Intelli-J Slice

A hybrid Job Scheduling Algorithm based on round robin, priority and SJF

PROJECT REPORT

Course: Operating Systems (CSE2005)

By

16BCE2135	Pranav Karnani
16BCE0986	Siddharth Malhotra
16BCE0871	Ujjwal Khosla

Slot: A2

Faculty In-charge: PROF. SELVAKUMAR K

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

VIT® UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
VELLORE ▪ CHENNAI
www.vit.ac.in

October, 2017

CERTIFICATE

This is to certify that the project work entitled "Intelli-J Slice" *(A hybrid Job Scheduling Algorithm based on round robin, priority and SJF)*" that is being submitted by "Pranav Karnani(16BCE2135), Ujjwal Khosla(16BCE0871) , Siddharth Malhotra(16BCE0986)" for OPERATING SYSTEMS (CSE2005) is a record of bonafide work done under my supervision.

The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted for any other CAL course.

Place: Vellore Date: 29/10/2017

Signature of Students:

PRANAV KARNANI

SIDDHARTH MALHOTRA

UJJWAL KHOSLA

Signature of the Faculty: PROF. SELVAKUMAR K

ACKNOWLEDGEMENT

This project was supported by our OPERATING SYSTEMS teacher, Professor Selvakumar K. We thank our sir for his help and guidance throughout the project. His knowledge provided great insight and expertise that greatly assisted the research and his guidelines minimized our mistakes exponentially.

We would like to thank the VIT University Management, and our Dean for giving us an opportunity to carry out our studies in this prestigious university, and help in pursuit of a safe and secure future.

PRANAV KARNANI (16BCE2135)

SIDDHARTH MALHOTRA (16BCE0986)

UJJWAL KHOSLA (16BCE0871)

ABSTRACT

Primitive job scheduling algorithms such as Round Robin can not be used in real time OS because of low throughput, large waiting time, large turnaround time, large response time, high context switch rates. Our intentions with this research was to come up with a better solution to such problems by employing the merits of prevalent scheduling algorithms. The proposed algorithm is a hybrid of priority, RR and SJF. It also deals with the problem of starvation by assigning variable quantum time (referred to as intelligent time slice) after each cycle. We also intend to compare, via this paper, the results of our new hybrid algorithm with those of prevalent scheduling algorithms.

INTRODUCTION

CPU scheduling is a process which allows one process to use the CPU while the execution of another process is on hold (in waiting state) due to unavailability of any resource like I/O etc, thereby making full use of CPU. The aim of CPU scheduling is to make the system efficient, fast and fair.

Whenever the CPU becomes idle, the operating system must select one of the processes in the ready queue to be executed. The selection process is carried out by the short-term scheduler (or CPU scheduler). The scheduler selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them.

There exists a number of CPU scheduling algorithms like First Come First Serve, Shortest Job First Scheduling, Round Robin scheduling, Priority Scheduling etc, but due to a number of disadvantages these are rarely used in real time operating systems except Round Robin scheduling.

A number of assumptions are considered in CPU scheduling which are as follows:

1. Job pool consists of run able processes waiting for the CPU.

2. All processes are independent and compete for resources.

3. The job of the scheduler is to distribute the limited resources of CPU to the different processes fairly and in a way that optimizes some performance criteria.

The scheduler is the component of the kernel that selects which process to run next. Operating systems features three distinct types of schedulers, a long term scheduler, a mid-term or medium term scheduler and a short-term scheduler.

Long term scheduler or job scheduler: selects processes from the job pool and loads them into memory for execution.

Short term scheduler, or CPU scheduler: selects from among the processes that are ready to execute, and allocates CPU to one of them.

Medium term scheduler: removes processes from memory and reduces the degree of multiprogramming results in the scheme of swapping.

Swapping is the scheme which is performed by dispatcher which is the module that gives control of the CPU to the process selected by the short-term scheduler .

The CPU scheduling also plays an important role in the real time operating system which always has a time constraint on computations. A real time system is the one whose applications are mission-critical, where real-time tasks should be scheduled to be completed before their deadlines. Most real-time systems control unpredictable environments and may need operating systems that can handle unknown and changing tasks. So, not only a dynamic task scheduling is required, but both system hardware and software must adapt to unforeseen configurations.

There are two main types of real-time systems: Hard Real-Time System, Firm or Soft Real-Time System. In Hard Real-Time System, it requires that fixed deadlines must be met otherwise disastrous situation may arise whereas in Soft Real-Time System, missing an occasional deadline is undesirable, but nevertheless tolerable. System in which performance is degraded but not destroyed by failure to meet response time constraints is called soft real time systems.

In real time systems each task should be invoked after the ready time and must be completed before its deadline , an attempt has been made to satisfy these constraints. Simple round robin architecture is not suitable to implement in Soft real time due to more number of context switches, longer waiting and response times. This in turn leads to low throughput in the system. If a real-time process having relatively larger CPU burst it will leads to the problem of starvation .Priority scheduling may be a better option for real-time scheduling but it will face the similar problem i.e. low priority processes will always starve.

KEYWORDS

First Come First Serve Scheduling
Shortest Job First Scheduling
Round Robin Scheduling
Priority Scheduling
Long term scheduler or job schedulerShort term scheduler, or CPU scheduler

Medium term scheduler

Swapping

Dispatcher

Hard Real-Time System

Soft Real-Time System

First Come First Serve (FCFS)

- Jobs are executed on first come, first serve basis.
- It is a non-preemptive, pre-emptive scheduling algorithm.
- Easy to understand and implement.
- Its implementation is based on FIFO queue.
- Poor in performance as average wait time is high.

Shortest Job First(SJF)

- This is a non-preemptive, pre-emptive scheduling algorithm.

- Best approach to minimize waiting time.
- Easy to implement in Batch systems where required CPU time is known in advance.
- Impossible to implement in interactive systems where required CPU time is not known.
- The processer should know in advance how much time process will take.

Priority Based Scheduling

- Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
- Each process is assigned a priority. Process with highest priority is to be executed first and so on.
- Processes with same priority are executed on first come first served basis.
- Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Round Robin Scheduling

- Round Robin is the preemptive process scheduling algorithm.
- Each process is provided a fix time to execute, it is called a quantum.
- Once a process is executed for a given time period, it is preempted and other process executes for a given time period.
- Context switching is used to save states of preempted processes.

Our motive:

We tried to come up with a modified round robin scheduling algorithm which works on a dynamic time quantum, which changes after every round of execution. This will help in reducing the average waiting time, average turnaround time and reduce the number of context switch.

ALGORITHM:

An intelligent program which just takes the number of processes, their burst times and their priority as input.

The machine then asks the user as of how he/she want's all his/her processes to be scheduled and depending on that the corresponding scheduling algorithm is used.

Step 1: Start

Step 2:Read the burst time , priority of all the processes

Step 3:Read the user's choice

Step 4:Create a new priority index such that, for a process i in n

Step 5: P = ((a*burst time of i'th process + b*priority of i'th process)/n)*r

Step 6: Sort the processes in ascending order based on the new priority

Step 7: Calculate the number of processes having burst time > 0

Step 8: Set c = THE number of processes having burst time = 0

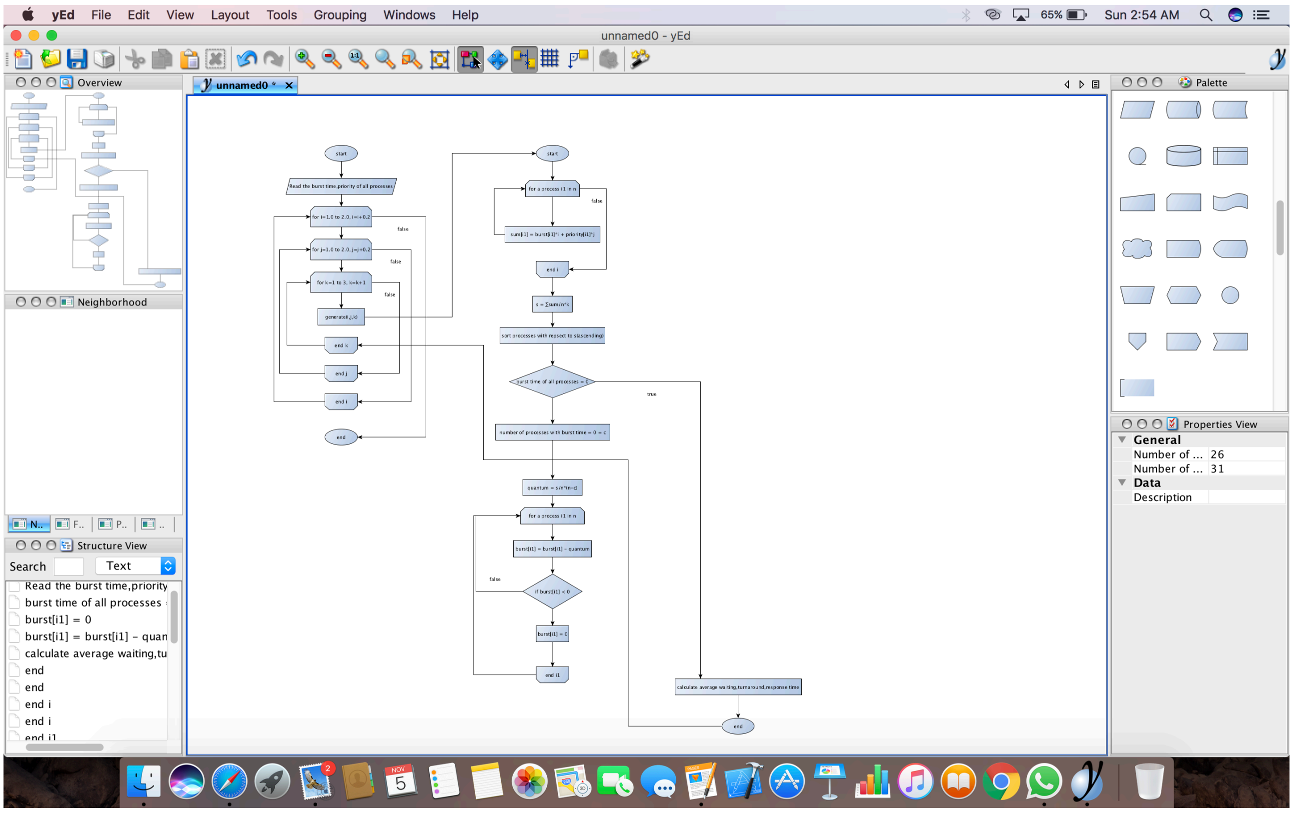
Step 9:Set the time quantum (∑ P)/n-c

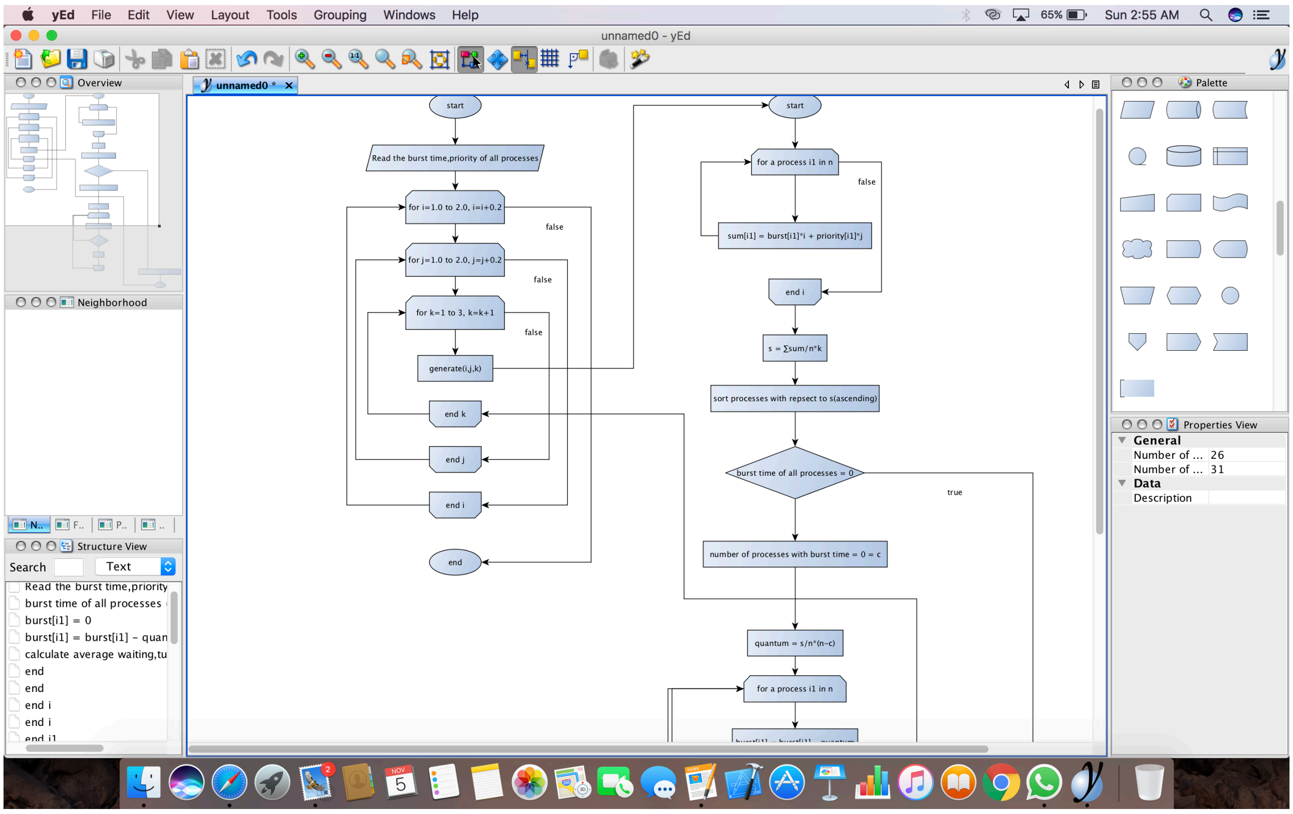
Step 10: Repeat the same process from step 7

Step 11: With this time quantum, we subtract the burst time of each process with the time quantum for that particular

Step 12: Calculate the average waiting time , turnaround time, response time for that sequence of processes.

FLOWCHART


start
Read the burst time,priority of all processes
for i=1.0 to 2.0, i=i+0.2
for j=1.0 to 2.0, j=j+0.2
for k=1 to 3, k=k+1
generate(i,j,k)
end k
end j
end i
end
false
false
false
start
for a process i1 in n
sum[i1] = burst[i1]*i + priority[i1]*j
end i
s = ∑sum/n*k
sort processes with repsect to s(ascending)
burst time of all processes = 0
true
number of processes with burst time = 0 = c
quantum = s/n*(n-c)
for a process i1 in n
burst[i1] = burst[i1] - quantum
if burst[i1] < 0
false
burst[i1] = 0
end i1
calculate average waiting,turnaround,response time
end


start
Read the burst time,priority of all processes
for i=1.0 to 2.0, i=i+0.2
false
for j=1.0 to 2.0, j=j+0.2
false
for k=1 to 3, k=k+1
false
generate(i,j,k)
end k
end j
end i
end
start
for a process i1 in n
false
sum[i1] = burst[i1]*i + priority[i1]*j
end i
s = ∑sum/n*k
sort processes with repsect to s(ascending)
burst time of all processes = 0
true
number of processes with burst time = 0 = c
quantum = s/n*(n-c)
for a process i1 in n

RESULT AND DISCUSSION

The above code generates more than 400 outcomes for the process to follow. The system automatically checks which time quantum would be best suitable for the asked user's process.

Example :

INPUT:

```
Enter 1 for better waiting time, 2 for better turnaround , 3 for better response, 4 for all results: 2
Enter the number of processes: 5

Enter burst time for process 1: 22
Enter the priority of the process 1: 4
Enter burst time for process 2: 18
Enter the priority of the process 2: 2
Enter burst time for process 3: 9
Enter the priority of the process 3: 1
Enter burst time for process 4: 10

Enter the priority of the process 4: 3
Enter burst time for process 5: 4
Enter the priority of the process 5: 5
```

OUTPUT:

```
For multipliers with value : 1 1         1

Time quantum : 3.12
Round 1
0.88    5.88    6.88    14.88   18.88
```

```
Turnaround time after current round:
15.6     15.6     15.6     15.6     15.6
Waiting time after current round:
12.48    12.48    12.48    12.48    12.48
Response after current round:
0        3.12     6.24     9.36     12.48
Time quantum : 3.12
Round 2
0        2.76     3.76     11.76    15.76
Turnaround time after current round:
16.48    28.96    28.96    28.96    28.96
Waiting time after current round:
12.48    22.72    22.72    22.72    22.72
Response after current round:
0        3.12     6.24     9.36     12.48
Time quantum : 3.9
Round 3
0        0        0        7.86     11.86
Turnaround time after current round:
16.48    31.72    35.48    43.28    43.28
Waiting time after current round:
12.48    22.72    25.48    33.14    33.14
Response after current round:
0        3.12     6.24     9.36     12.48
Time quantum : 7.8
Round 4
0        0        0        0.06     4.06
Turnaround time after current round:
16.48    31.72    35.48    58.88    58.88
Waiting time after current round:
12.48    22.72    25.48    40.94    40.94
Response after current round:
0        3.12     6.24     9.36     12.48
```

```
Time quantum : 7.8
Round 5
0       0       0       0       0
Turnaround time after current round:
16.48   31.72   35.48   58.94   63
Waiting time after current round:
12.48   22.72   25.48   40.94   41
Response after current round:
0       3.12    6.24    9.36    12.48
Average waiting time: 28.524
Average response time: 6.24
Average turnaround time: 41.124
```

There are much more cases generated for the following sequence with different time quantum's but the user's choice gives a desired output:

```
Desired Output
Time quantum : 18.36
Round 1
0       0       0       0       3.64
Turnaround time after current round:
4       13      23      41      59.36
Waiting time after current round:
0       4       13      23      41
Response after current round:
0       4       13      23      41
Time quantum : 91.8
Round 2
0       0       0       0       0
Turnaround time after current round:
4       13      23      41      63
Waiting time after current round:
0       4       13      23      41
```

```
Response after current round:
0         4         13        23        41
Average waiting time: 16.2
Average response time: 16.2
Average turnaround time: 28.8
```

CONCLUSION AND FUTURE WORK

The algorithm can furthermore fastened by directly checking which program and time quantum sequence would be best for an Operating System.

From the previous work and comparisons, we observed that our proposed algorithm is performing better than the algorithm PBDRR and also a proposed in paper [1] in most cases in terms of average waiting time, average turnaround time and number of context switches thereby saving the memory spaces and in addition to that, we enabled the algorithm to choose any of the ready process will be the next based on the waiting time and service.

We can also add a parameter for deadline which would help the code to analyse and find the best process sequence and time quantum and would thereby be used in Hard real time Systems.

REFERENCES

1. Rakesh Mohanty, H. S. Behera, Khusbu Patwari , Monisha Dash and M. Lakshmi Prasanna, “Priority Based Dynamic Round Robin (PBDRR) Algorithm with Intelligent Time Slice for Soft Real Time Systems,” International Journal of Advanced Computer Science and Applications (IJACSA)Vol. 2, No.2, Feb 2011.

2. Priority Based Dynamic Round Robin with Intelligent Time Slice and Highest Response Ratio Next Algorithm for Soft Real Time System by Zena Hussain Khalil 1, Ameer Basim Abdulameer Alaasam

3. Research Papers on an improved scheduling algorithm for real time systems. Real-Time Scheduling Algorithms by Vimal Kumar

4. A New Improved Round Robin (NIRR) CPU Scheduling Algorithm by Abdulrazaq Abdulrahim, Saleh E Junaidu and B. Sahalu Iya Abubakar

5. An Improved CPU Scheduling Algorithm by Himanshi arora & Deepanshu arora.