OpenVMS Technical Journal V12

# Intelligent Scheduling using fuzzy logic in applications

Sushruta C. Nadig

In an enterprise environment, multiple applications performing specific tasks on the production systems can cause huge performance bottlenecks, reducing system efficiency and productivity. To reduce these bottlenecks, some tasks, such as those that consume CPU, perform lot of I/O, or require time to complete, can be scheduled in intervals when the load on the system will be smaller. Job scheduling can be done based on the type of job, CPU consumption, priority, and other factors. But to do all these tasks automatically, an intelligent scheduler is required.

An automated system that can schedule activities intelligently and will provide improved performance without human intervention would be helpful for most system administrators – and fuzzy logic can do it.

## The problem with current schedulers

Every application is designed for a specific task – defragmentation tool, backup solution, database application, operating system scheduler dispatching jobs. Scheduling is mostly done based on user-specified times or by straightforward rules, such as backing up, running defragmentation every day, after certain data is written, etc.

In an operating system, jobs can be scheduled based on the FIFO basis, priority, CPU time, etc. But these application or operating system schedulers are rule-based and do not have intelligence embedded that will decide when to schedule, what to schedule, and how to schedule.

## Automating: scheduling by adding intelligence

Automated scheduling triggers jobs by monitoring certain conditions or parameters in an IT environment. Unlike the traditional method of scheduling, sometimes there could be occasions where regular jobs do not need to be executed. Typically, we would require the following things for automating scheduling:

- **Determine the inputs for automation.** While setting up the automated scheduling, the input parameters for any application will have to be determined. For example, CPU utilization and I/O activity on a system and disk can be the input parameters for triggering a defragmentation process.

- **Analyze the input and make a decision.** For example, if a disk is highly fragmented, it is ideal to trigger defrag when CPU utilization on the system is lowest and I/O activity is also low.

  In any scenario, there are many possible outcomes to consider before any decision is made. A decision table can be formed by considering all the values for inputs. This is similar to what the human brain does while deciding something: Multiple conditions will be considered and an action will be decided upon.

- **Based on the decision, trigger the action that is expected.** Generally, the result of automating schedules is the running of specific jobs. Based on the desired outcome, we can customize the decision table. For example, the system will either trigger defragmentation and stop ongoing processes, or the system will not trigger defragmentation.

  The parameters can be monitored periodically or continuously and, based on the data, a decision can be made. The inputs can be very specific or generic. For example:

  **Specific input:** If CPU load is equal to 22%, then execute Backup.

  **Generic input:** If CPU load is low and I/O is high, then trigger fewer jobs. Low, high, and fewer are all linguistic variables that can take a range of values. While the human brain can handle this generic input, a system cannot handle it directly. However, by applying fuzzy logic, inexact variables can be handled and action taken.

## Fuzzy logic, fuzzy sets, rules, and inference model

Fuzzy logic is a superset of conventional Boolean logic and extends it to deal with new aspects such as ambiguity and uncertainty. The basic elements of fuzzy logic are linguistic variables, fuzzy sets, and fuzzy rules. Fuzzy logic provides a method to assign values between 0 and 1.

The linguistic variables' values are words – specifically, adjectives such as small, little, medium, and so on. The fuzzy set generalizes the concept of a classical set, allowing its elements to have a partial membership. To assign this partial membership, separate functions will be written that are called membership functions. The membership function of a fuzzy set corresponds to the indicator function of the classical sets. For example:

The set Low ranges from 0 to ∞; any point from 0 to ∞ can become a member of the set. Membership functions determine how much each point belongs to the set. The value 0 will be the lowest of all the points in the range specified. Therefore, 0 is assigned the value of 1 and ∞ is assigned the value of 0. All the points between 0 and ∞ will be decimal values between 0 and 1.

Membership functions assign these values to the number of points we would like to have. This operation normalizes all inputs to the same range and has a direct effect on system performance and accuracy.

Fuzzy rules form the basis of fuzzy reasoning. They describe relationships among imprecise, qualitative, linguistic expressions of the system's input and output. Generally, these rules are natural language representations of human or expert knowledge and provide an easily understood knowledge representation scheme.

A typical conditional fuzzy rule assumes a form like the following:

IF CPU load is “Low” AND IO is “High” THEN trigger applications that do not do IO.

“CPU load is ‘Low’ AND IO is ‘High’” are the inputs; “trigger applications that do not do IO” is the consequence.

The inputs specified might not be completely true or false, and their degree of truth may range from 0 to 1. We compute this value by applying the membership functions of the fuzzy sets labeled “Low” and “High” to the actual value of the input variables “CPU load” and “IO”. After that, fuzzy logic is applied to the conclusion, depending on the conclusion model. Generally there are two types of models for deciding the inputs:

- Output is direct and precise; this is also called a singleton output membership function.
- Output is ambiguous and needs de-fuzzification for taking action.

## Fuzzy logic in scheduling

The following steps are used to automate scheduling using fuzzy logic. This example shows how to automate scheduling for a defragmentation tool. Fuzzy logic is used here only to determine the input sets, which will be fragmentation index and CPU load. The output is binary: defrag or stop.

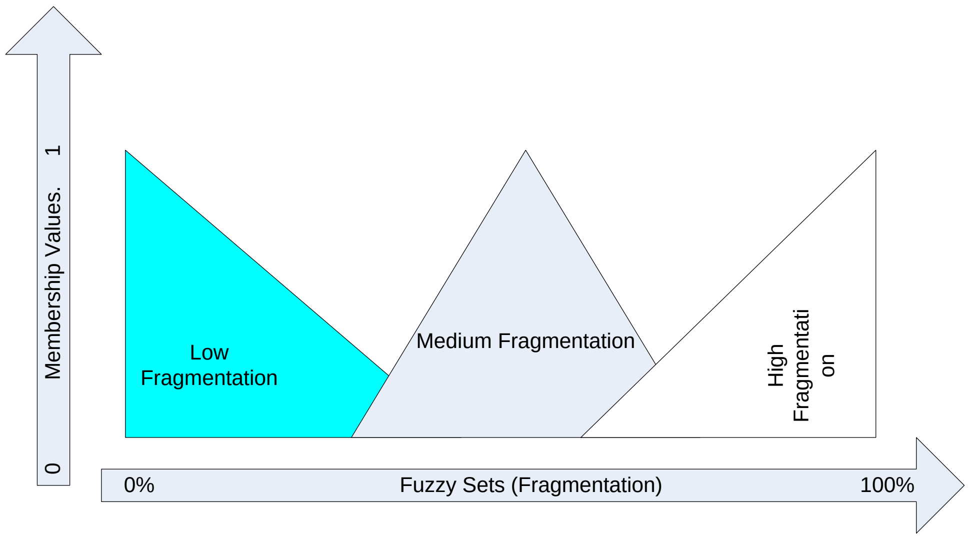
- **As mentioned above, determine the inputs for the system you are trying to automate.** Defragmentation is a very high CPU-consuming activity and can cause other applications to stall. If an automated scheduler can be associated with the defrag tool, that will take care of triggering defrag operation when the system is less occupied and when the disk is fragmented above a certain level. We can consider CPU load and fragmentation level as inputs for this system.

- **Check whether the input holds uncertainty. If yes, fuzzy logic can be applied.** While triggering defragmentation, rules such as “if CPU is Low and fragmentation is High then trigger defrag” will be used. Therefore, we need to fuzzify values for CPU load and fragmentation index.

- **Form fuzzy sets for the inputs decided.** In this scenario, Low, Medium, and High are the linguistic variables that can be associated with CPU load. The same variables can be applied to the fragmentation index. Before we proceed further, we need to decide upon the limits for the fuzzy sets we want to create.

  Assigning 0 to ∞ is an ideal way of representing the fuzzy set “Low”. But to be realistic, we will have to set the boundaries for the fuzzy sets we are creating.

**Figure 1. Typical division of fuzzy sets**



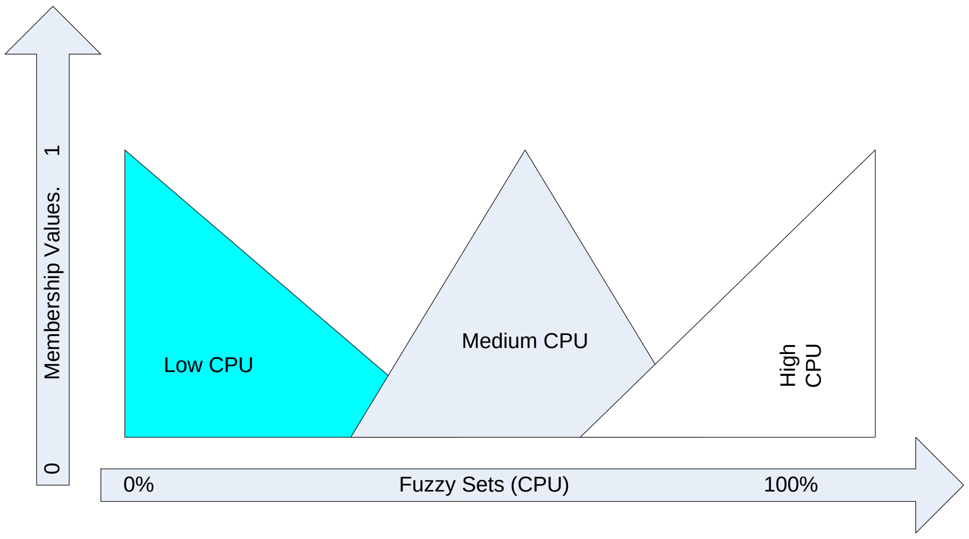


Figure 1 shows a typical division of fuzzy sets for the inputs decided in this example. The triangles in the figures indicate the ranges each set is occupying. For example, fuzzy set "Low" for the CPU ranges from 0 to 45. This range is decided based on what kind of input we are working on, to make it more realistic.

- Assign membership charcateristics for all the values the inputs can take and associate that value to the fuzzy set it belongs to.

  Consider CPU load as the input. If the CPU load on the system is 45%, map the value to the sets divided above and see where it belongs. 45% can belong to both the Low and Medium fuzzy sets. If a value corresponds to two fuzzy sets, check for how much it belongs to a particular set. A value of 45 in this example can be 2% of the set Low but 10% of the set Medium, so it can be considered an input corresponding to the set Medium. Work out this association for every input.

- Form a decision table to compare the inputs and take appropriate action according to that table. Based on the classification of inputs from the fuzzy sets, a decision would be made by considering both the inputs, as discussed above. This decision can be changed based on the outcome of the results from the tests carried out.

The idea is the decision should not be generic – it should be specific. We are eliminating the traditional way of scheduling based on the time-ordered queue. We have tried implementing this for DFO, and have found that it works well in improving performance.

| **Decision** | **Low CPU load** | **Mid CPU load** | **High CPU load** |
|---|---|---|---|
| Low fragmentation | Execute defrag | Execute defrag | Not to execute |
| Medium fragmentation | Execute defrag | Not to execute | Not to execute |
| High fragmentation | Execute defrag | Not to execute | Not to execute |

This completes the steps to automate scheduling defragmentation using fuzzy logic. In the examples above, there are more implications that would have to be handled during implementation. However, this is the basic approach to automating scheduling for applications using fuzzy logic. The same model can be used for applications such as backing up, OS scheduling, cleaning jobs, etc.

## For more information

http://paper.ijcsns.org/07_book/200603/200603A13.pdf