Fuzzy Logic in Critical Section of Operating System

Hossein Kardan Moghaddam, Amir Rajaei
Department of Computer Science, University of Mysore, Mysore, India
km_farda2006@yahoo.com, amir_rajaei@hotmail.com

Abstract: *In this paper, the methodology of using fuzzy logic techniques is discussed to improve the performance and speed of computation which they needs huge calculation and comparison simultaneously. Through analysis of operating system, we have observed that some elements such as process management, storage management, file management, etc can be fuzzified. With use of this technique, we can do the optimization, through reducing the time complexity of operating system in the case of huge computation along with some amount of risk, by way of fuzzy logic .while using fuzzy logic in operating system, its task will be similar to the function of brain organism in human body. Therefore operating system will be efficient and economically under this situation.*

Keywords: *Critical Section, Operating System, Fuzzy Logic*

1. Introduction

Operating System: The operating system is the most important type of system software in a computer system. Without an operating system, a user cannot run an application program on their computer, unless the application program is self booting. For hardware functions such as input and output and memory allocation, the operating system acts as an intermediary between application programs and the computer hardware, although the application code is usually executed directly by the hardware and will frequently call the OS or be interrupted by it [3].

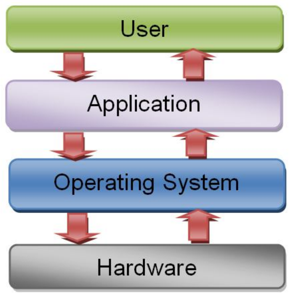

Fig.1*: Structure of operating system*

In concurrent programming a critical section is a piece of code that accesses a shared resource that must not be concurrently accessed by more than one thread of execution. A critical section will terminate in the fixed time, thread, task or process will have to wait a fixed time to enter it. A critical section is typically used when a multithreaded program must update multiple related variables without a separate thread making conflicting changes to that data. In a related situation, a critical section may be used to ensure a shared resource [3]. In operating systems we have a segment of code which will be shared by many processes. But when two processes shares the code and if one updates the code while other reading it, there occurs a conflict. This is called critical section problem [1].

Briefly we can summarize:

1- *N* processes all competing to use some shared data

2- Each process has a code segment, called *critical section*, in which the shared data is accessed.

3- Problem: ensure that when one process is executing in its critical section, no other process is allowed to execute in its critical section.

Structure of process by critical section is shown as bellow[1]:

```
Repeat
     Entry section
          Critical Section
     Exit section
          Remainder section
Until false
```

1.2 Critical Section Problems

In the case of solving the problem of Critical Section, we have two ways, Software and Hardware which discuss below.

Software Technique

1. Peterson's Algorithm: Is a concurrent programming algorithm for mutual exclusion that allows two processes to share a single-use resource without conflict, using only shared memory for communication(based on busy waiting)[5].
2. Semaphores: a semaphore is a variable or abstract data type that provides a simple but useful abstraction for controlling access by multiple processes to a common resource in a parallel programming environment [1].
3. Monitors: A monitor is an object or module intended to be used safely by more than one thread. The defining characteristic of a monitor is that its methods are executed with exclusion. This mutual exclusion greatly simplifies reasoning about the implementation of monitors compared to reasoning about parallel code that updates a data structure [7].

Hardware Technique

1. Exclusive access to memory location
2. Interrupts that can be turned off
3. Test-and-Set: special machine-level instruction
4. Swap: atomically swaps contents of two words

A proper solution to the critical section problem must meet three requirements:

1. *Mutual Exclusion*: If process *Pi* is executing in its critical section, then no other processes can be executing in their critical sections.
2. *Progress*: If no process is executing in its critical section and there exist some processes that wish to enter their critical section, then the selection of the processes that will enter the critical section next cannot be postponed indefinitely.
3. *Bounded Waiting*: A bound must exist on the number of times that other processes are allowed to enter their critical sections after a process has

made a request to enter its critical section and before that request is granted [1].

1.3 Fuzzy logic

Fuzzy logic is a form of many valued logic; it deals with reasoning that is approximate rather than fixed and exact. In contrast with traditional logic theory, where binary sets have two valued logic: true or false, fuzzy logic variables may have a truth value that ranges in degree between 0 and 1. Fuzzy logic has been extended to handle the concept of partial truth, where the truth value may range between completely true and completely false.[8]. Furthermore, when linguistic variables are used, these degrees may be managed by specific functions. If X is a collection of objects denoted generically by x, then a *fuzzy set* A in X is defined as a set of ordered pairs:

$$A = \{ (x, \mu A(x)) \mid x X\}$$

Where *µA(x)* is called the *membership function* for the fuzzy set A. The membership function maps each element of X (the *universe of discourse*) to a *membership grade* between 0 and 1. A fuzzy control system is a control system based on fuzzy logic, a mathematical system that analyzes analog input values in terms of logical variables that take on continuous values between 0 and 1, in contrast to classical or digital logic, which operates on discrete values of either 1 or 0 (true or false respectively). The first implementation of a FLC ("fuzzy logic control") was reported by Mamdani and Assilian [9]. There are specific components characteristic of a fuzzy controller to support a design procedure .In the block diagram in Fig. 2, the controller is between a preprocessing block and a post-processing block.

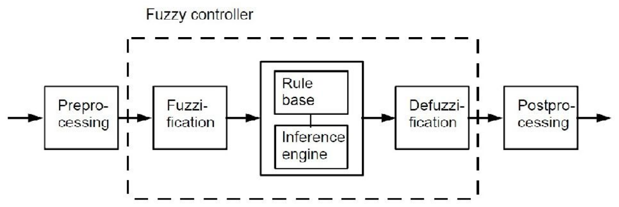

Fig 2 : *Blocks of a Fuzzy Controller*

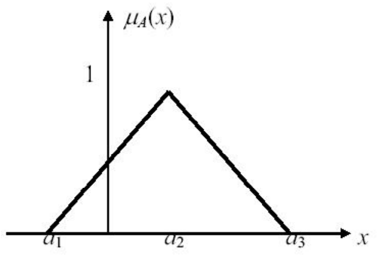

Fig 3 *: Triangular Fuzzy Number A= (a1, a2, a3)*

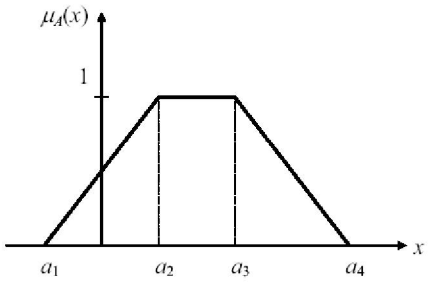

Fig 4 *: Trapezoidal Fuzzy Number A= (a1, a2, a3, a4)*

2. Proposed Approach

This method, instead of using semaphore or other said methods, is used to block a process or thread and prevent other processes or threads from entering the critical section. Calculating the error rate and possibility of error rate in allowing a process for entering the critical section of another process based on fuzzy logic and calculating and saving the error rate enables the processor to increase the processing speed and amount of the system’s simultaneous performance. It meanwhile prevents the long starvation and deadlocks and prompts the operating system’s response by permitting processes to enter the critical section of other processes. When very long calculations are required and various items interfere this method can be applied such as calculation of the longevity of the earth / sun. During the performance of an operating system, a process is allowed to enter the critical section of another process, when the calculated entrance error rate of the former is lower than that of the latter. Operating system allows the process for entering other critical section providing that the error rate of a process at the time of entering critical section is lower than the α-cuts. In case the calculated error rate is medium decision is to be made based upon the user's programming whether to allow the process to enter the critical section or do not allow. The more the distance of the low error rate and the medium error rate, the less interference is between the errors. Thus, the error's ambiguity will be decreased and the decision making will be more efficient. When there is an insignificant distance between the low and medium error a common area will be formed and this is one of the flaws of this method. Simultaneous performance of processes leads to an increase of speed processing. In order to break the deadlocks and starvations faster we can define a time constraint. However, it causes a reduction in accuracy. Calculation tends to develop error. We, during the performance of the operating system, can apply one of the different fuzzy methods such as α-cut .Performance of the operating System dealing with error, based upon fuzzy method, is in the shape of triangle or trapezium or a combination of both. When operating system's performance is in one of the said shapes we can calculate the error rate by calculating α-cut. Then we can calculate the error rate of the operating system by comparison and optimization of α-cuts. When we compare the calculated error's threshold values and operating system's error rates, the error rates of the operating system should be less. Comparing the operating system when there is interference in critical section with that of without interference, based upon fuzzy method, we can calculate error rates. We are able to compare and find the optimized α-cuts error.

3. Conclusions

Calculating the error rate and possibility of error rate in allowing a process for entering the critical section of another process based on fuzzy logic and calculating and saving the error rate enables the processor to increase the processing speed and amount of the system's simultaneous performance.

When we compare the calculated error's threshold values and operating system's error rates, the error rates of the operating system should be less. Comparing the operating system when there is interference in critical section with that of without interference, based upon fuzzy method, we can calculate error rates. We are able to compare and find the optimized α-cuts error.

Reference

1. Silberschatz , Galvin, *Operating System Concepts*.
2. Kumar, A, *What is the Critical Section in terms of Operating System.*
3. www.wikipedia.org.
4. Silberschatz, A, Galvin,B., *Operating System Concepts*. John Wiley & Sons, 5th Edition, 1999.
5. Peterson, G. L., *Myths About the Mutual Exclusion Problem*, Information Processing Letters 12(3) 1981.
6. Hofri, M., *As discussed in Operating Systems Review*, January 1990.
7. Hoare, C. A. R. Monitors: an operating system structuring concept. Comm. A.C.M. **17(10)**, 1994.
8. Novák, V., Perfilieva, I., očkoř, J., *Mathematical principles of fuzzy logic* , Kluwer Academic.
9. Mamdani, E.H., Assilian N.S., *A case study on the application of fuzzy set theory to automatic control*, *Proc. IFAC Stochastic Control Symp*, Budapest, 1974.