Scheduling Simulator

Vineeth Bhat
Dept.Computer Science and Engineering
Frankfurt University of Applied Sciences
Frankfurt, Germany
vineeth.bhat@stud.fra-uas.de

Jathin Sreenivas
Dept.Computer Science and Engineering
Frankfurt University of Applied Sciences
Frankfurt, Germany
Jathin.Sreenivas@stud.fra-uas.de

***Abstract*—**

I. Introduction

Real-Time systems have wide application and demand in various fields. This is due to the fact that the system provides results or output within a predefined time constraint. Depending on this time constraint the systems can be further classified as Hard real time systems, that can never miss its deadline and of it misses the deadline the result would be catastrophic. Soft real time systems, that can miss the deadline occasionally, with an acceptable probability. Missing of such deadlines would not result in catastrophes.

The ability of these systems to meet the deadline depends highly on the the scheduler that schedules the tasks and resources for its execution in an optimal way. Scheduler is responsible for the timely execution of tasks, management of resource for the execution, deciding the feasibility of the task. For the operation of this scheduler there are are various scheduling algorithms that fulfills the objective depending on the requirement. There can be preemptive and non-preemptive scheduling algorithms. In preemptive algorithms the task is assigned a specific time-slot and the execution of that task is carried within in that time-frame irrespective of the task can be completed or not. If the execution is not completed within the time-frame the execution is interrupted for the next task. Whereas in non-preemptive scheduling the task has to wait for execution till the current task execution in completed.

Visualization of the scheduling process by various algorithm, can be done in the form of a graph considering each unit time and the task that is being executed at that time. Considering this idea, this paper presents a solution to simulate various scheduling algorithm and display it in an graphical way for better understanding of the algorithm.

II. State of the Art

There are several real-time scheduling simulators. Majority of these application display the scheduling process or the feasibility of the tasks for a specific algorithm and are usually executed as a program and the resulting output of the scheduling process is displayed in the console which is difficult to understand how the tasks have been executed during the process, which algorithm executes better. Graphical representation of execution trace of any algorithm provides us with better information to analyse the behaviour of the algorithm. Even considering this graphical representation, there are only few simulators that are easily accessible that are implemented using MATLAB or various other platforms and majority of which are desktop applications, thereby making them machine dependent. These simulators tend to focus on one single algorithm at a time and implemented in specific platform, there by reducing the flexibility in the system and devoid the user from comparing execution of various algorithms. Also the interface to read the inputs display the output are not intuitive. To develop an application that combines several scheduling algorithms and also provide an user friendly graphical interface, requires combination of several technologies and good programming skill.

Identifying the short comings from existing simulators and also considering the requirements that would make the simulator a complete and user friendly system along with the real time scenario such as a task might arrive at any give time, this paper focuses on development of a web based scheduling simulator. The proposed simulator focuses scheduling and execution of tasks provided by the user, using various scheduling algorithms and providing the result of execution in an intuitive way. Web application provides the flexibility of being machine independent and provides various options to develop the system and customise to tend to the requirements. One such simulator that is similar to proposed system is developed using MATLAB, that can be found in the literature [1], which focuses on the performance of various scheduling algorithm, also providing the energy consumed by the processor with a good graphical user interface. ”Architecture of this simulator is, first it loads the algorithm input that contains various information like execution time, period for each tasks. then the algorithm loader loads the implemented algorithm for execution. After which various schedulability tests are performed to analyse the schedulability of the tasks. Once this is completed the performance is evaluated in the performance evaluation module and then the result is displayed in graphical user interface” [1].

III. Methods and Materials

IV. Requirements

A. Functional Requirements

- User must be able to change the limit of the timeline for until which the algorithm schedules the tasks.

- User must be able to add, edit and delete tasks and also its parameters such as period etc.
- The simulator must schedules the tasks entered by the user using all the algorithms on submit.
- Input to the simulator can be uploaded via excel sheet as well.
- A console log must be displayed on the screen, showing the outputs of the algorithms
-

B. Nonfunctional Requirements

- The simulator must show the visualization of scheduling in bar graph and line graph by all the scheduling algorithms.
- User must be able to hide/view visualization graphs of the algorithms
- Easy to use User Interface for changing the parameters of the tasks.
- The graphs must be easily understandable
- The simulator must show all the graphs of the algorithms at once, to be able to compare the outputs of algorithms
- The simulator must allow to hide a tasks in the graph, hence a single task alone can be viewed in the graph.

V. Architecture

The model used for the Simulator is based on the Model-View-Controller architecture (MVC) [5] using Django Framework as shown in 1. A user request is handled in the following way:

- The client enters the necessary tasks and submits on the Web browser which then sends a request for a page to the controller on the server.
- The controller retrieves the data it needs from the model, i.e, the scheduling algorithms are triggered, the model takes the input that's required from the controller and performs its execution and then sends the output to the controller, in order to respond to the request.
- The controller gives the retrieved data from the model to the view.
- The view is rendered and sent back to the client for the browser to display.

In the simulator, the model comprises of the implementation of the scheduling algorithms used in the simulator in Python. The controller are the JavaScript files which interacts with the model using JSON. Views are the HTML, CSS, JS files which are used to design the entire web page consisting of graphs, tables etc. As shown in the diagram 2 All the python files are the Model, the Controller is a JavaScript file in the Static/JS folder, the templates, the CSS, other JS files and images all belong to the View.

VI. Implementation

As the simulator is a web application, a base HTML is retrieved with no data/graphs when the user first loads the page of the simulator. The application provides a table structure with Arrival Time, Execution Time, Period and Deadline as

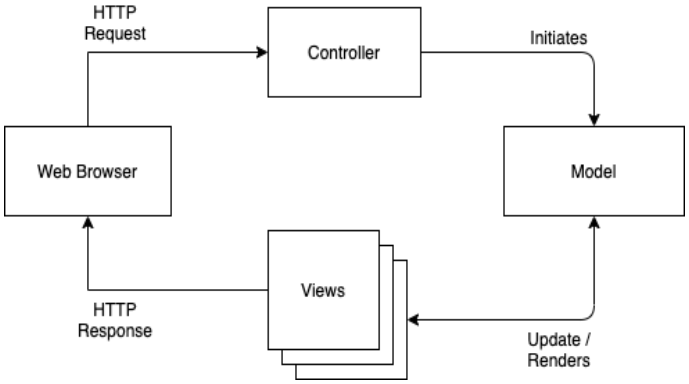

Figure 1. Architecture

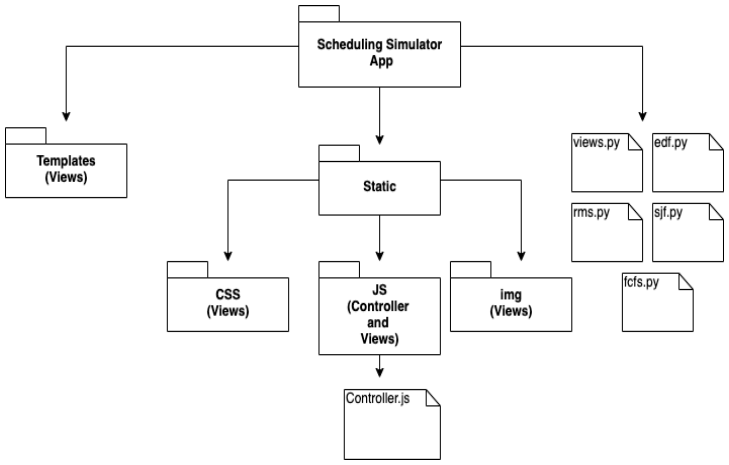

Figure 2. Folder Structure

table headers, the user can add, edit and delete as many as number of tasks. When the user uploads an excel as input, this input is converted to table data and user can view this excel data on the screen in the table. The user is now still allowed to add, edit and delete tasks. When the user enters the necessary input to the simulator directly to the table on the web page, or user uploads the excel to the simulator and then clicks on submit, on the click of submit that is when the Controller is invoked and also certain client side validations to check the input entered by the user is accurate. The controller that is the JavaScript file, converts this input which is a table structure entered by the user into a JSON in a format which the model understands, and this JSON is sent to the server - Django for processing, Django initially converts this input JSON to Python Objects, and then the models are triggered by passing each model the same input received, now the execution of all the scheduling algorithm takes place, and these models each returns an python object. The model comprises of implementation of all the scheduling algorithms of the simulator, each model returns its own output based on the implementation of the algorithm. Django converts these Python objects to JSON and then sends to the Views. Now the view is rendered using various HTML, CSS and JS

files. Firstly, the JSON is converted to JavaScript objects, and then graphs are generated using JavaScript based on the output given by the model, and then the console is printed, And then finally, this view - web page with CSS and other visualizations is send back to the browser to display.

VII. First Come First Serve

The first come first serve algorithm is one of the most simple scheduling algorithms, as the implementation of this algorithm is not complicated, since the only parameter used for scheduling is the arrival time of the process. The algorithm assigns the priority to the process based on the arrival times, the process with the least arrival time is executed first. The algorithm is Non Preemptive, that is there are no interruptions by other process, a next process can be started only after the completion of previous process. Similar to the implementation of the queue data structure. The First Come First Serve algorithm never fails to schedule, it always completed the scheduling successfully.

A. Implementation

The implementation of the algorithm in the application is done based on the arrival, period, execution time and also the LCM(range), the application shows execution of the process for the entire range, that is If a process has a arrival time 0, execution time 5, and period 10. The application considers that this process arrives at every 10 units until the LCM is reached.

Algorithm 1 First Come First Serve

1) The parameters arrival time, execution time, period are considered as inputs to the algorithm.
2) Based on the period of the process LCM is calculated or the LCM entered on the UI is considered.
3) The process with the least execution time is executed first based on its execution time.
4) Once this process completes its execution, the next process with the least arrival time is chosen.
5) Step 3 and 4 is repeated until the algoithm completes the execution till the LCM.
6) A process can arrives again for execution based on the period.
7) Waiting time for a process k is calculated using

   waitingTime[k] = executionTime[k-1] + waitingTime[k-1]

   waitingTime[0] = 0, as the first process never waits, its executed as soon as it arrives.
8) Average Waiting Time can be calculated using,

   averageWaitingTime = totalWaitingTimeForAllProcesses / numberOfProcesses
9) The turn around time for a process k is calculated using,

   turnAroundTime[k] = executionTime[k] + waitingTime[k]
10) Total Turn Around Time is calculated using,

    averageTurnAroundTime = totalTurnAroundTimeForAllProcesses / numberOfProcess

B. Advantages

- Simplest Scheduling Algorithm.
- Implementation not complicated.
- First come First Served.

C. Disadvantages

- As the algorithm is Non Preemptive, the average waiting time is very high.
- Short processes that are at the back of the queue have to wait for the long process at the front to finish.
- No Concurrent process execution.
- FCFS is not very efficient.

VIII. Shortest Job First

Shortest job first can be preemptive and non-preemptive. Preemptive is where the execution of a task can be interrupted by another task that has least execution time than the one that is being executed. Whereas in non-preemptive algorithm the task with shortest execution time is executed till its completion. The application implements the non-preemptive version of algorithm. Following steps explains the steps involved in scheduling the tasks based on shortest job first,

Algorithm 2 Euclid's algorithm

1: **procedure** EUCLID(a, b) ▷ The g.c.d. of a and b
2: $r \leftarrow a \bmod b$ **while** $r \neq 0$ **do**

We have the answer if r is 0
3: $a \leftarrow b$
4: $b \leftarrow r$
5: $r \leftarrow a \bmod b$
6:
7: **return** b ▷ The g.c.d. is b
8: **end procedure**

A. Advantages

- Shortest jobs are executed early.
- Algorithm provides optimal average waiting time for the tasks scheduled.

B. Disadvantages

- Possibility of starvation if shortest jobs keep occurring.
- Not possible to implement this at short term CPU scheduling level.

IX. Development Environment

Following list of technologies are used along with certain external libraries for the development of this simulator. Technologies used:

- Django Framework (Python 3)
- JavaScript
- HTML
- CSS
- JSON

External Libraries Used:

- JavaScript: Chart.js [3]
- CSS: Black Dashboard [4]

This application is developed using Django Framework [2], which is a Python based open-source web framework, that follows the Model-View-Controller (MVC) architectural pattern. This provides good support for the integration of GUI using HTML, CSS and JavaScript which forms the core component of the UI. For displaying the scheduling outputs as graphs in an animated and interactive way, Charts.js is used which is an open-source community project for rendering charts, graphs using JavaScript. [3] The styling, that is color font, layout of the front-end of the application is developed with the help of Bootstrap, which is a open-source front-end framework. [4]

Following tools and softwares were incorporated to carryout the development process. Tools and Softwares used:

- IDE: PyCharm
- Version Control: GitHub
- Browser: Chrome, Safari
- Documentation: Overleaf (LaTeX)

PyCharm was the Integrated Development Environment(IDE) used for the development, which is a dedicated IDE by JetBrains for the development of Python. To maintain the collaboration within the team, GitHub was used as the version controlling tool. The development was mainly tested and executed in the Chrome browser and also in Safari.

Figure 3. Application UI

Documentation is done in Overleaf, which is an online LaTeX editor.

X. Results

Figure 3, shows the layout of the application. Input box in the screen takes the input, console box displays the output in the form of texts. That is displaying the inputs, timeline length, the process is schedualble or not all these messages are displayed. The buttons RMS, EDF, FCFS and SJF allows the user to select the algorithm whose output is displayed below. The button ”Bar-graph/Line-graph” within each of the scheduling algorithm boxes which displays the execution trace in the form of bar-graph by default lets the user switch between bar-graph or line-graph. The navigation arrow at the end allows the user to move along the timeline in increments of 25 on either direction. The user can provide the input through the table provided or an excel file with task details in the same order as in table can uploaded using the ”Choose file” option. This will fill out the table from the excel using the processData() method from controller.js. The user can delete, add or edit the input if needed.

XI. Conclusion And Future work

References

[1] A. S. Pillai and T. B. Isha, ”A new real time simulator for task scheduling,” 2012 IEEE International Conference on Computational Intelligence and Computing Research, Coimbatore, 2012, pp. 1-4, doi: 10.1109/ICCIC.2012.6510232.

[2] django https://www.djangoproject.com/ Accessed: 22.08.2020

[3] Chart.js https://www.chartjs.org/ Accessed: 22.08.2020

[4] Black Dashboard https://www.creative-tim.com/product/black-dashboard Accessed: 22.08.2020

[5] Dragos-Paul Pop, Adam Altar, ”Designing an MVC Model for Rapid Web Application Development” Procedia Engineering, Volume 69, 2014, Pages 1172-1179, ISSN 1877-7058.