Design and Implementation of a Scheduling Algorithms Visualizer using JavaScript

Kyohei NISHIYAMA* Koji KAGAWA** and Yoshiro IMAI**
*Graduate School of Engineering, Kagawa University
2217-20 Hayashi-cho, Takamatsu, Kagawa 761-0396, Japan
**Faculty of Engineering, Kagawa University
2217-20 Hayashi-cho, Takamatsu, Kagawa 761-0396, Japan
kagawa@eng.kagawa-u.ac.jp

ABSTRACT

In this paper, we describe design and implementation of a scheduling algorithms visualizer that is implemented in JavaScript. We aim at a system which is handier for teachers to improve and easier for learners to grasp by taking advantages of JavaScript. We will explain the design of the proposed system and explain its internal structure precisely. Then, we discuss advantages and future directions of JavaScript-based simulation and visualization programs.

KEYWORDS

Web, JavaScript, Algorithm Visualization, Smart Device, CPU Scheduling Algorithm

1 INTRODUCTION

Along with the spreading of smart devices such as smartphones and tablets, they are more and more used in various situations of classes such as preparation, online quizzes and review. Online teaching materials displayed in smart devices can employ not only texts and still images but also animations and even interactive games. In traditional computers, Java applets are popularly used as a visualization platform. Java applet are, however, not available on iOS or Android OS which are two popular operating systems on smart devices. Even on traditional platforms such as Windows and Linux, Java applets have become harder to use because of security issues. On the other hand, the expressive power of the combination of HTML and JavaScript is getting higher and higher. JavaScript is available on both iOS and Android. HTML5 introduces some new attributes for form tags to make adaptation to smart devices easier. Moreover, it is handier to improve user interface such as internationalization and color arrangement according to users' feedback. Section 2 will explain Java applet and JavaScript in more details.

In this paper, we will introduce a visualizer for CPU scheduling algorithms implemented in JavaScript. Though we already had an Adobe Flash-based visualizer for CPU scheduling algorithms [1], we decided to implement a new system using JavaScript considering the future prospect. We aim at a system which is handier for teachers to improve and easier for learners to grasp by taking advantages of JavaScript. The system lets learners input some parameters, and visualizes the results of three scheduling algorithms, i.e. First Come First Served (FCFS), Shortest Processing Time First (SPTF), and Round Robin (RR), using color-coded graphs on HTML5 canvas. We will explain these algorithms in Section 3. They are the three basic algorithms introduced in the class "Operating Systems."

To evaluate the achievement of the design objective of the proposed system, we carried out a questionnaire for learners. The result of the questionnaire has been reported in our previous paper [2]. So this time, we decide to explain the internal behavior and module structures of our

system more precisely in Section 4. Related papers and researches are introduced in Section 5 for future perspective. Finally, Section 6 describes our summarized conclusion.

2 JAVA APPLET VS JAVASCRIPT

Until recently, Java applets were popular as a platform of visualization, especially, Web-based visualization systems. However, they are not an option to be used on tablets and smartphones as we explained in Section 1. Adobe Flash, which has been another popular platform for visualization, has a similar problem.

Even on traditional OSs such as Windows and Linux, since JDK7u51, code signing becomes mandatory for Java applets. This means that JAR files that includes Java applets need to be signed with a trusted certificate that is obtained from some certificate authority (CA). Alternatively, we can ask each user of Java applets to register the Web sites that distribute Java applets in the exception site lists in his or her browser. Anyway, it becomes more difficult for teachers to use Java applets as teaching materials.

On the other hand, JavaScript has become widely used recently. There are some reasons for the popularity. First, HTML5 and its related technologies have increased the expressive power of JavaScript. Second, utility libraries such as jQuery reduce the problem of browser dependencies. And finally, performance of hardware has advanced steadily. JavaScript is adopted in many Web-based visualization systems (e.g. OpenDSA [3]). One of the advantages of using the combination of HTML and JavaScript is that it is very easy to improve systems gradually. Moreover, it would be important to tune up user interface when the system is used in small display devices of smartphones and tablets.

3 DESIGN OF OUR PROPOSED SYSTEM

The proposed scheduling algorithms visualizer is implemented in HTML and JavaScript. It only consists of client-side programs and therefore does not employ server-side programs in order to realize an efficient load-balanced educational tool.

3.1 Overview

As we explained in the previous section, JavaScript has plenty of useful utility libraries. The system uses jQuery as a utility for event handling and DOM manipulation, HTML5 Canvas API for graphics and intro.js (http://introjs.com/) for displaying tutorials. The usage of intro.js is quite easy. After including appropriate JavaScript and CSS files, we only have to introduce two attributes data-intro and data-step to insert tutorial contents into appropriate HTML tags. The data-intro attribute represents the tutorial text to be displayed and the data-step attribute is used to designate the order in which the tutorial texts are presented.

The system also uses some new form attribute adopted since HTML5 in addition to JavaScript. Figure 1 show appearance of the system.

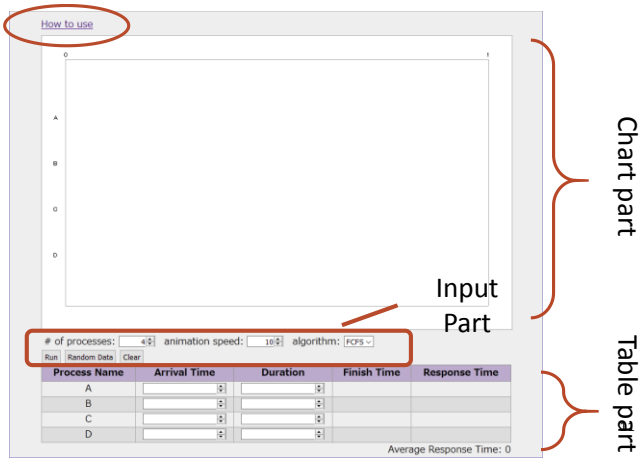

Figure 1. Appearance of the system.

For example, it uses an attribute type= ”number” for input tags, which usually attaches a spinner to a text field. Moreover, we can specify max, min and step attributes for a tag

ISBN: 978-1-941968-40-6 ©2016 SDIWC

with type=”number”. Then, we can limit the range covered by the spinner. We are aware that on iOS (version 9.2, safari version 9.0 Mobile), Android (version 4.4.2, Chrome 30 Mobile) and Edge (version 25.10596.0.0) on Windows 10, spinners are not displayed for the tag input type="number". Then, users must input numbers using screen keyboard, which is rather awkward. Such differences among platforms will hopefully be decreased gradually.
The user interface of the system consists of the chart part which displays the output of algorithms as a timeline, the input part to let users select parameters such as the number of processes and the kind of algorithm, and the table part which allows users to input parameters such as the arrival time and the duration of processes, as is shown in Figure 1.

3.2 Usage

We will explain how to use the system in the following. It should be noted that when a user clicks the “how to use” button in the page, a tutorial equivalent to the following explanation would be displayed using intro.js. This allows teachers to omit detailed explanation of the usage of the system in the lecture. Figure 2 shows a scene when intro.js displays a tutorial.

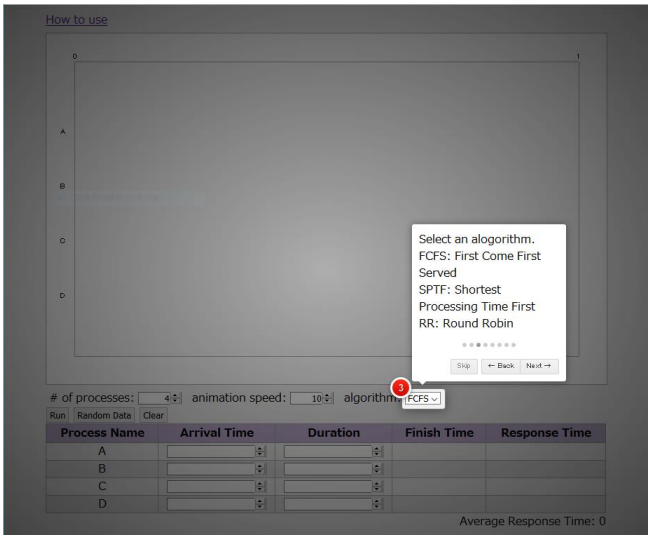

Figure 2. Tutorial displayed by intro.js.

Conventionally, almost all the Japanese educational tools are designed for Japanese user to manipulate, so the menu and messages are written in Japanese for user convenience. Though the original language used in the system is Japanese, it is modified to use English for this paper. It is one of the advantages of JavaScript that such localization and internationalization is easy.

- First, the user determines the number of processes. The initial value is four. By changing this value, the number of rows of the table part changes accordingly so that the user can input the parameters of each process.
- Then, she specifies the animation speed as necessary.
- She selects the scheduling algorithm to be simulated among FCFS, SPTF and RR.
- She inputs the parameters (the arriving time and the processing time) of each process in the table part. By clicking the “random values” button, she can use parameters that are randomly generated. This feature reduces the labor of learners.
- When she finishes filling all the parameters, she can click the “run” button.

Then, the function corresponding to the selected scheduling algorithm is invoked and the animated representation of the algorithm is displayed as a timeline in the chart part. This part is implemented using the HTML5 Canvas API and the setTimeout function. Moreover, the completion time and the response time of each process are displayed in the table part, as well as the average response time on the right bottom corner of the table. By applying different algorithms for the same parameter, learners can understand the difference among algorithms more clearly.

3.3 Overview of JavaScript Functions

The JavaScript program of the proposed system consists of the major four components—the

ISBN: 978-1-941968-40-6 ©2016 SDIWC

event handling component, the UI generation component, the data initialization component and the scheduling algorithms execution component. The event handling component deals with events such as a click of buttons and a change of input fields. The UI generation component generates the table tag used to input the process parameters. The data initialization component initializes the data used in scheduling algorithms and the scales in the chart part. The scheduling algorithms component can perform a regarding simulation of CPU scheduling according to the specified scheduling algorithm.

3.4 Event Handling Component

This component monitors the operations selected and carried out by the user. According to the kind of events such as a click of a button and a change of the number of processes, it determines the next action such as UI generation, data initialization and invocation of a scheduling algorithm. It is designed to use jQuery for handling events.

3.5 UI Generation Component

Since the table for inputting process parameters changes its number of rows according to the number of processes given by the user, it is dynamically and automatically generated by JavaScript (the createTable function). The system also employs jQuery to manipulate DOM, which eases dynamic generation of UI elements.

3.6 Data Initialization Component

The data initialization component initializes the counters used in the scheduling algorithm and draws the scales of the chart part (the axisInit function). The scales change automatically according to the finish time of the last process for the parameters input by the user. To obtain the finish time of the last process, it executes the CPU scheduling algorithm described shortly beforehand and calculates the schedule of the processes. When the interval of the scales gets too narrow, it automatically changes the increment of numbers on the scale from 1 to 5 then from 5 to 10 to avoid overlaps of numbers.

3.7 Scheduling Algorithms Component

When the user clicks the run button, the function corresponding to the selected scheduling algorithm is invoked and executed. All the functions corresponding to scheduling algorithms increment the counter of each process to represent elapsed time and use a global variable to simulate mutual exclusion. The initial value of the global variable is -1. Then, the process with the highest priority determined by each algorithm sets the global variable to its process ID number to indicate that it has the right to execute.

The execution in this simulation means the increment of the counter and the end of process implies that the counter becomes the duration assigned to the process. When the running process finishes or is interrupted, it releases the right to execute and the global variable is reset to -1. By iterating this process, the component simulates the scheduling algorithms.

The component supports the following three functions corresponding to algorithms:

- firstComeFirstServed(),
- shortestProcessingTimeFirst() and
- roundRobin().

The internal workings and visualizing of each function will be explained in the next section. It describes the three functions for First Come First Served algorithm, Shortest Processing Time First one and the Round Robin one, respectively.

4 VISUALIZATION OF SCHEDULING

This section demonstrates CPU scheduling to visualize how to calculate the priority of each process and how to handle interrupt processing.

4.1 Function: firstComeFirstServed()

ISBN: 978-1-941968-40-6 ©2016 SDIWC

This function simulates CPU scheduling using the First Come First Served (FCFS) algorithm. The FCFS algorithm simply starts with the process that arrived first. Once a process has been started, it goes on executing until termination. Since processes switch only at the termination of a process, they are never interrupted.
The function follows the FCFS algorithm and simply starts processes and increments the counter until termination according to the order they arrived. Figure 3 provides the result of the FCFS algorithm that can visualize how each process was executed in the way of FCFS order, namely process C arrived, then D, B and A arrived sequentially so that in the order of C, D, B and A each process was executed. (Note that the arriving order of A, B, C and D is the same as those for Figure 4 and 5. The system can show the different results for the same parameters based on the different algorithm.)

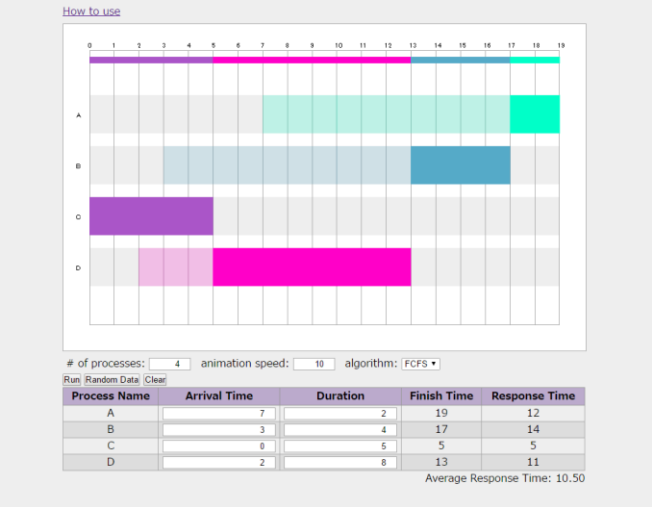

Figure 3. The result of the FCFS algorithm.

Figure 3 shows a screenshot in the end of animation the system actually produced. The servicing period is filled with vivid colors while the waiting period is filled with pale colors.

4.2 Function: shortestProcessingTimeFirst()

This function simulates the Shortest Processing Time First (SPTF) algorithm. The SPTF algorithm starts the process that has the least duration in the waiting queue. Accordingly, when a new process arrives that has a duration shorter than that of the currently running process, the scheduler interrupts the running process and switches to the new one.

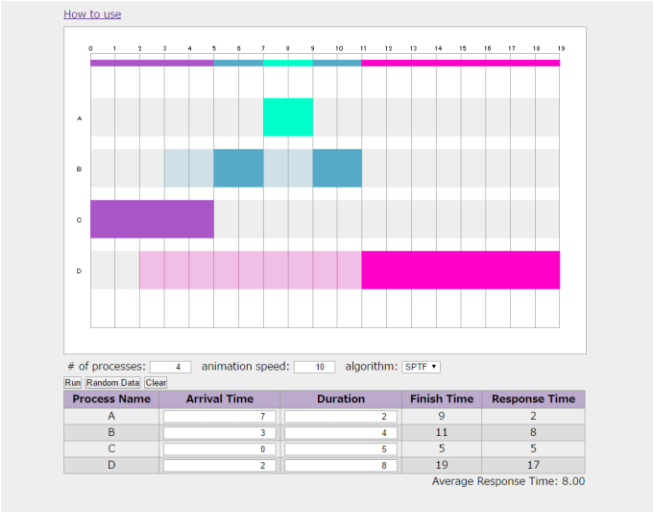

Figure 4. The result of the SPTF algorithm.

This function compares the remaining duration of each process in setting the global variable designating the current process ID as well as procedures common to all the algorithms. Even when the global variable is already set to some process ID, the algorithm checks the remaining duration of the other processes. If there exists a process with remaining duration shorter than the current process, it must release the right of execution. Figure 4 shows a screenshot in the end of animation the system actually produced for the same set of parameters as Figure 4. We can see from Figure 4 that in some cases, even when some process is already executed, a newly arrived process is given priority and is executed.

4.3 Function: roundRobin()

This function simulate CPU scheduling using the Round Robin (RR) algorithm. The RR algorithm assigns a certain amount of time to each process and executes them in circular order.

The assigned amount of time is called a time slice. The current system uses a time slice of time amount 1. That is, each time the scheduler increments the counter, the current process is switched to another process in the waiting queue. Figure 5 shows a screenshot in the end of animation the system actually produced. We can see that every arriving process is executed equally in turn.

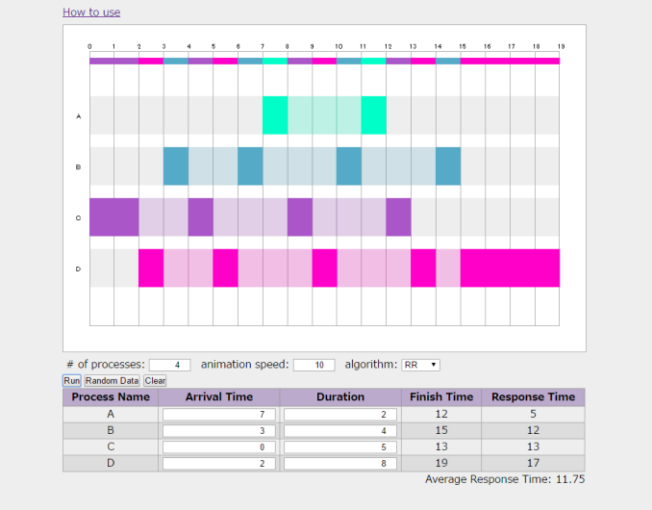

Figure 5. The result of the RR algorithm.

The function uses a variable named readyQueue to simulate the round robin algorithm, where readyQueue is a simple array. When a process arrives, its ID is inserted to the beginning of readyQueue using the "unshift" method. Then, process IDs are taken out from the beginning of readyQueue and assigned to the global variable designating the current running process. As the unit of time slice is 1 in our simulation, the process releases its right of execution after incrementing the counter by 1, when the process inserts its ID into the end of readyQueue using the push method. Thus, processes are executed in a circular order.

With the above three functions and others, our proposed scheduling algorithms visualizer has been implemented in HTML and JavaScript.

5 RELATED WORK

There are already several JavaScript-based educational visualization systems. Some of them work together with server-side programs written in Java or other languages. For example, Python Tutor [4] uses HTML, CSS and JavaScript for front end and Python for back end. JSAV [5] is "a JavaScript library for program visualization." JSAV is used in OpenDSA [3], which is an open source project to provide teaching materials for data structures and algorithms courses.

JavaScript has several weaknesses. It lacks some sort of libraries and tools equipped with other languages. It is, however, now getting easier to run program components originally written in other languages such as C and Java in JavaScript using technologies such as Emscripten (http://kripken.github.io/emscripten-site/) and DoppioJVM (http://plasma-umass.github.io/doppio-demo/). Emscripten is a set of tools to compile C and C++ programs into JavaScript using LLVM [6]. DoppioJVM is a Java Virtual Machine implemented in JavaScript [7]. For example, simulation programs in the computer science area might use helper components such as parsers and simulators in the physics area might want to use physics engines, which are easily constructed using these languages. Note that, unfortunately, DoppioJVM does not (yet) support Swing. Therefore, it is not possible to execute GUI Java programs using DoppioJVM.

When programs are getting larger, JavaScript may become helpless, since it lacks type checking. We can consider using some so called altJS (alternative JavaScript) languages. For example, DoppioJVM and the Doppio runtime system are actually written in TypeScript, and then com-piled into JavaScript. TypeScript is a language with static typing developed by Microsoft. Another workable option for constructing large scale programs would be to use server-side programs. For server-side programs, Java is still one of the leading choices, which has a large set of useful libraries and powerful frameworks such as Play and Apache Wicket.

6 CONCLUSION

In this paper, we have proposed a JavaScript-based visualization system for CPU scheduling algorithms and explained its internal workings precisely. It can be executed efficiently on both traditional PC and smart devices. Though it is designed with smart devices in mind, this time, we could not obtain enough response to evaluate comfortableness of the system when used with smart devices. Despite this, obtained response from users is encouraging. Especially, tutorials introduced by intro.js are much appreciated by first users. Its user interface is favorably accepted by users. It is also one of advantages that the user interface can be gradually improved by teachers. We expect that it can be further improved for comfortable use in smart devices in future. Encouraged by this, we would like to provide more simulation and visualization tools in several other topics of computer science in JavaScript using libraries such as jQuery, intro.js and JSAV, tools such as emscripten and DoppioJVM and altJS languages such as TypeScript with modest help of server-side programs.

Acknowledgements

This work is partially supported by JSPS KAKENHI Grant Number 15K01075.

REFERENCES

[1] K. Takeichi, and Y. Imai, “Development and Evaluation of Adobe Flash based CPU Scheduling Simulator Executable on Major Multiple Web Browsers,” In: 2015 International Conference on Intelligent Networking and Collaborative Systems, pp. 149--155. Taipei, TAIWAN, 2015.

[2] K. Nishiyama, K. Kagawa and Y. Imai, “A Scheduling Algorithms Visualizer using JavaScript,” In: Proceedings of EdMedia: World Conference on Educational Media and Technology 2016 (ED-Media 2016), pp. 1472—1478, Vancouver, CANADA, 2016.

[3] E. Fouh, V. Karavirta, D. A. Brekiron, S. Hamaouda, S. Hall, T. Haps and C. A. Shaffer, “Design and architecture of an interactive eTextbook — The OpenDSA system,” In: Science of Computer Programming, 88, pp. 22—40, 2014.

[4] P. J. Guo, “Online Python Tutor: Embeddable Web-Based Program Visualization for CS Education,” In: the 44th ACM Technical Symposium on Computer Science Education, ACM, pp. 549—584, 2013.

[5] V. Karavirta and C. A. Shaffer, “JSAV: the JavaScript algorithm visualization library,” In: Proceedings of the18th ACM conference on Innovation and technology in computer science education, pp. 159—164, 2013.

[6] A. Zakai, “Emscripten: an LLVM-to-JavaScript compiler,” In: OOPSLA '11 Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion, pp. 301—312, 2011.

[7] J. Vilk and E. D. Berger, “Doppio: breaking the browser language barrier,” In: PLDI '14 Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation, pp. 508—518, 2014.