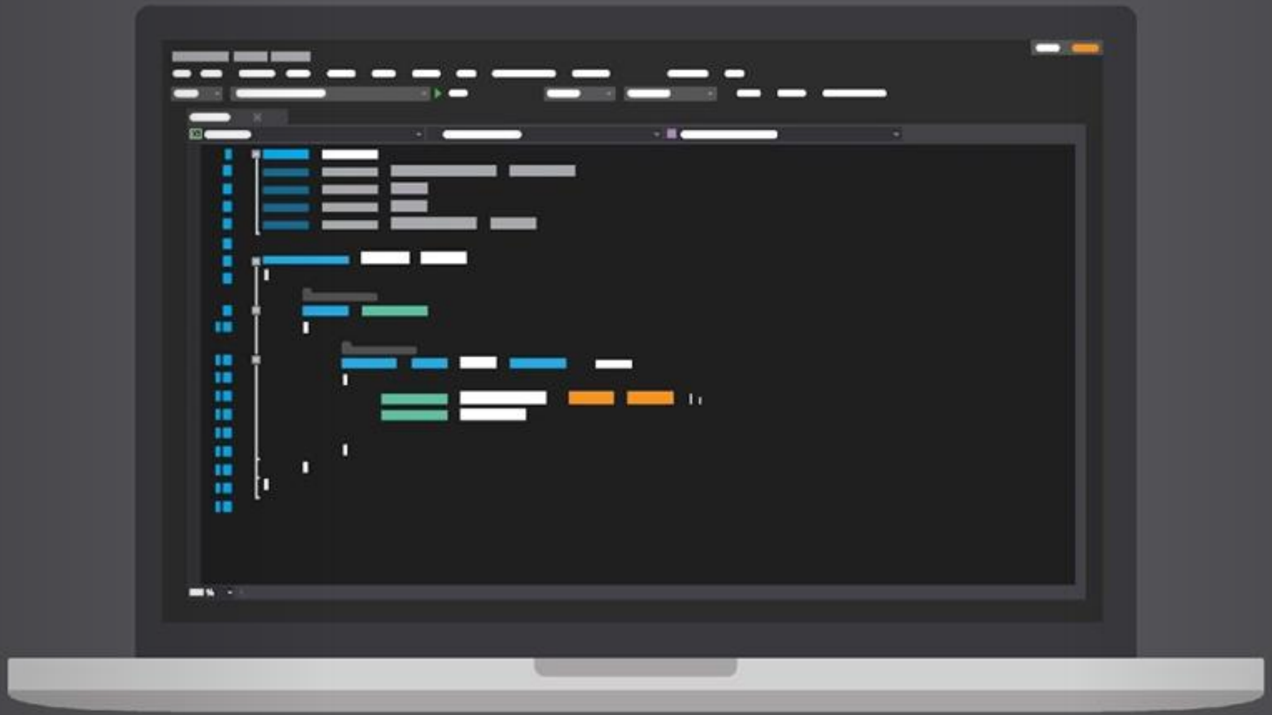
IMPROVED ROUND ROBIN-BASED SCHEDULING ALGORITHM



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.

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

LITERATURE SURVEY 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
```

LITERATURE SURVEY COMPARISON (GANTT CHART)

Process	Arrival Time	Burst Time
A	0	10
B	5	15
C	10	5

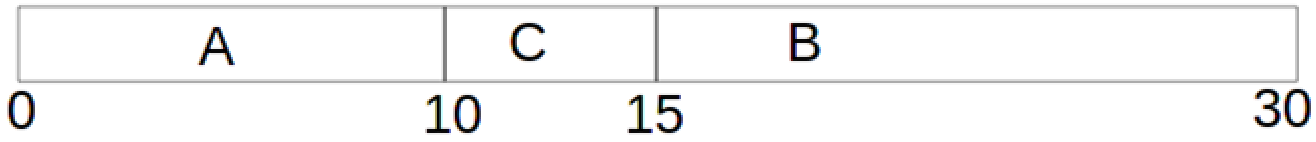
Regular Round Robin Scheduling

Average waiting time = (3.5 + 7 + 13.5 + 6.5)/3
= 30.5/3

LITERATURE SURVEY COMPARISON (GANTT CHART)

PROCESS	ARRIVAL TIME	BURST TIME
A	0	10
B	5	15
C	10	5

Modified Round Robin Scheduling



Average waiting time = 10/3

PREDICTED IMPROVEMENTS WITH THE ALGORITHM

1. Using the Algorithm helps in reducing Average Waiting, Turn Around and Responsive 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
0.8	50	95.40816	90.66816	101.46816	56
0.85	50	93.81462	92.37462	99.87462	53
0.9	50	93.12	93.12	99.18	49
0.95	50	93.12	93.12	99.18	49
0.8	100	222.1464	216.2964	229.0364	109
0.85	100	219.36	219.36	226.25	99
0.9	100	219.36	219.36	226.25	99
0.95	100	219.36	219.36	226.25	99
0.8	150	239.78	239.78	244.9066667	149
0.85	150	239.78	239.78	244.9066667	149
0.9	150	239.78	239.78	244.9066667	149
0.95	150	239.78	239.78	244.9066667	149
0.8	200	329.36	329.36	334.675	199
0.85	200	329.36	329.36	334.675	199
0.9	200	329.36	329.36	334.675	199
0.95	200	329.36	329.36	334.675	199

A NEW CONCEPT - PIPELINING INTO ROUND ROBIN

Ib Fill This LUL

ROUND ROBIN WITH PIPE LINING

```
Algorithm 1: Pseudo-Code
Result: Improved Round Robin with Pipe Lining
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 = 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 T
  Context Switches
6.End
```

IMPROVEMENTS USING PIPE LINING

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 PIPE LINING

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