ISSN 0974 - 9373

Vol. 22 No. 4 (2018) **Journal of International Academy of Physical Sciences** pp. 345- 354

Revised Formula for Estimating CPU-Burst

Vandana

Department of Computer Applications
SCRIET, CCSU Meerut
Email: ranavd@gmail.com

(Received October 25, 2018)

Abstract: The purpose of any scheduling algorithm is to arrange processes in ready queue in a manner to produce efficient results in terms of average waiting time, throughput and fairness etc. For this purpose, a number of scheduling algorithms are in existence. SJF is one of them. Though it provides optimal results, yet it is difficult to estimate the processing time of a process needed for its implementation. Present paper modifies average exponential formula and suggests a new computation technique based on fuzzy approximation for knowing CPU requirement of a process.
Keywords: Process, Ready-queue, SJF, Fuzzification, Defuzzification, Fuzzy Inference System(FIS).

1. Introduction

For CPU scheduling, OS uses a number of scheduling strategies like FCFS, SJF, Round Robin, Priority scheduling, which handle the removal of running process and select the next process from remaining processes. The performance of operating system is greatly depending upon the proper CPU utilization. There exists a class of algorithms that need to know the CPU requirement of a process well in advance, for arranging the processes in ready queue, in order to improve some performance criteria. Therefore, these algorithms have not been put into the practice because of not knowing the exact CPU time requirement of the process before executing it. SJF, one among them is considered to be the optimal one, in terms of producing minimum average-waiting time. For making this algorithm applicable, the exact amount of CPU-time a process will require, must be known in advance. Though a number of methods are available in literature, that may either take user's estimations or may statistically compute it from the past

observations but somehow or other they all are based upon the blind estimates. Pourali et al.[1] suggested a fuzzy based technique for predicting next CPU burst time. Pandey et al.[2] has used fuzzy estimation technique in his work. Danesh et al.[3] proposed a new technique with the help of the compiler for calculating the exact process execution time. Tanseem et al.[4] have proposed a technique to calculate remaining time of a process. The application of fuzzy technique have been done by Vandana et al.[5] in her research work for developing CPU Scheduling algorithms.

In the present paper we are suggesting a new fuzzy based computation technique for predicting CPU requirement of a process. This method is the modified version of exponential average formula. An attempt has been made to use fuzzy estimation technique that does not blindly assume the estimated time of a process; rather selects it on the basis of the actual CPU requirement of first process of the ready queue by generating a fuzzy number for this process. Apart from it, the weight factor has also been assumed using fuzzy techniques.

The rest of paper is organized as follows. Section 2 discusses about SJF, its limitations, methods for waiting time and prediction of next CPU-burst. In Section 3 fuzzy inference system has been explained. Section 4 is devoted to the development of revised formula for predicting next CPU burst. Section 5 and 6 deal with results and conclusion respectively.

2. SJF and its Limitations

In SJF scheduling algorithm the processes are arranged in ready queue according to the length of their CPU bursts. CPU is then given to the process with the minimal CPU burst requirement from the ready queue. SJF is provably optimal, in that for a given set of processes and their CPU bursts requirement it produces the least average waiting time.

2.1 Mathematical formulation of Average Waiting time: The average waiting time for a process is defined by:

$$W_S = \left(W_1 + W_2 + ... + W_n\right)/n, \tag{2.1}$$

where

$$W_k = W_{k-1} + t_{k-1} \tag{2.2}$$

is the waiting time for a k^{th} process and t_i is the execution time of i^{th} process; $1 <= k, i <= n$ (actually, the execution time of the last process in the queue, t_n, does not affect any waiting time), and $W_0 = 0$. Using (2.1) and (2.2), we get

$$W_S = \left((n-1)t_1 + (n-2)t_2 + ... + (n-k)t_k + ... + t_{n-1}\right)/n \quad (2.3)$$

Now let us suppose that we have an arbitrary set of n CPU bursts $\{t_1, t_2, .., t_n\}$. The average process waiting time in such a set is given in (2.3). If we take from that set two processes, $k-j$ and k, such that $t_{k-j} > t_k, k > j$ and switch them, the new average waiting time is

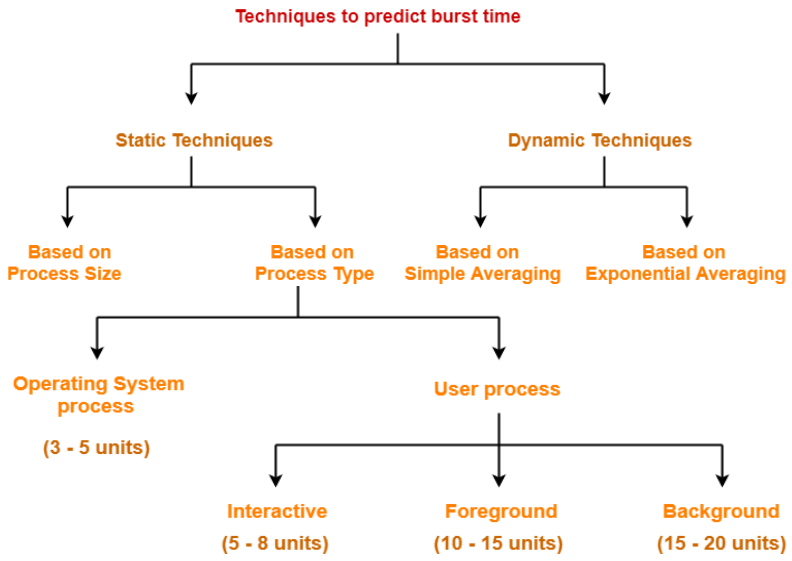
$$W_{S_1} = \left((n-1)t_1 + .. + (n-k+j)t_k + .. + (n-k)t_{k-j} + ... + t_{n_1}\right)/n \; . \quad (2.4)$$

Subtracting (2.4) from (2.3) we get

$$\begin{aligned} W_S - W_{S_1} &= \left((n-k+j)t_{k-j} + (n-k)t_k - (n-k+j)t_k - (n-k)t_{k-j}\right)/n \\ &= j\left(t_{k-j} - t_k\right)/n > 0 \; . \end{aligned} \quad (2.5)$$

Therefore $W_{S_1} < W_S$. By repeating the process over and over and putting shorter jobs in front of longer ones, we will eventually completely order the starting set and achieve the minimal average waiting time.

2.2 Methods of predicting the next CPU burst: In spite of producing the optimal average waiting time, the fundamental problem for implementing SJF is how to get the CPU requirement of a process in advance, before actually executing it. As per the on line resources[6] , the following methods are available to forecast the next CPU-burst time of a process which is a kind of time series.



2.2.1 Static Method: We can predict the CPU requirement time by two factors:

(a) Process size: This technique predicts the burst time for a process, on the basis on its size. Burst time of the already executed process of similar size is taken as the burst time for the process to be executed. The predicted burst time may not always be right.

(b) Process type: On the basis of type, can forecast CPU burst-time of process. Operating System processes like- scheduler, dispatcher, interrupt, fragmentation, etc. are faster than user process like- gaming, word processor, payroll, spread sheet etc. Burst-time for any new OS process can be predicted from any old processes of either kind. Though Static method for burst time prediction is not much reliable.

2.2.2 Dynamic Method: For dynamic method let t_i be the actual burst-time of i^{th} process, the predicted burst-time T_{n+1} for $(n+1)^{th}$ process can be calculated by using either of the following method.

(a) Simple average: Burst time for the process to be executed is taken as the average of all n processes $(P_1, P_2 \ldots P_n)$ that are executed till now.

(2.6) $$T_{n+1} = \frac{1}{n}\sum_{i=1}^{n} t_i \ .$$

(b) Exponential Average Formula (EAF): To forecast the next CPU-burst time of a process which is a kind of time series, we use average exponential formula. This formula is based on the history of past CPU-burst times of the processes which have been executed by the processor.

(2.7) $$T_{n+1} = \alpha\ t_n + (1-\ \alpha) T_n \ .$$

In general term, the formula can be written as

$$T_{n+1} = \alpha\ t_n + (1\ -\ \alpha)\alpha t_{n-1} + (1\ -\ \alpha)^2\ \alpha t_{n-2} \ldots$$

$$+ (1\ -\ \alpha)^j\ \alpha t_{n-j} \ldots + (1\ -\ \alpha)^{n+1}\ \mathrm{T}_0 ,$$

where t_n is the actual CPU burst-time, T_n is the estimation which has been taken abruptly to take initial approximation, T_{n+1} is the predicted value for next process, T_0 is a constant or overall system average. α (alpha) is some value between 0 & 1. It controls the relative weight of recent and past history in our prediction.

$$T_{n+1} = \alpha\ t_n + T_n - \ \alpha\ T_n = \alpha(t_n - \ T_n) + T_n \ .$$

Letting $(t_n - T_n)$ to be an error ε,

(2.8)
$$\begin{aligned} T_{n+1} &= \alpha\ \varepsilon_n + \ T_n \\ &= \alpha\ \varepsilon_n + \ \alpha\ \varepsilon_{n-1} + \ T_{n-1} \\ &= \alpha\ \varepsilon_n + \ \alpha\ \varepsilon_{n-1} + \ \alpha\ \varepsilon_{n-2} + T_{n-2} \\ &= \alpha\ \varepsilon t_n + \ \alpha\ \varepsilon t_{n-1} + \ \ldots\ldots + \ \alpha\ \varepsilon_0 + T_0 \end{aligned}$$

(2.9) $$T_{n+1} = \alpha \sum_{k=0}^{n} \varepsilon_k + T_0$$

Fig. 1 from Silberschatz et al.[7], shows how next CPU-burst can be predicted from the past history and current process by using Exponential Average Formula.

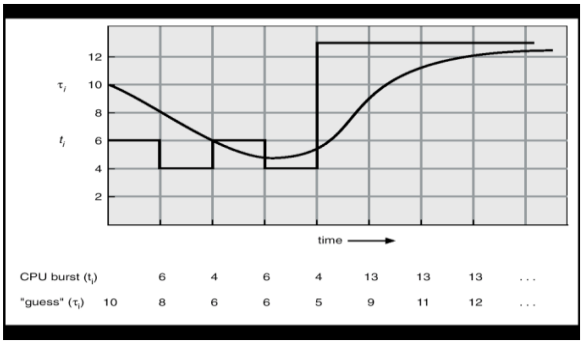


Figure 1. Example using Exponential Average Method

2.3 Limitations of EAF: Among the available methods of predicting or estimating CPU burst, EAF appears to be more appropriate, but this method has certain limitations.

(i) Initial value of T_0 has been chosen arbitrarily.

(ii) The value of α lies between 0 and 1. If we take a value closer to 1 then our prediction totally depends upon the CPU burst of recent process or if we take it closer to 0 then we may ignore current usage entirely and giving importance to the past observation. Taking $\alpha = 1/2$, recent and past history are equally weighted.

3. Fuzzy Inference System (FIS)

With incomplete, vague or imprecise knowledge the use of fuzzy systems introduced by Zadeh[8] will prove to be helpful. Fuzzy logic has been prolonged to grasp the concept of fractional truth, where the truth value may range between completely true and completely false. Now a day a recognized number of researchers are shifting towards fuzzy logic. In place of taking blind estimations, it is better to use fuzzy based systems. A fuzzy system is a rule-based system that uses fuzzy inference engine. Fuzzy inference system tries to process the given inputs and produce an output with the help of existing knowledge base. The five steps toward a fuzzy inference system, shown in Fig. 2, are as follows:

(i) Fuzzifying Inputs.
(ii) Applying Fuzzy Operators.
(iii) Applying Implication methods.
(iv) Aggregating all outputs.
(v) De-fuzzifying outputs.

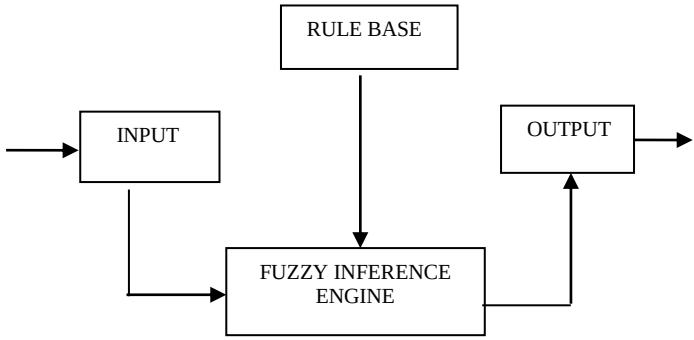


Figure 2: FIS

4. Development of Revised Average Exponential Formula

For expression (2.9), we prefer to begin with a fuzzy approximation to initial CPU-burst rather than taking blind approximation for it. The value of α, a weight-parameter is also considered to be a fuzzy number about $1/2$.

4.1 Determination of initial approximation (T_1): Let x_1 be the estimated CPU burst time of the first arrived process and *a* be any constant chosen in proportion of x_1, then

$$x = defuzz\left(\left[x_1 - 2a,\ x_1 - a,\ x_1\right] U \left[x_1 - a,\ x_1,\ x_1 + a\right] U \left[x_1,\ x_1 + a,\ x_1 + 2a\right]\right),$$

$T_1 = x$.

A triangular fuzzy number $[a,b,c]$ is given by the membership function

$$\mu_A(x) = \begin{cases} 0, & if\ x \le a \\ \dfrac{x-a}{b-a}, & if\ a < x \le b \\ \dfrac{c-x}{c-b}, & if\ b \le x < c \\ 0, & if\ x \ge c. \end{cases} \tag{4.1}$$

4.2 Determination of alpha (α): Let β is aggregation of α_1, α_2 and α_3 as shown in Fig. 3.

$$\alpha = dfuzz(\beta) \tag{4.2}$$

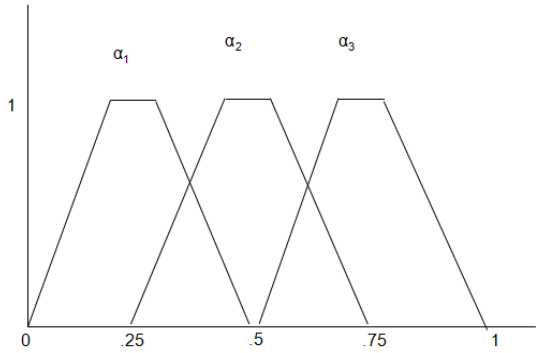


Figure 3. Determination of α

Using (4.1) and (4.2) in (2.9), if we consider the value of ε_1 to be very small, then the CPU burst for the next process will be considered as a fuzzy set about the CPU burst of first process.
Let $\tilde{A}_1$ be a fuzzy set about t_1

$$\varepsilon_1 = defuzz(A_1) - t_1,$$

$$T_2 = \alpha\,\varepsilon_1 + T_1,$$

$$T_3 = \alpha\left(\varepsilon_0 + \varepsilon_1\right) + T_1,$$

$$\ldots\ldots\ldots\ldots\ldots\ldots$$

$$\ldots\ldots\ldots\ldots\ldots\ldots$$

$$T_{n+1} = \alpha\left(\varepsilon_n + \varepsilon_{n+1} + \ldots. + \varepsilon_1\right) + T_1.$$

Using (2.8), T_1 will be

$$\alpha\varepsilon_0 + T_0 = \alpha\left[\,defuzz\left(A_0\right) - t_1\right] + \tilde{A}_0.$$

If α is a fuzzy set about $(1/2) = \tilde{\alpha}$, $\varepsilon_1 = defuzz\left(A_1\right) - t_1$, $T_1 = \tilde{A}_1$, then $T_2 = \alpha\left[\,defuzz\left(\tilde{A}_1\right) - defuzz\left(\tilde{A}_0\right) - t_0 - t_1\right] + T_1$ and hence the Revised Exponential Averaging Formula will be

(4.3) $$T_{n-1} = \alpha \sum_{k=1}^{n}\left[\,defuzz\left(A_k\right) - t_k\right] + T_1.$$

5. Results

For experimental study, we have taken randomized set of processes and calculated the results for CPU prediction. Comparison of revised exponential average formula has been made with existing EAF using MATLAB and C++. Graphical comparisons are shown in Fig 4, 5 and 6, with three different values for α as 0, 1, ½ respectively. Through the results shown in graphs, it is evident that the value of approximations calculated through revised exponential average formula (4.3) are closer to actual CPU-burst than those calculated using exponential average formula.

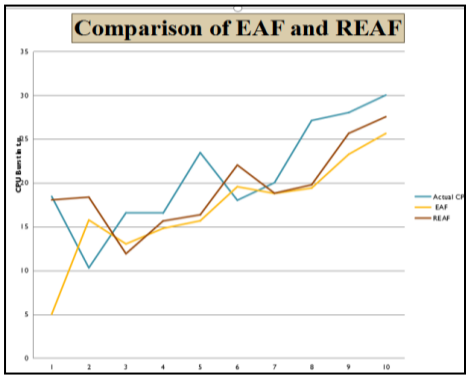


Figure 4. $\alpha = 0$

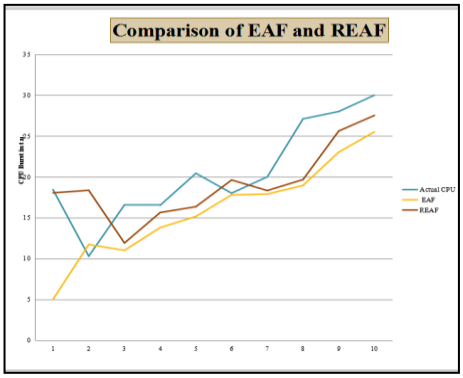


Figure 5. $\alpha = 1$

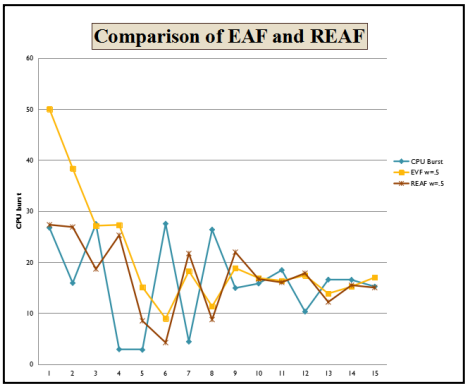


Figure 6. $\alpha = 1/2$

6. Conclusion

The aim of this paper is to revise EAF formula using fuzzy systems. The main purpose of the proposed method is to predict and estimate the execution time of processes prior to their execution. The use of fuzzy inference engine is to achieve high flexibility and desirable speed in calculation.

References

1. A. Pourali and A. M. Rahmani, A Fuzzy Based Scheduling Algorithm for Prediction of Next CPU-Burst Time to Implement Shortest Process Next, *International Association of Computer Science and Technology-Spring Conference Singapore,* (2009), 217-220.

2. D. Pandey, Vandana and M. K. Sharma, CPU Scheduling: FCFS with Shorter Processes First, *MR International Journal of Science and Technology*, **1 (1&2)** (2008), 11-17.

3. E. Danesh and A. M. Rahmani, A New Technique to Calculate the Exact Process Execution Time with the Help of the Compiler, *Journal of Applied Sciences* **7(21)** (2007), 3217-3225.

4. S. Tasneem, R. Ammar and H. Sholl, A Methodology to Compute Task Remaining Execution Time, *Proceedings. ISCC 2004. Ninth International Symposium on Computers and Communications*, Alexandria, Egypt,**1** (2004), 74-79.

5. Vandana and D. Pandey, *A Fuzzy Set Theoretic Approach To CPU Scheduling: CPU Scheduling Using Fuzzy Techniques*, Lap Lambert Academic Publishing, Germany, 2012.

6. Gate Vidhyalay for Predicting Burst time SJF Scheduling, *https://www.gatevidyalay.com*

predicting-burst-time-sjf-scheduling

7. A. Silberschatz, P. B. Galvin and G. Gagne, *Operating System Concepts*, Wiley Student Edition, John Wiley, 2004.

8. L. A. Zadeh, *Fuzzy Sets, Fuzzy Logic and Fuzzy Systems*, World Scientific, 1996.