Efficient vaccine scheduler based on CPU scheduling algorithms

Vasu Gondaliya
Department of Computer Science and Engineering
School of Technology
Pandit Deendayal Energy University
Gandhinagar, India
vpgondaliya1@gmail.com

Shreya Patel
Department of Computer Science and Engineering
School of Technology
Pandit Deendayal Energy University
Gandhinagar, India
shreyapatel342000@gmail.com

Jay Hemnani
Department of Computer Science and Engineering
School of Technology
Pandit Deendayal Energy University
Gandhinagar, India
jayhemnani992000@gmail.com

Samir Patel
Department of Computer Science and Engineering
School of Technology
Pandit Deendayal Energy University
Gandhinagar, India
samir.patel@sot.pdpu.ac.in

2021 International Conference on Artificial Intelligence and Machine Vision (AIMV) | 978-1-6654-4211-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/AIMV53313.2021.9670986

***Abstract*—This paper presents an efficient algorithm for scheduling vaccination process which is based on the CPU scheduling algorithms of an operating system. Based on a custom-designed scoring system of the given schedule, an analysis of why the first-come, first-served basis of scheduling vaccines is inefficient. The ranking system is based on the concept that health care personnel should be given higher priority, followed by front-line workers, major healthcare patients, elderly people, and finally the general public. This is the category in the dataset, the higher the importance of the person who needs to be vaccinated, the better the score. Different CPU scheduling methods are analyzed based on Arrival Time, Turn Around Time, Waiting Time, and Response Time. We obtain the resultant schedule after providing the dataset, the number of vaccines per day, and the selected algorithm for scheduling, and we get a customized schedule based on the data by entering the Aadhar number. The FCFS and Priority algorithms were compared to visualize the differences in efficiency for both algorithms, as well as an analysis of how many vaccines to choose per day and the related length of schedule in days.**

***Index Terms*—operating system, scheduling algorithms, vaccine scheduler**

I. Introduction

A. Operating System

The operating system is the core program that controls the operation of any electronic device. It's a program that lets many processes share resources such as processing time, external device, memory, and storage. It uses different types of strategies and algorithms to achieve this. Different scheduling techniques are used to assign processing time to each process [1] .

B. Scheduling Algorithms

The operating system employs a variety of ways to efficiently allot processing time to processes. To accomplish this, it employs a variety of schedulers, which are short-term, mid-term, and long-term schedulers [2].
We need to know the types of processes in the system to determine the scheduling algorithm's criteria.

II. Basic Terminology

A. Types of CPU Scheduling

1) Preemptive: A process in the running queue can be removed (preempted) by other higher priority processes or with better criteria satisfaction or the given time quantum is utilized [3].
2) Non-Preemptive: Once a process enters the running queue, it will only leave when the required CPU Burst Time is completed or it needs an I/O time.

B. Types of Times

- Arrival Time (AT): The time at which a process enters the ready queue.
- Burst Time (BT): The amount of time a process takes to finish its execution is called burst time. It is also known as CPU time. This cannot be computed ahead because we cannot determine the burst time until the process is completed [4].
- Completion Time (CT): The time at which the process completes its execution.
- Turn Around Time (TAT): The time between arrival time and completion time.

$$TAT = CT - AT \tag{1}$$

- Waiting Time (WT): The time not utilized for execution, which is the difference between turn around time and burst time.

$$WT = TAT - BT \tag{2}$$

978-1-6654-4211-4/21/$31.00 ©2021 IEEE

- Response Time (RT): The amount of time between arrival time and the time at which the process enters the running queue for the first time [5].
- Input-Output Time (IO): The amount of time a process needs for input and output.
- Context Switching Time (CS): The time required for a process to exit the running queue and save progress and move out to ready queue or IO or block queue according to its burst time [6].

C. Algorithms

1) First Come First Serve (FCFS):
   Type of CPU Scheduling: Non-preemptive
   Criteria: The first process that arrived in the ready queue is processed first.
   FCFS is the simplest algorithm of all, and also the easiest to implement.
   The processes running with the FCFS algorithm can suffer from a convoy effect. It happens when the processes of large burst times enter the ready queue and it will cause starvation to all the other processes in ready queue [7].
   So this algorithm is preferred only when processes are of smaller burst times.
2) Shortest Job First (SJF):
   Type of CPU Scheduling: Non-preemptive
   Criteria: The shortest job in the ready queue is processed first.
   SJF is an algorithm that performs well theoretically but it cannot be implemented practically because we cannot know the burst times of the processes beforehand and thus not possible to assign which process to enter the ready queue.
   This algorithm also causes starvation to processes with longer burst times because they will be processes at last [8].
3) Longest Job First (LJF):
   Type of CPU Scheduling: Non-preemptive
   Criteria: The longest job in the ready queue is processed first.
   LJF is also an algorithm that cannot be practically implemented because we cannot know the burst times of the processes beforehand and thus not possible to assign which process to enter the ready queue.
   This algorithm also causes starvation to processes with shorter burst times because they will be processes at last.
   This algorithm is amongst the worst performers of the nine algorithms.
4) Shortest Remaining Time First (SRTF):
   Type of CPU Scheduling: Preemptive
   Criteria: The shortest job in the ready queue is processed first.
   SRTF is the preemptive version of the SJF algorithm. It performs better than SJF but also has more overheads because of the higher number of context switches resulting in higher waiting time and turn around time.
5) Largest Remaining Time First (LRTF):
   Type of CPU Scheduling: Preemptive
   Criteria: The longest job in the ready queue is processed first.
   LRTF is the preemptive version of the LJF algorithm. It performs better than LJF but also has more overheads because of the higher number of context switches resulting in higher waiting time and turn around time.
6) Round Robin (RR):
   Type of CPU Scheduling: Preemptive
   Criteria: The jobs in the ready queue are given a fixed time quantum.
   When a process enters the ready queue, it is given a fixed amount of time it can stay in the ready queue. The process is preempted when this time is completed or it completes its execution. This time is known as time quantum [9].
   It follows a queue data structure. Because of this no process suffers from starvation and thus results in low response time.
   This algorithm is the most practical algorithm which can be implemented. As we do not need the burst times of the processes, it can be used in operating systems.
7) Priority Non Preemptive (PNP):
   Type of CPU Scheduling: Non-preemptive
   Criteria: The highest priority job in the ready queue is processed first.
   In priority algorithm, processes are assigned priority and higher priority processes are given preference [10].
   This algorithm is practically implementable because it does not need burst times to schedule processes.
   This algorithm gives rise to starvation to lower priority processes because they are processed at last.
8) Priority Preemptive (PP):
   Type of CPU Scheduling: Preemptive
   Criteria: The highest priority job in the ready queue is processed first.
   Priority algorithm with preemptive mode performs well than the non-preemptive version but has more overhead due to higher number of context switches which increases waiting time and completion time.
9) Highest Response Ratio Next (HRRN):
   Type of CPU Scheduling: Non-preemptive
   Criteria: The job with the highest response ratio in the ready queue is processed first.
   Response Ratio R is given by [11],

$$R = (w + s)/s \tag{3}$$

   where w is waiting time for a process so far and s is service time or burst time of a process so far.
   This algorithm not only supports shorter jobs but also limits the waiting time of longer jobs. It is practically implementable because we only need waiting time and service time which has been passed so far.

Process Id	Priority	Arrival Time	Process Time				
P1	1	0	CPU	IO	CPU		
			3	5	1		
P2	2	0	CPU	IO	CPU	IO	CPU
			2	1	3	1	4
P3	3	2	CPU	IO	CPU		
			1	3	3		
P4	1	5	CPU				
			4				
P5	2	7	CPU	IO	CPU		
			2	1	5		

TABLE I: Example of input of the processes with arrival time, priority, burst time, IO time, and process ID. Here algorithm selected is SRTF.

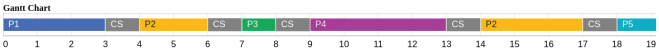


Fig. 1: Calculated Gantt Chart for the given input.

III. Analysis of CPU scheduling algorithms

A GUI has been made where processes can be added with a number of burst time and IO time and algorithm can be selected for which we get resultant Gantt chart and Timeline chart [12].

The figures from Fig. 1 to Fig. 4 and the GUI of CPU scheduling algorithm is at [13].

From Fig. 3, we find that if we want average waiting time for all processes, we select the Round Robin algorithm, FCFS is practical and can be used for smaller processes. SJF and SRTF are good theoretically but are not practical as burst times cannot be known in advance. LJF and LRTF result in high waiting times and completion times.

From Fig. 4, we find that low time quantum gives high waiting time, completion time, and turn around time because the context switches increase which results in high overhead. But if the time quantum is high, this algorithm transforms into FCFS and SJF. So some time in the middle needs to be selected to get the full potential of the Round Robin algorithm.

IV. Vaccine Scheduler

We need dataset in the format given in Table. II.

A. Columns of Dataset

- Name
- Aadhar Card Number (12 digits)
- Phone Number
- Date of Birth
- Category
- Registration Time and Date

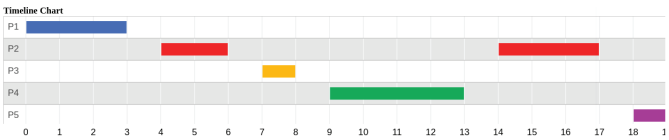


Fig. 2: Calculated Timeline Chart for the given input.

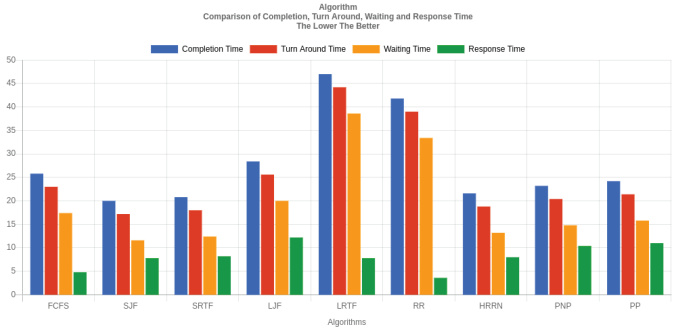


Fig. 3: Comparison between all algorithms according to Completion Time, Turn Around Time, Waiting Time, and Response Time.

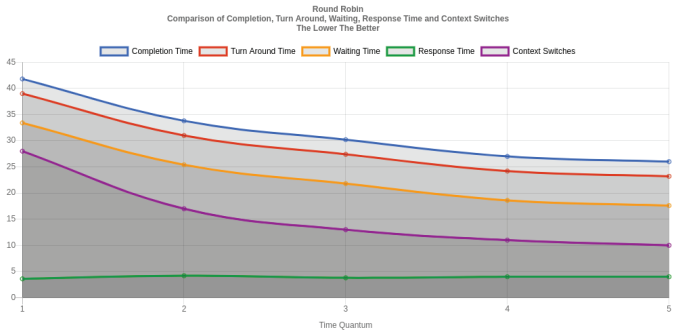


Fig. 4: Comparison between all the possible time quantum for the round-robin algorithm to compare all the output times and get which one is most efficient.

B. Types of Category

1) Health Care Workers like doctors, hospital staff
2) Front Line Workers like police and army
3) People with serious health problems
4) People with age > 45
5) General Public

Distribution of different categories in the dataset is given in Fig. 5.

C. Vaccine Scheduler Algorithms

1) First Come First Serve : According to the registration time and date, the vaccination date is scheduled for the person.
2) Priority : According to the category and then according to the registration date and time, a vaccination date is scheduled for the person. The lower the category, the higher the priority for the person.

Name	Aadhar Card Number	Phone Number	Date Of Birth	Category	Registration Time Stamp
Chloe Morris	4722 6965 3137	9463889623	02/10/1955	2	12/05/2021 19:45:39
Johanna Montoya	5146 7162 7507	9373479524	18/07/1934	3	14/05/2021 23:36:55
Mylie Lamb	3437 7485 1740	9185188559	27/08/1930	1	13/05/2021 09:48:57

TABLE II: Format for dataset input for vaccine scheduler.

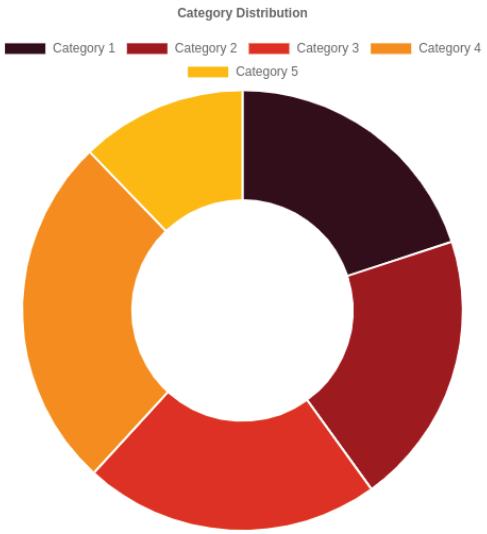


Fig. 5: Distribution of different categories in given dataset.

Category/Days From Registration	1	2	3	4	5
1	10000	8000	6000	4000	2000
2	9999	7999	5999	3999	1999
3	9998	7998	5998	3998	1998
...	...	...	...	...	...
1998	8003	6003	4003	2003	3
1999	8002	6002	4002	2002	2
2000	8001	6001	4001	2001	1

TABLE III: Scoring System for comparing algorithms.

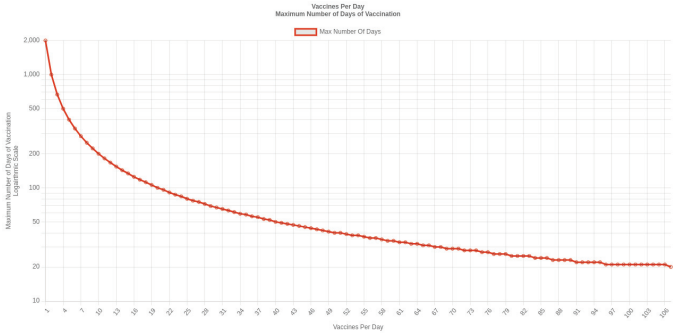


Fig. 6: Logarithmic plot between number of vaccines per day v/s number of days the vaccination will go on with this number of vaccines per day

D. Scoring System

It works on these two principles in the same order :

1) Choose higher priority whenever possible.
2) Choose the one which has the prior registration whenever possible.

The number of registrations N should be known.

$$N = 2000$$

To calculate the score of a cell as given Table. III directly from output and dataset input,

$$x_i = 5 * (N - d_i) + (6 - c_i) \tag{4}$$

where,
d = Number of days from registration after which vaccination scheduled and
c = Category

After we get the corresponding cell value for all data points,

$$x_1, x_2, x_3 ... x_{1999}, x_{2000}$$

Squaring and adding these values and then taking root,

$$Score = \sqrt{{x_1}^2 + {x_2}^2 + ... + {x_{2000}}^2} \tag{5}$$

This score can be used to find which algorithm is better.

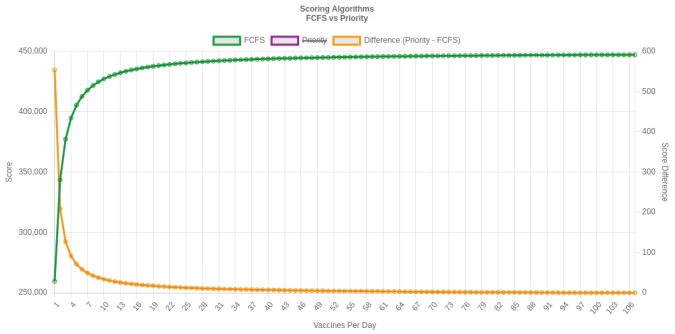
E. Analysis

The figures from Fig. 5 to Fig. 9 and the GUI of Vaccine scheduler are at [14]. We get a complete schedule according to the dataset, the number of vaccines per day, and the algorithm selected to schedule the data.

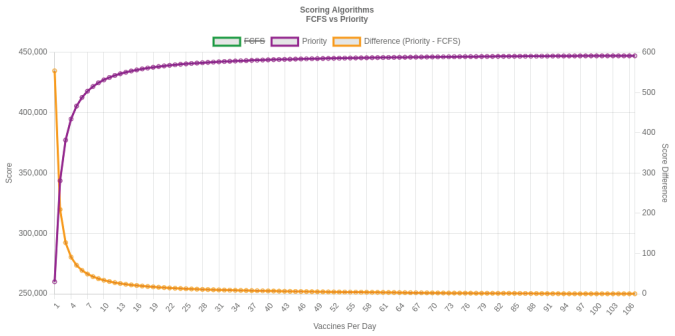
From Fig. 6, we find the number of days the vaccination process will go on irrespective of the algorithms selected according to the number of vaccines per day allocated to a center. So how many days is acceptable for the schedule to go on.

From Fig. 7, we find the score of both algorithms according to the custom scoring system. As the number of vaccines per day increases, scores for both algorithms will increase as all people will get vaccinated earlier (green and violet plot). We find that the priority algorithm is better for lower vaccines per day but as the vaccines per day increases, the score difference decreases (yellow plot) because it becomes irrelevant what category registration is as all will be vaccinated each day. So vaccines per day should be selected according to it.

From Fig. 8, we get that FCFS results in constant distribution of each category (color) on each day because no importance is given to the category of people. But in the

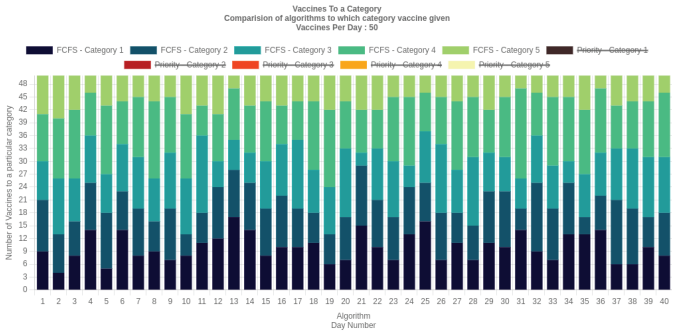


(a) FCFS

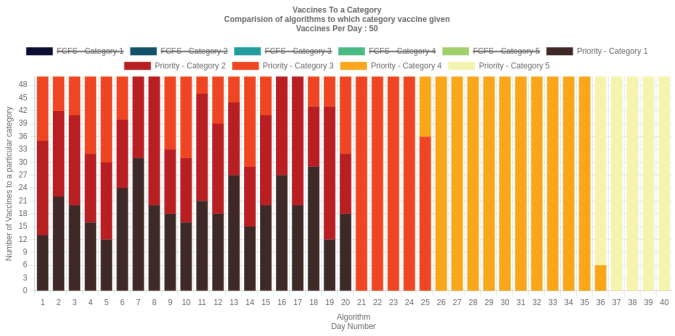


(b) Priority

Fig. 7: Plot between FCFS and Priority algorithms according to the custom scoring system made by using category and number of days of waiting time between registration and vaccination.



(a) FCFS



(b) Priority

Fig. 8: A plot of what categories are vaccinated each day for both algorithms. The number of Vaccines per day is 50. The darker the color, the higher the priority.

priority algorithm, we see that the darker colors fade as days pass by. It means, higher priority (darker color) people are vaccinated first and others later. FCFS is the blue color palette and priority is the red color palette.

We can get the schedule of person by entering Aadhar number as seen in Fig. 9.

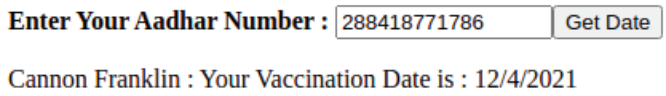


Fig. 9: Get Schedule by entering the Aadhar Number

V. Conclusion

We see that how the CPU Scheduling Algorithm can be used to make a more efficient Vaccine Scheduler. This algorithm can be used in many other instances where different categories are given different priorities.

This algorithm was needed because the vaccination process in India was not efficient because of which many lives got lost in the second wave where vaccines were available but no one was taking them and when people started registering, only a limited number of people were given chance to register in a limited window and followed first come first serve based approach.

The health workers and the front line workers were given the vaccination first then the older people and then the general public. This led to extra vaccines per day on each day because only a particular category person was allowed to register.

So using this algorithm we can get the vaccination to all of the citizens with efficiency.

Acknowledgment

I am thankful to Pandit Deendayal Energy University for supporting this work.

References

[1] P. B. Hansen, *Operating system principles*. Prentice-Hall, Inc., 1973.
[2] N. Goel and R. Garg, "A comparative study of cpu scheduling algorithms," 07 2013.
[3] I. Qureshi, "Cpu scheduling algorithms: A survey," *International Journal of Advanced Networking and Applications*, vol. 5, no. 4, p. 1968, 2014.
[4] E. Oyetunji and A. Oluleye, "Performance assessment of some cpu scheduling algorithms," *Research Journal of Information Technology*, vol. 1, no. 1, pp. 22–26, 2009.
[5] M. Gahlawat and P. Sharma, "Analysis and performance assessment of cpu scheduling algorithms in cloud using cloud sim," *Int. J. Appl. Inf. Syst*, vol. 5, no. 9, pp. 5–8, 2013.
[6] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Concepts*, 9th ed. Wiley Publishing, 2012.
[7] O. Mac, "Operating system," *CWE Version 1.5*, p. 66, 2009.
[8] G. D. Bibu and G. C. Nwankwo, "Comparative analysis between first-come-first-serve (fcfs) and shortest-job-first (sjf) scheduling algorithms," *Int J Comput Sci Mobile Comput*, vol. 8, no. 5, pp. 176–181, 2019.
[9] A. Singh, P. Goyal, and S. Batra, "An optimized round robin scheduling algorithm for cpu scheduling," *International Journal on Computer Science and Engineering*, vol. 2, no. 07, pp. 2383–2385, 2010.

[10] B. Andersson, S. Baruah, and J. Jonsson, "Static-priority scheduling on multiprocessors," in *Proceedings 22nd IEEE Real-Time Systems Symposium (RTSS 2001)(Cat. No. 01PR1420)*. IEEE, 2001, pp. 193–202.

[11] H. Behera, B. K. Swain, A. K. Parida, and G. Sahu, "A new proposed round robin with highest response ratio next (rrhrrn) scheduling algorithm for soft real time systems," *International Journal of Engineering and Advanced Technology*, vol. 37, pp. 200–206, 2012.

[12] P. Singh, V. Singh, and A. Pandey, "Analysis and comparison of cpu scheduling algorithms," *International Journal of Emerging Technology and Advanced Engineering*, vol. 4, no. 1, pp. 91–95, 2014.

[13] V. Gondaliya, "Cpu scheduling algorithms," 2020, [accessed 04-December-2020]. [Online]. Available: https://vasu-gondaliya.github.io/cpu-scheduling-algorithms/

[14] V. Gondaliya, "Vaccine scheduler," 2021, [accessed 29-June-2021]. [Online]. Available: https://vasu-gondaliya.github.io/vaccine-scheduler/