Model-Free Fuzzy Control of CPU Utilization for Unpredictable Workloads

Can Basaran, Mehmet H. Suzer, Kyoung-Don Kang
Department of Computer Science
State University of New York at Binghamton
{cbasaran,msuzer,kang}@cs.binghamton.edu

Xue Liu
School of Computer Science
McGill University
xueliu@cs.mcgill.ca

Abstract

In a number of real-time applications, workloads are unknown a priori but dynamically vary. Feedback control has been applied to support the desired real-time performance even in the presence of dynamic workloads. A key challenge for feedback control of software system performance is modeling, since software systems cannot be modeled via physics laws unlike mechanical or chemical systems. In this paper, we present a novel closed-loop approach based on fuzzy logic to reduce the complexity of system modeling. Especially, we aim to support the specified CPU utilization set-point even in the presence of dynamic workloads. In this way, a real-time system is desired to be neither overloaded nor underutilized. In a real-time kernel, we implement and evaluate our fuzzy logic utilization controller, the PI utilization controller (PIC) [8], and the model predictive utilization controller (MPC) [9] for an extensive set of workloads. Our fuzzy closed-loop system considerably outperforms the PIC and MPC in terms of the utilization overshoot, undershoot, and settling time.

1 Introduction

Workload may dynamically vary in a number of real-time systems. For example, the execution times of real-time tasks for target tracking or traffic control may vary significantly when the number of targets and traffic density dynamically vary. In these systems, traditional real-time scheduling techniques [4] requiring precise a priori knowledge of the workload usually defined by the real-time task periods and worst case execution times are not directly applicable to support timing constraints.

In a real-time system, it is important to control the CPU utilization to avoid overload causing deadline misses. Linear PID control techniques [13] have been applied to manage real-time performance for dynamic workloads [8, 2]. However, PID controllers and their variants may fail to support the set-point, i.e., target performance, when system dynamics deviate from a specific operating range derived offline [5]. To address the problem, model predictive control is applied to manage the utilization in dynamic environments [9, 15]. However, approximate models are used to reduce the complexity of online predictive modeling of the controlled real-time system. For example, the authors of [9, 15] assume that the actual execution times of real-time tasks are equal to their estimated execution times. Also, the predictive system model derived online may have non-trivial errors when workloads change fast [3].

In this paper, we apply formal fuzzy logic control theory [12] to support the desired utilization even when workloads vary dynamically. Notably, fuzzy control is not tied to a mathematical model of the controlled system. Due to the model-free nature of a fuzzy logic controller, there is less risk of introducing design errors due to, for example, statistical inaccuracies existing in an approximate black-box plant model [13, 10]. Rather than relying on an approximate system model, we develop novel linguistic IF-THEN rules to control the utilization based on the logical understanding of the relation between the workload and utilization.

For fair and realistic performance evaluation, we implement the fuzzy controller, PIC [8], and MPC [9] in the Real-Time Application Interface (RTAI) for Linux kernel [1]. The fuzzy logic controller shows the smallest deviation from and fastest convergence to the utilization set-point. Further, it is lightweight; it only consumes 0.53% CPU utilization and a small amount of memory to store fuzzy rules and a few control variables. Despite its virtues, fuzzy logic control theory [12]...
has rarely been applied to manage the performance of real-time systems [14, 7].

2 Problem Formulation

In this paper, we assume that there are N periodic real-time tasks in the system. Task \( \tau_i \) \((1 \leq i \leq N)\) is associated with the estimated execution time \( C_i \), period \( T_i \), and relative deadline \( D_i \). The accurate execution time is unknown. A job \( \tau_{ij} \) is the \( j \)th instance of the periodic task \( \tau_i \). We assume that every task starts at time 0. Also, we assume that an arbitrary job’s deadline is equal to the period. Tasks are scheduled via the earliest deadline first (EDF) algorithm [4], but our approach is not tied to a specific scheduling algorithm.

When the actual utilization diverges from the setpoint, the task periods are increased under overload and vice versa. The real-time system computes the period adaptation factor \( F_e(k+1) \) according to \( \Delta w(k) \):

\[
F_e(k+1) = F_e(k) \cdot (1 - K_{\Delta w} \Delta w(k))
\]  

Using \( F_e(k+1) \), task periods are adapted to adjust workloads:

\[
P_i(k+1) = P_i(k) \cdot F_e(k+1)
\]  

where \( P_i(k+1) \) is the period of an arbitrary task \( \tau_i \) in the task set during the \( (k + 1) \)th sampling period.

As a result, the adapted workload in the \( (k + 1) \)th sampling period as follows:

\[
w(k+1) = w(k) + K_{\Delta w} \Delta w(k)
\]  

A certain load that lets the system converge to the setpoint is called the convergent load \( W \). The difference between \( W \) and the current workload is formulated as:

\[
\tilde{w}(k) = W - w(k).
\]

In reality, \( W \) is unknown and may vary in time. The key objective of fuzzy control is to continuously adapt the workload based on \( e(k) \) and \( \Delta e(k) \), if necessary, to support \( U_s \) by minimizing \( |\tilde{w}(k)| \).

3 Fuzzy Logic Control

In this section, a detailed discussion of our rule-base design is given.

3.1 Fuzzy Logic Control Procedure

The universe of discourse is the domain of an input (output) to (from) the FLC [12]. Figure 2 shows the universe of discourse for the utilization error, change
with the certainty or \( \Delta w \) of the linguistic control signal in error, and control output. From Eq 1 and Eq 2, the universe of discourse for \( e(k) \) and \( \Delta e(k) \) is \([-1, 1]\). The universe of discourse for the control output is \([-0.75, 0.75]\).

Linguistic variables describe the input/output variables in fuzzy control. For instance, two inputs to the fuzzy controller at time \( kSP \) are error (i.e., fuzzified \( e(k) \)) and change in error (i.e., fuzzified \( \Delta e(k) \)). Also, the output from the FLC is called control signal—the required workload adjustment expressed linguistically.

Linguistic variables are associated with linguistic values to describe characteristics of the variables. Figure 2 shows linguistic values for the linguistic variables error, change in error, and workload control signal used in this paper.

A set of IF premise THEN consequent linguistic rules are used to map the inputs to output(s) of a FLC. For example, if error = NL and change in error = NM at the \( k \)th sampling point, i.e., time \( kSP \), then the system is overloaded and the degree of overload is increasing considerably according to Eq 1 and Eq 2. Thus, the corresponding rule in Table 1 generates a NL signal that dictates the real-time system to significantly reduce the load to support \( U_c \). The design of the rule-base in Table 1 is discussed in Section 3.2.

A membership function (MF) in Figure 2 quantifies the certainty an \( e(k) \), \( \Delta e(k) \), or \( \Delta w(k) \) value to be associated with a specific linguistic value. Specifically, the horizontal axis of Figure 2 represents \( e(k) \), \( \Delta e(k) \), or \( \Delta w(k) \) while the vertical axis indicates the membership value. For MFs (except for the leftmost or rightmost ones), we use symmetric triangles of an equal base and 50% overlap with adjacent MFs, similar to [11, 12]. Unlike traditional set theory, in fuzzy set theory underlying fuzzy control theory, set membership is not binary but continuous to deal with uncertainties [6, 12]. Thus, a fuzzy input or output may belong to up to two adjacent sets in our MFs with different certainty values. For example, \( e(k) = -0.25 \) belongs to NS in Figure 2 with the certainty \( \mu_{NS}(-0.25) = 1 \). If \( \Delta e(k) = 0.0625 \), \( \mu_{ZE}(0.0625) = 0.75 \) and \( \mu_{PS}(0.0625) = 0.25 \). Note that the control signal \( \Delta w(k) \in [-0.75, 0.75] \), because \( \Delta w(k) \) is equal to -0.75 or 0.75 when the linguistic control signal is NL or PL in Figure 2 with certainty 1.

Based on the fuzzified \( e(k) \) and \( \Delta e(k) \), the inference mechanism in Figure 1 determines which rules to apply at the \( k \)th sampling point. Thus, in the previous example, the IF-THEN rules, rule(NS, ZE) = NS and rule(NS, PS) = ZE, in Table 1 apply. To compute the certainty value of the premise in the corresponding IF premise THEN consequent rule(s), we take the minimum between the certainty values of \( e(k) \) and \( \Delta e(k) \), because the consequent cannot be more certain than the premise [12]. Thus, \( \mu(\text{NS, ZE}) = \min\{1, 0.75\} = 0.75 \) and \( \mu(\text{NS, PS}) = \min\{1, 0.25\} = 0.25 \). Also, note that maximum four rules apply at a sampling point, since the error or change in error can belong to up to two MFs in Figure 2. Thus, the worst case time complexity of our fuzzy logic control is \( O(1) \). Also, storing the rule-base in Table 1 consumes a small amount of memory.

Finally, the control signal is computed via defuzzification. Let \( i \) and \( j \) (\( 1 \leq i, j \leq 7 \)) represent the row and column indexes in Table 1 corresponding to \( e(k) \) and \( \Delta e(k) \), respectively. For a rule(i, j) in Table 1, let \( \mu(i, j) \) denote the membership function and \( c(i, j) \) denote the center of the consequent’s MF. For triangle MFs, the center is the middle of the triangle’s base. The fuzzy utilization control output is computed using all the relevant rules for specific \( e(k) \) and \( \Delta e(k) \) [12]:

\[
\Delta w(k) = \frac{\sum_{i,j} c(i, j) \cdot \mu(i, j)}{\sum_{i,j} \mu(i, j)}
\] (7)

In Figure 2, the center of NS and ZE is -0.25 and 0.0, respectively. Thus, in the previous example, \( \Delta w(k) = ((-0.25) \cdot 0.75 + (0.0) \cdot 0.25)/(0.75 + 0.25) = -0.1875 \).

### 3.2 Fuzzy Rule-Base Design

<table>
<thead>
<tr>
<th>( e/\Delta e )</th>
<th>NL</th>
<th>NM</th>
<th>NS</th>
<th>ZE</th>
<th>PS</th>
<th>PM</th>
<th>PL</th>
</tr>
</thead>
<tbody>
<tr>
<td>NL</td>
<td>NL</td>
<td>NL</td>
<td>NL</td>
<td>NL</td>
<td>NL</td>
<td>NS</td>
<td>ZE</td>
</tr>
<tr>
<td>NM</td>
<td>NL</td>
<td>NL</td>
<td>NL</td>
<td>NL</td>
<td>NM</td>
<td>NS</td>
<td>ZE</td>
</tr>
<tr>
<td>NS</td>
<td>NL</td>
<td>NL</td>
<td>NL</td>
<td>NM</td>
<td>NS</td>
<td>ZE</td>
<td>PS</td>
</tr>
<tr>
<td>ZE</td>
<td>NL</td>
<td>NM</td>
<td>NS</td>
<td>ZE</td>
<td>PS</td>
<td>PM</td>
<td>PL</td>
</tr>
<tr>
<td>PS</td>
<td>NM</td>
<td>NS</td>
<td>ZE</td>
<td>PS</td>
<td>PM</td>
<td>PL</td>
<td>PL</td>
</tr>
<tr>
<td>PM</td>
<td>ZE</td>
<td>PS</td>
<td>PM</td>
<td>PL</td>
<td>PL</td>
<td>PL</td>
<td>PL</td>
</tr>
</tbody>
</table>

**Table 1. Fuzzy Rule-Base**

![Figure 3. Fuzzy Utilization Control Characteristics](image-url)
As shown in Figure 3, there are five zones that characterize dynamic real-time system’s behaviors from which we derive the rule-base for utilization control in Table 1.

**Zone 1.** $e(k) \geq 0$ and $\Delta e(k) \leq 0$: In this zone, the actual utilization is smaller than the set point, but it comes closer to the set point. In this zone, we need to consider the following three cases.

- If $|e(k)| > |\Delta e(k)|$; that is, the magnitude of the error is bigger than the magnitude of the change-in-error. In this case, $\bar{w}(k) \geq 0$ in Eq. 6 and the current load is equal to $W$. For example, if $e(k) \in \{PM, PL\}$ and $\Delta e(k) \in \{NS\}$, then the current load is lower than $W$ and the utilization is increasing too slowly. Thus, the controller should apply a positive signal to further increase the load. As a result, $\Delta w(k) > 0$.

- If $|e(k)| = |\Delta e(k)|$, then $\bar{w}(k) = 0$ in Eq. 6. For example, if $e(k) \in \{PS\}$ and $\Delta e(k) \in \{NS\}$, then the current load is equal to $W$ ($\bar{w}(k) = 0$). Thus, $\Delta w(k) = 0$.

- If $|e(k)| < |\Delta e(k)|$, then $\bar{w}(k) < 0$. For example, if $e(k) \in PS$ and $\Delta e(k) \in \{NM, NL\}$, then the current load is higher than $W$. In this case, the utilization increases too fast. Thus, the controller applies a negative signal, $\Delta w(k) < 0$, to avoid an overshoot.

**Zone 2.** $e(k) < 0$ and $\Delta e(k) \leq 0$: In this zone, the utilization is higher than the set-point and it is further increasing. It indicates that the current load is higher than $W$; that is, $\bar{w}(k) < 0$. Hence, the controller applies $\Delta w(k) < 0$ to reverse the current trend.

**Zone 3.** $e(k) \leq 0$ and $\Delta e(k) > 0$: In this zone, the utilization is higher than the set-point, but it comes closer to the set point. In this zone, the following three cases should be considered.

- If $|e(k)| > |\Delta e(k)|$, then $\bar{w}(k) < 0$. For example, if $e(k) \in \{NM, NL\}$ and $\Delta e(k) \in \{PS\}$ then the current load is higher than $W$; that is, $\bar{w}(k) < 0$. As the utilization is decreasing too slowly, the controller should apply a negative signal to further reduce the load.

- If $|e(k)| = |\Delta e(k)|$, then $\bar{w}(k) = 0$. For example, if $e(k) \in \{NS\}$ and $\Delta e(k) \in \{PS\}$, then the current load is equal to $W$. Thus, $\Delta w(k) = 0$.

- If $|e(k)| < |\Delta e(k)|$, then $\bar{w}(k) > 0$. For example, if $e(k) \in \{NS\}$ and $\Delta e(k) \in \{PM, PL\}$, then the current load is lower than $W$. The utilization is decreasing too fast in this case. Thus, the controller should apply a positive signal to increase the load to support $U_s$ (i.e., $\Delta w(k) > 0$).

**Zone 4.** $e(k) > 0$ and $\Delta e(k) \geq 0$: In this zone, the actual utilization is lower than the set-point and it is further decreasing. It indicates that the current workload is lower than $W$ (i.e., $\bar{w}(k) > 0$). Thus, $\Delta w(k) > 0$.

**Zone 5.** $|e(k)| \leq \epsilon$ and $|\Delta e(k)| \leq \epsilon$ where $\epsilon$ is a small predefined real number: In this case, the real-time system is in the steady state. $\Delta w(k) = 0$, as the current workload is equal to $W$ (i.e., $\bar{w}(k) = 0$).

To summarize, the relationship between the control output and inputs in Table 1 can be formulated in linguistic terms:

$$\Delta w(k) = e(k) + \Delta e(k)$$

(8)

The linguistic value of $\bar{w}(k)$ can be determined from these five zones. Our fuzzy logic rule-base containing the five zones implies the following linguistic equation:

$$\bar{w}(k) = \Delta w(k)$$

(9)

which can be validated by inspecting the rule base and explanation of the fuzzy control actions in the five zones. In our rule-base, the sign of $\Delta w(k)$ is equal to the sign of $\bar{w}(k)$. This is because, in each zone, the sign of $\Delta w(k)$ is determined based on the sign of $\bar{w}(k)$. Also, the control signal’s magnitude is proportional to the difference between $W$ and the current load. Based on Eq 8 and Eq 9, the stability can be proved via the Lyapunov direct method [12, 3].

## 4 Performance Evaluation

**Experimental Settings.** We have implemented the FLC, MPC, and PIC in the RTAI 3.6 [1]. The Linux kernel version 2.6.22 is installed on a 2.3GHz Pentium 4 machine with 1 gigabytes of RAM. We have implemented and tuned the PIC as described in [8]. Also, we have implemented the MPC with the prediction horizon size of 2 and control horizon size of 1 as described in [9]. In this paper, SP is set to 1s and $U_s$ is set to 0.7 for all the tested controllers as shown in Table 2.

Each job is associated with an actual execution time: $AET_{ij} = \alpha \cdot EET_{ij}$ where $EET_{ij}$ is the estimated execution time of job $\tau_{ij}$ in the system and $\alpha$ is the execution time factor, similar to [8, 9]. In this way, fair performance comparisons are possible among the
PIC, MPC, and FLC. Note that the scheduler and controllers are unaware of actual execution times. When $\alpha > 1$, they may underestimate execution times and miss deadlines and vice versa. Thus, we evaluate how closely the FLC, MPC, and PIC can support $U_s$ when $\alpha$ varies.

We have created several different experimental profiles summarized in Table 2. Our approach outperformed the other approaches for all the tested profiles. Due to space limitations, in this paper, we mainly discuss the results for the pulse load. The pulse load tests the robustness of the controller given a sudden load increase and decrease in a step manner. There are five variations of the pulse load. Each of them starts with $\alpha = 1$ and an initial load of 60%. At 100s, $\alpha$ is increased to 2, 3, 4, and 5 for Pulse-2, Pulse-3, Pulse-4, and Pulse-5, respectively. Further, $\alpha$ is decreased to 0.3 at 200s. In this paper, the average of 10 runs is reported where each run executes a random task set for 300s.

**Experiment Results.** Figure 4 shows the results for the Pulse-5 load that tests the robustness of the controllers against abrupt changes in task execution times. Since $\alpha$ value suddenly jumps from 1 to 5 at 100s, all the tested approaches show utilization overshoots. We omit the results for the other $\alpha$ values due to space limitations.) As a result, the utilization saturates at 1 at 100s in Figure 4.

However, as shown in the figure, the FLC’s settling time is about 10s, which is approximately half the settling time of the MPC. Moreover, the FLC’s settling time is approximately five times shorter than the PIC’s settling time of 55s. Furthermore, the FLC achieves the smallest set-point tracking error:

$$E_{agg} = \sqrt{\frac{1}{n} \sum_{k=1}^{n} (U_s - u(k))^2}$$

(10)

where $n$ is the number of the sampling points in one experimental run. Specifically, $E_{agg} = 0.0611, 0.0714, and 0.1227$ for the FLC, MPC, and PIC, respectively. Thus, the FLC reduces $E_{agg}$ by half compared to the PIC with less complex controller design than the MPC.

For the ramp profile, $\alpha$ is continuously increased from 0.3 to 5 in each 300s run. The FLC enhances $E_{agg}$ by an order of magnitude compared to the PIC and MPC. For the sawtooth profile, which concatenates several ramp loads with repeatedly increasing and decreasing $\alpha$ values, the performance gap between FLC and PIC and MPC is more pronounced.

In summary, the FLC achieves the most robust control performance based on the logical understanding of the system behavior requiring no mathematical modeling of the underlying controlled system, which is tied to an operating range and subject to modeling errors due to simplified approximations or online/offline statistical modeling errors.

**Table 3. Control Overhead Comparisons**

<table>
<thead>
<tr>
<th>Controller</th>
<th>CPU Utilization</th>
<th>Code Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC</td>
<td>0.25%</td>
<td>3 lines</td>
</tr>
<tr>
<td>FLC</td>
<td>0.33%</td>
<td>100 lines</td>
</tr>
<tr>
<td>MPC</td>
<td>0.95%</td>
<td>600 lines</td>
</tr>
</tbody>
</table>

Finally, Table 3 shows the overhead of the tested controllers. All the controllers are lightweight and consume less than 1% CPU utilization for the sampling period of 1s. The PIC has the lowest overhead while the MPC has the highest overhead. The FLC consumes approximately 0.5% CPU utilization and a small amount of memory.

## 5 Conclusion

To closely support the specified utilization set-point for dynamic workloads with timing constraints, we develop a novel fuzzy closed-loop system. Our work is model-free unlike well known approaches based on PI [8] and model predictive control [9]. Hence, it is less subject to errors due to approximations/simplifications and online/offline modeling. Extensive experiments are performed in a real-time kernel to thoroughly evaluate the fuzzy, PI [8], and model predictive [9] controllers. The fuzzy logic controller shows the smallest overshoots and undershoots as well as the shortest settling time for all the tested workloads. To the best of our knowledge, no prior work has designed a fuzzy control system for real-time performance management with formal stability analysis, while comparing it to the PI and model predictive controllers. In the future, we will develop more advanced fuzzy control techniques for real-time performance management.
References


Figure 4. Pulse-5 load results