Design and Evaluation of a Fuzzy-Based CPU Scheduling Algorithm

Shatha J. Kadhim[1] and Kasim M. Al-Aubidy[2]

[1] Computer Eng. Dept, Al-Blaqa' Applied University, Al-Salt, Jordan
[2] Computer Eng. Dept, Philadelphia University, Amman, Jordan
kma@philadelphia.edu.jo

Abstract. Scheduling in computer science means determining which tasks run when there are multiple runnable tasks. Several CPU scheduling algorithms have different features, and no single one is ideal absolutely for every application. This paper presents an attempt to apply fuzzy logic in the design and implementation of a rule-based scheduling algorithm to solve the shortcoming of well-known scheduling algorithms. Results given in this paper demonstrate that the average waiting time and the average turnaround time in the proposed algorithm are better than that obtained using priority scheduling, and closed to that obtained from shortest-job-first (SJF) scheduling. The new proposed algorithm is a dynamic scheduling algorithm which deals with both task priority and its execution time, while the SJF algorithm doesn't.

Keywords: Task Scheduling; Fuzzy Decision Making; Operating Systems; Real-Time Systems.

1 Introduction

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 [1,2]. Most real-time systems control unpredictable environments and may need operating systems that can handle unknown and changing task populations. In this case, not only a dynamic task scheduling is required, but both system hardware and software must adapt to unforeseen configurations [3].

In computer science, a scheduling algorithm is the method by which tasks, processes, threads or data flow are given access to system resources [2]. The need for a scheduling algorithm arises from the requirement for most modern systems to perform more than one process at a certain time. Operating systems, such as Windows NT, have sophisticated CPU scheduling algorithms. Windows NT-based operating systems use a multilevel feedback queue with 32 priority levels and users can select 5 of these priorities and assign them to a running application. The kernel may change the priority level of a thread depending on its I/O and CPU usage and whether it is interactive. It usually raises the priority of interactive of I/O bounded processes and lowering that of CPU bound processes, to increase the responsiveness of interactive applications. The scheduler was modified in Windows Vista to use the cycle counter register of modern processors to keep track of exactly how many CPU cycles a thread

V. V Das et al. (Eds.): BAIP 2010, CCIS 70, pp. 45 – 52, 2010.

has executed, rather than just using an interval-timer interrupt routine. Vista also uses a priority scheduler for the I/O queue so that disk defragmenters and other such programs don't interfere with foreground operations [2,4].

Operating systems may feature up to three distinct types of schedulers: a long-term scheduler, a mid-term or medium-term scheduler and a short-term scheduler. The short-term scheduler (also called dispatcher) decides which of the ready, in-memory processes are to be executed. Thus the short-term scheduler makes scheduling decisions much more frequently than the long-term or mid-term schedulers. This scheduler can be preemptive, implying that it is capable of forcibly removing processes from a CPU when it decides to allocate that CPU to another process, or non-preemptive, in which case the scheduler is unable to "force" processes off the CPU [2]. So far, a number of research works have been performed on CPU scheduling problems for different applications. Swin and his group [3] addressed the difficult problem of dynamic task scheduling in real-time distributed systems. An enhanced SJF scheduling algorithm was suggested by Shahzad and Afzal [5] to ensure that higher tasks are not going to starve, and each task is executed in a certain definite time.

Limited amounts of literature have addressed the application of fuzzy logic to CPU scheduling problems. There are four main approaches reported in the literature for the fuzzy scheduling problems; fuzzifying directly the classical dispatching rules [6], using fuzzy ranking [7], fuzzy dominance relation methods[8], and solving mathematical models to determine the optimal schedules by heuristic approximation methods[9] including genetic algorithms[10]. In this paper, an attempt will be made to apply fuzzy logic in the design and implementation of a modified scheduling algorithm to overcome the shortcoming of well-known scheduling algorithms.

2 Scheduling Algorithms

Scheduling means determining which tasks run when there are multiple runnable tasks. There are several competing goals that scheduling algorithms aim to fulfill. These includes; throughput, turnaround, response time, and others. Before discussing these goals and features of the proposed scheduling algorithm, three types of the well-known scheduling algorithms need to be identified.

2.1 First-Come, First-Served (FCFS)

The first-come first-service (FCFS) is non-preemptive, and its implementation can be considered as a simple first-in first-out (FIFO) queue. This scheduler runs each task until it either terminates or leaves the task due to an input/output or other resource blockage. Now consider six tasks (T_1-T_6) with run times as given in Table 1. By running these tasks in that order, the turnaround times for these tasks are 34, 56, 74, 86, 92, 96, for an average of 73. It is clear that the FCFS scheduler is usually unsatisfactory for interactive systems as it favors long tasks.

2.2 Shortest-Job-First (SJF)

The shortest-job-first (SJF) scheduler is non-preemptive, and tries to improve response for short tasks over FCFS. However, it requires explicit information about the service time requirements for each task. The task with the shortest time

Table 1. An example

Task No.	T_1	T_2	T_3	T_4	T_5	T_6
Run Time	34	22	18	12	6	4
Priority	5	3	2	6	1	4

Table 2. Fuzzy rules

		Pre-Priority (P_o)				
		VHP_o	HP_o	MP_o	LP_o	VLP_o
Execution Time (Tx)	VBT	MP_n	LP_n	VLP_n	VLP_n	VLP_n
	BT_x	MP_n	MP_n	LP_n	LP_n	VLP_n
	MT_x	HP_n	MP_n	LP_n	LP_n	LP_n
	ST_x	VHP_n	HP_n	HP_n	MP_n	MP_n
	VST_x	VHP_n	VHP_n	HP_n	HP_n	MP_n

requirement is chosen. Operating system will change tasks if the next CPU burst is less than what is left of the current one. Now, consider running the six tasks given in Table 1 using SJF algorithm. The turnaround times are 4, 10, 22, 40, 62, 96, for an average of 39.

It is clear that the SJF algorithm performance is better than that for FCFS algorithm for the given tasks[11]. One problem with SJF is that long processes wait a long time or suffer starvation. Short tasks are treated very favorably. Another problem with SJF is that there has to be some way of estimating the time requirement; which will be based on the previous performance.

2.3 Priority Scheduling (PrS)

Priority scheduling requires that each task is assigned a priority. At each scheduling event, the task queue is sorted according to priority, and the task of highest priority chosen first. Tasks of equal priority are scheduled as FCFS. Again, consider running of the same tasks given in Table 1 using priority scheduling algorithm, lower numbers represent higher priorities. The turnaround times are 6, 24, 46, 50, 84, 96, for an average of 50.333. It is clear that the main problem with priority scheduling algorithm is the blocking of low-priority tasks. To prevent this, it is required to modify the scheduling algorithm. Forms of priority scheduling are used in interactive systems, real-time systems, and in batch systems. In many real-time systems, it is necessary to have some priority mechanism.

As mentioned above, different scheduling algorithms have different features, and no single one is ideal absolutely for every situation. It is required to specify techniques for quantifying the characteristics of each algorithm in order to investigate a new scheduler to overcome the mentioned drawbacks of the available scheduling algorithms. To make a scheduler very efficient, we need to specify the quantitative metrics that may include;

- CPU Utilization: keep CPU utilization as high as possible.
- Throughput: number of tasks completed per unit time.
- Turnaround Time: mean time from submission to completion of task.
- Waiting Time: amount of time spent ready to run but not running.
- Response Time: time between submission of requests and first response to the request.

3 Fuzzy-Based Scheduling (FuzS)

A modified rule-based fuzzy scheduler that deals with both task priority and its execution time is presented in this section. A fuzzy-based decision maker (FDM) has

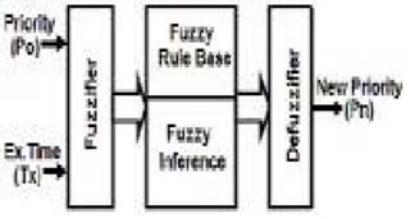


Fig. 1. Fuzzy decision maker layout

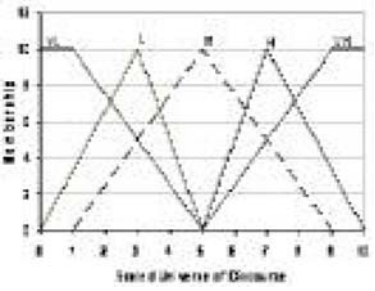


Fig. 2. Universe of discourse of the variables

been proposed to compute the new priority (Pn) of all CPU tasks according to the task pre-priority (Po) and its execution time (Tx), as shown in Fig.(1). The measured variables are inverted into suitable linguistic variables. In this application, the following linguistic variables are used for pre-priority (Po), and new calculated priority (Pn); Very Low (VL), Low (L), Medium (M), High (H), and Very High(VH). The fuzzy sets definition for execution time (Tx) are; Very Short (VS), Short (S), Medium (M), Big (B), and Very Big (VB). Figure (2) shows the universe of discourse and fuzzy sets of these variables. Fuzzy sets can have a variety of shapes. However, a triangular or a trapezoid can often provide an adequate representation of the knowledge [11].

The proposed fuzzy decision maker is a collection of linguistic rules which describe the relationships between measured variables (Po & Tx), and calculated output (Pn). Table 2 contains 25 rules, since we have five fuzzy sets for each variable. Each rule is represented by IF and THEN statement such as;

IF MPo and STx THEN HPn

This means that if the pre-priority is medium (MPo) and the execution time is short (STx), then the new calculated priority is high (HPn). The Mamdani-style inference process is used[11], and the center of gravity defuzzification method is applied to convert the fuzzy output into a crisp value that represents the new priority of a task.

It is clear that the average waiting time and average turnaround time obtained from the FuzS algorithm are better than that obtained from the PrS algorithm and close to that obtained from the SFJ algorithm.

4 Results and Performance Evaluation

To demonstrate the applicability and performance of the fuzzy-based scheduling algorithm, it is compared with two well-known scheduling algorithms; SFJ and PrS algorithms. Four case studies were considered in this comparison in terms of average waiting time (WT) and average turnaround time (TA).

Table 3. Case study 1

Task No.	T_1	T_2	T_3	T_4	T_5	T_6
Priority	10	9	1	10	5	8
Execute Time	30	1	20	3	5	10
New priority	4.151	6.413	3.562	6.541	5.500	5.090

Table 4. Tasks time specifications, case study1

Task No.	T_1	T_2	T_3	T_4	T_5	T_6	Average
W Time	0	33	49	30	44	34	31.66
TA Time	30	34	69	33	49	44	43.166

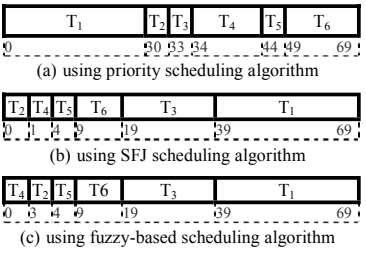
(a) using priority scheduling algorithm

Task No.	T_1	T_2	T_3	T_4	T_5	T_6	Average
W Time	39	0	19	1	4	9	12.0
TA Time	69	1	39	4	9	19	23.5

(b) using SFJ scheduling algorithm

Task No.	T_1	T_2	T_3	T_4	T_5	T_6	Average
W Time	39	3	19	0	4	9	12.333
TA Time	69	4	39	3	9	19	23.833

(c) using fuzzy-based scheduling algorithm



(a) using priority scheduling algorithm

(b) using SFJ scheduling algorithm

(c) using fuzzy-based scheduling algorithm

Fig. 3. The Gant chart of CPU time for case-study 1

4.1 Case-Study 1

Consider the running of the six tasks given in Table 3 using three scheduling algorithms; PrS, SFJ and FuzS. Figure 4 shows the Gant chart of CPU time for the given tasks.

Table 5. Case study 2

Task No.	T_1	T_2	T_3	T_4	T_5
Priority	6	5	1	4	2
Execute Time	3	24	6	9	8
New priority	5.961	4.407	4.891	5.081	4.967

Table 6. Tasks time specifications, case study 2

Task No.	T_1	T_2	T_3	T_4	T_5	Average
W Time	0	3	44	27	36	22
TA Time	3	27	50	36	44	32

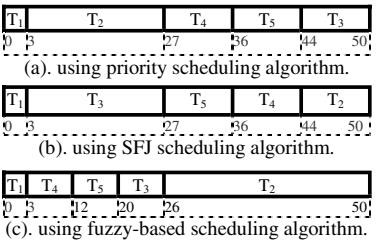
(a). using priority scheduling algorithm.

Task No.	T_1	T_2	T_3	T_4	T_5	Average
W Time	0	26	3	17	9	11.000
TA Time	3	50	9	26	17	21.000

(b). using SFJ scheduling algorithm.

Task No.	T_1	T_2	T_3	T_4	T_5	Average
W Time	0	26	20	3	12	12.200
TA Time	3	50	26	12	20	22.200

(c). using fuzzy-based scheduling algorithm.



(a). using priority scheduling algorithm.

(b). using SFJ scheduling algorithm.

(c). using fuzzy-based scheduling algorithm.

Fig. 4. The Gant chart of CPU time, case study 2

Table 7. Case study 3

Task No.	T_1	T_2	T_3	T_4
Priority	0	5	8	10
Execute Time	20	8	2	9
New priority	2.70	4.96	6.50	5.72

4.2 Case-Study 2

Now consider the running of tasks given in Table 5 using the same scheduling algorithms; PrS, SFJ and FuzS. The Gant chart of the CPU time for the given tasks is shown in Fig. 4.

4.3 Case-Study 3

In this case only four tasks, given in Table 7, are considered to test the ability of the FuzS algorithm compared with PrS and SFJ algorithms. The Gant chart of the CPU time for the given tasks is shown in Fig. 5.

Table 8. Tasks time specifications, case study 3

Task No.	T_1	T_2	T_3	T_4	Average
W Time	19	11	9	0	9.750
TA Time	39	19	11	9	19.500

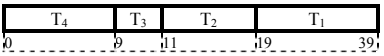
(a) using priority scheduling algorithm

Task No.	T_1	T_2	T_3	T_4	Average
W Time	19	2	0	10	7.750
TA Time	39	10	2	19	17.500

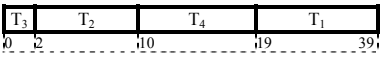
(b) using SFJ scheduling algorithm

Task No.	T_1	T_2	T_3	T_4	Average
W Time	19	11	0	2	8.000
TA Time	39	19	2	11	17.750

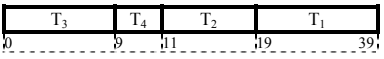
(c) using fuzzy-based scheduling algorithm



(a) using priority scheduling algorithm



(b) using SFJ scheduling algorithm



c) using fuzzy-based scheduling algorithm

Fig. 5. The Gant chart of CPU time, case study 3

4.4 Case-Study 4

Again four tasks are considered in this case but with different priorities and execution times, as in Table 9. The Gant chart of the CPU time for the given tasks is shown in Fig. 6.

Table 9. Case study 4

Task No.	T_1	T_2	T_3	T_4
Priority	7	1	15	4
Execute Time	10	5	4	8
New priority	3.90	4.27	5.80	2.95

Table 10. Tasks time specifications, case study 4

Task No.	T_1	T_2	T_3	T_4	Average
W Time	4	22	0	14	10.000
TA Time	14	27	4	22	16.750

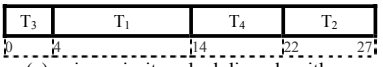
(a) using priority scheduling algorithm

Task No.	T_1	T_2	T_3	T_4	Average
W Time	17	4	0	9	7.500
TA Time	27	9	4	17	14.250

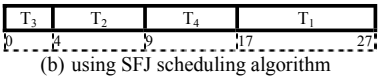
(b) using SFJ scheduling algorithm

Task No.	T_1	T_2	T_3	T_4	Average
W Time	19	4	0	9	8.000
TA Time	27	9	4	19	14.750

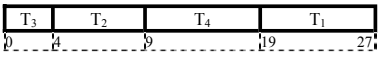
(c) using fuzzy-based scheduling algorithm



(a) using priority scheduling algorithm



(b) using SFJ scheduling algorithm



(c) using fuzzy-based scheduling algorithm

Fig. 6. The Gant chart of CPU time, case study 4

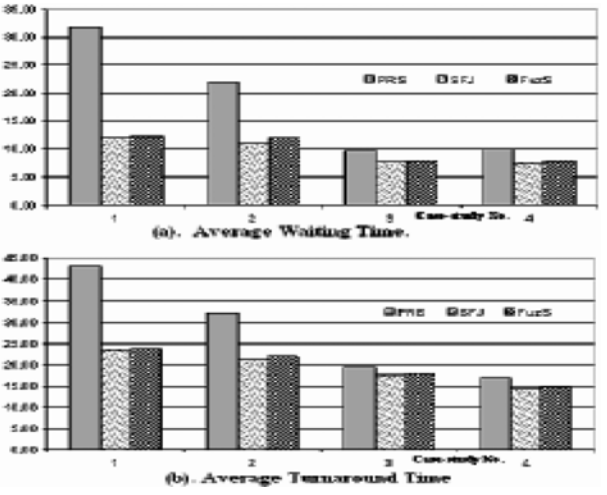


Fig. 7. Comparisons between SJF, PrS and FuzS

5 Conclusion

This paper addresses different problems of task scheduling in computer-based systems. In some applications, SJF scheduling algorithm is more suitable than PrS algorithm since it provides less waiting time and less turnaround time. In real-time applications, PrS algorithm must be used to deal with different priorities, since each task has a priority order.

In order to obtain an efficient scheduling algorithm, a rule-based fuszzy decision maker deals with both task priority and its execution time is designed and evaluated to overcome the shortcoming of well-known scheduling algorithms. As illustrated in Fig. 7, a simple comparison between fuzzy-based scheduling algorithms with other mentioned scheduling algorithms (PrS and SFJ), the following points can be pointed out:

- Several CPU scheduling algorithms have different characteristics, and no single one is ideal for absolutely every application.
- The average waiting time and the average turnaround time in the FuzS algorithm are better than that obtained using priority scheduling, and closed to that obtained from shortest-job-first scheduling.
- The FuzS algorithm is a dynamic scheduling algorithm deals with task priority, while the SJF doesn't.

References

1. Blazewicz, J., Ecker, K.H., Pesch, E., Schmidt, G., Weglarz, J.: Scheduling Computer and Manufacturing Processes. Springer, Berlin (2001)
2. Stallings, W.: Operating Systems Internals and Design Principles, 5th edn. Prentice-Hall, Englewood Cliffs (2004)
3. Swin, B.R., Tayli, M., Benmaiza, M.: Prospects for Predictable Dynamic Scheduling in RTDOS. Journal King Saud University, Computer & Information Science 9, 57–93 (1997)

4. http://en.wikipedia.org/wiki/CPU_Scheduling
5. Shahzad, B., Afzal, M.T.: Optimized Solution to Shortest Job First by Eliminating the Starvation. In: The 6th Jordanian Inr. Electrical and Electronics Eng. Conference (JIEEEC 2006), Jordan (2006)
6. Ozelkan, E.C., Duckestine, L.: Optimal Fuzzy Counterparts of Scheduling Rules. European Journal of Operational Research (113), 593–609 (1999)
7. McCahon, C.S., Lee, E.S.: Job Sequencing with Fuzzy Processing Times. Computers & Mathematics with Applications (19), 31–41 (1990)
8. Asano, M., Ohta, H.: Signal Machine Scheduling Using Dominance Relation to Minimize Earliness Subject to Ready and Due Times. International Journal of Production Economics (44), 35–43 (1996)
9. Hapke, M., Slowinski, R.: Fuzzy Priority Heuristics for Project Scheduling. Fuzzy Sets and Systems (83), 291–299 (1996)
10. Sakawa, M., Kubota, R.: Fuzzy Programming for Multi-objective Job Shop Scheduling with Fuzzy Processing and Fuzzy Due Date Through Genetic Algorithms. European Journal of Operational Research (120), 393–407 (2000)
11. Yen, J., Langari, R.: Fuzzy Logic; Intelligence, Control, and Information. Prentice Hall, Englewood Cliffs (1999)